Google Hacks

By Paul Bausch, Tara Calishain, Rael Dornfest

.....

Publisher: O'Reilly
Pub Date: August 2006
Print ISBN-10: 0-596-52706-3
Print ISBN-13: 978-0-59-652706-8
Pages: 543

Table of Contents | Index

Overview

Everyone knows that Google lets you search billions of web pages. But few people realize that Google also gives you hundreds of cool ways to organize and play with information.

Since we released the last edition of this bestselling book, Google has added many new features and services to its expanding universe: Google Earth, Google Talk, Google Maps, Google Blog Search, Video Search, Music Search, Google Base, Google Reader, and Google Desktop among them. We've found ways to get these new services to do even more.

The expanded third edition of *Google Hacks* is a brand-new and infinitely more useful book for this powerful search engine. You'll not only find dozens of hacks for the new Google services, but plenty of updated tips, tricks and scripts for hacking the old ones. Now you can make a Google Earth movie, visualize your web site traffic with Google Analytics, post pictures to your blog with Picasa, or access Gmail in your favorite email client. Industrial strength and real-world tested, this new collection enables you to mine a ton of information within Google's reach. And have a lot of fun while doing it:

- Search Google over IM with a Google Talk bot
- Build a customized Google Map and add it to your own web site
- Cover your searching tracks and take back your browsing privacy
- Turn any Google query into an RSS feed that you can monitor in Google Reader or the newsreader of your choice
- Keep tabs on blogs in new, useful ways
- Turn Gmail into an external hard drive for Windows, Mac, or Linux

- Beef up your web pages with search, ads, news feeds, and more
- Program Google with the Google API and language of your choice

For those of you concerned about Google as an emerging Big Brother, this new edition also offers advice and concrete tips for protecting your privacy. Get into the world of Google and bend it to your will!

PREV

Google Hacks

By Paul Bausch, Tara Calishain, Rael Dornfest

.....

Publisher: O'Reilly
Pub Date: August 2006
Print ISBN-10: 0-596-52706-3
Print ISBN-13: 978-0-59-652706-8
Pages: 543

Table of Contents | Index

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Foreword

Working at Google means that you're exposed to technologies that are nothing short of amazing. When Google Maps was deployed for internal testing, I really couldn't believe my eyes. The sheer fun and usability of the maps were something that had transcended technology and made their way into the realm of magic. Look, I can drag the map onscreen, and I don't have to download anything...it's all in the browser! No reloading! Who know that JavaScript could do such cool stuff!

Once Maps launched, I had an idea in my head of how long it would take for our maps to appear on someone else's web page, contrary to our terms of service. I guessed it would take developers a few months to tease apart the JavaScript and do something interesting with it. Paul Rademacher delivered well before then, combining Google Maps with Craig's List to create a neat little tool for people to find places to live. Simple, clean, and smart, Paul's Housingmaps.com took the Web by storm and made the word *mash-up* a part of the modern web developer's lexicon.

When we saw what Paul had done with our fabulous little maps, we thought, "How can we make this kind of technology more reliably available to everyone?" Using an interface that isn't publicly specifie is no way to build a reliable service, for sure. As we changed things in the web site, Paul's site and others failed. We didn't want that, but we also didn't have a suitable way to properly interact with the Pauls of the world. After quite a bit of work, we came up with an API and released it at O'Reilly's Where 2.0 conference. We exposed an API that was super-friendly and allowed web developers to easily add a map to their web sites, combining it with whatever data they liked and coupled to terms that were very reasonable.

While we were certainly not the only reason for their popularity, mashups became all the rage. Maps appeared *everywhere*. Combining maps with cool data from other sites from their own user bases became de rigueur online. The combinations that people have put together are truly amazing. Want a map of crime data overlaid on a map of your neighborhood? How about satellite-tracking maps? Or maybe a map of all the places where people have spotted U2 frontman Bono?

It turns out that if you can give someone a handy interface for a useful tool, they'll take it from there This is hardly epic, amazing news, but watching it happen is always quite satisfying. People find their own uses for things, if you just get out of their way.

But this book is not just about maps. It actually details the continuum of ways you can interact with Google data and services. Even if you use only http://google.com to search and that's it, thank-you-very-much, you'll find information in this book that will make Google, and thus maybe the Internet, a bit better for you. If you want to create and distribute information using Google tools, this book will help you. If you want to make sure Google knows when you create or present new information on your site, this book will help you. If you want to find out new, innovative ways of using tools that we make available to you, then, also, this book will help you. If you want to make cool web sites like Paul's Housingmaps.com, this book will help you with that too.

Not surprisingly, these uses match up nicely with the contents of this book. The folks at O'Reilly have put together an eminently usable tome, and I hope you enjoy it as much as I do.

Chris DiBonaOpen Source Programs Manager for Google, Inc.

PREV

Credits

About the Authors

Rael Dornfest is Chief Technology Officer at O'Reilly Media. He assesses, experiments, programs, fiddles, fidgets, and writes for the O'Reilly Network and various O'Reilly publications. Rael is Series Editor of the O'Reilly Hacks series (http://hacks.oreilly.com) and has edited, contributed to, and coauthored various O'Reilly books, including *Mac OS X Panther Hacks*, *Mac OS X Hacks*, *Google Pocket Guide*, *Google: The Missing Manual*, *Essential Blogging*, and *Peer to Peer: Harnessing the Power of Disruptive Technologies*. He is also Program Chair for the O'Reilly Emerging Technology Conference (http://conferences.oreilly.com/etech). In his copious free time, Rael develops bits and bobs of freeware, particularly the Blosxom weblog application (http://www.blosxom.com), and (more often than not) maintains his Raelity Bytes blog (http://www.raelity.org).

Paul Bausch is an independent web developer living in Corvallis, Oregon. When he's not hacking together web applications, he's writing about hacking together web applications. He put together *Amazon Hacks* for O'Reilly in 2003, *Yahoo! Hacks* in 2005, and coauthored *Flickr Hacks* in 2005. Paul also helped create the popular weblog application Blogger (http://www.blogger.com), and maintains a directory of Oregon weblogs called ORblogs (http://www.orblogs.com). When he's not working on a book, Paul posts thoughts and photos to his personal blog *onfocus* (http://www.onfocus.com).

Tara Calishain is the editor of ResearchBuzz (http://www.researchbuzz.com), a weekly newsletter on Internet searching and online information resources. She's also a regular columnist for *Searcher* magazine. She's been writing about search engines and searching since 1996; her most recent book is *Web Search Garage* (Prentice Hall PTR).

Contributors

The following people contributed their hacks, writing, and inspiration to this book:

- DJ Adams (http://www.pipetree.com/qmacro) is an SAP hacker who pines for the days when he wrote job control language and S/370 assembler and got around central London on his skateboard. Currently, he is knee-deep in NetWeaver technologies and uses up spare brain cycles playing with REST, RDF, and Jabber. He wrote O'Reilly's *Programming Jabber: Extending XML Messaging* and co-wrote *Google Pocket Guide*, also from O'Reilly. He lives in Europe with Sabine and Joseph.

- Doug Adams is the webmaster of "Doug's AppleScripts for iTunes" (http://www.malcolmadams.com/itunes), a web site that offers free AppleScripts for iTunes and resources for people who write them. The site was started in late 2001 and originally offered AppleScripts for SoundJam MP, the wicked cool MP3 player for Macintosh computers that was

acquired by Apple and that eventually evolved into iTunes. Doug has been working with AppleScript since its debut during the days of System 7, but he has been programming anything that moves since buying a mail order Commodore 64 in 1983. In addition to the iTunes AppleScripts site, he maintains the "AppleScripts for Tex-Edit Plus Archives" (http://www.malcolmadams.com/te/). Doug lives in Providence, Rhode Island with his wife Natalie and daughter Ellen. When he's not AppleScripting (which, believe it or not, is most of the time), Doug is a freelance audio producer and commercial voiceover announcer.

- Tim Allwine is a Senior Software Engineer at O'Reilly Media. He develops software for the Marke Research groupvarious spidering tools that collect data from disparate sitesand is involved in the development of web services at O'Reilly.

- AvaQuest (http://www.avaquest.com) is a Massachusetts-based IT services firm that specializes in applying advanced information retrieval, categorization, and text-mining technologies to solve real-world problems. GooglePeople and GoogleMovies, created by AvaQuest consultants Nathan Treloar, Sally Kleinfeldt, and Peter Richards, came out of a web-mining consulting project the team worked on in the summer of 2002, shortly after the Google Web API was announced.

- Erik Benson (http://www.erikbenson.com).

- Justin Blanton (http://justinblanton.com) has a B.S. in computer engineering and is currently attending law school in Silicon Valley, where he is focusing on intellectual property law, and will likely practice both patent prosecution and litigation. Much of his "free time" is spent writing about various things on his web site, including Mac OS X, mobile phones and other gadgets, general tips and tricks for the Movable Type CMS, and life in general.

- CapeScience.com (http://www.capescience.com) is the development community for Cape Clear Software, a web services company. In addition to providing support for Cape Clear's products, CapeScience makes all sorts of fun web services stuff, including live services, clients for other services, utilities, and other geekware.

- Antoni Chan (http://www.alltooflat.com) is one of the founders of All Too Flat, a bastion of quirky content, pranks, and geeky humor. The Google Mirror is a 2,500-line CGI script that was developed over the period of a year starting in October 2001. When not working on his web site, he enjoys playing music, bowling, and running after a Frisbee.

- Tanya Harvey Ciampi (http://www.multilingual.ch) grew up in Buckinghamshire, England, and went on to study in Zurich, where she obtained her diploma in translation. She now lives in Ticino, the Italian-speaking region of Switzerland, where she works as an English technical translator (from Italian, German, and French) and proofreader, and teaches translation and Internet search techniques based on her WWW Search Interfaces for Translators. In her free time, she enjoys fishing with her father on the west coast of Ireland, writing poems, and playing Celtic music.

- Peter Drayton (http://www.razorsoft.net/weblog/) is a program manager in the CLR team at Microsoft. Before joining Microsoft, he was an independent consultant, trainer for DevelopMentor, and author of *C# Essentials* and *C# in a Nutshell* (O'Reilly).

- Schuyler Erle is a linguist by training and a Free Software developer by vocation. He got into GIS and digital cartography with Rich several years ago, while trying to analyze the best lines-of-sight for a rural wireless community network. He actually believes that maps and GIS, properly applied, can tell compelling stories and help improve people's lives. As of this writing, he lives with his wife near 42.375 N, 71.106 W.

- Andrew Flegg (http://www.bleb.org) works for IBM in the UK, having graduated from the University of Warwick a few years ago. He's currently the webmaster of Hursley Lab's intranet site. Most of his work (and fun) at the moment is taken up with Perl, Java, HTML, and CSS. Andrew is particularly keen on clean, reusable code, which always ends up saving time in the long run. He's written several open source projects, as well as a couple of commercial applications for RISC OS (as used in the Iyonix PC, the first desktop computer to use an Intel XScale).

- Rich Gibson believes that the world is made of stories, and has unlimited curiosity in the world. He indulges his brilliant, semi-manic children in super-long storytimes, weird science projects, and adventures of many varieties. It is only the steady support of his loving wife that permits him to organize his eccentricity into occasional coherent bursts of creative productivity. Life is very, very good.

- Andrew Goodman is Principal of Page Zero Media (http://www.page-zero.com), a Toronto-based search-marketing firm. He is the author of *Winning Results with Google AdWords* (McGraw-Hill), and recently edited Mona Elesseily's *Yahoo Search Marketing Handbook*. He has a high Quality Score, and a higher golf handicap.

- Kevin Hemenway (http://www.disobey.com), better known as Morbus Iff, is the creator of disobey.com, which bills itself as "content for the discontented." He's a publisher, developer, and writer of more home cooking than you could possibly imagine (like the popular open source syndicated reader AmphetaDesk, the best-kept gaming secret Gamegrene.com, the popular Ghost Sites and Nonsense Network, the giggle-inducing articles on the O'Reilly Network, a few pieces at Apple's Internet Developer site, etc.), and an ardent supporter of cloning merely so he can get more work done. He cooks with a Fry Pan of Intellect +2 and lives in Concord, New Hampshire.

- Jack D. Herrington is a programmer who has been developing applications since he was 13, almost 25 years ago. Over the years, he has written in every major programming language and for every environment.

- Mark Horrell (http://www.markhorrell.com) has worked in search engine optimization since 1996 when he joined Net Resources International, a publisher of industrial-engineering web sites, where he conceived and developed the company's Internet marketing strategy. He left in 2002 and is now a freelance web developer based in London, specializing in search enginefriendly design.

- Judy Hourihan (http://judy.hourihan.com).

- Leland Johnson (http://protoplasmic.org) is currently a student at Illinois Institute of Technology. He tried learning Perl in 1999, then tried again and was successful in 2001, and now uses it for everything except his classes. When he's not busy with classes, he updates his blog, explores Chicago, and plays far too many video games.

- Steven Johnson (http://www.stevenberlinjohnson.com/) is the author of two books, *Emergence* (Scribner) and *Interface Culture* (Perseus). He cocreated the sites FEED and Plastic.com, and now blogs regularly at http://www.stevenberlinjohnson.com. He writes the monthly "Emerging Technology" column for *Discover* magazine, and his work has appeared in many publications, including the *New York Times*, *Harper's*, *Wired*, and *The New Yorker*. He lives in Brooklyn, New York.

- Richard Jones (http://richard.jones.name) has spent the last four years working as a software

engineer for Agent Oriented Software (http://www.agent-software.com). AOS is responsible for a leading intelligent-agent development platform known as JACK Intelligent Agents. Before AOS he worked as a software engineer for Senate Software (a small search technology company), where he developed web page relevance heuristics. Before that, Richard was a co-founder of Earthmen Technology, which developed network intrusion detection technologies. At Earthman, he was responsible for a majority of the development, which included low-level TCP/IP networking code, Linux kernel hacking, and fast pattern-matching algorithms. He has two degrees, one in computer science and another in cognitive science, both from LaTrobe University (http://www.latrobe.edu.au). While in school, Richard majored in computer science, linguistics, and psychologyareas he retains a keen interest in. Richard is also a squash-playing Buddhist.

- Stuart Langridge (http://www.kryogenix.org) gets paid to hack on the Web during the day, and does it for free at night when he's not arguing about Buffy or Debian GNU/Linux. He's keen on web standards, Python, and strange things you can do with JavaScript, all of which can be seen at his web site and blog. He's also slightly surprised that the Google Art Creator, which was an amusing little hack done in a day, is the most popular thing he's ever written and got him into a book.

- Beau Lebens (http://www.dentedreality.com.au) is a PHP web developer who believes that even complex systems can be made simple for an end user. Originally from Perth, Western Australia, he is currently working in Hawaii. He has released a number of projects on his web site, including webpad, the web-based text editor; AvantBlog, a Palm/Pocket PC Blogging application; and the PHP Blogger API, which provides PHP developers with access to the Blogger API. Beau is a big believer in simpler, distributed technologies such as Atom, REST, and RSS for the future of the Web.

- Philipp Lenssen (http://blog.outer-court.com) was born in 1977 and currently lives in Stuttgart, Germany. He works as a developer for the web sites of a popular German car-maker. He once spent nine months living in Malaysia and prefers very spicy foods. In his spare time, Philipp is the author of Google Blogoscoped (a daily blog covering Google, online research, and Internet fun in general) and searches for new and exciting ways to tap the consciousness of the Web.

- Mark Lyon (http://marklyon.org) is the creator of the Google Gmail Loader. A former programmer for the U.S. Army Corps of Engineers, he gave up his aspirations of programming greatness after an unsuccessful interview at Google. He is now a law student at Mississippi College in Jackson and plans to practice intellectual property and technology law. In his spare time, he writes novel but mediocre software in whatever language strikes his fancy.

- Mikel Maron (http://brainoff.com/) is an independent software developer and ecologist. He has built several geographically oriented projects around the worldKit mapping package, including World as a Blog and mapufacture. Previously, he led the development of My Yahoo! in the pre-RSS days. Mikel was awarded a master's degree from the University of Sussex for building a simulation of the evolution of complexity in food webs. Originally from California, Mikel is now based mostly in Brighton, UK, with his wife Anna. Links to various things can be found at his web site.

- Paul Mutton (http://www.jibble.org) currently works for Netcraft in the UK. He graduated with first-class honors in computer science, winning the IEE Institution Prize for being the best overall student in his department. He uses Google on a daily basis and Internet Relay Chat (IRC) to collaborate with fellow Ph.D. students in other countries. In his spare time, he uses his Sun Certified Java Programmer skills to develop all sorts of open source software on his personal web site (http://www.jibble.org). Some of his research has culminated in the creation

of the popular PieSpy application (http://www.jibble.org/piespy), which infers and visualizes social networks on IRC, and even appeared on Slashdot once. He can normally be found jibbling around in #jibble and #irchacks on the freenode IRC network with the nickname Jibbler, or Paul on smaller networks.

- Mark Pilgrim (http://diveintomark.org), author of *Greasmonkey Hacks* (O'Reilly), is an accessibility architect by day. By night, he is a husband and father who lives in North Carolina with his wife, his two sons, and his dog. He spends his copious free time sunbathing, skydiving, and reading Immanuel Kant's *The Critique of Pure Reason* in the original Klingon.

- Andrew Savikas works in the O'Reilly Digital Media Group. Andrew is the author of *Word Hacks*, also published by O'Reilly. He developed and maintains the custom Word template and VBA macros used by all the O'Reilly authors who don't insist on writing in POD. Except for the ones who insist on writing in XML. Or Troff. Andrew also works with FrameMaker, FrameScript, InDesign, DocBook XML, Perl, Python, Ruby, and whatever else he finds lying around the office. He has a degree in communications from the University of Illinois at Urbana-Champaign, and lives in Boston with his wife Audrey, who loves to see her name in print.

- Chris Sells (http://www.sellsbrothers.com) is an independent consultant, speaker, and author specializing in distributed applications in .NET and COM. He's written several books and is currently working on *Windows Forms for C# and VB.NET Programmers* and *Mastering Visual Studio .NET*. In his free time, Chris hosts various conferences, directs the Genghis source-available project, plays with Rotor, and makes a pest of himself in general at Microsoft design reviews.

- Alex Shapiro (http://www.touchgraph.com) is the founder and CTO of TouchGraph LLC. Alex graduated from Columbia's computer science program in 2000 and spent his early career at a consulting company. After the stock-market bubble burst, he decided to spend time developing a network visualization product he had conceived. Through network visualization, Alex found that he could combine his interests in user interface design, graph theory, and sociology. After seeing a business demand for his technology, Alex founded TouchGraph LLC, which is slowly gathering a list of respected clients.

- Kevin Shay (http://www.staggernation.com) is a writer and web programmer who lives in Brooklyn, New York. His Google API scripts, Movable Type plug-ins, and other work can be found at the soon-to-launch staggernation.com.

- Gary Stock (http://www.googlewhack.com/stock.htm) coined the term "Google whack" while he was supposed to be doing research for UnBlinking (http://www.unblinking.com). When Gary writes for UnBlinking, he would do better to focus on his role as CTO of the news-clipping and briefing service Nexcerpt (http://www.nexcerpt.com). Gary works at Nexcerpt to get a break from stewardship of the unusual flora and fauna on the 160 acres of woods and wetland that he owns, which in turn keeps him from spending time with his wife (and Nexcerpt CEO) Julie, who he married to offset his former all-consuming career as an above-top-secret computer spy, which he had entered to avoid permanently becoming a jazz arranger and pianist. Seriously.

- Aaron Swartz (http://www.aaronsw.com) is a teenage writer, coder, and hacker. He is a co-author of the RSS 1.0 specification, a member of the W3C RDF Core Working Group, and metadata adviser to the Creative Commons. He's also the guy behind the Google Weblog (http://google.blogspace.com). He can be reached at me@aaronsw.com.

- Brett Tabke (http://www.webmasterworld.com) is the owner/operator of WebmasterWorld.com, the leading news and discussion site for web developers and search engine marketers. Brett has

been involved in computing since the late 1970s and is one of the Internet's foremost authorities on search engine optimization.

- Adam Trachtenberg (http://www.trachtenberg.com) is Manager of Technical Evangelism at eBay, where he preaches the gospel of the eBay platform to developers and businessmen around the globe. Before eBay, Adam co-founded and served as Vice President for Development at two companies, Student.Com and TVGrid.Com. At both firms, he led front- and middle-end web site design and development. Adam began using PHP in 1997 and is the author of *Upgrading to PHP 5* and co-author of *PHP Cookbook*, both published by O'Reilly. He lives in San Francisco and has a B.A. and M.B.A. from Columbia University.

- Phillip M. Torrone is a feature columnist for http://www.engadget.com and a contributing editor for *Popular Science*. Coauthor of *Flash Enabled: Design and Development for Mobile Devices* (New Riders), Phillip has also contributed to numerous books and magazines on hardware hacking, cell phones, and PDAs. Phillip's latest work and more can be found at http://www.flashenabled.com.

- Matt Webb is an engineer and designer, splitting his working life between R&D with BBC Radio & Music Interactive and freelance projects (primarily in the social-software world), most recently co-authoring *Mind Hacks* for O'Reilly. Online, he can be found at Interconnected (http://interconnected.org/home) and, in the real world, in London.

Acknowledgments

We would like to thank all those who contributed their ideas and code for Google hacks to this book. Many thanks to Nelson Minar and the rest of the Google Engineering Team, Nate Tyler, and everyone else at Google who provided ideas, suggestions, and answersnot to mention the Google Web API itself. And to Andy Lester and Justin Blanton, our technical editors along the way, goes much appreciation for their thorough nitpicking.

Rael

First and foremost, to Asha, Sam, and Miraalways my inspiration, joy, and best friends.

My extended family and friends, both local and virtual, who'd begun to wonder if they needed to send in a rescue party.

Brian Sawyer has, over the course of the last year, been my production liaison, co-editor, editor, "man Friday," and friend. Hat's off ;-) to Brian, and long may he stet.

I'd like to thank Dale Dougherty for bringing me in to work on the Hacks series; it's been a circle of wide circumference from *Google Hacks* to *Google Hacks*, Third Edition, and quite the journey of discovery. The O'Reilly editors, production, product management, and marketing staff are consummate professionals, hackers, and mensches. Extra special thanks goes out to my virtual cube mate, Nat Torkington; to Laurie Petrycki for showing me the ropes; and to Tim O'Reilly for his unfailing support and friendship.

Tara, it's been fabulous traveling this road with you, and I intend to make sure our paths keep on crossing at interesting intersections.

Karma points to Clay Shirky and Steven Johnson for egging me on to do more with the Google API than late-night fiddling. And, of course, a shout-out goes to the blogosphere population and folks in my Google neighborhood for their inspired prattling on APIs and all other things geek-worthy.

Paul

To my wife Shawnde, thanks again for the continuous feedback, frontline editing, and for cheerfully discussing Google day and night.

Many thanks go to Brian Sawyer for providing direction and encouragement, and for fine-tuning the text.

Thanks to Rael and Tara for blazing the hacks trail, and to Rael for inspiring me to get involved with the Hacks series in the first place.

And thanks to my friends both online and off for chewing on hack ideas and sending those one or two bits of info that make a hack work.

Tara

Everyone at O'Reilly has been great in helping pull this book together, but I wouldn't have gotten to participate in this book if it hadn't been for Tim Allwine, who first helped me with Perl programs a couple of years ago.

My family, especially my husband, has been great about tolerating my distraction as I sat around muttering to myself about variables and subroutines.

Even as this book was being written, I needed help understanding what Perl could and couldn't do. Kevin Hemenway was an excellent teacher, patiently explaining, providing examples, and, when all else failed, pointing and laughing at my code.

Of course, most of this book wouldn't exist without the release of Google's API. A big thanks to Google for building a playground for us thousands of search-engine junkies. And just as big a thanks to the many contributors who so generously allowed their applications to appear in this book.

Finally, a big, big, he-gets-his-own-paragraph thanks to Rael Dornfest, who is a great co-author/editor and a lot of fun to work with.

Preface

Search engines for large collections of data preceded the World Wide Web by decades. There were those massive library catalogs, hand-typed with painstaking precision on index cards and eventually, to varying degrees, automated. There were the large data collections of professional information companies such as Dialog and LexisNexis. Then there are the extant private, expensive medical, real estate, and legal search services.

Those data collections were not always easy to search, but with a little finesse and a lot of patience, i was always possible to search them thoroughly. Information was grouped according to established ontologies, the data preformatted according to particular guidelines.

Then came the Web.

Information on the Webas anyone who has ever looked at half a dozen web pages knowsis not all formatted the same way. Nor is it necessarily accurate. Nor up to date. Nor spellchecked. Nonetheless, search engines cropped up, trying to make sense of the rapidly increasing index of information online. Eventually, special syntaxes were added for searching common parts of the average web page (such as title or URL). Search engines evolved rapidly, trying to encompass all the nuances of the billions of documents online, and they continue to evolve today.

Google™ threw its hat into the ring in 1998. The second incarnation of a search engine service knowr as BackRub, the name *Google* was a play on the word *googol*: a one followed by a hundred zeros. From the beginning, Google was different from the other major search engines onlineAltaVista, Excite, HotBot, and others.

Was it the technology? Partially. The relevance of Google's search results was outstanding. But more than that, Google's focus and more human face made it stand out online.

With its friendly presentation and constantly expanding set of options, it's no surprise that Google continues to draw lots of fans. There are blogs devoted to it. Search engine newsletters, such as ResearchBuzz, spend a lot of time covering Google. Legions of devoted fans spend a lot of time uncovering undocumented features, creating games (such as *Google whacking*), and even coining new words (such as *Googling*, the practice of checking out a prospective date or hire via Google's search engine). People Google prospective employers and blind dates, goods and services, school reports and movie reviews, facts and fiction, fun and profit.

In April 2002, Google reached out to its fan base by offering the Google API. The Google API gives programmers a way to access the Google search results with automated queries. While you can do al the searching, sifting, and sorting by hand, there's nothing like getting your computer to do it for you.

At the time of this writing, Google is reaching out further with many more products that achieve its stated mission to "organize the world's information and make it universally accessible and useful." Not only is Google organizing public information on the Web with Google Search offerings, books by Google Print, and geographic information with Google Maps, but it is also organizing our personal information with products such as Google Desktop, Gmail, and the recently announced Google

Calendar.

Google has changed the way people and computers alike approach information.

Why Google Hacks?

Hacks are generally considered to be "quick-and-dirty" solutions to programming problems or interesting techniques for getting a task done. But what does this kind of hacking have to do with Google?

Considering the size of the Google index, there are times when you might want to do a particular kin of search but you get too many results for the search to be useful. Or you may want to do a search that the current Google interface does not support.

The idea of *Google Hacks* is not to give you an exhaustive manual on how every command in the Google syntax works (although we do give this more than a fair shake), but rather to show you some tricks for making the best use of a search, to show off just what's possible when you automate your queries with a little programming know-how, and to shine a light into some of the overlooked corners of Google's offerings. In other words, *hacks*.

How This Book Is Organized

The combination of Google's myriad services and over four billion pages of constantly shifting data can do strange things to your imagination and give you lots of new perspectives on how best to search. This book goes beyond the instruction page to the idea of*hacks*: tips, tricks, and techniques you can use to make your Google searching experience more fruitful, more fun, or (in a couple of cases) just more weird.

This book is divided into several chapters:

Chapter 1, Web

> This chapter describes the fundamentals of how Google's search works. You'll find tips and tricks for Google's special syntax (think "special sauce"); specialty searches such as the phonebook, calculator, package, and stock tracking; the Google cache; related links; and more Beyond a mere list of "this syntax means that," we'll take a look at how to eke out every last bit of searching power from each syntaxand how to mix and match for some truly monstrous searches.

Chapter 2, Advanced Web

> Kick your newfound search expertise into high gear, automating your trawling, crawling, and recombination by hacking Google programmatically. By letting your fingers do the walking and your eyeballs do the scanning, you'll meander farther, dig deeper, and come up with results that you never would have found otherwise.

How to Use This Book

You can read this book from cover to cover if you like, but for the most part, each hack stands on its own. So feel free to browse, flipping around to whatever sections interest you most. If you're a Perl newbie, you might want to try some of the easier hacks, and then tackle the more extensive ones as you get more confident.

How to Run the Hacks

The programmatic hacks in this book run either on the command line (that's Terminal for Mac OS X folk, DOS command window for Windows users) or as CGI scripts: dynamic pages living on your web site, accessed through your web browser.

Command-Line Scripts

Running a hack on the command line invariably involves the following steps:

1. Type the program into a garden-variety text editor: Notepad on Windows, TextEdit on Mac OS X, `vi` or `Emacs` on Unix/Linux, or anything else of the sort. Save the file as directedusually as *scriptname.pl* (*pl* stands for Perl, the predominant programming language used in *Google Hacks*).

2. Alternatively, you can download the code for all of the hacks online at http://www.oreilly.com/catalog/googlehks2, in a ZIP archive filled with individual scripts already saved as text files.

3. Get to the command line on your computer or remote server. In Mac OS X, launch the Terminal (Applications Utilities Terminal). In Windows, click the Start button, select Run..., type `command`, and hit the Enter/Return key on your keyboard. In Unix...well, we'll just assume you know how to get to the command line.

4. Navigate to where you saved the script. This varies from operating system to operating system, but usually involves something like `cd ~/Desktop` (that's your Desktop on the Mac).

5. Invoke the script by running the programming language's interpreter (e.g., Perl) and feeding it the script (e.g., *`scriptname.pl`*), like so:
6. `$ perl`

   `scriptname.pl`

7. Most often, you'll also need to pass along some parameters: your search query, the number of results you'd like, and so forth. Simply drop them in after the script name, enclosing them in

quotes if they're more than one word or if they include an odd character or three:

8. `$ perl`

   scriptname.pl '"much ado about nothing" script' 10

9. The results of your script are almost always sent straight back to the command-line window in which you're working, like so:
10. `$ perl`

    scriptname.pl '"much ado about nothing" script' 10

```
1. "Amazon.com: Books: Much Ado About Nothing: Screenplay ..."
[http://www.amazon.com/exec/obidos/tg/detail/-/0393311112?v=glance]
2. "Much Ado About Nothing Script"
[http://www.signal42.com/much_ado_about_nothing_script.asp]
...
```

The elllpsis (`...`) signifies that we've cut off the output for brevity's sake.

1. To keep output from scrolling off your screen faster than you can read it, on most systems you can "pipe" (read: redirect) the output to a little program called more:
2. `$ perl`

   scriptname.pl

   | more

3. Hit the Enter/Return key on your keyboard to scroll through line by line, and the spacebar to leap through page by page.

4. You'll also sometimes want to direct output to a file for safekeeping, import it into your spreadsheet application, or display it on your web site. This is as easy:
5. `$ perl`

   scriptname.pl

   >

   output_filename.txt

6. To pour input into your script from a file, simply do the opposite:
7. `$ perl`

scriptname.pl

<

input_filename.txt

Don't worry if you can't remember all of this; each hack has a "Running the Hack" section, and some even have a "The Results" section that shows you just how it's done.

CGI Scripts

CGI scriptsprograms that run on your web site and produce pages dynamicallyare a little more complicated if you're not used to them. While fundamentally they're the same sort of scripts as those run on the command line, they are more troublesome because setups vary so widely. You may be running your own server, your web site may be hosted on an Internet service provider's (ISP) server your content may live on a corporate intranet serveror anything in between.

Since going through every possibility is beyond the scope of this (or any) book, you should check your ISP's knowledge base, call its technical support department, or ask your local system administrator for help.

Generally, though, the methodology is the same:

1. Type the program into a garden-variety text editor: Notepad on Windows, TextEdit on Mac OS X, `vi` or `Emacs` on Unix/Linux, or anything else of the sort. Save the file as directedusually as *scriptname.cgi* (*cgi* reveals that you're dealing with a CGIthat's common gateway interfacescript).

2. Alternatively, you can download the code for all of the hacks online at http://www.oreilly.com/catalog/googlehks2, in a ZIP archive filled with individual scripts already saved as text files.

3. Move the script to wherever your web site lives. You should have some directory on a server somewhere in which all of your web pages (all those*.html* files) and images (ending in*.jpg*, *.gif*, etc.) live. Within this directory, you'll probably see something called a*cgi-bin* directory. This is where CGI scripts usually must live in order to be run rather than just displayed in your web browser when you visit them.

4. You usually need to bless CGI scripts as executableto be run rather than displayed. Just how you do this depends on your server's operating system. If you're on a Unix/Linux or Mac OS X

system, this usually entails typing the following on the command line:

5. `$ chmod 755`

scriptname.cgi

6. Now the script should run as expected when you point your web browser to it, and behave in a manner similar to that described in the "Running the Hack" section of the hack at hand.

7. Just what URL you use once again varies wildly. It should, however, look something like this: *http://www.<your_domain.com>/<cgi-bin>/<scriptname.cgi>*, where *your_domain.com* is your web site domain, *cgi-bin* is the directory in which your CGI scripts live, and *scriptname.cgi* is the script itself.

8. If you don't have a domain and are hosted at an ISP, the URL is more likely to look like this: *http://www.<your_isp.com>/<~your_username>/<cgi-bin>/<scriptname.cgi>*, where *your_isp.com* is your ISP's domain, *~your_username* is your username at the ISP, *cgi-bin* is the directory in which your CGI scripts live, and *scriptname.cgi* is the script itself.

If you come up with something called an "Internal Server Error" or see the error code 500, something's gone wrong somewhere in the process. At this point, you can take a crack at debugging (read: shaking the bugs out) yourself or ask your ISP or system administrator for help. Debuggingespecially CGI debuggingcan be a little more than the average newbie can bear, but there is help in the form of a famous Frequently Asked Question (FAQ): "The Idiot's Guide to Solving Perl CGI Problems." Google for it and step through as directed.

Using the Google API

Be sure to consult Chapter 8 for an introduction to the Google API, how to sign up for a developer's keyyou'll need one for many of the hacks in this bookand the basics of programming Google in a selection of languages to get you going.

Learning to Code

Fancy trying your hand at a spot of programming? O'Reilly's best-selling*Learning Perl* by Randal L. Schwartz and Tom Phoenix is a good start. Apply what you learn to understanding and using the hacks in this book, and perhaps even take on the "Hacking the Hack" sections to tweak and fiddle with the scripts. This is a useful way to get a little programming under your belt if you're a searching nut, since it's always a little easier to learn how to program when you have a task to accomplish and existing code to leaf through.

Where to Go for More

There's so much to Google that it's easy to miss minor tweaks and major new offerings alike. Visit Google's "More, more, more" page (http://www.google.com/options) on a regular basis; stay on top

of all things Google by reading or subscribing to the Google blogs, unofficial (http://blog.outer-court.com) and official (http://googleblog.blogspot.com); and be sure to look at the Appendix for even more ways to keep up with Google.

Ga-ga over Google? Pick up a Google-branded tchotchkegreen lava lamp, double latte mug, t-shirt, backback, or bookat the official Google Store (http://www.googlestore.com).

Conventions Used in This Book

The following is a list of the typographical conventions used in this book:

Italic

Used to indicate new terms, URLs, filenames, file extensions, directories, and program names, and to highlight comments in examples. For example, a path in the filesystem will appear as */Developer/Applications*.

`Constant width`

Used to show code examples, the contents of files, and console output, as well as the names of variables, commands, and other code excerpts.

`Constant width bold`

Used to highlight portions of code, typically new additions to old code.

`Constant width italic`

Used in code examples and tables to show sample text to be replaced with your own values.

Gray type

Used to indicate a cross-reference within the text.

You should pay special attention to notes set apart from the text with the following icons:

This is a tip, suggestion, or general note. It contains useful supplementary information about the topic at hand.

This is a warning or note of caution, often indicating that your money or your privacy might be at risk.

The thermometer icons, found next to each hack, indicate the relative complexity of the hack:

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Google Hacks*, Third Edition, by Rael Dornfest, Paul Bausch, and Tara Calishain. Copyright 2006 O'Reilly Media, Inc., 0-596-52706-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that let's you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you nee the most accurate, current information. Try it for free at http://safari.oreilly.com.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). As a reader of this book, you can help us to improve future editions by sending us your feedback. Please let us know about any errors, inaccuracies, bugs, misleading or confusing statements, and typos that you find anywhere in this book.

Please also let us know what we can do to make this book more useful to you. We take your comments seriously and will try to incorporate reasonable suggestions into future editions. You can write to us at:

O'Reilly Media, Inc.
1005 Gravenstein Hwy N.
Sebastopol, CA 95472
800-998-9938 (in the U.S. or Canada)
707-829-0515 (international/local)
707-829-0104 (fax)

To ask technical questions or to comment on the book, send email to:

bookquestions@oreilly.com

The web site for *Google Hacks*, Third Edition, lists examples, errata, and plans for future editions. You can find this page at:

http://www.oreilly.com/catalog/googlehks3

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com

Got a Hack?

To explore Hacks books online or to contribute a hack for future titles, visit:

http://hacks.oreilly.com

PREV

Chapter 1. Web

Google's front page is deceptively simple: a search form and a couple of buttons. Yet that basic interfaceso alluring in its simplicitybelies the power of the Google engine underneath and the wealth of information at its disposal. If you use Google's search syntax to its fullest, the Web is your oyster.

Searching in Google doesn't have to be a case of just entering what you're looking for in the search box and hoping for the best. Google offers you many waysvia special syntax and search optionsto refine your search criteria and help Google better understand what you're looking for. We'll dig into Google's powerful, all-but-undocumented special syntax and search options, and show how to use them to their fullest. We'll cover the basics of Google searching, wildcards, word limits, syntax for special cases, mixing syntax elements, advanced search techniques, and using specialized vocabularies, including slang and jargon.

PREV

Google Web Search Basics

Whenever you search for more than one keyword at a time, a search engine has a default strategy for handling and combining those keywords. Can those words appear individually anywhere in a page or do they have to be right next to each other? Will the engine search for both keywords or for either keyword?

Phrase Searches

Google defaults to searching for occurrences of your specified keywords anywhere in the page, whether side by side or scattered throughout. To return the results of pages containing specifically ordered words, enclose them in quotes, turning your keyword search into a *phrase search* , to use Google's terminology.

On entering a search for the keywords:

```
to be or not to be
```

Google will find matches where the keywords appear anywhere on the page. If you want Google to find you matches where the keywords appear together as a phrase, surround them with quotes, like this:

```
"to be or not to be"
```

Google will return matches in which only those words appear together (not to mention explicitly including stop words such as "to" and "or"; see the section "Explicit Inclusion" a little later).

Phrase searches are also useful when you want to find a phrase but aren't quite sure of the exact wording. This is accomplished in combination with wildcards, explained later in the chapter in "Full-Word Wildcards."

Basic Boolean

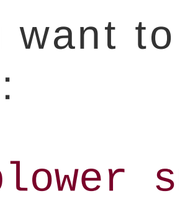
Whether an engine searches for all keywords or any of them depends on what is called its *Boolean default* . Search engines can default to Boolean `AND` (searching for all keywords) or Boolean `OR` (searching for any keywords). Of course, even if a search engine defaults to searching for all keywords, you can usually give it a special command to instruct it to search for any keyword. Lacking specific instructions, the engine falls back on its default setting.

Google's Boolean default is `AND`, which means that if you enter query words without modifiers, Google will search for all your query words. For example, if you search for:

```
snowblower Honda "Green Bay"
```

Google will search for all the words. If you prefer to specify that any one word or phrase is acceptable, put an `OR` between each:

```
snowblower OR snowmobile OR "Green Bay"
```



Make sure you capitalize `OR`; a lowercase `or` won't work correctly.

If you want to search for a particular term along with two or more other terms, group the other terms within parentheses, like so:

```
snowblower (snowmobile OR "Green Bay")
```

This query searches for the word "snowmobile" or phrase "Green Bay" along with the word "snowblower." A stand-in for `OR`, borrowed from the computer-programming realm, is the `|` (pipe) character, as in:

```
snowblower (snowmobile | "Green Bay")
```

Negation

If you want to specify that a query item must *not* appear in your results, prepend a - (minus sign or dash):

```
snowblower snowmobile -"Green Bay"
```

This will search for pages that contain both the words "snowblower" *and* "snowmobile," but *not* the phrase "Green Bay."

Note that the symbol must appear directly before the word or phrase that you don't want. If there's space between, as in the following query, it won't work as expected:

```
snowblower snowmobile - "Green Bay"
```

Be sure, however, to place a space *before* the `-` symbol.

Explicit Inclusion

On the whole, Google will search for all the keywords and phrases that you specify (with the

exception of those you've specifically negated with, of course). However, there are certain words that Google will ignore because they are considered too common to be of any use in the search. These words"I," "a," "the," and "of," to name a feware called*stop words* .

You can force Google to take a stop word into account by prepending a+ (plus) character, as in:

```
+the king
```

Stop words that appear inside of phrase searches are not ignored. Searching for:

```
"the move" glam
```

will result in a more accurate list of matches than:

```
the move glam
```

simply because Google takes the word "the" into account in the first example but ignores it in the second.

Synonyms

Every so often, you get the feeling that you're missing out on some useful results because the keyword or keywords you've chosen aren't the only way to express what you're looking for.

The Google synonym operator, the ~ (tilde) character, prepended to any number of keywords in your query, asks Google to include not only exact matches, but also what it thinks are synonyms for each of the keywords. Searching for:

```
~ape
```

turns up results for monkey, gorilla, chimpanzee, and others (both singular and plural forms) of the ape or related family, as if you'd searched for:

```
monkey gorilla chimpanzee
```

along with results for some words you'd never have thought to include in your query.

Google figures out synonyms algorithmically, so you may be surprised to find results that your garden-variety thesaurus would not have suggested. (Synonyms are bolded along with exact keyword matches on the results page, so they're easy to spot.)

Number Range

One of the more difficult things to convey in an Internet search query is a rangeof dates, currency,

size, weight, height, or any two arbitrary values.

The number range operator, `..` (two periods), looks for results that fall inside your specified numeric range.

Looking for that perfect pair of Prada pumps, size 5 or 6? Try this for size:

```
prada pumps size 5..6
```

Perhaps you're looking to spend $800 to $1,000 on a nice digital SLR camera; Google for:

```
slr digital camera 3..5 megapixel $800..1000
```

The one thing to remember is always to provide some clue as to the meaning of the range, e.g.,`$`, `size`, `megapixel`, `kg`, and so forth.

You can also use the number range syntax with just one number, making it the minimum or maximum of your query. Do you want to find some land in Montana that's at least 500 acres? No problem:

```
acres Montana land 500..
```

On the other hand, you might want to make sure that raincoat you buy for your terrier doesn't cost more than $30. That's possible too:

```
raincoat dog ..$30
```

> Google normally does not recognize special characters such as`$` in the search process. But because the `$` sign was necessary for the number feature, you can use it in all sorts of searches. Try the search `"yard sale" bargains 10` and then `"yard sale" bargains $10`. Notice how the second search gives you far fewer results? That's because Google is matching `$10` exactly.

Simple Searching and Feeling Lucky

The I'm Feeling Lucky™ button is a thing of beauty. Rather than giving you a list of search results from which to choose, you're whisked away to what Google believes is the most relevant page given your search (i.e., the first result in the list). Entering`washington post` and clicking the I'm Feeling Lucky button takes you directly to http://www.washingtonpost.com. Trying `president` will land you at http://www.whitehouse.gov.

Case Sensitivity

Some search engines are case-sensitive; that is, they search for queries based on how the queries

are capitalized. A search for `"GEORGE WASHINGTON"` on such a search engine would not find "George Washington," "george washington," or any other case combination.

Google is case-insensitive. If you search for `Three`, `tHRee`, `THREE`, or even `THREE`, you get the same results.

PREV

Full-Word Wildcards

Some search engines support a technique called *stemming*, in which you add a wildcard characterusually `*` (asterisk) but sometimes `?` (question mark)to part of your query, requesting the search engine to return variants of that query using the wildcard as a placeholder for the rest of the word. For example, `moon*` would find moons, moonlight, moonshot, etc.

Google doesn't support explicit stemming. It didn't used to support stemming at all, but now it implicitly stems for you. So, `canine dietary` will yield results for dog diet, diets, and other variations on the theme.

Google does offer a full-word wildcard. While a wildcard can't stand in for part of a word, you can insert a wildcard (Google's wildcard character is `*`) into a phrase, and the wildcard will act as a substitute for one full word. Searching for `tHRee * mice`, therefore, finds three blind mice, three blue mice, three green mice, etc.

What good is the full-word wildcard? It's certainly not as useful as stemming, but then again, it's not as confusing to the beginner. `*` is a stand-in for one word; `**` signifies two words, and so on. The full-word wildcard comes in handy in the following situations:

- Checking the frequency of certain phrases and derivatives of phrases, such as: `intitle:"methinks the * doth protest too much"` and `intitle: "the * of Seville"` (`intitle:` is described next in "Special Syntax").

- Filling in the blanks on a fitful memory. Perhaps you remember only a short string of song lyrics; search using only what you remember rather than randomly reconstructed full lines.

- Let's take as an example the disco anthem "Good Times" by Chic. Consider the following line: "You silly fool, you can't change your fate."

- Perhaps you've heard that lyric, but you can't remember if the word "fool" is correct or if it's something else. If you're wrong (if the correct line is, for example, "You silly child, you can't change your fate"), your search will find no results and you'll come away with the sad conclusion that no one on the Internet has bothered to post lyrics to Chic songs.

- The solution is to run the query with a wildcard in place of the unknown word, like so:
- `"You silly *, you can't change your fate"`

- You can use this technique for quotes, song lyrics, poetry, and more. You should be mindful, however, to include enough of the quote to find unique results. Searching for `"you * fool"` will glean far too many irrelevant hits.

PREV

Special Syntax

In addition to the basic `AND`, `OR`, and phrase searches, Google offers some rather extensive special syntax for narrowing your searches.

As a full-text search engine, Google indexes entire web pages instead of just titles and descriptions. Additional commands, called *special syntax* , or *advanced operators*, let Google users search specific parts of web pages for specific types of information. This comes in handy when you're dealing with more than eight billion web pages and need every opportunity to narrow your search results. Specifying that your query words must appear only in the title or URL of a returned web page is a great way to specify your results without making your keywords themselves too specific. Following are descriptions of the special syntax elements, ordered by common usage and function.

Some of these syntax elements work well in combination. Others don't fare quite as well. Still others do not work at all. For a detailed discussion of what does and does not mix, see "Mixing Syntax" later in this chapter.

`intitle:`

`intitle:` restricts your search to the titles of web pages. The variation `allintitle:` finds pages in which all the specified words appear in the title of the web page. Using `allintitle:` is basically the same as using `intitle:` before each keyword:

```
intitle:"george bush"
allintitle:"money supply" economics
```

You may wish to avoid the `allintitle`: variation because it doesn't mix well with some of the other syntax elements.

`intext:`

`intext:` searches only body text (i.e., it ignores link text, URLs, and titles). While its uses are limited, it's perfect for finding query words that might be too common in URLs or link titles:

```
intext:"yahoo.com"
intext:html
```

There's an `allintext:` variation; but again, this doesn't play well with others.

`inanchor`:

`inanchor:` searches for text in a page's link anchors. A link anchor is the descriptive text of a link. For example, the link anchor in the HTML code `<a href="http://www.oreilly.com">O'Reilly Media</a>` is "O'Reilly Media."

```
inanchor:"tom peters"
```

As with other `in*:` syntax elements, there's an `allinanchor:` variation, which works in a similar way (i.e., all the keywords specified must appear in a page's link anchors).

site:

`site:` allows you to narrow your search by a site or by a top-level domain. The AltaVista search engine, by contrast, has two syntax elements for this function (`host:` and `domain:`), but Google has only the one:

```
site:loc.gov
site:thomas.loc.gov
site:edu
site:nc.us
```

Be aware that `site:` is no good for searching for a page that exists beneath the main or default site (i.e., in a subdirectory such as */~sam/album/*). For example, if you're looking for something below the main GeoCities site, you can't use `site:` to find all the pages in http://www.geocities.com/Heartland/Meadows/6485/; Google returns no results. Use `inurl:` instead.

inurl:

`inurl:` restricts your search to the URLs of web pages. This syntax usually works well for finding search and help pages because they tend to be regular in composition. An `allinurl:` variation finds all the words listed in a URL but doesn't mix well with some other special syntax

```
inurl:help
allinurl:search help
```

You'll see that using the `inurl:` query instead of the `site:` query has one immediate advantage: you can use it to search subdirectories.

> While the `http://` prefix in a URL is ignored by Google when used with `site:`, search results come up short when it is included in an `inurl:` query. Be sure to remove prefixes in any `inurl:` query for the best (read: any) results.

link:

`link:` returns a list of pages that link to the specified URL. Enter `link:www.google.com` and you'll get a list of pages that link to the Google home page, http://www.google.com (not anywhere in the google.com domain). Don't worry about the *http://* bit; you don't need it and,

indeed, Google appears to ignore it even if you do put it in. `link:` works just as well with "deep" URLs http://www.raelity.org/apps/blosxom/, for instanceas with top-level URLs such as *raelity.org*.

`cache:`

`cache:` finds a copy of the page that Google indexed even if that page is no longer available at its original URL or has since changed its content completely:

```
cache:www.yahoo.com
```

If Google returns a result that appears to have little to do with your query, you're almost sure to find what you're looking for in the latest cached version of the page at Google.

The Google cache is particularly useful for retrieving a previous version of a page that changes often.

`filetype:`

`filetype:` searches the suffixes or filename extensions. These are usually, but not necessarily, different file types; `filetype:htm` and `filetype:html` will give you different result counts, even though they're the same file type. You can even search for different page generatorssuch as ASP, PHP, CGI, and so forthpresuming the site isn't hiding them behind redirection and proxying. Google indexes several different Microsoft formats, including PowerPoint (*ppt*), Excel (*.xls*), and Word (*.doc*):

```
homeschooling filetype:pdf
"leading economic indicators" filetype:ppt
```

`related:`

`related:` , as you might expect, finds pages that are related to the specified page. This is a good way to find categories of pages; a search for `related:google.com` returns a variety of search engines, including Lycos, Yahoo!, and Northern Light:

```
related:www.yahoo.com
related:www.cnn.com
```

While an increasingly rare occurrence, you'll find that not all pages are related to other pages.

`info:`

`info:` provides a page of links to more information about a specified URL. This information includes a link to the URL's cache, a list of pages that link to the URL, pages that are related to the URL, and pages that contain the URL:

```
info:www.oreilly.com
info:www.nytimes.com/technology
```

Note that this information is dependent on whether Google has indexed the specified URL; if it hasn't the information will obviously be far more limited.

`phonebook:`

`phonebook:` , as you might expect, looks up phone numbers:

```
phonebook:John Doe CA
phonebook:(510) 555-1212
```

The phonebook is covered in detail in "Google Phonebook: Let Google's Fingers Do the Walking" [Hack #5].

`define:`

`define:` gives you a page full of definitions of a word from around the Web:

```
define:paradigm
```

Google often displays related phrases in addition to definitions and the URLs where the definitions were found.

`movie:`

Use the `movie:` syntax to find reviews of movies on the Web, like this:

```
movie:matrix
```

You can also use a zip code or a city and state combination to find local theater listings and movie showtimes:

```
movie:97333
movie:corvallis, or
```

`music:`

`music:` explicitly searches for music-related information:

```
music:pink floyd
```

You're given a page that splits results into matching artists, albums, and lyrics, and you can choose to explore any of these areas in depth.

PREV

Mixing Syntax

There was a time when you couldn't mix Google's special syntax elements; you were limited to one per query. Even as Google released ever more powerful special syntax elements, not being able to combine them for their composite power stunted many a search.

This has since changed. While there remain some syntax elements that you just can't mix, there are plenty to combine in clever and powerful ways. A thoughtful combination can do wonders to narrow a search.

How Not to Mix Syntax

There are some simple rules to follow when mixing syntax elements. These, for the most part, revolve around how *not* to mix:

- Don't mix syntax elements that will cancel out each other, such as:
- `site:ucla.edu -inurl:ucla`

- Here, you're saying you want all results to come from ucla.edu, but that site results should not have the string "ucla" in the results. Obviously, that's not going to produce many URLs.

- Don't overuse single syntax elements, as in:
- `site:com site:edu`

- While you might think you're asking for results from either *.com* or *.edu* sites, what you're actually saying is that site results should come from both simultaneously. Obviously, a single result can come from only one domain. Take the example `perl site:edu site:com`. This search will get you exactly zero results. Why? Because a result page cannot come from a *.edu* domain and a *.com* domain at the same time. If you want results from *.edu* and *.com* domains only, rephrase your search like this:
- `perl (site:edu | site:com)`

- With the pipe character (`|`), you specify that you want results to come either from the *.edu* or the *.com* domain.

- Don't use `allinurl:` or `allintitle`: when mixing syntax. It takes a careful hand not to misuse these in a mixed search. Instead, stick to `inurl:` or `intitle:`. If you don't put `allinurl:` in exactly the right place, you'll create odd search results. Let's look at an example:

- `allinurl:perl intitle:programming`

- At first glance, it looks like you're searching for the string "perl" in the result URL and the word "programming" in the title. And you're right: this will work fine. But what happens if you move `allinurl:` to the right of the query?
- `intitle:programming allinurl:perl`

- This won't bring any results. Stick to `inurl:` and `intitle:`, which are much more forgiving of where you put them in a query.
- The same advice goes for `allintext:` and `allinanchor:`.
- Don't use so much syntax that you get too narrow, as in:
- `title:agriculture site:ucla.edu inurl:search`

- You might find that your search is too narrow to give you any useful results. If you're trying to find something so specific that you think you need a narrow query, start by building a little bit o the query at a time. Say you want to find plant databases at UCLA. Instead of starting with the query:
- `title:plants site:ucla.edu inurl:database`

- try something simpler:
- `databases plants site:ucla.edu`

- and then try adding syntax to keywords you've already established in your search results:
- `intitle:plants databases site:ucla.edu`

- or:
- `intitle:database plants site:ucla.edu`

How to Mix Syntax

If you're trying to narrow down search results, the `intitle:` and `site:` syntax elements are your best bet.

Titles and sites

For example, say you want to get an idea of what databases are offered by the state of Texas. Run this search:

```
intitle:search intitle:records site:tx.us
```

You'll find something on the order of 30 very targeted results. And, of course, you can narrow down your search even more by adding keywords:

```
birth intitle:search intitle:records site:tx.us
```

It doesn't seem to matter whether you put plain keywords at the beginning or at the end of the search query; I put them at the beginning because they're easier to keep up with.

The `site:` syntax, unlike site syntax on other search engines, allows you to get as general as a domain suffix (`site:com`) or as specific as a domain or subdomain (`site:thomas.loc.gov`). So if you're looking for records in El Paso, you can use this query:

```
intitle:records site:el-paso.tx.us
```

and you'll get approximately one result.

Title and URL

Sometimes you want to find a certain type of information, but you don't want to narrow by title. Instead, you want to narrow by theme (e.g., you want sites about "help" or about "search engines"). In such cases, you need to search text within the URL.

The `inurl:` syntax searches for a string in the URL but doesn't count it if it appears within a larger word. So, for example, if you search for `inurl:research`, Google will not find pages from http://www.researchbuzz.com, but it will find pages from http://www.research-councils.ac.uk.

Say you want to find information on neurosurgery, with an emphasis on learning or assistance. Try:

```
intitle:neurosurgery inurl:help
```

This returns a more manageable 880 or so results. The whole point is to get a number of results that includes what you need but isn't so large as to be overwhelming. If you find that 880 results are too much, you can easily mix the `site:` syntax into the search and limit your results to universities:

```
intitle:neurosurgery inurl:help site:edu
```

Beware, however, of using too much special syntax. As mentioned earlier, you can quickly detail yourself into no results at all.

The Antisocial Syntax Elements

The antisocial syntax elements don't mix and should be used individually for maximum effect. If you try to use them with other syntax elements, you won't get any results.

The syntax elements that request special information `rphonebook:` , `bphonebook:` , `movie:`, `music:`, `define:`, and `phonebook:` are all antisocial. That is, you can't mix them and expect to get a reasonable result.

The other antisocial syntax element is `link:`, which shows pages that link to a specified URL. Wouldn't it be great if you could specify the domains you want the pages to be from? Sorry, you can't. The `link:` syntax does not mix with anything elsenot even plain old keywords.

For example, say you want to find out which pages link to O'Reilly Media, Inc., but you don't want to include pages from the *.edu* domain. The query `link:www.oreilly.com -site:edu` will not work because the `link:` syntax does not work in combination. Well, that's not quite correct; you will get results, but they'll be for the phrase "link:www.oreilly.com" from domains that are not *.edu*.

If you want to search for links and exclude the *.edu* domain, there's no single command that absolutely works. This one's a good try, though:

```
inanchor:oreilly -inurl:oreilly -site:edu
```

This search looks for the word "oreilly" in *anchor text*, the text that's used to define links; excludes pages that contain "oreilly" in the *search result* (e.g., oreilly.com); and, finally, excludes those pages that come from the *.edu* domain.

But this type of search is nowhere near complete. It finds only those links to O'Reilly that include the string "oreilly": if someone creates a link such as `<a HRef="http://perl.oreilly.com/">Camel Book</a>`, it won't be found by the preceding query. Furthermore, there are other domains that contain the string "oreilly," and there may be domains that link to "oreilly" that contain the string "oreilly" but aren't oreilly.com. You could alter the string slightly to omit the oreilly.com site itself but not other sites containing the string "oreilly":

```
inanchor:oreilly -site:oreilly.com -site:edu
```

However, you would still include many O'Reilly sitesXML.com and MacDevCenter.com, for instancethat aren't at oreilly.com.

All the Possibilities

While it is possible to write down every syntax-mixing combination and briefly explain how they migh be useful, there wouldn't be room for much else in this book.

Experiment. Experiment a lot. Constantly keep in mind that most of these syntax elements do not stand alone, and you can get more done by combining them than by using them individually.

Depending on the kind of research you are doing, different patterns will emerge over time. For example, you may discover that focusing on only PDF documents (`filetype:pdf`) finds you the results you need. You may discover that you should concentrate on specific file types in specific domains (`filetype:ppt site:tompeters.com`). Mix up the syntax in as many ways as is relevant to your research and see what you get.

As with anything else, the more you use Google's special syntax, the more natural it will become to you. And Google is constantly adding more, much to the delight of regular web combers.

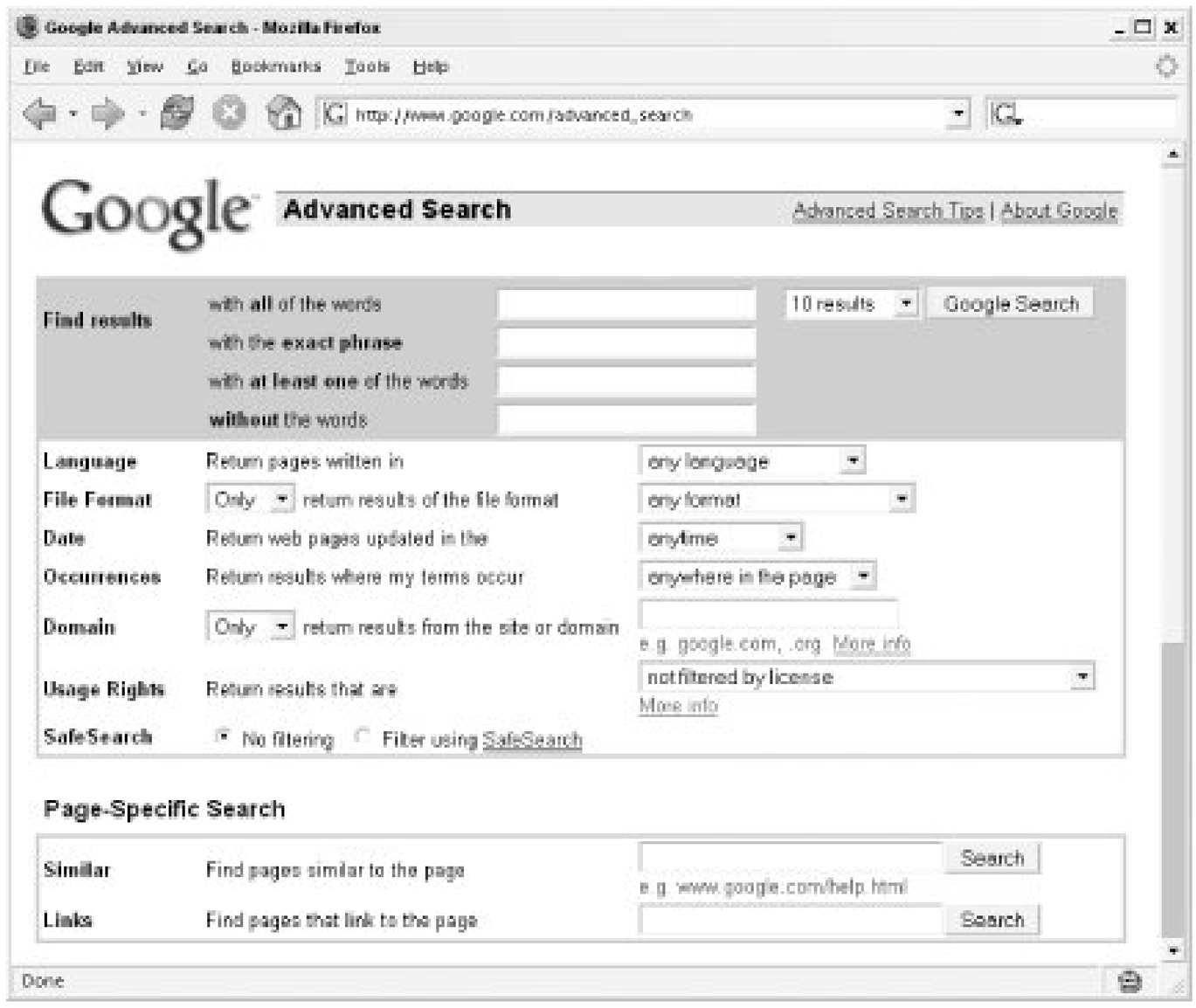
If, however, you want something more structured and visual than a single query line, Google's Advanced Search should fit the bill.

PREV

Advanced Search

Google's default simple search allows you to do quite a bit, but not everything. Google's Advanced Search page (http://www.google.com/advanced_search), shown in Figure 1-1, provides more options, such as date search and filtering, with "fill in the blank" searching options for those who don't take naturally to memorizing special syntax.

Figure 1-1. Google's Advanced Search page



Most of the options presented on this page are self-explanatory, but we'll take a quick look at the kinds of searches that would be more difficult using the single-text-field interface of a simple search.

Query Words

Because Google uses Boolean `AND` by default, it's sometimes hard to logically build out the nuances of a particular query. Using the text boxes at the top of the Advanced Search page, you can specify words that *must* appearexact phrases or lists of words, at least one of which must appearand words to be excluded.

Language

Using the Language pull-down menu, you can specify thelanguage all returned pages must be in, from Arabic to Turkish.

File Format

The File Format option lets you include or exclude severaldifferent file formats, including Microsoft Word and Excel. A couple Adobe formats (most notably PDF) and Rich Text Format are options here, too. This is where the Advanced Search is at its most limited: there are literally dozens of file formats that Google can search for, and this set of options represents only a small subset. To get at the others, use the `filetype:` special syntax described earlier in "Special Syntax."

Date

Date allows you to specify search results updated in the last three, six, or twelve months. This date search is much more limited than the `daterange:` special syntax, which can give you results as narrow as one day, but Google stands behind the results generated using the Date option on the Advanced Search, while not officially sanctioning the use of the `daterange:` search.

Occurrences

Using the Occurrences pull-down menu, youcan specify where the terms should occur. The options here, other than the default, generally reflect the `allin*:` syntax elementsin the title (`allintitle:`), text (`allintext:`), URL (`allinurl:`), and link anchors (`allinanchor:`) of the page.

Domain

The Domain feature is an interface to the `site:` syntax. It also allows negation (explained earlier) to explicitly *not* return results from a site or domain.

Usage Rights

If you're looking for materials that you can legally reuse in your reports, presentations, or other compilations, you can specify that you're looking for materials licensed with alternative copyright systems, such as Creative Commons licenses (http://creativecommons.org). You can look for files that are "free to use or share," "free to use, share, or modify," and other variations on this theme.

Safe Search

Google's Advanced Search also gives you the option to filter your results usingSafeSearch. SafeSearch filters only sexually explicit content (as opposed to some filtering systems that filter pornography, hate material, gambling information, etc.). Please remember that machine filtering isn' 100 percent perfect.

Page-Specific Search

The last two fields in the form provide a simple way to use the`related:` and `link:` syntaxes. You can use these special searches to find more information about any specific site.

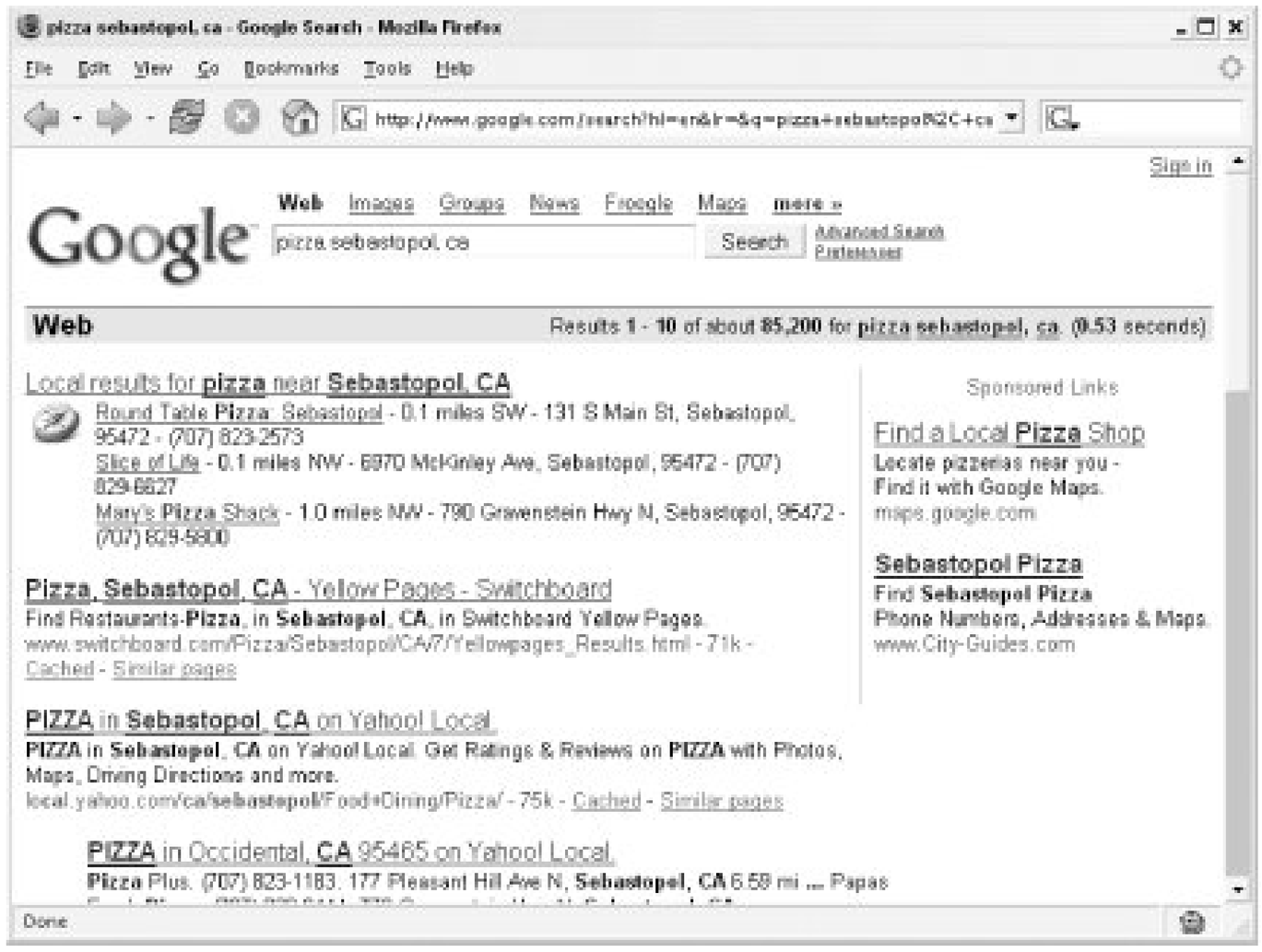
The Advanced Search page is handy when you need to use its unique features or need help putting together a complicated query. Its "fill-in-the-blank" interface comes in handy for the occasional searcher or anyone who wants to get an advanced search exactly right. That said, it is limiting in other ways. It's difficult to use mixed syntax or build a single syntax search using`OR`. For example, there's no way to search for `site:edu OR site:org` using the Advanced Search. This search must be done from the Google search box.

Of course, there's another way you can alter the search results that Google gives you, and it doesn't involve the basic search input or the Advanced Search page. It's the preferences page, described in "Setting Preferences" later in this chapter.

Quick Links

If you're a Google regular, you've no doubt noticed those snippets of linked information proliferating near the top-left of the first results page (see Figure 1-2). Where once there was only a sponsored link or two between you and your results, now there are spelling suggestions, news headlines, stock quotes, and all other manner of bits and bobs of rather useful information.

Figure 1-2. Quick links augmenting search results with relevant, current, and local information



Google is going beyond web search results to include relevant finds from its other properties and those of third parties. Here, briefly, is the current catalog of quick links:

Spelling

One nice side effect of Google's listening to the Web is that it picks up a lot of words along the way. Some appear in the dictionary, while others haven't quite made their way into common parlance. Some are made up, while others are simply misspelled. Query Google for something that is commonly spelled another way, and it'll proffer some suggestions. "Consult the Dictionary" delves further into the wonders of Google's spell checker.

Definitions

TLAs (that's "three-letter acronyms") and geek speak abound. Rather than smiling knowingly when you've not a clue what someone just said, ask Google if it knows what your friend, boss, or medical professional is talking about. Prepend just about any word, obscure or garden-variety, with `define` (e.g., `define happy`), and the first item on your results page will in all probability be a definition pulled from one of any number of web dictionaries. Use `define:` (note the colon e.g., `define:osteichthyes`) to pull up a whole page full of definitions [Hack #6].

News Headlines

Google News (http://news.google.com; see Chapter 3) scrapes stories from thousands of news sources. Don't be surprised if there's something new and noteworthy related to your Google search.

Travel Information

Before you hop on that plane, Google your destination using the airport name (e.g., `Los Angeles`) or code (e.g., `LAX`) and the word `airport`. Click the "View conditions at [in this case] Los Angeles International (LAX), Los Angeles, California" link to visit the Federal Aviation Administration's (FAA) real-time airport status information. At the moment of this writing, LAX has no destination-specific delays, and both departures and arrivals are experiencing fewer than 15-minute gate hold and airborne delays, respectively.

Street Maps

If Google gleans something that looks like a geographic location in your search query, it'll provide a link to a Google Map pinpointing the location, along with links to Yahoo! and MapQuest maps of the area.

Google Maps

Include the name of a city, state, or zip code anywhere in the U.S. or Canada in your search, and Google Local (http://local.google.com) just might suggest a local find. Google for `indian food portland oregon`, and you'll find yourself tempted by the flavors of Swagat Indian Cuisine on NW Lovejoy Street or India Grill on E Burnside.

Calculator

You might remember a few important numbers from math class: pi or e or C, for instance. But numbers hold a very special place in Google's collective heart; after all, the name Google comes from *googol*, or 10^{100}. So it shouldn't come as a surprise that the geeks at Google have taught the search engine to pay attention to particular patterns of numbers, including anything that looks like a calculation. Type any equation into the search form, and you'll get an answer back:

```
365/12
```

```
9*3
```

You can also use the Google Calculator to convert units. Simply type out the conversion you want to perform:

```
12 ounces in pounds
3 meters in yards
```

Google can also convert currency in the same way. Simply include the two types of currency you'd like to compare:

```
12 USD in Euros
```

Google by Numbers, 1-2-3

In addition to calculations, Google looks for special patterns usually found in particular reference numbers, including:

- UPS, FedEx, and U.S. Postal Service tracking numbers (e.g., `1Z9999W99999999999`). Google links to the package service's tracking page and fills in the number to get you going.
- Vehicle ID (VIN) numbers (e.g., `AAAAA999A9AA99999`).
- UPC codes (e.g., `073333531084`) at http://www.upcdatabase.com.
- Telephone area codes (e.g., `510`) at http://www.whitepages.com.
- Patent numbers (e.g., `patent 4920273`) in the U.S. Patent Database.
- Federal Aviation Administration (FAA) airplane registration numbers (e.g., `n199ua`). These are particularly entertaining when you're waiting to board your plane, smartphone in hand and "Google on the Go." Look for them on the plane's tail.
- Federal Communications Commission (FCC) equipment ID numbers (e.g., `fcc B4Z-34009-PIR`).

Stock Quotes

Search for a stock symbol [Hack #16] and you'll be quick-linked to the company's financial information at Google Finance, Yahoo! Finance, and a number of other sites that offer stock information.

Froogle Products

If Froogle (http://froogle.google.com) finds a product that seems to be what you're after, it'll link to "Product search results" and to two or three offerings at sites such as eBay, Golfsmith, Buy.com, and many more.

Weather

> Type in the word `weather` followed by a city name for a quick look at current conditions and the five-day forecast.

There are sure to be more quick links by the time you read this. To keep apprised of what's new, periodically visit the Google Web Search Features (http://www.google.com/help/features.html), or just keep Googling and see what appears.

PREV

Language Tools

In the early days of the Web, it seemed like most web pages were in English. But as more and more countries have come online, materials have become available in a variety of languagesincluding languages that have not originated from a particular country (such as Esperanto and Klingon).

Google offers several language tools, including one for translation and one for Google's interface. The interface option is much more extensive than the translation option, but the translation option has a lot to offer.

The language tools are available by clicking the Language Tools link on the front page or by going to http://www.google.com/language_tools.

Search Specific Languages or Countries

The first tool allows you to search for materials from a certain country and/or in a certain language. This is an excellent way to narrow your searches; searching for French pages from Japan gives you far fewer results than searching for French pages from France. You can narrow the search further by searching for a slang word in another language. For example, search for the English slang word*bonce* on French pages from Japan.

Translate

The second tool on this page allows you to translate either a block of text or an entire web page from one language to another. Most of the translations are to or from English.

Machine translation is not nearly as good as human translation, so don't rely on this translation as either the basis of a search or as a completely accurate translation of the page you're looking at. Instead, use it to get the gist of whatever it translates.

You don't have to come to this page to use the translation tools. When you enter a search, you'll see that some search results that aren't in your language of choice (which you set via Google's preferences) have "[Translate this page]" next to their titles. Click on one of these and you'll be presented with a framed, translated version of the page. The Google frame at the top allows you to view the original version of the page, as well as return to the results or view a copy suitable for printing.

Interface Language

The third tool lets you choose the interface language for Google, from Afrikaans to Welsh. Some of these languages are imaginary (Bork, bork, bork! and Elmer Fudd), but they do work.

Be warned that if you set your language preference to Klingon, for example, you'll need to know Klingon to figure out how to set it back.

As one of our *Google Hacks* readers, Jacek Artymiak, pointed out (http://hacks.oreilly.com/pub/h/360), if English is your native tongue, point your browser at http://www.google.com/intl/en. If you're not an English speaker but remember or care to guess at the language code (e.g., zu for Zulu), drop it in instead of en at the end of the URL. Further discussion revealed that simply suffixing the http://www.google.com URL with a period http://www.google.com. has the same delocalizing effect, reverting the interface to English.

If you're really stuck, delete the Google cookie from your browser and reload the page; this should reset all preferences to the defaults.

How does Google manage to have so many interface languages when it has so few translation languages? The Google in Your Language program gathers volunteers from around the world to translate Google's interface. (You can get more information on this program at http://www.google.com/intl/en/language.html.)

Local Domain

Finally, the Language Tools page contains a list of region-specific Google home pages over 100 of them, from Deutschland to the Pitcairn Islands.

Making the Most of Google's Language Tools

While you shouldn't rely on Google's translation tools to give you more than the gist of the meaning (since machine translation isn't that good), you can use translations to narrow your searches. I described the first method earlier: use unlikely combinations of languages and countries to narrow your results. The second way involves using the translator.

Select a word that matches your topic and use the translator to translate it into another language. (Google's translation tools work very well for single-word translations like this.) Now, search for that word in a country and language that don't match it. For example, you might search for the German word "Landstra\xa7 e" (highway) on French pages in Canada. Of course, you must be sure to use words that don't have English equivalents or you'll be overwhelmed with results.

Whew! By now it should be fairly clear that a simple interface such as the one on Google's front page does not necessarily imply limited power. Still waters run deep indeed. Now that we have all the tools, tips, and techniques under our belt to help us ask Google for what we want before it dives into the depths of web content, it's time to turn our attention to understanding what it brings back to the surface.

Anatomy of a Search Result

You'd think a list of search results would be pretty straightforward, wouldn't youjust a page title and a link, possibly a summary? Not so with Google. Google encompasses so many search properties and has so much data at its disposal that it fills every results page to the rafters. Within a typical search result, you can find sponsored links, ads, links to stock quotes, page sizes, spelling suggestions, and more.

By knowing more of the nitty-gritty details of what's what in a search result, you'll be able to make some guesses ("Wow, this page that links to my page is very large; perhaps it's a link list") and correct roadblocks ("I can't find my search term on this page; I'll check the version Google has cached").

Let's use the word "flowers" to examine this anatomy.Figure 1-3 shows the result page for `flowers`.

Figure 1-3. Results page for "flowers"

First, note that at the top of the page a selection of tabs allows you to repeat your search across other Google search categories besides web pages, including Google Images, Google Groups, Google News, Froogle, and Google Maps. Beneath that is a count of the number of results and how long the search took: about 524,000,000 results in 0.14 seconds (this will vary, sometimes by quite a bit).

Sometimes results/sites are called out on colored backgrounds at the top or right of the results page (see Figure 1-3). These are called *sponsored links* (read: advertisements). Google has a policy of very clearly distinguishing ads and sticking to text-based advertising only rather than throwing flashing banners in your face like other sites do.

You might also see Quick Links for some queries that Google thinks it has a direct answer for, but for the most part you'll see a list of 10 results. The first real (i.e., nonsponsored) result of the search for `flowers` is shown in Figure 1-4.

Figure 1-4. A typical search result

Let's break this down into chunks, shall we?

The top line of each result is the page title, hyperlinked to the original page.

The second line offers a brief extract from this site. Sometimes this is a description of the site or a selected sentence or two. Sometimes it's HTML mush. Google tends to use description metatags when they're available; it's rare when you can look at a Google search result and not have even a modicum of an idea what the site is about.

The next line sports several informative bits of metadata. First, there's the URL. Second, there's the size of the page (Google makes the page size available only if the page has been cached). Third, there's a link to a cached version of the page if one is available. Finally, there's a link to find similar pages.

Why would you bother reading the search-result metadata? Why not simply visit the site and see if it has what you want?

If you have a broadband connection and all the time in the world, you might not want to bother checking out the metadata. But if you have a slower connection and time is at a premium, consider the search-result information.

First, check the page summary. Where does your keyword appear? Does it appear in the middle of a list of site names? Does it appear in a way that makes it clear that the context is not what you're looking for?

Check the size of the page if it's available. Is the page very large? Perhaps it's just a link lista page full of hyperlinks, as the name suggests. Is it just 1 or 2 KB? It might be too small to find the level of detail that you're looking for. If your aim is link lists, be on the lookout for pages larger than 20 KB, and see "Browse the Google Directory" [Hack #1].

> Page size in Google results will never be more than 101 KB. This is because Google doesn't index more than 101 KB of a given web page.

PREV

Setting Preferences

Google's Preferences page, shown in Figure 1-5, provides a nice, easy way to set and save your search preferences.

Figure 1-5. Google's Preferences page

Interface Language

You can set your Interface Language, the language in which tips and messages are displayed.

Search Language

Not to be confused with Interface Language, Search Language restricts the languages that are considered when searching Google's page index. The default is any language, but you could be interested only in web pages written in Chinese and Japanese, or French, German, and Spanishthe combination is up to you.

SafeSearch Filtering

Google's SafeSearch filtering affords you a method of avoiding search results that may offend your sensibilities. No filtering means you're offered anything in the Google index. Moderate filtering rules out explicit images, but not explicit language. Strict filtering filters both text and images. The default is moderate filtering.

Number of Results

By default, Google displays 10 results per page. For more results, click any of the Result Page: 1 2 3... links at the bottom of each result page, or simply click the Next link.

You can specify your preferred number of results per page (10, 20, 30, 50, or 100), along with whether you want results to open in the current window or a new browser window.

Results Window

You can choose to open search results in a new browser windowhandy for keeping your search results in place. If you've ever clicked from site to site only to find you've completely lost the page of results you'd like to return to, try enabling this option.

Settings for Researchers

For the purpose of research, it's best to have as many search results as possible on the page. Because it's all text, it doesn't take that much longer to load 100 results than it does to load 10. If you have a computer with a decent amount of memory, it's also good to have search results open in a new window, which will keep you from losing your place and leave you a window with all the search results readily available.

If you can stand it, turn off filtering, or at least limit the filtering to moderate instead of strict. Machine filtering is not perfect and, unfortunately, enabling it might mean that you'll miss something valuable. This is especially true when you're searching for a phrase that might be caught by a filter, such as "breast cancer."

Unless you're absolutely sure you always want to do a search in one language, I advise against setting your language preferences on this page. Instead, alter language preferences as needed using the Google Language Tools ["Language Tools" earlier in this chapter].

Between the simple search, advanced search, and preferences, you have all the tools necessary to build the Google query to suit your particular purposes.

If cookies are turned off in your browser, setting preferences in Google isn't going to do you much good. You'll have to reset them every time you open your browser. If you can't have cookies and want to use the same preferences every time, consider making a customized search form [Hack #9].

PREV

Understanding Google URLs

If you're like most people, you usually pay little attention to the URLs in your browser's address bar as you surf from one site to the next. And you might choose to stick with this habit while searching Google. You ought to know, however, that a subtle alteration made to the URL that Google returns after a search can be an efficient method of tweaking your result set. In fact, there's at least one thing you can do by fiddling with (we like to call it*hacking*) the URL that you can do no other way, and there are quick tricks that might save you a trip back to the Advanced Search page.

Say you want to search for `tHRee blind mice`. The URL of the page of results will vary depending on the preferences you've set, but it will look something like this:

```
http://www.google.com/search?num=100&hl=en&q=%22three+blind+mice%22
```

The query itself`q=%22tHRee+blind+mice%22`, `%22` being a URL-encoded `"` (double quote)is pretty obvious, but let's break down what those extra bits mean.

`num=100` refers to the number of search results per page100 in this case. Google accepts any number from 1 to 100. Altering the value of`num` is a nice shortcut to altering the preferred size of your result set without having to meander over to the Advanced Search page and rerun your search.

Don't see the `num=` in your URL? Simply append it by clicking at the end of the URL in your browser's address bar and typing it in. To set the number of results per page to 20, for instance, add`&num=20`.

> You can add or alter any of the modifiers described here by appending them to the URL or changing their valuesthe part after the`=` (equals)to something within the accepted range for the modifier in question. If you're adding a modifier, you must use an`&` (ampersand) too. Look at how the modifiers are joined together on URLs for other search results to see how it's done.

`hl=en` refers to the language interface (the language in which you use Google, reflected in the home page, messages, and buttons). Here, it's in English. Google's Language Tools ["Language Tools" earlier in this chapter] page provides a list of language choices. Run your mouse over each language choice and notice the change reflected in the URL. The URL for Pig Latin looks like this:

```
http://www.google.com/intl/xx-piglatin/
```

The language code is the bit between`intl/` and the last `/xx-piglatin`, in this case. Apply this to the search URL at hand by altering the existing value of`hl`:

```
hl=xx-piglatin
```

What if you put multiple `hl` modifiers in a result URL? Google honors whichever comes last, reading from left to right. While it makes for confusing URLs, this means you can always resort to laziness and add an extra modifier at the end rather than editing what's already there, like so:

```
http://www.google.com/search?num=100&hl=en&q=%22three+blind+mice%22&hl=xx-piglatin
```

There's one more modifier that, appended to your URL, may provide some useful modifications of your results:

`safe=off`

> Means the SafeSearch filter is off. The SafeSearch filter removes search results of a sexually explicit nature. `safe=on` means the SafeSearch filter is on.

Playing about with Google's URLs [Hack #17] might not seem like the most intuitive way to get results quickly, but it's much faster than reloading the Advanced Search form.

PREV

Hack 1. Browse the Google Directory

Google has a searchable subject index in addition to its Web Search.

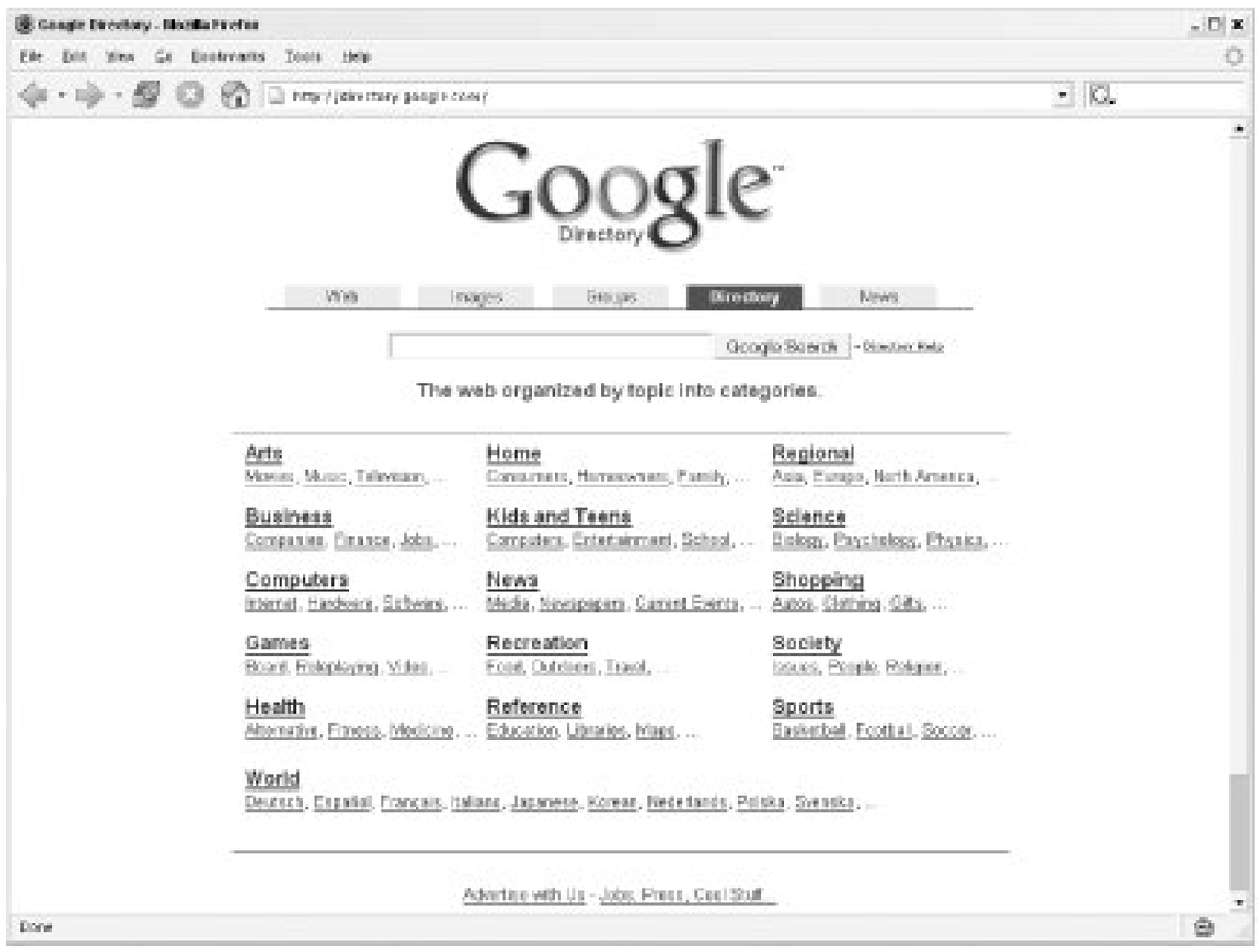
Google's Web Search indexes billions of pages, which means it isn't suitable for all searches. When you have a search that you can't narrow downfor example, if you're looking for information on a person about whom you know nothingbillions of pages will get very frustrating very quickly.

But you don't have to limit your searches to the Web. Google also has a searchable subject index, th Google Directory, at http://directory.google.com. Instead of indexing the entirety of billions of pages, the directory describes sites instead, indexing about five million URLs. This makes it a much better search for general topics.

Does Google spend time building a searchable subject index in addition to a full-text index? No, Google bases its directory on the Open Directory Project data at http://dmoz.org/. Unlike the results at the standard Google Web Search, the collection of URLs at the Open Directory Project is gathered and maintained by a group of human volunteers rather than automatic algorithms, but Google does add some of its own Googlish magic to it.

As you can see in Figure 1-6, the front of the site is organized into several topics. To find what you're looking for, you can either do a keyword search or *drill down* through the hierarchies of subjects.

Figure 1-6. The Google Directory



Beside most listings, as shown in Figure 1-7, you'll see a green bar. The green bar is an approximate indicator of the site's PageRank in the Google search engine. (Not every listing in the Google Directory has a corresponding PageRank in the Google web index.) Web sites are listed in the default order of Google PageRank, but you also have the option to list them in alphabetical order.

Figure 1-7. Individual listings under Science Physics Quantum Mechanics People Feynman, Richard

One thing you'll notice about the Google Directory is how the annotations and other information vary between categories. This is because the information in the directory is maintained by a small army of thousands of volunteers who are each responsible for one or more categories. For the most part, annotation is pretty good.

Searching Versus Browsing

There are two different kinds of shoppers, and they illustrate the difference between *searching* and *browsing*. Some shoppers know exactly what they're after, and they want to find a store with the item, locate the item, and purchase it as quickly as possible. As with a web search, it helps to know a bit about what you're looking for if this is your style.

Other shoppers want to explore a particular store, see what the store offers, and choose an item if the right one comes along. This style of browsing is suited for people who want to get a larger survey of items in a particular category before they necessarily know what they're looking for.

If you were interested in looking at sites about child psychology, you might try a search at http://search.google.com with the query `child psychology`. You would find thousands of sites in the search results, along with news articles about child psychology, college papers about the topic, and even pages that mention the terms *child* and *psychology* without relating to the topic. But browsing the Child Psychology category in the Google Directory (http://directory.google.com/Top/Science/Social_Sciences/Psychology/Child_Psychology/) gives you hundreds of links selected by Open Directory volunteers as being relevant to the topic.

There are still times when you need to search the directory, and Google has provided a couple ways to accomplish this.

Searching the Google Directory

Because the Google Directory is a far smaller collection of URLs, ideal for more general searching, it does not have the various complicated special syntaxes for searching that the Web Search does. However, there are a couple of special syntaxes that you should know about:

`intitle:`

Just like the Google web special syntax, `intitle:` restricts the query word search to the title of a page.

`inurl:`

`inurl:` restricts the query word search to the URL of a page.

When you're searching on Google's web index, your overwhelming concern is probably how to reduce your list of search results to something manageable. With that in mind, you might start with the narrowest possible search.

That's a reasonable strategy for the web index, but because you have a narrower pool of sites in the Google Directory, you want that search to be more general.

For example, say you were looking for information on author P. G. Wodehouse. A simple search on `P. G. Wodehouse` in Google's web index gets you over 1,100,000 results, possibly compelling you to immediately narrow your search. But doing the same search in the Google Directory returns only 176 results. You might consider that a manageable number of results, or you might want to carefully narrow your results further.

The Directory is also good for searching for events. A Google web search for `Korean War` will find over 24 million results, while searching the Google Directory will find just over 138,000. This is a case where you will probably need to narrow your search. Use general words indicating what kind of information you want `timeline`, for example, or `archives`, or `lesson plans`. Don't narrow your search with names or locations; that's not the best way to use the Google Directory.

Hack 2. Glean a Snapshot of Google in Time

Google Zeitgeist provides a weekly, monthly, and yearly overview of what the Web was interested in.

Turning to Google itself for a definition of*zeitgeist* (`define:zeitgeist`), there's consensus that it refers to "the spirit of the times." And Google Zeitgeist (http://www.google.com/press/zeitgeist.html) is just that: a mirror that the Web (according to Google) holds up to us, providing a snapshot of the week, month, or year that was.

A typical weekly Google Zeitgeist, shown inFigure 1-8, lists the top 15 gaining queries.

Figure 1-8. The week's top 15 gaining queries

It takes only a few moments of visiting Google Zeitgeist before you're itching to go back a little further in time: the week your second child was born, the month during which the Olympics were held, the year you graduated from high school. Click the Archive link to choose any year from the Google Zeitgeist Archive and display links such as those shown inFigure 1-9 for every week, month, and year since January 2001.

Weekly Zeitgeist updates actually started in June 2001, at the same time the monthlies switched from PDF to HTML format. In August 2005, Google stopped listing declining queries and started listing 5 more of the top gaining queries, bringing the total to 15.

Figure 1-9. The Zeitgest Archive pages, displaying weekly, monthly, and year-end reports dating back to 2001

Monthly reports provide some information about Google News queries and Google Image Search queries, and you can find monthly reports for countries around the world by clicking the Zeitgeist Around the World link on the front page. Year-end reports provide even more detail with trend graphs.

While Google Zeitgeist's statistics aren't earth-shattering (e.g., searches for`iraq` more than doubled on March 19, 2003, the date that Operation Iraqi Freedom beganimagine that!), it does provide a snapshot of what the world in aggregate found interesting enough to look up.

See Also

- If Google Zeitgeist piques your interest, you might also try theYahoo! Buzz Index (http://buzz.yahoo.com), a similar collection of statistics around popular Yahoo! Searches: the day's top movers (overall and by various Yahoo! categories), most viewed and emailed Yahoo!

news items, and a market trendlike chart (click the View Complete Chart... link associated with any of the buzz listings on the front page) of leaders and movers, according to*buzz score* (http://help.yahoo.com/help/us/buzz/#buzz-04).

- Google Trends (http://www.google.com/trends) is a new product from the Google Labs that graphs the mentions of words of phrases over time. Type in two words separated by commas tc get a quick visual sense of the popularity. For example, "Google, Yahoo" shows you which search engine is mentioned more across time, regions, news stories, and languages.

PREV

Hack 3. Visualize Google Results

The TouchGraph Google Browser is the perfect Google complement for those who appreciate visual displays of information.

Some people are born text crawlers. They can retrieve the mostly text resources of the Internet and browse them happily for hours. But others are more visually oriented and find that the flat text results of the Internet leave something to be desired, especially when it comes to search results.

If you're the type that appreciates visual displays of information, you're bound to like the TouchGraph Google Browser (http://www.touchgraph.com/TGGoogleBrowser.html). This Java applet allows you to start with pages that are similar to one URL, and then expand outward to pages that are similar to the first set of pages, on and on, until you have a giant map of *nodes* (a.k.a. URLs) on your screen.

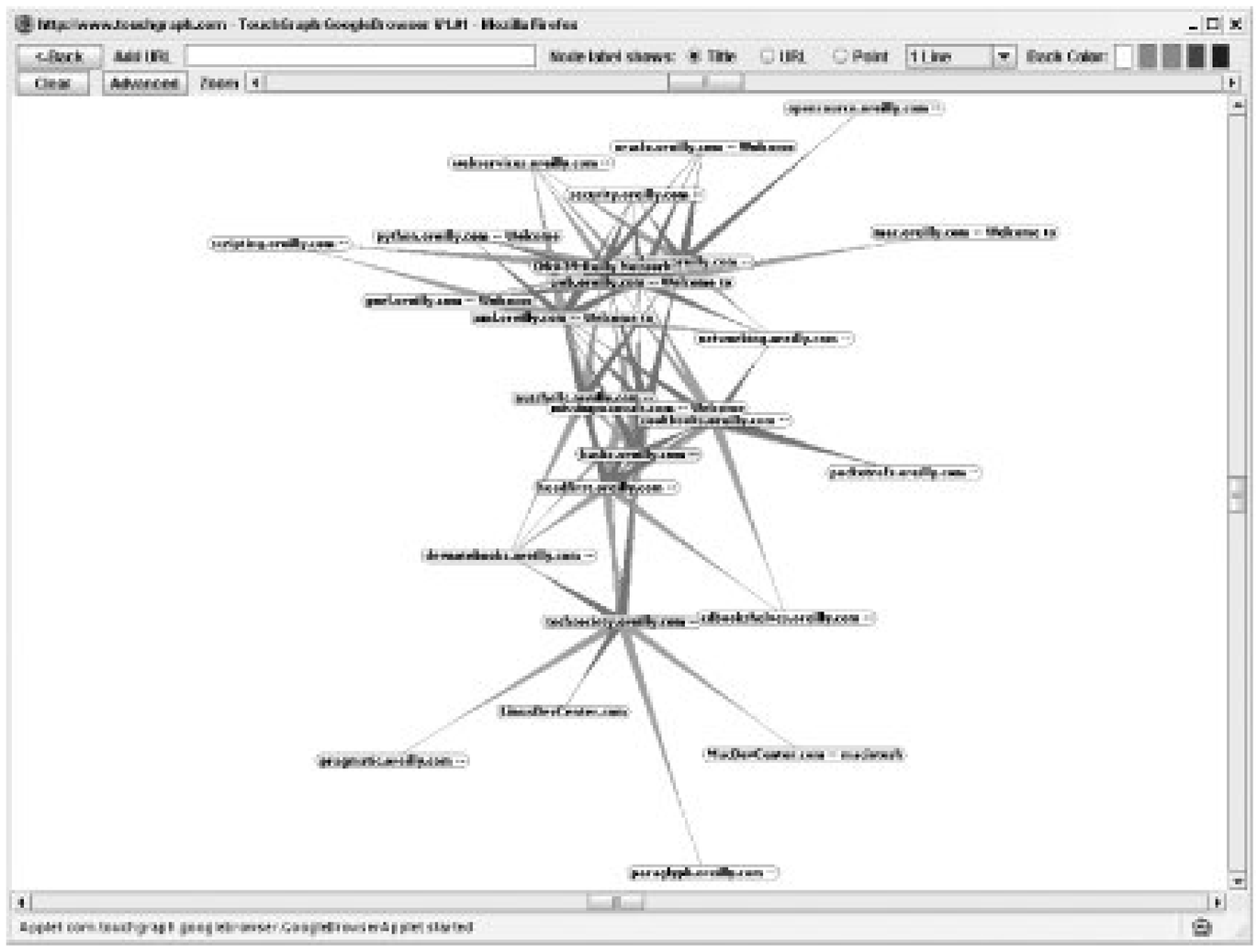
> The TouchGraph Google Browser was created by Alex Shapiro (http://www.touchgraph.com/).

Note that you're finding URLs that are similar to another URL, just as you would if you used the `related:` syntax. You aren't doing a keyword search, and you're not using the `link:` syntax. You're searching by Google's measure of similarity.

Starting to Browse

Start your journey by entering a URL on the TouchGraph home page and clicking the Graph It link. Your browser will launch the TouchGraph Java applet, covering your window with a large mass of linked nodes, as shown in Figure 1-10.

Figure 1-10. Mass of linked nodes generated by TouchGraph



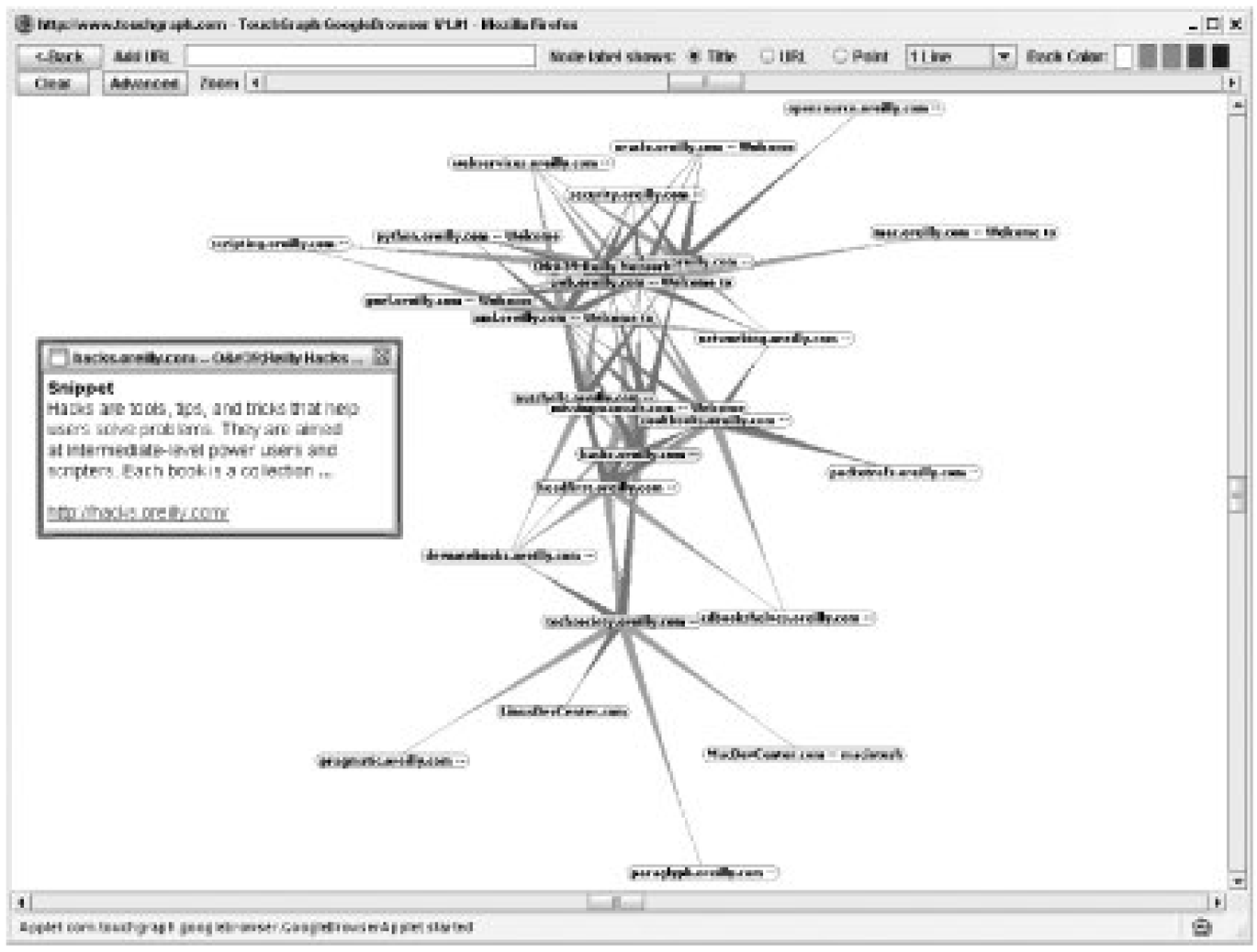
You'll need a web browser capable of running Java applets. If Java support in your preferred browser comes in the form of a plug-in, your browser should have the smarts to launch a plug-in locator/downloader and walk you through the installation process.

If you're easily entertained like me, you might amuse yourself for a while just by clicking and dragging the nodes around. But there's more to do than that.

Expanding Your View

Hold your mouse over one of the items in the group of pages. A little box labeled*info* pops up. Click on that, and a box of information about that particular node appears, as shown inFigure 1-11.

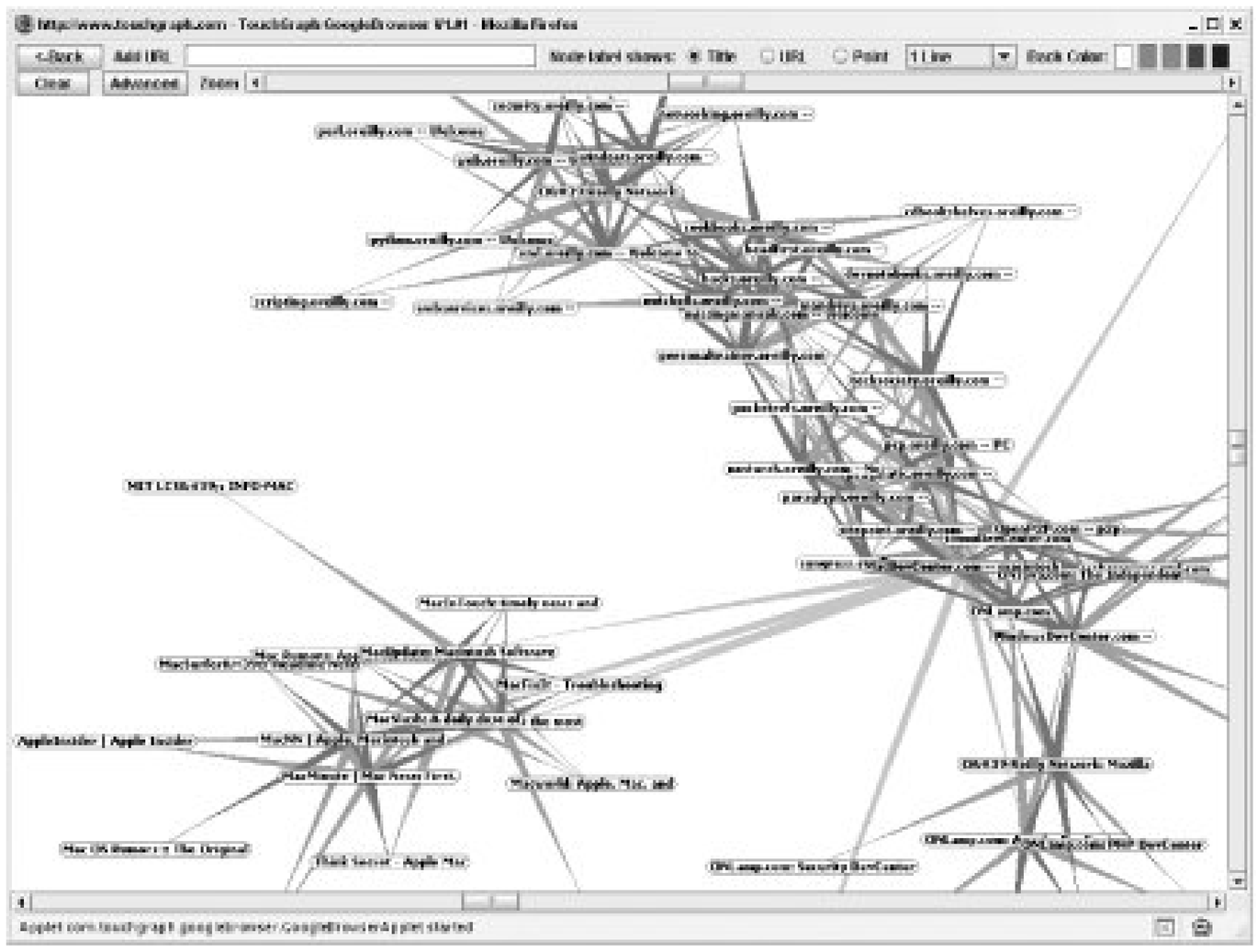
Figure 1-11. Node information pop-up box



The box of information contains title, snippet, and URLpretty much everything you'd get from a regular search result. Click on the URL in the box to open that URL's web page itself in another browser window. If your browser is set to block pop-up windows, you might need to enable them from the touchgraph.com domain.

Not interested in visiting web pages just yet? Want to do some more search visualization? Double-click on one of the nodes. TouchGraph uses the API to request from Google pages similar to the URL of the node you double-clicked. Keep double-clicking at will; when no more pages are available, a green C will appear when you put your mouse over the node (no more than 30 results are available for each node). If you do this often enough, you'll end up with a screen full of nodes with lines denoting their relationship to one another, as Figure 1-12 shows.

Figure 1-12. Node mass expanded by double-clicking on nodes



Visualization Options

Once you've generated similarity page listings for a few different sites, you'll find yourself with a pretty crowded page. TouchGraph has a few options to change the look of what you're viewing.

For each node, you can show page title, page URL, or *point* (the first two letters of the title). If you're just browsing page relationships, the title is probably best. However, if you've been working with the applet for a while and have mapped out a plethora of nodes, the point or URL options can save some space. The URL option removes the *www* and *.com* from the URL, leaving the other domain suffixes. For example, *www.perl.com* shows as *perl*, while *www.perl.org* shows as *perl.org*.

Speaking of saving space, there's a zoom slider at the upper right of the applet window. After you've generated several distinct groups of nodes, zooming out allows you to see the different groupings more clearly. However, it becomes difficult to see relationships between the nodes in the different groups.

To customize the display even further, click the Advanced button to see more TouchGraph options. You'll find the option to view the *singles*: the nodes in a group that have a relationship with only one other node. This option is off by default; check the Show Singles checkbox to turn it on. I find it's better to leave out singles; they crowd the page and make it difficult to establish and explore separate groups of nodes.

The Radius setting specifies how many nodes will be displayed around the node you've clicked. A radius of 1 will show all nodes directly linked to the node you've clicked, a radius of 2 will show all nodes directly linked to the node you've clicked as well as all nodes directly linked to those nodes, and so on. The higher the radius, the more crowded things get. The groupings do, however, tend to settle themselves into nice little discernable clumps. A drop-down menu beside the Radius setting specifies how many search results (i.e., how many connections) are shown. A setting of 10 is, in my experience, optimal.

For a look at all the ways you can customize the TouchGraph Google browser, be sure to check out the Full Instructions page at http://www.touchgraph.com/TGGB_FullInstructions.html.

Making the Most of These Visualizations

Yes, it's cool. Yes, it's unusual. And yes, it's fun dragging those little nodes around. But what exactly is the TouchGraph good for?

TouchGraph does two rather useful things. First, it allows you to see at a glance the similarity relationship between large groups of URLs. You can't do this with several flat results to similar URL queries. Second, if you do some exploring, you can sometimes get a list of companies in the same industry or area. This comes in handy when you're researching a particular industry or topic.It'll take time, though, so keep trying.

PREV

Hack 4. Check Your Spelling

Google sometimes takes the liberty of "correcting" what it perceives to be a spelling error in your query.

Most of us couldn't communicate with the outside world without a spellchecker. As you send off an email or put the finishing touches on a document, a trustyspellchecker makes sure you haven't made any blatant errors. Google also has a built-in spellchecker, and when Google thinks it can spell individual words or complete phrases in your search query better than you can, it suggests a "better" search, hyperlinking it directly to a query.

For example, if you search for `hydrecefallus`, Google will ask if you meant `hydrocephalus`, as shown in Figure 1-13.

Figure 1-13. Offering spelling suggestions when Google thinks it knows better

Suggestions aside, Google assumes that you know of what you speak and returns your requested results, provided your query gleaned results.

If your query found no results for the spellings you provided and Google believes it knows better, it will automatically run a new search of its own suggestions. Thus, a search for `hydrecefallus` finding (hopefully) no results sparks a Google-initiated search for `hydrocephalus`.

Mind you, Google does not arbitrarily come up with its suggestions, but builds them based on its own database of words and phrases found while indexing the Web. If you search for nonsense like `kweghgjdlsggaa`, you'll get no results and be offered no suggestions.

This is a lovely side effect and a quick and easy way to check the relative frequency of spellings. Query for a particular spelling, and note the number of results. Then click on Google's suggested spelling and note the number of results. It's surprising how close the counts are sometimes, indicating an oft-misspelled word or phrase.

If you find yourself turning to Google to compare spellings, you might want to automate the process of comparing phrases [Hack #26].

Embrace Misspellings

Don't make the mistake of automatically dismissing the proffered results from a misspelled word, particularly a proper name. I've been a fan of cartoonist Bill Mauldin for years now, but I repeatedly misspell his name as "Bill Maudlin." And judging from a quick Google search, I'm not the only one. There is no law stating that every page must be spellchecked before it goes online, so it's often worth taking a look at results despite misspellings.

As an experiment, try searching for two misspelled words on a related topic, such as `normotensive hydrocephalis`. What kind of information did you get? Could the information you got, if any, be grouped into a particular online *genre*?

At the time of this writing, the search for `normotensive hydrocephalis` gets only three results. The content here is generally from people dealing with various neurosurgical problems. Again, there is no law that states that all web materials have to be spellchecked.

Use this to your advantage as a researcher. When you're looking for layman accounts of illness and injury, the content you desire might actually be more often misspelled than not. On the other hand, when looking for highly technical information or references from credible sources, filtering out misspelled queries will bring you closer to the information you seek.

Spelling on the Command Line

The fact that Google gathers its spellings from across the Web instead of a dictionary means it can out-spell most email and word-processor spellcheckers. An email spellchecker won't catch that you've just misspelled the name of comedian Dave Shapel (or is it Dave*Chapelle*?), while Google's spellchecker will catch the error.

While this hack won't replace your standard spellcheckers with Google, the code in this section will show you how to bring the spellchecker a bit closer to your desktop.

The code

This code contacts the Google API and asks for a spelling suggestion for the supplied word or phrase. If you're not already accustomed to using the command line to get things done, this hack probably won't make contacting Google any easier than opening a web browser. But for command-line junkies it's a quick way to tap the power of Google spelling.

Save the following code as *spell.pl*, and be sure to replace *insert your key* with your own Google API key:

```
#!/usr/local/bin/perl
# spell.pl
# Contact Google for spelling suggestions!
# Usage: perl spell.pl <query>

# Your Google API developer's key.
my $google_key='insert your key';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

use strict;

# Use the SOAP::Lite Perl module.
use SOAP::Lite;

# Take the query from the command line.
my $query = join(' ',@ARGV) or die "Usage: perl spell.pl <query>\\n";

# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
my $google_search = SOAP::Lite->service("file:$google_wsdl");

# Query Google.
my $results = $google_search ->
    doSpellingSuggestion($google_key, $query);

# No results?
if ($results) {
    print $results;
}
```

This script is similar to any bare-bones Perl script [Hack #90] for contacting the Google API, but it uses the `doSpellingSuggestion` method instead of the standard search method.

Running the hack

Run the script from the command line, passing in any word or phrase you want to check, like this:

```
% perl spell.pl

                    insert word or phrase
```

By passing in `Dave Shapel`, you can see how Google suggests you spell his name:

```
% perl spell.pl Dave Shapel
Dave Chapelle
```

If you pass in a correct spelling, the script simply returns no suggestions at all.

You still need to figure out which words are questionable to use this script, but when you need to double-check a name or phrase quickly, you can think of Google as your own personal lexiconographer (or is that *lexicographer*?).

PREV

Hack 5. Google Phonebook: Let Google's Fingers Do the Walking

Google makes an excellent phonebook, even to the extent of doing reverse lookups.

Google combines residential and business phone number information and its own excellent interface to offer a phonebook lookup that provides listings for businesses and residences in the United States. However, the search offers three different syntaxes, different levels of information provide different results, the syntaxes are finicky, and Google doesn't provide documentation.

The Three Syntaxes

Google offers three ways to search its phonebook:

`phonebook`

: Searches the entire Google phonebook

`rphonebook`

: Searches residential listings only

`bphonebook`

: Searches business listings only

> The result page for `phonebook:` lookups lists only five results for both residential and business numbers. The more specific `rphonebook:` and `bphonebook:` searches provide up to 30 results per page. For a better chance of finding what you're looking for, use the appropriate targeted lookup.

Using the Syntaxes

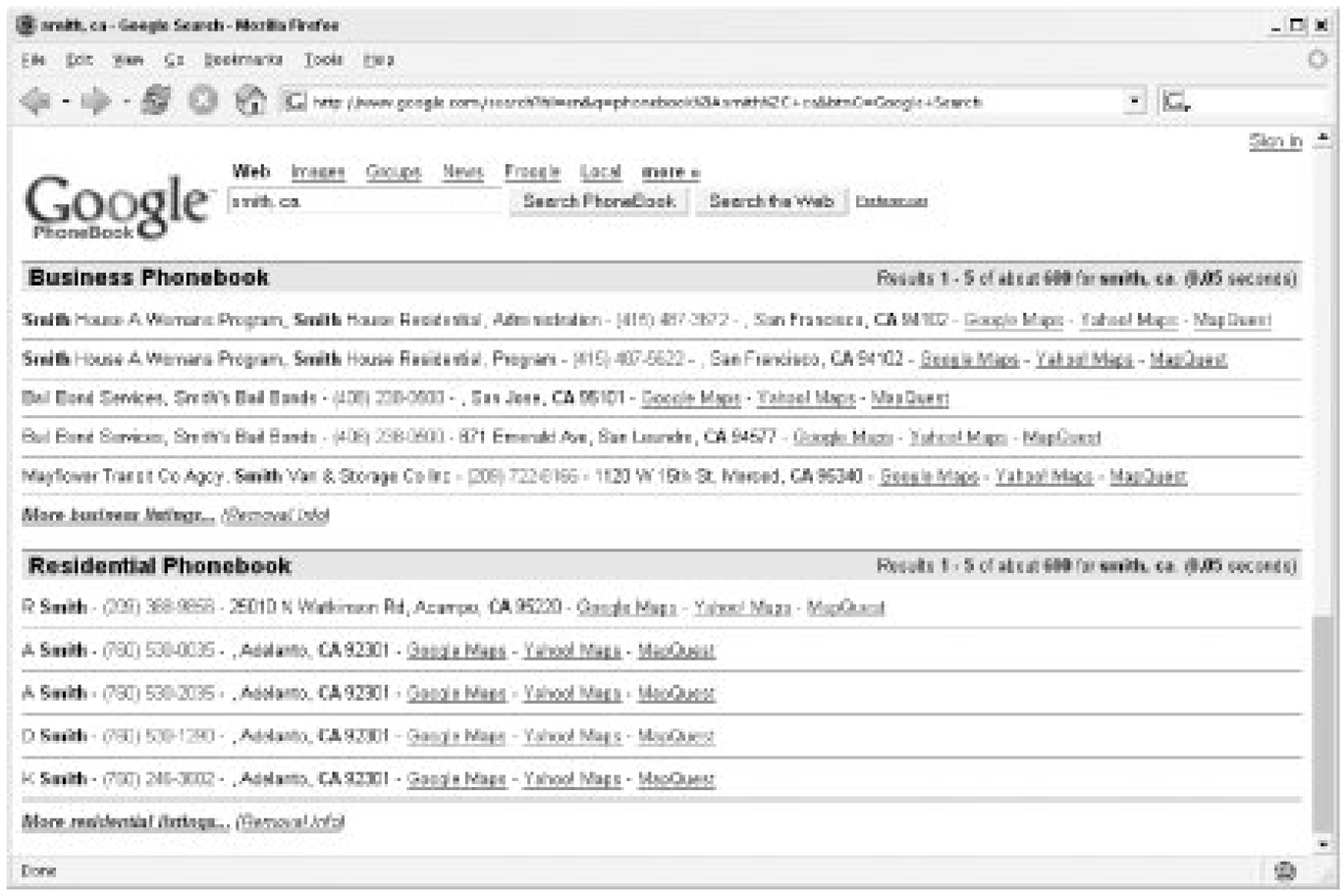
Using a standard phonebook requires knowing quite a bit of information about what you're looking for: first name, last name, city, and state. Google's phonebook requires no more than last name and

state to get started. Casting a wide net for all the Smiths in California is as simple as:

```
phonebook:smith ca
```

Try giving 411 a whirl with that request! Figure 1-14 shows the results of the query.

Figure 1-14. Results of a phonebook: query



Notice that while intuition might tell you that there are thousands of Smiths in California, the Google phonebook says that there are only 600. Just as Google's regular search engine maxes out near 1,000 results, its phonebook maxes out at 600. Fair enough. Try narrowing your search by adding a first name, city, or both:

```
phonebook:john smith los angeles ca
```

At the time of this writing, the Google phonebook found 2 business and 20 residential listings for John Smith in Los Angeles, California.

Caveats

The phonebook syntaxes are powerful and useful, but they can be difficult to use if you don't remember a few things about how they work.

Syntaxes are case-sensitive

Searching for `phonebook:john doe ca` works, while `Phonebook:john doe ca` (notice the capital P) doesn't.

Wildcards don't work

Then again, they're not needed, since the Google phonebook does all the wildcarding for you. For example, if you want to find shops in New York with "Coffee" in the title, don't bother trying to envision every permutation of "Coffee Shop," "Coffee House," and so on. Just search for `bphonebook:coffee new york ny` and you'll get a list of all businesses in New York whose names contain the word "coffee."

Exclusions don't work

Perhaps you want to find coffee shops that aren't Starbucks. You might think `phonebook:coffee -starbucks new york ny` would do the trick. After all, you're searching for coffee and not Starbucks, right? Unfortunately not; Google thinks you're looking for both the words "coffee" and "starbucks," yielding just the opposite of what you were hoping for: everything Starbucks in NYC.

OR doesn't always work

You might be wondering if Google's phonebook accepts `OR` lookups. You then might experiment, trying to find all the coffee shops in Rhode Island or Hawaii: `bphonebook:coffee (ri | hi)`. Unfortunately, that doesn't work; the only listings you'll get are for coffee shops in Hawaii. This is because Google doesn't see the `(ri | hi)` as a state code, but rather as another element of the search.

So, if you reverse the previous search and search for `coffee (hi | ri)`, Google would find listings that contain the word "coffee" and either the strings "hi" or "ri." This means you'll find Hi-Tide Coffee (in Massachusetts) and several coffee shops in Rhode Island.

It's neater to use `OR` in the middle of your query and specify a state at the end. For example, if you want to find coffee shops that sell either donuts or bagels, this query works fine: `bphonebook:coffee (donuts | bagels) ma`. It finds stores in Massachusetts that contain the word "coffee" and either the word "donuts" or the word "bagels." The bottom line: you can use an `OR` query on the store or resident name, but not on the location.

> Try some phonebook lookups that you can't do by dialing 411. For example, try searching by last name and area code, or last name and zip code! Google's phonebook lookup is very accommodating.

Reverse Phonebook Lookup

All three phonebook syntaxes support reverse lookup, though it's probably best to use the general `phonebook:` syntax to avoid not finding what you're looking for due to a residential or business classification.

To do a reverse search, just enter the phone number with area code. Lookups without area code won't work:

```
phonebook:(707) 827-7000
```

(This is the phone number of O'Reilly world headquarters in Sebastopol, California, USA.)

Keep in mind that Google's phonebook service doesn't include cell phone numbers.

Reverse lookups on Google are a hit-or-miss proposition and don't always produce results. If you're not having any luck, consider using a more dedicatedphonebook site such as WhitePages.com (http://www.whitepages.com).

PREV

Hack 6. Look Up Definitions

Do you find yourself smiling knowingly when your boss mentions that well-known business principle you've never heard of? Overwhelmed with "geek speak"? Chances are Google's heard it mentionedand possibly even definedsomewhere before.

Most specialized vocabularies remain, for the most part, fairly static; words don't suddenly change their meaning all that often. Not so with technical and computer-related jargon. It seems like every 12 seconds someone comes up with a new buzzword or term relating to computers or the Internet, and then 12 minutes later it becomes obsolete or means something completely differentoften more than one thing at a time. Maybe it's not that bad. It just feels that way.

Google can help you in two ways: by helping you look up words and by helping you figure out what words you don't know but need to know.

Google Definitions

Before you assume you're going to be in for a lot of Googling, try the`define` search syntax mentioned in the "Quick Links" section earlier in this chapter. Simply prepend the definition you're after with the special syntax keyword `define`, like so:

```
define google juice
define julienne
define 42
```

Google tells you that these are defined as "power of a website to turn up in Google," "cut food into thin sticks," and "being two more than forty," thanks to Wikipedia, Low Carb Luxury, and WordNet at Princeton, respectively.

Click the associated "Definition in context" link to visit the page from which the definition was drawn.

Click the "Web definitions for..." link or prefix the word you're defining with`define:` (note the addition of a colon) in the first place, and you'll net a full page of definitions drawn from all manner of places. For instance, `define:TLA` finds turns up oodles of definitions (all about the same, mind you), as shown in Figure 1-15.

Figure 1-15. A page chock-full of definitions for TLA

> The `define word` syntax is still subject to spelling suggestions, so you don't have to worry too much about misspelling. The `define:word` form, however, doesn't perform a web search at all, so it returns no results or spelling suggestions whatsoever if it finds no definitions to offer you.

If all that didn't turn up anything useful, move on to Google Web Search proper.

Slang

We have distinctive speech patterns that are shaped by our educations, our families, and our location. Further, we may use another set of words based on our occupation. When a teenager says something is "phat," that's *slang*a specialized vocabulary used by a particular group. When a copywriter scribbles "stet" on an ad, that's not slang, but it's still specialized vocabulary or jargon used by a certain groupin this case, the advertising industry.

Being aware of these specialty words can make all the difference when it comes to searching. Adding specialized words to your search querywhether slang or industry jargoncan really change the slant of your search results.

Slang gives you one more way to break up your search engine results into geographically distinct areas. There's some geographical blurriness when you use slang to narrow your search engine results, but it's amazing how well it works. For example, search Google for`football`. Now search for `football bloke`. Totally different result sets, aren't they? Search for`football bloke bonce`. Now you're into soccer narratives.

Of course, this is not to say that everyone in England automatically uses the word "bloke" any more

than everyone in the southern U.S. automatically uses the word "y'all." But adding well-chosen bits o slang (which will take some experimentation) gives your search results a whole different tenor and may point you in unexpected directions. You can find slang from the following resources:

The Probert EncyclopediaSlang (http://www.probertencyclopaedia.com/slang.htm)

This site is browseable by first letter orsearchable by keyword. (Note that the keyword search covers the entire *Probert Encyclopedia* ; slang results are near the bottom.) The slang presented here is from all over the world. It's often cross-linked, especially drug slang. As with most slang dictionaries, this site contains material that might offend.

A Dictionary of Slang (http://www.peevish.co.uk/slang/)

This site focuses on slang heard in the United Kingdom, which means slang from other places as well. It's browseable by letter or via a search engine. Words from outside the UK are marked with their place of origin in brackets. Definitions also indicate typical usage: humorous, vulgar, derogatory, etc.

Surfing for Slang (http://www.spraakservice.net/slangportal)

Of course, each area in the world has its own slang. This site has a good metalist of English an Scandinavian slang resources.

Urban Dictionary (http://www.urbandictionary.com)

You can browse this collaborative dictionary by word and find dozens or hundreds of definitions for each word. The definitions are added by site visitors, and each definition is open to votes from other visitors. The most widely accepted definitions for each word bubble up to the top.

Start by searching Google for your query without theslang. Check the results and decide where they're falling short. Are they not specific enough? Are they not located in the right geographical area? Are they not covering the right demographicteenagers, for example?

Introduce one slang word at a time. For example, in a search for`football`, add the word `bonce` and check the results. If they're not narrow enough, add the word `bloke`. Add one word at a time until you get the results you want. Using slang is an inexact science, so you have to do some experimenting.

Here are some things to be careful of when using slang in your searches:

- Try many different slang words.
- Don't use slang words that are generally considered offensive, except as a last resort. Your results will be skewed.
- Be careful when using teenage slang, which changes constantly.
- Try searching for slang when using Google Groups. Slang crops up often in conversation.

- Minimize your searches for slang when searching for more formal sources, such as newspaper stories.
- Don't use slang phrases if you can help it; in my experience, slang changes too much to be consistently searchable. Stick to established words.

Industrial Slang

Specialized vocabularies are those used in particular subject areas and industries. Good examples of specialized vocabularies are used in the medical and legal fields, although there are many others.

When you need to tip your search to the more technical, the more specialized, and the more in-depth, think of a specialized vocabulary. For example, do a Google search for`heartburn`. Now do a search for `heartburn GERD`. Now do a search for `heartburn GERD gastric acid`. You'll see that each is very different.

With some fields, finding specialized-vocabulary resources is a snap. But with others, it's not that easy. As a jumping-off point, try the Glossarist site at http://www.glossarist.com, which is a searchable subject index of about 6,000 different glossaries covering dozens of different topics. There are also several other large online resources covering certain specialized vocabularies. These resources include:

The On-Line Medical Dictionary (http://cancerweb.ncl.ac.uk/omd/)

> This dictionary contains vocabulary relating to biochemistry, cell biology, chemistry, medicine, molecular biology, physics, plant biology, radiobiology, and other sciences and technologies. It currently has over 46,000 listings.
>
> You can browse the dictionary by letter or search it by word. Sometimes you can search for a word that you know (`bruise`) and find another term that might be more common in medical terminology (`contusion`). You can also browse the dictionary by subject. Bear in mind that this dictionary is in the UK, and some spellings may be slightly different for American users (e.g., "tumour" versus "tumor").

MedTerms.com (http://www.medterms.com)

> MedTerms.com has far fewer definitions (around 15,000), but it also has extensive articles from MedicineNet. If you're starting from absolute square one with your research and need some basic information and vocabulary to get started, search MedicineNet for your term (`bruise` works well) and then move to MedTerms.com to search for specific words.

Law.com's legal dictionary (http://dictionary.law.com/lookup2.asp)

> Law.com's legal dictionary is excellent because you can search either words or definitions; you can browse, too. For example, you can search definitions for the word`inheritance` and get a list of all the entries that contain the word "inheritance." This is an easy way to get to the words "muniment of title" without knowing the path.

As with slang, add specialized vocabulary slowlyone word at a timeand anticipate that your search results will be narrowed very quickly. For example, take the word "spudding," often used in association with oil drilling. Searching for `spudding` by itself finds about 33,900 results on Google. Adding `Texas` knocks it down to 852 results, and this is still a very general search! Add specialized vocabulary very carefully, or you'll narrow your search results to the point where you can't find what you want.

Researching Terminology with Google

First things first: for heaven's sake, please don't just plug the abbreviation into the query box! For example, searching for `XSLT` will net you over 29 million results. While combing through the sites that Google turns up may eventually lead you to a definition, there's simply more to life than that. Instead, add `"stands +for"` to the query if it's an abbreviation or acronym. `"XSLT stands +for"` returns around 199,000 results, and the first is a tutorial glossary. If you're still getting too many results (`"XML stands +for"` gives you around six million results), try adding `beginners` or `newbie` to the query. `"XML stands +for" beginners` brings in 463 results, the fourth being a general, gentle "Introduction to XML."

If you're still not getting the results you want, try `"What is X?"` or `" X +is short +for"` or `" X beginners FAQ"`, where *X* is the acronym or term. These should be regarded as second-tier methods, because most sites don't tend to use phrases such as "What is X?" on their pages, "X is short for" is uncommon language usage, and X might be so new (or so obscure) that it doesn't yet have a FAQ entry. Then again, your mileage may vary, and it's worth a shot; there's a lot of terminology out there.

If you have hardware- or software-specific, as opposed to hardware- or software-related, terminology, try the word or phrase along with anything you might know about its usage. For example, as a Perl module, DynaLoader is software-specific terminology. That much known, simply give the two words a spin:

```
DynaLoader Perl
```

If the results are too advanced, assuming you already know what a DynaLoader is, start playing with the words `beginners`, `newbie`, and the like to bring you closer to information for beginners:

```
DynaLoader Perl Beginners
```

If you still can't find the word in Google, there are a few possible causes: perhaps it's slang specific t your area, your coworkers are playing with your mind, you heard it wrong (or there's a typo on the printout you got), or it's very, very new.

Where to Go When It's Not on Google

Despite your best efforts, you're not finding good explanations of the terminology on Google. There are a few other sites that might have what you're looking for:

Whatis (http://whatis.techtarget.com)

A searchable subject index of computer terminology, from software to telecom. This is especially useful if you have a hardware- or software-specific word because the definitions are divided into categories. You can also browse alphabetically. Annotations are good and are ofter cross-indexed.

Webopedia (http://www.pcwebopaedia.com)

Searchable by keyword or browsable by category. This site also has a list of the newest entries on the front page so that you can check for new words.

Netlingo (http://www.netlingo.com)

This site is more Internet-oriented. It shows up with a frame on the left that contains the words, with the definitions on the right. It includes lots of cross-referencing and really old slang.

Tech Encyclopedia (http://www.techweb.com/encyclopedia/)

Features definitions and information for over 20,000 words.The top 10 terms searched for are listed so you can see if everyone else is as confused as you are. Though entries had before-the-listing and after-the-listing lists of words, I saw only moderate cross-referencing.

Wikipedia (http://www.wikipedia.com)

This public encyclopedia that anyone can edit is surprisingly accurate and up to date with technology slang. Because new entries don't need to be approved by one or two editors, and because the work of editing is done by thousands of volunteers across disciplines and industries, Wikipedia is constantly evolving with the times.

Geek terminology proliferates almost as quickly as web pages. Don't worry too much about deliberately keeping up; it's just about impossible. Instead, use Google as a "ready reference" resource for definitions.

Hack 7. Find Directories of Information

Use Google to find directories, link lists, and other collections of information.

Sometimes you're more interested in large information collections than scouring for specific bits and bobs. You could always take a stroll through the Google Directory (http://directory.google.com) to see what's available, but sometimes a topic-specific directory is what you need.

Using Google, there are a couple of different ways to find directories, link lists, and other information collections from across the Web. The first uses Google's full-word wildcards ["Full-Word Wildcards" earlier in this chapter] and the `intitle:` syntax ["Special Syntax" earlier in this chapter]. The second is a judicious use of particular keywords.

Title Tags and Wildcards

Pick something you'd like to find collections of information about. We'll use "trees" as our example. The first thing we look for is any page with the words "directory" and "trees" in its title. In fact, we build in a little buffering for words that might appear between the two using a couple of full-word wildcards (`*` characters). The resultant query looks something like this:

```
intitle:"directory * * trees"
```

This query finds "directories of evergreen trees," "South African trees," and of course "directories containing simply trees."

What if you want to take things up a notch, taxonomically speaking, and find directories of botanical information? Use a combination of `intitle:` and keywords, like so:

```
botany intitle:"directory of"
```

and you get almost 10,000 results. Changing the tenor of the information might be a matter of restricting results to those coming from academic institutions. Appending an `edu` site specification brings you to:

```
botany intitle:"directory of" site:edu
```

This gets you around 150 results, a mixture of resource directories, and, unsurprisingly, directories o university professors.

Mixing these syntaxes works rather well when searching for something that might also be an offline print resource. For example:

```
cars intitle:"encyclopedia of"
```

This query pulls in results from Amazon.com and other sites that sell car encyclopedias. Filter out some of the more obvious book finds by tweaking the query slightly:

```
cars intitle:"encyclopedia of" -site:amazon.com
-inurl:book -inurl:products
```

The query specifies that search results should not come from Amazon.com and should not have the word "products" or "book" in the URL, which eliminates a fair amount of online stores. For some interesting finds, play with this query by changing the word "cars" to whatever you like.

Of course, there are many sites that sell books online, but when it comes to injecting "noise" into results when you're trying to find online resources and research-oriented information, Amazon.com is the biggest offender. If you're actually looking for books, try `+site:amazon.com` instead.

If mixing syntaxes doesn't find the resources you want, there are some clever keyword combinations that might just do the trick.

Finding Searchable Subject Indexes with Google

There are a few major searchable subject indexes and myriad minor ones that deal with a particular topic or idea. You can find the smaller subject indexes by customizing a few generic searches. `"what's new" "what's cool" directory`, while gleaning a few false results, is a great way to find searchable subject indexes.

`directory "gossamer threads" new` is an interesting one. Gossamer Threads is the creator of a popular link directory program. This is a good way to find searchable subject indexes without too many false hits.

`directory "what's new" categories cool` doesn't work particularly well, because the word "directory" is not a very reliable search term, but you will pull in some things with this query that you might otherwise have missed.

Let's put a few of these into practice:

```
"what's new" "what's cool" directory phylum
"what's new" "what's cool" directory carburetor
"what's new" "what's cool" directory "investigative journalism"
"what's new" directory categories gardening
directory "gossamer threads" new sailboats
directory "what's new" categories cool "basset hounds"
```

The real trick is to use a more general word, but make it unique enough that it applies mostly to you topic and not to many other topics.

Take acupuncture, for instance. Start narrowing it down by topic. What kind of acupuncture? For people or animals? If for people, what kinds of conditions are being treated? If for animals, what kinds of animals? Maybe you should search for `"cat acupuncture"`, or maybe you should search for `acupuncture arthritis`. If this first round doesn't narrow the search results enough, keep going. Are you looking for education or treatment? You can skew results one way or the other using the`site:` syntax. So maybe you want `"cat acupuncture" site:com` or `arthritis acupuncture site:edu`. By taking just a few steps to narrow things down, you can get a reasonable number of search results focused around your topic.

PREV

Hack 8. Cover Your Bases

Try all possible combinations of your search keywords at once, and find related keywords with Google Sets.

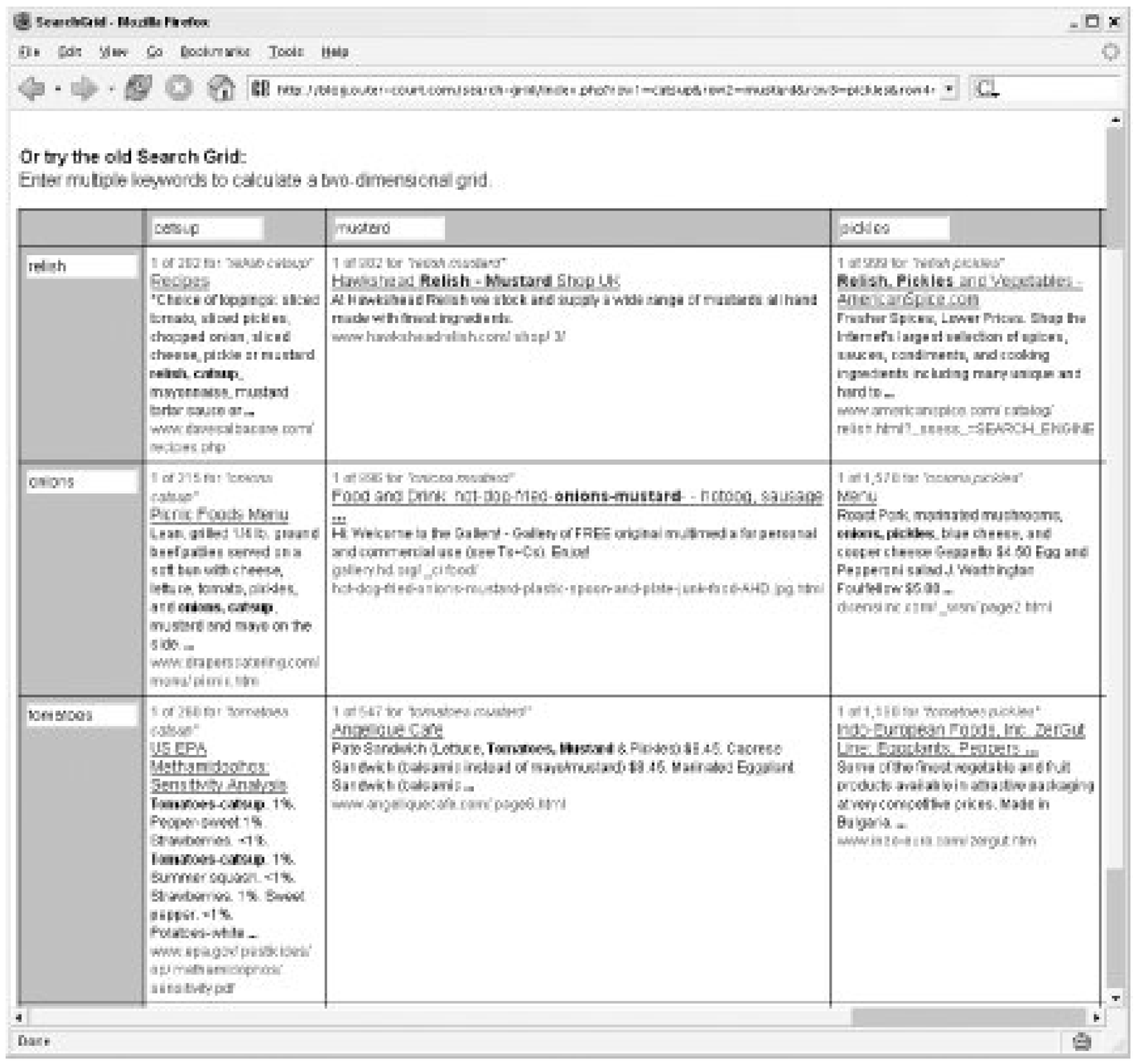
Imagine you have a set of query words but are not sure that they're the right set; you certainly don't want to miss any results by picking the wrong combination of keywords, including or excluding the wrong word. But the thought of typing a dozen-plus permutations of keywords has your carpal tunne flaring up in horror. With some existing tools, you can fine-tune your Google queries by playing with word setsleading you down paths you might not have discovered.

Search Grid (http://blog.outer-court.com/search-grid), by German programmer Philipp Lenssen, lets you explore a wide range of Google search results by automatically searching for multiple combinations of keywords you specify. This gives you a quick overview of paths you can follow for a given set of keywords. You might, for example, put `catsup`, `mustard`, and `pickles` on the x-axis and `relish`, `onions`, and `tomatoes` on the y-axis, as shown in Figure 1-16.

Figure 1-16. Search Grid populated with keywords to combine

Search Grid combines the results*relish catsup*, *relish mustard*, *relish pickles*, *onions catsup*, *onions mustard*, *onions pickles*, etc.and provides you with the first result of each possible combination, shown in Figure 1-17.

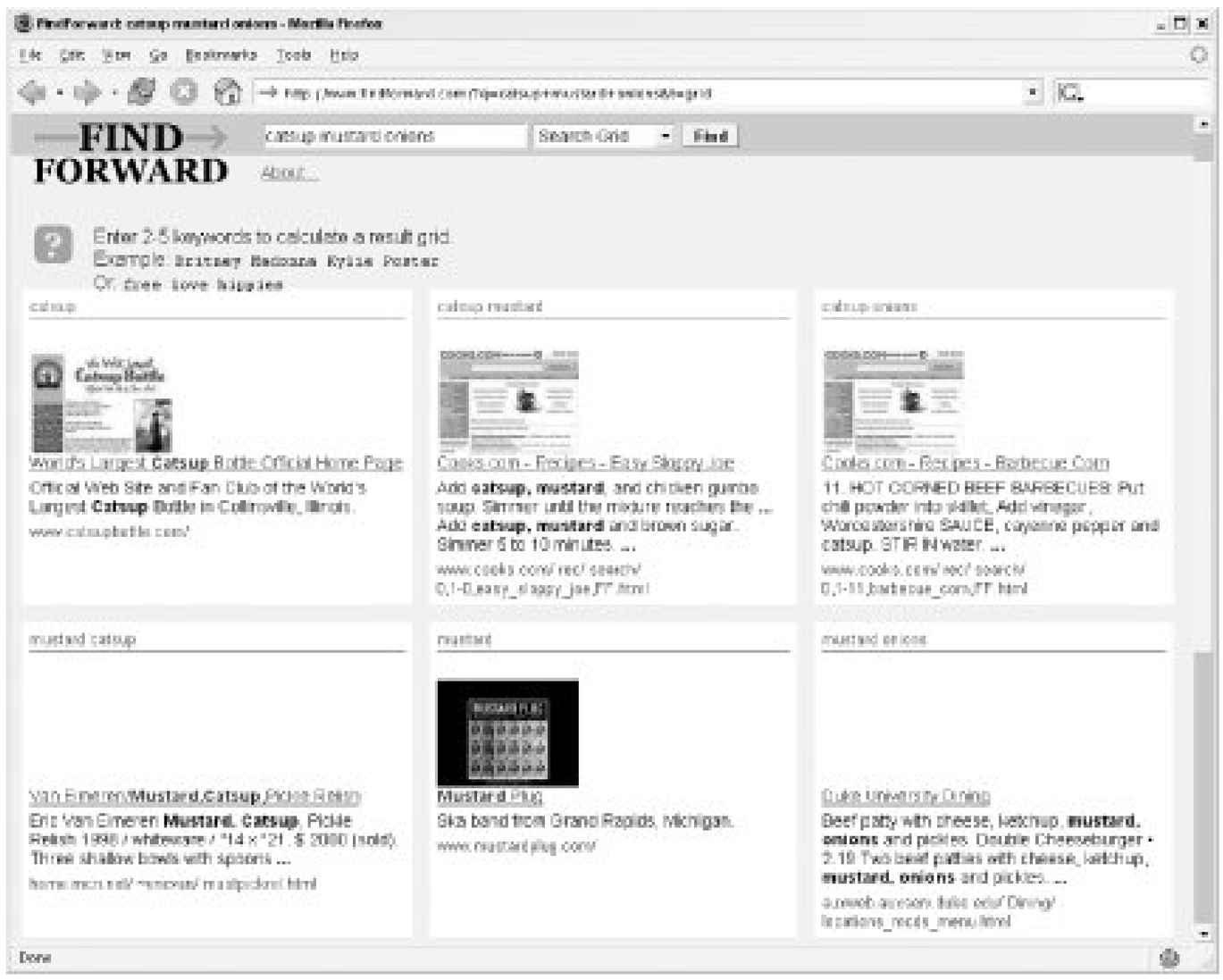
Figure 1-17. The first of several different searches, all in one grid



Note that you get nothing but the first result; this is not the tool to use if you want an in-depth search of each query. Instead, it's meant to give you a*bird's-eye view* of how the different combinations of search words impact the query.

There's also a version of Search Grid that's been integrated into a web tool calledFindForward (http://www.findforward.com/?t=grid), which gives you screenshots of some Google search results. FindFoward requires less typing: enter two to five words for which you want to check possible permutations. You get a large grid of search results, with screenshots available for some of the pages, as shown in Figure 1-18.

Figure 1-18. Search results for keyword combinations with screenshots!



Note that this grid searches each of your keywords individually (one square for mustard, one for pickles, one for relish) and searches every possible combination of two words (pickles relish, pickles mustard, mustard relish, etc.), but it doesn't search for three- and four-word permutations. In other words, this tool doesn't find every last possible permutation of your search. Again, it's an *overview* that gives you an idea of how different word combinations can affect your search, and it is not meant to be exhaustive.

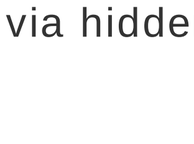
Buy why limit yourself to keyword sets that you can dream up? Google has its own tool in development to expand your keyword vocabulary based on a small set of words.Google Sets (http://labs.google.com/sets) allows you to enter several keywords and have Google predict similar keywords in a large or small set. For example, plug `catsup`, `mustard`, and `pickles` into the form and click Large Set. You should see a list of 25 or more words that run the condiment gamut from *Lettuce* to *Black Olives*, as shown in Figure 1-19.

Figure 1-19. Google Sets predictions based on a few keywords

You can click any of the words in the set to see a standard Google Search with that word. You can also click the Shrink Set to get a list of fewer (but potentially more accurate) items based on your original keywords. Google Sets can be handy if you want to expand your search possibilities but aren't sure which direction to go. You can even take the keyword suggestions from Google Sets back to the grid tools to see how using them in combination will affect your results.

Use the tools in this hack when you want to get a sense of how different queries will affect your search, when you're not sure about what set of search words will return the results you're looking for, and when you want to experiment with expanding your search without having to type several sets of keywords over and over again.

Hack 9. Hack Your Own Google Search Form



Build your own personal, task-specific Google search form.

If you want to do a simple search with Google, you need only the standard Simple Search form (the Google home page). But if you want to craft specific Google searches to use on a regular basis or provide for others, you can simply put together your own personalized search form.

Start with a garden-variety Google search form; something like this will do nicely:

```
<!-- Search Google -->
<form method="get" action="http://www.google.com/search">
<input type="text" name="q" size=31 maxlength=255 value="">
<input type="submit" name="sa" value="Search Google">
</form>
<!-- Search Google -->
```

This is a very simple search form. It takes your query and sends it directly to Google, adding nothing to it. But you can embed some variables to alter your search as needed. You can do this in two ways via hidden variables or by adding more input to your form.

Hidden Variables

As long as you know how to identify a search option in Google, you can add it to your search form via a hidden variable. The fact it's hidden just means that form users can't alter it. They can't even see it unless they look at the source code. Let's look at a few examples.

> While it's perfectly legal HTML to put your hidden variables anywhere between the opening and closing `<form>` tags, it's rather tidy and useful to keep them together after all the visible form fields.

File Type

As the name suggests, File Type specifies that your results are filtered by a particular file type (e.g., Word *.doc*, Adobe *.pdf*, PowerPoint *.ppt*, plain text *.txt*). Add a PowerPoint file type filter, for example, to your search form, like so:

```
<input type="hidden" name="as_filetype" value="PPT">
```

Site Search

Narrows your search to specific sites. While a suffix such as *.com* will work just fine, something more fine-grained such as the *example.com* domain is probably better suited:

```
<input type="hidden" name="as_sitesearch" value="example.com">
```

URL Component

Specifies a particular path component to look for in URLs. This can include a domain name but doesn't have to. The following tries to tease out documentation in your result set:

```
<input type="hidden" name="hq" value="inurl:docs">
```

Date Range

Narrows your search to pages indexed within the stated number of months. Acceptable values are between 1 and 12. Restricting your results to items indexed only within the last seven months is just a matter of adding:

```
<input type="hidden" name="as_qdr" value="m7">
```

Number of Results

Indicates the number of results you'd like to appear on each page, specified as a value of `num` between 1 and 100; the following asks for 50 per page:

```
<input type="hidden" name="num" value="50">
```

What would you use this for? If you regularly look for an easy way to create a search engine that finds certain file types in a certain place, this works really well. If this is a one-time search, you can always just hack the results URL (see "Understanding Google URLs" earlier in this chapter), tacking the variables and their associated values to the URL of the results page.

Mixing Hidden File Types: an Example

The O'Reilly web site (http://www.oreilly.com) contains hundreds of chapter previews from O'Reilly books in Adobe PDF format. If you want to find just the PDF files on the site, you must figure out how the site's search engine works or pester O'Reilly to add a file type search option. But you can put together your own search form that finds PDF files with the matching search terms on the oreilly.com site and read some free chapters from O'Reilly books in the process.

> Even though you're creating a handy search form, you're still resting on the assumption that Google's indexed most or all of the site you're searching. Until you know otherwise, assume that any search results Google gives you are incomplete.

Your form looks something like this:

```
<!-- Search oreilly.com for PDFs -->
<form method="get" action="http://www.google.com/search">
<input type="text" name="q" size=31 maxlength=255 value="">
<input type="submit" name="sa" value="Search Google">
<input type="hidden" name="as_filetype" value="pdf">
               <input type="hidden" name="as_sitesearch" value="oreilly.com">
               <input type="hidden" name="num" value="100">
</form>
<!-- Search oreilly.com for PDFs -->
```

Using hidden variables is handy when you want to search for one particular thing all the time. But if you want to be flexible in what you're searching for, creating an alternate form is the way to go.

Creating Your Own Google Form

Some variables work well hidden; however, for other options, you can give your form users visible options to provide more flexibility.

Let's go back to the previous example. You want to let your users search for PDF files, but you also want them to be able to search not only oreilly.com, but also the State of California or the Library of Congress web sites. Obviously, there are various ways to design this form; this example uses a couple of simple pull-down menus.

```
<!-- Custom Google Search Form-->
<form method="get" action="http://www.google.com/search">
<input type="text" name="q" size=31 maxlength=255 value="">
               Search for file type:
               <select name="as_filetype">
                  <option value="ppt">PowerPoint</option>
                  <option value="xls">Excel</option>
                  <option value="doc">Word</option>
               </select><br />
               Search site:
               <select name="as_sitesearch">
                  <option value="oreilly.com">oreilly.com</option>
                  <option value="state.ca.us">State of California</option>
                  <option value="loc.gov">The Library of Congress</option>
               </select>
<input type="hidden" name="num" value="100">
<input type="submit" value="Search Google">
</form>
<!-- Custom Google Search Form-->
```

FaganFinder (http://www.faganfinder.com/engines/google.shtml) is a wonderful example of a thoroughly customized form.

If you find yourself running fairly complex queries on a regular basis, you can speed things up by setting a few options in a custom form. And chances are good that if you find the convenience of a custom form helpful, others will too. So, making your custom form available on your web site is a good way to let others share in your productivity.

PREV

Hack 10. Compare Google and Yahoo! Search Results

Pit Google and Yahoo! against each other and find more search results in the process.

If you've ever searched for the same phrase at both Google and Yahoo!, you've probably noticed that the results can be surprisingly different. That's because Google and Yahoo! have different ways of determining which sites are relevant for a particular phrase. Though both companies keep the exact way of how they determine the rank of results a secretto thwart people who would take advantage of itboth Yahoo! and Google provide some clues about what goes into their ranking system.

At the heart of Google's ranking system is a proprietary method it calls PageRank, and Google doesn't give detailed information about it. But Google does say this:

Google's order of results is automatically determined by more than 100 factors, including our PageRank algorithm.

Here's the official word from Yahoo!:

Yahoo! Search ranks results according to their relevance to a particular query by analyzing the web page text, title, and description accuracy as well as its source, associated links, and other unique document characteristics.

Though we might never know exactly *why* results are different between the two search engines, at least we can have some fun spotting the differencesand end up with more search results than either one of the sites would have offered on their own.

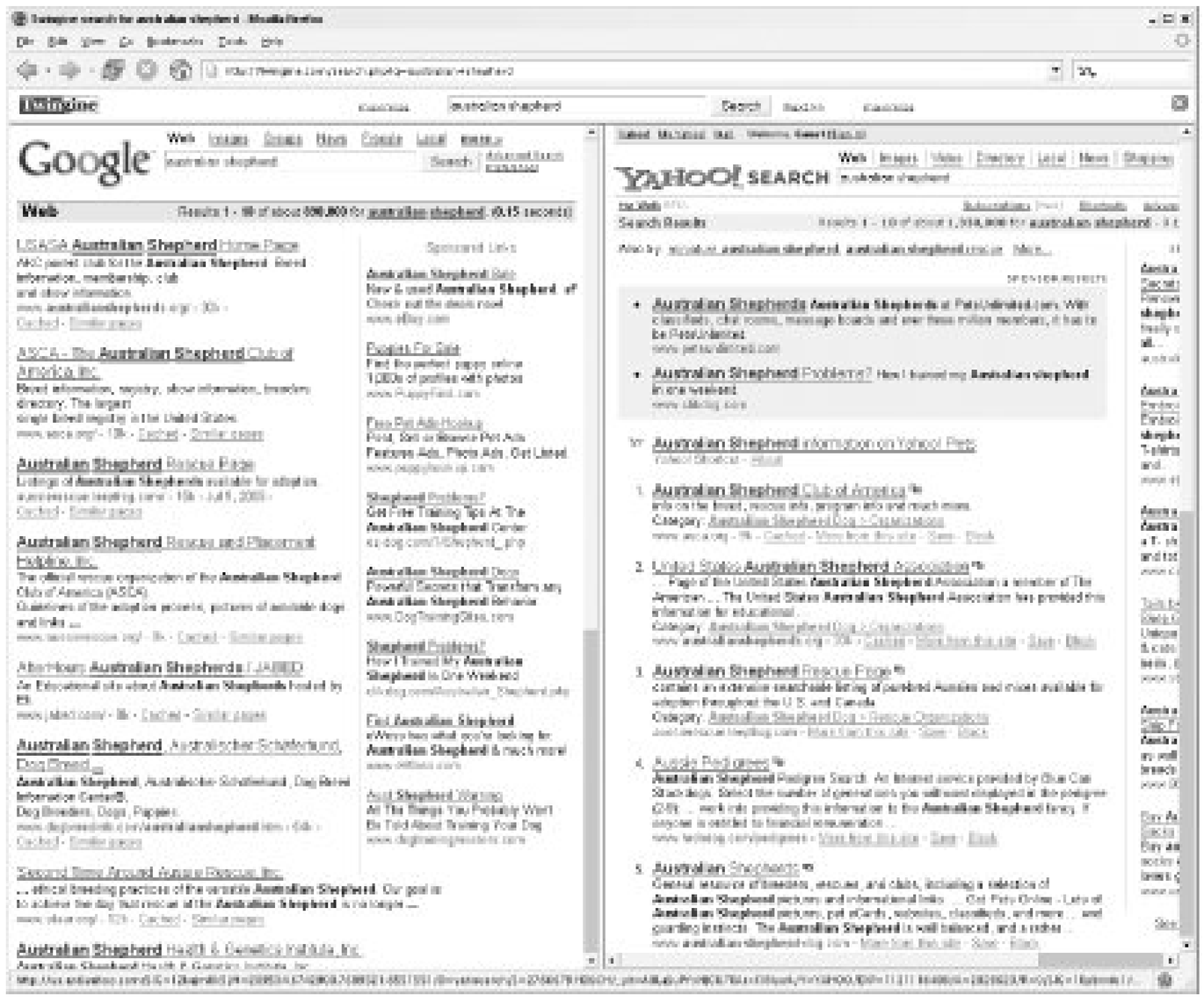
One way to compare results is to simply open each site in separate browser windows and manually scan for differences. If you search for your favorite dog breedsay,`"australian shepherd"`you'll find that the top few sites are the same across both Yahoo! and Google, but the two search engines quickly diverge into different results. At the time of this writing, both sites estimate exactly 1,030,000 total results for this particular query, but estimated result counts might be a way to spot differences between the sites.

Viewing both sets of results in different windows is a bit tedious, and a clever Norwegian developer named Asgeir S. Nilsen has made the task easier, at a site called Twingine.

Twingine

The Twingine site (http://twingine.com) contains a blank search form into which you can type any search query. When you click Search, the site brings up the results pages for that query from both Yahoo! and Google, side by side. To be fair, the sides on which Google and Yahoo! appear change at random, so people who prefer one side of the screen to the other won't be biased. Plugging `"australian shepherd"` into Twingine yields a page such as the one shown in Figure 1-20.

Figure 1-20. Google and Yahoo! going head to head at Twingine



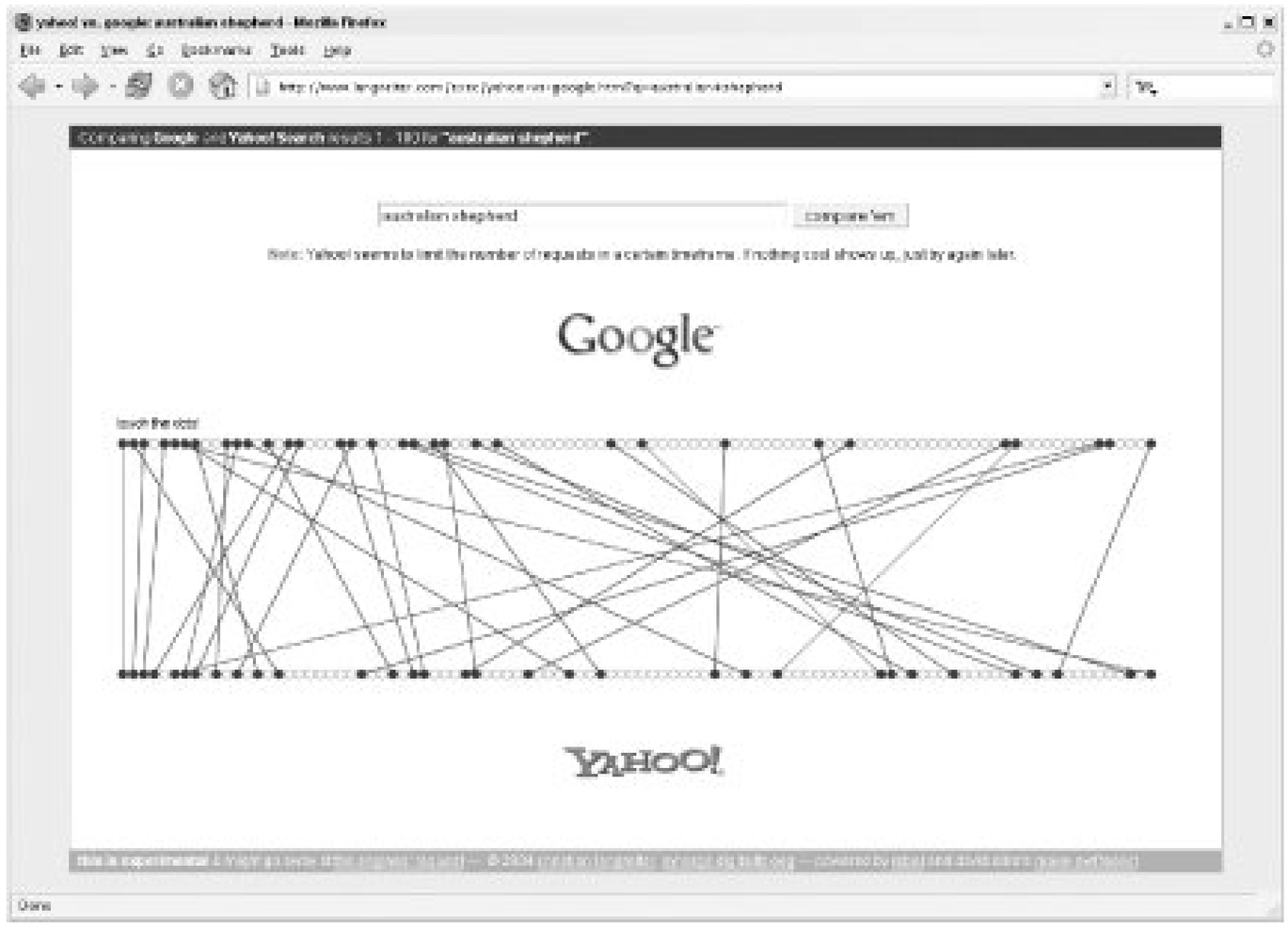
Clicking Next or Previous in the top frame at Twingine takes you to the next or previous page in the search results at both sites.

Surfing the pages in the search results at Twingine can be a bit tricky. You'll probably want to open linked search results in a new window or tab, so that you can keep your place in the search results at both Yahoo! and Google. You can open links in a new window by right-clicking the link (Ctrl-click on a Mac) and choosing Open Link in New Window from the menu. You can also set your search preference at either search engine to automatically open links in a new window when you click a search result.

Yahoo! Versus Google Diagram

Another site, developed by Christian Langreiter, adds a bit of analysis to the different sets of search results between Yahoo! and Google. If you have Flash installed, you can type a search query into the form at http://www.langreiter.com/exec/yahoo-vs-google.html, and the site fetches the search results from both engines in the background using their open APIs. The site delivers the results in a chart, as shown in Figure 1-21.

Figure 1-21. Mapping the differences between Yahoo! and Google results



Each blue or white dot in the diagram represents a search result URL, and the position of the dot represents the ranking. The dots on the far left are the top search results, and the further right you go, the further down you go in the search results. The blue lines represent the same URL, so you can see exactly where Google and Yahoo! line up.

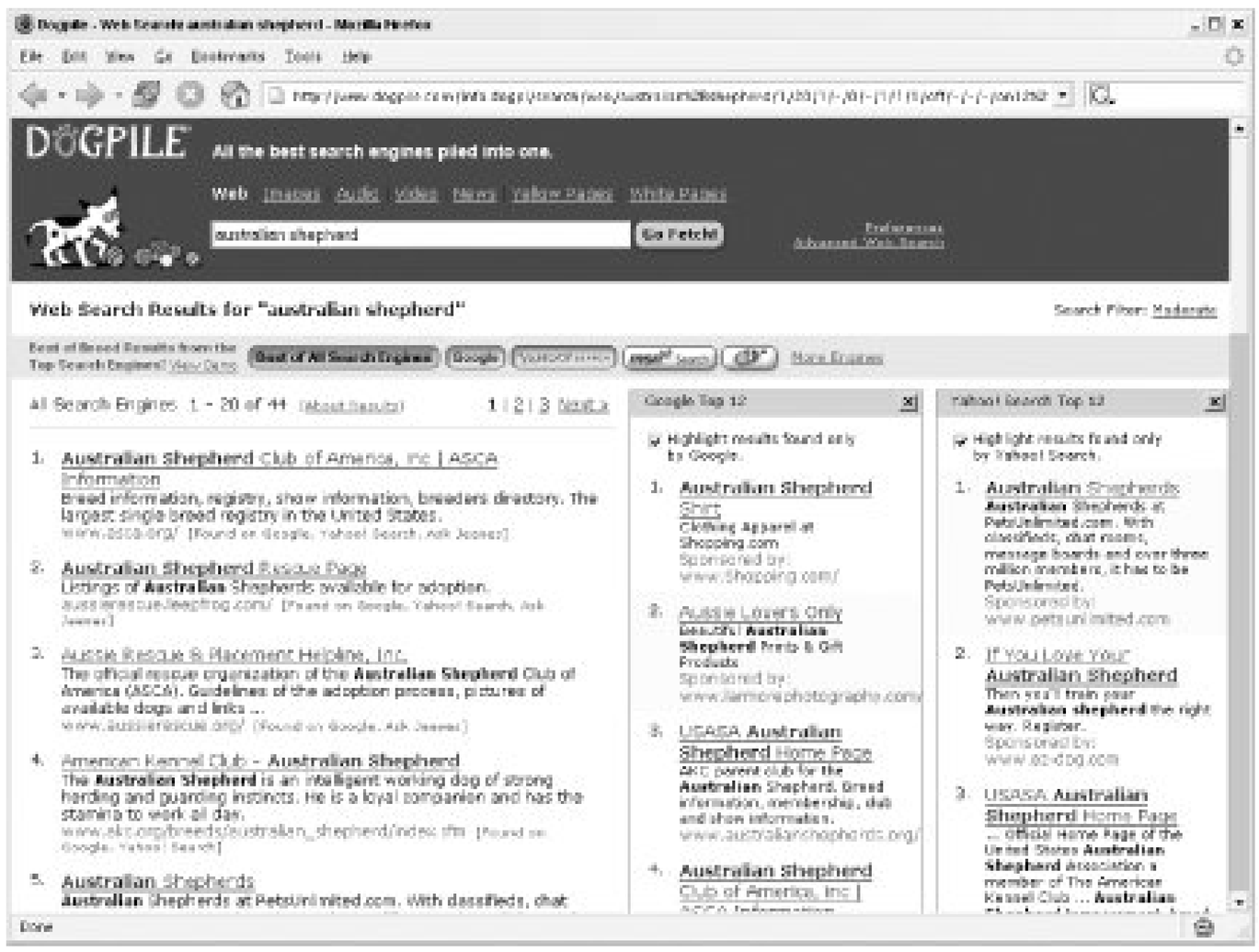
In Figure 1-21, you can see that the top search result for `"australian shepherd"` is the same URL, but the lines aren't as evenly matched further down in the results. As you hover over each dot, you see the URL, which you can click to visit that particular search result.

The white dots in the diagram represent a URL that one search has in the results that the other does not. And as this diagram demonstrates, neither search engine has a monopoly on matching pages, nor does each engine's index have every page on a particular topic.

Piling Results Together

If you want to compare even more results than the big two provide, a service calledDogpile (http://www.dogpile.com) will gather responses from six different search engines into a single page. As shown in Figure 1-22, each individual match for the query lists the search engines where that result was found.

Figure 1-22. Comparing Google and Yahoo! results at Dogpile.com



By clicking the search engine buttons at the top of the page, you can directly compare the top 12 results from Google, Yahoo!, and other search engines. Any listing unique to a particular search engine is highlighted in yellowso you can see at a glance what you'd be missing by using either Google or Yahoo! alone.

While the individual search results in the main column show the "Best of All Search Engines," be aware that some of the individual results are from advertising on search enginesnot simply the most organic search results. Each listing indicates which search engines it came from, and ads are clearly labeled.

If you already do serious research with search engines, you're very aware that having several search tools at your disposal is better than relying on one. And with the methods mentioned in this hack, you can compare and contrast the tools, giving you more results to choose from.

Hack 11. Cover Your Tracks

By understanding how your browser stores information related to your Google searches, you can be sure that your searches are your own.

Most of us think of our Google searches as something private, an exchange between one individual and Google. But if you share a computer with others, your searches might not be as private as you think. Whether you're searching for a surprise birthday gift, a private medicalconcern, legal advice, or "researching" some risqu\x8e topic, there are times when your browser's memory can come back to haunt you.

By defaultin an effort to help your memoryyour computer remembers your past Google searches and stores them so you can access them later. There are several ways your computer accomplishes this, and you should be aware of each of them if you want to cover your tracks completely.

Browser History

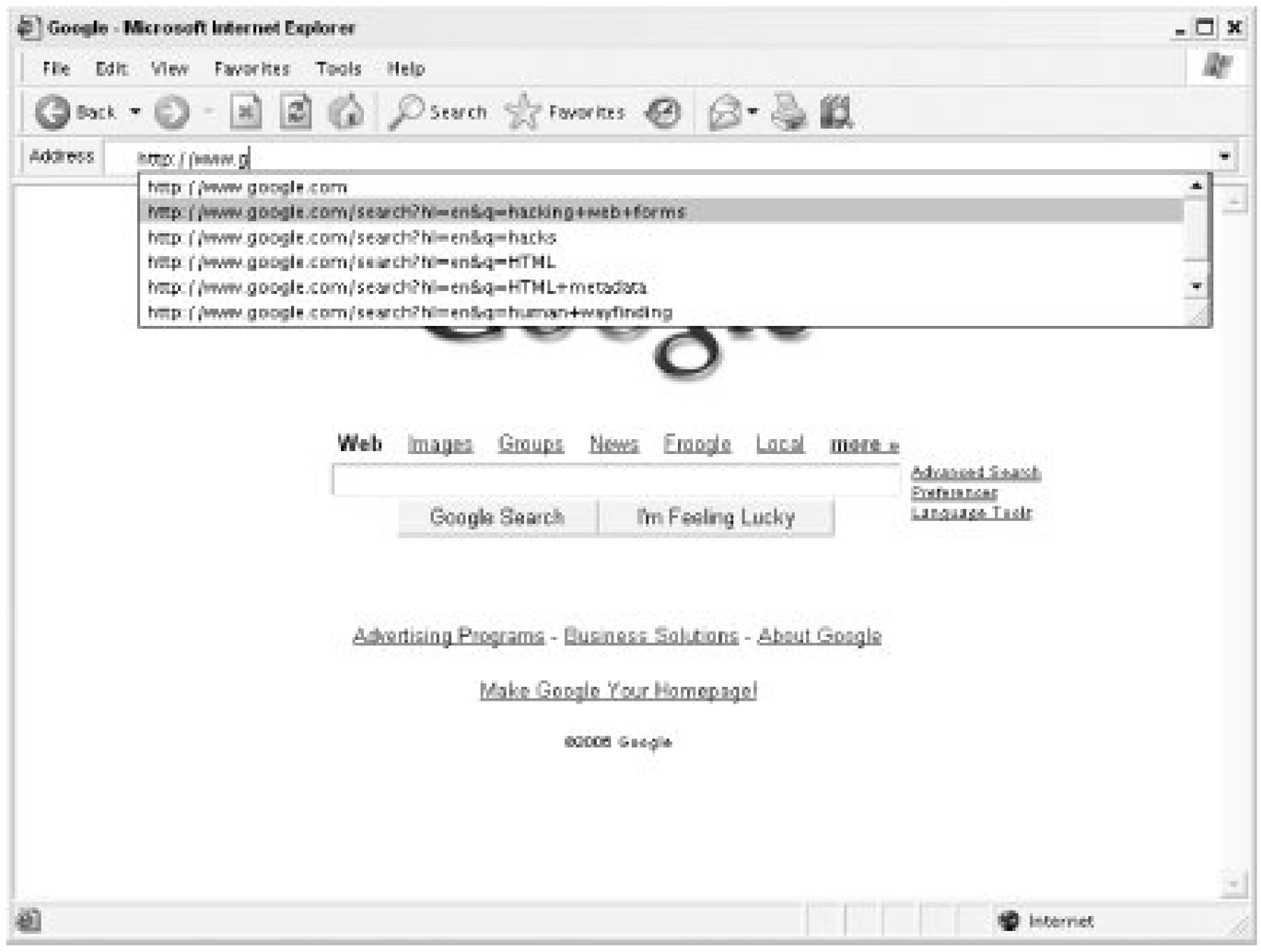
The first and most obvious place that your browser stores your past searches is in your browser history. You can quickly view your current browser history in Firefox or Internet Explorer by typing Ctrl-H (Command-Shift-H on a Mac). A new pane will open that includes all of the sites you've visited recently, along with the specific pages at those sites, as shown inFigure 1-23.

Figure 1-23. Browser history pane in Firefox

From the pane on the left, you can easily revisit sites. Open the*google.com* folder to see recent searches, and note that other Google searches, such as Google images, are stored in its own folder, *images.google.com*. If you see a search you'd rather not share with others, you can simply highlight that particular entry, right-click, and click Delete on the menu.

Also be aware that your browser history is exposed through your address bar. As you start typing a URL into the address bar, the browser tries to guess where you want to go by offering matching URLs in your search history. If you start typing `http://www.g`, you'll find a list of recent Google searches, as shown in Figure 1-24.

Figure 1-24. Address history in Internet Explorer



By studying the URL, you can see what search term was used, and can highlight the entry to visit that page of search results. In Firefox, you can delete any entry by highlighting it and typing Shift-Delete. Internet Explorer users can only selectively delete from the History pane.

If you want to completely remove your browser history, there's a faster way than deleting each entry one at a time. Here are the steps for purging your history:

Internet Explorer

Choose Tools Internet Options from the top menu. Look for the History section on the General tab and click Clear History. You can also adjust the number of days you'd like to keep pages in your browser history; set this to 0 to disable your history completely. Click OK, and your browser history will be gone.

Firefox

Choose Tools Options (Firefox Preferences on a Mac) and click Privacy on the top menu. Choose the History tab and click Clear Browsing History Now. You can also set the number of days you'd like to keep pages herewith 0 disabling the feature.

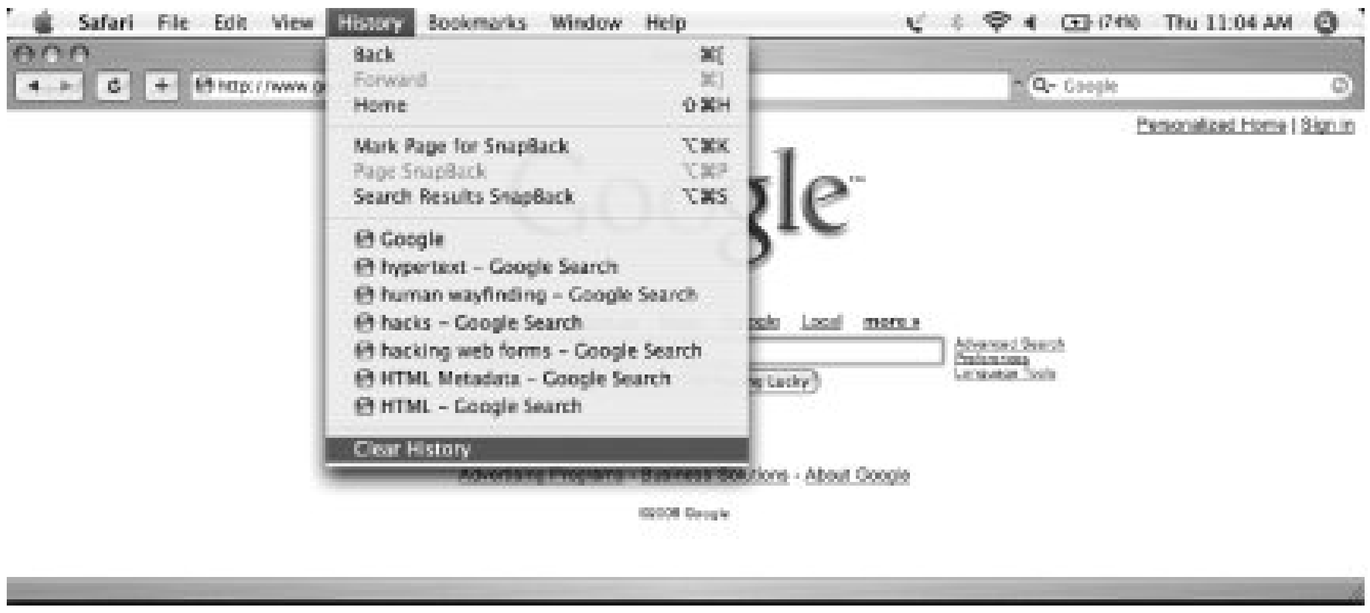
Opera

Opera stores typed-in addresses and visited pages in two distinct places, so you want to be sure to clear both. Choose Tools Preferences (Opera Preferences on a Mac) from the top menu, click the Advanced tab, and then click History from the menu. Click Clear next to Typed in Addresses and Visited Addresses (only Addresses on a Mac). You can also use this opportunity to set the number of entries you'd like Opera to rememberup to 500 typed-in

addresses and up to 10,000 visited addresses. Set this to 0 to disable your history.

Safari users on Mac OS X can manage their browser history through the History menu option shown in Figure 1-25.

Figure 1-25. The History menu in Safari



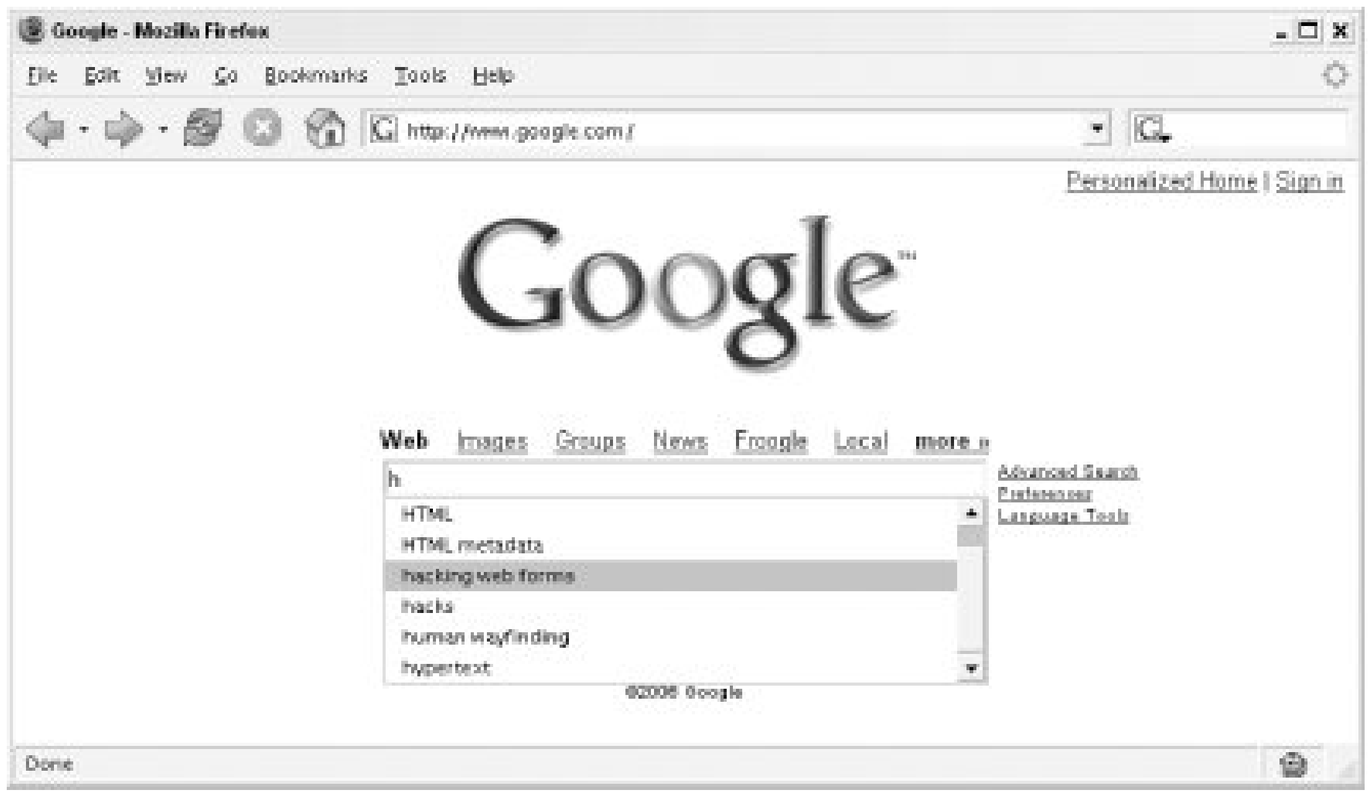
Unfortunately, you can't selectively delete entries from your Safari browsing history, but you can clic Clear History to remove all of your past browsing.

Safari users on Mac OS X can take advantage of a feature called Private Browsing. WithPrivate Browsing enabled, sites aren't added to the browser history and form data isn't saved. You can use the Private Browsing mode at all times to effectively disable your browser history.

Saved Form Data

Another place where your past Google searches can be found is insaved form data. Having this data available is a convenience, because you can type a single letter into the Google search form and get a list of your past searches that start with that letter, as shown inFigure 1-26.

Figure 1-26. Saved form data in Firefox



Instead of retyping complex queries that you put together in the past, you can simply choose your past query, click or type Enter, and the search is re-run. But if you'd rather not share these past queries with others on your computer, you need to delete them.

You can selectively delete entries from this menu in both Firefox and Internet Explorer by highlighting an entry and typing Shift-Delete.

Here are the steps to delete all your saved form data in one go:

Internet Explorer

Choose Tools Internet Options from the top menu and then click the Content tab. Click AutoComplete and then the Clear Forms button. Uncheck the box next to Forms and click OK to disable AutoComplete.

Firefox

Choose Tools Options (Firefox Preferences on a Mac) from the top menu, click Privacy, and choose the Saved Forms tag. Click the button labeled Clear Saved Form Data Now to remove your past form entries. You can also take the opportunity to uncheck the box next to "Save information" to disable the feature.

Safari

Choose Safari Preferences from the top menu and choose AutoFill. Click Edit... next to "Other forms" and highlight google.com (and any others you'd like to remove) on the list. Click Remove, and your saved form data is deleted. You can also uncheck the box next to "Other forms" to disable the AutoFill feature.

At the time of this writing, Opera has a feature called Wand that saves usernames and passwords, but the browser doesn't save form data as the other browsers do.

Even with your browser history and saved form data gone, there are still ways for persistent snoops to find your Google searches.

Browser Cache

All browsers use a cache to store recently accessed web pages and images. With a local copy of the files on your computer, the browser can display pages much faster if you visit the site again in the future. The cache also leaves a trail of your surfing history, including Google searches.

Figure 1-27 shows the *Temporary Internet Files* folder where Internet Explorer stores cached items.

Figure 1-27. Viewing the Internet Explorer cache

As you can see, the Internet Address is in plain view, along with the search queries used. The first trick to removing items from your cache is finding the cache folder. Typically, these are buried deep in your filesystem and given cryptic names because they're not intended to be accessed by humans. Luckily, they're easy to browse if you know how to get there:

Internet Explorer

To find your cache, choose Tools Options from the menu and click Settings... under Temporary Internet Files. Click View Files... to bring up the files in an Explorer window. From there, you can selectively delete any files in your cache, including Google pages.

Firefox

To view your cache, type `about:cache` in the address bar and click List Cache Entries to see the files in your cache. Though you can't selectively delete through this page, the cache directory path is listed at the top. From there, you can browse to the files with Windows Explorer (or the Finder on a Mac).

Opera

To find your cache directory, choose Help About Opera from the top menu. Your cache directory is listed on the page, and from there you can delete the past Google results pages you've visited.

Safari

In Finder, browse to your Safari cache folder, *~/Library/Caches/Safari/*, and selectively delete Google pages.

While you might not need to go to this extreme to remove your past Google searches, knowing where the information is located gives you the choice. And even going through these steps is no guarantee that the information is gone. In the hands of a hard-disk forensics expert, even deleted information can often be recovered.

A complete privacy strategy is beyond the scope of this book, but you can turn to*Computer Privacy Annoyances* by Dan Tynan (O'Reilly) for even more information about keeping your personal information private.

Hack 12. Improve Google's Memory

With a feature called Search History, Google stores the searches you've made and the links you've followed so you can go back to them in the future.

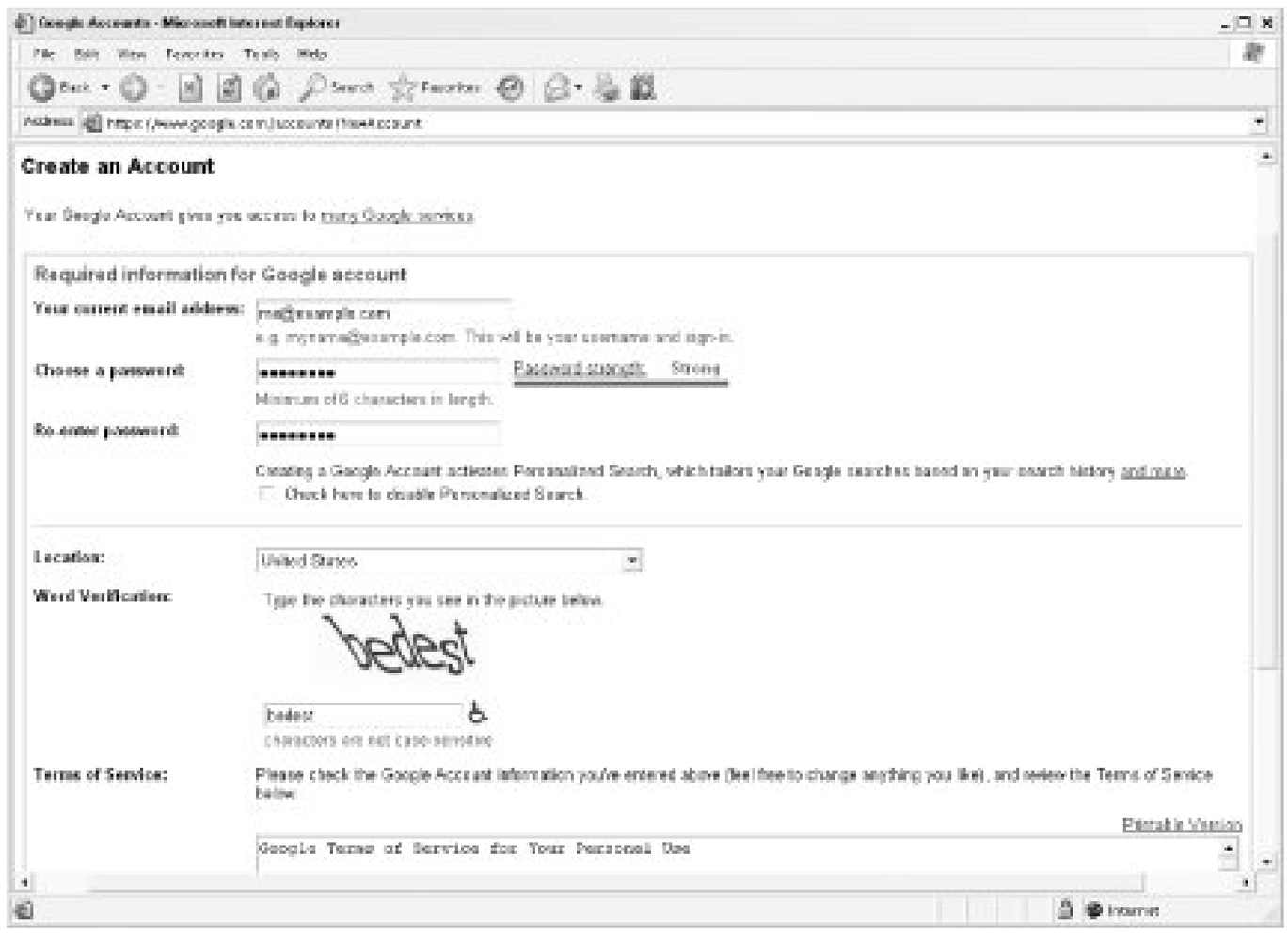
Google is an impressive organizer of information, but it's not very personable. Google is very much the same for me as it is for you. In fact, if you search Google with the word `personable`, you'll see the same results I do. However, Google is working on technology that will tailor its search results to you as an individual. One step in that direction is the Search History (a.k.a. Personalized Search) feature, in beta testing at the time of this writing.

You've probably already experienced how Google's memory can help you recall a search you did in the past. As you type letters into the main Google Search form, your browser tries to complete your thought, recalling past searches. This limited form of memory [Hack #11] can be handy, but it's not terribly accurate or organized. For one, you can't tell your browser which searches were successful and which weren't. You can't highlight favorite results or organize them in any way.

If you turn on Google's memory through the Search History feature, you can let Google do the work of remembering how you use the site. In addition, you have access to your search history, no matter how you access the Web, because your history is stored at Google instead of your local computer.

The best way to get to know how Search History works is to try it out. You need a Google Account to use Search History; if you have a Gmail Account, you're ready to go. If you don't have a Google Account yet, browse to https://www.google.com/accounts/NewAccount and sign up. Google offers the option to disable Personalized Search when you create an account, as shown in Figure 1-28, but if you're there to try out Search History, you need to leave this unchecked.

Figure 1-28. Google Account sign-up page



Once you have an account, browse to the Google home page, click Sign In (if you're not signed in already), then click My Account at the top of the page. From your account page, click the Personalized Search link under Try New Services. From there, your Search History is activated.

Once your account is activated, you can use Google as you normally would and have access to the following list of features.

Searches and documents clicked

> Search History will remember every search you make at Google and every site you click to visitrecording the time and date of each click.

Searches without clicked results

> Even searches in which you don't click any results will be stored for later reference.

Bookmarks

> From your Search History page, you can highlight links you've clicked to save as bookmarks. You can give each bookmark a number of labels so you can find it later. For example, you might give a bookmark to the O'Reilly Hacks site (http://hacks.oreilly.com) the labels *books*, *O'Reilly*, and *geek* to help you find the bookmark later. You can also add your own notes to each bookmark.

Search Trends

> Once you've used Search History for a while, Google will help you visualize your searching

activity by spotting trends and tabulating the amount of searches per day.

Personalized Results

As Google gets to know you, it will refine the standard Google Search results page to match your past searching activity. Links you've bookmarked will show their labels in searches, and you can block some sites from showing up in search results.

Searchable History

Your history itself is searchable. So if you want to get back to that geeky book site you know you once found through Google, you can try a search on your unique history to find the site.

At the time of this writing, Search History is supported across Web, Images, News, and Froogle searches. Other Google searchessuch as Groups, Local, or Book searcheswill not appear in your history.

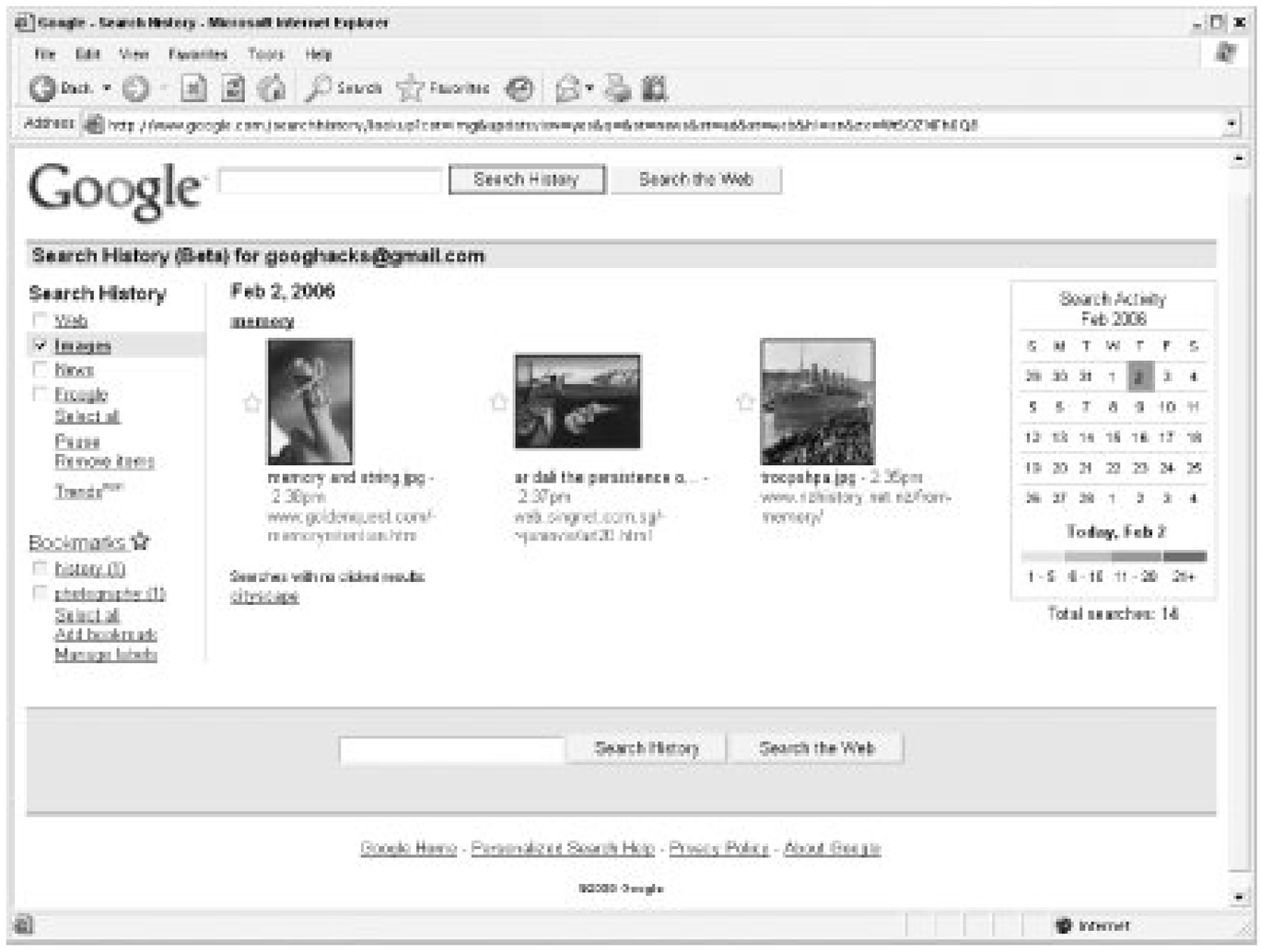
To access the features, click Search History at the top of the Google main page. You'll see a list of your recent searches organized by date, as shown in Figure 1-29.

Figure 1-29. Google Search History page

Each bold entry in the center column is a search you performed at Google, with links clicked from that search directly below. Use the links to the left to filter the list to searches performed at Google

properties. For example, check the Images box to see your recent Google Image searches, as shown in Figure 1-30.

Figure 1-30. Google Images Search History



If there's a link or image that you want to save more permanently for future reference, click the star next to that item's listing. This saves the item in your Bookmarks. In addition, if there's an item you want Google to forget, click the Remove items link on the left side of the page, choose the item, and click Remove.

You can view any item you've bookmarked by clicking the Bookmarks link on the left side of the page. From there, you can edit any bookmark by adding labels or notes. As you label your bookmarks, each label appears under the Bookmarks link on the left side of the page. Clicking a specific label shows you only those bookmarks with that particular labelshowing just a list of your bookmarked sites related to photography, for example.

There's currently no way to share your history or bookmarks with others, and your search history is as private as your Google Account password. So, while it might feel odd to save all your Search History for review, it's a step toward your own personal search engines with results just for you. And if you ever want to part ways with your Search History, click the My Account link and choose Delete Personalized Search from the menu. Your history will be history.

Hack 13. Find Out What Google Thinks ___ Is

What does Google think of you, your friends, your neighborhood, or your favorite movie?

If you've ever wondered what people think of your hometown, your favorite band, your favorite snack food, or even you, Googlism (http://www.googlism.com) may provide you with something useful.

The Interface

The interface is dirt simple. Enter your query and check the appropriate radio button to specify whether you're looking for a *who*, a *what*, a *where*, or a *when*. Figure 1-31 shows a representative results page for Sherlock Holmes, famous fictional detective. You can also use the tabs to see what other objects people are searching for and what searches are the most popular.

Figure 1-31. Googlism results for Sherlock Holmes

Some of the results you find are not safe for work.

What You Get Back

Googlism responds with a list of things Google believes about the query at hand, be it a person, place, thing, or moment in time. For example, a search for`Perl` and "What" returns, along with a laundry list of others:

```
Perl is y2k compliant
Perl is not my favourite programming language
Perl is the coder's language of choice
Perl is the language of love
```

These are among the more humorous results for`Steve Jobs` and "Who":

```
steve jobs is my new idol
steve jobs is at it again
steve jobs is trying to kill me
```

To figure out what page any particular statement comes from, simply copy and paste it into a plain old Google search, with the complete phrase in quotes. That last statement, for instance, came from a 2002 blog post about iMacs at http://www.fismo.com/KeepUp/fog0000000025.html.

Practical Uses

For the most part, this is a party hacka good party hack. It's a fun way toaggregate related statements into a silly (and occasionally profound) list.

But that's just for the most part. Googlism also works as a handy ready-reference application, allowing you to quickly find answers to simple or simply asked questions. Just ask a question of Googlism in a way that can end with the word "is." For example, to discover the capital of Virginia, enter `The capital of Virginia`. To learn why the sky is blue, try`The reason the sky is blue`.

Sometimes, this doesn't work very well; try`the oldest person in the world`, and you're immediately confronted with a variety of contradictory information. You'd have to visit each page represented by a result and see which answer, if any, best suits your research needs.

Expanding the Application

This application is a lot of fun, but it can be expanded. The trick is to determine how web page creators generate statements.

For example, when initially describing an acronym, many writers use the words`"stands for"`. So you can add a Googlism that searches for your keyword and the phrase "stands for." Do a Google search for `"SETI stands for"` and `"DDR stands for"` and you'll see what I mean.

When referring to animals, plants, and even stones, the phrase "are found" is often used, so you can add a Googlism that locates things. Do a Google search for `sapphires are found` and `jaguars are found` and see what you find.

See if you can think of any phrases that are in common usage, and then check those phrases in Google to see how many results each phrase has. You might get some ideas for a topic-specific Googlism tool yourself.

PREV

Hack 14. Browse the World Wide Photo Album

Take a random stroll through the world's photo album using some clever Google Image searches (and, optionally, a smidge of programming know-how).

The proliferation of digital cameras and the growing popularity of camera phones are turning the Web into a worldwide photo album. It's not only the holiday snaps of your Aunt Minnie or the minutiae of your moblogging friend's day that are available to you. You can actually take a stroll through the publicly accessible albums of perfect strangers if you know where to look. Happily, Google has copies, and a couple of hacks know just where to look.

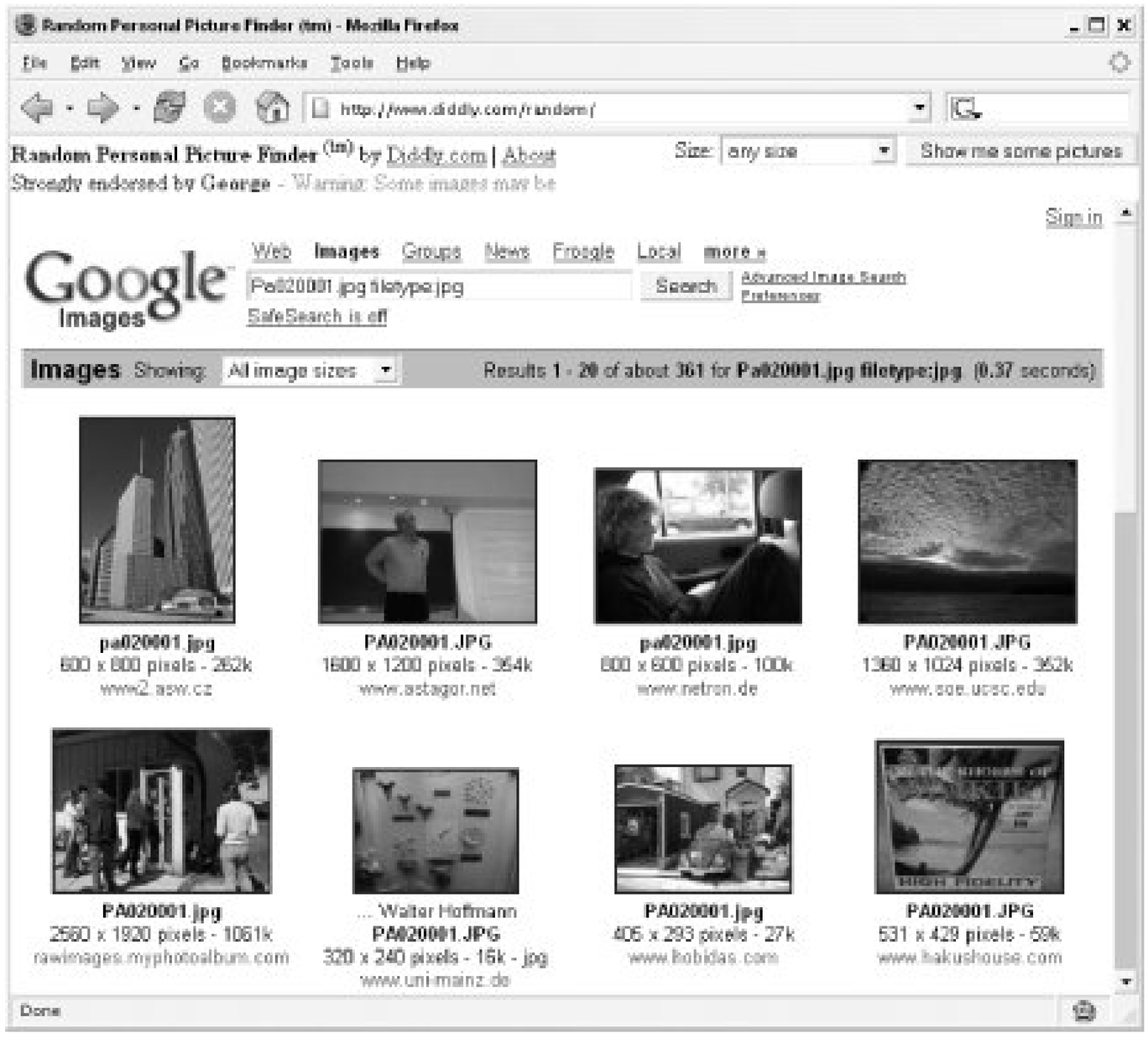
Random Personal Picture Finder

Digital photo files have relatively standard filenames (e.g.,*DSC01018.JPG*) by default and are usually uploaded to the Web without being renamed. TheRandom Personal Picture Finder (http://www.diddly.com/random) sports a clever little snippet of JavaScript code that simply generates one of these filenames at random and queries Google Images for it.

The result, shown in Figure 1-32 , is something like looking through the world's photo album: people eating, working, posing, and snapping photos of their cats, furniture, or toes. And since it's a normal Google Images search, you can click on any photo to see the story behind it, and the other photos nearby.

Neat, huh?

Figure 1-32. The Random Personal Picture Finder



Note that people snap pictures of not just their toes (or the toes of others). While an informal series of Shift-Reloads in my browser turned up only a couple of questionable bits of photographic work, you should assume the results are not workplace- or child-safe.

The code behind the scenes, as I mentioned, is really very simple: a swatch of JavaScript (view the source of http://www.diddly.com/random/random.html in your browser to see the JavaScript bits for yourself) and list of camera types and their respective filename structures (http://www.diddly.com/random/about.html). You're simply redirected to Google Images with generated search query in tow.

A smidge of Python illustrates just how simple it is to generate a link to a random collection of photos shot with a Canon digital camera:

```
$ python
ActivePython 2.4 Build 244 (ActiveState Corp.) based on
Python 2.4 (#60, Feb  9 2005, 19:03:27) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from random import randint
>>> linkform = 'http://images.google.com/images?q=IMG_%s.jpg'
>>> print linkform % str(randint(1, 9999)).zfill(4)
http://images.google.com/images?q=IMG_3275.jpg
```

You can easily use this as the basis of a CGI script that acts in the same manner as the Random Personal Picture Finder.

Searching Personal Sites

In addition to finding personal photos based on common filenames, you can also use Google Images to search sites that host personal photos. The collision of digital photography and blogs [Hack #41]means millions are posting their snapshots along with their posts. If you limit your image searches to common blog domains, you'll find thousands of personal photographs.

For example, it's common for people to post to their blog when they get a new car and include a picture for their friends, family, and complete strangers to look at. If you want to find some personal pictures of cars, browse to Google Images (http://images.google.com) and try the following query using the `site:` keyword:

```
"new car" site:blogger.com
```

You'll find pictures that have been posted to Blogger's image-hosting service. You can often click the photo, find the post, and read an entire story behind a particular photo.

If you want to search across several services at once, you can combine queries. Say you want to search for photos of cars across both Blogger and competitor TypePad. Try the following query:

```
"new car" site:typepad.com OR site:blogger.com
```

There are hundreds of sites that host personal photos; all you need to do is find the domains. Here are a few to get you started: `xanga.com` , `geocities.com` , `textamerica.com` , `flickr.com` , and `smugmug.com` . To find more, take a look at the Photo Sharing category in theGoogle Directory:

http://www.google.com/Top/Computers/Internet/On_the_Web/Web_Applications/Photo_Sharing/

The Web has become our global photo album. And while browsing through the millions of personal photos available can verge on voyeuristic, it's a reminder that we all love to take, share, and look at photographs.

Paul Bausch and Aaron Swartz

Hack 15. Find Similar Images

Explore the Web in a new way by finding other images of the same name.

I will be the first to admit that this hack has no practical purpose. I originally conceived it in an IRC channel, when someone posted a link to http://images.google.com/images?q=P5170003. That particular keyword is a filename used by a particular brand of digital camera. Some cameras generate filenames based on the date the photo was taken and a unique identifier within the camera others simply use an incrementing identifier starting with 1. Many people take digital images and then simply publish them online, without giving the photo a more meaningful filename. The end resu is that you can use Google Images to find a random selection of images published by different people. (This particular query finds photos taken on May 17, my wedding anniversary.)

> This hack relies on the Greasemonkey Plugin (http://greasemonkey.mozdev.org/) for the Firefox web browser (http://www.mozilla.com/firefox/).

Anyway, this hack converts all unlinked images into links to Google Images to find other random images with the same filename. If that sounds silly, that's because it is. It's also surprisingly fun, if you like that sort of thing.

The Code

This user script runs on all pages. It uses the `document.images` collection to find all the images on the page and wraps each of them in a link to *http://images.google.com/images?q=* plus the image filename. Firefox seriously dislikes replacing an element with another element that contains the original element, so we use the `cloneNode` method to make a copy of the original `<img>` element, put it in an `<a>` element, and then replace the original `<img>`.

Save the following user script as *similarimages.user.js*:

```
// ==UserScript==
// @name           Find Similar Images
// @namespace      http://diveintomark.org/projects/greasemonkey/
// @description    links images to find similar images on Google Image Search
// @include        http://*
// @exclude        http://*.google.tld/*
// ==/UserScript==

for (var i = document.images.length - 1; i >= 0; i--) {
```

```
    var elmImage = document.images[i];
    var usFilename = elmImage.src.split('/').pop(  );
    var elmLink = elmImage.parentNode;
    if (elmLink.nodeName != 'A') {
        var elmLink = document.createElement('a');
        elmLink.href = 'http://images.google.com/images?q=' +
            escape(usFilename);
        elmLink.title = 'Find images named ' + usFilename;
        var elmNewImage = elmImage.cloneNode(false);
        elmLink.appendChild(elmNewImage);
        elmImage.parentNode.replaceChild(elmLink, elmImage);
    }
}
```

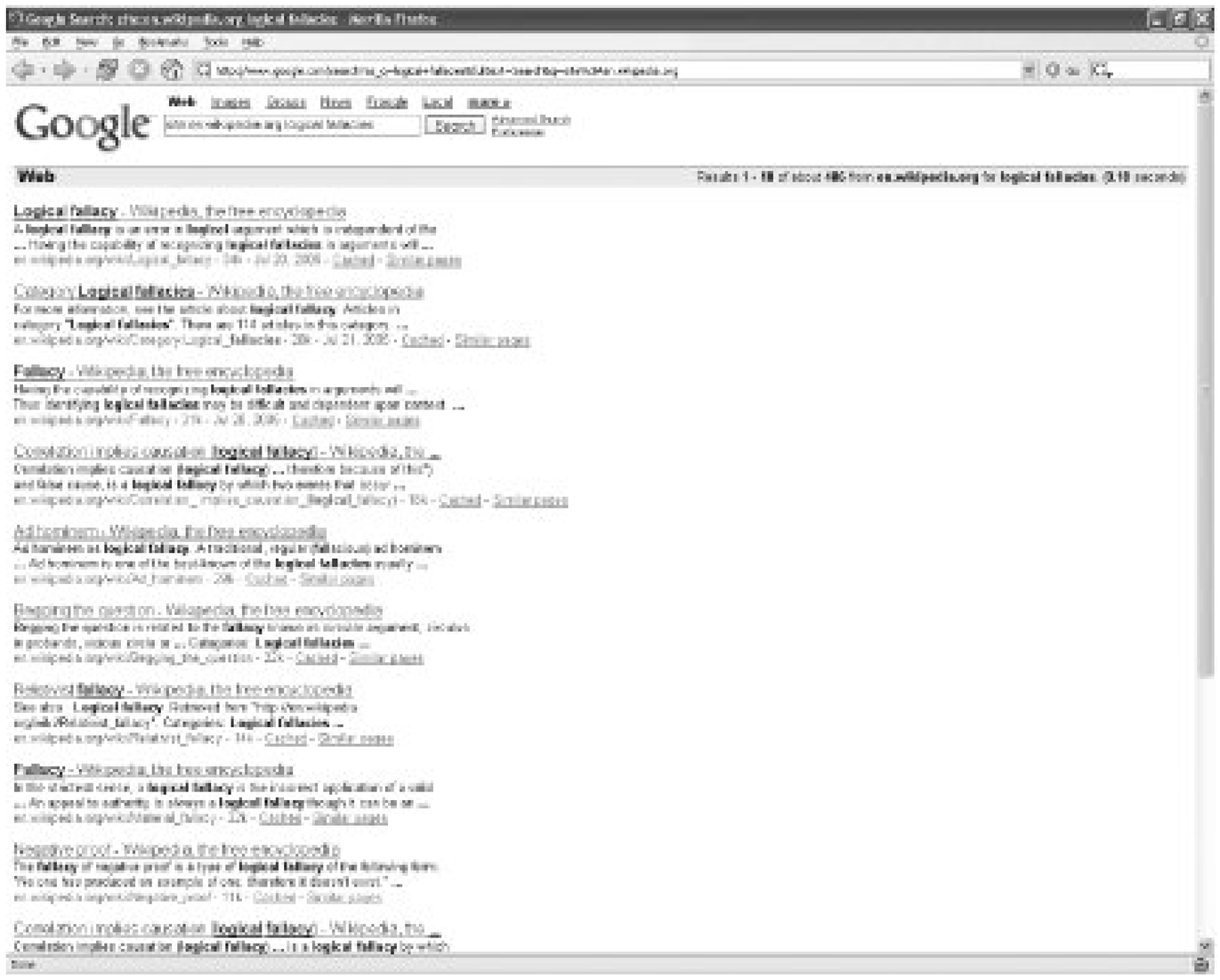
Running the Hack

After installing the user script (Tools→Install This User Script), visit http://randomness.org.uk/photos/index.cgi/months/may_2003. When you move your cursor over an image, you will see a tool tip displaying the filename of the image, as shown in Figure 1-33.

Figure 1-33. Image tool tips

Each image on the page is now a link to a Google Images search for images of the same name. This can lead to some pretty random results, as shown in Figure 1-34.

Figure 1-34. Other images named P5170003



Have fun exploring accidental cross-sections of the Web!

Mark Pilgrim

Hack 16. Track Stocks

A well-crafted Google query will usually net you company information beyond that provided by traditional stock services.

You can get a quick look at how a stock is performing by simply using a ticker symbol in the Google search form. For example, if you want to see how Google (the company) is faring during the day, type `GOOG` into Google, click Search, and you'll find some quick data, as shown in Figure 1-35.

Figure 1-35. Google quick stock data lookup

You'll see a recent stock price, data for the day, a chart showing recent performance, and links to more information at Google Finance, Yahoo! Finance, MSN Money, and other sites that track stocks. Click the ticker symbol or the chart to go to the Google Finance page for that stock, where you can compare dips and spikes in prices with company news, find background information on the company, and take part in discussions about the stock.

Beyond Google for Basic Stock Information

If you want a second opinion about stock performance, I recommend going straight to Yahoo! Finance (http://finance.yahoo.com) to quickly look up stocks by symbol or company name. There, you'll find all the basics: quotes, company profiles, charts, and recent news. For more in-depth coverage, I heartily recommend Hoovers (http://www.hoovers.com). Some of the information is free. For more depth, you must pay a subscription fee.

More Stock Research with Google

Try searching Google for:

```
"Tootsie Roll"
```

Now add the stock symbol, `tr`, to your query:

```
"Tootsie Roll" TR
```

Aha! Instantly, the search results shift to financial information. Now, add the name of the CEO:

```
"Tootsie Roll" TR "Melvin Gordon"
```

You end up with a nice, small, targeted list of results, as shown in Figure 1-36.

Figure 1-36. Using a stock symbol to limit results

Stock symbols are great "fingerprints" for Internet research. They're consistent, they often appear along with the company name, and they're usually enough to narrow your search results to relevant information.

There are also several words and phrases that you can use to narrow your search for company-related information. Replacing *company* with the name of the company you're looking for, try these:

- For press releases: `" company announced"`, `" company announces"`, `" company reported"`
- For financial information: *company* `"quarterly report"`, *company* `SEC`, *company* `financials`, *company* `"p/e ratio"`
- For location information: *company* `parking airport location` (doesn't always work but sometimes works amazingly well)

PREV

Chapter 2. Advanced Web

If you've just arrived from Chapter 1 and think you have more than enough information to Google yourself silly, hold on to your hat. It's time to put into high gear all you've learned about the ins and outs of Googling.

In this chapter, you'll measure Google Mindshare, range farther across the Web, twist and recombine your queries, squeeze the last drop of results out of every search, and even go beyond the bounds of Google's indexall without wearing out your fingers.

Because your computer will do the lion's share of the work for you.

This chapter hacks Google programmatically. Through bite-sized programs, we'll introduce you to the kind of trawling, crawling, and recombination that's possible with just a few lines of code. And it's all thanks to something called the Google APIthat's Application Programming Interface, or Google for computers.

In April 2002, Google introduced an alternate interface to the friendly search box on Google.com. It opened up its index to anyone with a little programming know-how and a reasonable amount of patience. Initially, this wasn't much to write home about. Some of the earliest applications simply Googled and incorporated the results into a web pageso-called Google boxes[Hack #19]. But as more people experimented with the API, the variety of applications grew from the marginally interesting to the seriously useful. And so was born the book you're holding in your hands.

This chapter, and the rest of this book, contain hacks that take advantage of this alternate interface. Some simply automate the sorts of tasks that might take you forever and a day to do by hand. Others run automatically to keep tabs on searchesand resultsof interest to you. And still others provide a bird's-eye view of your results in context, which is just not possible by eyeballing individual results pages.

Assumptions

These hacks demand a little more than an adventurous spirit and a researcher's tenacity. We assume you already have a programming background or are willing to learn the basics as you go along. In fact, we've been happy to hear about the many readers who picked up and learned a little programming through the hacks in the previous editions of this book; learning to program is so much easier if you have a particular task in mind.

You'll need to type in (or download) programs or scripts and run them from the command line (that's Terminal in Mac OS X or the DOS command window in Windows). Some are run as CGI scripts, which are bits of dynamic content running on your web site and talked to through your web browser. For more information on running hacks on the command line and as CGI scripts in your browser, see "How to Run the Hacks" in the Preface.

Almost all of the hacks are written in Perl (http://www.perl.com), with a few Python (http://www.python.org), PHP (http://www.php.net), Java (http://java.sun.com), and .NET (http://www.microsoft.com/net) programs sprinkled throughout. To run a particular hack, you'll need the appropriate language on your computer. Since instruction on installing and using these languages is beyond the scope of this book, you should start by visiting the language's home page and consider picking up a copy of one of O'Reilly's fine selection of books(http://www.oreilly.com). *Learning Perl* by Randal L. Schwartz and Tom Phoenix will be particularly useful.

Most of the hacks use the Google API. For an introduction to the programmatic side of Google, a detailed walkthrough of the Google API, and examples of programming Google using Perl, Python, PHP, Java, and .NET, turn to Chapter 8.

There are also a few hacks that involve *spidering*, or *screen scraping* which is essentially using your program to read a site's web pages and extract salient informationto get to data that is either not available through the Google API or is on another site entirely. If spidering appeals to you, you might want to check out *Spidering Hacks* by Kevin Hemenway and Tara Calishain (O'Reilly).

Hack 17. Assemble Advanced Search Queries

By understanding how Google Advanced Search URLs are structured, you can create your own

In addition to the simple web search form at http://www.google.com, Google offers an Advanced Web S results to a more useful list.

For example, if you want to find information about a generic topic, such as astronomy, you can go to Gc results, you can browse to the Advanced Web Search form, type `astronomy` , and limit the results by top-

Figure 2-1.

A search for `astronomy` across *.gov* sites limits results to pages at government sites such as NASA (http: gives you astronomy magazines at the top of the results.

You can further refine your search by limiting it to a specific file format, such as PDF files, Excel spreads preferences settings for language, number of results, and adult-content filtering for this search only.

If you're going to perform advanced searches on a regular basis, but you don't want to fill out the Googl together a Firefox Quick Search extension, or simply tame long Google URLs, you might also want to try

Anatomy of an Advanced Search URL

To get started with hacking URLs, type a term into the Advanced Web Search form and click the Google looks something like this:

```
http://www.google.com/search?as_q=astronomy&num=10&hl=en&btnG=Google+Search&as_epq=&as_
```

For any given search URL, some of the variables in the URL are redundant or unnecessary. The web forr understanding the pieces of the URL, you can construct your own queries using shorter URLs without the

Keep in mind that some characters like spaces can't be used within a URL and mu constructing a URL from scratch the phrase becomes `astronomy+magazine` in a URL

Note that the domain is followed by */search?* , and then followed by a series of variable/value pairs sepa keep the URLs as short as possible, the variables are a bit cryptic, so here's a list of the relevant variabl

The `as_` prefix probably lets Google know that the search came from the Advanced

`as_q`

Use this variable to indicate you're looking for all of the words in a particular query. The value `astr`

`as_epq`

This variable indicates that you want to match a specific phrase. So the value `astronomy+magazine`

`as_oq`

This variable indicates a search for any of the words in a particular query. So the value `astronomy+`

`as_eq`

This variable should include words that should not appear in any of the pages, and it must be used in the text.

`num`

The number of results is controlled by the `n` variable, and can be set to any number between 1 and

`as_qdr`

This variable can be set to limit results to pages that have been updated within a specific timefram

`as_occt`

This variable can be set to apply the primary query to a limited context. The default value `any` tells
you'd expect, they limit the search to document titles, bodies, URLs, and the text of links to a part

`as_sitesearch`

This variable tells Google to limit the search to a particular domain or top-level domain. So you ca

`safe`

Setting this variable to `active` turns on adult-content filtering for the results; by default, the result

`as_rights`

This variable lets you limit results to content licensed under copyright alternatives such as Creativ
The values are based on various types of licenses and are quite complex to assemble, so your bes

There are other variables in Advanced Search URLs, but these are a few that affect the content of searcl
your own advanced Google searches on the fly.

Building Advanced Search URLs

Now that you know the variables you can use, you can startwith the base URL (http://www.google.com/
want to search for the term *astronomy* across government domains, as in the first example. Your URL w

`http://www.google.com/search?as_q=`**`astronomy`**`&as_sitesearch=`**`.gov`**

Notice that this URL is quite a bit shorter than the one produced by the Advanced Search form. Say you

`http://www.google.com/search?as_q=astronomy&as_sitesearch=.gov&`**`num=100`**

Or imagine you want to find sites that contain the term *astronomy* , but not the term *telescope* :

`http://www.google.com/search?as_q=`**`astronomy`**`&as_eq=`**`telescope`**

Building your own Advanced Search URLs might seem a bit tedious when there are simple forms to do tl
different ways, and tweak the results without a trip to a special form.

PREV

Hack 18. Like a Version: Search with Synonyms

Gather a list of what Google thinks are synonyms for a keyword you provide.

The Google ~ synonym operator ["Special Syntax" in Chapter 1] widens your search criteria to include your search, but also your query words. For example, `food facts` might match only a handful of pages of seeks out nutrition information, cooking trivia, and more. And finding these synonyms is an entertaining of itself. Here's one way....

Suppose you're looking for all the synonyms for the word *car* . First, search Google for `~car` to find all the *car* . In its search results, Google highlights synonyms in bold, just as it highlights regular keyword matc page is shown in Figure 2-2) for `~car` finds *car* , *cars* , *motor* , *auto* , *vehicle* , and other synonyms in bc

Figure 2-2. Turning up boldfaced synonyms in Google search re

Now, let's focus on the synonyms rather than your original keyword, *car* . You do this by excluding the w `-car` . This saves you from having to wade through page after page of matches for the word *car* .

Once again, you scan the search results for new synonyms. (I ran across *automotive* , *racing* , *vehicle* , a

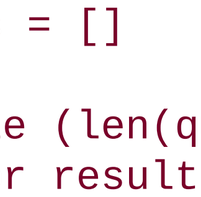
Make a note of any new boldfaced synonyms and subtract them from the query (e.g., `~car -car -automo`

until you hit Google's 32-word limit, after which Google ignores any additional words you tack on.

In the end, you'll have compiled a goodly list of synonyms, some of which you wouldn't have found in yo Google's algorithmic approach to synonyms.

The Code

If you think this all sounds a little tedious and more appropriate for a job description of a computer prog Python script to do all the iteration for you. It takes in a starting word and spits out a list of synonyms t



You'll need the PyGoogle [Hack #95] library to provide an interface to the Google API.

Save the following code as *synonyms.py* :

```
#!/usr/bin/python
# Available at http://www.aaronsw.com/2002/synonyms.py
import re
import google # get at http://pygoogle.sourceforge.net/
sb = re.compile('<b>(.*?)</b>', re.DOTALL)
def stripBolds(text, syns):
  for t in sb.findall(text):
    t = t.lower(  ).encode('utf-8')
    if t != '...' and t not in syns: syns.append(t)
  return syns
def findSynonyms(q):
  if ' ' in q: raise ValueError, "query must be one word"
  query = "~" + q
  syns = []

  while (len(query.split(' ')) <= 32):
    for result in google.doGoogleSearch(query).results:
      syns = stripBolds(result.snippet, syns)

    added = False
    for syn in syns:
      if syn in query: continue
      query += " -" + syn
      added = True
      break

    if not added: break # nothing left

  return syns
if __name_  _ == "_  _main_  _":
  import sys
  if len(sys.argv) != 2:
    print "Usage: python " + sys.argv[0] + " query"
```

```
  else:
    print findSynonyms(sys.argv[1])
```

Running the Hack

Call the script on the command line ["How to Run the Hacks" in the Preface], passing it a starting word

```
% python synonyms.py
                    car
```

You get back a list of synonyms like these:

```
['cars', 'car', 'bmw', 'auto', 'automotive', 'vehicle', 'car auto racing', 'motor', 'rac
```

In addition to an unconventional thesaurus, this method of gathering synonyms can be useful if you're t keywords [Hack #8] to use when researching a topic.

Aaron Swartz

Hack 19. Capture Google Results in a Google Box

Add a little box of Google results to any page on your web site.

A *Google box* is a small HTML snippet that shows Google search results for whatever you're searching for yours, or the top hits for a search that might be of interest to your readers.

Google boxes as a conceptthe idea of taking a shortened version of Google results andintegrating them i ubiquitous in blog and content management software. The Google box is easy to implement and was on lot of developers whip up a Google box just to see if they can. Do a Google search for`Google Box` to see s

What goes in a Google box, anyway? Why would anybody want to integrate them into a web page?

It depends on the page. Putting a Google box that searches for your name onto a blog provides a bit of right). If you have a topic-specific page, set up a Google box that searches for the topic (the more speci for the news topic. Google boxes can go pretty much anywhere because Google updates its index often

The Code

Here's a classic piece of Perl code to produce a Google box as a regular text file filled with garden-variet

```
#!/usr/local/bin/perl
# google_box.pl
# A classic Google box implementation.
# Usage: perl google_box.pl <query> <# results>

# Your Google API developer's key.
my $google_key='insert key here';

# Location of the GoogleSearch WSDL file.
my $google_wdsl = "./GoogleSearch.wsdl";

use strict;

use SOAP::Lite;

# Bring in those command-line arguments.
@ARGV == 2
  or die "Usage: perl googlebox.pl <query> <# results>\\n";
my($query, $maxResults) = @ARGV;
$maxResults = 10 if ($maxResults < 1 or $maxResults > 10);

# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
```

```
my $google_search = SOAP::Lite->service("file:$google_wdsl");

# Query Google.
my $results = $google_search ->
  doGoogleSearch(
    $google_key, $query, 0, $maxResults, "false", "",
    "false", "", "latin1", "latin1"
  );

# No results?
@{$results->{resultElements}} or die "no results";

print join "\\n",
  map( {
    qq{<a href="$_->{URL}">} .
    ($_->{title} || $_->{URL}) .
    qq{</a> <br />}
  } @{$results->{resultElements}} );
```

Save the code to a file called *google_box.pl* . Be sure to replace *insert key here* in the seventh line with

Running the Hack

This Google box takes two bits of information on the command line ["How to Run the Hacks" in the Prefa
don't provide the number of results, the Google box defaults to 10. Run it as follows:

```
% perl google_box.pl " query
                "  # of result
                s
```

where *query* is the search query you want to run against Google and *# of results* is the maximum num

This prints the results to the screen. To save them to a text file for inclusion in your web pages, specify

```
% perl google_box.pl " query "  # of results  >  google_box.html
```

You can leave out *# of results* , and the script defaults to 10 results in your Googl

Here's a sample Google box for `"camel book"` , a reference to O'Reilly's popular *Programming Perl* title:

```
<a href="http://www.oreilly.com/catalog/pperl3/">oreilly.com -- Online Catalog: Programm
<a href="http://www.oreilly.com/catalog/pperl2/">oreilly.com -- Online Catalog: Programm
```

```
<a href="http://www.catb.org/jargon/html/C/Camel-Book.html"><b>Camel Book</b></a> <br />
<a href="http://www.amazon.com/exec/obidos/tg/detail/-/1565921496?v=glance">Amazon.com:
<a href="http://www.amazon.com/exec/obidos/tg/detail/-/0596000278?v=glance">Amazon.com:
```

Integrating a Google Box

When you incorporate a Google box into your web page, you have two considerations: refreshing the co
content of the box, you need to run the program regularly using something like`cron` under Unix or the W

To include the content on your web page, Server Side Includes (SSI) is always effective. With SSI, inclu

```
<!-- #include virtual="./google_box.html" -->
```

For more information on using SSI, check out the NCSA SSI Tutorial (http://hoohoo.
`Tutorial` .

Google boxes are a nice addition to your web pages, whether you run a blog or a news site. But for man
search words.

The Official Google Box

If you're not comfortable writing your own code, there is a feature from Google that involves some simp
Google Related Links (http://www.google.com/relatedlinks/). The service puts a bit of dynamic HTML th
Links provide three distinct link types: searches, news, and web page links that are related to your site's

To add a Google Related Links box to your site, browse to the Related Links site (http://www.google.com
customize the size of the box, the types of links you want to show, and the colors. Finally, copy the code
Links box on a Blogger-powered blog.

Figure 2-3. A Google Relat

Readers can click on related searches, web pages, or news, depending on the types of links you selected
of the sizes fit your site, you might have to go the custom Google box route.

No matter which type of Google box you choose, Google can provide extended information for your read

Hack 20. Cook with Google

Learn the art of Google cooking to transform random ingredients in your fridge into a wonder

Google can help you find news, catalogs, discussions, web pages, and so much moreand it can also help tonight! One evening, Judy Hourihan was looking at some leftovers in her kitchen, trying to figure out w Swiss chard. She opened up Google, typed in the ingredients, and found a recipe that used both. She ca has spread from kitchen to kitchen like a favorite recipe.

While you could simply use ingredients in your kitchen to find recipes, this can be a time-consuming pro that offer recipes, and wading through all of the results for a set of ingredients would take longer than n

This hack shows how you can mix a dash of the Google API with a pinch of Perl to transform those rand dinner. Well, you do have to do some of the work. But it all starts with this hack.

The Code

This hack comes with a built-in form that calls the query and the recipe type, so there's no need to set u Google API, you need to make sure the `SOAP::Lite` (http://soaplite.com/) module and the common CGI (http://search.cpan.org/dist/CGI.pm/CGI.pm) are installed. You'll also need your own Google API key.

Save the following code to a file called *goocook.cgi* , and don't forget to include your own unique Google

```
#!/usr/bin/perl
# goocook.cgi
# Finding recipes with google.
# goocook.cgi is called as a CGI with form input.

# Your Google API developer's key
my $google_key='insert your Google API Key';

# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "your/path/to/GoogleSearch.wsdl";

use strict;
use SOAP::Lite;
use CGI qw/:standard/;

#Set Recipe Types
my %recipe_types = (
  "General" => "site:allrecipes.com | site:cooking.com | site:epicurious.com | site:reci
  "Vegetarian/Vegan"  => "site:living-foods.com",
  "Wordwide Cuisine" => "site:Britannia.org | inurl:thegutsygourmet | inurl:simpleinterr
```

```
);

#Initialize Error Handling
use CGI::Carp qw( fatalsToBrowser );
BEGIN {
    sub carp_error {
        my $error_message = shift;
        print "<pre>$error_message</pre>";
    }
    CGI::Carp::set_message( \\&carp_error );
}

#Print page HTML
print
  header(  ),
  start_html("GooCook"),
  h1("GooCook"),
  start_form(-method=>'GET'),
  'Ingredients: ', textfield(-name=>'ingredients'),
  br(  ),
  'Recipe Type: ', popup_menu(-name=>'recipe_type',
    -values=>[keys %recipe_types], -default=>'General'),
  br(  ),
  submit(-name=>'submit', -value=>"Get Cookin'!"),
  submit(-name=>'reset', -value=>"Start Over"),
  end_form(  ), p(  );

#Do a Google Search with parameters sent
if (param('ingredients')) {
  my $google_search = SOAP::Lite->service("file:$google_wsdl");

  my $results = $google_search->doGoogleSearch(
      $google_key,
      param('ingredients') . " " . $recipe_types{param('recipe_type')},
      0, 10, "false", "",  "false", "", "latin1", "latin1"
    );

  @{$results->{'resultElements'}} or print "None";
  foreach (@{$results->{'resultElements'}}) {
    print p(
      b($_->{title}||'no title'), br(  ),
      a({href=>$_->{URL}},$_->{URL}), br(  ),
      i($_->{snippet}||'no snippet')
    );
    print "\\n\\n";
  }
}

print end_html(  );
```

As you can see, the `%recipe_types` variable sets several parameters to narrow search results to specific

you're looking for. So, instead of simply searching Google for the keyword`salmon` , your choice of "World searches for `salmon` at specific sites, or at sites with certain recipe-related keywords in their URLs.

Running the Hack

This hack runs as a CGI script, producing a dynamic web page alongside the rest of the pages in your w run CGI scripts varies from server to server and ISP to ISP, you may need to ask your administrator or assembling your own four-course meal, but in the end, you'll have your own Google-powered recipe eng

Once the script is in place, call it by pointing your web browser at the file's location on your web server,

`http://example.com/goocook.cgi`

Once there, fill in your ingredients, select a recipe type, hit the Get Cookin'! button, and, with any luck, Figure 2-4 .

Figure 2-4. Recipes found with goocook.cgi

Take a look at the snippet and click the links to view recipes that look appetizing.

Hacking the Hack

Of course, the most obvious way to hack this hack is to add new cuisine options. This involves finding ne hack.

Adding new recipe sites entails finding the domains you want to search. Use the cooking section of the G

http://directory.google.com/Top/Home/Cooking/Recipe_Collections/ .

Next, find what you want and build it into a query supplement such as the one in the form, surrounded |
|. Remember, using the `site:` syntax means that you'll be searching for an entire domain, so if you find
http://www.geocities.com/reallygreat/food/recipes/ , don't use the `site:` syntax to search it; use the `inu`
(`inurl:geocities.com/reallygreat/food/recipes`). Just remember that an addition like this counts heav

Let's look at an example. The cookbook section of the Google Directory has a seafood section with sever
a constraint on your query:

```
(site:simplyseafood.com | site:coastangler.com | site: welovefish.com | site:sea-ex.com)
```

Next, test the query constraints live in Google by adding a query (in this case,`salmon`) and running it as

```
salmon (site:simplyseafood.com | site:coastangler.com | site:welovefish.c
```

Run a few different queries with a few different query words (`salmon` , `scallops` , whatever) and make su
Once you're confident that you have a good selection of recipes, add this new option to the hack:

```
my %recipe_types = (
  "General"           => "site:allrecipes.com | site:cooking.com | site:
epicurious.com | site:recipesource.com",
  "Vegetarian/Vegan"  => "site:fatfree.com | inurl:veganmania | inurl:
vegetarianrecipe | inurl:veggiefiles",
  "Wordwide Cuisine" => "site:Britannia.org | inurl:thegutsygourmet |
inurl:simpleinternet | inurl:soupsong"
);
```

Simply add the name you want to call the option (a`=>`) and the search string. Make sure you add it befc
Your code should look something like the code shown next:

```
my %recipe_types = (
  "General"           => "site:allrecipes.com | site:cooking.com | site:
epicurious.com | site:recipesource.com",
  "Vegetarian/Vegan"  => "site:fatfree.com | inurl:veganmania | inurl:
vegetarianrecipe | inurl:veggiefiles",
  "Wordwide Cuisine" => "site:Britannia.org | inurl:thegutsygourmet |
inurl:simpleinternet | inurl:soupsong"
  "Seafood" =>  "site:simplyseafood.com | site:baycooking.com | site :coastangler.com |

);
```

You can add as many search sets to the hack as you want. You may want to add Chinese Cooking, Dess
options.

As with regular cooking, Google cooking requires time, patience, and the proper ingredients. With your C
know you can always find the right recipe for the ingredients you have on hand. Bon appetit!

Tara Calishain and Judy Hourihan

PREV

Hack 21. Permute a Query

Run all permutations of query keywords and phrases to squeeze the last drop of results from

Google, ah, Google. A search engine of over eight billion pages and zillions of possible results. If you're a things are more entertaining than trying various tweaks to your Google search to see what exactly make

It's amazing what makes a difference. For example, you wouldn't think that word order would make mu buried in Google's documentation is the admission that the word order of a query impacts search results queries `three blind mice` , `blind mice tHRee` , and `mice tHRee blind` will give you different sites. These di phrase can lead down entirely different paths.

While knowing that word order affects results is an interesting thought, who has time to generate and ru multiword query? Google API to the rescue! This hack takes a query of up to four keywords or*quoted ph* runs all possible permutations, showing result counts by permutation and the top results for each permu

The Code

This hack uses two nonstandard Perl modules that you'll need. The`Algorithm::Permute` module can be fo

http://search.cpan.org/search?query=algorithm%3A%3Apermute&mode=all

and finds different permutations for a query; the`Number::Format` module, which can be found at:

http://search.cpan.org/~wrw/Number-Format-1.45/Format.pm

adds commas to the count totals to make them easier to read.

Save the following code as a CGI script ["How to Run the Hacks" in the Preface] named *order_matters.c* being sure to replace *`insert key here`* with your Google API key:

```
#!/usr/local/bin/perl
# order_matters.cgi
# Queries Google for every possible permutation of up to 4 query keywords,
# returning result counts by permutation and top results across permutations.
# order_matters.cgi is called as a CGI with form input

# Your Google API developer's key
my $google_key='insert key here';

# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

use strict;
```

```
use SOAP::Lite;
use CGI qw/:standard *table/;
use Algorithm::Permute;
use Number::Format qw(:subs);

#Initialize Error Handling
use CGI::Carp qw( fatalsToBrowser );
BEGIN {
    sub carp_error {
        my $error_message = shift;
        print "<pre>$error_message</pre>";
    }
    CGI::Carp::set_message( \\&carp_error );
}

print
  header(  ),
  start_html("Order Matters"),
  h1("Order Matters"),
  start_form(-method=>'GET'),
  'Query:   ', textfield(-name=>'query'),
  '   ',
  submit(-name=>'submit', -value=>'Search'), br(  ),
  '<font size="-2" color="green">Enter up to 4 query keywords or "quoted phrases"</font>
  end_form(  ), p(  );

if (param('query')) {

 # Glean keywords.
 my @keywords = grep !/^\\s*$/,  split /([+-]?".+?")|\\s+/, param('query');

 scalar @keywords > 4 and
  print('<font color="red">Only 4 query keywords or phrases allowed.</font>'), last;

 my $google_search = SOAP::Lite->service("file:$google_wsdl");

 print
  start_table({-cellpadding=>'10', -border=>'1'}),
 Tr([th({-colspan=>'2'}, ['Result Counts by Permutation' ])]),
 Tr([th({-align=>'left'}, ['Query', 'Count'])]);

my $counts = {}; # keep track of permutation results counts
my $results = {}; # keep track of what we've seen across queries

# Iterate over every possible permutation.
my $p = new Algorithm::Permute( \\@keywords );
 #while (my @query = $p->next) {
 #  print join(", ", @res), "<br />\\n";
 #}

 while (my $query = join(' ', $p->next)) {
```

```
# Query Google.
my $r = $google_search ->
 doGoogleSearch(
  $google_key,
  $query,
  0, 10, "false", "",  "false", "", "latin1", "latin1"
 );
$counts->{$query} = {
    count => $r->{'estimatedTotalResultsCount'}
};
#print Tr([td({-align=>'left'}, [a({href=>$query_url},$query), format_number($r->{'es
@{$r->{'resultElements'}} or next;

#print Dumper($r->{'resultElements'});
#die;

# Assign a rank.
  # Assign a rank.
  my $rank = 10;
  foreach (@{$r->{'resultElements'}}) {
   $results->{$_->{URL}} = {
    title => $_->{title},
    snippet => $_->{snippet},
    seen => ($results->{$_->{URL}}->{seen}) + $rank
   };
   $rank--;
  }
}

# Show top count first
foreach ( sort { $counts->{$b}->{count} <=> $counts->{$a}->{count} } keys %$counts ) {
  my $query_url = "http://www.google.com/search?q=$_";
  print Tr([td({-align=>'left'}, [a({href=>$query_url},$_), format_number($counts->{$_}-
}

print
  end_table(  ), p(  ),
  start_table({-cellpadding=>'10', -border=>'1'}),
  Tr([th({-colspan=>'2'}, ['Top Results across Permutations' ])]),
  Tr([th({-align=>'left'}, ['Score', 'Result'])]);

foreach ( sort { $results->{$b}->{seen} <=> $results->{$a}->{seen} } keys %$results ) {
  print Tr(td([
    $results->{$_}->{seen},
    b($results->{$_}->{title}||'no title') . br(  ) .
    a({href=>$_}, $_) . br(  ) .
    i($results->{$_}->{snippet}||'no snippet')
  ]));
}

  print end_table(  ),
}
```

```
print end_html(  );
```

Running the Hack

Point your web browser at the CGI script *order_matters.cgi* on your web server. Enter the query you wa
phrases). The script first searches for every possible combination of the search words and phrases, as sh

Figure 2-5. Permutations for Perl syntax hacks

The script then displays the top 10 search results across all permutations of the query, as shown in Figu

Figure 2-6. Top results for permutations of Perl syntax

At first blush, this hack looks like a novelty with few practical applications. But if you're a regular resear
of interest.

If you're a regular researcherthat is, there are certain topics you research on a regular basisyou might w
and see if you can detect a pattern in how your regular search terms are impacted by changing word or
searching so that certain words always come first or last in your query.

If you're a web publisher, you need to know where your page appears in Google's search results. If your
because of a shift in a query arrangement, you may want to add more words to your text or shift your e

Hack 22. Summarize Results by Domain

Get an overview of the sorts of domains (educational, commercial, foreign, and so forth) foun the results of a Google query.

You want to know about a topic, so you do a search. But what do you have? A list of pages. You can't ge good idea of the types of pages these are without taking a close look at the list of sites.

This hack is an attempt to get a *snapshot* of the types of sites that result from a query. It does this by ta *suffix census* , a count of the different domains that appear in search results.

This is most ideal for running `link:` queries, providing a good idea of what kinds of domains (commercia educational, military, foreign, etc.) link to a particular page.

You can also run it to see where technical terms, slang terms, and unusual words are turning up. Which mention a particular singer more often? Or a political figure? Does the word *astronomy* come up more oft *.com* or *.edu* sites?

Of course, this snapshot doesn't provide a complete inventory, but as overviews go, it's rather interestir

The Code

Save the code as suffixcensus.cgi, a CGI script ["How to Run the Hacks" in the Preface] on your web se

```
#!/usr/local/bin/perl
# suffixcensus.cgi
# Generates a snapshot of the kinds of sites responding to a
# query. The suffix is the .com, .net, or .uk part.
# suffixcensus.cgi is called as a CGI with form input.

# Your Google API developer's key.
my $google_key='insert key here';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Number of times to loop, retrieving 10 results at a time.
my $loops = 10;

use strict;
use SOAP::Lite;
use CGI qw/:standard *table/;
```

```
print
  header(  ),
  start_html("SuffixCensus"),
  h1("SuffixCensus"),
  start_form(-method=>'GET'),
  'Query: ', textfield(-name=>'query'),
  '   ',
  submit(-name=>'submit', -value=>'Search'),
  end_form(  ), p(  );

if (param('query')) {
  my $google_search  = SOAP::Lite->service("file:$google_wsdl");
  my %suffixes;

  for (my $offset = 0; $offset <= $loops*10; $offset += 10) {

    my $results = $google_search ->
      doGoogleSearch(
        $google_key, param('query'), $offset, 10, "false", "",  "false",
        "", "latin1", "latin1"
      );

    last unless @{$results->{resultElements}};

    map { $suffixes{ ($_->{URL} =~ m#://.+?\\.(\\w{2,4})/#)[0] }++ }
      @{$results->{resultElements}};
  }

  print
    h2('Results: '), p(  ),
    start_table({cellpadding => 5, cellspacing => 0, border => 1}),
    map( { Tr(td(uc $_),td($suffixes{$_})) } sort keys %suffixes ),
    end_table(  );
}

print end_html(  );
```

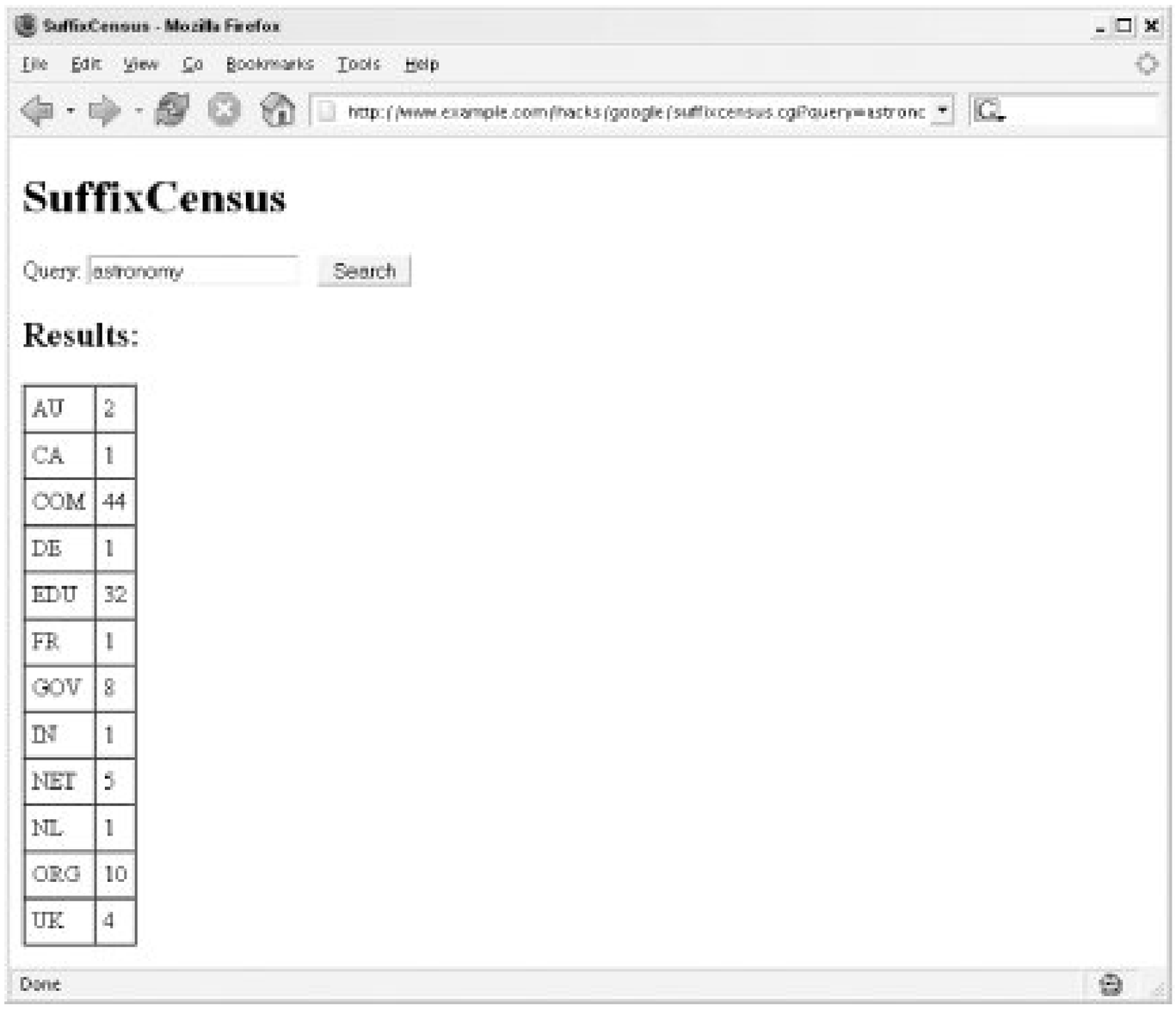
Be sure to replace *insert key here* with your Google API key.

Running the Hack

This hack runs as a CGI script ["How to Run the Hacks" in the Preface]. Point your browser at *suffixcens* to run it.

Searching for the prevalence of *astronomy* by suffix finds that the most mentions are split between *.com* and *.edu* s, as shown in Figure 2-7 .

Figure 2-7. Prevalence of "astronomy" by domain suffix



Hacking the Hack

There are a couple of ways to hack this hack.

Going back for more

This script, by default, visits Google 10 times, grabbing the top 100 (or fewer, if there aren't that many) results. To increase or decrease the number of visits, simply change the value of the `$loops` variable at t of the script. Bear in mind, however, that making `$loops = 50` might net you 500 results, but this also ea quickly into your daily allotment of 1,000 Google API queries.

Returning comma-separated output

It's rather simple to adjust this script to run from the command line and returna comma-separated outp suitable for Excel or your average database. Remove the starting HTML, form, and ending HTML output, alter the code that prints the results. In the end, you come to something like this (changes in bold):

```
#!/usr/local/bin/perl
# suffixcensus_csv.pl
# Generates a snapshot of the kinds of sites responding to a
# query. The suffix is the .com, .net, or .uk part.
# Usage: perl suffixcensus_csv.pl query="your query" > results.csv

# Your Google API developer's key
my $google_key='1BcTFcWyRzzIb/dggoXyAB5KjOFYUtjE';
```

```
# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "e:/onfocus/web/hacks/google/GoogleSearch.wsdl";

# Number of times to loop, retrieving 10 results at a time.
my $loops = 10;

use SOAP::Lite;
use CGI qw/:standard/;
     param('query')
                    or die qq{usage: suffixcensus_csv.pl query="{query}" [> results.csv]
                       print qq{"suffix","count"\\n};

my $google_search  = SOAP::Lite->service("file:$google_wsdl");

my %suffixes;

for (my $offset = 0; $offset <= $loops*10; $offset += 10) {

  my $results = $google_search ->
    doGoogleSearch(
      $google_key, param('query'), $offset, 10, "false", "",  "false",
      "", "latin1", "latin1"
    );

  last unless @{$results->{resultElements}};

  map { $suffixes{ ($_->{URL} =~ m#://.+?\\.(\\w{2,4})/#)[0] }++ }
    @{$results->{resultElements}};
}
     print map { qq{"$_", "$suffixes{$_}"\\n} } sort keys %suffixes;
```

Invoke the script from the command line like so:

```
$ perl suffixcensus_csv.pl query="insert query" > results.csv
```

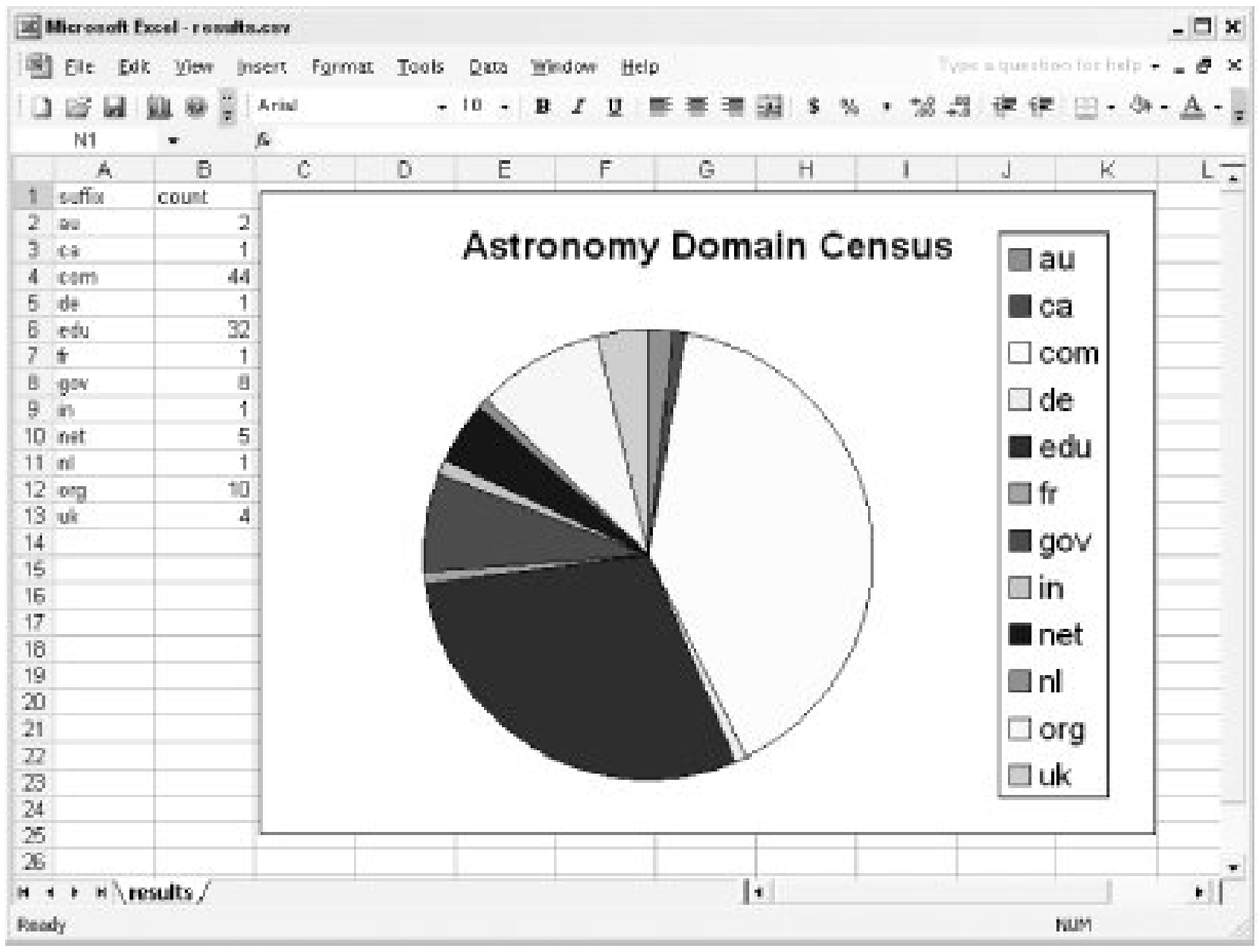
Searching for mentions of *astronomy* , sending the output straight to the screen rather than to a *results.* file, looks like this:

```
$ perl suffixcensus_csv.pl query="astronomy"
"suffix","count"
"au", "2"
"ca", "1"
"com", "44"
"de", "1"
"edu", "32"
"fr", "1"
"gov", "8"
```

```
"in", "1"
"net", "5"
"nl", "1"
"org", "10"
"uk", "4"
```

If you do pipe the results to a CSV file, it's easy to open those results in a spreadsheet program such as and graph them for easy visualization, as shown in Figure 2-8.

Figure 2-8. Graphing the distribution of "astronomy" across domains



suffix	count
au	2
ca	1
com	44
de	1
edu	32
fr	1
gov	8
in	1
net	5
nl	1
org	10
uk	4

Knowing the distribution of a certain keyword across different domains can give you new insights into a you're researching or point out specific areas you could be targeting in your searches.

Hack 23. Measure Google Mindshare

Measure the Google mindshare of a particular person within a topic area.

Based on an idea by author Steven Johnson (http://www.stevenberlinjohnson.com), this hack determines the Google *mindshare* of a person within a particular set of Google queried keywords. What's Willy Wonka's Google mindshare of "Willy"? What percentage of "weatherman" does Al Roker hold? Who has the greater "The Beatles" Google mindshare, Ringo Starr or Paul McCartney? More importantly, what Google mindshare of your industry does your company own? Or even closer to home, what mindshare of a topic do you own?

Google mindshare is calculated as follows. Determine the size of the result set for a keyword or phrase. Determine the result set size for that query along with a particular person. Divide the second by the first and multiply by 100, yielding the percentage of Google mindshare. For example, a query for `Willy` yields about 49,700,000 results. `"Willy Wonka" +Willy` finds 66,350,000. We can concludehowever unscientificallythat Willy Wonka holds roughly a 13 percent ((66,350,000/49,700,000)x100 = ~ 13.35%) Google mindshare of "Willy."

Sure, it's a little silly, but there's probably a grain of truth in it somewhere.

The Code

This hack does all the heavy lifting for you by placing this logic within a Perl script and calculating mindshare. You'll need the `SOAP::Lite` module and a Google API key.

Once you have the prerequisites, save the following code as a CGI script ["How to Run the Hacks" in the Preface] called *google_mindshare.cgi* in your web site's *cgi-bin* directory:

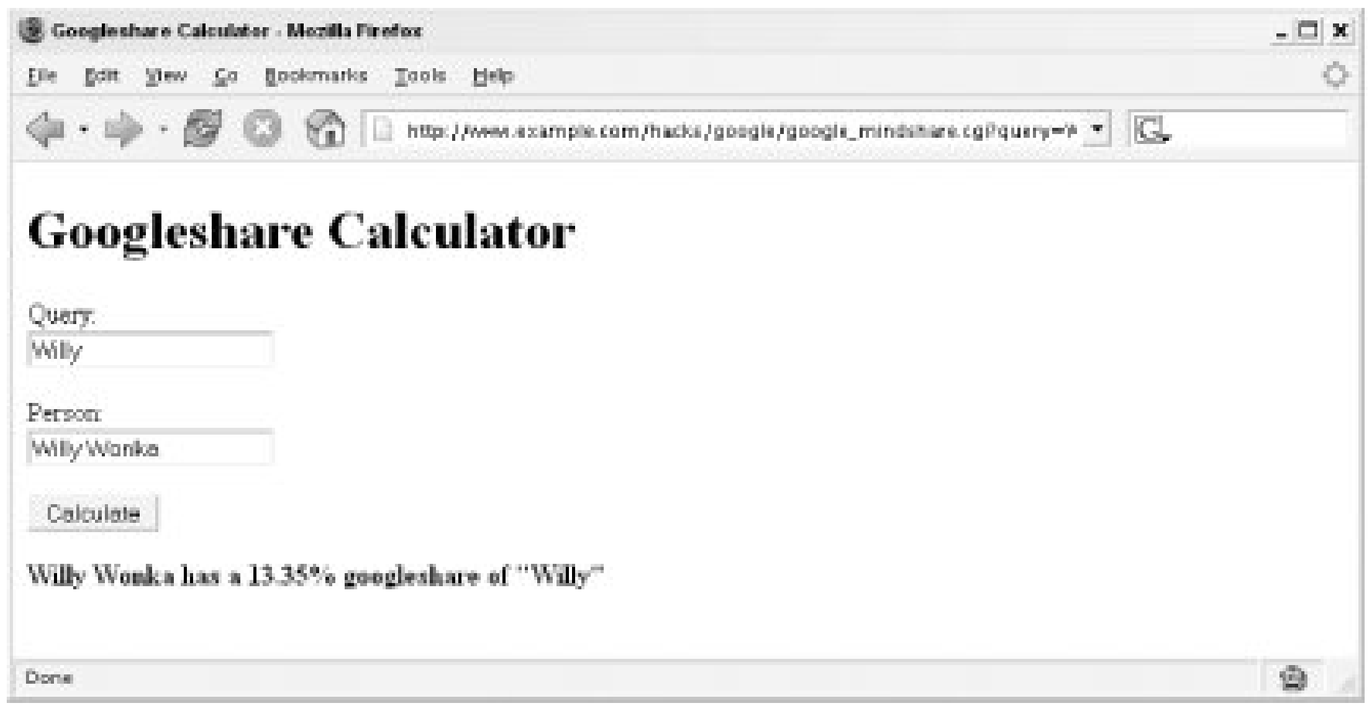
```
#!/usr/local/bin/perl
# google_mindshare.cgi
# This implementation by Rael Dornfest,
# http://raelity.org/blog/articles/2002/11/16/googleshare
# Based on an idea by Steven Johnson,
# http://web.archive.org/web/20021120091833/
# www.stevenberlinjohnson.com/movabletype/archives/000009.html

# Your Google API developer's key.
my $google_key='insert key here';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";
```

```
use SOAP::Lite;
use CGI qw/:standard *table/;

print
  header(  ),
  start_html("Googleshare Calculator"),
  h1("Googleshare Calculator"),
  start_form(-method=>'GET'),
  'Query: ', br(  ), textfield(-name=>'query'),
  p(  ),
  'Person: ',br(  ), textfield(-name=>'person'),
  p(  ),
  submit(-name=>'submit', -value=>'Calculate'),
  end_form(  ), p(  );

if (param('query') and param('person')) {
  my $google_search  = SOAP::Lite->service("file:$google_wsdl");

# Query Google for they keyword, keywords, or phrase.
my $results = $google_search ->
  doGoogleSearch(
    $google_key, '"'.param('query').'"', 0, 1, "false", "",  "false",
    "", "latin1", "latin1"
  );

# Save the results for the Query.
my $query_count = $results->{estimatedTotalResultsCount};

my $results = $google_search ->
  doGoogleSearch(
    $google_key, '+"'.param('query').'" +"'.param('person').'"', 0, 1,
    "false", "",  "false", "", "latin1", "latin1"
  );
# Save the results for the Query AND Person.
my $query_person_count = $results->{estimatedTotalResultsCount};

print
  p(
    b(sprintf "%s has a %.2f%% googleshare of %s",
      param('person'),
      ($query_person_count / $query_count * 100),
      '"'.param('query').'"'
    )
  )
}

print end_html(  );
```

Running the Hack

Visit the CGI script in your browser. Enter a query and a person. The name doesn't necessarily have to be a person's full name; it can be a company, location, proper noun, or just about anything, actually. Click the Calculate button and enjoy. Figure 2-9 shows the Willy Wonka example.

Figure 2-9. Google mindshare for Willy Wonka



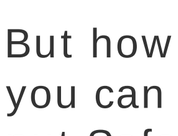
Fun Hack Uses

You can't do too many practical things with this hack, but you can have a lot of fun with it. Playing *unlikely percentages* is fun; see if you can find a name/word combo that gets a higher percentage than other percentages that you would consider more likely. Here are the answers to the questions posted at the beginning of this hack, and more:

- Willy Wonka has a 13.35 percent Google mindshare of "Willy."
- Al Roker has a 1.05 percent Google mindshare of "weatherman."
- Ringo Starr has a 3.70 percent Google mindshare of "The Beatles."
- Paul McCartney has a 10.54 percent Google mindshare of "The Beatles."
- Linus Torvalds has a 1.52 percent Google mindshare of "Linux."
- Microsoft has a 66.79 percent Google mindshare of "Linux."

Another entertaining way to tap into the collective mind with Google is to pit two search queries against each other to see which query is more popular [Hack #26].

Hack 24. SafeSearch Certify URLs



Feed URLs into Google's SafeSearch to determine whether they point at questionable content.

Only three things in life are certain: death, taxes, and accidentally visiting a once family-safe web sit that now contains text and images that would make a horse blush.

As you probably know if you've ever put up a web site, domain names are registered for finite length of time. Sometimes registrations accidentally expire; sometimes businesses fold and allow the registrations to expire; sometimes other companies take them over.

Other companies want just the domain name, some companies want the traffic the defunct site generated, and, in a few cases, the new owners of the domain name hold it hostage, offering to sell i back to the original owners for a great deal of money. (This doesn't work as well as it used to because of the dearth of Internet companies that actually have a great deal of money.)

When a site isn't what it once was, that's no big deal. When it's not what it once was and is now X-rated, that's a bigger deal. When it's not what it once was, is now X-rated, and is on the link list of a site you run, that's a really big deal.

But how to keep up with all the links? You can visit each link periodically to determine if it's still okay you can wait for hysterical emails from site visitors, or you can just not worry about it. Or you can put SafeSearch to work through the Google API.

SafeSearch is the term Google applies to itsadult-content filtering mechanism available in your Google preferences. To try out SafeSearch for yourself, browse to the main Google page and click the Preferences link to the right of the query box. On the Preferences page, scroll down to the section labeled SafeSearch Filtering and choose your filtering level. By default, Google uses*moderate* filtering, which means that some images, but not sites, are removed from your searches. You have the option to use *strict* filtering, which removes adult material from your searches, or*no* filtering.

You can go back and forth between the preferences page and search results to see which sites show up in the results with each setting, but that's a tedious task that's best left to code.

The Code

This hack lets you check a list of URLs in Google's SafeSearch mode. If they appear in the SafeSearcl mode, they're probably okay. If they don't appear there, they're either not in Google's index or not "safe" enough to pass through Google's filter. The program then checks the URLs missing from a SafeSearch with a nonfiltered search. If they do not appear in a nonfiltered search, they're labeled as `unindexed`. If they do appear in a nonfiltered search, they're labeled as`suspect`.

Save the following Perl source code as a text file named *suspect.pl*:

```
#!/usr/local/bin/perl
# suspect.pl
# Feed URLs to a Google SafeSearch. If inurl: returns results, the
# URL probably isn't questionable content. If inurl: returns no
# results, either it points at questionable content or isn't in
# the Google index at all.

# Your Google API developer's key.
my $google_key = 'put your key here ';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

use strict;
use SOAP::Lite;

$|++; # turn off buffering

my $google_search = SOAP::Lite->service("file:$google_wsdl");

# CSV header
print qq{"url","safe/suspect/unindexed","title"\\n};

while (my $url = <>) {
  chomp $url;
  $url =~ s!^\\w+?://!!;
  $url =~ s!^www\\.!!;

  # SafeSearch
  my $results = $google_search ->
      doGoogleSearch(
      $google_key, "inurl:$url", 0, 10, "false", "",  "true ",
      "", "latin1", "latin1"
    );

  print qq{"$url",};

  if (grep /$url/, map { $_->{URL} } @{$results->{resultElements}}) {
    print qq{"safe"\\n};
  }
  else {
    # unSafeSearch
    my $results = $google_search ->
        doGoogleSearch(
        $google_key, "inurl:$url", 0, 10, "false", "",  "false ",
        "", "latin1", "latin1"
      );

    # Unsafe or Unindexed?
    print (
```

```
        (scalar grep /$url/, map { $_->{URL} } @{$results->{resultElements}})
          ? qq{"suspect"\\n}
          : qq{"unindexed"\\n}
        );
    }
}
```

Note that the difference between the safe and unsafe search in this code is the seventh parameter of the `doGoogleSearch` API method. If this parameter is set to `TRue`, Google returns only sites that are deemed "safe" by SafeSearch. The value `false` lets everything through.

Running the Hack

To run the hack, you'll need a text file that contains the URLs you want to check, one line per URL. For example:

```
http://www.oreilly.com/catalog/essblogging/
http://www.playboy.com/
hipporhinostricow.com
```

The program runs from the command line ["How to Run the Hacks" in the Preface]. Enter the name of the script, a less-than sign, and the name of the text file that contains the URLs you want to check. The program returns results that look like this:

```
% perl suspect.pl < urls.txt
"url","safe/suspect/unindexed"
"oreilly.com/catalog/essblogging/","safe"
"http://www.playboy.com/","suspect"
"hipporhinostricow.com","unindexed"
```

The first item is the URL being checked, and the second is its probable safety rating, as follows:

`safe`

: The URL appeared in a Google SafeSearch for the URL.

`suspect`

: The URL did not appear in a Google SafeSearch but did in an unfiltered search.

`unindexed`

: The URL did not appear in either a SafeSearch or an unfiltered search.

You can redirect output from the script to a file for import into a spreadsheet or database:

```
% perl suspect.pl < urls.txt > urls.csv
```

Hacking the Hack

You can use this hack interactively, feeding it URLs one at a time. Invoke the script with `perl suspect.pl`, but don't feed it a text file of URLs to check. Enter a URL and hit the Return key on your keyboard. The script replies in the same manner it does when fed multiple URLs. This is handy when you just need to spot-check a couple of URLs on the command line. When you're ready to quit, break out of the script using Ctrl-D under Unix or Ctrl-Break on a Windows command line.

Here's a transcript of an interactive session with *suspect.pl*:

```
% perl suspect.pl
"url","safe/suspect/unindexed","title"
http://www.oreilly.com/catalog/essblogging/
"oreilly.com/catalog/essblogging/","safe"
http://www.playboy.com/
"xxxxxxxxxx.com/preview/home.htm","suspect"
hipporhinostricow.com
"hipporhinostricow.com","unindexed"
```

While Google's SafeSearch filter is good, it's not infallible. (I have yet to see an automated filtering system that is.) So, if you run a list of URLs through this hack and they all show up in a SafeSearch query, don't take that as a guarantee that they're all completely inoffensive. Merely take it as a pretty good indication that they are. If you want absolute assurance, you'll have to visit every link personally and frequently.

Here's a fun idea if you need an Internet-related research project. Take 500 or so domain names at random and run this program on the list once a week for several months, saving the results to a file each time. It'd be interesting to see how many domains/URLs end up being filtered out of SafeSearch over time.

Hack 25. Search Google Topics

Run queries against some of the available Google API specialty topics.

Google doesn't talk about it much, but it does make specialty web searches available. And I'm not just talking about searches limited to a certain domain. I'm talking about searches devoted to a particular topic (http://www.google.com/options/specialsearches.html). You'll find four specialty searches related to technology, one that limits results to U.S. government sites, and hundreds of specialty searches limited to specific universities across the U.S. The Google API makes four of these searches available: the U.S. Government, Linux (an alternative operating system), BSD (Berkeley Software Distribution, another alternative operating system), and Macintosh (your friendly Apple cult).

In this hack, we'll look at a program that takes a query and provides a count of results in each specialty topic, as well as the top results for each topic.

Why Topic Search?

Why would you want to topic search? Because Google currently indexes billions of pages. If you try t do more than very specific searches, you might find yourself with far too many results. If you narrow your search by topic, you can get good results without having to zero in on your search.

You can also use it to do some decidedly unscientific research. Which topic contains more iterations c the phrase "open source"? Which contains the most pages from *.edu* (educational) domains? Which topic, Macintosh or FreeBSD, has more on user interfaces? Which topic holds the most for Monty Python fans?

The Code

As with many hacks in this book, the `SOAP::Lite` (http://soaplite.com) module must be installed. Save the following code as a CGI script ["How to Run the Hacks" in the Preface] named *gootopic.cgi* in the *cgi-bin* directory on your web server:

```
#!/usr/local/bin/perl
# gootopic.cgi
# Queries across Google Topics (and All of Google), returning
# number of results and top result for each topic.
# gootopic.cgi is called as a CGI with form input

# Your Google API developer's key
my $google_key='insert key here';
```

```
# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Google Topics
my %topics = (
  search   => 'All of Google',
  unclesam => 'U.S. Government',
  linux    => 'Linux',
  mac      => 'Macintosh',
  bsd      => 'FreeBSD'
);

use strict;

use SOAP::Lite;
use CGI qw/:standard *table/;

# Display the query form.
print
  header(  ),
  start_html("GooTopic"),
  h1("GooTopic"),
  start_form(-method=>'GET'),
  'Query: ', textfield(-name=>'query'), '   ',
  submit(-name=>'submit', -value=>'Search'),
  end_form(  ), p(  );

my $google_search  = SOAP::Lite->service("file:$google_wsdl");

# Perform the queries, one for each topic area.
if (param('query')) {
  print
    start_table({-cellpadding=>'10', -border=>'1'}),
    Tr([th({-align=>'left'}, ['Topic', 'Count', 'Top Result'])]);

  foreach my $topic (keys %topics) {

    my $results = $google_search ->
      doGoogleSearch(
        $google_key, param('query'), 0, 10, "false", $topic,  "false",
        "", "latin1", "latin1"
      );

    my $result_count = $results->{'estimatedTotalResultsCount'};

    my $top_result = 'no results';

    if ( $result_count ) {
      my $t = @{$results->{'resultElements'}}[0];
      $top_result =
        b($t->{title}||'no title') . br(  ) .
```

```
            a({href=>$t->{URL}}, $t->{URL}) . br(  ) .
            i($t->{snippet}||'no snippet');
        }

        my $query_url = "http://www.google.com/$topic?q=" . param('query');

        # Output
        print Tr([ td([
          a({href=>$query_url}, $topics{$topic}) . br(  ),
          $result_count,
          $top_result
          ])
        ]);
      }

      print
        end_table(  ),
    }

    print end_html(  );
```

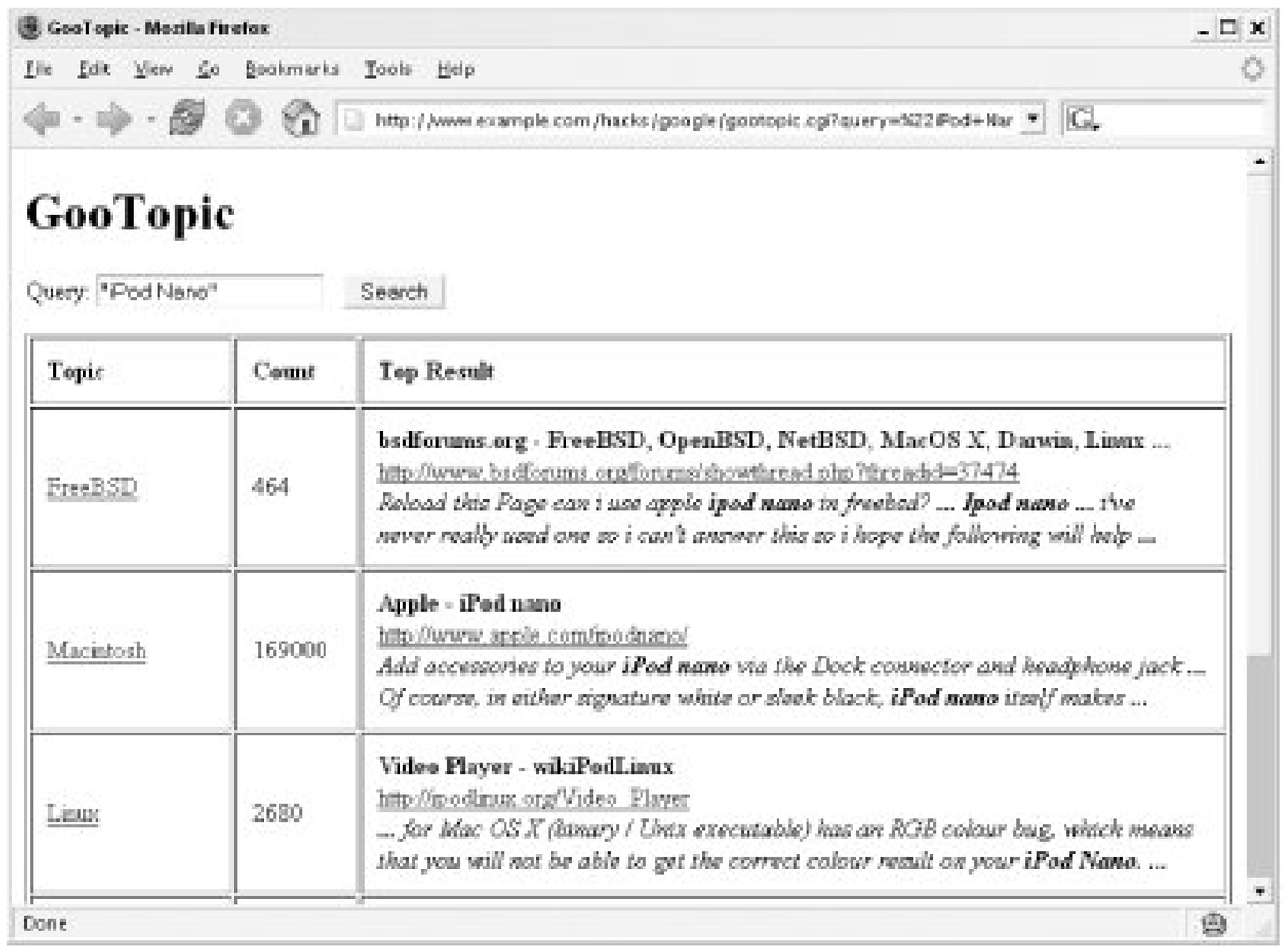
Be sure to replace *`insert key here`* with your Google API key.

Running the Hack

Point your web browser at *gootopic.cgi*.

Provide a query and the script searches each special topic area, providing you with an overall ("All of Google") count, topic area count, and the top result for each. Figure 2-10 shows a sample run for `"iPod Nano"`, with Macintosh (not surprisingly) coming out on top.

Figure 2-10. Topic search for "iPod Nano"



The topic title in the results is a link to all of the query results at Google. In this example, click Macintosh to see all of the "iPod Nano" (http://www.google.com/mac?q=%22iPod%20Nano%22) results at the Macintosh specialized search.

Search Ideas

Trying to figure out how many pages each topic finds for particular top-level domains (e.g.,*.com*, *.edu*, *.uk*) is rather interesting. You can query for `inurl:` *xx* `site:` *xx*, where *xx* is the top-level domain you're interested in. For example, `inurl:va site:va` searches for any of the Vatican's pages in the various topics (there aren't any). `inurl:mil site:mil` finds an overwhelming number of results in the U.S. Government special topicno surprise there.

See Also

Speaking of U.S. Government sources, Google has a customized version of its Homepage service filled with up-to-date sources of information about Uncle Sam. It's called the Google U.S. Government Search (http://www.google.com/ig/usgov), and, in addition to the specialty search feature, you'll find headlines from a variety of government sources.

Hack 26. Run a Google Popularity Contest

Put two terms, spelling variations, animals, vegetables, or minerals head to head in a Google contest.

What is the most popular word? Which spelling is more commonly used? Who gets more mentions, Fred and other equally critical questions are answered by Google Smackdown (http://www.onfocus.com/googl

Why would you want to compare search counts? Sometimes finding out which terms appear more often your queries better. Why use a particular word if it gets almost no results? Comparing misspellings can find terms or phrases. And sometimes it's just fun to run a popularity contest.

If you're just searching for keywords, Google Smackdown is very simple. Enter one word in each query developer's key [Chapter 8] if you have one, and click the "throw down!" button. Smackdown returns th approximate count of each search.

If you're planning to use a special syntax, you'll have to be more careful. Unfortunately, the `link:` syntax Interestingly, `phonebook:` does; do more people named Smith or Jones live in Boston, Mass.?

To use any special syntaxes, enclose the query in quotes: `"intitle:windows"` .

The next tip is a little backwards. If you want to specify a phrase, do not use quotes; Smackdown, by de phrase. If you want to search for the two words on one page but not necessarily as a phrase (`jolly` *and* `roger"`), do use quotes. The reason the special syntaxes and phrases work this way is because the prog encloses phrases in quotes. If you add quotes, you're sending a double-quoted query to Google (`"Google` into a double quote like this, it just strips out all the quotes.

> If you want to try a Google Smackdown without having to run it yourself, there's a live at http://www.onfocus.com/googlesmack/down.asp .

The Code

Google Smackdown is written for ASP pages running under the Windows operating system and Microsoft Server (IIS):

```
<%
'-----------------------------------------------------
' Set the global variable strGoogleKey.
'-----------------------------------------------------
Dim strGoogleKey
```

```
strGoogleKey = "insert your key"

'-------------------------------------------------------
' The function GetResult(  ) is the heart of Google Smackdown.
' It queries Google with a given word or phrase and returns
' the estimated total search results for that word or phrase.
' By running this function twice with the two words the user
' enters into the form, we have our Smackdown.
'-------------------------------------------------------
Function GetResult(term)

 '-------------------------------------------------------
 ' Set the variable the contains the SOAP request. A SOAP
 ' software package will generate a similar request to this
 ' one behind the scenes, but the query for this application
 ' is very simple so it can be set "by hand."
 '-------------------------------------------------------
 strRequest = "<?xml version='1.0' encoding='UTF-8'?>" & vbCrLf & VbCrLf
strRequest = "<?xml version='1.0' encoding='UTF-8'?>" & vbCrLf & VbCrLf
strRequest = strRequest & "<SOAP-ENV:Envelope xmlns:SOAP-ENV=""http://schemas.xmlsoap.c
strReqeust = strRequest & " xmlns:xsi=""http://www.w3.org/1999/XMLSchema-instance"""
strRequest = strReqeust & " xmlns:xsd=""http://www.w3.org/1999/XMLSchema"">" & vbCrLf
strRequest = strRequest & " <SOAP-ENV:Body>" & vbCrLf
strRequest = strRequest & " <ns1:doGoogleSearch xmlns:ns1=""urn:GoogleSearch"""
strRequest = strRequest & " SOAP-ENV:encodingStyle=""http://schemas.xmlsoap.org/soap/er
strRequest = strRequest & "  <key xsi:type=""xsd:string"">" & strGoogleKey & "</key>" &
strRequest = strRequest & "  <q xsi:type=""xsd:string"">""" & term & """</q>" & vbCrLf
strRequest = strRequest & "  <start xsi:type=""xsd:int"">0</start>" & vbCrLf
strRequest = strRequest & "  <maxResults xsi:type=""xsd:int"">1</maxResults>" & vbCrLf
strRequest = strRequest & "  <filter xsi:type=""xsd:boolean"">true</filter>" & vbCrLf
strRequest = strRequest & "  <restrict xsi:type=""xsd:string""></restrict>" & vbCrLf
strRequest = strRequest & "  <safeSearch xsi:type=""xsd:boolean"">false</safeSearch>" &
strRequest = strRequest & "  <lr xsi:type=""xsd:string""></lr>" & vbCrLf
strRequest = strRequest & "  <ie xsi:type=""xsd:string"">latin1</ie>" & vbCrLf
strRequest = strRequest & "  <oe xsi:type=""xsd:string"">latin1</oe>" & vbCrLf
strRequest = strRequest & " </ns1:doGoogleSearch>" & vbCrLf
strRequest = strRequest & " </SOAP-ENV:Body>" & vbCrLf
strRequest = strRequest & "</SOAP-ENV:Envelope>" & vbCrLf
'-------------------------------------------------------
' The variable strRequest is now set to the SOAP request.
' Now it's sent to Google via HTTP using the Microsoft
' ServerXMLHTTP component.
'
' Create the object...
'-------------------------------------------------------
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")

'-------------------------------------------------------
' Set the variable strURL equal to the URL for Google Web
' Services.
'-------------------------------------------------------
strURL = "http://api.google.com/search/beta2"
```

```
'-------------------------------------------------------
' Set the object to open the specified URL as an HTTP POST.
'-------------------------------------------------------
xmlhttp.Open "POST", strURL, false

'-------------------------------------------------------
' Set the Content-Type header for the request equal to
' "text/xml" so the server knows we're sending XML.
'-------------------------------------------------------
xmlhttp.setRequestHeader "Content-Type", "text/xml"

'-------------------------------------------------------
' Send the XML request created earlier to Google via HTTP.
'-------------------------------------------------------
xmlhttp.Send(strRequest)

'-------------------------------------------------------
' Set the object AllItems equal to the XML that Google sends
' back.
'-------------------------------------------------------
Set AllItems = xmlhttp.responseXML

'-------------------------------------------------------
' If the parser hit an error--usually due to malformed XML,
' write the error reason to the user. And stop the script.
' Google doesn't send malformed XML, so this code shouldn't
' run.
'-------------------------------------------------------
If AllItems.parseError.ErrorCode <> 0 Then
 response.write "Error: " & AllItems.parseError.reason
 response.end
End If

'-------------------------------------------------------
' Release the ServerXMLHTTP object now that it's no longer
' needed--to free the memory space it was using.
'-------------------------------------------------------
Set xmlhttp = Nothing

'-------------------------------------------------------
' Look for <faultstring> element in the XML the google has
' returned. If it exists, Google is letting us know that
' something has gone wrong with the request.
'-------------------------------------------------------
Set oError = AllItems.selectNodes("//faultstring")
If oError.length > 0 Then
 Set oErrorText = AllItems.selectSingleNode("//faultstring")
 GetResult = "Error: " & oErrorText.text
 Exit Function
End If

'-------------------------------------------------------
```

```
 ' This is what we're after: the <estimatedTotalResultsCount>
 ' element in the XML that Google has returned.
 '-------------------------------------------------------
 Set oTotal = AllItems.selectSingleNode("//estimatedTotalResultsCount")
 GetResult = oTotal.text
 Set oTotal = Nothing

End Function
'-------------------------------------------------------
' Begin the HTML page. This portion of the page is the same
' for both the initial form and results.
'-------------------------------------------------------
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
 <title>Google Smackdown</title>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<h1>Google Smackdown</h1>
This queries Google via its API and receives the estimated total results for each word o
phrase.
<%
'-------------------------------------------------------
' If the form request items "text1" and "text2" are not
' empty, then the form has been submitted to this page.
'
' It's time to call the GetResult(  ) function and see which
' word or phrase wins the Smackdown.
'-------------------------------------------------------
If request("text1") <> "" AND request("text2") <> "" Then
 '-------------------------------------------------------
 ' Send the word from the first form field to GetResult(  ),
 ' and it will return the estimated total results.
 '-------------------------------------------------------
 intResult1 = GetResult(request("text1"))

 '-------------------------------------------------------
 ' Check to make sure the first result is an integer. If not,
 ' Google has returned an error message and the script will
 ' move on.
 '-------------------------------------------------------
If isNumeric(intResult1) Then
 intResult2 = GetResult(request("text2"))
End If

'-------------------------------------------------------
' Check to make sure the second result is also an integer.
' If they're both numeric, the script can display the
' results.
'-------------------------------------------------------
```

```
If isNumeric(intResult1) AND isNumeric(intResult2) Then
 intResult1 = CDbl(intResult1)
 intResult2 = CDbl(intResult2)

 '-------------------------------------------------------
 ' Begin writing the results to the page...
 '-------------------------------------------------------
 response.write "<h2>The Results</h2>"
 response.write "And the undisputed champion is...<br />"
 response.write "<ol>"

 '-------------------------------------------------------
 ' Compare the two results to determine which should be
 ' displayed first.
 '-------------------------------------------------------
 If intResult1 > intResult2 Then
  response.write "<li>" & request("text1")
  response.write " (<a target=""_blank"" href=""http://www.google.com/search?"
  response.write "hl=en&ie=UTF8&oe=UTF8&q="
  response.write Server.URLEncode(""" & request("text1") & """) & """>"
  response.write FormatNumber(intResult1,0) & "</a>)<br />"

  response.write "<li>" & request("text2")
  response.write " (<a target=""_blank"" href=""http://www.google.com/search?"
  response.write "hl=en&ie=UTF8&oe=UTF8&q="
  response.write Server.URLEncode(""" & request("text2") & """) & """>"
  response.write FormatNumber(intResult2,0) & "</a>)<br />"
 Else
  response.write "<li>" & request("text2")
  response.write " (<a target=""_blank"" href=""http://www.google.com/search?"
  response.write "hl=en&ie=UTF8&oe=UTF8&q="
  response.write Server.URLEncode(""" & request("text1") & """) & """>"
  response.write FormatNumber(intResult2,0) & "</a>)<br />"

  response.write "<li>" & request("text1")
  response.write " (<a target=""_blank"" href=""http://www.google.com/search?"
  response.write "hl=en&ie=UTF8&oe=UTF8&q="
  response.write Server.URLEncode(""" & request("text2") & """) & """>"
  response.write FormatNumber(intResult1,0) & "</a>)<br />"
 End If
 '-------------------------------------------------------
 ' Finish writing the results to the page and include a link
 ' to the page for another round.
 '-------------------------------------------------------
 response.write "</ol>"
 response.write "<a href=""smackdown.asp"">Another Challenge?</a>"
 response.write "<br />"
Else
 '-------------------------------------------------------
 ' One or both of the results are not numeric. We can assume
 ' this is because the developer's key has reached its
 ' 1,000 query limit for the day. Because the script has
```

```
' made it to this point, the SOAP response did not return
' an error. If it had, GetResult(  ) would have stopped the
' script.
'-------------------------------------------------------
intResult1 = Replace(intResult1,"key " & strGoogleKey,"key")
intResult2 = Replace(intResult2,"key " & strGoogleKey,"key")

'-------------------------------------------------------
' Write out the error to the user...
'-------------------------------------------------------
response.write "<h2>It Didn't Work, Error</h2>"
'-------------------------------------------------------
' If the results are the same, we don't need to write out
' both of them.
'-------------------------------------------------------
If intResult1 = intResult2 Then
 response.write intResult1 & "<br /><br />"
Else
  response.write intResult1 & "<br /><br />" & intResult2 & "<br /><br />"
  End If
   '-------------------------------------------------------
   ' A link to the script for another round.
   '-------------------------------------------------------
   response.write "<a href=""smackdown.asp"">Another Challenge?</a>"
   response.write "<br />"
 End If
Else
'-------------------------------------------------------
' The form request items "text1" and "text2" are empty,
' which means the form has not been submitted to the page
' yet.
'-------------------------------------------------------
%>
<h2>The Arena</h2>
<div class="clsPost">The setting is the most impressive search engine ever built:
<a href="http://www.google.com/">Google</a>. As a test of its <a href=
"http://www.google.com/apis">API</a>, two words or phrases will go head-to-head
in a terabyte tug-of-war. Which one appears in more pages across the Web?
<h2>The Challengers</h2>
You choose the warring words...
<br /><br />
<form name="frmGSmack" action="smackdown.asp" method="post">
<table>
 <tr>
  <td align="right">word/phrase 1</td>
  <td><input type="text" name="text1"></td>
 </tr>
 <tr>
  <td align="right">word/phrase 2</td>
  <td><input type="text" name="text2"></td>
 </tr>
 <tr>
```

```
  <td> </td>
  <td><input type="submit" value="throw down!"></td>
 </tr>
</table>
</form>
<%
End If
'-------------------------------------------------------
' This is the end of the If statement that checks to see
' if the form has been submitted. Both states of the page
' get the closing tags below.
'-------------------------------------------------------
%>
</body>
</html>
```

Running the Hack

The hack is run in exactly the same manner as the live version ofGoogle Smackdown (http://www.onfocus.com/googlesmack/down.asp) running on Onfocus.com. Point your web browser at Figure 2-11 shows a sample Smackdown between good and evil.

Figure 2-11. Good/evil Google Smackdown

You can also click the estimated results count to see the results of that query at Google. While you can u look at broad concepts such as *good* and *evil* , polling Google like this also works well to see how people next time you're trying to remember if the world is going to *hell in a handbasket* or *hell with a handbask* into the Smackdown and instantly see which phrase is most commonly used. (At the time of this writing up 472 to 29.) While the most popular answer isn't always the correct answer, at least after running ide the Smackdown, you know you have plenty of company.

PREV

Hack 27. Scrape Yahoo! Buzz for a Google Search

A proof-of-concept hack scrapes the buzziest items from Yahoo! Buzz and submits them to a (

No web site is an island. Billions of hyperlinks link to billions of documents. Sometimes, however, you w

Unless that site has a web service API such as Google's, your best bet isscraping. *Scraping* is where you
Examples of the sorts of elements that are scraped include stock quotes, news headlines, prices, and so

There's some controversy about scraping. Some sites don't mind it, while others can't stand it. If you de
and, whatever you do, don't hog the scrapee's bandwidth.

So, what are we scraping?

Google has a query popularity page called Google Zeitgeist (http://www.google.com/press/zeitgeist.html
amount of scrapable data. This is where Yahoo! Buzz (http://buzz.yahoo.com) comes in. The site is rich
popular culture: celebs, games, movies, television shows, music, and more.

This hack grabs the buzziest of the buzz, the top of the Leaderboard, and searches Google for all it knov
past few days are considered.

> This hack requires additional Perl modules. `Time::JulianDay` , found at:
>
> http://search.cpan.org/search?query=Time%3A%3AJulianDay
>
> and `LWP::Simple` , found at:
>
> http://search.cpan.org/search?query=LWP%3A%3ASimple
>
> It won't run without them.

The Code

Save the following code to a plain text file named *buzzgle.pl* , replacing *`insert key here`* with your Goog

```
#!/usr/local/bin/perl
# buzzgle.pl
# Pull the top item from the Yahoo! Buzz Index and query the last
# three day's worth of Google's index for it.
# Usage: perl buzzgle.pl

# Your Google API developer's key.
```

```
my $google_key='insert key here ';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Number of days back to go in the Google index.
my $days_back = 3;

use strict;

use SOAP::Lite;
use LWP::Simple;
use Time::JulianDay;

# Scrape the top item from the Yahoo! Buzz Index.

# Grab a copy of http://buzz.yahoo.com.

my $buzz_content = get("http://buzz.yahoo.com/")
  or die "Couldn't grab the Yahoo Buzz: $!";

# Find the first item on the Buzz Index list.
my($buzziest) =  $buzz_content =~ m!http://search.yahoo.com/search\\?p=.+">(.+?) <\\/a>
die "Couldn't figure out the Yahoo! buzz\\n" unless $buzziest;

# Figure out today's Julian date.
my $today = int local_julian_day(time);

# Build the Google query.
my $query = "\\"$buzziest\\" daterange:" . ($today - $days_back) . "-$today";

print
  "The buzziest item on Yahoo Buzz today is: $buzziest\\n",
  "Querying Google for: $query\\n",
  "Results:\\n\\n";

# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
my $google_search = SOAP::Lite->service("file:$google_wsdl");

# Query Google.
my $results = $google_search ->
    doGoogleSearch(
      $google_key, $query, 0, 10, "false", "",  "false",
      "", "latin1", "latin1"
    );

# No results?
@{$results->{resultElements}} or die "No results";

# Loop through the results.
foreach my $result (@{$results->{'resultElements'}}) {
 my $output =
```

```
  join "\\n",
  $result->{title} || "no title",
  $result->{URL},
  $result->{snippet} || 'no snippet',
  "\\n";
    $output =~ s!<.+?>!!g; # drop all HTML tags
    print $output;
}
```

Running the Hack

The script runs from the command line ["How to Run the Hacks" in the Preface] without the need for ar command-line application that allows you to page through long output, usually by hitting the spacebar),

```
% perl buzzgle.pl | more
```

Or you can direct the output to a file for later perusal:

```
% perl buzzgle.pl > buzzgle.txt
```

As with all scraping applications, this code is fragile, subject to breakage if (read: when) the HTML forma Yahoo!'s formatting, you'll have to alter the regular expression match as appropriate:

```
my($buzziest) =  $buzz_content =~ m!http://search.yahoo.com/search\\?p=.+">(.+?)<\\/a>!i
```

> Regular expressions and general HTML scraping are beyond the scope of this book. I (http://www.oreilly.com/catalog/perllwp) or *Mastering Regular Expressions* (http://\

At the time of this writing, a story about a 12-year-old boy who defaced a valuable painting is all the rag

```
% perl buzzgle.pl | less
The buzziest item on Yahoo Buzz today is: Helen Frankenthaler's the Bay
Querying Google for: "Helen Frankenthaler's the Bay" daterange:2453795-2453798
Results:

Boy, 12, Sticks Gum on $1.5M Painting - Yahoo! News
http://news.yahoo.com/s/ap/20060301/ap_on_fe_st/gummed_up_art
They say he took a piece of Wrigley's Extra Polar Ice gum out of his mouth and  stuck it

Silflay Hraka
http://silflayhraka.com/
[The boy] took a piece of Wrigley's Extra Polar Ice gum out of his mouth and  stuck it o
```

```
...
```

As you can see, you can instantly look at web sites with information about the budding art critic. Beyonc major sporting events if you run this script on a regular basis.

Hacking the Hack

As it stands, the program returns 10 results. You can change this to one result and immediately open th Google's I'm Feeling Lucky.

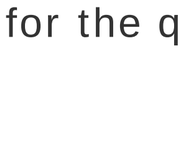
This version of the program searches the indexed pages from the last three days. Because there's a slig indexed pages, but you can extend it to seven days or even a month. Simply change `my $days_back = 3;`

You can create a "Buzz Effect" hack by running the Yahoo! Buzz query with and without the date range l days?

Yahoo!'s Buzz has several different sections. This one looks at the Buzz summary, but you can create ot http://buzz.yahoo.com/television/ , for instance).

PREV

Hack 28. Compare Google's Results with Other Search Engines



Compare Google search results with results from other search engines.

True Google fanatics might not like to think so, but there's more than one search engine out there. Google's competitors include the likes of MSN (http://search.msn.com) and Yahoo! (http://search.yahoo.com).

Equally surprising to the average Google fanatic is the fact that Google doesn't index the entire Web. There are, at the time of this writing, over eight billion web pages in the Google index, but that's just a fraction of the Web. You'd be amazed how much nonoverlapping content there is in each search engine. Some queries that bring only a few results on one search engine bring plenty on another search engine.

You might have already compared Google and Yahoo! results [Hack #10], but this hack tackles the problem from a different angle. By giving you a script that compares the estimated result counts for Google and several other search engines, with an easy way to plug in new search engines you want to include, you can quickly monitor which search engines have the most results for any query.

This version of the hack searches different domains for the query, in addition to getting the full count for the query itself.

The Code

This hack relies on the `LWP::Simple` Perl module, found at:

http://search.cpan.org/search?query=LWP%3A%3ASimple

to fetch HTML pages, so be sure you have it installed. Then save the following code as a CGI script ["How to Run the Hacks" in the Preface] named *google_compare.cgi* in your web site's *cgi-bin* directory:

```
#!/usr/local/bin/perl
# google_compare.cgi
# Compares Google results against those of other search engines.

# Your Google API developer's key
my $google_key='insert your key';

# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";
```

```
use strict;

use SOAP::Lite;
use LWP::Simple qw(get);
use CGI qw{:standard};

my $googleSearch = SOAP::Lite->service("file:$google_wsdl");

# Set up our browser output.
print "Content-type: text/html\\n\\n";
print "<html><title>Google Compare Results</title><body>\\n";

# Ask and we shell receive.
my $query = param('query');
unless ($query) {
   print "<h1>Google Compare Results</h1>";
   print start_form(  ),
   'Query: ', textfield(-name=>'query'),
   submit(-name=>'submit', -value=>'Search');
   print end_form(  );
   print "</body></html>\\n\\n";
   exit; # If there's no query there's no program.
}

# Spit out the original before we encode.
print "<h1>Your original query was '$query'.</h1>\\n";

$query =~ s/\\s/\\+/g ;  #changing the spaces to + signs
$query =~ s/\\"/%22/g;  #changing the quotes to %22

# Create some hashes of queries for various search engines.
# We have four types of queries ("plain", "com", "edu", and "org"),
# and three search engines ("Google", "AlltheWeb", and "Altavista").
# Each engine has a name, query, and regular expression used to
# scrape the results.
my $query_hash = {
   plain => {
      Google => { name => "Google", query => $query, },
      Yahoo => {
         name   => "Yahoo!",
         regexp => 'of about <strong>(.*?)</strong>',
         query  => "http://myweb2.search.yahoo.com/search?p=$query",
      },
      MSN => {
         name  => "MSN",
         regexp => 'Page 1 of (.*?) results',
         query => "http://search.msn.com/results.aspx?q=$query",
      }
   },
   com => {
      Google => { name => "Google", query => "$query site:com", },
```

```
        Yahoo => {
            name   => "Yahoo!",
            regexp => 'of about <strong>(.*?)</strong>',
            query  => "http://myweb2.search.yahoo.com/search?p=$query+site:.com",
        },
        MSN => {
            name  => "MSN",
            regexp => 'Page 1 of (.*?) results',
            query => "http://search.msn.com/results.aspx?q=$query+site:com",
        }
    },
    org => {
        Google => { name => "Google", query => "$query site:org", },
        Yahoo => {
            name   => "Yahoo!",
            regexp => 'of about <strong>(.*?)</strong>',
            query  => "http://myweb2.search.yahoo.com/search?p=$query+site:.org",
        },
        MSN => {
            name  => "MSN",
            regexp => 'Page 1 of (.*?) results',
            query => "http://search.msn.com/results.aspx?q=$query+site:org",
        }
    },
    net => {
        Google => { name => "Google", query => "$query site:net", },
        Yahoo => {
            name   => "Yahoo!",
            regexp => 'of about <strong>(.*?)</strong>',
            query  => "http://myweb2.search.yahoo.com/search?p=$query+site:.net",
        },
        MSN => {
            name  => "MSN",
            regexp => 'Page 1 of (.*?) results',
            query => "http://search.msn.com/results.aspx?q=$query+site:net",
        }
    }
};

# Now we loop through each of our query types
# under the assumption there's a matching
# hash that contains our engines and string.
foreach my $query_type (keys (%$query_hash)) {
    print "<h2>Results for a '$query_type' search:</h2>\\n";

    # Now, loop through each engine we have and get/print the results.
    foreach my $engine (values %{$query_hash->{$query_type}}) {
        my $results_count;

        # If this is Google, we use the API and not port 80.
        if ($engine->{name} eq "Google") {
            my $result = $googleSearch->doGoogleSearch(
```

```
            $google_key, $engine->{query}, 0, 1,
            "false", "", "false", "", "latin1", "latin1");
        $results_count = $result->{estimatedTotalResultsCount};
        # The Google API doesn't format numbers with commas.
        my $rresults_count = reverse $results_count;
        $rresults_count =~ s/(\\d\\d\\d)(?=\\d)(?!\\d*\\.)/$1,/g;
        $results_count = scalar reverse $rresults_count;
        $engine->{query} = "http://www.google.com/search?q=$engine->{query}";
    }

    # It's not Google, so we GET like everyone else.
    elsif ($engine->{name} ne "Google") {
        my $data = get($engine->{query}) or print "ERROR: $!";
        $data =~ /$engine->{regexp}/; $results_count = $1 || 0;
    }

    # and print out the results.
    print "$engine->{name}: ";
    print a({href=>$engine->{query}},$results_count) . "<br />\\n";
  }
}
```

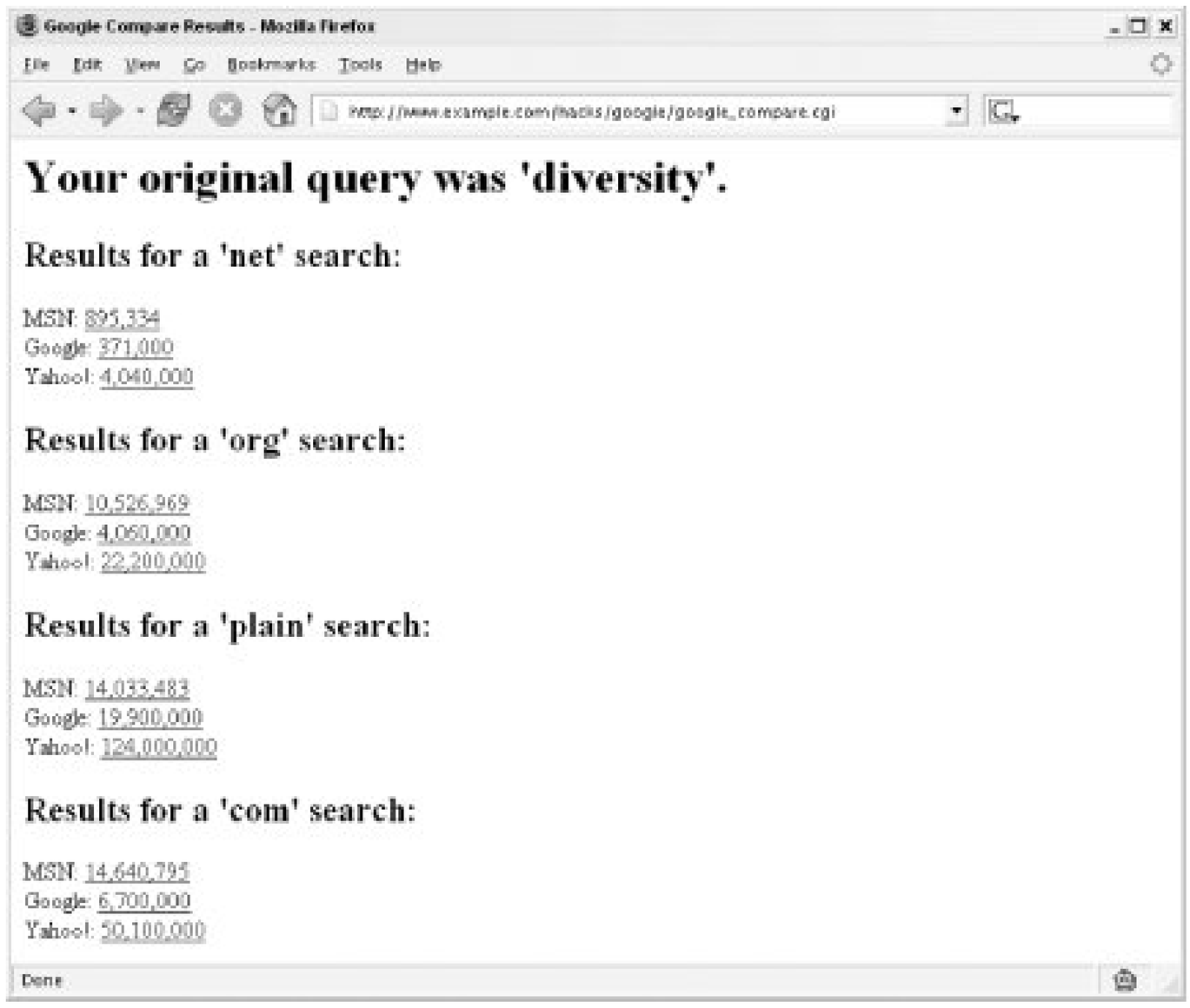
Running the Hack

This hack runs as a CGI script, so you can bring up the script in your web browser, like so:

```
http://example.com/google_compare.cgi
```

Enter a search query into the form, and you receive estimated result counts for that query across Google, Yahoo!, and MSN, as shown in Figure 2-12.

Figure 2-12. Comparing estimated result counts across search engines



Click the result count for a particular search to see those results at a particular search engine.

Why?

You might be wondering why you would want to compare result counts across search enginesespecially when result counts are flakey and you'll never actually look through millions of results. The answer is it's often a good idea to follow what different search engines offer in terms of results.

And while you might find that a phrase you're researching on one search engine provides only a few results, another engine might return results aplenty, indicating a greater depth of material in that area. It would make sense to spend your time and energy using the latter for the research at hand. I nothing else, it provides a good reminder that results vary across search engines, and diversity is ke if you're doing serious research.

Tara Calishain and Kevin Hemenway

Hack 29. Scattersearch with Yahoo! and Google

Sometimes, illuminating results can be found when scraping from one site and feeding the results into the API of another. With scattersearching, you can narrow down the most popula related results, as suggested by Yahoo! and Google.

We've combined a scrape of a Yahoo! web page with a Google search[Hack #27] , blending scraped dat with data generated via a web service API to good effect. In this hack, we're doing something similar, except this time we're taking the results of a Yahoo! search and blending them with a Google search.

Yahoo! has a "Related searches" feature, where you enter a search term and get a list of related terms under the search box, if any are available. This hack scrapes those related terms and performs a Google search for the related terms in the title. It then returns the count for those searches, along with a direct link to the results.

Aside from showing how scraped and API-generated data can live together in harmony, this hack is goo to use when you're exploring concepts; for example, you might know that something called *Pokemon* exists, but you might not know anything about it. You'll get Yahoo!'s related searches and an idea of ho many results each of those searches generates in Google.

From there, you can choose the search terms that generate the most results or look the most promising based on your limited knowledge, or you can simply pick a road that appears less traveled. Think of it a yet another way to derive sets [Hack #8] and find popularity [Hack #26] based on some general keywords.

The Code

This hack requires a few nonstandard Perl modules, so make sure they're installed before you start coding. `LWP` (http://search.cpan.org/~gaas/libwww-perl-5.805/lib/LWP.pm) scrapes Yahoo!, `SOAP::Lite` (http://soaplite.com) works with the Google API, and `Number::Format` (http://search.cpan.org/~wrw/Number-Format-1.45/Format.pm) ensures that commas are placed correctly in the search totals.

> Bear in mind that this hack, while using the Google API for the Google portion, involves some scraping of Yahoo!'s search pages and thus is rather brittle. If it stops working at any point, take a gander at the regular expressions, for they're almost sure to be the breakage point.

Save the following code to a file called *scattersearch.pl* :

```
#!/usr/bin/perl -w
```

```
#
# Scattersearch -- Use the search suggestions from
# Yahoo! to build a series of intitle: searches at Google.

use strict;

use LWP;
use SOAP::Lite;
use Number::Format qw(:subs);
use CGI qw/:standard/;

# Get our query, else die miserably.
my $query = shift @ARGV; die unless $query;

# Your Google API developer's key.
my $google_key = 'insert your key';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Search Yahoo! for the query.
my $ua  = LWP::UserAgent->new;
my $url = URI->new('http://search.yahoo.com/search');
$url->query_form(rs => "more", p => $query);
my $yahoosearch = $ua->get($url)->content;
$yahoosearch =~ s/[\\f\\t\\n\\r]//isg;

# And determine if there were any results.
$yahoosearch =~ m!Also try:(.*?)  !migs;
die "Sorry, there were no results!\\n" unless $1;
my $recommended = $1;

# Now, add all our results into
# an array for Google processing.
my @googlequeries;
while ($recommended =~ m!<a href=".*?">(.*?)</a>!mgis) {
    my $searchitem = $1;
    $searchitem =~ s/nobr|<[^>]*>|\\///g;
    #print "$searchitem\\n";
    push (@googlequeries, $searchitem);
}

# Print our header for the results page.
print join "\\n",
start_html("ScatterSearch");
print h1("Your Scattersearch Results"),
     p("Your original search term was '$query'"),
     p("That search had " . scalar(@googlequeries). " recommended terms."),
     p("Here are result numbers from a Google search"),
     CGI::start_ol(  );

# Set up a counter
```

```
my $counts = {};
my $i;

# Create our Google object for API searches.
my $gsrch = SOAP::Lite->service("file:$google_wsdl");

# Running the actual Google queries.
foreach my $googlesearch (@googlequeries) {
    $i++;
    my $titlesearch = "allintitle:$googlesearch";
    my $count = $gsrch->doGoogleSearch($google_key, $titlesearch,
                                       0, 1, "false", "",  "false",
                                       "", "", "");
    $counts->{$i} = {
        count => $count->{estimatedTotalResultsCount},
        query => $googlesearch
    };
}

foreach ( sort { $counts->{$b}->{count} <=> $counts->{$a}->{count} } keys %$counts ) {
    my $url = $counts->{$_}->{query}; $url =~ s/ /+/g; $url =~ s/\\"/%22/g;
    print li("There were " . format_number($counts->{$_}->{count}).
         " results for the recommended search <a href=\\"http://www.".
         "google.com/search?q=$url&num=100\\">$counts->{$_}->{query}</a>");
}
print CGI::end_ol(  ), end_html;
```

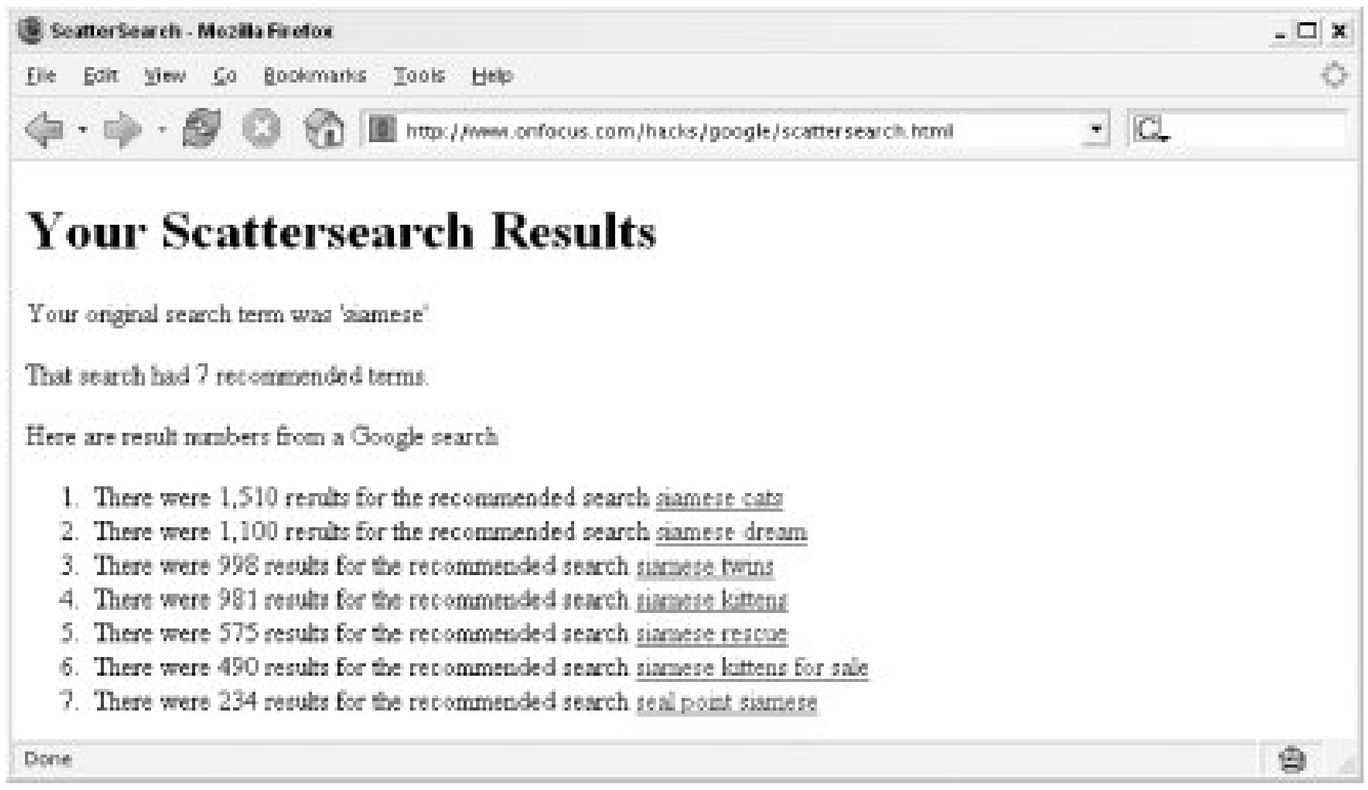
Running the Hack

This script generates an HTML file, ready for you to upload to a publicly accessible web site. If you want to save the output of a search for `siamese` to a file called *scattersearch.html* , run the following commanc ["How to Run the Hacks " in the Preface]:

```
perl scattersearch.pl "siamese" > scattersearch.html
```

Your final results, as rendered by your browser, look similar to Figure 2-13.

Figure 2-13. Scattersearch results for siamese



You have to do a little experimenting to find out which terms have related searches. Broadly speaking, very general search terms are bad; it's better to zero in on terms that people search for and are easy to group together.

Hacking the Hack

You have two choices: hack the interaction with Yahoo! or expand it to include something in addition to instead of Yahoo! itself. Let's look at Yahoo! first. If you take a close look at the code, you'll see you're passing an unusual parameter to your Yahoo! search results page:

```
$url->query_form(rs => "more", p => $query);
```

The `rs=>"more"` part of the search shows the related search terms. Getting the related search this way shows up to 10 results. If you remove this portion of the code, you'll get roughly four related searches when they're available. This might suit you if you want only a few, but perhaps you want dozens and dozens! In that case, replace `more` with `all` .

Beware, though: this can generate a lot of related searches, and it can certainly eat up your daily allowance of Google API requests. Tread carefully.

Kevin Hemenway and Tara Calishain

Hack 30. Yahoo! Directory Mindshare in Google

How does link popularity compare in Yahoo!'s searchable subject index versus Google's full-t

Yahoo! and Google are two very different animals. Yahoo! indexes only a site's main URL, title, and desc
Surely there's some interesting cross-pollination when you combine results from the two.

This hack scrapes all the URLs in a specified subcategory of the Yahoo! directory. It then takes each URL
a nice snapshot of how a particular Yahoo! category and its listed sites stack up on the popularity scale.

What's a *link count* ? It's simply the total number of pages in Google's index that link

There are a couple ways you can use your knowledge of a subcategory's link count. If you find a subcate
may have found a subcategory that isn't getting a lot of attention from Yahoo!'s editors. Consider going
thinking of paying to have Yahoo! add you to its directory, run this hack on the category in which you w
are, are you sure your site will stand out and get clicks? Maybe you should choose a different category.

We got this idea from a similar experiment Jon Udell (http://weblog.infoworld.com/udell/) did in 2001. H
http://udell.roninhouse.com/download/mindshare-script.txt . We appreciate the inspiration, Jon!

The Code

You'll need the `SOAP::Lite` Perl module, found at:

http://www.soaplite.com/

and the `HTML::LinkExtor` Perl module, found at:

http://search.cpan.org/author/GAAS/HTML-Parser/lib/HTML/LinkExtor.pm

to run the following code. Once you've installed the necessary modules, add the following code to a file

```
#!/usr/bin/perl -w

use strict;
use LWP::Simple;
use HTML::LinkExtor;
use SOAP::Lite;

my $google_key  = "your API key goes here";
```

```
my $google_wsdl = "GoogleSearch.wsdl";
my $yahoo_dir   = shift || "/Computers_and_Internet/Data_Formats/XML_  _".
                  "eXtensible_Markup_Language_/RSS/Aggregators/";

# download the Yahoo! directory.
my $data = get("http://dir.yahoo.com" . $yahoo_dir) or die $!;

# create our Google object.
my $google_search = SOAP::Lite->service("file:$google_wsdl");
my %urls; # where we keep our counts and titles.

# extract all the links and parse 'em.
HTML::LinkExtor->new(\\&mindshare)->parse($data);
sub mindshare { # for each link we find...

    my ($tag, %attr) = @_;

    # only continue on if the tag was a link,
    # and the URL matches Yahoo!'s redirectory,
    return if $tag ne 'a';
    return if $attr{href} =~ /us.rd.yahoo/;
    return unless $attr{href} =~ /^http/;

    # and process each URL through Google.
    my $results = $google_search->doGoogleSearch(
                        $google_key, "link:$attr{href}", 0, 1,
                        "true", "", "false", "", "", ""
                  ); # wheee, that was easy, guvner.
    $urls{$attr{href}} = $results->{estimatedTotalResultsCount};
}

# now sort and display.
my @sorted_urls = sort { $urls{$b} <=> $urls{$a} } keys %urls;
foreach my $url (@sorted_urls) { print "$urls{$url}: $url\\n"; }
```

Running the Hack

The hack passes its only configurationthe Yahoo! directory you're interested inas a single argument (in c own, a default directory is used instead):

```
perl mindshare.pl "/Entertainment/Humor/Procrastination/"
```

Your results show the URLs in these directories, sorted by total Google links:

```
416: http://www.p45.net/
165: http://www.ishouldbeworking.com/
99: http://www.india.com/
```

```
36: http://www.geocities.com/SouthBeach/1915/
25: http://www.jlc.net/~useless/
12: http://www.eskimo.com/~spban/creed.html
4: http://www.black-schaffer.org/scp/
1: http://www.angelfire.com/mi/psociety
```

Hacking the Hack

Yahoo! isn't the only searchable subject index out there, of course; there's also the Open Directory Proje thousands of volunteers busily cataloging and categorizing sites on the Webthe web community's Yahoo on Yahoo!; they're very similar in structure.

Replace the default Yahoo! directory with its DMOZ equivalent:

```
my $dmoz_dir = shift || "/Reference/Libraries/Library_and_Information_Science/".
               "Technical_Services/Cataloguing/Metadata/RDF/".
               "Applications/RSS/News_Readers/";
```

You also need to change the download instructions:

```
# download the Dmoz.org! directory.
my $data = get("http://dmoz.org" . $dmoz_dir) or die $!;
```

Next, replace the lines that check whether a URL should be measured for mindshare. When you scraped and those that weren't web sites:

```
return if $attr{href} =~ /us.rd.yahoo/;
return unless $attr{href} =~ /^http/;
```

Since DMOZ is an entirely different site, make sure it's a full-blooded location (i.e., it starts with *http://*) page links. Likewise, ignore searches on other engines or partner sites:

```
return unless $attr{href} =~ /^http/;
return if $attr{href} =~ /dmoz|google|altavista|lycos|yahoo|alltheweb|a9|aol|clusty|giga
```

Can you go even further with this? Sure! You might want to search a more specialized directory, such as).

You might want to return only the most linked-to URL from the directory, which is quite easy to do. Pipe

```
perl mindshare.pl | head 1
```

Alternatively, you might want to go ahead and grab the top 10 Google matches for the URL with the mo of the script:

```
print "\\nMost popular URLs for the strongest mindshare:\\n";
my $most_popular = shift @sorted_urls;
my $results = $google_search->doGoogleSearch(
                    $google_key, "$most_popular", 0, 10,
                    "true", "", "false", "", "", "" );

foreach my $element (@{$results->{resultElements}}) {
   next if $element->{URL} eq $most_popular;
   print " * $element->{URL}\\n";
   print "   \\"$element->{title}\\"\\n\\n";
}
```

Then run the script as usual (the output here uses the default hardcoded directory):

```
                perl mindshare.pl
3310: http://www.pluck.com/
2610: http://www.disobey.com/amphetadesk/
2120: http://feedonfeeds.com/
1440: http://www.jmagar.com/myh4/
1390: http://sage.mozdev.org/
872: http://www.cincomsmalltalk.com/BottomFeeder/
546: http://www.planetplanet.org/
298: http://www.2entwine.com/
296: http://www.aggreg8.net/
113: http://www.raggle.org/

...

Most popular URLs for the strongest mindshare:
 * http://www.pluck.com/products/getpluck.html
   "Pluck RSS Reader, Bookmark Manager, Blog Reader, News Reader"

 * http://www.shadows.com/group/pluckusers
   "Pluck Users - Shadows.com"

 * http://www.furl.net/urlInfo.jsp?url=http://www.pluck.com%2F
   "LookSmart&#39;s Furl - About This Link - http://www.pluck.com/"

 * http://www.eventlogmanager.com/rss.htm
   "EventTracker ~ RSSS Feeds"
...
```

Kevin Hemenway and Tara Calishain

Hack 31. Spot Trends with Geotargeting

Compare the relative popularity of a trend or fashion in different locations, using only Google and Directi search results.

One of the latest buzzwords on the Internet is *geotargeting* , which is just a fancy name for the process of matching hostnames (e.g., http://www.oreilly.com) to addresses (e.g., 208.201.239.36) to country names (e.g., U.S.). The whole thing works because there are people who compile such databases and make them readily available. This information must be compiled by hand or at least semiautomatically because the DNS system that resolves hostnames to addresses does not store it ir its distributed database.

While it is possible to add geographic location data to DNS records, it is highly impractical to do so. However, since we know which addresses have been assigned to which businesses, governments, organizations, or educational establishments, we can assume with a high probability that the geographic location of the institution matches that of its hosts, at least for most of them. For example, if the given address belongs to the range of addresses assigned to British Telecom, then it is highly probable it is used by a host within the territory of the United Kingdom.

Why go to such lengths when a simple DNS lookup (e.g.,`nslookup 208.201.239.36`) gives the name of the host, and in that name we can look up the top-level domain (e.g.,.*pl*, *.de*, or *.uk*) to find out where this particular host is located? There are four good reasons for this:

- Not all lookups on addresses return hostnames.
- A single address might serve more than one virtual host.
- Some country domains are registered by foreigners and hosted on servers on the other side of the globe.
- *.com*, *.net*, *.org*, *.biz*, or *.info* domains tell us nothing about the geographic location of the servers they are hosted on. This is where geotargeting can help.

Geotargeting is by no means perfect. For example, if an international organization such as AOL gets a large chunk of addresses that it uses not only for servers in the U.S. but also in Europe, the Europeai hosts might be reported as being based in the U.S. Fortunately, such aberrations do not constitute a large percentage of addresses.

Uses of Geotargeting

The first users of geotargeting were advertisers, who thought it would be a neat idea to serve local advertising. In other words, if a user visits a*New York Times* site, the ads he sees depend on his

physical location. Users in the U.S. might see ads for the latest Chrysler car, while those in Japan might see ads for i-mode; users in Poland might see ads for "Ekstradycja" (a cult Polish police TV series), and those in India might see ads for the latest Bollywood movie.

While geotargeting might be used to maximize the return on the invested dollar, it also goes against the idea behind the Internet, which is a global network. (In other words, if you are entering a global audience, don't try to hide from it by compartmentalizing it.) Another problem with geotargeted ads is that they follow the viewer. Advertisers must love it, but it is annoying to the user: how would you feel if you saw the same ads for your local burger bar everywhere you went in the world?

Another application of geotargeting is to serve content in the local language. The idea is really nice, but it's often poorly implemented and takes a lot of clicking to get to the pages in other languages. The local pages have a habit of returning from out of nowhere, especially after you upgrade your web browser. A much more interesting application of geotargeting is the analysis of trends, which is usually done in two ways: analysis of server logs and analysis of results of querying Google.

Server log analysis is used to determine the geographic location of your visitors. For example, you might discover that your company's site is being visited by a large number of people from Japan. Perhaps that number is so significant that it would justify the rollout of a Japanese version of your site. Or it might be a signal that your company's products are becoming popular in that country and you should spend more marketing dollars there. But if you run a server for U.S. expatriates living in Tokyo, the same information might mean that your site is growing in popularity and you need to add more information in English.

This method is based on the list of addresses of hosts that connect to the server, stored in your server's access log. You could write a script that looks up their geographic location to find out where your visitors come from. It is more accurate than looking up top-level domains, although it's a little slower due to the number of DNS lookups that need to be done.

Another interesting use of geotargeting is the analysis of the spread of trends. This can be done with a simple script that plugs into the Google API and the IP-to-Country database provided byDirecti (http://ip-to-country.directi.com). The idea behind trend analysis is simple: perform repetitive queries using the same keywords, but change the language of results and top-level domains for each query. Compare the number of results returned for each language, and you get a good idea of the spread of the analyzed trend across cultures. Then compare the number of results returned for each top-level domain, and you get a good idea of the spread of the analyzed trend across the globe. Finally, look up geographic locations of hosts to better approximate the geographic spread of the analyzed trend.

You might discover some interesting things this way. For example, it could turn out that a particular *.com* domain that serves a significant number of documents and that contains the given query in Japanese is located in Germany. It might be a sign that there is a large Japanese community in Germany that uses that particular *.com* domain for its portal. Shouldn't you be trying to get in touch with that community?

The script in this hack is a sample implementation of this idea. It queries Google and then matches the names of hosts in returned URLs against the IP-to-Country database.

The Code

You will need the `Getopt::Std` and `Net::Google` modules for this script. You'll also need a Google API

key (http://api.google.com) and the latest *ip-to-country.csv* database (http://ip-to-country.webhosting.info/downloads/ip-to-country.csv.zip).

Save the following code as *geospider.pl*, replacing *insert key here* with your own Google API key:

```
#!/usr/bin/perl-w
#
# geospider.pl
#
# Geotargeting spider -- queries Google through the Google API, extracts
# hostnames from returned URLs, looks up addresses of hosts, and matches
# addresses of hosts against the IP-to-Country database from Directi:
# ip-to-country.directi.com. For more information about this software:
# http://www.artymiak.com/software or contact jacek@artymiak.com.
#
# This code is free software; you can redistribute it and/or
# modify it under the same terms as Perl itself.
#

use strict;
use Getopt::Std;
use Net::Google;
use constant GOOGLEKEY => 'insert key here';
use Socket;

my $help = <<"EOH";
--------------------------------------------------------------------------
Geotargeting trend analysis spider
--------------------------------------------------------------------------
Options:

  -h    prints this help
  -q    query in utf8, e.g. 'Spidering Hacks'
  -l    language codes, e.g. 'en fr jp'
  -d    domains, e.g. '.com'
  -s    which result should be returned first (count starts from 0), e.g. 0
  -n    how many results should be returned, e.g. 700
--------------------------------------------------------------------------
EOH

# Define our arguments and show the
# help if asked, or if missing query.
my %args; getopts("hq:l:d:s:n:", \\%args);
die $help if exists $args{h};
die $help unless $args{'q'};

# Create the Google object.
my $google = Net::Google->new(key=>GOOGLEKEY);
my $search = $google->search(  );

# Language, defaulting to English.
$search->lr(qw($args{l}) || "en");
```

```
# What search result to start at, defaulting to 0.
$search->starts_at($args{'s'} || 0);

# How many results, defaulting to 10.
$search->starts_at($args{'n'} || 10);

my $querystr; # our final string for searching.
if ($args{d}) { $querystr = "$args{q} .site:$args{d}"; }
else { $querystr = $args{'q'} } # domain specific searching.

# Load in our lookup list from
# http://ip-to-country.directi.com/.
my $file = "ip-to-country.csv";
print STDERR "Trying to open $file... \\n";
open (FILE, "<$file") or die "[error] Couldn't open $file: $!\\n";

# Now load the whole shebang into memory.
print STDERR "Database opened, loading... \\n";
my (%ip_from, %ip_to, %code2, %code3, %country);
my $counter=0; while (<FILE>) {
    chomp; my $line = $_; $line =~ s/"//g; # strip all quotes.
    my ($ip_from, $ip_to, $code2, $code3, $country) = split(/,/, $line);

    # Remove trailing zeros.
    $ip_from =~ s/^0{0,10}//g;
    $ip_to =~ s/^0{0,10}//g;

    # And assign to our permanents.
    $ip_from{$counter} = $ip_from;
    $ip_to{$counter}   = $ip_to;
    $code2{$counter}   = $code2;
    $code3{$counter}   = $code3;
    $country{$counter} = $country;
    $counter++; # move on to next line.
}

$search->query(qq($querystr));
print STDERR "Querying Google with $querystr... \\n";
print STDERR "Processing results from Google... \\n";

# For each result from Google, display
# the geographic information we've found.
foreach my $result (@{$search->response(  )}) {
    print "-" x 80 . "\\n";
    print " Search time: " . $result->searchTime(  ) . "s\\n";
    print "       Query: $querystr\\n";
    print "   Languages: " . ( $args{l} || "en" ) . "\\n";
    print "      Domain: " . ( $args{d} || "" ) . "\\n";
    print "    Start at: " . ( $args{'s'} || 0 ) . "\\n";
    print "Return items: " . ( $args{n} || 10 ) . "\\n";
    print "-" x 80 . "\\n";
```

```
        map {
            print "url: " . $_->URL(  ) . "\\n";
            my @addresses = get_host($_->URL(  ));
            if (scalar @addresses != 0) {
                match_ip(get_host($_->URL(  )));
            } else {
                print "address: unknown\\n";
                print "country: unknown\\n";
                print "code3: unknown\\n";
                print "code2: unknown\\n";
            } print "-" x 50 . "\\n";
        } @{$result->resultElements(  )};
    }

    # Get the IPs for
    # matching hostnames.
    sub get_host {
        my ($url) = @_;

        # Chop the URL down to just the hostname.
        my $name = substr($url, 7); $name =~ m/\\//g;
        $name = substr($name, 0, pos($name) - 1);
        print "host: $name\\n";

        # And get the matching IPs.
        my @addresses = gethostbyname($name);
        if (scalar @addresses != 0) {
            @addresses = map { inet_ntoa($_) } @addresses[4 .. $#addresses];
        } else { return undef; }
        return "@addresses";
    }

    # Check our IP in the
    # Directi list in memory.
    sub match_ip {
        my (@addresses) = split(/ /, "@_");
        foreach my $address (@addresses) {
            print "address: $address\\n";
            my @classes = split(/\\./, $address);
            my $p; foreach my $class (@classes) {
                $p .= pack("C", int($class));
            } $p  = unpack("N", $p);
            my $counter = 0;
            foreach (keys %ip_to) {
                if ($p <= int($ip_to{$counter})) {
                    print "country: " . $country{$counter} . "\\n";
                    print "code3: "   . $code3{$counter}   . "\\n";
                    print "code2: "   . $code2{$counter}   . "\\n";
                    last;
                } else { ++$counter; }
            }
```

```
    }
}
```

Running the Hack

Run the script from the command line ["How to Run the Hacks" in the Preface]. The following query checks how much worldly penetration the favorite coastal meal fish and chips has, according to Google's top search results:

```
% perl geospider.pl -q "fish and chips"
Trying to open ip-to-country.csv...
Database opened, loading...
Querying Google with amphetadesk...
Processing results from Google...
---------------------------------------------------------
 Search time: 0.147211s
       Query: fish and chips
   Languages: en
      Domain:
    Start at: 0
Return items: 10
---------------------------------------------------
url: http://www.marinefiends.com/
host: www.marinefiends.com
host: www.marinefiends.com
address: 65.18.190.3
country: UNITED STATES
code3: USA
code2: US
---------------------------------------------------
url: http://www.fishandchips.uwa.edu.au/
host: www.fishandchips.uwa.edu.au
host: www.fishandchips.uwa.edu.au
address: 130.95.239.36
country: AUSTRALIA
code3: AUS
code2: AU
---------------------------------------------------
url: http://www.greatbritishkitchen.co.uk/eh_farflung.htm
host: www.greatbritishkitchen.co.uk
host: www.greatbritishkitchen.co.uk
address: 206.126.20.150
country: UNITED STATES
code3: USA
code2: US
---------------------------------------------------
...etc...
```

As you can see, even though the last result is at a *co.uk* domain, the IP address indicates the server

is actually located in the United States. While this might not be pointing out a great fish and chips conspiracy, geotargeting can give you another tool to use when researching a topic.

Hacking the Hack

This script is only a simple tool. You will make it better, no doubt. The first thing you can do is implement a more efficient way to query the IP-to-Country database. Storing data from *ip-to-country.csv* in a database would speed up script startup time by several seconds. Also, the answers to address-to-country queries could be obtained much faster.

You might ask if it would be easier to write a spider that doesn't use the Google API and instead downloads page after page of results returned by Google at http://www.google.com. Yes, it is possible, and it is also the quickest way to get your script blacklisted for breaching Google's user agreement. Google is not only the best search engine, it is also one of the best-monitored sites on the Internet.

Jacek Artymiak

PREV

Hack 32. Bring the Google Calculator to the Command Line

Perform feats of calculation on the command line, powered by the magic of the Google calculator.

Everyone, whether they admit it or not, forgets how to use the Unix `dc` command-line calculator a few moments after they figure it out for the nth time and stumble through the calculation at hand. And, let's face it, the default desktop (and I mean *computer desktop*) calculator usually doesn't go beyond the basics: add, subtract, multiply, and divide; if you're lucky, you have some grouping ability with clever uses of M+, M-, and MR.

What if you're interested in more than simple math? I've lived in the U.S. for years now and still don' know a yard from three feet, let alone converting ounces to grams or stones to kilograms. This is where the Google Calculator comes to the rescue.

Type in any simple arithmetic or unit conversion into the Google Search form, and you receive an answer instantly. Want to know how far 25 miles is in kilometers? Type `25 miles in kilometers` into the form at Google, click Search, and you get the answer shown in Figure 2-14.

Figure 2-14. A Google calculator answer

Not even your pocket calculator can convert miles into kilometers if you don't know the formula.

This two-line PHP script by Adam Trachtenberg (http://www.trachtenberg.com) brings the Google calculator to your command line so you don't have to skip a beator open your browserwhen you neec to calculate something quickly.

The Code

The script uses PHP (http://www.php.net), better known as a web-programming and templating language, on the command line, passing your calculation query to Google, scraping the returned results, and dropping the answer into your virtual lap.

This hack assumes PHP is installed on your computer and lives in the */usr/bin* directory. If PHP is somewhere else on your system, alter the path on the first line accordingly (e.g., *#!/usr/local/bin/php5*). If you're running PHP on Windows, be sure the path to *php.exe* is in your system PATH variable found in My Computer Properties Advanced Environment Variables.

Save the following code to a file called *calc* in your path (I keep such things in a *bin* in my home directory):

```
#!/usr/bin/php
<?php
preg_match_all('{<b>.+= (.+?)</b>}',
  file_get_contents('http://www.google.com/search?q=' .
     urlencode(join(' ', array_splice($argv, 1)))), $matches);
print str_replace('<font size=-2> </font>', ',',
  "{$matches[1][0]}\\n");
?>
```

Make the code available to run by typing `chmod +x calc` on the command line.

Running the Hack

Invoke your new calculator on the command line ["How to Run the Hacks" in the Preface] by typing `calc` (or `./calc` if you're in the same directory and don't feel like fiddling about with paths) followed by any Google calculator query that you might run through the regular Google web search interface.

Windows users need to preface the command with `php` to let the computer know the script should be run by *php.exe*. In other words, type `php calc` instead of `calc`.

Here are a few examples:

```
% calc 21 * 2
42
% calc 26 ounces + 1 pint in ounces
42 US fluid ounces
% calc pi
3.14159265
```

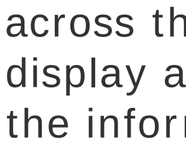
```
% calc 300 feet in meters
91.44 meters
% calc answer to life, the universe and everything
42
```

If your shell gives you a parse error or returns garbage, try placing the calculation inside quotation marks.

There's absolutely no error checking in this hack, so if you enter something that Google doesn't think is a calculation, you'll likely get garbage or nothing at all. Likewise, remember that if Google changes its HTML output, the regular expression could fail; after all, as we point out several times in this book, scraping web pages is a brittle affair. That said, if this were made more robust, it'd no longer be a hack, now would it?

PREV

Hack 33. Build Your Own Google Search Feeds



Keep your finger on the pulse of Google by monitoring Google search results in your favorite newsreader.

Many Google searches are disposable: once you perform the search and find what you're looking for, you don't need to revisit that search again. Other Google searches are recurring: the keywords are topics you frequently revisit. Imagine you build robots at home, and you want to keep up with robotics sites. Most likely, you'd search for phrases such as`"home robotics"`, `"lego mindstorm"`, or `"robotic automation"` periodically to see what's bubbling up to the top of Google search results. Ever searching for your own name to find mentions of yourself across the Web is a perfect recurring search.

Google's index is constantly in flux, and keeping close track of recurring queries by hand would be a tedious job. You could copy the search results, run the query again the next day, and compare the two to see which sites weren't there the last time. Luckily, computers are much better at tedious tasks, and spotting new results in recurring searches is a perfect task for a*news feed*.

News feeds are structured XML documents intended to be read by machines rather than humans, and they've revolutionized how people read sites on the Web. Instead of browsing hundreds of pages across the Web every day, you can use software called*newsreaders* to subscribe to news feeds and display any new information in a friendly, consistent format. But news articles aren't the only type of the information that can be stored in feeds, and this hack shows how to build your own Google search feed.

Unfortunately, Google doesn't offer news feeds of its search results, but with a bit of Perl and the Google API, you can start building your own feeds in no time.

The Code

This script accepts a Google search query and returns an RSS news feed you can add to any newsreader. You'll need`SOAP::Lite` (http://soaplite.com) to talk with the Google API, a local copy of the Google Search WSDL file (http://api.google.com/GoogleSearch.wsdl), and your own Google API key. Save the following code to a file called*google_feed.pl*:

```
#!/usr/local/bin/perl
# google_feed.pl
#
# Builds an RSS feed based on a Google search using
# the Google API.
#
# Usage: google_feed.pl <insert query>
```

```
use strict;
use SOAP::Lite;

# Your Google API developer's key
my $google_key='insert your key';

# Full path to the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Set the Number of loops (10 results/loop)
my $loops = 2;

# Grab the query from the command line
# join(' ', @ARGV)
my $query = join(' ', @ARGV) or die "Usage: perl google_feed.pl <query>\\n";

# Start the RSS file
print <<"END_HEADER";
<?xml version="1.0"?>
<rss version="2.0">
<channel>
<title>Google Search: $query</title>
<link>http://www.google.com/search?q=$query</link>
<description>A Google search generated with google_feed.pl</description>
<language>en-us</language>
END_HEADER

# Create a new Soap::Lite instance
my $google_search  = SOAP::Lite->service("file:$google_wsdl");

for (my $offset = 0; $offset <= ($loops-1)*10; $offset += 10) {

# Query Google for they keyword, keywords, or phrase.
my $results = $google_search ->
  doGoogleSearch(
    $google_key, $query, $offset, 10, "true", "",  "false",
    "", "latin1", "latin1"
  );

last unless @{$results->{resultElements}};

# Loop through results, creating RSS item nodes
foreach my $result (@{$results->{resultElements}}) {
   my $title = $result->{title} || "no title";
   my $link = $result->{URL};
   $link =~ s!&!&amp;!gis;
   my $desc = $result->{snippet} || "no snippet";
   print "<item>\\n";
   print "   <title>$title</title>\\n";
   print "   <link>$link</link>\\n";
   print "   <description><![CDATA[$desc]]></description>\\n";
   print "</item>\\n";
```

```
}
}

# Finish the RSS File
print "</channel>\\n";
print "</rss>";
```

The five `print` commands toward the end of the script determine how RSS items appear in your newsreader. As you can see, each RSS item includes a title, link, and description, much like each iter on a Google Search results page.

Running the Hack

Run the code from a command prompt and pipe the results to a file, like this:

```
google_feed.pl
                    insert query > insert output file
```

Sticking with the example, constructing a feed for the query*home robotics* would look something like this:

```
google_feed.pl home robotics > home_robotics.xml
```

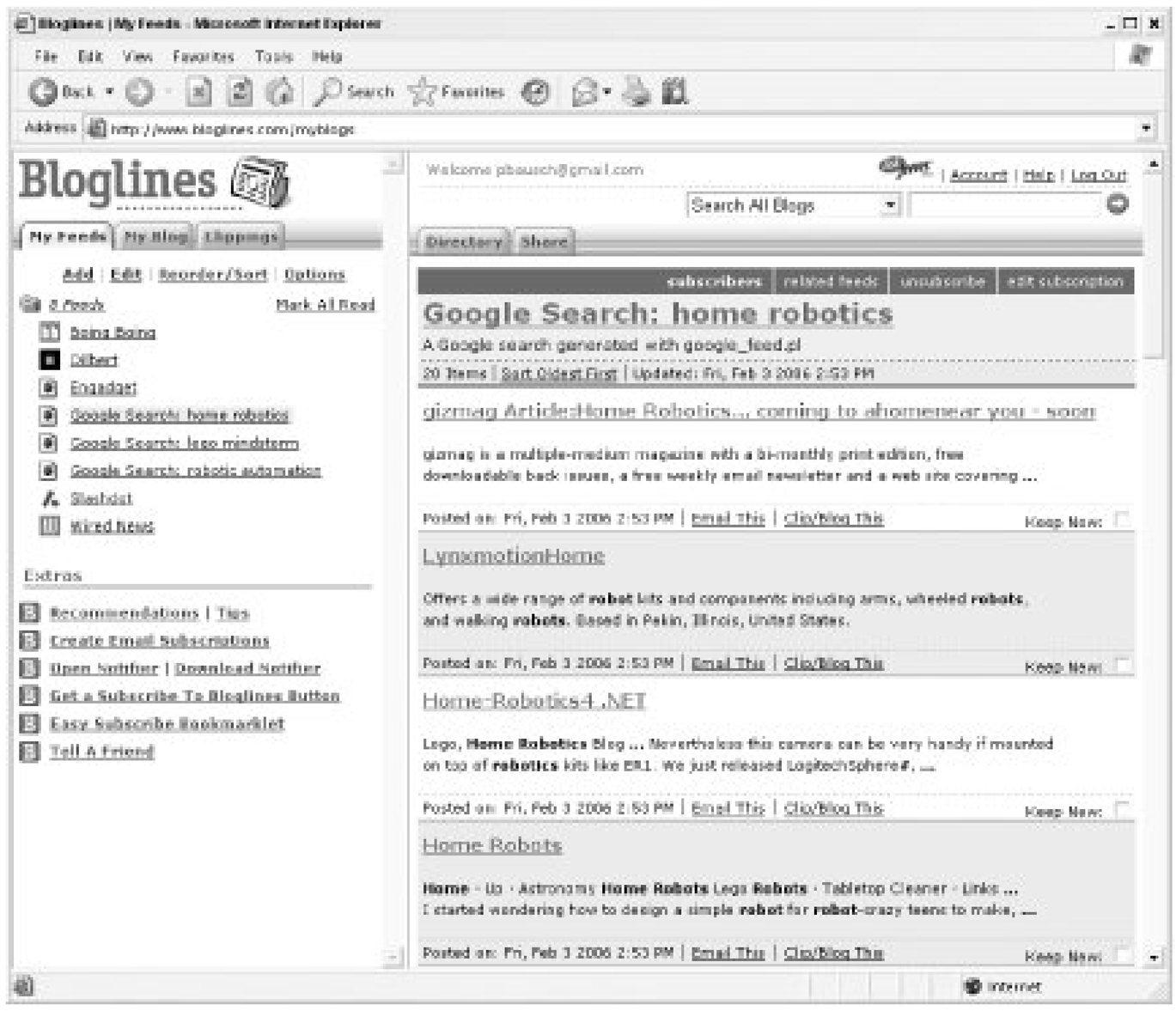
Now that your output file is ready to go, upload it to a publicly addressable web site, which should look like this:

`http://www.example.com/home_robotics.xml`

To make this hack useful, keep the feed up to date by generating the file on a regular schedule. Use `cron` on Unix-based machines or the Windows Scheduler to run the command once a day.

With your URL in hand, and the script updating every day in the background, you can add the new feed to your favorite newsreader. Figure 2-15 shows the feed in the Bloglines (http://www.bloglines.com) web-based newsreader.

Figure 2-15. Viewing a Google Search feed at Bloglines



The first time you read the feed in your newsreader, you'll see all 20 search results in the feed. But as you read the feed periodically, you'll see only search results that are new in the top 20 results. You'll have a quick look at links that are breaking into the top results of your favorite topics, saving you the trouble of running and rerunning the query yourself.

Hacking the Hack

This script works around the Google API's 10-result limit[Hack #93] to include 20 results in the feed. If you want to go even deeper into a topic, simply change the number of loops you want the script to do. If you want 30 results in your feed, edit the value for the`$loops` variable, like this:

```
my $loops = 3;
```

Keep in mind that as you go deeper into the search results, you get more churn in the links that appear there. So you'll find that a feed with 40 results shows you more new sites in your newsreader than a feed with 20 results. You should adjust your feeds to match your appetite for new information

Hack 34. Search Google by Link Graph

Use Google's Web Services API and a Flikr-style link graph to search Google.

Google is a great search engine, but sometimes I find myself looking at the page snippets more than I d around the search term. It's a fascinating way to get more insight into a search phrase.

The Code

Save the code in Example 2-1 as *index.php* .

A DHTML link graph that uses Google as a data source

```
<?php
require_once("Services/Google.php");

$ignore = array( 'the','for','and','with','the','new','are','but','its','that','was',
'your', 'yours', 'also', 'all', 'use', 'could', 'would', 'should', 'when',
    'they',
'far', 'one', 'two', 'three', 'you', 'most', 'how', 'these', 'there', 'now',
    'our',
'from', 'only', 'here', 'will' );
$ignorehash = array(  );
foreach( $ignore as $word ) { $ignorehash[ $word ] = 1; }

$term = "Code Generation";
if( array_key_exists( 'term', $_GET ) )
  $term = $_GET['term'];

$key = "
                    GOOGLE_KEY
                  ";

$google = new Services_Google( $key );
$google->queryOptions['limit'] = 50;
$google->search( $term );

$data = array(  );
foreach($google as $key => $result)
{
  $data []= array(
    'title' => $result->title,
```

```
    'snippet' => $result->snippet,
    'URL' => $result->URL
  );
}

function jsencode( $text )
{
  $text = preg_replace( '/\\'/', '', $text );
  return $text;
}

function get_words( $text )
{
  $text = preg_replace( '/<(.*?)>/', '', $text );
  $text = preg_replace( '/[.]/', '', $text );
  $text = preg_replace( '/,/', '', $text );
  $text = html_entity_decode( $text );
  $text = preg_replace( '/<(.*?)>/', '', $text );
  $text = preg_replace( '/[\\'|\\"|\\-|\\+|\\:|\\;|\\@|\\/|\\\\\\\\|\\#|\\!|\\(|\\)]/',
    $text );
  $text = preg_replace( '/\\s+/', ' ', $text );

  $words = array(  );
  foreach( split( ' ', $text ) as $word )
  {
    $word = strtolower( $word );
    $word = preg_replace( '/^\\s+/', '', $word );
    $word = preg_replace( '/\\s+$/', '', $word );
    if( strlen( $word ) > 2 )
      $words []= $word;
  }
  return $words;
}

$found = array(  );

$id = 0;
foreach( $data as $row )
{
  $row['id'] = $id; $id += 1;

  $words = @get_words( $row['snippet'] );
  foreach( $words as $word )
  {
    if ( !array_key_exists( $word, $found ) )
    {
      $found[$word] = array(  );
      $found[$word]['word'] = $word;
      $found[$word]['count'] = 0;
      $found[$word]['rows'] = array(  );
    }
    $found[$word]['count'] += 1;
```

```
    $found[$word]['rows'][$row['URL']] = $row;
  }
}

$good = array(  );

foreach( array_keys( $found ) as $text )
{
  if ( $found[$text]['count'] > 1 && array_key_exists( $text, $ignorehash ) ==
    false )
    $good []= $found[$text];
}

$min = 1000000;
$max = -1000000;

function row_compare( $a, $b ) { return strcmp( $a['word'], $b['word'] ); }

usort( $good, 'row_compare' );

foreach( $good as $row )
{
  if ( $row['count'] < $min ) $min = $row['count'];
  if ( $row['count'] > $max ) $max = $row['count'];
}

$ratio = 10.0 / (float)( $max - $min );
?>
<html>
<head>
<style type="text/css">
.word-link { line-height: 18pt; }
.title { border-bottom: 1px dotted black; margin-top: 5px; }
.snippet { margin-left: 20px; font-size:small; margin-top: 5px; margin-bottom: 5px; }
</style>
<script language="Javascript">
var pages = [
<?php
foreach( $data as $row ) {
?>
{
  url: '<?php echo( $row['URL'] ); ?>',
  snippet: '<?php echo( jsencode( $row['snippet'] ) ); ?>',
  title: '<?php echo( jsencode( $row['title'] ) ); ?>'
},
<?php
}
?>
];

function display( items )
{
```

```
  var obj = document.getElementById( 'found' );
  var html = "";
  for( i in items )
  {
    var p = pages[ items[ i ] ];
    html += "<div class=\\"title\\"><a href=\\""+p.url+"\\" target=\\"_blank\\">"+p.
        title+"</a></div>";
    html += "<div class=\\"snippet\\">"+p.snippet+"</div>";
  }
  obj.innerHTML = html;
}
</script>
</head>
<body>
<table width="600" cellspacing="0" cellpadding="5">
<tr>
<td colspan="2">
<form>
Search term: <input type="text" name="term" value="<?php echo($term); ?>" /> 
<input type="submit" value="Search">
</form>
</td>
</tr>
<tr>
<td width="50%" valign="top">
<?php
foreach( $good as $row )
{
$val = (float)( $row['count'] - $min );
$fontsize = floor( 10.0 + ( $val * $ratio ) );
$row_ids = array(  );
foreach( $row['rows'] as $r ) { $row_ids []= $r['id']; }
$rows = join(',', $row_ids );
?>
<a class="word-link" href="javascript:display([<?php echo($rows); ?>]);" style="font-si
<?php } ?>
</td>
<td width="50%" id="found" valign="top">
</td>
</tr>
</table>
</body>
</html>
```

This script is a combination of PHP and JavaScript. The PHP uses the `Services_Google PEAR` module [Hacl the results and breaks up the text into words. It counts the number of hits on each word and stores tha the page.

After that, it's up to the browser, which displays the found terms on the lefthand side of the display. The

on the righthand side of the display to show the found articles.

All of this occurs in the JavaScript `display( )` function.

Running the Hack

Edit the file to replace the value of `$key` with the value that you get when you sign up for Google's Web A
module [Hack #2 in PHP Hacks]. The final step is to upload the *index.php* file to the server and browse t

Figure 2-16. Searching for

The lefthand column is showing me all of the words that show up several times in the snippet associatec
which makes perfect sense. But there are some interesting ones as well, such as the names of the other

Clicking on any one of these items will list the pages that had that word in the snippet, as shown in Figu

Figure 2-17. Clicking on a snippet ter

I wrote this little page for this book as a test of the Google Web Services API, but it's turned out to be m
this information to a whole new level.

See Also

- "Create Link Graphs" [Hack #24 in PHP Hacks]

Jack D. Herrington

Hack 35. Download Google Videos as AVI Files

With a little digging, you can download videos from Google Video to your computer for safekeeping.

Google Video (http://video.google.com) gathers video files from around the Web into one convenient place. You can search for videos about specific topics, browse through results, and watch the videos within your browser without leaving Google. For example, a search on Google Video for`"google hacks"` yields a handful of videos, including an appearance by*Google Hacks* coauthor Rael Dornfest on a show called *The Screen Savers*. Click on the result, and the video starts to play in your browser, as shown in Figure 2-18.

Figure 2-18. A video playing in the browser

You can watch the entire video in your browser if you like, and even send it to others or put a copy on your site. If you want to keep a copy of the video locally on your computer, however, things get trickier.

You might have noticed the big Download button inFigure 2-18, but, at the time of this writing,

clicking the button doesn't download the video as you might expect. If you've installed the Google Video Player, clicking the Download button downloads a special text file that tells the Google Video Player the location of the video online. If you haven't installed the Google Video Player, clicking the Download button starts a download of the Google Video Player.

If you're perfectly happy with your current video player, you might be frustrated by the ways Google tries to control how you watch video files. This hack shows how to download videos and convert them to play a more widely viewable format.

Converting FLV Video to AVI

Video at Google Video is in the Macromedia Flash Video (FLV) formata format well suited for playing within a browser, but not widely supported among desktop video players. However, finding the FLV version of a Google Video is the first step to converting the video to something more widely supported.

If you choose View→Page Source on any Google Video page and take a look at the HTML, you'll find a JavaScript function called `insertFlashHtmlOnLoad( )` at the top of the page. The JavaScript code within this function contains the URL of the original FLV file hosted on Google's servers, and some simple Perl code can find that URL and download the file.

Once the FLV file is on your local computer, a program called MEncoder can convert the video to the widely used AVI format. So the first step in running this hack is to install MEncoder, a command-line tool included with the freely available MPlayer (http://www.mplayerhq.hu). Download and install MPlayer, and be sure to note your installation location.

The Code

The Perl script in this hack accepts a Google Video URL, finds and downloads the FLV version of the video, and converts the video with MEncoder. Be sure to set your path to *mencoder.exe* at the top of the script. You'll also need the `LWP::Simple` module (http://search.cpan.org/dist/libwww-perl/lib/LWP/Simple.pm) to scrape Google Video pages, and `URI::Escape` (http://search.cpan.org/~gaas/URI-1.35/URI/Escape.pm) to decode the JavaScript at the top of the page.

Add the following code to a file called *grabVideo.pl*:

```
#!/usr/bin/perl
#
# grabVideo.pl
#
# Given a Google Video URL, this script will
# save a local copy of the video and convert
# the video to the more widely watchable AVI
# format.
#
# This script requires the MEncoder command-
# line tool available with MPlayer:
#
```

```
# http://www.mplayerhq.hu/
#
# Be sure to set your path to mencoder.exe.

use strict;
use LWP::Simple;
use URI::Escape;

# MEncoder location
my $mencoder = "c:\\\\mplayer\\\\mencoder.exe";

# Get the Google Video URL
print "Paste in a Google Video URL and press Enter.\\n% ";
my $url = <STDIN>;

# Scrape the Google Video page
my $response = get($url);

# Find the video file
while ($response =~ m!videoUrl\\\\u003d(.*?)\\\\"!gis) {
    my $videoURL = $1;
    $videoURL = uri_unescape($videoURL);
    $videoURL =~ s!\\\\u003d!=!gs;

    # Find the video filename
    my $head = head($videoURL);
    my $filename = $head->{_headers}->{'content-disposition'};
    $filename =~ s!attachment; filename=!!gis;

    # Download the video file
    print "Downloading $filename...\\n";
    getstore($videoURL,$filename);

    # Make sure downloaded file is there
    if (-e $filename) {
        # Change the extension
        my $newfilename = $filename;
        $newfilename =~ s!flv!avi!gis;
        print "Converting to $newfilename...\\n";

        # Use MEncoder to convert to AVI
        my $cmd = "$mencoder $filename -ofps 15";
                  $cmd .= " -vf scale=300:-2 -oac lavc";
                  $cmd .= " -ovc lavc -lavcopts";
                  $cmd .= " vcodec=msmpeg4v2:acodec=mp3:abitrate=64";
                  $cmd .= " -o $newfilename";

        system($cmd) == 0
           or die "Can't re-encode video: $?";

        print "Removing $filename...\\n";
        unlink($filename);
```

```
        print "Saved $newfilename!";
    }
}
```

Running the Hack

Run the script from the command line, like so:

```
% perl grabVideo.pl
```

Once you start the script, you're prompted to paste in a Google Video URL:

```
Paste in a Google Video URL and press Enter.
%
```

If you want the video of Rael, try pasting in the following URL:

```
http://video.google.com/videoplay?docid=6272710823098922710&q=google+hacks
```

The script fetches the FLV version of the video and saves it to your local computer. From there, the script calls MEncoder, and you'll probably see a lot of video-encoding information fly by in your command prompt. Don't worry; that's simply MEncoder doing its job.

Once the script is finished, you'll have an AVI version of the file suitable for playing with just about any video player, including Windows Media Player, as shown in Figure 2-19.

Figure 2-19. A Google Video clip in Windows Media Player

Google wants you to use its player and its formats for video, but with a little scripting, you can open up Google Video to a larger world.

Chapter 3. News and Blogs

Hacks 36–46

The Internet is a worldwide conversation, and nowhere is that better reflected than in the flow of news coverage by "official" news sources and bloggers alike, as well as in the tangled discussions of Usenet news and mailing lists. Google trawls through our conversations, threads them together, tidies them up (just a tad), and reflects them back at us in Google News, Google Blog Search, and Google Groups. Google also gives anyone the opportunity to take part in the worldwide conversation with its free blog tool Blogger.

PREV

Google News

At the time of this writing, Google News (http://news.google.com) culls over 4,500 news sourcesfrom the *Scotsman* to the *China Daily*, from the *New York Times* to the *Minneapolis Star Tribune*.

The front page, shown in Figure 3-1, is updated algorithmically without any involvement by puny humansaside, of course, from those writing the news in the first placeseveral times a day. The "most relevant news" rises to the top.

Figure 3-1. The Google News front page

Stories are organized into clusters, drawing together coverage and photographs from various news sources around the Web. Click the "all*n* related" link for a list of all stories falling within that cluster. Click "sort by date" to see how the story unfolded across sources over time.

All of this doesn't apply just to the front page, but to all the newspaper-like sections within: World, U.S., Business, Sci/Tech, Sports, Entertainment, and Health.

For a text-only and PDA/smartphone-friendlier version of Google News, click the Text Version link in the left column or point your browser at http://news.google.com/news?ned=tus. You might notice that it takes a little longer to load; this is because each section, from Top Stories to Health, is combined into one text-only page.

Google News Search Syntax

When you search Google News, the default is to search for your query keywords anywhere in the news article's headline, story text, source, or URL.

`iht` finds stories that appear in the *International Herald Tribune* (http://www.iht.com), even if "iht" appears nowhere in the headline, story, or source's proper name.

Google News Search uses basic Boolean just like Google's Web Search ["Basic Boolean" in Chapter 1].

Google News supports the following special search syntax:

`intitle:`

Finds words in an article headline:

```
intitle:beckham
```

An `allintitle:` variation finds stories in which all the search keywords appear in an article headlineeffectively the same as using `intitle:` before each keyword:

```
allintitle:miners strike benefits
```

`intext:`

Finds search terms in the body of a story:

```
intext:"crude oil"
```

An `allintext:` variation finds stories in which all the search keywords appear in article texteffectively the same as using `intext:` before each keyword:

```
allintext:US stocks rebound
```

`inurl:`

Looks for particular keywords in a news story's URL:

```
ipod inurl:reuters
```

`source`:

Finds articles from a particular source. Unfortunately, Google News does not offer a list of its over 4,500 sources, so you have to guess a little. Also, you need to replace any spaces in the source's name with underscore characters; e.g., the *New York Times* becomes `new_york_times` (case-insensitive):

```
miners source:international_herald_tribune
"international space station" source:new_york_times
```

`location`:

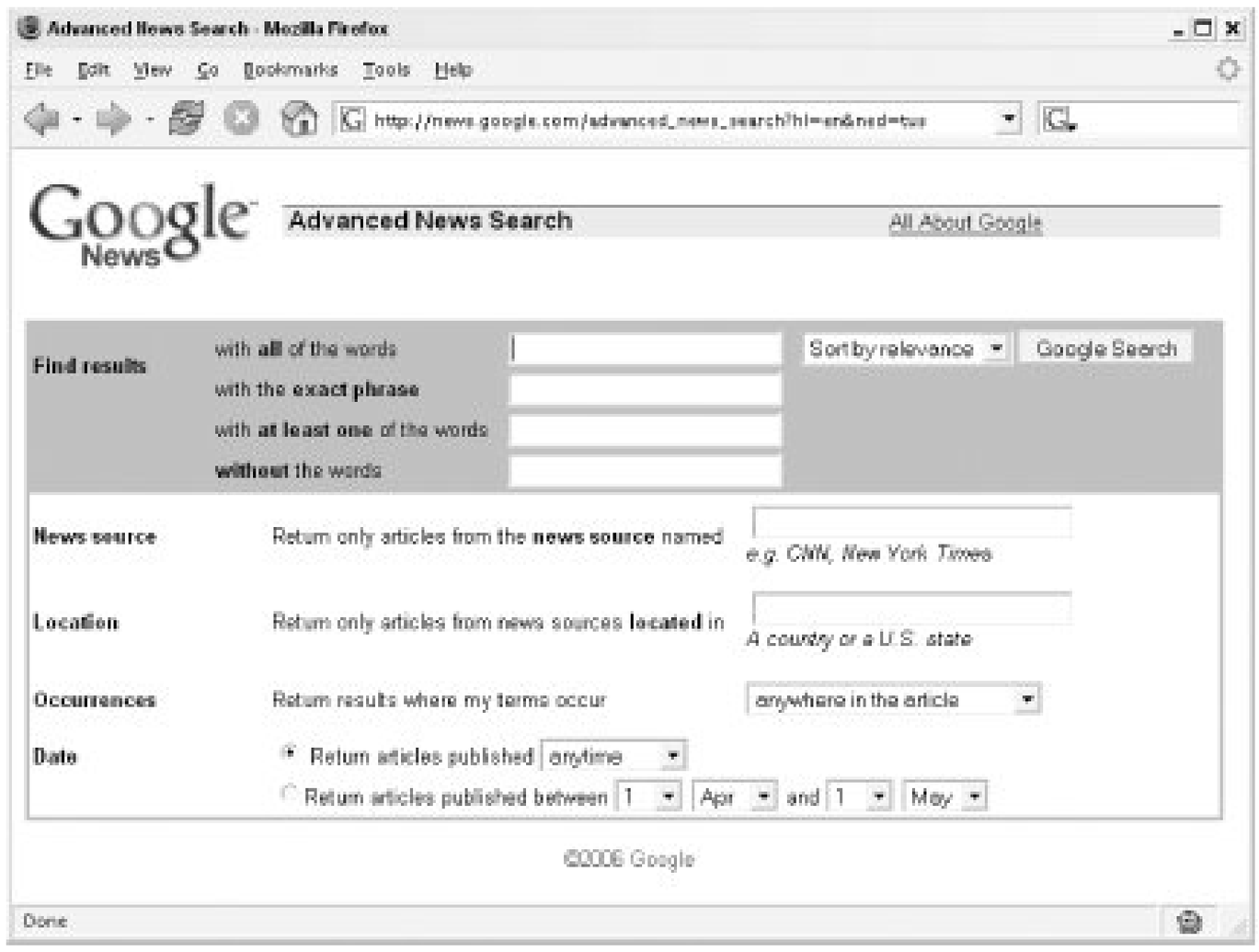
Filters articles from sources located in a particular country or state. For country names consisting of more than one word, replace any spaces with underscore characters; e.g., South Africa becomes `south_africa` (case-insensitive). In the case of state names, use official abbreviations such as `ca` for California and `id` for Idaho:

```
"organic farming" location:france
election 2004 location:ca
```

Advanced News Search

Google Advanced News Search, shown in Figure 3-2, is much like the Advanced Web Search. It provides access to the Google News special syntax from the comfort of a web form. Notice the set of fields and pull-down menus associated with Date; use these to search for articles published in the las hour, day, week, month, or between any two particular days.

Figure 3-2. The Google Advanced News Search form



Fill in the fields, click the Search button, and notice how your query is represented in the search box on the results page.

Making the Most of Google News

The best thing about Google News is its clustering capability. On an ordinary news search engine, a breaking news story can overwhelm search results. For example, in late July 2002, a story broke that hormone replacement therapy might increase the risk of cancer. Suddenly, using a news search engine to find the phrase `"breast cancer"` was an exercise in futility, because dozens of stories around the same topic were clogging the results page.

This doesn't happen when you search the Google News engine because Google groups similar stories by topic. You'd find a large cluster of stories about hormone replacement therapy, but they'd be in one place, leaving you to find other news about breast cancer.

Some searches cluster easily; they're specialized or tend to spawn limited topics. But other queries (such as `"George Bush"`) spawn lots of results and several different clusters. If you need to search for a famous name or a general topic (such as `crime`), narrow your search results in one of the following ways:

- Add a topic modifier that will significantly narrow your search results, as in: `"George Bush" environment crime arson`.
- Limit your search with one of the special syntaxes. For example: `intitle:"George Bush"`.
- Limit your search to a particular source. Be aware that while this works well for a major breaking news story, you might miss local stories. If you're searching for a major American story, CNN is a good choice (`source:cnn`). If the story you're researching is more international

in origin, the BBC works well (`source:bbc_news`).

Receiving Google News Alerts

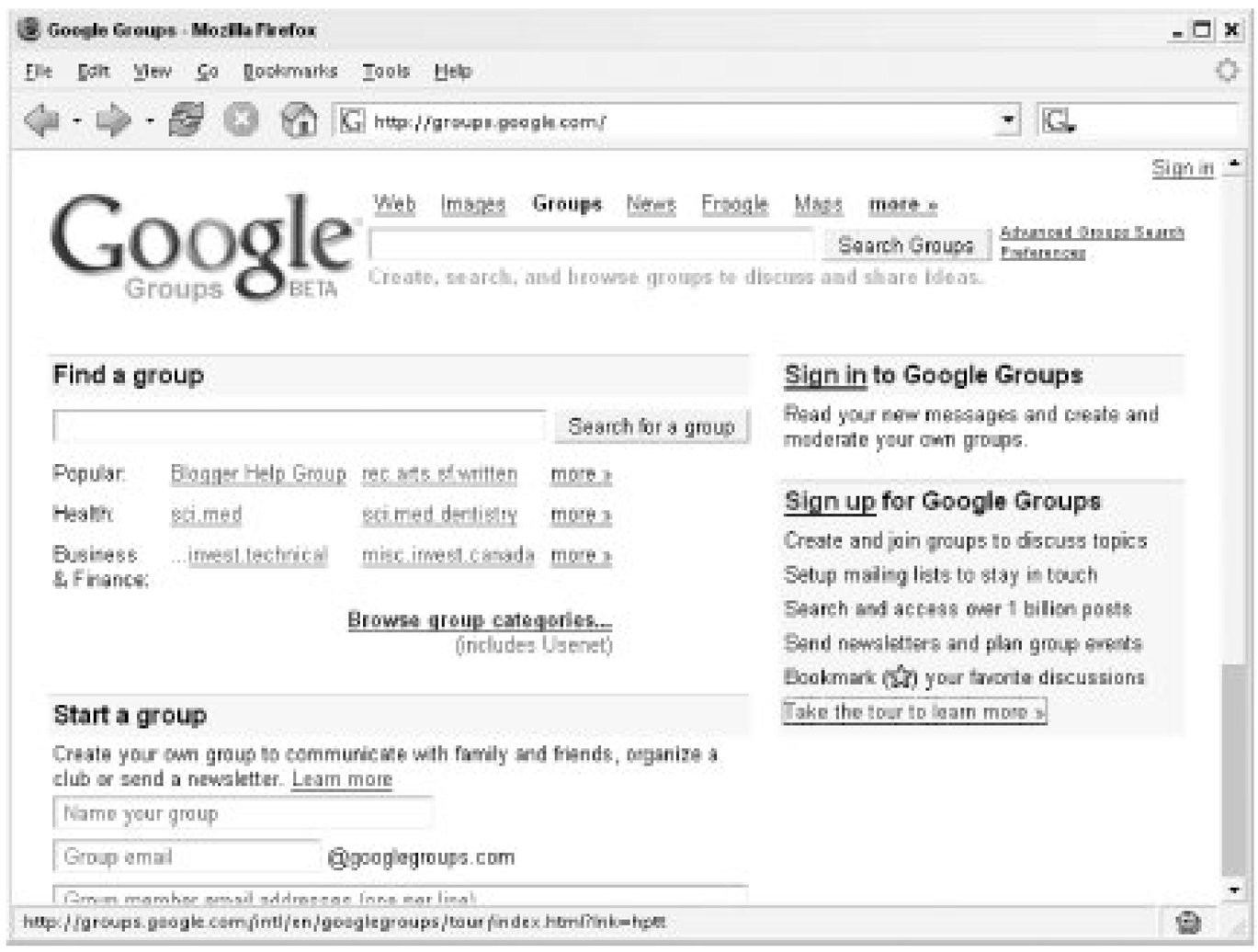
Google Alerts keep tabs on your Google News searches [Hack #47], notifying you if any news stories appear that match your search criteria. They're easy to set up, alter, and deleteand they're free.

PREV

Google Groups

Usenet groups, text-based discussion groups that cover literally hundreds of thousands of topics, have been around since long before the World Wide Web.Deja News used to be *the* repository of Usenet information until it sold off its archive to Google in early 2001. Google filled it out even furthe and relaunched it as Google Groups (http://groups.google.com). Its search interface, shown in Figure 3-3, is rather different from the Google Web Search, as all messages are divided into groups, and the groups themselves are divided into topics called hierarchies.

Figure 3-3. The Google Groups home page



The Google Groups archive begins in 1981 and covers up to the present day. Just shy of 850 million messages are archived. As you might imagine, that's a pretty big archive, covering literally decades of discussion. Stuck in an ancient computer game? Need help with that sewing machine you bought in 1982? You might be able to find the answers here.

Google Groups also allows you to form your own ad hoc groups to collaborate on or discuss topics. See the Google Groups tour (http://groups.google.com/intl/en/googlegroups/tour/index.html) for instructions on how to create your own newsgroup. You have to first choose where you want your group to be categorized, which means understanding the hierarchy.

Ten Seconds of Hierarchy Funk

There are regional and smaller hierarchies, but Usenet relies on *alt*, *biz*, *comp*, *humanities*, *misc*, *news*, *rec*, *sci*, *soc*, and *talk*. Most Usenet groups are created through a voting process and are put under the hierarchy that's most applicable to the topic. But you can create a group that's available via Google Groups without any input.

Browsing Groups

From the main Google Groups page, you can browse through the list of groups by picking a hierarchy from the front page. You'll see there are subtopics, sub-subtopics, sub-sub-subtopics, andwell, you get the picture. For example, in the *comp* (computers) hierarchy, you'll find the subtopic *comp.sys*, or computer systems. Beneath that lie 75 groups and subtopics, including *comp.sys.mac*, a branch of the hierarchy devoted to the Macintosh computer system. There are 24 Mac subtopics, one of which is *comp.sys.mac.hardware*, which has, in turn, 3 groups beneath it. Once you've drilled down to the most specific group applicable to your interests, Google Groups presents the postings themselves, sorted in reverse chronological order.

This strategy works fine when you want to read a slow (i.e., containing little traffic) or moderated group, but when you want to read a busy, free-for-all group, you may wish to use the Google Groups Search engine. The search on the main page works much like the regular Google search, except for the Google Groups tab and the associated group and posting date that accompanies each result.

The Advanced Groups Search (http://groups.google.com/advanced_group_search), however, looks much different. You can restrict your searches to a certain newsgroup or newsgroup topic. For example, you can restrict your search as broadly as the entire *comp* hierarchy (`comp*` would do it) or as narrowly as a single group such as *comp.robotics.misc*. You can restrict messages to subject and author, or restrict them by message ID.

> Of course, any options on the Advanced Groups Search page can be expressed via a little URL hacking ["Understanding Google URLs" in Chapter 1].

Possibly the biggest difference between Google Groups and Google Web Search is the date searching. With Google Web Search, date searching is notoriously inexact (*date* refers to when a page was added to the index rather than when the page was created). Each Google Groups message is stamped with the day it was actually posted to the newsgroup. Thus, the date searches on Google Groups are accurate and indicative of when content was produced.

Google Groups Search Syntax

By default, Google Groups looks for your query keywords anywhere in the posting subject, body, group name, or author name. It uses the same basic Boolean as Google Web Search ["Basic Boolean" in Chapter 1].

Google Groups is an archive of conversations. Thus, when you're searching, you'll be more successful if you try looking for conversational and informal language, not the carefully structured language found on Internet siteswell, some Internet sites anyway.

And, thanks to some special syntax, you can do some precise searching if you know the magic incantations:

`insubject:`

Searches posting subjects for query words:

```
insubject:rocketry
```

`group:`

Restricts your search to a certain group or set of groups (topic). The `*` (asterisk) wildcard modifies a `group:` syntax to include everything beneath the specified group or topic. `rec.humor*` or `rec.humor.*` (effectively the same) find results in the group *rec.humor*, as well as *rec.humor.funny*, *rec.humor.jewish*, and so forth:

```
group:rec.humor*
group:alt*
group:comp.lang.perl.misc
```

`author:`

Specifies the author of a newsgroup post. This can be a full or partial name, or even an email address:

```
author:fred
author:"fred flintstone"
author:flintstone@bedrock.gov
```

Mixing Syntaxes in Google Groups

Google Groups is much more friendly to syntax mixing ["Mixing Syntax" in Chapter 1] than Google Web Search. You can mix any two or more syntaxes in a Google Groups Search, as exemplified by the following typical searches:

```
intitle:literature group:humanities* author:john
intitle:hardware group:comp.sys.ibm* pda
```

Some common search scenarios

There are several ways you can mine Google Groups for research information. Remember, though, to view any information you get here with a certain amount of skepticism. Usenet is just hundreds of thousands of people tossing around links; in that respect, it's just like the Web.

Tech support

Ever used Windows and discovered there's a program running you've never heard of? Uncomfortable isn't it? If you're wondering if HIDSERV is something nefarious, Google Groups can tell you. Just search Google Groups for `HIDSERV`. You'll find that plenty of people had the same question before you did, and it's been answered.

I find that Google Groups is sometimes more useful than manufacturers' web sites. For example, I was trying to install a set of flight devices (a joystick, throttle, and rudder pedals) for a friend. The web site for the manufacturer couldn't help me figure out why they weren't working. I described the problem as best I could in a Google Groups searchusing the name of the parts and the manufacturer's brand nameand, though it wasn't easy, I was able to find an answer.

Sometimes your problem isn't as serious but it's just as annoying. For example, you might be stuck in a computer game. If the game has been out for more than a few months, your answer is probably in Google Groups. If you want answers to an entire game, try the magic word*walkthrough*. So, if you're looking for a walkthrough for Quake II, try the search `"quake ii" walkthrough`. (You don't need to restrict your search to newsgroups; "walkthrough" is a word strongly associated with gamers.)

Finding commentary immediately after an event

With Google Groups, date searching is very precise (unlike date-searching Google's Web index), so it's an excellent way to get commentary during or immediately after events.

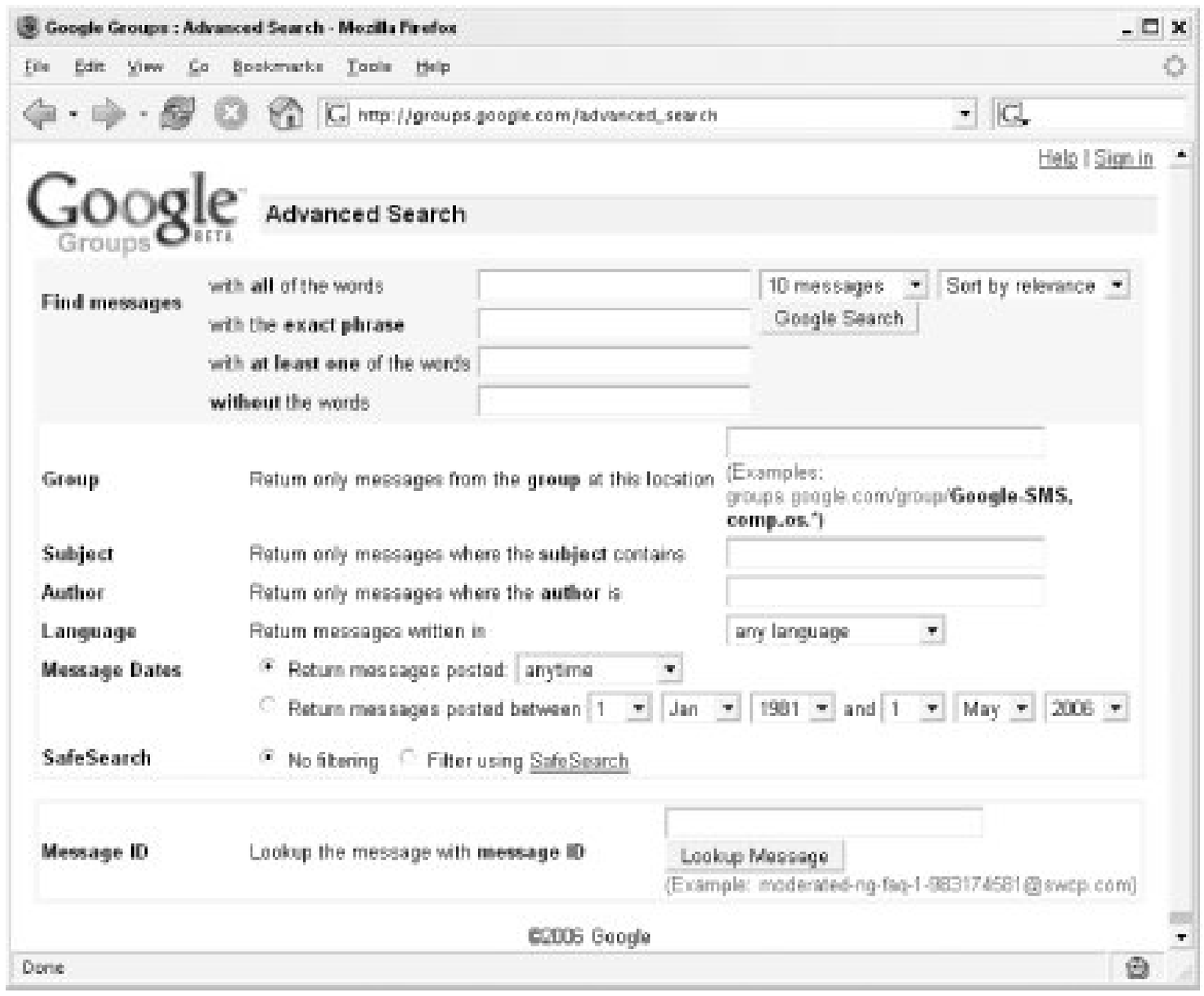
Barbra Streisand and James Brolin were married on July 1, 1998. Searching for `"Barbra Streisand" "James Brolin"` between June 30, 1998 and July 3, 1998 leads to over 48 results, including reprinted wire articles, links to news stories, and commentary from fans. Searching for `"barbra streisand" "james brolin"` without a date specification finds more than 1,800 results.

Usenet is also much older than the Web and is ideal for finding information about an event that occurred before the Web. Coca-Cola released New Coke in April 1985. You can find information about the release on the Web, of course, but finding contemporary commentary would be more difficult. After some playing around with the dates (just because it's been released doesn't mean it's in every store), I found plenty of commentary about New Coke in Google Groups by searching for the phrase `"new coke"` during the month of May 1985. Information included poll results, taste tests, and speculation on the new formula. Searching later in the summer yields information on Coke re-releasing old Coke under the name "Coca-Cola Classic."

Advanced Groups Search

The Advanced Groups Search, shown in Figure 3-4, is much like the Advanced Web Search and Advanced News Search.

Figure 3-4. The Google Groups Advanced Search form



Rather than fiddling with the special syntax detailed earlier, simply fill out the form, hit the Search button, and let Google Groups compose the query for you. You can restrict your search to a specific newsgroup or section of hierarchy (e.g., `comp.os.*`), a particular person, a particular language, or posts arriving in the past 24 hours, week, month, 3 months, 6 months, or year. You can even search for a particular message if you know the message ID. And since Usenet can be just as woolly as the Web, you might want to turn on SafeSearch.

Blogs

On the surface, weblogs (or *blogs* for short) are simply a format for publishing information online by placing new information at the top of the page. But dig a little deeper, and you realize that blogs have changed the way people communicate and consume information.

At the time of this writing, the blog-tracking serviceTechnorati (http://www.technorati.com) estimates that 75,000 new blogs are created every day; over 35 million blogs are already in its index. This global network of blogs (often called the*blogosphere*) shows no signs of stopping, and Google offers some specialized tools to help you tune in and take part.

Blogger

To start publishing in the blogosphere, look no further thanBlogger (http://www.blogger.com), shown in Figure 3-5.

Figure 3-5. Blogger home page

Blogger is a free service that provides everything you need to start writing a blog, including web-hosting space. The signup process literally takes less than five minutes, but don't let its simplicity foc you. With Blogger, you can start multiple blogs, post by email, customize your blogs' designs, collect comments on posts from readers, and publish your blog to a remove site via FTP or Secure FTP.

Blogger.com provides a simple posting interface where you type your rants, raves, opinions, or news into a form. Click Publish Post, and your words are on the Web.

Google Blog Search

Google recognized that blogs are a bit different from standard web sites, so it created a search engine specifically for finding news and commentary on blogs. TheGoogle Blog Search is available at both Blogger (http://blogsearch.google.com) and Google (http://blogsearch.google.com), but both faces use the same index in the background.

Instead of searching the open Web for content, the Google Blog Search finds content in XML news feeds. Because of this, any blogs that don't also publish a news feed are not included in the Google Blog Search index. Also, Google started collecting content for the index when it launched in late 2005, so the index goes much further back in time.

It's also important to note that the Google Blog Search results in page returns that Google feels are the best matches for a particular query. But timeliness is a key aspect of blogs and could be to your search as well. Click Sort by date at the top of the results page to see search results listed from newest to oldestlike a blog!

Google Blog Search Syntax

Use Google Blog Search just as you would Google News Search. You can use the standard Google search syntaxes such as `site:` or `intitle:` to refine your searches. There are also a few special search syntaxes unique to Blog Search:

`blogurl:`

This searches a specific blog by including its URL, like this:
`blogurl:radar.oreilly.com google`

This search finds all mentions of "google" on the O'Reilly Radar blog.

`inblogtitle:`

As you'd expect, this limits a search to blogs with the specified word in its title:
`inblogtitle:ipod battery`

This example searches for the word "battery" among blogs with the word "ipod" in their title.

`inposttitle`:

Searching in post titles can be useful when you want to narrow your search to specific topics. Post titles often include keywords related to the content of a post:

```
inposttitle:ipod iTunes video
```

`inpostauthor:`

This filters posts by an author name, which can be handy if you know who wrote something bu can't remember where you read it:

```
author:paul hacks
```

This query finds posts that use the word "hacks" by people named Paul. Keep in mind that not every blog publishes author information along with each post, so the results are limited to just those blogs with author info.

You can always skip the special syntax and head over to theBlog Search Advanced Search page (http://search.blogger.com/advanced_blog_search) to perform these and other specialized searches such as finding posts within a date range.

Beyond Google for News and Blogs

After a long dry spell, news and blog-related search engines have popped up all over the Internet. Here are my top four:

Rocketinfo (http://www.rocketnews.com)

> Does not use the most extensive sources in the world, but lesser-known press release outlets (such as PETA) and very technical outlets (e.g., OncoLink, BioSpace, Insurance News Net) can be found here. Rocketinfo's main drawback is its limited search and sort options.

Yahoo! Daily News (http://news.yahoo.com)

> Unlike Google News, Yahoo! relies on human editors to assemble its news portal. A 30-day index means that you can sometimes find things that have slipped off the other engines. Yahoo! Daily News provides free news alerts for registered Yahoo! users.

Technorati (http://www.technorati.com)

> Technorati can help you zero in on conversations within the blogosphere. Many blog authors tag their posts with keywords to help Technorati determine how its posts should be categorized, and you can search for posts by tag.

BlogPulse (http://www.blogpulse.com/)

> BlogPulse is geared toward tracking trends across blogs. You can use its Trend Search tool to graph the frequency of mentions of words or phrases across blogs.

Hack 36. Scrape Google News

Scrape Google News Search results to get the latest from thousands of aggregated news sour

Google News, with its thousands of news sources worldwide, is a veritable treasure trove for any news h

This hack does just that, gathering results into a comma-delimited file that can be loaded into a spreads

To find an RSS feed that can be translated into a spreadsheet, run a Google News search and make sure by choosing the "Sort by date" link on the results page or by adding `&scoring=d` to the end of the results

Also, make sure you get the maximum number of results by adding `&num=100` to the end of the results UF

Note the RSS and Atom links on the left side of the results page. These links are news feeds that allow y note the URL. The feed URL should look something like this:

```
http://news.google.com/news?hl=en&ned=us&q=Iraq+War&ie=UTF-8&scoring=d&output=rss
```

Note that the feed separates stories into parts only by title, link, and description. And the description is description.

At the time of this writing, a typical Google News Search result as HTML looks a little something like this

```
<table border=0 width=75% valign=top cellpadding=2 cellspacing=7><tr><td valign=top>
<a href="http://www.localnewsleader.com/brocktown/stories/index.php?action=fullnews&id=1
</font><br></table>
```

While for most of you this is utter gobbledygook, it is probably of some use to those trying to spot patte XML parsing and regular expressions to translate the news into a data-friendly format.

The Code

You'll need a couple nonstandard Perl modules to run this script.`LWP( )` fetches the Google News RSS fe

Save the following code to a file called *news2csv.pl* :

```
#!/usr/bin/perl
# news2csv.pl
# Google News Results exported to CSV suitable for import into Excel.
# Usage: perl news2csv.pl <query>

use strict;
use LWP;
use XML::Simple;
use URI::Escape;

# Grab incoming query
my $query = join(' ', @ARGV) or die "Usage: perl news2csv.pl <query>\\n";
$query = uri_escape($query);

# Start the CSV file
print qq{"title","link","source","date or age", "description"\\n};

# Set the client for fetching pages
my $browser = LWP::UserAgent->new;
$browser->agent("Mozilla/5.01 (windows; U; NT4.0; en-us) Gecko/25250101");

# Fetch the Google RSS Feed for the query
my $feed = "http://news.google.com/news?hl=en&ned=us&q=$query&ie=UTF-8&scoring=d&num=10
my $google = $browser->get($feed);

if (!$google->is_success(  )) {
    die "News feed not found! $google->status_line(  )";
}

# Parse the Google News RSS
my $xmlsimple = XML::Simple->new(  );
my $rss = $xmlsimple->XMLin($google->content);
```

```
# Pick through the items, grabbing the byline
foreach my $item (@{$rss->{channel}->{item}}) {
    my $title = $item->{title};
    my $link = $item->{link};
    my $desc = $item->{description};
    my ($byline, $lonlat);
    while ($desc =~ m!<a href="([^"]+)">(.+?)</a><br>(.+?)<nobr>(.+?)</nobr>.*?<br>(.+?)
        my ($url, $atitle, $source, $date_age, $description) =
                ($1||'',$2||'',$3||'',$4||'', $5||'');
        my $output = qq{"$title","$link","$source","$date_age","$description"\\n};
        $output =~ s!<.+?>!!g; # drop all HTML tags
        # Do some quick conversion of HTML entities
        $output =~ s!&#39;!'!g; # drop all HTML tags
        $output =~ s! ! !g; # drop all HTML tags

        # Send the record to the file
        print $output;
    }

}
```

Running the Script

Run the script from the command line ["How to Run the Hacks" in the the Preface], specifying a Google

```
$ perl news2csv.pl  Iraq >
                news.csv
```

Leaving off the `>` and CSV filename sends the results to the screen for your perusal.

The following output shows some of the 128,000 results returned by a Google News Search for`Iraq` and

```
$ perl news2csv.pl Iraq
"title","link","source","date or age", "description"
"Abu Ghraib Officer Defends Use of Dogs","http://www.localnewsleader.com/brocktown/stori
"'Operation Swarmer' Expected to Last Days","http://www.localnewsleader.com/brocktown/st
```

Each listing actually occurs on its own line.

Opening a CSV file generated with *news2csv.pl* brings up a spreadsheet such as the one in Figure 3-7 .

With this news in this new, sortable format, you can see which news outlets are covering a particular sto

For even more fun dissecting and analyzing Google News search results, you r

Hacking the Hack

You'll want to leave most of the *news2csv.pl* script alone, since it was built to make sense of the Google
you choose. Be sure to keep a comma between each one:

```
my $output =
  qq{"$title","$url","$source","$date_age","$description"\\n};
```

For example, perhaps you want only the URL and title. The line should read:

```
my $output =
  qq{"$url","$title"\\n};
```

That `\\n` specifies a new line, and the `$` characters specify that `$url` and `$title` are variable names; keep

Of course, by default, your output doesn't match the header at the top of the CSV file:

```
print qq{"title","link","source","date or age", "description"\\n};
```

As before, simply change this to match:

```
print qq{"url","title"\\n};
```

PREV

Hack 37. Visualize Google News

Watch stories aggregated by Google News unfold over time, coverage broaden and fade, and hotspots emerge and fade again into the background.

Newsmap (http://www.marumushi.com/apps/newsmap) is a whizbang, Flash-based treemap representation (http://www.cs.umd.edu/hcil/treemap/index.shtml) of the stories flowing through Google News. The Newsmap home page describes it best:

Treemaps are traditionally space-constrained visualizations of information. Newsmap's objective takes that goal a step further and provides a tool to divide information into quickly recognizable bands which, when presented together, reveal underlying patterns in news reporting across cultures and within news segments in constant change around the globe.

Point your web browser at the Newsmap page and click the LAUNCH button to begin.Figure 3-8 shows Newsmap in action.

Figure 3-8. Newsmaps banded layout, focusing on U.S. coverage of business and technology news

Each color-coded band (you'll have to take my word that they're in color) represents a Google News section: from left to right are World, Nation, Business, Technology, Sports, Entertainment, and Health. Notice that I've selected only Business and Technology by checking their associated checkboxes at the bottom-right corner of the page. Also notice that I've selected news only from the U.S. in the Countries tab across the top.

The colors appear in a gradient from brightest ("less than 10 minutes ago") to darkest ("more than 1 hour ago"), such that the latest stories stand right out. The more substantial the band and bigger the enclosed headline, the greater the number of related stories. You can easily spot the freshest and most covered stories: they're the big, bright blocks.

Hover your mouse over any story for a brief description drawn from the primary sourcethe story around which others are clusteredas chosen by Google News.

There's also a Squarified version (Figure 3-9), which I prefer; more so than with the Standard version (Figure 3-8), you can see the spread of coverage across all news categories. Switch between the two layouts by clicking the appropriate Layout button in the bottom-right corner.

Figure 3-9. Newsmap's Squarified layout, drawing from U.S. coverage of news across all Google News categories

Newsmap provides a fascinating bird's-eye view of news as it unfolds on the Web. Here are a couple of my favorite Newsmap settings:

- Select only one news category (World works best) and draw in coverage from two or three countries. Set the layout to Squarified. Now take a gander at the headlines and notice how they differ in title and coverage by country.

- Select only one news category and one country from which to draw sources. Set the layout to Standard. Now meander back through the archive (bottom-left corner) day-by-day or hour-by-hour and watch how the stories unfold over time. Bands widen and narrow, hotspots appear and disappear, and the headline changes right along with the primary source.

PREV

PREV

Hack 38. Map Google News

Google News gathers stories from media outlets across the globe. By plugging Google News i you can visualize where stories are from.

As you browse through stories on Google News (http://news.google.com), you find that every article ex name of the media outlet that published the story, along with the location of that outlet. Here are a few sources as they're listed at Google News:

- Melbourne Herald Sun, Australia
- Fort Wayne News Sentinel, IN
- Monterey County Herald, CA
- NewKerala.com, India

As you study a list of news sources from Google News, patterns start to emerge. For example, U.S. new include the two-letter state abbreviation, while international news stories typically include only the coun location of the news outlet always follows the name of the news outlet after a comma.

With these patterns in mind, it's possible to tie almost every story that flows through Google News to a which means you can create a map of the news outlets and their stories that appear in Google News.

If you've already tried scraping news stories [Hack #36], you know you can access the information at G programmatically, separating the components of a news excerpt into pieces such as title, URL, and exce separates the excerpts even further, isolates the location, and plots the locations on your own Google M

Geocoding

An important aspect of adding locations to a Google Map is*geocoding* : turning plain language locations into a set of coordinates that represent a location's longitude and latitude. The Google Maps API doesn't service, so it's up to every map producer to supply the coordinates for the places they want to map.

Luckily, there are services online that can help you geocode locations.GeoNames (http://www.geonames service that can give you a longitude and latitude for just about any geographic name. If you browse to in `California` , the first result gives the coordinates of the geographic center of California: 37.25, 119.7 offers a web services interface to its data, so you can include this geocoding service in your scripts.

Another piece of the geocoding puzzle is converting abbreviations of physical locations to their full-text e hack, a Perl module called `Geography::USStates` , found at:

http://search.cpan.org/~dionalm/Geography-USStates-0.12/USStates.pm

handles the conversion of *CA* into something GeoNames can understand: *California* .

The following code encapsulates all of this conversion and geocoding into a single set of instructions.

The Code

As you know by now, this code requires several nonstandard Perl modules, so you need to spend some modules before you get started. Here are the required modules:

`LWP` (http://search.cpan.org/~gaas/libwww-perl-5.805/lib/LWP.pm)

This module handles communication between the script and services required to build the map, in Google News to fetch an RSS feed and contacting GeoNames to find coordinates.

`XML::Simple` (http://search.cpan.org/~grantm/XML-Simple-2.14/lib/XML/Simple.pm)

The services return data as XML, and this module lets you access specific pieces of that data.

`HTML::GoogleMaps` (http://search.cpan.org/~nmueller/HTML-GoogleMaps-3/lib/HTML/GoogleMaps.pm)

Instead of writing your own JavaScript to define points on a Google Map, this module generates th

`URI::Escape` (http://search.cpan.org/~gaas/URI-1.35/URI/Escape.pm)

This module escapes invalid characters (such as spaces) into their encoded equivalents for use in

`Geography::USStates` (http://search.cpan.org/~dionalm/Geography-USStates-0.12/USStates.pm)

As mentioned earlier, this module converts a U.S. state abbreviation into its full-text name.

`CGI` (http://search.cpan.org/~lds/CGI.pm-3.17/CGI.pm)

This is the standard Perl module that provides common functions for building web scripts.

Once you have installed the modules, copy the following code to a file named*map-news.cgi* :

```
#!/usr/local/bin/perl
# map-news.cgi
# Queries Google News for a given subject and
# maps the news sources on a Google Map. Click
# a point on the map to read the article
# summary.
#
# Grab a Google Maps API key here:
#
# http://www.google.com/apis/maps/
```

```
use strict;
use LWP;
use XML::Simple;
use HTML::GoogleMaps;
use URI::Escape;
use Geography::USStates;
use CGI qw/:standard/;

my $google_maps_key = "insert your Google Maps key";

#Initialize Error Handling
use CGI::Carp qw( fatalsToBrowser );
BEGIN {
    sub carp_error {
        my $error_message = shift;
        print "<pre>$error_message</pre>";
    }
    CGI::Carp::set_message( \\&carp_error );
}

# Start the page
print "Content-Type: text/html\\n\\n";

# Grab the incoming query and format for use in URL
my $query = param('q');
my $query_esc = uri_escape($query);
my $news = "<h2>News Stories</h2>\\n";

# Start the Google Map
my $map = HTML::GoogleMaps->new(key => $google_maps_key, height => 525, width => 975);
$map->zoom(15);
$map->controls("large_map_control", "map_type_control");

# Set the client for fetching pages
my $browser = LWP::UserAgent->new;
$browser->agent("Mozilla/5.01 (windows; U; NT4.0; en-us) Gecko/25250101");

# Fetch the Google RSS Feed for the query
my $feed = "http://news.google.com/news?hl=en&ned=us&q=$query_esc&ie=UTF-8&scoring=d&nu
my $google_response = $browser->get($feed);

if (!$google_response->is_success(  )) {
    die "News feed not found! $google_response->status_line(  )";
}

# Parse the Google News RSS
my $xmlsimple = XML::Simple->new(  );
my $google_rss = $xmlsimple->XMLin($google_response->content);

# Pick through the items, grabbing the byline
foreach my $item (@{$google_rss->{channel}->{item}}) {
```

```
my $title = $item->{title};
my $link = $item->{link};
my $desc = $item->{description};
my ($byline, $lonlat);
while ($desc =~ m!<a href="([^"]+)">(.+?)</a><br>(.+?)<nobr>(.+?)</nobr>.*?<br>(.+?)
    $byline = $3;
    $byline =~ s!<[^>]+>!!gis;
    $byline =~ s! ! !gis;
    $byline =~ s!- !!gis;
}
my $article = "<a href='$link'>$title</a>";
my @byline = split(/,/, $byline);

# Grab the location from the byline
my $location = trimwhitespace(@byline->[1]);
$location =~ s!Oregon!OR!gis;
$location =~ s!UK!United Kingdom!gis;

    # If the location is a state abbreviation, geocode
    if ($location =~ m!^\\S{2}$!gis) {
        my $state = getState($location);
        if ($state) {
            $lonlat = getStatelonlat($state);
        }
    # If the location is a country name, geocode
    } else {
        $lonlat = getWorldlonlat($location);
    }
    $desc =~ s!'!\\\\'!mgis;
    $desc =~ s!"!\\\\"!mgis;

    # Add the point to the Google Map
    if ($lonlat) {
        $map->add_marker(point => $lonlat, html => $desc);
    }

    # Print out the item to the page
    $news .= $desc;
}

# Render the entire map, and print out the page
my ($head, $map, $body) = $map->render;
print "<html><head><title>Google News, Mapped</title>$head</head><body>\\n";
print "<h2>Google News about $query, Mapped</h2>";
print "$map $body $news";
print "</body></html>";


# Supporting Functions --------------------------------

# Find the longitude and latitude of a country
sub getWorldlonlat($) {
```

```
    my $loc = shift;
    if ($loc ne "") {
        my $esc_location = uri_escape($loc);
        my $url = "http://maps.google.com/maps?q=$esc_location&output=js";
        my $response = $browser->get($url)->content;
        # Note if the location has a related longitude/latitude
        if ($response =~ m!center: {lat: (.*?),lng: (.*?)}!gis) {
           my $lat = $1;
           my $lon = $2;
           my $lonlat = [$lon,$lat];
           return $lonlat;
        # Otherwise, warn the user that the coordinates can't be found
        } else {
            warn "\\nNo coordinates found for location $loc";
        }
    } else {
        return 0;
    }
}

# Find the longitude and latitude of a US state
sub getStatelonlat($) {
    my $loc = shift;
    if ($loc ne "") {
        my $esc_location = uri_escape($loc);
        my $url = "http://ws.geonames.org/search?q=$loc&fclass=A&maxRows=10&country=us";
        my $response = $browser->get($url)->content;
        # Note if the location has a related longitude/latitude
        if ($response =~ m!<geoname>.*?<lat>(.*?)</lat>.*?<lng>(.*?)</lng>.*?</geoname>!
           my $lat = $1;
           my $lon = $2;
           my $lonlat = [$lon,$lat];
           return $lonlat;
           last;
        # Otherwise, warn the user that the coordinates can't be found
        } else {
            warn "\\nNo coordinates found for location $loc";
        }
    } else {
        return 0;
    }
}

# Clean up text
sub trimwhitespace($) {
    my $string = shift;
    $string =~ s/^\\s+//;
    $string =~ s/\\s+$//;
    return $string;
}
```

Even with the help of `HTML::GoogleMaps` , there's still quite a bit of code required to generate a Google M parsing a Google News RSS feed and geocoding place names, and that means over 150 lines of code are stories and generate the map.

Running the Hack

Upload *map-news.cgi* to your web server and run it by passing in the news subject you want to map. Th query string variable `q` , like so:

```
http://example.com/map-news.cgi?q=insert news topic
```

To map the distribution of stories about a worldwide problem such as Avian Flu, for example, call the sc

```
http://example.com/map-news.cgi?q=Avian%20Flu
```

Note that spaces in a URL are escaped as `%20` , because a space isn't a valid character in a URL.

The script takes some time to gather news stories from Google, geocode the source of the story, and pl Map. Once assembled, you should see a map like the one in Figure 3-10.

Figure 3-10. Google News about Avian Flu on a Google Map

Click any point on the map to see a summary of the story. Click the story headline to leave the map and the web site where it was originally published. The script also prints out every story excerpt found just b you can browse through stories there as well.

Keep in mind that this script maps the location of news outlets, which is not necessarily of the *subject* of the article. For example, the *Monterey County Herald* in California migh story about something happening in China. That story would have a pointer in Californi It's also important to note that Google News U.S. skews toward U.S. sources, so you'll more stories mapped within the U.S.

Not every news topic is of worldwide importance with references across the globe. But mapping news to sense of where a certain story makes the news and remind you that Google News is gathering stories fr the world.

PREV

Hack 39. Track Your Favorite Sites

Use Google Reader or Google Homepage to stay up to date with your favorite web sites that h

Syndication has changed how people consume web sites by offering headlines and articles in a machine- people can read content from news web sites or independent blogs at a completely independent web sit news content for efficient reading. If you like to read the*New York Times* (http://www.nytimes.com) an BoingBoing (http://www.boingboing.net), you're in luck, because they both offer news feeds. Instead of look for new articles or posts, you can simply subscribe to them with a program called a*newsreader* and sites in this third, independent location.

Google provides two tools for consuming news feeds.Google Personalized Homepage (http://www.google headlines from around the Web in one space, andGoogle Reader (http://www.google.com/reader) is sp consuming feeds and reading their entire contents.

RSS stands for *Really Simple Syndication* or *Rich Site Summary* , depending on who you ask, and Atom i important is that RSS and Atom are both standard XML formats for sharing headlines and news summar web page is formatted for display in a web browser, news feeds are formatted for display in newsreader first key to consuming feeds at Google is finding feed URLs.

Finding Feeds

Keep in mind that not every news source or blog out there offers a news feed. And those that do don't a find. Part of the skill of adding content to Google is being able to find the feeds you care about. The key *feed URL* , so you can copy and paste the URL into a form at Google. Like an address for a house, a feed where to find updated information. Here are some tips for feed URLspotting.

Go to the source

The first place to look for feed URLs is at your favorite web sites. Most sites that offer an RSS feed have letters that says *XML* , *RSS* , or *Atom* . Figure 3-11 shows a number of variations you might see on the f

Figure 3-11. Variations on the white-on-orange XML the

Nine times out of 10, this image links to the site's feed URL.

Remember that RSS and Atom are XML formats, which is why the terms are used inter
images.

To copy the feed URL, right-click the icon and choose Copy Link Location (or Copy Shortcut in Internet E
point, the feed URL is available at your virtual clipboard, ready to paste into Google.

The square icon with the symbol is an emerging standard for linking to news feeds. If y
want to use the new symbol, visit the home of the icon (http://www.feedicons.com) to
use on your site in a number of different graphics formats.

Look for autodiscovery

Even sites that don't include an orange and white XML icon might leave clues about the RSS feed URL in
the problem of finding feeds, a standard called *RSS autodiscovery* has emerged. Sites that want to make
feed URL can include a special HTML tag in the source of their pages to let applications such as web brov

Once browsers are "aware" of autodiscovery and are looking for the autodiscovery tag, they can let user
an RSS or Atom feed URL in a web page. Firefox lets users know by displaying an orange icon at the far
shown in Figure 3-12 .

Figure 3-12. Firefox with the orange feed indicator in the add

Even though you might not be able to spot the O'Reilly Network XML icon or a link to its RSS feed on the
orange RSS feed indicator in Firefox, you can use Firefox's View Source feature to find the autodiscovery
URL.

To view the source of any web page, choose View Page Source from the browser's top menu. Finding
tag is always located toward the top of the HTML page, between the opening and closing `<head>` tags. For
Network page in Figure 3-12 has the following autodiscovery tag in its HTML source:

```
<link rel="alternate" type="application/atom+xml" title="Weblogs" href="http://www.orei
```

Note the URL contained in the `HRef` element. This is the site's Atom feed URL, ready for copying and past
newsreader.

You can also take Firefox's autodiscovery feature a little further to speed up the proces
rather not dig through source HTML to find feed URLs.

Look for Add to Google

Some site authors go a step further and let you add their feeds to Google with one click. If you see the i simply click the button and you're offered the choice to add a site's feed to Google Homepage or Google

Figure 3-13. Add to Google button

At the time of this writing, the Add to Google button is brand-new, and not many sites use it. But if you the fastest way to add a feed to your preferred Google newsreader.

If you maintain a feed and want to offer the Add to Google button on your web site, vis Information for Publishers (http://www.google.com/webmasters/add.html) to pick up s copy and paste into your site.

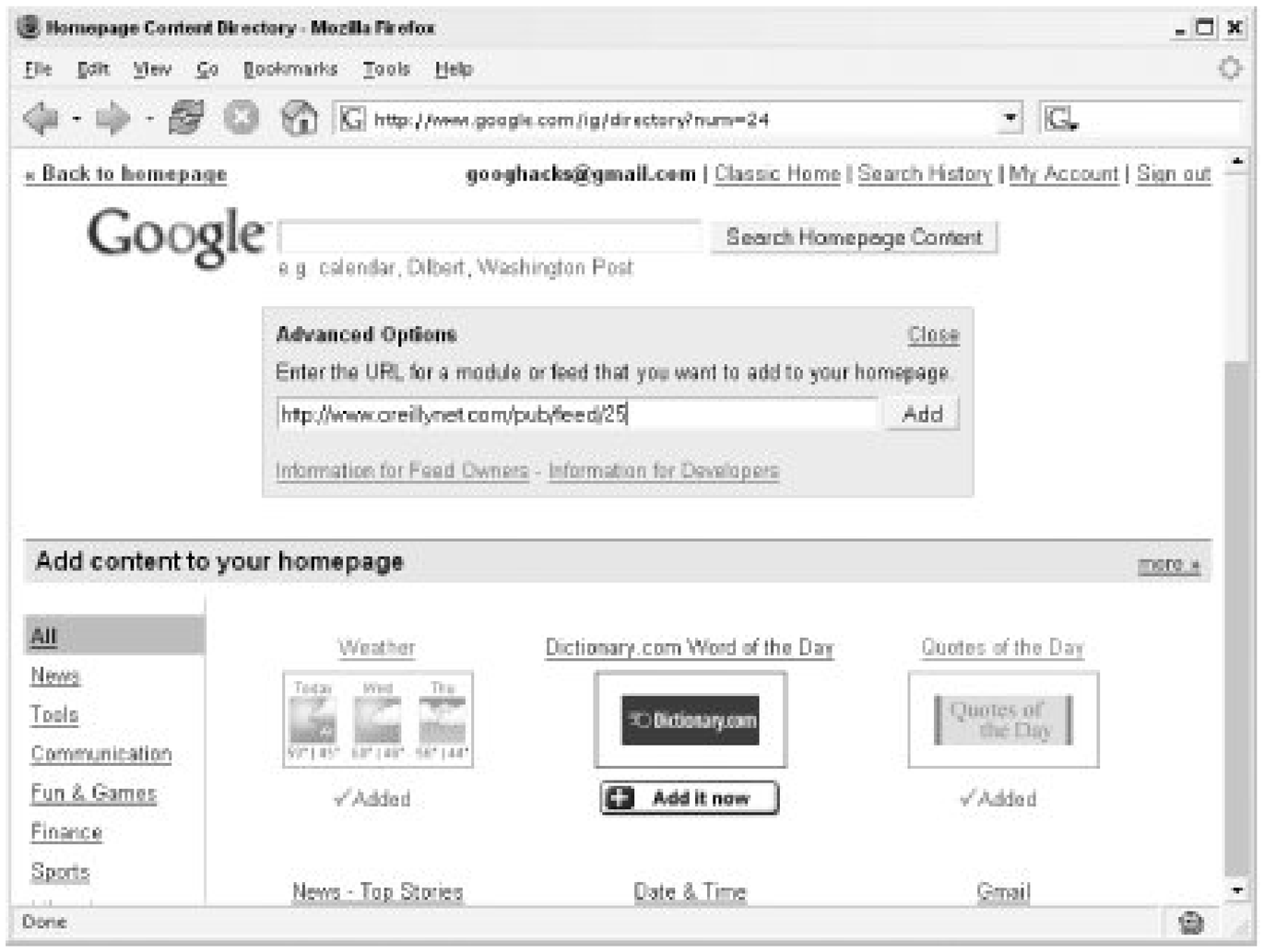
Adding to Google Homepage

Once you have copied a feed URL, visit Google Homepage (http://www.google.com/ig) and click "Add cc

If you haven't already started to customize Google Homepage, you might need to click before you can add feeds.

From there, click the Advanced Options link next to the Search Homepage Content button, and you shou Figure 3-14 .

Figure 3-14. Adding a feed to Google Homepage



Paste the URL into the form and click Add. The new feed appears in the upper left of your Google Homep

Figure 3-15. The O'Reilly feed on Google Homepage

As you can see, Google Homepage offers only the latest headlines from the site, and you need to click e
the site to read the story. If you want to do a bit more reading at Google, you can turn to the appropria

Adding to Google Reader

Google Reader is designed for serious feed reading, and adding a feed is quite simple. Browse toGoogle I (http://reader.google.com), click "Edit subscriptions" toward the top of the page, and then click "Add a and click Preview. From there, you see all of the items in the feed and can decide whether to subscribe. items in the main Google Reader window shown in Figure 3-16.

Figure 3-16. The O'Reilly feed in Google Reader

Not only can you find the latest headlines, but you can read entire articles from the site within the Goog

No matter which Google newsreader you prefer, it's easy to add outside sources to either, giving you a v favorite content online when it's updated, without visiting hundreds of sites each day.

Hack 40. Scrape Google Groups

Pull results from Google Groups searches in the form of a comma-delimited file.

It's easy to look at the Internet and say that it's a group of web pages or computers or networks. But lo they can consider their philosophies, make contact, or buy their products and services.

Nowhere is the Internet-as-conversation idea more prevalent than in Usenet newsgroups. Google Group

Because Google Groups is not searchable by the current version of the Google API, you can't build an au

The first thing you need to do is run a Google Groups Search. See the "Google Groups" section earlier in

It's best to sort the pages you're going to scrape by date; that way, if you scrape more pages later, it's

```
perl group:google.public.web-apis
```

On the right side of the results page is an option to sort either by relevance or date; click the "Sort by d

Save this page to your hard drive, naming it something memorable, such as*groups.html* .

Scraping is brittle at best. A single change in the HTML code underlying Google

At the time of this writing, a typical Google Groups Search result looks like this:

```
<table cellspacing="0" cellpadding="2" border="0"><tbody><tr>
<td style="width: 38em">
  <font size="+0"><a href="/group/google.public.web-apis/browse_thread/thread/1a3c3a03c
  <br>I've tried to adapt the bit of <b>perl</b> code in the readme, but that didn't wor
description 'file:GoogleSearch.wsdl' can't be loaded: 404 File <b>...</b>
 
  <br><a href="/group/google.public.web-apis?lnk=sg" class="gl"><b>google.public.web-api
 - 1 message - 1 author
</td>
</tr>
</tbody></table><br>
```

As with the HTML example given for Google News [Hack #36], this might be utter gobbledygook for son

The Code

Save the following code as*groups2csv.pl* :

```
#!/usr/bin/perl
# groups2csv.pl
# Google Groups results exported to CSV suitable for import into Excel.
# Usage: perl groups2csv.pl < groups.html > groups.csv

# The CSV Header.
print qq{"title","url","group","date","author"\\n};

# Rake in those results.
my($results) = (join '', <>);

# Perform a regular expression match to glean individual results.
while ( $results =~ m! <font size="\\+0"><a.*?href="(.*?)">(.*?)</a>.*? (.*?) 
    my($url, $title, $snippet, $groupURL, $group, $date, $author) =
        ($1||'',$2||'',$3||'',$4||'',$5||'',$6||'',$7||'');
    $title =~ s!"!""!g; # double escape " marks
    $title =~ s!<.+?>!!g; # drop all HTML tags
    $group =~ s!<.+?>!!g; # drop all HTML tags
    print qq{"$title","$url","$group","$date","$author"\\n};
}
```

Running the Hack

Run the script from the command line ["How to Run the Hacks" in the Preface], specifying the Google G as your output:

```
$ perl groups2csv.pl <  groups.html >  groups.csv
```

Leaving off the `>` and CSV filename sends the results to the screen for your perusal.

Using `>>` before the CSV filename appends the current set of results to the CSV file, creating it if it doesr

```
$ perl groups2csv.pl  <  results_1.html >  results.csv

$ perl groups2csv.pl  <  results_2.html >>  results.csv
```

Scraping the results of a search for `perl group:google.public.web-apis` for anything mentioning the Perl

```
$ perl groups2csv.pl < groups.html
"title","url","group","date","author"
"Query syntax (newbie)","http://groups.google.com/group/google.public.web-apis/browse_f
...
"Perl SOAP::Lite error: '400 Error unmarshalling envelope'","http://groups.google.com/g
...
```

PREV

Hack 41. Seek Out Blog Commentary

Turn to Google Blog Search, or build your own queries to find only recent commentary appearing in blogs.

There was a time when, if you needed to find current commentary, you couldn't turn to a full-text search engine such as Google. You searched Usenet, combed mailing lists, or searched through current news sites such as CNN.com and hoped for the best.

Today, millions of people offer their own running commentary and associated links on blogs that are often updated dailyand, indeed, even more often in many cases. Google indexes many of these sites on an accelerated schedule. As blogs have grown in popularity, the number of ways to find recent commentary across blogs has grown as well. If you're looking for casual conversation about a subject rather than official documentation, blog commentary puts you in touch with the person on the street.

Google Blog Search

The first place to look for blog commentary is Google's blog-specific search engine. You can find it in one of two locations: the standard Google site (http://blogsearch.google.com) or as part of the Blogger site (http://search.blogger.com). At the time of this writing, the two versions are a bit different, so you need to pay attention to which Blog Search you're using.

> Keep in mind that, at the time of this writing, the Google Blog Search is currently in beta testing, which means its features are far from finalized. Google will probably continue to tweak and tune the service, so think of this description as a snapshot of the Google Blog Search early days.

Both searches work exactly the same as a Google Web Search: type your query into a form, click Search, and you get a page with several blog posts that contain that query. If you're using the Blogger search (http://search.blogger.com), you have the option to limit your query to a single blog in the results (magnifying-glass icon) or to view all posts from a specific blog (page icon), as shown in Figure 3-18.

Figure 3-18. Blog Search results options

The Google Blog Search also has an Advanced Search form (http://search.blogger.com/advanced_blog_search) that lets you refine your queries even further by limiting the dates or by searching by blog title, author, or URL. Each page of results also sports an RSS or Atom feed, so you can track the query over time in your favorite newsreader.

If you enjoy looking for personal commentary on a particular topic, you might also want to use Google to find personal photos [Hack #14].

Plugging any current topic into the Blog Search usually yields hundreds or thousands of comments, but keep in mind that the Blog Search doesn't track every single available blog. It tracks only those blogs that also publish their content in an RSS feed. While most blogs include a feed these days, thanks to automated blog tools such as Blogger, you might also want to use the standard Google Web Search to cover all your blogging bases.

Google Web Search

When blogs first appeared on the Internet, they were generally updated manually or by using homemade programs. Thus, there were no standard words you could add to a search engine to find them. Now, however, many blogs are created using either specialized software packages, such as Movable Type (http://www.movabletype.org) or WordPress (http://www.wordpress.org), or as web services, such as Google's own Blogger (http://www.blogger.com/). These programs and services are more easily found online with some clever use of special syntaxes or magic words.

For hosted blogs, the `site:` syntax makes things easy. Blogger blogs hosted at blog*spot (http://www.blogspot.com) can be found using `site:blogspot.com`. Even though WordPress is a software program that can post its blogs to any web server, you can find hundreds of WordPress blogs at the hosted server (http://www.wordpress.com) using `site:wordpress.com`.

Finding blogs powered by blog software and hosted elsewhere is more problematic; Movable Type blogs, for example, can be found all over the Internet across hundreds of different domains. However, most of them sport a "powered by Movable Type" link of some sort; searching for the phrase `"powered by movable type"` can, therefore, find many of them.

Finding "magic words."

It comes down to *magic words*shout-outs, if you will, to the software or hosting sitesthat are typically found on blog pages. The following is a list of some of these packages and services and the magic words used to find them in Google:

Blogger

`"powered by blogger"` or `site:blogspot.com`

Blosxom

`"powered by blosxom"`

LiveJournal (a service)

`site:livejournal.com`

Movable Type

`"powered by movable type"`

Radio Userland

`intitle: "radio weblog"` or `site:radio.weblogs.com`

TypePad

`site:typepad.com` or `"powered by typepad"`

WordPress

`"powered by wordpress"`

Xanga

`site:xanga.com inurl:user`

Yahoo! 360

`site:blog.360.yahoo.com`

Using these "magic words."

Because you can't have more than 32 words in a Google query, there's no way to build a query that includes every conceivable blog's magic words. It's best to experiment with the various words and see which blogs have the materials you're interested in.

First of all, realize that blogs are usually informal commentary and that you have to keep an eye out for misspelled words, names, etc. Generally, it's better to search by event than by name, if possible. For example, if you're looking for commentary on a potential baseball strike, the phrase`"baseball strike"` would be a better search, initially, than a search for the name of the Commissioner of Major League Baseball: `"Bud Selig"`.

You can also try to search for a word or phrase relevant to the event. For a baseball strike, you can try searching for `"baseball strike" "red sox"` (or `"baseball strike" bosox`). If you're searching for

information on a wildfire and wondering if anyone had been arrested for arson, try`wildfire arrested`; if that doesn't work, try `wildfire arrested arson`.

Why not search for `arson` to begin with? Because it's not certain that a blog commentator would use the word "arson." Instead, he might just refer to someone being arrested for setting the fire. "Arrested" in this case is a more reliable word than "arson."

On the Web, everyone can be a publisher, and whether you're looking for rants, advice, conversation reviews, or idle chitchat, you're bound to find it on blogs if you know where to look.

PREV

Hack 42. Glean Blog-Free Google Results

With so many blogs being indexed by Google, you might worry about too much emphasis on the hot topic of the moment. In this hack, we'll show you how to remove the blog factor from your Google results.

Weblogs (or blogs)those frequently updated, link-heavy personal pagesare quite the fashionable thing these days. There are probably over 20 million active blogs across the Internet, covering almost every possible subject and interest. For humans, they're good reading, but for search engines, they're heavenly bundles of fresh content and links galore.

Some people think that the search engine's delight in blogs slants search results by placing too much emphasis on too small a group of recent rather than evergreen content. At the time of this writing, for example, I am the ninth most important Ben on the Internet, according to Google. This rank comes solely from my blog's popularity.

This hack searches Google, discarding any results that come from blogs. It uses the Google Web Services API (http://api.google.com) and the API of Technorati (http://technorati.com/developers/apikey.html), a blog-tracking site that indexes millions of blogs. Both APIs require keys, available from the URLs mentioned.

Finally, you need a simple HTML page with a form that passes a text query to the parameter `q` (the query that runs on Google)something like this:

```
<form action="googletech.cgi" method="POST">
Your query: <input type="text" name="q">
<input type="submit" name="Search!" value="Search!">
</form>
```

Save the form as *googletech.html*.

The Code

You'll need the `XML::Simple` and `SOAP::Lite` Perl modules to run this hack.

Save the following code ["How to Run the Hacks" in the Preface] to a file called *googletech.cgi*, replacing *`insert google key`* and *`insert technorati key`* with your own respective API keys:

```
#!/usr/bin/perl -w
# googletech.cgi
# Getting Google results
# without getting weblog results.
```

```
use strict;
use SOAP::Lite;
use XML::Simple;
use CGI qw(:standard);
use HTML::Entities (  );
use LWP::Simple qw(!head);

my $technorati_key = "insert technorati key";
my $google_key = "insert google key";

# Set up the query term
# from the CGI input.
my $query = param("q");

#Initialize Error Handling
use CGI::Carp qw( fatalsToBrowser );
BEGIN {
    sub carp_error {
        my $error_message = shift;
        print "<pre>$error_message</pre>";
    }
    CGI::Carp::set_message( \\&carp_error );
}

# Initialize the SOAP interface and run the Google search.
my $google_wsdl = "http://api.google.com/GoogleSearch.wsdl";
my $google_search = SOAP::Lite->service($google_wsdl);

# Query Google.
my $results = $google_search ->
    doGoogleSearch(
      $google_key, $query, 0, 10, "false", "",  "false",
      "", "latin1", "latin1"
    );

# Start returning the results page;
# do this now to prevent timeouts.
my $cgi = new CGI;

print $cgi->header(  );
print $cgi->start_html(-title=>'Blog Free Google Results');
print $cgi->h1('Blog Free Results for '. "$query");
print $cgi->start_ul(  );

# Go through each of the results.
foreach my $result (@{$results->{resultElements}}) {

    # Encode the result URL
    my $url = HTML::Entities::encode($result->{URL});

    # Request the Technorati information for each result.
    my $technorati_result = get("http://api.technorati.com/bloginfo?".
```

```
                                    "url=$url&key=$technorati_key");

    # Parse this information.
    my $parser = XML::Simple->new(suppressempty => undef);
    my $parsed_feed = $parser->XMLin($technorati_result);

    # If Technorati considers this site to be a weblog,
    # go onto the next result. If not, display it, and then go on.
    if ($parsed_feed->{document}{result}{weblog}{name}) { next; }
    else {
        print $cgi->p('<a href="'.$url.'">'.$result->{title}.'</a>',
                      '<br />'.$result->{snippet},
                      '<br />'.i($result->{URL}));
    }
}
print $cgi -> end_ul(  );
print $cgi->end_html;
```

Let's step through the meaningful bits of this code. First, pull in the query from Google. Notice the `10` in the `doGoogleSearch`; this is the number of search results requested from Google. If you find you're searching for terms that are extremely popular in the blogging world and you're not getting any results at all, try editing the script to fetch more than 10 results [Hack #93]. That might be the only way to find nonblog results for some terms.

Since we're about to make a web services call for every one of the returned results, which might take a while, we should start to return the results page now; this helps prevent connection timeouts. To do this, we spit out a header using the *CGI* module, and then jump into our loop.

We then get to the final part of our code: actually looping through the search results returned by Google and passing the HTML-encoded URL to the Technorati API as a `get` request. Technorati then returns its results as an XML document.

Be careful that you do not run out of Technorati requests. At the time of this writing, Technorati is offering 500 free requests a day, which, with this script, is around 50 searches. If you make this script available to your web site audience, you will soon run out of Technorati requests. One possible workaround is forcing the user to enter her own Technorati key. You can get the user's key from the same form that accepts the query. See "Hacking the Hack" for a way to do this.

You can keep up with changes to the Technorati API at the Technorati Developer's Wiki (http://developers.technorati.com/wiki/TechnoratiApi).

Parsing this result is a matter of passing it through `XML::Simple`. Since Technorati returns only an XML construct containing `name` when the site is thought to be a blog, we can use the presence of this construct as a marker.

Note that we've set the parser to treat empty XML elements as undefined with the line `XML::Simple->new(suppressempty => undef)`. If the program sees a defined blog name for a particular URL, it skips to the next result. If it doesn't, Technorati does not consider the site to be a blog, and we display a

link to it, along with the title and snippet (when available) returned by Google.

Running the Hack

To run the hack, point your browser at the form *googletech.html*.

Hacking the Hack

As mentioned previously, this script can burn through your Technorati allowances rather quickly under heavy use. The simplest way to solve this is to force the end user to supply his own Technorati key. First, add a new input to your HTML form for the user's key:

```
Your query: <input type="text" name="key">
```

Then, suck in the user's key as a replacement to your own:

```
# Set up the query term
# from the CGI input.
my $query = param("q");
$technoratikey = param("key");
```

And if you want to turn this hack into a "blog only" search, simply edit the line that checks for a defined blog name, like so:

```
if (!$parsed_feed->{document}{result}{weblog}{name})
```

The exclamation point tells Perl to test for a defined rather than undefined value and prints out any result confirmed to be from a blog. But then again, you could just pop over to the Google Blog Search (http://search.blogger.com) and save your Google and Technorati daily query allotment.

Ben Hammersley

Hack 43. Find Blog Commentary for Any URL with a Single C

A bit of JavaScript and the Google Blog Search can give you instant access to commentary ab

If you've already played around with the Google Blog Search [Hack #41], you're well aware of the sheer problem, though, is that the blogosphere is complete chaos, and it's hard to connect with commentary t

This is where the Google Blog Search can come in handy, limiting your blog search results to a specific t when you're looking for posts that reference your web site, but you can also use this to find commentar

For example, say you happen across an article predicting the end of the Internet, such as the one shown

This article makes some interesting points about a topic near and dear to every blogger's heart, so it ma head to Google Blog Search, and use the `link:` syntax to find posts that link to the article by typing a qu

```
link:http://www.thenation.com/doc/20060213/chester
```

At the time of this writing, there are 183 posts that link to the article. While you might not have time to

This process is a bit tedious, so this hack shows how to speed it up so you can find blog commentary ab

The Code

A *bookmarklet* is a bit of JavaScript code stored in a web browser bookmark. Bookmarklets give you a w
about the current page. With bookmarklets, you're in control of the script, because it runs when you clic

In order to implement this hack, the only thing you need is a browser that has bookmarks and understa

Here's a look at some nicely formatted JavaScript that gets the current URL for the page you're looking

Keep in mind that this code is nicely formatted to show you how it operates; the functioning bookmarkle

```
// Dissected JavaScript bookmarklet for Google weblog commentary

// Set d to the document object as a shortcut
var d = document;

// Build the URL that will link to Blog Search results
var url = 'http://search.blogger.com/?';
url += 'as_lq=';

// include the URL of the current page
url += '.url='+escape(d.location.href)+'&';

url += 'as_drrb=q&';
url += 'lang=all&';
url += 'scoring=d';

// open a new window to add the bookmark and show the results
window.open(url,
            '_blank',
            'width=640,height=440,status=yes,resizable=yes,scrollbars=yes');
```

Unfortunately, a bookmarklet is no place for readable code with comments and line breaks. Instead, the

```
javascript:d=document;t=d.selection?d.selection.createRange(   ).text:d.getSelection(
```

As you can see, it looks similar to the preceding code, but with some important changes. The`javascript`
it stops the expression it surrounds from returning a value. In this case, we don't really care what value

Running the Hack

The installation process for the bookmarklet is unique to the browser you want to use it with. If you kno
valid protocol, but you can ignore that message. You'll also want to give your bookmarklet a snappy, sh

Once the bookmark is in place, browse to any page and click away! Once you click, a new window opens

Not every URL has blog commentary, especially if an article was published within the last few hours. But
commentary that's even more relevant to you than the original source you were reading.

Hack 44. Track Topics on Blogs over Time

Visualize topics discussed on blogs by counting the total number of mentions of a specific phr

Reading a blog is a bit like reading a conversation that someone has typed out. Blogs are informal, off tl dialogues are in text form means they can be indexed and studied like any other text.

Perhaps that's one reason Google put together the Google Blog Search (http://blogsearch.google.comor able to search blogs on their own. Because the vast majority of blogs are personal opinions and commer the chattering classes to their keyboards is a new product announcement from Apple, and a look at the

The key to being able to track a keyword in blogs over time is the ability to isolate posts by day. Luckily (http://blogsearch.google.com/blogsearch/advanced_blog_search) allows you to limit searches by time. retrieves them by specifying that date as the start and end date. If no blogs mentioned that phrase on t

Once you isolate posts to a particular day, you can find out how many posts contain the term you're inte September 7, 2005. Figure 3-21 shows the Google Blog Search result, along with the estimated number

Figure 3-21. Total posts that mentic

By contrast, only 13 stories mentioned *iPod Nano* on September 6, 2005. You can probably connect the

This hack shows how to automate the process of tracking keywords across blog posts, allowing you to d

The Code

Google's Web Services API doesn't include access to its Blog Search, so this hack uses the Google Blog S
Search results page. Even advanced search queries include an RSS feed of results, and within that feed

```
<description>Google Blog Search Results: &lt;b&gt;748&lt;/b&gt; results for &lt;b&gt;iPo
```

Note that the `<b>` tags are escaped as `<b>` to make the XML valid. It looks like a confusing mess, b

Another key component of the Blog Search RSS feeds is that they have a predictable URL, so the Advan
construct your own URLs. An advanced Blog Search feed URL looks like this:

```
http://blogsearch.google.com/blogsearch_feeds?as_q=&as_epq=iPod+Nano&as_drrb=b&as_mind=1
=rss
```

As you can see, the `as_mind` , `asminm` , and `as_miny` variables hold the start date, and `as_maxd` , `as_maxm` ,

You'll need a couple Perl modules for this hack, including `LWP::Simple` to fetch the feed and `Date::Manip` t

```
#!/usr/bin/perl
# track_blogs.pl
# Builds a Google Search URL for every day
# between the specified start and end dates, returning
# the date and estimated total results as a CSV list.
# usage: track_news.pl query="{query}" start={date} end={date}
# where dates are of the format: yyyy-mm-dd, e.g. 2006-02-30

use strict;
use Date::Manip;
use LWP::Simple qw(!head);
use CGI qw/:standard/;

# Get the query
my $query = param('query');

# Regular Expression to check date validity
my $date_regex = '(\\d{4})-(\\d{1,2})-(\\d{1,2})';

# Make sure all arguments are passed correctly
( param('query') and param('start') =~ /^(?:$date_regex)?$/
  and param('end') =~ /^(?:$date_regex)?$/ ) or
  die qq{usage: track_news.pl query="{query}" start={date} end={date}\\n};

# Set timezone, parse incoming dates
Date_Init("TZ=PST");
my $start_date = ParseDate(param('start'));
```

```
my $end_date = ParseDate(param('end'));

# Print the CSV column titles
print qq{"date","count"\\n};

# Loop through the dates
while ($start_date <= $end_date) {
    my $month = int UnixDate($start_date, "%m");
    my $day = int UnixDate($start_date, "%d");
    my $year = int UnixDate($start_date, "%YYYY");
    my $date_f = UnixDate($start_date,"%Y-%m-%d");
    my $total;

    # Construct a Google Blogsearch URL
     my $blog_url = "http://blogsearch.google.com/blogsearch_feeds?";
        $blog_url .= "as_q=";
        $blog_url .= "&as_epq=$query";
        $blog_url .= "&as_drrb=b";
        $blog_url .= "&as_mind=$day";
        $blog_url .= "&as_minm=$month";
        $blog_url .= "&as_miny=$year";
        $blog_url .= "&as_maxd=$day";
        $blog_url .= "&as_maxm=$month";
        $blog_url .= "&as_maxy=$year";
        $blog_url .= "&num=10";
        $blog_url .= "&output=rss";

    # Make the request
    my $blogs_response = get($blog_url);

    # Find the number of results
    my $regex = "Google Blog Search Results: &lt;b&gt;(.*?)&lt;/b&gt; results";
    if ($blogs_response =~ m!$regex!gi) {
        $total = $1;
    } else {
        $total = 0;
    }

    # Print out results
    print
      '"',
      $date_f,
      qq{","$total"\\n};

    # Add a day, and continue the loop
    $start_date = DateCalc($start_date, " + 1 day");
}
```

Running the Hack

Run the script from a command line, specifying the query term and dates. Here's the query for *iPod Nanc*

```
track_blogs.pl query="iPod Nano" start=2005-08-25 end=2005-09-25
```

If you want to pipe the script output to a text file, simply call it like so:

```
track_blogs.pl query="iPod Nano" start=2005-08-25 end=2005-09-25 > nano.csv
```

The truncated results look like this:

```
...
"2005-08-30","0"
"2005-08-31","0"
"2005-09-01","0"
"2005-09-02","2"
"2005-09-03","0"
"2005-09-04","1"
"2005-09-05","10"
"2005-09-06","13"
"2005-09-07","748"
"2005-09-08","583"
"2005-09-09","270"
...
```

Just glancing at this list, you can see there were no mentions of iPod Nano, and then, suddenly, the phr

Working with the Results

With a short list, it's easy to see where the spikes in media mentions are. But with longer lists, it might double-click it to open it with Excel. The chart wizard can give you a quick overview, such as the one for

Figure 3-22. Excel graph showi

You can see the blip when the Nano was released, and then a steady decline.

Not every phrase you try will show such a distinct pattern, but looking at posts across time can help you

Hack 45. Blog from Your Desktop

Desktop blogging clients use the power of your local computer to add features and automate common tasks.

Writing text in a browser window can be a frustrating experience, especially if you're writing something longer than an average email. When you write a post on your Blogger blog, you normally browse to the site and type into a form, editing the text and HTML by hand. If you've ever experienced a browser crash, or a dropped Internet connection, then you know that writing text into a browser can result in lost work. And if you compare the editing form at Blogger with a traditional word processor such as Word, you find a big difference between the available features.

You can enable a visual editor (sometimes called a*WYSIWYG* editor, which stands for "what you see is what you get") for the web form at Blogger. Log into Blogger, choose a blog, click the Settings tab, scroll to Global Settings, set Show Compose Mode to Yes, and click Save Settings. Then, while writing a post, choose the Compose tab at the upper-right corner of the form. Then, as you bold words, you'll see them bold in the editor, as you would in a traditional word processor.

A big reason for the difference in features is that Word can take advantage of the processing power of your local computer, while the browser typically needs to stay lightweight to transfer pages quickly. But that doesn't mean you need to stay tied to the browser. Blogger offers an API for working with its blogs, and a number of developers have put together their own interfaces for publishing with Blogger.

These applications function more like traditional word-processing applications and offer some extended features that Blogger doesn't offer. To get started, you simply need time to experiment, along with a Blogger username and password. This hack presents three desktop blogging clients that might change the way you post to your blog.

w.bloggar

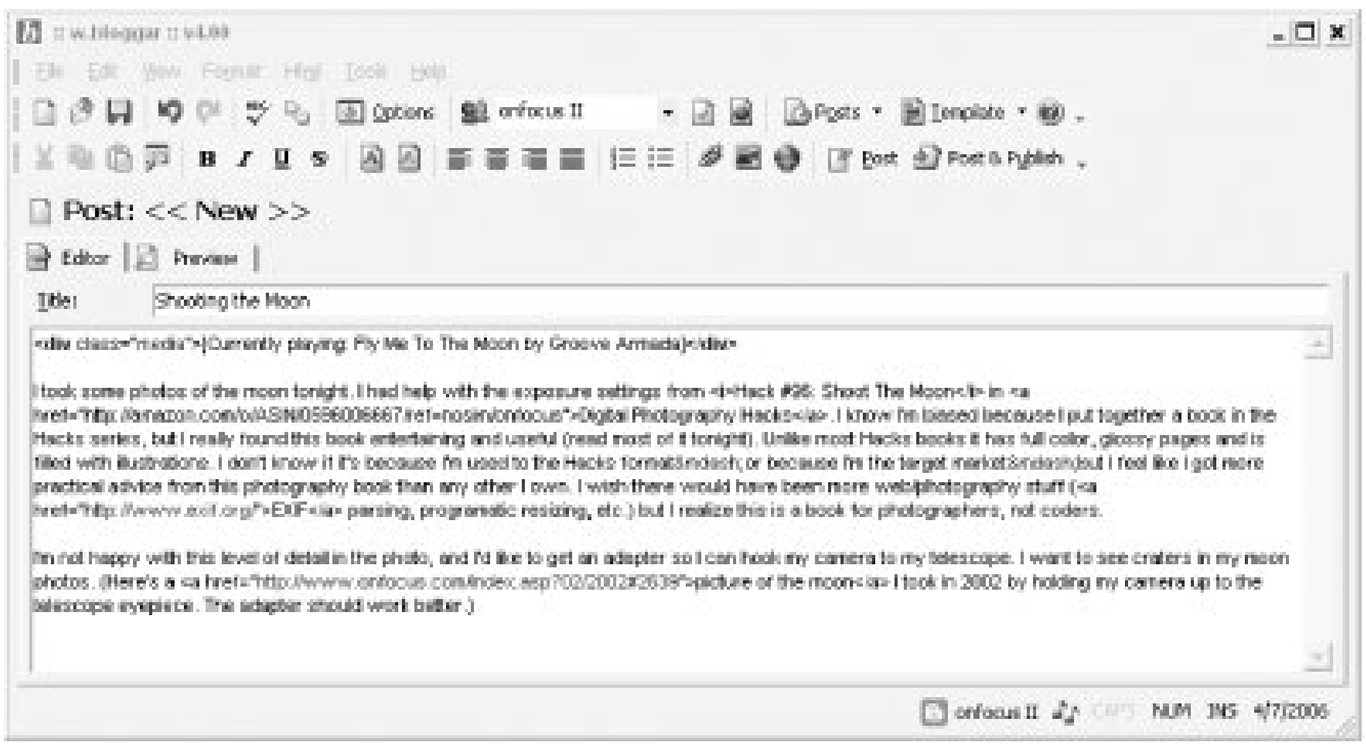
w.bloggar (http://wbloggar.com) is a free client for Windows that can post to many blog systems, including Blogger. The program is basically an HTML editor that offers point-and-click access to common HTML tags for building headings, lists, font colors, block quotes, tables, and more. You can even define your own HTML tags and access them from the Html menu.

When you install w.bloggar, enter your Blogger username and password. The program retrieves your list of blogs, displaying them in a drop-down menu in the editing interface. You can choose one of

your blogs to post to from the menu, or choose Tools Post to Many Blogs from the top menu to send a single post to several blogs on your list.

w.bloggar doesn't offer a WYSIWYG interface, but the HTML in a post is color-coded, so you can quickly spot the difference between tags and text, as shown in Figure 3-23.

Figure 3-23. Writing a new post in w.bloggar



Once you've entered some text in the editor, you can click the Preview tag to see how the post will look when it's published. When you're finished composing your post, click Post or Post & Publish to send the post to your blog. You also have the option of saving the text to a local file, which can serve as a backup in case anything goes wrong in the publishing process.

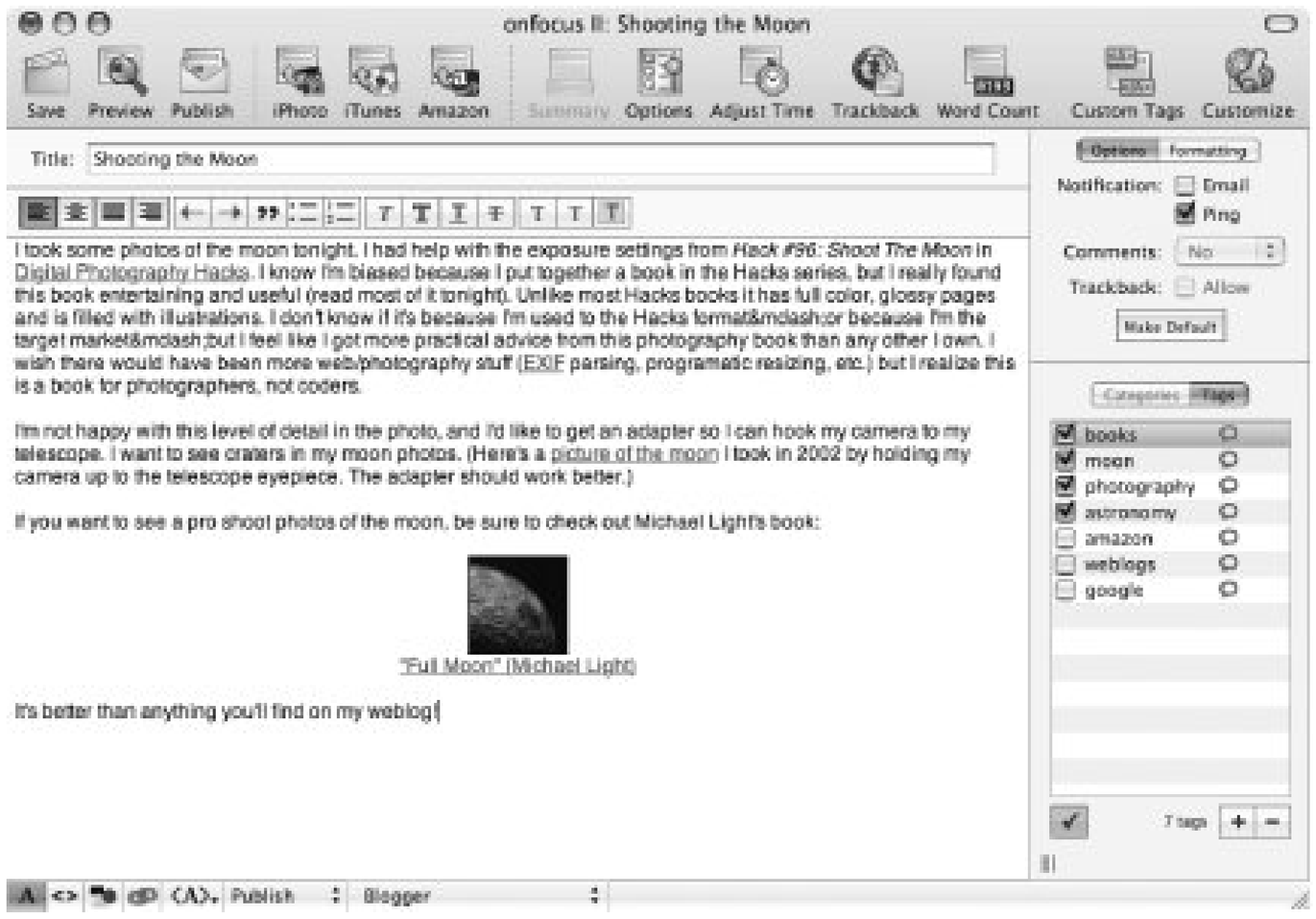
With an optional Media Player plug-in available at the w.bloggar download page (http://wbloggar.com/download.php), you can have one-click access to the current song you're listening to. If you're blasting Kraftwerk in the background while you write, you can click the notes icon at the bottom of the page to insert the track and artist, letting your readers know the background music for the post.

Ecto

Ecto (http://ecto.kung-foo.tv/) offers quite a few more features than w.bloggar, but it will cost you $17.95. (You can try the program free for two weeks.) Ecto was originally developed for Mac OS X, but at the time of this writing, there's a Windows version in beta testing. This description focuses on the Mac version, but many of the same features are in the Windows client.

Ecto sports a WYSIWYG interface, showing formatting and images inline, as shown in Figure 3-24.

Figure 3-24. Writing a new post in Ecto



Ecto can store post templates, which is handy if you frequently post a few types of posts with similar styling. As with many traditional word processors, Ecto spellchecks as you type, underlining misspelled words. You can click the misspelled word to see a list of alternates.

Another handy shortcut is the ability to insert links to products at Amazon.com. Click Amazon to bring up the Amazon Tool shown in Figure 3-25.

Figure 3-25. Composing an Amazon link in Ecto

Enter a keyword, and Ecto communicates with Amazon in the background, bringing up a list of products that match the keyword. From there, you can select a product and click Create Link to auto-insert a mention of it into your post. Ecto composes the HTML necessary to display a picture of the book and link to the book's page on Amazon. Click Options in the Amazon Tool to enter your Amazon Associates tag (http://www.amazon.com/associates) and earn referral fees for sending people to Amazon.

Beyond integration with Amazon, Ecto can also hook your blog into the larger blogosphere through *ping services* and *tags*.

Ping services

A ping service is a site you can notify when you add a new post to your blog. Once*pinged*, the ping service in turn notifies other readers and services that your blog has been updated.

By itself, Blogger offers pinging of only one serviceWeblogs.comwhich you can enable in Settings Publishing.

Many ping services are available, including Technorati, Yahoo!, Blogrolling, and others. You can set Ecto to ping these services as you post by choosing Weblog from the top menu, clicking the Ping button, and adding ping URLs for the various services you want to notify.

Tags

Like ping services, *tags* are a way to connect your blog with the larger blogging world. Tags tell others what your posts are about, and with Ecto, you can set up a list of common tags and simply

check them on the right side of the editing window, as shown in Figure 3-24. As you post, Ecto assembles the HTML necessary to include tags with your post, which are then gathered by Technorat and other services.

Blogger for Word

If you're most comfortable writing text in a word processor, you can use Word itself as your editor, thanks to the aptly named Blogger for Word (http://buzz.blogger.com/bloggerforword.html) developed by Google. Download and install the plug-in, and you'll find a new toolbar when you start Word, as shown in Figure 3-26.

Figure 3-26. Writing a new post in Word with Blogger toolbar enabled

Before you begin, click Blogger Settings, enter your username and password, and choose the blog you want to post to. Compose your post as you would any Word document, and then click Publish to send the text out to the world. You're prompted to add a title to your post, and the plug-in generates the HTML necessary to display the post as you've formatted it. You can optionally set the plug-in to display the HTML source of the post before it's published.

Instead of a simple web form or WYSIWYG editor, you now have the power of a full-featured text editor, with all the Word tools at your disposal. This might seem like overkill for blog posts, but the power of the Blogger API lies in adapting Blogger to your most comfortable writing environment.

The tools mentioned in this hack just scratch the surface of available desktop tools. You can find many more at the Blogger help page for third-party applications at:

http://help.blogger.com/bin/answer.py?answer=1030&topic=43

Moving from the browser to the desktop can seem awkward at first, but with the time-saving options and shortcuts the programs offer, you might end up wondering how you ever blogged without them.

PREV

Hack 46. Program Blogger with PHP

Build Blogger into your applications by tapping into the Blogger API.

If you've ever used a desktop blogging tool [Hack #45] or posted directly to your blog from a web application such as Flickr (http://www.flickr.com), you've already used the Blogger API, though you may not have been aware of it. The Blogger API (http://code.blogger.com/archives/atom-docs.html) provides a way to add posts to your blog without going through the standard form at Blogger.com. So you can think of Blogger as a publishing platform that you can build into your own applications. And if you want to build a better way to manage your blog than Blogger provides, the API gives you access to all the functions you'll need.

Working directly with the API can be a bit of a challenge if you're new to programming, but there are some ways to speed things up. This hack shows a quick way to add posts to your blog and should give you a starting point for integrating Blogger with your own applications.

What You Need

This code uses the excellent PHP Atom API (http://dentedreality.com.au/phpatomapi/) by Beau Lebens to handle the communication with the Blogger API. Download the package and place the three files in your PHP *includes* directory. If you don't have access to the *includes* directory, place the package files in the same folder as the script.

Once the PHP Atom API is in place, you'll need the *blog ID* of the blog you want to send your posts to. A blog ID is simply a unique number that represents your blog in the Blogger system. Log in to Blogger.com, and you should see your Dashboard with your list of blogs. Click the title of the blog you want to send posts to automatically and note the URL. It should look like this:

```
http://www.blogger.com/posts.g?blogID=[numeric ID]
```

Jot down the numeric ID at the end of the URL; this is your blog ID.

The Code

Save the following code to a file called *post.php*, making sure to include your Blogger username, password, and the blog ID for the blog you want to send posts to:

```
<?php
require_once('class.atomapi.php');
```

```
// Set these to your Blogger.com username and password
$username = 'insert your username';
$password = 'insert your password';
$blogID = 'insert your blog ID';

// You shouldn't need to change these settings
$endpoint = 'https://www.blogger.com/atom';
$auth     = 'Basic';
$post_uri = $endpoint . "/" . $blogID;

// If this is a post, send away
if (isset($_POST['post'])) {
    // Create the new entry
    $entry = new AtomEntry(  );
    $entry->set_title($_POST['title']);
    $entry->set_content($_POST['body']);

    // Get an XML version of this entry
    $entry_xml = $entry->to_xml('POST');

    // Authenticate with the API
    require_once('class.basicauth.php');
    $auth_obj = new BasicAuth($username, $password);

    // POST the entry XML to the service.post for this blog
    $post = new AtomRequest('POST', $post_uri, $auth_obj, $entry_xml);
    $post->exec(  );

    // Check for errors
    if ($post->error(  )) {
        echo 'Error: ' . $post->error(  );
    } else {
        echo '<b>Post Added!</b>';
    }
}
?>
<html>
<body>
<h2>Post to Blogger</h2>
<form action="post.php" method="post">
Title:
<input type="text" name="title" value="" size="47" /><br />
<textarea name="body" rows="10" cols="40" /></textarea><br />
<input type="submit" name="post" value="Add Post" />
</form>
</body>
</html>
```

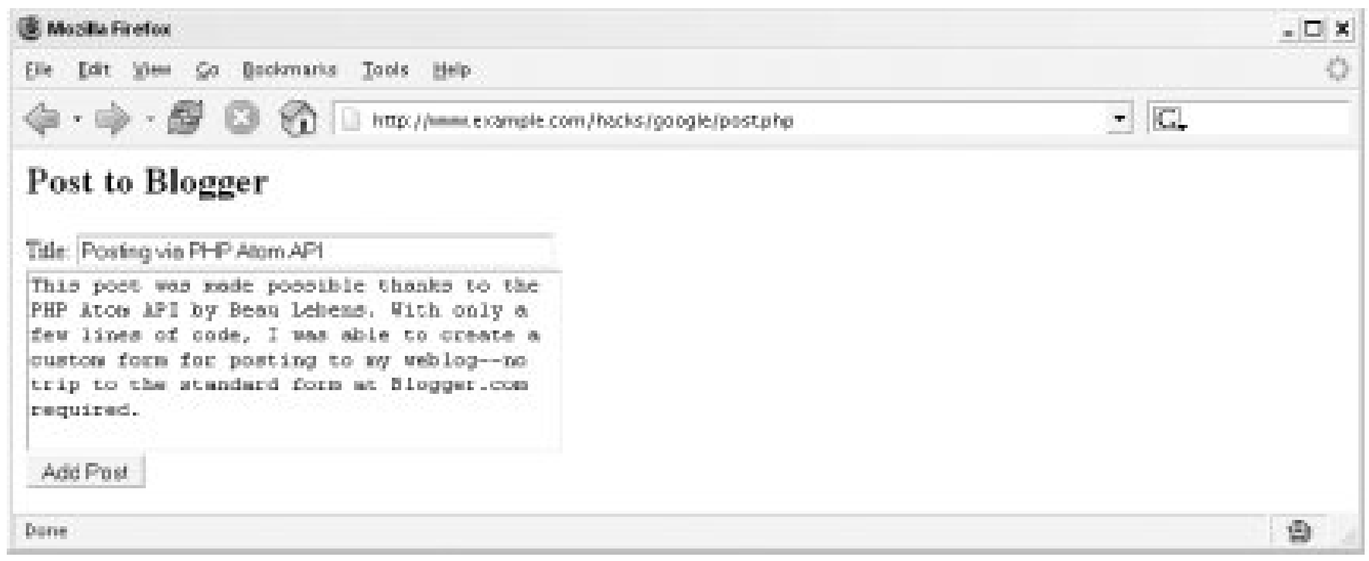
Note that the `require_once` functions point to files in the PHP Atom API package. You might need to adjust the location of the files if they're not in the PHP*includes* folder or the same folder as *post.php*.

Be sure to save *post.php* in a web folder that only you can access. Because the script stores your Blogger username and password, anyone visiting *post.php* has the same authority to post to your blog that you do. Even though your Blogger username and password are sent securely in the background by the PHP Atom API code, it's still your responsibility to secure access to the script.

Running the Hack

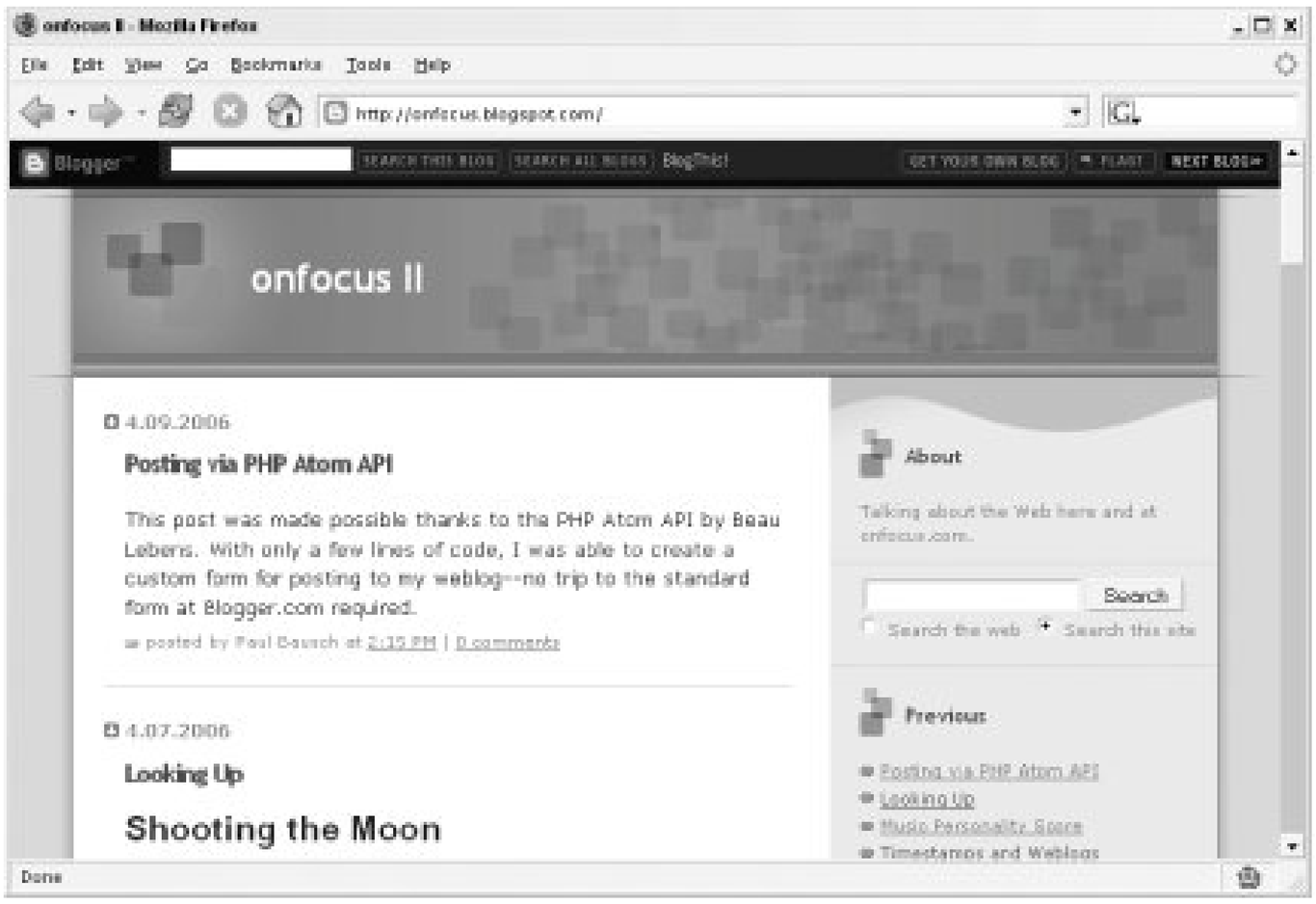
To run the code, browse to the page in your browser. You should see a simple posting form such as the one in Figure 3-27.

Figure 3-27. A custom Blogger post form at a remote site



Write your post as you would if you were at Blogger.com and click Add Post. At this point, your script communicates with the Blogger.com server, adding your text. If all goes well, you should end up with a new post on your blog, as shown in Figure 3-28.

Figure 3-28. A post added via a remote form



This hack illustrates a simple way to use the Blogger API and, hopefully, provides a starting point for your own applications. In addition to adding posts via the API, you can edit and delete posts, and get a list of a user's blogs. And because Blogger uses the open Atom API (http://www.atomenabled.org), any script you write to work with Blogger will also work with other Atom-enabled blog applications, such as TypePad (http://www.typepad.com) and Movable Type (http://www.sixapart.com/movabletype/).

Chapter 4. Extending Google

Hacks 4762

Google is *of* the Web, but this doesn't mean it's trapped in your browser. Google has become so much a part of the fabric of our everyday lives that it shows up just about everywhere: Google via instant messaging [Hack #52], from a chat room [Hack #50], on your mobile phone [Hack #51]; you can even tweak your browser [Hack #53] to take Google with you to every page you visit.

This chapter is a tour of some of the more interesting ways Google has leapt out of the pages of cyberspace onto your desktop, and into what hackers affectionately call *meat space*: everyday life, to you and me.

PREV

Hack 47. Keep Tabs on Your Searches with Google Alerts

Receive alerts in your email Inbox when something you're after makes its way into the Google Web index, a Google News story, or a post at Google Groups.

There are two classes of search that one generally runs in Google. One is of the sort that you generally run just once: you're trying to find information on some topic, a phone number, or that URl you visited yesterday but have since forgotten.

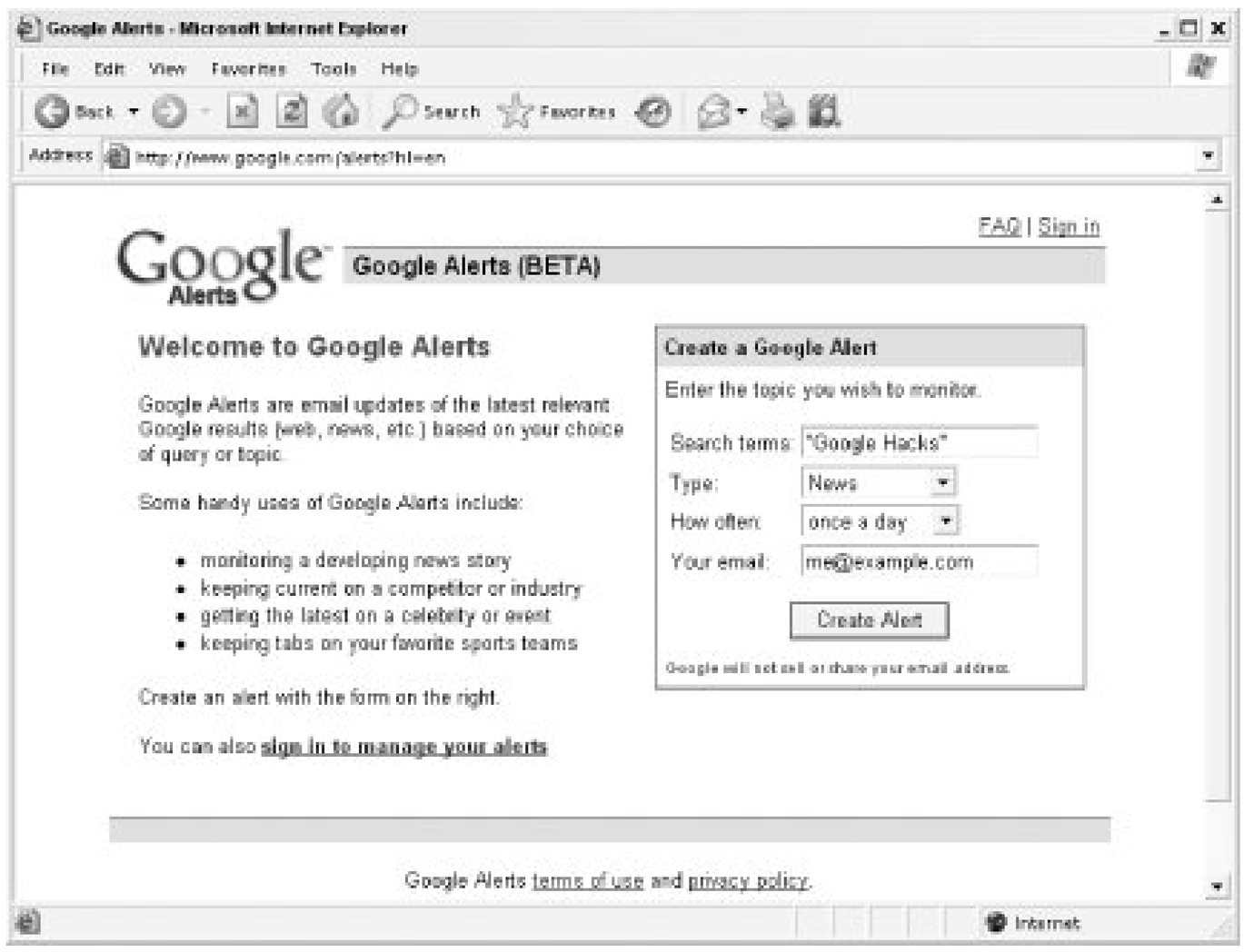
Then there's the search you'd run every day if you could. You're interested in a particular subject matter and want to know the moment Google finds and indexes something new on the topic.

Google Alerts notifies you of any new web pages or news stories that match your search criteria.

> Google's Web index does not consider a page "new" based on the date it was created. Instead, it considers a page new based on the date it was found and indexed by the Googlebot.

Google Alerts (http://www.google.com/alerts) allows you to monitor Google's Web index, Google News stories, and posts at Google Groups. To set up a Google Alert, visit the Google Alerts page. In the Create a Google Alert form (shown in Figure 4-1), type in a search query and choose whether to monitor news, the Web, both News & Web, or Groups.

Figure 4-1. Monitoring the Web, News stories, or Groups postings with Google Alerts



Keep in mind that even though the form is small, you have the full range of Google special syntax at your disposal. For example, if you want to find news about*Google Hacks*, but not every story that mentions the words *Google* and *hacks*, enclose the query in quotes as you would a standard web query.

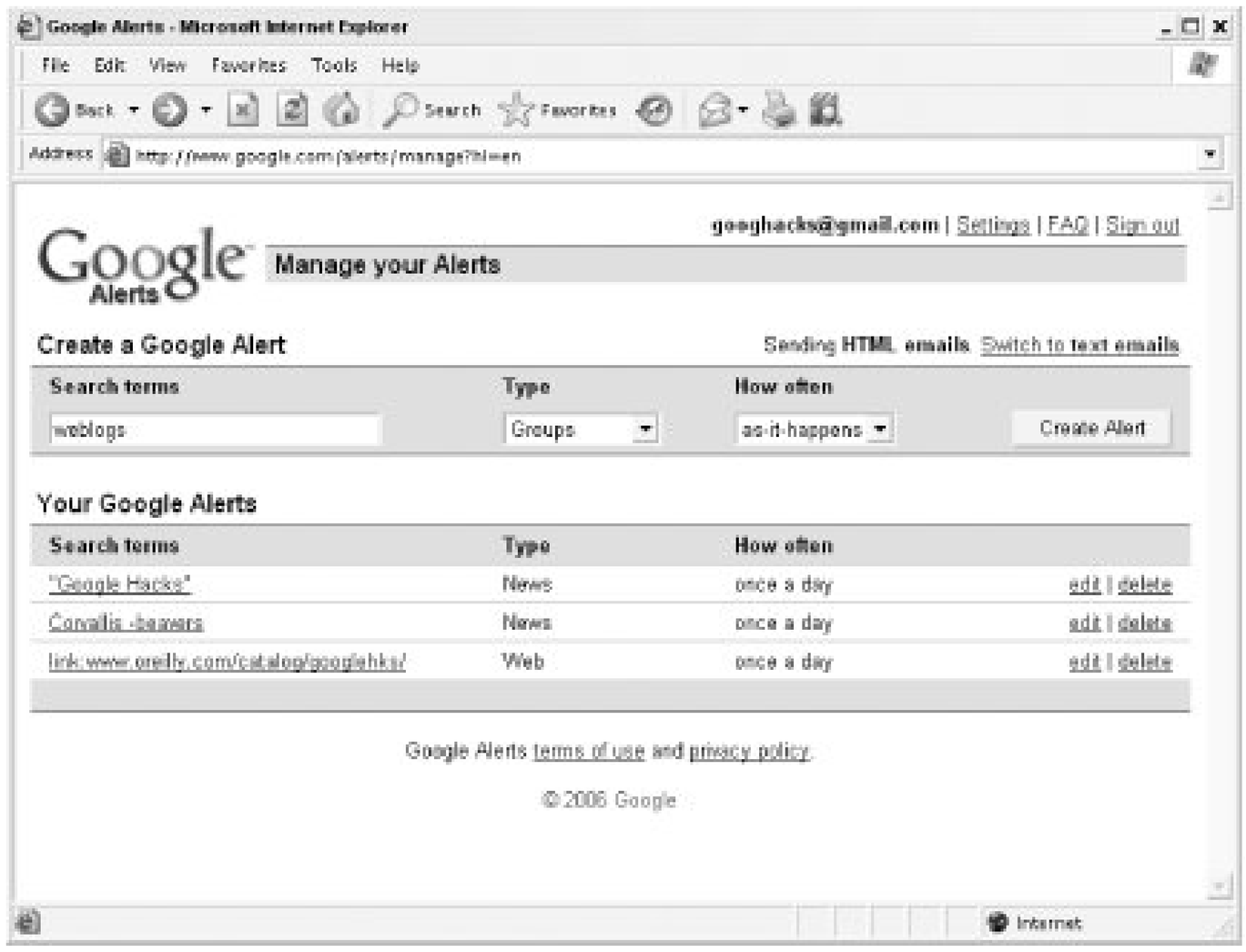
You have a choice when it comes to how often you're notified: as it happens, once a day or once a week. Provide your email address and click the Create Alert button, and you'll receive a confirmation email message a few moments later. Follow the link provided in the email messagethus confirming that your email address is legitimate and that it was you who requested the Google Alertand you're all set.

> Be careful of the update frequency option: monitoring Google News' 4,500 sources for even a slightly common word, phrase, or name and choosing to receive notification "as it happens" can fill your inbox with an avalanche of email.

Each alert you receive includes your search query, the found page's title, a snippet of content, and the URL (for Web index results) or story title, description, and source (for News stories). You can set up to 50 alerts per email address.

While all you need to sign up for Google Alerts is a valid email address, you can also sign up for a more hands-on approach to managing your alerts. On the Google Alerts page, click the "sign in to manage your alerts" link, and you'll find the Manage your Alerts page shown inFigure 4-2.

Figure 4-2. The Google Alerts management form



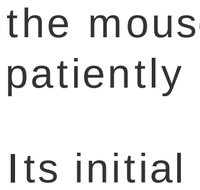
If you haven't already, you'll need to sign up for a free Google account. Membership has its privileges:

- You're provided with a nice overview of your active alerts.
- If you don't sign up to manage your Google Alerts, you can't edit the Google Alerts that you create. All you can do is delete them and create new ones.
- Google Alerts are delivered in HTML format as a default; by signing up, you can switch to text and back again.

In addition to monitoring Google for specific mentions of your business, your web site, or even your name, there are some other ways to use alerts to stay on top of the Web. Monitoring Google's Web index allows you to find search engines or directories of information that you might have missed otherwise. For example, I keep tabs on Google to find pages that don't tend to appear out of thin air all that often, such as those containing `"online museum"` or `"online reference service"`.

I tend to use broader search queries when monitoring Google News. While watching the Google Web index for `"online database"` or `"new search engine"` might net me thousands of resultsand those long after the sites were actually newonline news stories about new online databases and search engines tend to crop up less frequently and provide a higher signal-to-noise ratio.

Hack 48. Google Your Desktop



Google your desktop and the rest of your filesystem, mailbox, and instant messenger conversationseven your browser cache.

Not content just to help you find things on the Internet,Google takes on that teetering pile on your desktopyour computer's desktop, that is.

The Google Desktop (http://desktop.google.com) is your own private little Google server. It sits in the background, slogging through your files and folders, indexing your incoming and outgoing email messages, listening in on your instant messenger chats, and browsing the Web right along with you. Just about anything you see and summarily forget, the Google Desktop sees and memorizesit's like a photographic memory for your computer.

And it operates in real time.

Beyond the initial sweep, that is. When you first install Google Desktop, it uses any idle time to meander your filesystem, email application, instant messages, and browser cache. Imbued with a sense of politeness, the indexer shouldn't interfere at all with your use of your computer; it springs into action only when you step away, take a phone call, or doze off for 30 seconds or more. Pick up the mouse or touch the keyboard, and the Google Desktop scuttles off into the corner, waiting patiently for its next opportunity to look around.

Its initial inventory taken, the Google Desktop server sits back and waits for something of interest to come along. Send or receive an email message, strike up an AIM conversation with a friend, or get started on that PowerPoint presentation, and it's noticed and indexed within seconds.

The full-text Google Desktop indexes:

- Text files, Microsoft Office documents, and PDFs
- Address Book entries and calendars
- Email handled through most major email programs including Outlook, Outlook Express, and Thunderbird
- Instant Messenger conversations
- Web pages you visit

Additionally, any other files you have lying aboutphotographs, MP3s, moviesare indexed by their filename. So if the Google Desktop can't tell a portrait of Uncle Alfred (*uncle_alfred.jpg*) from a song by Uncle Cracker (*uncle_cracker_ _double_wide_ _who_s_your_uncle.mp3*), it files both in a search for `uncle`.

And the point of all this is to make your computer searchable with the ease, speed, and familiar interface you've come to expect of Google. The Google Desktop has its own home page on your computer, shown in Figure 4-3, whether you're online or not. Type in a search query as you would at Google proper and click the Search Desktop button to search your personal index. Or click Search the Web to send your query to Google.

Figure 4-3. The Google Desktop home page

But we're getting a little ahead of ourselves here.

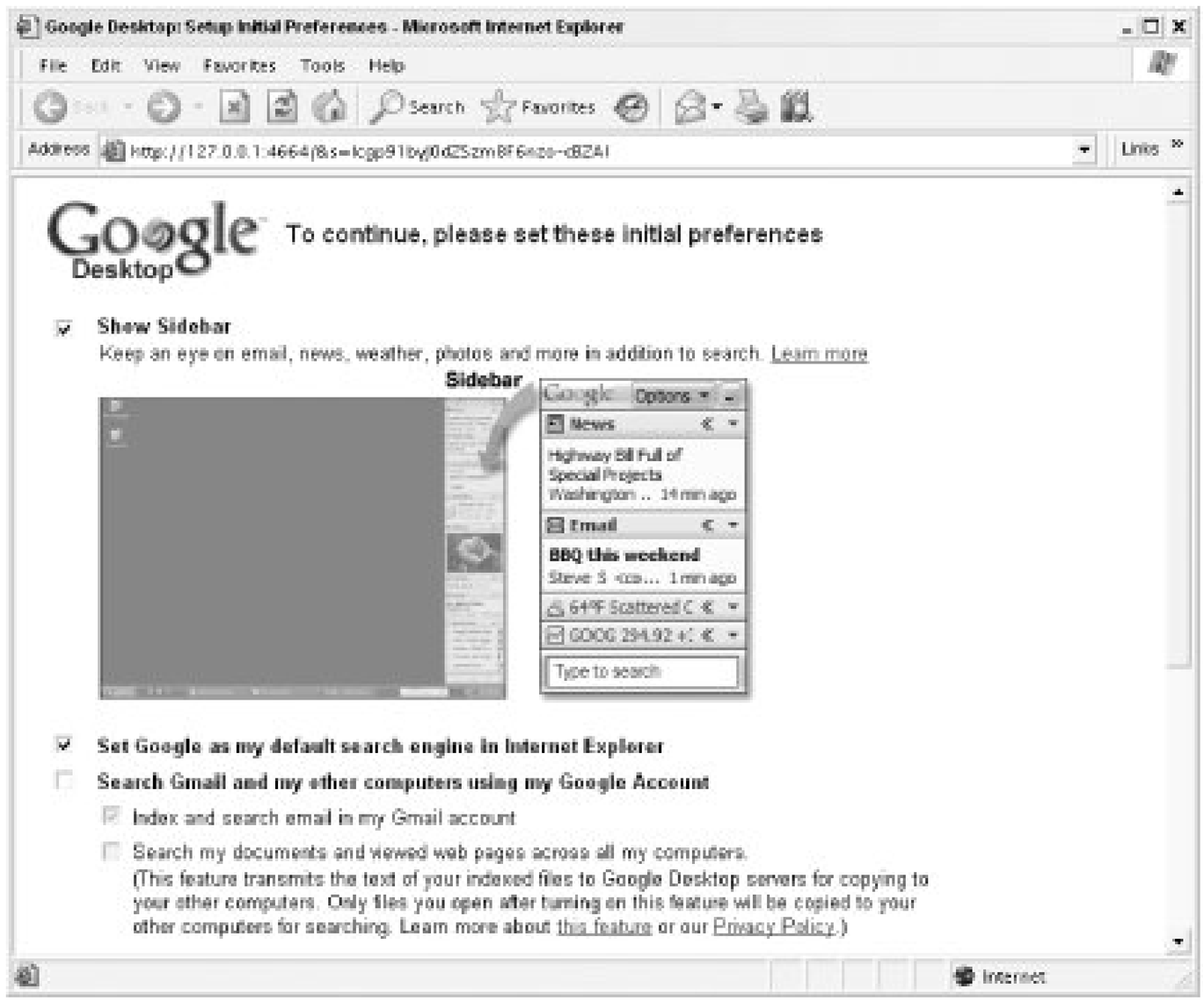
Let's take a few steps back, download and install the Google Desktop, and work our way back to searching again.

Installing the Google Desktop

The Google Desktop is a Windows-only application, requiring Windows XP or Windows 2000 Service Pack 3 or later. The application itself is tiny, but it consumes about 500 MB of room on your hard drive and works best with 400 MHz of computing horsepower and 128 MB of memory.

Point your browser at http://desktop.google.com, download, and run the Google Desktop installer. It installs the application, embeds a little swirly icon in your taskbar, and drops a shortcut onto your desktop. When it's finished installing and setting itself up, your default browser pops open and you're asked to step through a few preferences, as shown in Figure 4-4.

Figure 4-4. Setting Google Desktop preferences



As you step through the installation preferences, you'll notice warnings about privacy, such as the one in Figure 4-5.

Figure 4-5. Advanced Features warning

Google Desktop indexes just about everything on your machine, so it makes sense that Google is very careful about enabling features that communicate information back to the Google servers. If you're trying Google Desktop for the first time, you might want to err on the side of caution and disable Advanced Features; you can always enable them later.

Know that if you enable the "search across computers" option, you'll be sending the contents of your documents to Google's Servers. If you don't mind the idea of your files being posted to Google's servers, you can conveniently search your home computer from your work computer, and from other computers you use. But keep in mind that all of that personal data moving around the Web could be intercepted by a third-party at some point. The computer rights group Electronic Frontier Foundation advises people not to use Google Desktop; you can find its reasoning at its web site (http://www.eff.org/news/archives/2006_02.php#004400).

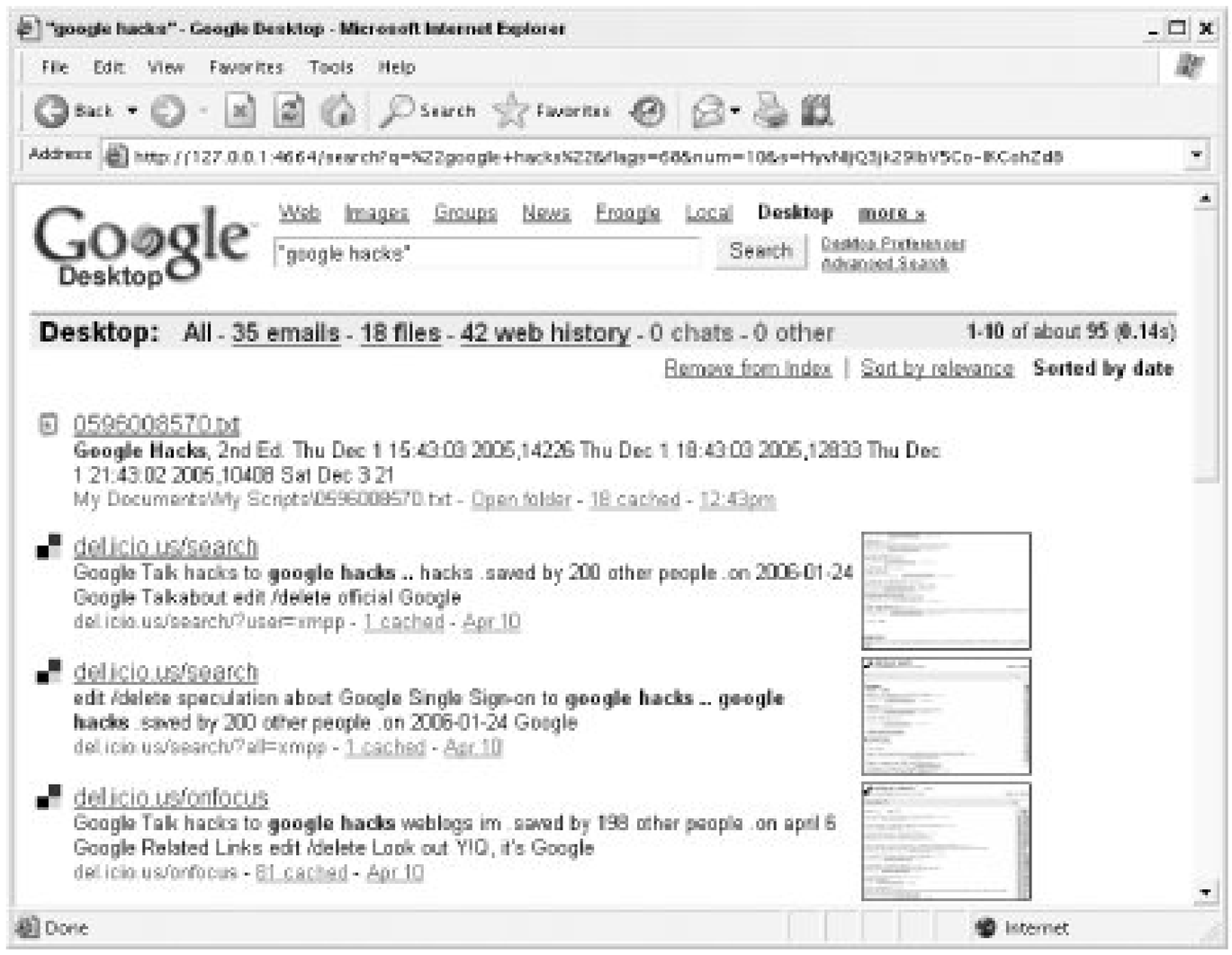
Once you've gone through the wizard, Google Desktop starts its initial indexing sweep.

Searching Your Desktop

From here on out, whenever you look for something on your computer, rather than invoking Windows search and waiting impatiently while it grinds away (and you grind your teeth) and returns with nothing, double-click the swirly Google Desktop taskbar icon and Google for it. Don't bother combing through an endless array of Inboxes, Outboxes, Sent Mail, and folders or wishing you could remember whether your AIM buddy suggested starving or feeding your cold. Click the swirl.

Figure 4-6 shows the results of a Google Desktop search for `"google hacks"`. Notice that it found 35 email messages, 18 files, and 42 items matching my query in my web-browsing history. As you can probably guess from the icons to the left of each result, the first item is a text file, and the rest are from a web site, displaying the site's icon and screenshot along with the result. These are sorted by date, but you can easily switch to relevance by clicking the "Sort by relevance" link at the top right o the results list.

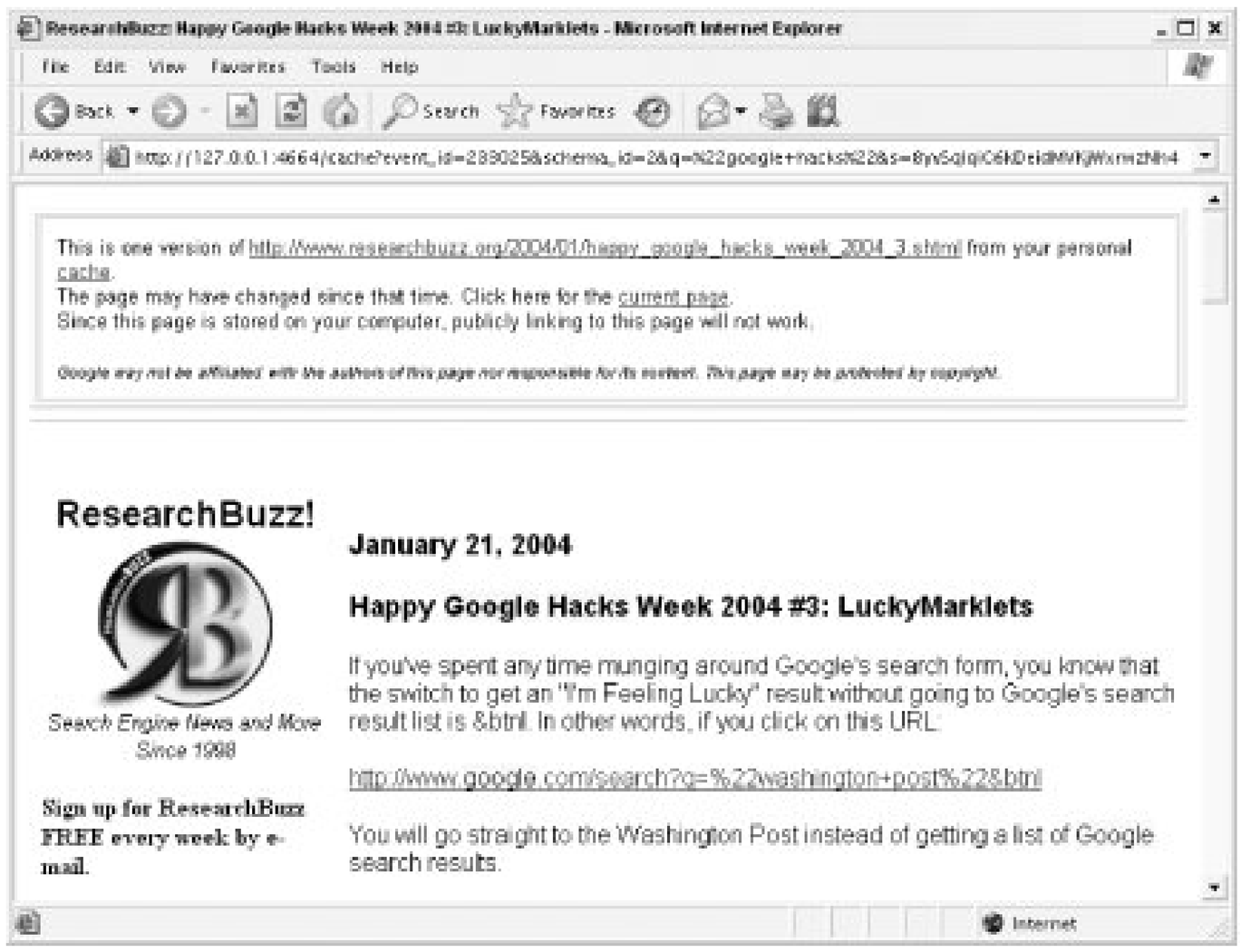
Figure 4-6. Google Desktop search results



Figures 4-7 and 4-8 show individual search results as I clicked through them. Note that each is displayed in a manner appropriate to the content.

Cached web pages are presented, as shown in Figure 4-7, in much the same manner as they are in the Google cache.

Figure 4-7. A cached web page



The various Reply, Reply to All, Forward, etc., links associated with an individual message result (Figure 4-8) work: click them, and the appropriate action is taken by your email program.

Figure 4-8. An email message

Google Desktop Search Syntax

It just wouldn't be a Google search interface without special search syntax to go along with it.

A `filetype:` operator restricts searches to only a particular type of file: `filetype:powerpoint` or `filetype:ppt` (*.ppt* being the PowerPoint file extension) both find only Microsoft PowerPoint files, while `filetype:word` or `filetype:doc` (*.doc* being the Word file extension) both restrict results to Microsoft Word documents.

Searching the Web

Now you'd think I'd hardly need to cover Googling...and you'd be right. But there's a little more to Googling via the Google Desktop than you might expect. Take a close look at the results of a Google search for `"google hacks"` shown in Figure 4-9.

Figure 4-9. Google Desktop Web Search results pack a little extra

Come on back when you're through with that double take.

If you missed it, notice the new quick links ["Quick Links" in Chapter 1]: "157 results stored on your computer."

> When the Google Desktop is running, you can bring up a Google Quick Search box by pressing Ctrl twice. You can also disable this feature in your Google Desktop preference if you find yourself accidentally bringing up the form with a twitchy Ctrl finger.

Yes, those are the same results (and then some, given that my indexer was hard at work) returned in my earlier Google Desktop Search of my local machine. As an added reminder, they're called out by that Google Desktop swirl. Click a local result and you end up in the same place as before: all 157 results, an HTML page, or Microsoft Word document. Click any other quick link or search result, and they'll act in the manner you'd expect from any Google.com results.

Behind the Scenes

Now before you start worrying about the results of a local searchor indeed your local filesbeing sent off to Google, read on. What's actually going on is that the local Google Desktop server intercepts any Google Web Searches, passes them to Google.com in your stead, and runs the same search against your computer's local index. It then intercepts the Web Search results as they come back from Google, pastes in local finds, and presents it in your browser as a cohesive whole.

All work involving your local data is done on your computer. Neither your filenames nor your files themselves are ever sent to Google.comas long as you don't enable the "search across computers" option, which is disabled by default.

For more on Google Desktop and privacy, right-click the Google Desktop taskbar swirl, select About, and click the Privacy link.

Google Desktop Sidebar

Google Desktop includes a desktop sidebar that you can optionally install. The sidebar lives just to the right of your desktop, and you can view it by moving your mouse all the way to the right.

The sidebar keeps track of email, news, weather, and information on your computer from one location. Figure 4-10 shows some of the Google Desktop Sidebar modules, though they're all stacked in a single column on your computer.

Figure 4-10. A slightly dissected view of the Google Desktop Sidebar

When you fire up the sidebar for the first time, the features are personalized for your preferences. Because Google Desktop knows which web pages you visit, you'll find Web Clips based on your personal browsing history. The photos box shows you pictures on your hard drive, and the sidebar

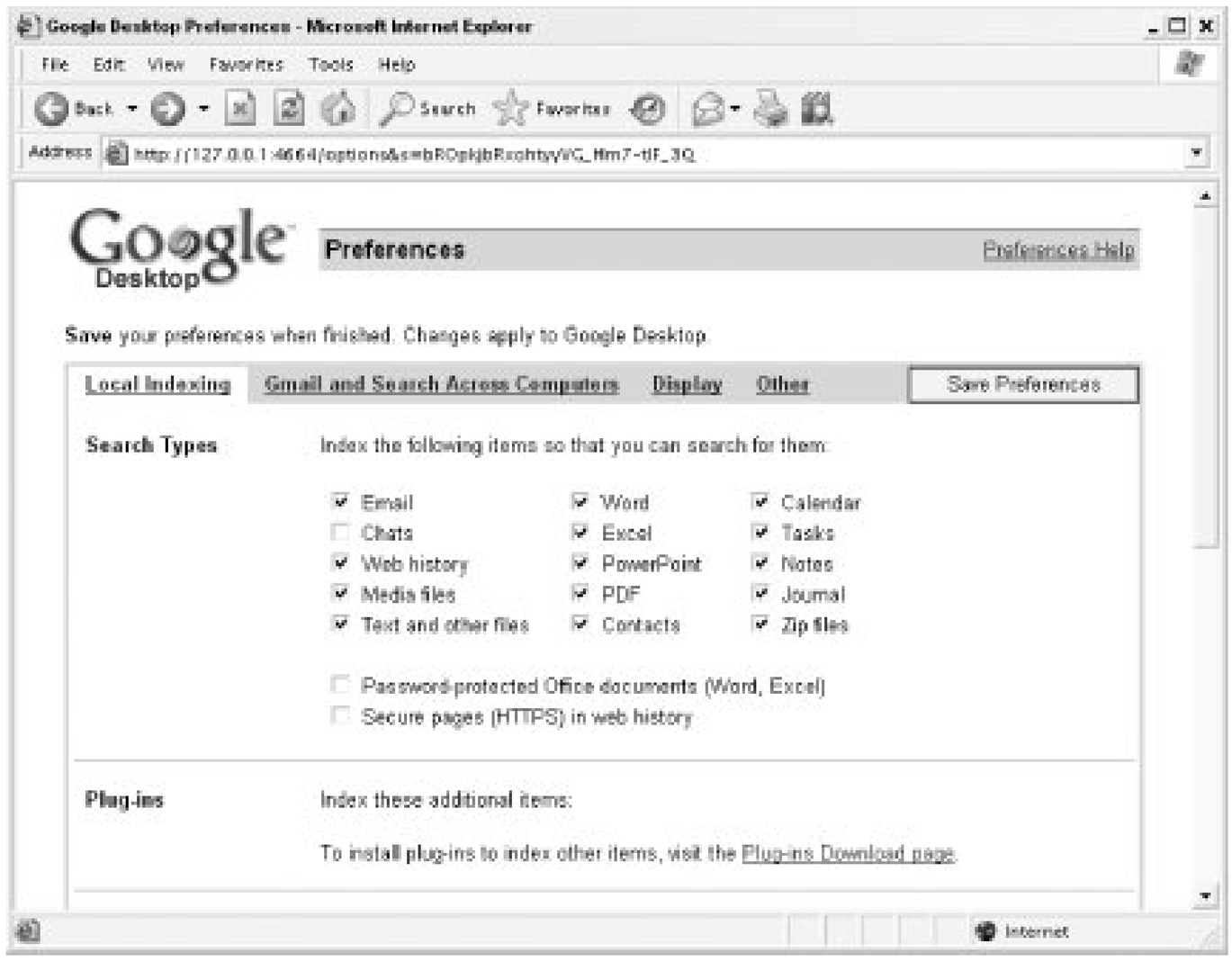
can make some good guesses about your location for displaying weather and maps.

Twiddling Knobs and Setting Preferences

There are various knobs to twiddle and preferences to set through the Google Desktop browser-based interface and taskbar swirl.

Set various preferences in the Google Desktop Preferences page. Click the Desktop Preferences link on the Google Desktop home page or any results page to bring up the settings shown in Figure 4-11.

Figure 4-11. Google Desktop Preferences



Hide your local results when sharing Google Web Search results with a friend or colleague by clicking the Hide link next to any visible Google Desktop quick links. You can also turn Desktop quick link results on and off from the Google Desktop Preferences page.

> Keep in mind that if you want to keep your search history private[Hack #11], the Google Desktop is another place that stores your browsing historyincluding Google searches you've performed. Uncheck the "Web history" option in Google Desktop preferences to keep your searches to yourself.

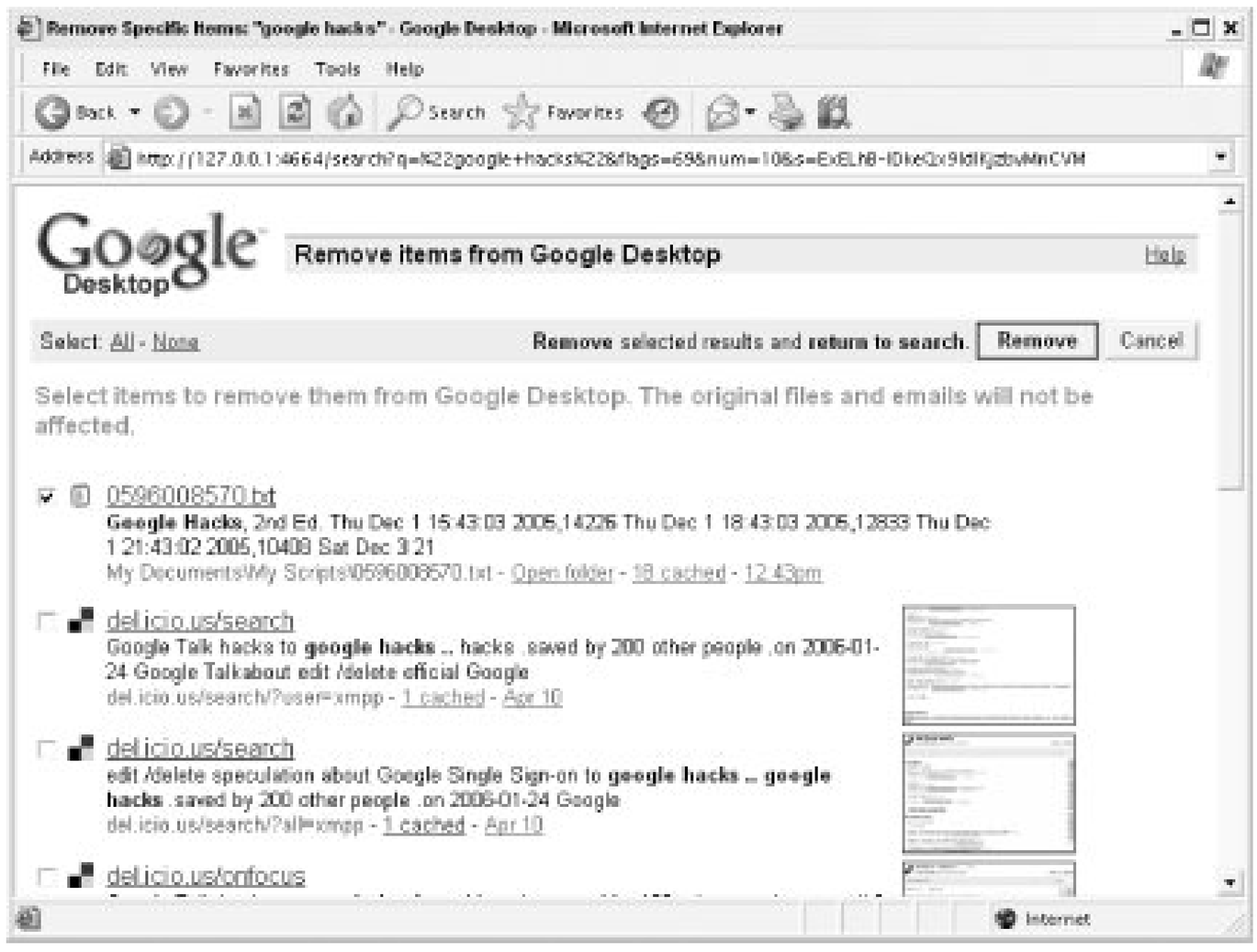
You can also include or exclude specific locations from your Google Desktop index. Just add a folder or web site to the form listed next to Don't Search These Items, and Google's indexer looks the other

way. In addition, you can specifically add a folder to your search if the indexer seems to be missing an important folder.

From the preferences page, you can also enable/disable the Search Across Computers feature that makes your desktop searchable from other locations, and the Advanced Features feature that sends some nonidentifiable information about your Google Desktop usage back to Google's servers. If you want to be "off the grid," make sure both these features are disabled.

If you see something in your search results that you'd rather not see, click the "Remove results" link next to the Search Desktop button on the top-right of any results page, and you can go through and remove those items from Google Desktop index, as shown in Figure 4-12. Note that if you open or view any of these items again, they are once again indexed and will start showing up in search results.

Figure 4-12. Removing items from your Google Desktop index



Search, set preferences, check the status of your index, pause or resume indexing, quit Google Desktop, or browse the "About docs" by right-clicking the Google Desktop taskbar swirl and choosing an item from the menu, shown in Figure 4-13.

Figure 4-13. The Google Desktop taskbar menu

Extending Google Desktop

Google sees Google Desktop as more than an application that helps you organize and manage your information. By offering a software development kit (SDK) for Google Desktop, Google hopes that third-party developers will create their own applications that work with Google Desktop and the Sidebar to manage your information.

To take a look at the available extensions, go to theGoogle Desktop plug-ins page (http://desktop.google.com/plugins/) and browse through the directory. You'll find new Sidebar modules, including one that tracks your AdSense revenue or provides real-time London subway information, and ways to extend Google Desktop, including a way to add Google Desktop to your Windows shell.

After evaluating the Google Desktop as an interface to find needles in my personal haystack, one thing still sticks in my mind: I stumbled across an old email message that I was sure I'd lost.

Hack 49. Google with Bookmarklets

Create interactive bookmarklets to perform Google functions from the comfort of your own browser.

You probably know what *bookmarks* are. But what are *bookmarklets*? Bookmarklets are like bookmarks but with an extra bit of JavaScript magic added. This makes them more interactive than regular bookmarks; they can perform small functions such as opening a window, grabbing highlighted text from a web page, or submitting a query to a search engine. There are several bookmarklets that allow you to perform useful Google functions right from the comfort of your own browser.

If you're using Internet Explorer for Windows, you're in gravy: all these bookmarklets will most likely work as advertised. But if you're using a less-appreciated browser (such as Opera) or operating system (such as Mac OS X), pay attention to the bookmarklet requirements and instructions; you might need special magic to get a particular bookmark working or, indeed, you might not be able to use the bookmarklet at all.

Google Bookmark (http://www.google.com/searchhistory/)

> At the top of the Search History page at Google, you'll find the option to add a Google Bookmark bookmarklet. Don't let the mouthful of a title scare you away; this bookmarklet is a handy way to add starred items (a.k.a. Google Bookmarks) to your personalized Search History [Hack #12]. Click the bookmarklet, and a new windows pops up so you can adjust the bookmark title, notes, or labels before you click Save.

Google Translate! (http://www.microcontentnews.com/resources/translator.htm)

> This puts Google's translation tools into a bookmarklet, enabling one-button translation of the current web page.

Highlight Query Terms (http://www.nimbustier.net/publications/web/bookmarklet-google.html.en)

> If you've ever performed a Google Search for a specific keyword, clicked on a result, and then wondered why that particular page was returned, this bookmarklet is for you. Click the bookmarklet after a Google Search, and all your query terms are highlighted in the page.

The Dooyoo Bookmarklets collection (http://dooyoo-uk.tripod.com/bookmarklets2.html)

This features several bookmarklets for use with different search enginestwo for Google. Similar to Google's Browser Buttons, one finds highlighted text and the other finds related pages.

Joe Maller's Translation Bookmarklets (http://www.joemaller.com/translation_bookmarklets.shtml)

This translates the current page into the specified language via Google or AltaVista.

Bookmarklets for Opera (http://www.philburns.com/bookmarklets.html)

This includes a Google translation bookmarklet,a Google bookmarklet that restricts searches to the current domain, and a bookmarklet that searches Google Groups. As you might imagine, these bookmarklets were created for use with the Opera browser.

LuckyMarklets (http://www.researchbuzz.org/2004/01/happy_google_hacks_week_2004_3.shtml)

Tara's bookmarklets take advantage of the I'm Feeling Lucky feature in Google Web Search, Google News, and Google Images.

Milly's Bookmarklets (http://www.imilly.com/bm.htm)

This is an incredible collection of bookmarklets for all things Google: Web Search, Images, Directory, Definitions, Cache, the Google site itself, and many more, Google or otherwise.

If you find these bookmarks useful, you might want to try building your own bookmarklet for spotting blog commentary [Hack #43] or for adding feeds to Google Homepage or Google Reader [Hack #57].

Hack 50. Google from IRC

Performing Google searches from IRC is not only convenient, but also efficient. See how fast you can Google for something on IRC and click on the URL highlighted by your IRC client.

When someone pops into your IRC channel with a question, you can bet your life that 9 times out of 10, he could have easily found the answer on Google. If you think this is the case, you can tell him that, or you can do it slightly more subtly by suggesting a Google search term to anIRC bot, which then goes and looks for a result.

Most IRC clients can highlight URLs in channels. Clicking on a highlighted URL opens your default web browser and loads the page. For some people, this is a lot quicker than finding the icon to start their web browser and then typing or pasting the URL. More obviously, a single Google search will present its result to everybody in the channel.

The goal is to have an IRC bot, called GoogleBot, that responds to the `!google` command. It responds by showing the title and URL of the first Google search result. If the size of the page is known, this is also displayed.

The Code

First, unless you've already done so, you need to grab a copy of theGoogle Web APIs Developer's Kit (http://www.google.com/apis/download.html), create a Google account, and obtain a license key [Chapter 8]. As I write this, the free-license key entitles you to 1,000 automated queries per day. This is more than enough for a single IRC channel.

The *googleapi.jar* file included in the kit contains the classes the bot uses to perform Google searches, so you need to make sure this is in your classpath when you compile and run the bot (the simplest way is to drop it into the same directory as the bot's code itself).

GoogleBot is built on the PircBot Java IRC API (http://www.jibble.org/pircbot.php), a framework for writing IRC bots. You need to download a copy of the PircBot ZIP file, unzip it, and drop*pircbot.jar* into the current directory, along with the*googleapi.jar*.

> For more on writing Java-based bots with the PircBot Java IRC API, be sure to check out "IRC with Java and PircBot" [Hack #35] in *IRC Hacks* by Paul Mutton (O'Reilly).

Create a file called *GoogleBot.java*:

```
import org.jibble.pircbot.*;
import com.google.soap.search.*;

public class GoogleBot extends PircBot {

    // Change this so it uses your license key!
    private static final String googleKey = "insert your api key";

    public GoogleBot(String name) {
        setName(name);
    }

    public void onMessage(String channel, String sender, String login,
            String hostname, String message) {

        message = message.toLowerCase(  ).trim(  );
        if (message.startsWith("!google ")) {
            String searchTerms = message.substring(8);

            String result = null;
            try {
                GoogleSearch search = new GoogleSearch(  );
                search.setKey(googleKey);
                search.setQueryString(searchTerms);
                search.setMaxResults(1);
                GoogleSearchResult searchResult = search.doSearch(  );
                GoogleSearchResultElement[] elements =
                        searchResult.getResultElements(  );
                if (elements.length == 1) {
                    GoogleSearchResultElement element = elements[0];
                    // Remove all HTML tags from the title.
                    String title = element.getTitle(  ).replaceAll("<.*?>", "");
                    result = element.getURL(  ) + " (" + title + ")";
                    if (!element.getCachedSize(  ).equals("0")) {
                        result = result + " - " + element.getCachedSize(  );
                    }
                }
            }
            catch (GoogleSearchFault e) {
                // Something went wrong. Say why.
                result = "Unable to perform your search: " + e;
            }

            if (result == null) {
                // No results were found for the search terms.
                result = "I could not find anything on Google.";
            }

            // Send the result to the channel.
            sendMessage(channel, sender + ": " + result);
        }
    }
}
```

```
}
```

Your license key is a simple string, so you can store it in the GoogleBot class as `googleKey`.

You now need to tell the bot which channels to join. If you want, you can tell the bot to join more than one channel, but remember, you are limited in the number of Google searches you can do per day.

Create the file *GoogleBotMain.java*:

```
public class GoogleBotMain {

    public static void main(String[] args) throws Exception {
        GoogleBot bot = new GoogleBot("GoogleBot");
        bot.setVerbose(true);
        bot.connect("irc.freenode.net");
        bot.joinChannel("#irchacks");
    }

}
```

Running the Hack

When you compile the bot, remember to include both *pircbot.jar* and *googleapi.jar* in the classpath:

```
C:\\java\\GoogleBot> javac -classpath .;pircbot.jar;googleapi.jar *.java
```

You can then run the bot like so:

```
C:\\java\\GoogleBot> java -classpath .;pircbot.jar;googleapi.jar GoogleBotMain
```

The bot then starts up and connects to the IRC server.

Figure 4-14 shows GoogleBot running in an IRC channel and responding with the URL, title, and size of each of the results of a Google search.

Figure 4-14. GoogleBot performing IRC-related searches

Performing a Google search is a popular task for bots to do. Take this into account if you run your bo in a busy channel because there might already be a bot there that lets users search Google.

Paul Mutton

PREV

Hack 51. Google on the Go

Being on the go and away from your laptop or desktop doesn't mean leaving Google behind.

As the saying goes, "You can't take it with you." Unless, that is, you're talking about Google. Just because you've left your laptop at home or at the office, that doesn't necessarily mean leaving the Web and Google behind. So long as you have your trusty cell phone or network-enabled PDA in your pocket, so too do you have Google.

Whether you have the top-of-the-line Treo 700, Blackberry, or Sidekick with integrated web browser; base-model cell phone that your carrier gave you for free; or anything in between, chances are you can Google on the go.

Google caters to the "on the go" crowd with its Google wireless interfaces: a simpler, lighter, gentler PDA- and smartphone-friendly version of Google, a WAP (read: wireless Web) flavor for cell phones with limited web access, and an SMS gateway for messaging your query to and receiving an almost instantaneous response from Google. You can also take the power of Google Maps and Google Local [Hack #63] with you so you won't get lost again. And there's even a mobile interface to Google's Froogle (http://froogle.google.com) product search.

Google by PDA or Smartphone

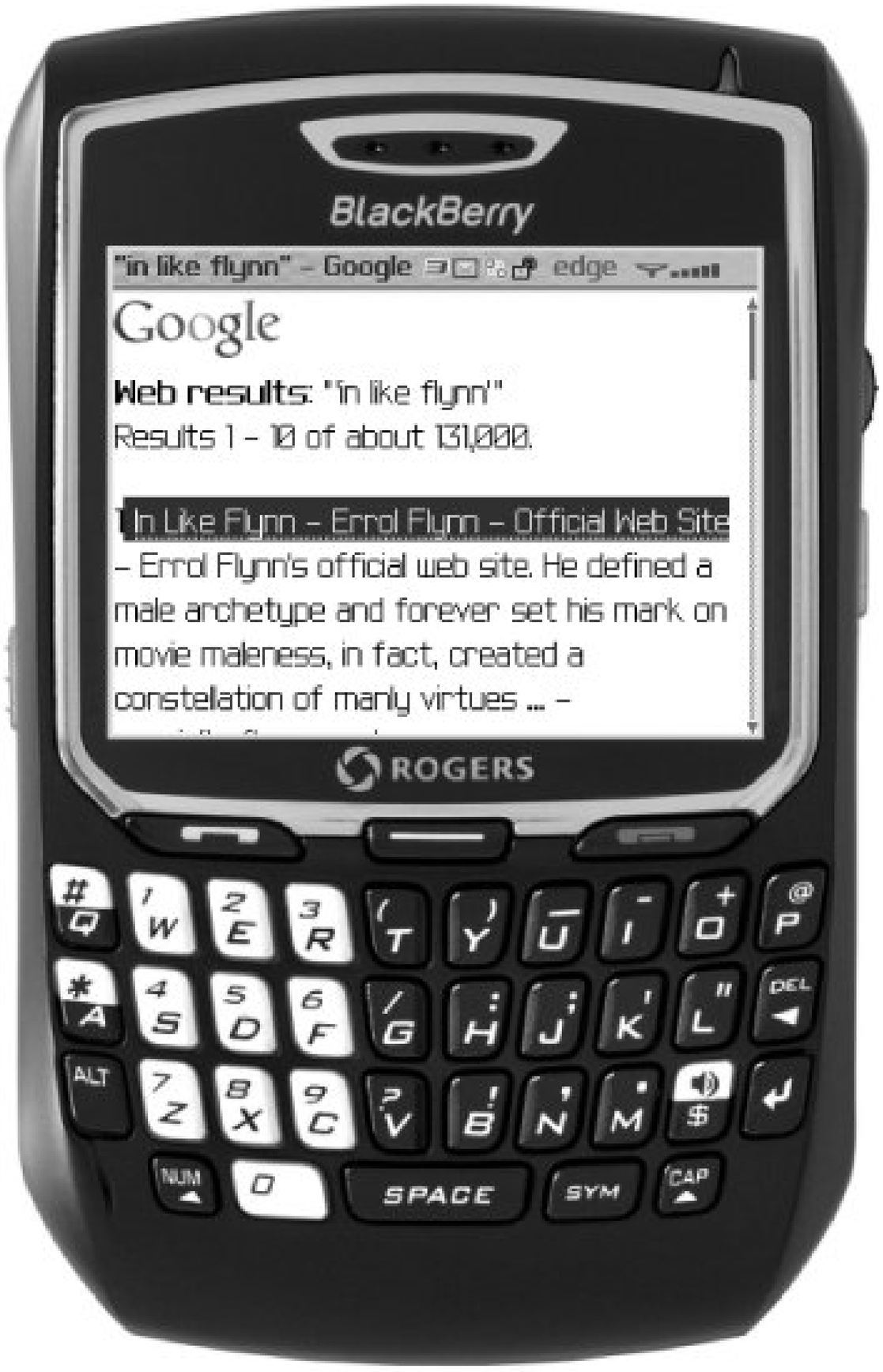
Google PDA Search (http://www.google.com/pda) brings all the power of Google to the PDA in your palm, hiptop on your belt, or cell phone in your pocket.

Settle that "in like Flynn" versus "in like Flint" dinner-table argument without leaving your seat. Find quickie reviews and commentary on that Dustmeister 2000 vacuum *before* making the purchase. Figure out where you've seen that bit-part actor before without having to wait for the credits.

Your modern PDA and the smarter so-called *smartphones* sport a full-fledged web browser on which you can surf all that the Web has to offer in living coloralbeit substantially smaller. You find the usual Address Bar, Back and Forward buttons, Bookmarks or Favorites, and point-and-click (or point-and-tap, as the case may be) hyperlinks. While the onboard browser might be able to handle the regular Google.com web pages, the Google PDA Search provides simpler, smaller, no-nonsense, plain HTML pages. And results pages pack in fewer results for faster loading.

Just point your mobile browser at http://google.com/pda, enter your search terms, click the Google Search button, and up come your results, as shown in Figure 4-15.

Figure 4-15. Google PDA search results on a Blackberry



You have the full range of Google Search syntax [Chapter 1] and complete Web index available to you, although it might be more than a little challenging to enter those quotes, colons, parentheses, and minus signs.

Google by Cell Phone

If you have a garden-variety cell phonethe kind your mobile provider either gives away free with signup or charges on the order of $40 foryou may already have a built-in browser...of a sort. Don't expect anything nearly as fast, colorful, or feature-filled as your computer's web browser. This is a text-only world, limited in both display and interactivity.

That said, you have the wealthif not the Technicolorof the Web right in your pocket.

Step one, however, is to find the browser in the first place. It's usually cleverly hidden behind some (possibly meaningless) moniker such as WAP, Web, Internet, Services, Downloads, or a brand name such as mMode or T-Zones. If nothing of the sort leaps out at you, look for an icon sporting your cell phone provider's logo, take a stroll through the menus, dig out your manual, or give your provider a ring (usually 611 on your cell phone).

Texting Sure Ain't QWERTY

Whether you're a 70-word-per-minute touch typist or hunt and peck your way through the QWERTY keyboard, you'll initially findtexting to be a pokey chore. Rather than the array of letters, numbers, symbols, and Shift keys on your computer keyboard, everything you type on your cell phone is confined to 12 keys: 09, *, and #. Frankly, it's an annoying system to learn, but once you get used to it, it's not too painful to use; some folks actually become rather adept at it, rivaling their regular keyboarding speeds.

Look closely at your phone and notice that each button also holds either a set of three to four alphabetic characters or obscure symbols not unlike those you'd expect to find on a UFO that landed in your backyard. Like your regular phone, the 1 button is devoid of letters, while 2 has ABC, 3 DEF, and so on up to 9, which has WXYZ.

When you're in web-browsing mode on your phone, you can tap the 2 button once to type an A, twice in quick succession for a B, and thrice for a C. Four times nets you a 2. Keep going and you'll make it back through A, B, C, and 2 againon some phones, encountering strange and wonderful foreign letters along the way. Do this for each and every letter in the word you're trying to spell out, spelling the word "google" like so: 4666 666455533. Notice the gap between the 666 and 666? What you're after is two "o"s in a row, but typing 666666 gets you either a single "o" or an "⊘" because your phone doesn't know when you want to move on to the next letter. To type two of the same characters one after another, either wait a second or so after tapping in the first "o" or move your phone's joystick to the right or down.

When it comes to special characters such as the dot (.) and slash (/) common in web addresses, you turn to the 1 button. A period or dot is a single tap. The slash is usually 15. For those of you keeping score at home, this leaves you with 92714666 6664555331 11111111111111196555 for wap.google.com/wml.

The texting equivalent of the spacebar is the 0 button.

What of digits? Surely, you don't need to type 17 or so 1sscrolling through all the symbols associated with the 1 button ([.,-?!'@:;/( )])just to get back to the 1 you wanted in the first place. Thankfully, all it takes is holding down the button for a second or so to jump right to the numeral. So instead of tapping through WXYZ to get to 9, hold down the 9 key for a moment or so and you're there.

There are more efficient input techniques, such as*T9* ("Text on 9 keys") and other predictive text systems, but they're not as useful for entering possibly obscure words such as those in web addresses and Google searches.

Browser in hand, point it at wap.google.com/wml, tap in a search (without tripping over your fingers), and click the Search button or link (as shown in Figure 4-16, left). A few moments later, your first set of results show up (as shown in Figure 4-16, right). Scroll to the bottom of the results and click the Next link to move to the next page of results.

Figure 4-16. Google wireless search home (left) and results (right)

Click any of the results to visit the page in question, just as you would in a normal browser. You'll notice immediately that the pages you visit by clicking a result link are dumbed downsimilar to Google's wireless search itselfto suit the needs of your mobile's display abilities.

Truth be told, you're not directly visiting the resulting page at all. What you see on your screen and in Figure 4-17 is courtesy of the Google WAP proxy, a service that turns HTML pages intoWAP/WML (think of it as HTML for wireless devices) on the fly. Click another link on the resulting page and you can continue browsing via the Google proxy. Google essentially turns the entire Web into a mobile Web.

Figure 4-17. A piece of the O'Reilly home page seen through the lens of the Google WAP proxy

In fact, you can actually surf rather than search the Web using the Google WAP proxy. Find your mobile browser's Options menu and click the Go to URL link. In the resulting page, enter any web site URL into the Go to URL box and click the Go button to visit a mobile version of that page.

> The Google WAP proxy is also a handy addition to your phone's bookmarks. Add the following URL to access the proxy directly: http://google.com/gwt/n. In fact, you can visit this link in a standard web browser to preview what your favorite sites will look like once they've been stripped to their bare essentials for mobile browsing.

Google by SMS

As a *New York Times* article, "All Thumbs, Without the Stigma" at:

> http://tech2.nytimes.com/mem/technology/techreview.html?res=9E00E6DE163FF931A2575BC0A9629C8B63

suggested recently, the thumb is the power digit. While the thumboard of choice for executives tends to be the Blackberry mobile email device (http://www.blackberry.com/), for the rest of the world (and for many of the kids in your neighborhood), it's the cell phone andSMS.

SMS messages are quick-and-dirty text messages (think mobile instant messaging) tapped into a cell phone and sent over the airwaves to another cell phone for around $.05 to $.10 apiece.

But SMS isn't just for person-to-person messaging. In the UK, BBC Radio provides so-called *shortcodes* (really just short telephone numbers) to which you can SMS your requests to the DJ's automated request-tracking system. You can SMS bus and rail systems for travel schedules. Your airline can SMS you updates on the status of your flight. And now you can talk to Google via SMS as

well.

Google SMS (http://www.google.com/sms/) provides an SMS gateway for querying the Google Web index, looking up phone numbers [Hack #5], seeking out definitions [Hack #6], and comparative shopping in the Froogle product catalog service (http://froogle.google.com).

Simply send an SMS message to U.S. shortcode `46645` (read: `GOOGL`) with one of the following forms of query:

Google Local Business Listing

Consult Google Local's business listings by passing it a business name or type and city, state combination, or zip code:

```
vegetarian restaurant Jackson MS
southern cooking 95472
scooters.New York NY
```

The Google SMS documentation suggests using a period (.) between your query and city name or zip code to be sure that you're triggering a Google Local Search.

Residential Phone Number

Find a residential phone number with some combination of first or last name, city, state, zip code, or area code. Or enter a full phone number without punctuation to do a reverse-lookup:

```
augustus gloop Chicago il
violet beauregard 95472
mike teevee ny
7078277000
```

As with any Google Phonebook [Hack #5] query, you'll find only listed numbers in your results.

Froogle Prices

Check the current prices of items for sale online through Froogle (http://froogle.google.com/). To trigger a Froogle lookup, prefix your query with an `F` (upper- or lowercase), `price`, or `prices` (the latter two also work at the end of the query):

```
price bmw 2002
ugg boots prices
```

Definition

Rather than scratching your head trying to understand just what Ms. Austen means by *disapprobation*, ask Google for a definition [Hack #6]. Prefix the word or phrase of interest with a `D` (upper- or lowercase) or the word `define`:

```
D disapprobation
define osteichthyes
```

Calculation

Perform feats of calculation and conversion using the Google Calculator [Hack #32]:

```
(2*2)+3
12 ounces in grams
```

Zip Code

Pass Google SMS a U.S. zip code to find out where it is in the country:

```
95472
```

> Google SMS is sure to sport more features by the time you read this. Be sure to consult the "Google SMS: How to Use" page at http://www.google.com/sms/howtouse.html for the latest orfor the real thumb jockeys among yousubmit your email address to an announcement list from the Google SMS home page.

Sports Scores

Send the name of a college or pro team and get back thescore of its most recent game:

```
sf giants
oregon state
```

Currency Conversion

Include the name of a currency and an amount in your message, and get back the current value:

```
300 usd in eur
500 yen in pounds
```

Facts and Figures

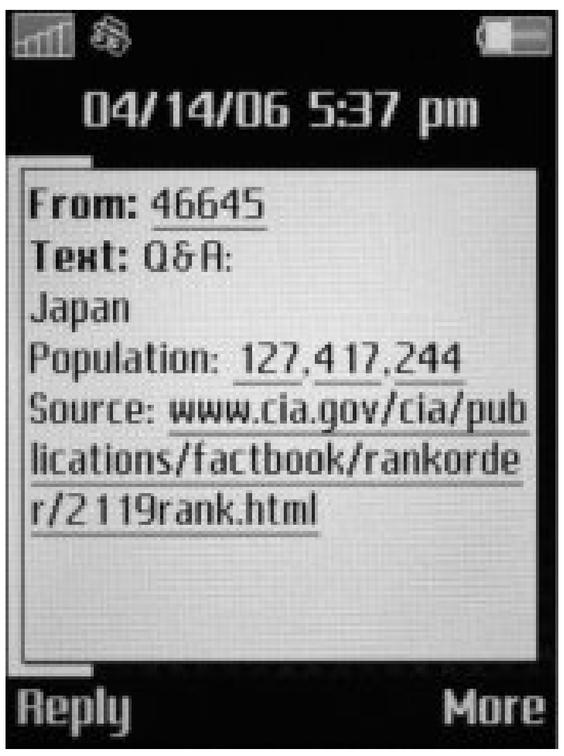
This one settles your bar bets. Send a question and get back an answer if Google has one. For

example:

```
calories in milk
people in japan
```

You receive your results as one or more SMS messages labeled, appropriately enough,*1of3*, *2of3*, etc. if the answer doesn't fit on a single screen. Notice that there are links to URLs in the responses, as shown in Figure 4-18.

Figure 4-18. A Google SMS query response



In addition to your answer, you can often find the source of the answer in the message. As in Figure 4-18, the answer is straight from the CIA Factbook.

> While the cost of sending an SMS messages is usually paid by the sender, automated messages such as those sent by Google SMS are usually charged to you, the receiver. Unless you have an unlimited SMS plan, all that Googling can add up. Be sure to check out what's included in your mobile plan, check your phone bill, or call your mobile operator before you spend a lot of time (and money) on this service.

Froogle on the Go

If you wish you could compare prices at that "One Day Sale" on kitchen gadgets without leaving the store, Wireless Froogle (http://froogle.google.com) is as much a part of the shopping experience as that credit card.

Point your mobile browser at http://wml.froogle.com and tap in the name of the product you're about to take to the checkout, and up pops a list of prices as advertised by online vendors, as shown in Figure 4-19.

Figure 4-19. Wireless Froogle Search results

You'll find everything from cellular phones to yogurt makers, abacuses to faux yak fur coats on Froogle.

At the time of this writing, Wireless Froogle is nowhere near as complete as one might hope. You can't constrain your results by price, group them by store, or sort them in any way. Results don't link to anywhere. That said, it is a still a handy price-check tool as you're standing in that checkout line.

$44 for a pashmina? Lemme at it! Sometimes instant gratification is worth it, and sometimes paying only $44 for silk is well worth the wait.

Maps on the Go

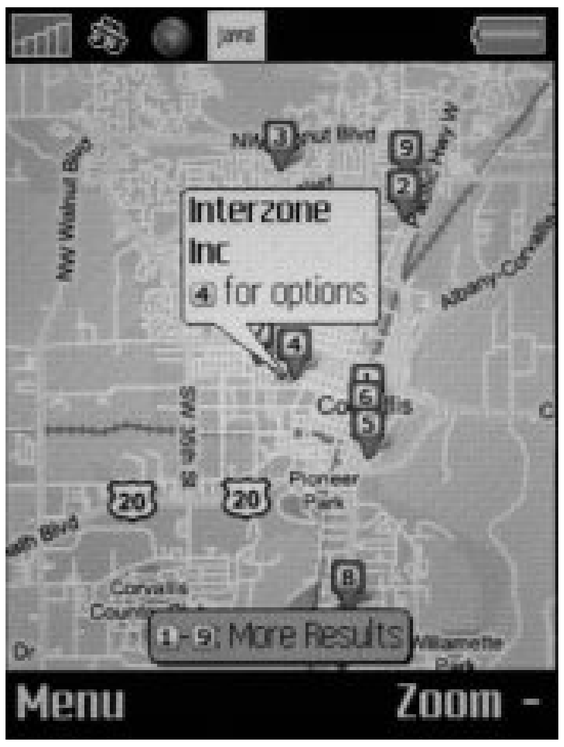
With a bit of prep beforehand, you can take the power of Google Maps[Hack #64] and Google Local with you as you travel. Local for mobile includes the clickable, dragable maps you find at Google Maps on your cell phone. Instead of a site that runs through your phone's browser, Local for mobile is an application you can download and install on your phone. Browse to the Local for mobile site (http://www.google.com/glm/) and click Get Started to see if your phone is supported.

If your phone is on the list, the web site walks you through the installation process. Once installed, the application is available on your phone and the maps are always available to you. When you start the application, the familiar maps interface appears, and you can start zooming and dragging the map.

Imagine you're out and about in Corvallis, Oregon, and you want to grab a cup of coffee. Key in the

phrase `coffee in corvallis` for a map of coffee shops, as shown in Figure 4-20.

Figure 4-20. Local for mobile results for "coffee in corvallis"



Key in the number of a result to see the name of the location, and again to see business details such as address and phone number. There's even a quick link for dialing that particular business with one click if you need to call ahead.

Local for mobile isn't a scaled-down version of Google Maps; it's a fully functional version of Google Maps. You can even choose the satellite view from the menu to see satellite images of a particular area. Figure 4-21 shows a satellite image of the Pacific Northwest and California on a cell phone.

Figure 4-21. Satellite view in Local for mobile

You might not need satellite photos to navigate your way to the nearest coffee shop, but you can travel comfortably knowing you can scope out the topography of your current location.

When you're at a computer, be sure to stop by Google Mobile (http://www.google.com/mobile/) to stay on top of all of Google's mobile offerings. Google is continually making its features available via mobile devices to make sure you can access your data and the Web whereveryou need it.

PREV

Hack 52. Google over IM

Build a Google Talk bot that will have you talking directly with Google Search results.

Instant messaging is no longer solely the domain of teenagers with too much spare time on their hands. IM has morphed from a fun toy to a serious productivity tool. Google offers its own IM client called Google Talk (http://www.google.com/talk/).

Thanks to the Google Talk and the Google API, you can skip the web site and bring Google Search results directly into your IM client with a Google Talk bot. A *bot* is simply a program that looks like an IM chat buddy (someone who receives and sends messages). Behind the scenes, though, the bot simply does what it's programmed to do. With a Google Talk bot up and running, you can find search results without leaving your IM client, which sounds very productive.

What You Need

Google Talk uses a standard messaging protocol called Extensible Messaging and Presence Protocol (XMPP). XMPP was developed as part of Jabber, an open IM protocol. Because Jabber has been around for a number of years, there are plenty of existing tools that speak Google Talk's language.

This hack is written in Python and requires the *jabber.py* module (http://jabberpy.sourceforge.net) for communicating through Google Talk. The code for talking with the Google API is adapted from the simple Python example [Hack #95] and requires the pyGoogle module (http://pygoogle.sourceforge.net). And, of course, you'll need a free Google API key, which you can pick up at Google Web APIs (http://www.google.com/apis/).

You'll also need a spare Google Account for your bot. Log into Gmail with your Google account and send yourself an invitation. Be sure to log out of Google completely, follow the instructions in your Gmail invite, and sign up for Google using your alternate identity. Jot down the alternate account username and password. Remember that your bot will be logging into Google Talk, so whichever name you give your bot when you sign up will be your bot's identity online.

The Code

This code provides a bare-bones bot that handles incoming messages and sends simple messages. In addition, the script queries Google and formats the response for instant messages. Be sure to include the login username and password for your bot. Include your Google API key as well, and then save the following code to a file called *queryBot.py*:

```
#!/usr/bin/python
# queryBot.py
```

```
# A Google Talk bot that returns Google Search
# results as messages for any incoming message.
# Usage: python queryBot.py

import sys
import string
import re
import jabber
import xmlstream
import google

username = 'insert google account name' # do not include @gmail.com
password = 'insert google account password'
google.LICENSE_KEY = 'insert google API key'
botname = 'queryBot'

def sendMsg(toid,msg):
    r = jabber.Message(toid,msg)
    r.setType('chat')
    con.send(r)

def messageCB(con,msg):
    if msg.getBody(  ):
        query = msg.getBody(  )
        fid = msg.getFrom(  )

        print '>>> query: %s' % query
        print '>>> from: %s' % fid

        # Query Google.
        data = google.doGoogleSearch(query)

        # Output.
        for result in data.results:
             # set the results as variables
            title = result.title
            URL = result.URL
            snippet = result.snippet

            # Strip HTML
            regex = re.compile('<[^>]+>')
            title = regex.sub(r'',title)
            snippet = regex.sub(r'',snippet)
            regex2 = re.compile('&#39;')
            title = regex2.sub(r"'",title)
            snippet = regex2.sub(r"'",snippet)

            title = '\\n*%s*' % title # Bold title

            # Format result
            response = string.join( (title, snippet, URL), "\\n")
```

```
            # Send result
            r = jabber.Message(fid,response)
            r.setType('chat')
            con.send(r)

def connect(  ):
    global con
    con = jabber.Client(host='gmail.com',debug=[],
                        log='xmpp.log',port=5223,
                        connection=xmlstream.TCP_SSL)
    con.connect(  )
    con.setMessageHandler(messageCB)
    con.auth(username, password, botname)
    con.requestRoster(  )
    con.sendInitPresence(  )
    print "[[ Bot is Online, ready for queries! ]]"

con = None
while 1:
    if not con:
        connect(  )
    con.process(1)
```

Note that when the Jabber client is initialized in the`connect( )` function, a logfile (*xmpp.log*) is set. You'll find a copy of all XMPP messages flying between your machine and the Google Talk server, and it's extremely useful for finding problems with your bot.

Running the Hack

Before you can run your bot, you'll need a bit of Google login shuffling. To combat spam, Google Talk requires every user to have explicit permission to talk to each other. Log into Google Talk as yourself and send a chat request to your bot's identity. Then log in using your bot's credentials and approve your real self for chatting. You'll need to approve everyone your bot chats with.

Also, make sure both *jabber.py* and *google.py* are in the same directory as your script. If they aren't, install them with the *setup.py* scripts that come with the modules.

Once everything is set, open a command prompt and start the script, like so:

```
python queryBot.py
```

The bot should start and give you the opening OK:

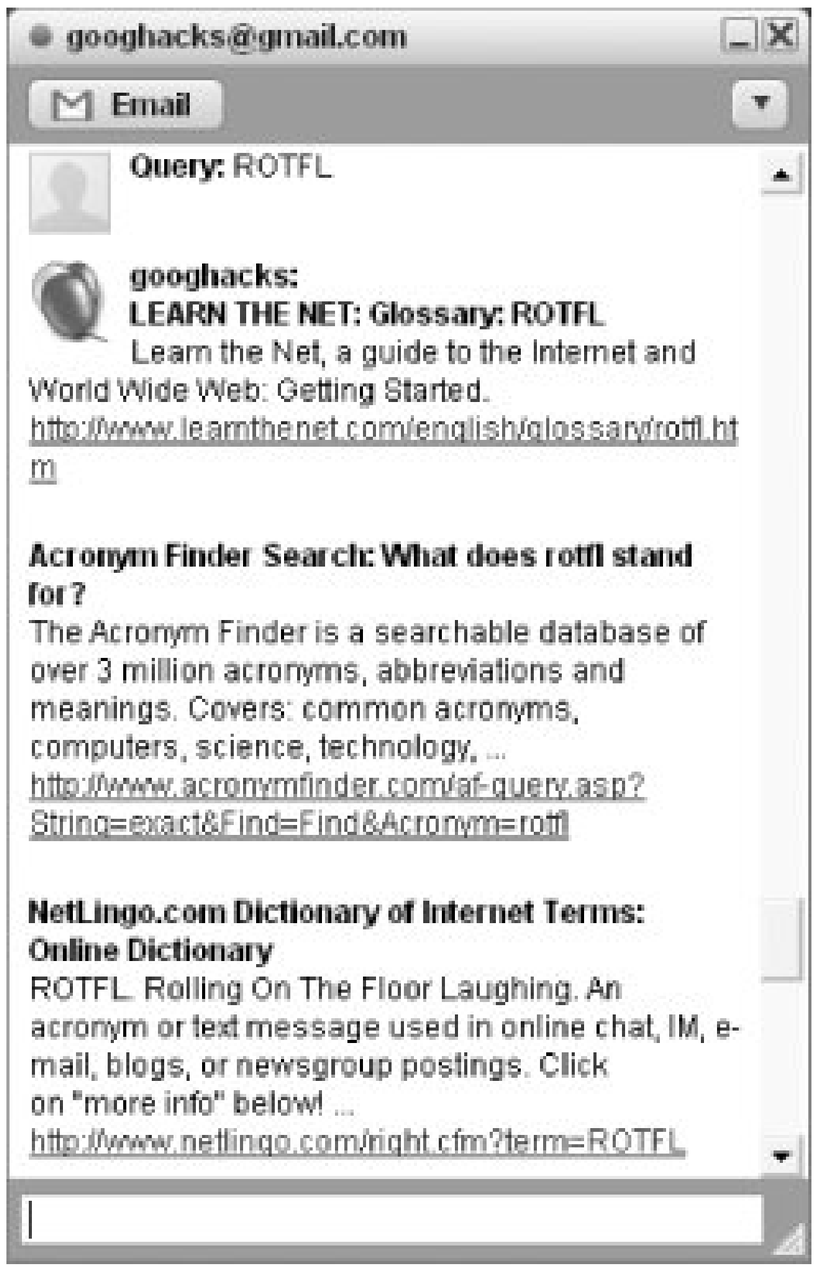
```
[[ Bot is Online, ready for queries! ]]
```

At this point, log into Google Talk as yourself and send a simple message to your bot. Back in the command window, you'll see the incoming query and the user who sent it:

```
>>> query: ROTFL
>>> from: user@example.com/Talk.v9222832159
```

At this point, the script takes the incoming message and queries the Google API for search results. As the results come in, they're sent back to the user, as shown in Figure 4-22.

Figure 4-22. QueryBot responding to the message "ROTFL"



The bot sends back the first 10 results for the query as individual messages.

When you're ready to put your bot to sleep, type Ctrl-C in the command window to take your bot offline.

See Also

- For a fully functioning Google Talk bot that can conference several users together and perform more complex commands, take a look at Google Talk: Conference Bot (http://coders.meta.net.nz/~perry/jabber/confbot.php) by Perry Lorier and Limodouthe inspiration for this hack. The code is freely available, and if you're familiar with Python you can

customize the bot for your own purposes.

- "Google from IRC" [Hack #50].

PREV

Hack 53. Googlify Your Browser

The Google Toolbar and a handful of other extensions can make Google a part of your web browser.

If you already use the Quick Search box in Firefox [Hack #55], you know the value of having instant access to Google Searches whenever you browse the Web. TheGoogle Toolbar (http://toolbar.google.com) gives you several options beyond web searching and provides one-click access to several Google features that interact with the current page you're browsing, from translating the page to posting information from the page to a blog.

Unlike the Quick Search box, you need to take some time to install the Google Toolbar, but you'll be up and running in just a few minutes. Once installed, the toolbar is a part of your browser, as shown in Figure 4-23.

Figure 4-23. The Google Toolbar in Firefox

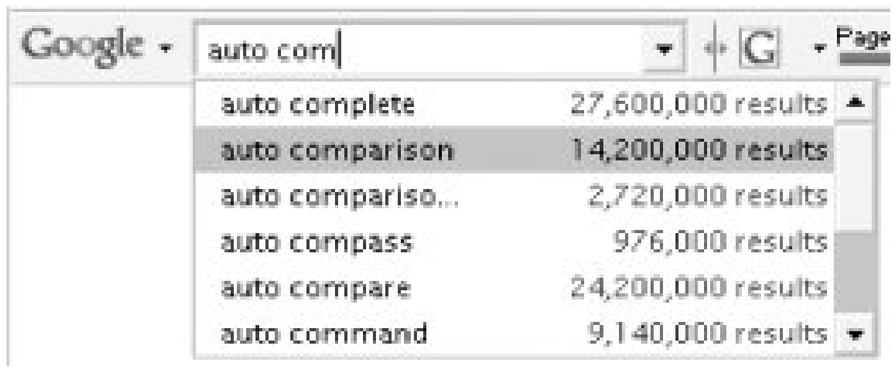
Features

Google doesn't provide access to all of its features via the toolbar, but there are some handy shortcuts that can give you more information about the page you're currently reading. You can add or remove features from the Google Toolbar by clicking the Google logo on the far left of the toolbar and choosing Options.

Web search

The most prominent feature of the Google Toolbar is the blank search form. As you type a word or phrase into the form, Google tries to guess what word you're after and offers a few suggestions along with estimated result counts in a drop-down box, as shown inFigure 4-24.

Figure 4-24. Google Toolbar autocompleting suggestions as you type



You can click a suggestion or continue typing to search for a particular phrase. Click Enter to leave the current page and view search results, or click the down arrow to choose from other Google searches such as Google Image, Google Groups, or the current site you're reading.

Firefox users can add autocomplete suggestions to the Quick Search box with an official extension by Google called Google Suggest for Firefox (http://www.google.com/tools/firefox/suggest/index.html). This is a handy option if you don't want to install the full Google Toolbar.

Another handy way to use the toolbar search is to highlight a word or phrase on the page, click and drag the text to the form, and then "drop" the text by releasing the mouse button. You're instantly taken to a search results page for the highlighted phrase.

PageRank

Google assigns every site in its index a popularity value from 0 to 10 called aPageRank, and the Google Toolbar is one of the only ways to find the numeric score for any particular site. As you browse pages, the toolbar contacts Google with the URL and displays a green graph in the toolbar with the corresponding PageRank. You can place your cursor over the PageRank graph to see the numeric score, as shown in Figure 4-25.

Figure 4-25. Viewing the PageRank of the current page with Google Toolbar

A site with a higher PageRank score means that Google believes the site has a higher authority and displays the site higher in its search results. Using the Google Toolbar, you can also use the PageRank score to help judge the authority of a particular source.

If you want to see the PageRank value of every page you visit but don't want to install the Google Toolbar, try the pagerankstatus extension for Firefox (http://pagerankstatus.mozdev.org). Once installed, you'll see the green PageRank indicator in your browser's lower status bar. Keep in mind that this extension isn't supported by Google in any way, and you'll be sending each site you visit to a third party.

Blog This!

If you publish a blog with Google's free tool Blogger (http://www.blogger.com), the Google Toolbar offers a quick way to quote other web sites. Click the orange B button (also known as Blog This!) on the toolbar to bring up a new window with a form for composing a new blog post. The text area includes the HTML necessary to display the title of the page you're viewing, linked to the page URL. I you highlight some text on the page before you clickBlog This!, as shown in Figure 4-26, the text is automatically quoted in the new entry as well.

Figure 4-26. Quoting a web site with the Blog This! feature of the Google Toolbar

Blog This! takes the work out of linking to interesting bits of information you find on the Web. It's a great way to start your own "web clippings file" to share with others.

If you want only the Blog This! feature and don't need the rest of the Google Toolbar, you can install a bookmarklet that functions exactly the same way by visiting the Blogger help page for Blog This! (http://help.blogger.com/bin/answer.py?answer=152).

Page information

The blue i button on the toolbar provides quick shortcuts to extended information about the page you're viewing at Google. Click the button to choose one of four options shown in Figure 4-27.

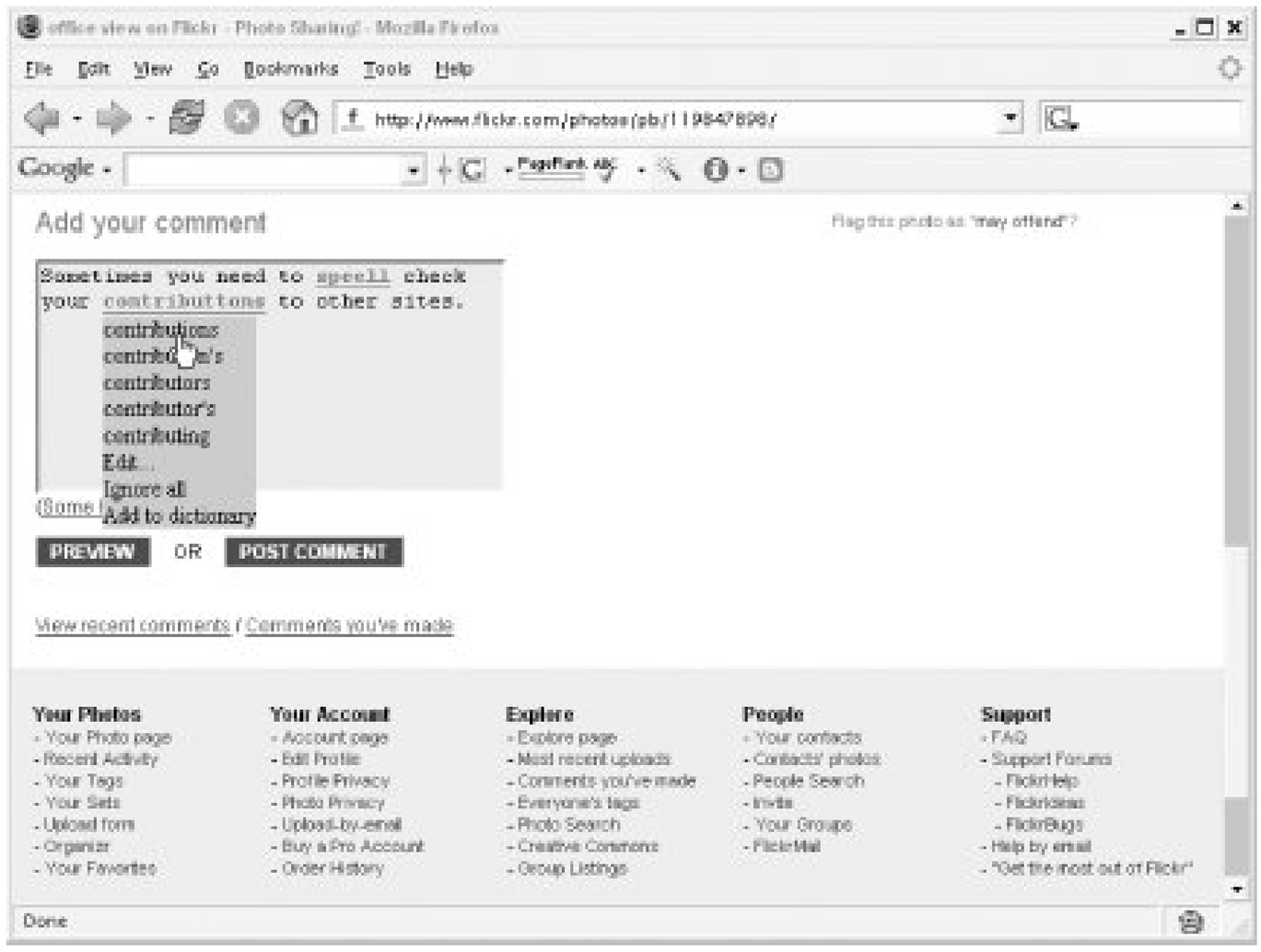
Figure 4-27. Finding extended information about the current page with Google Toolbar

Cached Snapshot of Page shows you the latest version of the page in Google's cache, if available; the Similar Pages link uses the `related:` syntax to show links to pages Google has determined are similar to the current page; Backward Links uses the `link:` syntax to find other pages that link to the current page; and, as you'd expect, the Translate Page into English link uses Google's Language Tools (http://www.google.com/language_tools) to translate the current page. You can change your default translation language in the toolbar options.

Spellchecking

Another useful feature of the toolbar is the ability to check your spelling on any web form. If you contribute to a number of different web sites, you know that not all of them provide spellchecks, and you often have to bring up another program, such as a word processor or email program, to check your text before you send it off. Instead, you can click the ABC button on the Google Toolbar, which highlights any misspelled words, as shown in Figure 4-28.

Figure 4-28. Checking your spelling in any web form with the Google Toolbar



Click a highlighted word to see a list of suggestions and click a suggestion to make the change. Click any empty space in the form to stop the spellchecker.

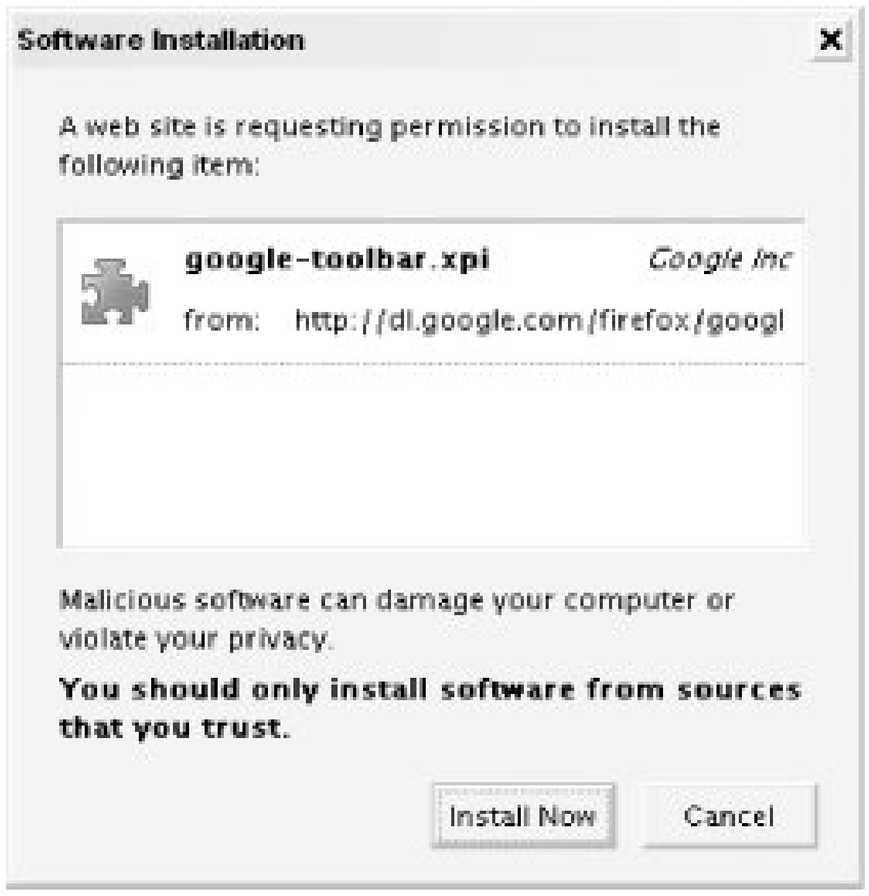
> This list is only a sampling of the tools available with the Google Toolbar, and the best way to get to know the features is to play around with them.

Installation

To install the Google Toolbar, point your browser to http://toolbar.google.com and click the blue Download Google Toolbar button. From there, you need to read through the Terms and Conditions and click Agree and Install to start the installation.

At the time of this writing, the toolbar is available for Internet Explorer on Windows and Firefox on Windows, Mac OS X, and Linux. Because the program is a browser extension rather than a traditiona application, the download and installation happen within the browser window. You need to approve some security requests along the way. Figure 4-29 shows the standard extension installation dialog for Firefox.

Figure 4-29. Installing the Google Toolbar in Firefox

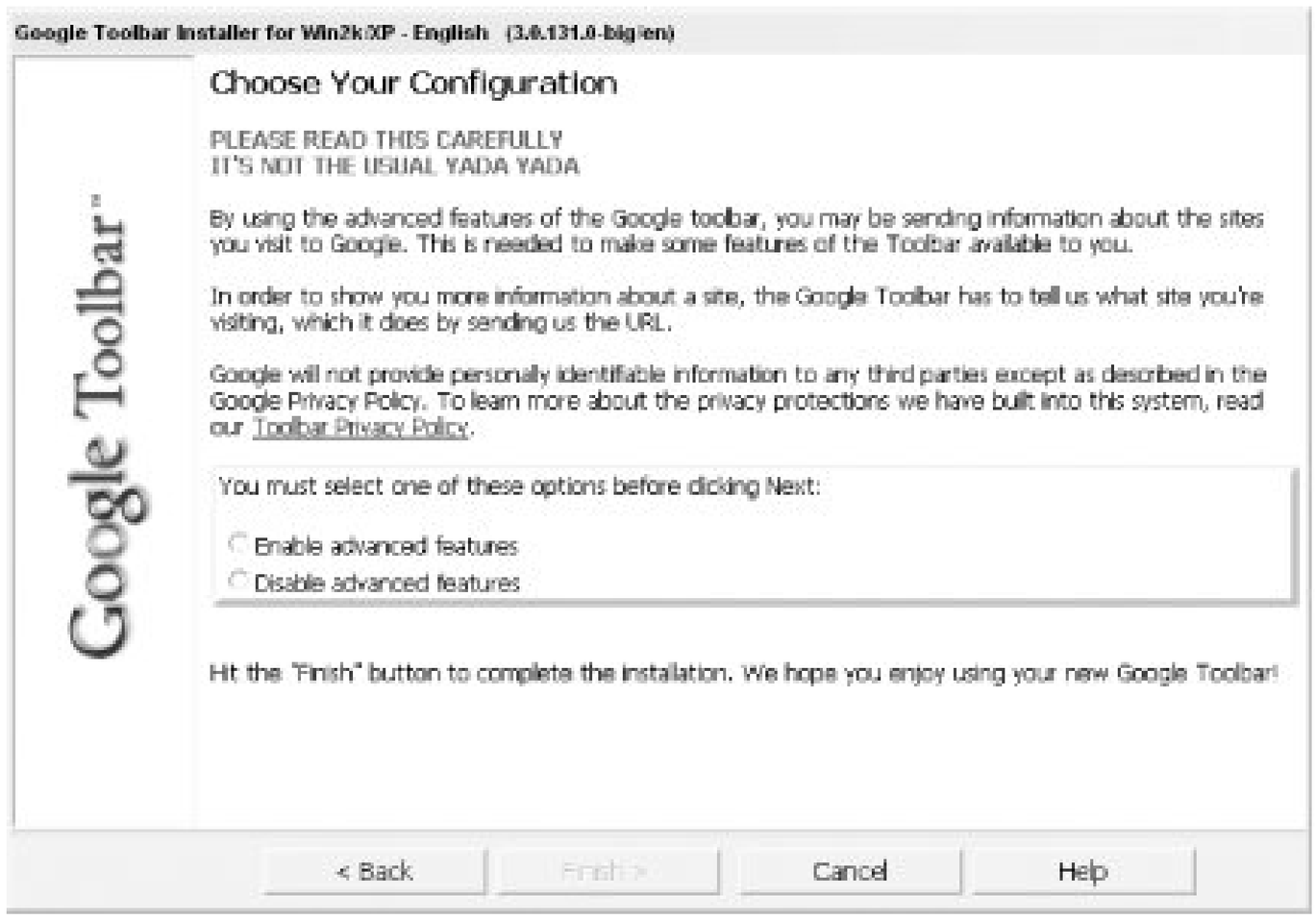


Click Install Now to download and install the toolbar, and then restart Firefox to start using the toolbar. The process for installing the toolbar in Internet Explorer is similar, but you'll see something like Figure 4-30.

Figure 4-30. Internet Explorer security warning for the Google Toolbar

Click Run to start the Google Toolbar installation. During the Internet Explorer installation process, you can choose to enable or disable features that "phone home" to Google with your browsing activities. Figure 4-31 shows the privacy notice from the installation process.

Figure 4-31. Internet Explorer privacy warning from the Google Toolbar installation



Some features require that the toolbar contact Google to get extended information about pages you're visiting. If you're uncomfortable with the idea that Google has access to every page you're visiting, you might want to disable the advanced features during installation. Keep in mind that you can always disable the advanced features at any point after installation.

> To get a better sense of how the Google Toolbar affects your browsing privacy, read the answers to the privacy questions at the Google Toolbar site (http://www.google.com/support/toolbar/bin/topic.py?topic=938).

Once you've made it through the security gauntlet, restart your browser; the Google Toolbar is waiting for you just below your main browser controls.

Privacy

As mentioned earlier, many of the features of the Google Toolbar require sending information to Google's servers to function. If you want to use to toolbar but aren't comfortable with sending every page you visit to Google's servers, you can disable the features that "phone home" to Google.

Click the Google logo on the far left of the toolbar and choose Options. From the Browse tab, unchecl PageRank Display, SpellCheck, WordTranslator, and AutoLink, and click OK to disable the features. Under the Search tab, uncheck "Suggest popular queries as you type" and click OK. Also, keep in mind that as Google adds features to the toolbar, you might need to disable features that contact Google's servers.

Remember that the Google Toolbar keeps a list of every query you type so you can access them quickly later. To clear your query cache at any time, click the Google logo and choose Clear Search History. You can also set the toolbar to forget your saved queries at the end of a session by

unchecking "Save the search history..." under the Search tab in the toolbar options.

Finally, if you want to remove the toolbar completely, you can quickly remove it by clicking the Google logo and choosing Help→Uninstall. A new window pops up, asking you to confirm the removal and provide some optional info about why you're removing the toolbar. Click Uninstall the Google Toolbar, restart your browser, and the Google Toolbar will be history.

Though you might give up a bit of privacy if you use the toolbar, you get quick access to some useful Google features in exchange, so you'll have to weigh the pros and cons of the toolbar for yourself.

PREV

Hack 54. Search with Google from Any Web Page

Searching the Web can be as simple as highlighting a term and clicking your mouse.

Imagine you're reading your favorite blog and the author starts rambling on about retro video games. Before you know it, you're knee-deep in gaming jargon as she compares her*SNES Emulator* to *MAME* or discusses her favorite *ROMs* and *Mods*. If such terms are Greek to you and you want to find out what they mean, wouldn't it be great to just highlight the term, click a button on your mouse, and have the answer? This type of context-menu search is easy to set up, if it isn't set up in your browser already.

If you use the Firefox browser (http://www.mozilla.com/firefox/), you're in luck, because a context search is built right in. Simply highlight any term on a web page, right-click with your mouse (Ctrl-click on a Mac), and click the "Search Web for..." option shown inFigure 4-32.

Figure 4-32. Firefox context search

Firefox opens a new tab with the Google Search results page for the word or phrase you highlighted.

Microsoft Internet Explorer (http://www.microsoft.com/windows/ie/) users don't have things quite so easy because there's no built-in context search. But you can get your own context search up and running in a few minutes with a bit of JavaScript and a new Registry entry.

The Code

This code handles the work of taking a highlighted term and opening a new browser window with a properly formatted Google URL. Open a text editor such as Notepad and create a new file called *GoogleSearch.html* with the following code:

```
<script language="JavaScript">
var searchURL = new String("http://www.google.com/search?q=");

var w = window.external.menuArguments;
var d = w.document;
var s = d.selection;
var r = s.createRange(  );
var term = new String(r.text);

window.open(searchURL + term);
</script>
```

Save this file on your computer in a memorable spot or create a new folder for it, such as *c:\\scripts*. Jot down the full path to this new file and open a blank file in Notepad again. This new file adds some information to your Windows Registry to let Internet Explorer know where to find *GoogleSearch.html* and when to execute it.

Add the following code and save the new file as*GoogleContext.reg*:

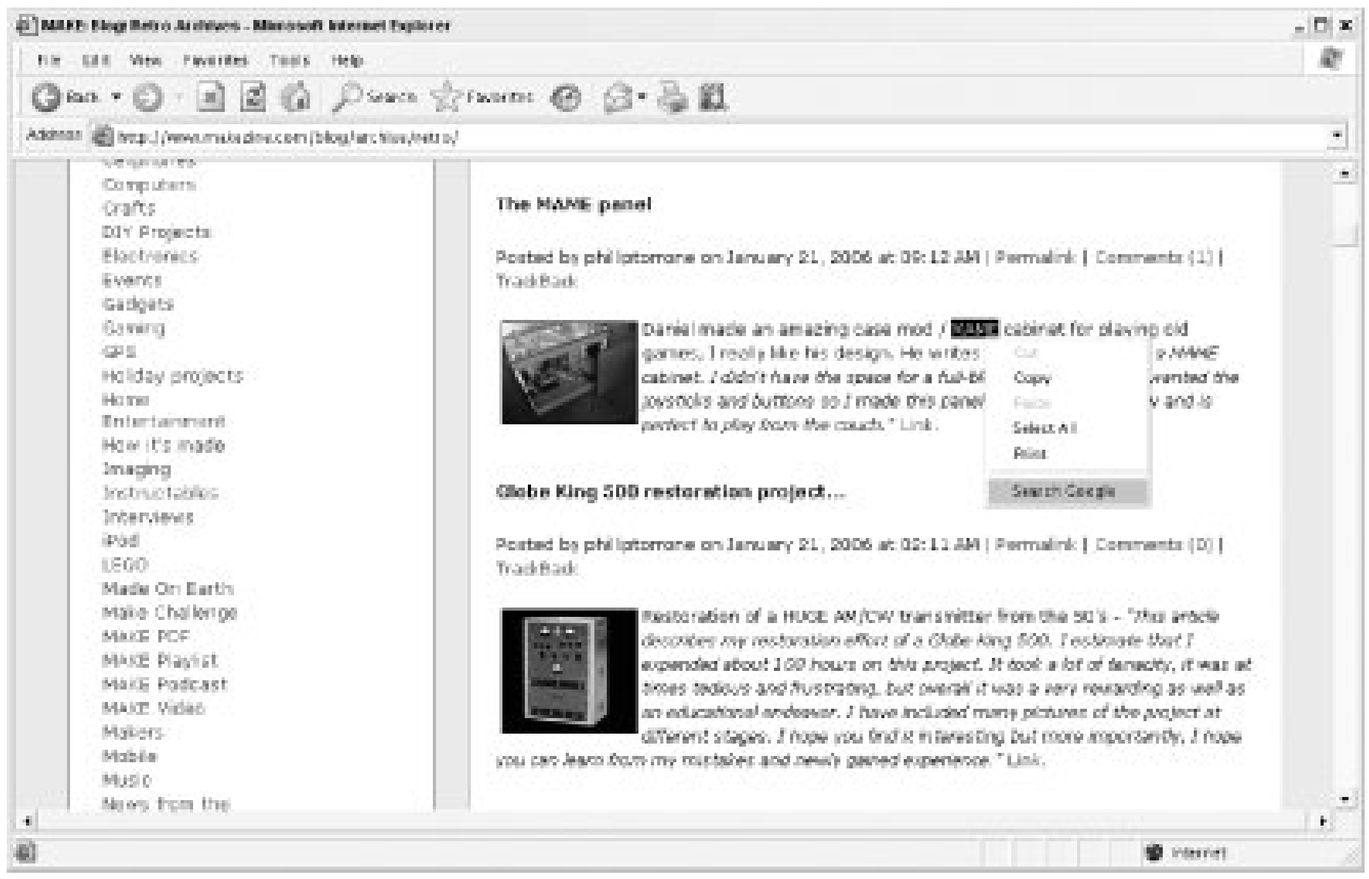
```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\\Software\\Microsoft\\Internet Explorer\\MenuExt\\Search Google]
@="c:\\\\scripts\\\\GoogleSearch.html"
"contexts"=dwords:00000010
```

Now double-click the file and confirm that you want to add the Registry information. You've just added a right-click menu entry called Search Google that will appear whenever you right-click on highlighted text within Internet Explorer. When you right-click and select the Search Google option, the JavaScript file you created earlier executes.

Running the Hack

Close any open Internet Explorer windows and then restart the browser. You should be able to highlight any text on a page and see the new context-menu entry shown inFigure 4-33.

Figure 4-33. Search Google context-menu entry



Choosing Search Google opens a new window such as the one inFigure 4-34, displaying search results for the selected term.

Figure 4-34. Google Search results window

With just a few minutes' coding, you'll have streamlined access to Google, giving you the knowledge that *MAME* stands for Multiple Arcade Machine Emulator and a number of links to follow for more information.

PREV

PREV

Hack 55. Customize the Firefox Quick Search Box

Though Google Web Search is the default option in the Firefox search box, with some quick c

If you use the Firefox web browser (available at http://www.mozilla.org/products/firefox/), you're proba
box and press Enter, and the search page comes up in the browser. Though Google is the default search

Figure 4-35. Th

The nice thing about this list of potential search engines is that you can add any search engine of your c
The New Search Engines section of the Mozilla site (the technology behind Firefox) contains the page sh
Firefox search box.

Figure 4-36. List of Google-

These are searches that others have found useful and decided to share with the larger Mozilla communit
(http://scholar.google.com) or Google Blogsearch (http://blogsearch.google.com), to searching Google
the Wikipedia with Google plug-in that searches the popular online encyclopedia using Google.

Keep in mind that all the plug-ins at Mozilla are contributed by users, and some m
in was tested and was found to be working. A blue question mark indicates a plug

To add a search engine from this Mozilla page, simply click the name of the search engine you'd like to a
scenes, Firefox has copied a small *.src* file and icon to the *searchplugins* directory of the Firefox installati

If you don't find the search of your dreams at the Mozilla page, it's fairly easy to build your own specialt
and an eye for spotting patterns in search URLs.

The Code

Imagine you find yourself frequently looking for academic papers on various subjects. You could use the
relevant papers. But tweaking the Advanced Search Form every time you want to run that particular sea

The first step in creating a custom entry is to perform a search and take a look at the URL. For this exar
change the file format to Adobe Acrobat PDF (*.pdf*), set the domain to `.edu` , and click Google Search. Y

Now take a closer look at the URL. The relevant pieces of the URL include the google.com domain, these

```
http://www.google.com/search?as_q=aerodynamics&num=10&hl=en&btnG=Google+Search&as_epq=&
```

This is where your skills at assembling Advanced Search URLs [Hack #17]come in handy. Note the impo

Knowing how Google Advanced Search URLs are constructed, you can write the file that tells Firefox whe

```
# Google PDF Search across .edu domains
#
# Created January 26, 2006

<SEARCH
    version="7.1"
    name="Google EDU Search"
    description="Search for PDFs across EDU domains"
    method="GET"
    action="http://www.google.com/search" >

<input name="as_filetype" value="pdf">
<input name="as_sitesearch" value=".edu">
<input name="as_q" user>

</search>
```

As you can see, this quick file begins with an opening`<SEARCH>` tag that holds the name of the search an tags let Firefox know which variable/value pairs should be included in the query. The`user` designation in

Running the Hack

Save the file and add it to the Firefox *searchplugins* directory, usually located at *C:\\Program Files\\Mozi* an icon for the search, and because Firefox comes with a Google Search option, you can simply copy the , in this example.

Once you restart Firefox, you'll find a new option in the search list called Google EDU Search. Choose th results such as the one shown in Figure 4-37.

Figure 4-37. Google Adv

If you find yourself using a particular Google search time and again, you might be able to speed up your

Hack 56. Build a Google Screensaver

With a bit of Perl and the built-in screensavers available in Mac OS X or Windows XP, you can create your own screensaver that shows pictures from Google Images.

Along with desktop backgrounds, screensavers have always been a feature of personal computers that people feel comfortable changing, tweaking, fiddling with, and hacking for fun. And by scripting Google Images, you can create a screensaver based on images from across the Web.

This hack relies on the screensavers that ship with Windows XP and Mac OS X. Each screensaver lets you specify a directory on your computer that contains images, and displays those images on your screen during your computer's idle moments. A Perl script downloads images from a Google Images search that you specify.

The Code

This code works on both Windows XP and Mac OS X systems, but you'll need a Perl component that isn't installed by default. The `WWW::Google::Images` module (http://search.cpan.org/dist/WWW-Google-Images/lib/WWW/Google/Images.pm) handles all of the hard work of gathering images from a Google Images search and saving a copy on your computer.

Copy the code to a file called *goosaver.pl* and put the file in a local folder path where the images will be stored. On Windows XP, you should specify a drive and folder, such as *C:\\goosaver*. On Mac OS X, you should specify a full path, such as */Users/pb/Photos/goosaver*.

The following code contacts Google Images with your query and downloads matching photos:

```
#!/usr/bin/perl
# goosaver.pl
# Downloads images from a Google Image
# search for a screensaver.

use strict;
use WWW::Google::Images;

# Take the query from the command line.
my $query = join(' ',@ARGV) or die "Usage: perl spell.pl <query>\\n";

# Create a new WWW::Google::Images instance.
my $agent = WWW::Google::Images->new(
            server => 'images.google.com');
```

```
# Query Google Images.
my $result = $agent->search($query . " inurl:wallpaper",
                            limit => 25,
                            iregex => 'jpg'
);

# Save each image in the result locally, with
# the format [query][count].[extension].
my $count;
while (my $image = $result->next(  )) {
    $count++;
    print $image->content_url(  ) . "\\n";
    print $image->save_content(base => $query . $count) . "\\n\\n";
}
```

Note that although the query is set on the command line, this script adds the`inurl:wallpaper` keyword to the query. This means Google Images will return only images that have the word *wallpaper* in the URL, taking advantage of how people naturally organize their files online. If you don't get good results with this addition, simply remove this bit from the script or try other options that people might use to organize images, such as`inurl:large` or `inurl:desktop`.

Also note that the `iregex => 'jpg'` option limits saved results to files that are JPEGs. If you want more varied file types to be returned, remove this line, but keep in mind that the system screensavers typically prefer JPEGs.

Running the Hack

How to run the script depends on which operating system you're using.

On Mac OS X

To set up your screensaver on Mac OS X, first create your Google screensaver photo folder. It can be anywhere, but your *Pictures* directory is a memorable place. Call the new folder*goosaver*.

To get the ball rolling, open a Terminal window (Applications Utilities Terminal), change to the *goosaver* directory (using the `cd` command), and run the script from the command line, like this:

```
% perl goosaver.pl

                        insert query
```

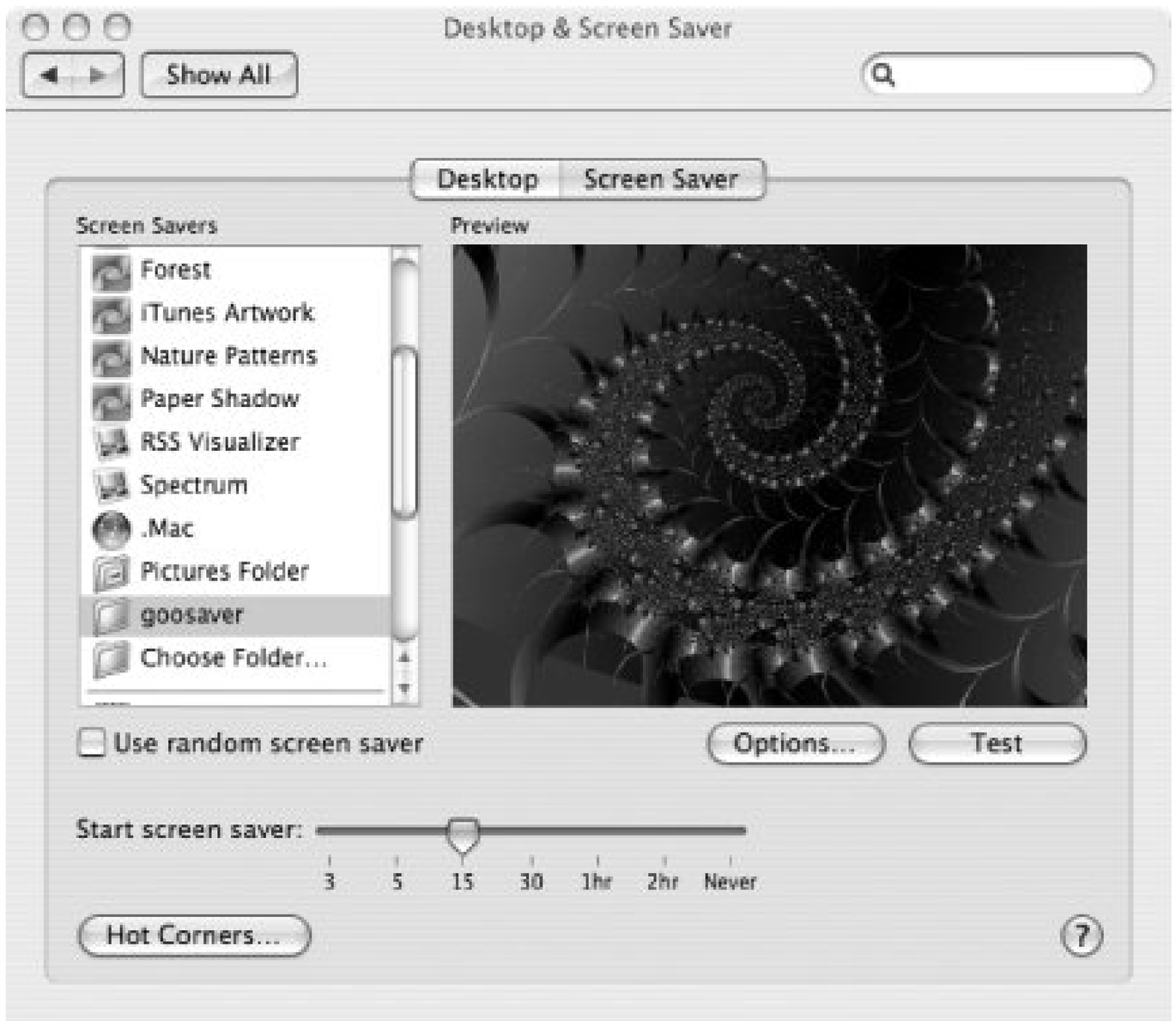
For example, if you want a screensaver with those mathematical visualizations called*fractals*, call the script like so:

```
% perl goosaver.pl fractal
```

This downloads and adds several fractal-related photos to your *goosaver* folder.

You can now set up your screensaver. Select System Preferences from the Apple menu and choose Desktop & Screen Saver. Click the Screen Saver button, and then click the Choose Folder... option. Select your *goosaver* folder in your *Pictures* directory, and you should see the pictures you've just downloaded in the preview window, as shown in Figure 4-38.

Figure 4-38. Setting a Mac screensaver folder



Your screensaver is now set to display the photos you downloaded from Google Images when the screensaver is activated.

On Windows XP

The process for setting up the screensaver on Windows is almost identical to the Mac OS X version. Unlike the Mac, however, Windows XP does not come with Perl installed, so you might need to do a bit more work to get started. If you want to run Perl on Windows, you can download the free ActivePerl for Windows by ActiveState from http://www.activestate.com/Products/ActivePerl/.

Don't forget that you'll also need to install the `WWW::Google::Images` module before you can use the script.

Once Perl is installed, create your new screensaver folder somewhere in your filesystem; *C:\\goosaver* is a good place. Run *goosaver.pl* from the command line (Start → Programs → Accessories → Command Prompt) to download some photos:

```
% perl goosaver.pl

                    insert query
```

For something specific, such as landscape photos, the command would look like this:

```
% perl goosaver.pl landscape
```

Now that some photos exist in the folder, set your screensaver by right-clicking any empty space on your desktop and choosing Properties. Choose the Screen Saver tab and select My Picture Slideshow from the list of screensavers. Click the Settings button, and you should see the options shown in Figure 4-39.

Figure 4-39. Windows XP screensaver options

Under the "Use pictures in this folder" heading, click Browse and choose the Google screensaver folder you created. Click OK, and your screensaver now shows the photos you just downloaded from Google Images.

It takes a bit of work on both systems to set up a custom Google Images screensaver, but you're rewarded with unexpected images from across the Web.

Hack 57. Add a Feed to Google Quickly

Speed up the time it takes to add RSS feeds to your Google Homepage or Google Reader.

Adding a news feed to either Google Homepage (http://www.google.com/ig) or Google Reader (http://reader.google.com) isn't a complex process, but it does involve some copying, pasting, clicking, and generally breaking out of the flow of reading a site to make it happen.

With a bit of browser hacking, you can reduce the friction of adding sites to Google. Because Internet Explorer and Firefox are quite different, adding feeds quickly requires different approaches in each browser.

Internet Explorer

One way to add shortcuts to Internet Explorer is through a custom *context menu*. A context menu is the menu that pops up when you right-click (or Ctrl-click on a Mac) an element on a web page. The *context* part of its name refers to the fact that different choices appear in different situations. For example, when you right-click a link, you're presented with the options Open Link in New Window, Copy Link Location, Bookmark This Link, and others. In another context, such as when clicking an image or clicking highlighted text, you have different choices in the menu.

If you've been reading personal blogs for a while, you've probably seen many variations of the white-on-orange buttons that indicate a link to an RSS feed, and if you haven't, you can find some examples [Hack #39] in this book.

Wouldn't it be great if you could right-click one of these buttons and have the option to add to Google Homepage or Google Reader? This would save quite a few steps, and you wouldn't have to break from the site you're currently reading to add the feed. Similar to quick searching in Internet Explorer [Hack #54], this hack shows how to add a custom context menu item for adding feeds to Google.

The code

Much like a bookmarklet [Hack #43], any JavaScript that runs via a context-menu entry has access to the page currently loaded in the browser. This means that when you click the context-menu entry you've added, the browser executes a script that performs a particular function using information from the current page.

In this case, it grabs the URL linked from the currently clicked image, constructs a special Google URL that includes the feed URL, and opens the new URL in a new browser window.

Add the following code to a file called *AddToGoogle.html*:

```
<script language="JavaScript">
var addURL = new String("http://fusion.google.com/add?feedurl=");

var w = window.external.menuArguments
var url = w.event.srcElement.parentElement.href;

window.open(addURL + url,null,
     "height=455,width=788,status=yes,scrollbars=yes,resizable=yes");
</script>
```

The `external.menuArguments` object holds information about the current document, and the `event.srcElement` is the document item the user clicked. Grabbing the `href` attribute of the element's parent gives you the link URL around the image tag. Save the file in a spot you'll remember. For simplicity in this hack, save it to a directory called *c:\\scripts*.

Now that the script is ready to go, you just need to add the context-menu entry to Internet Explorer and tell it to run this particular script when you click the entry. You can do this through the Windows Registry. The Registry is a system database that holds information about applications, including Internet Explorer. You can safely make additions to the Registry via *.reg* files. Create a new text file called *AddGoogleContext.reg* and add the following code:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\\Software\\Microsoft\\Internet Explorer\\MenuExt\\Add to Google]

@="c:\\\\scripts\\\\AddToGoogle.html"
"contexts"=dword:00000002
```

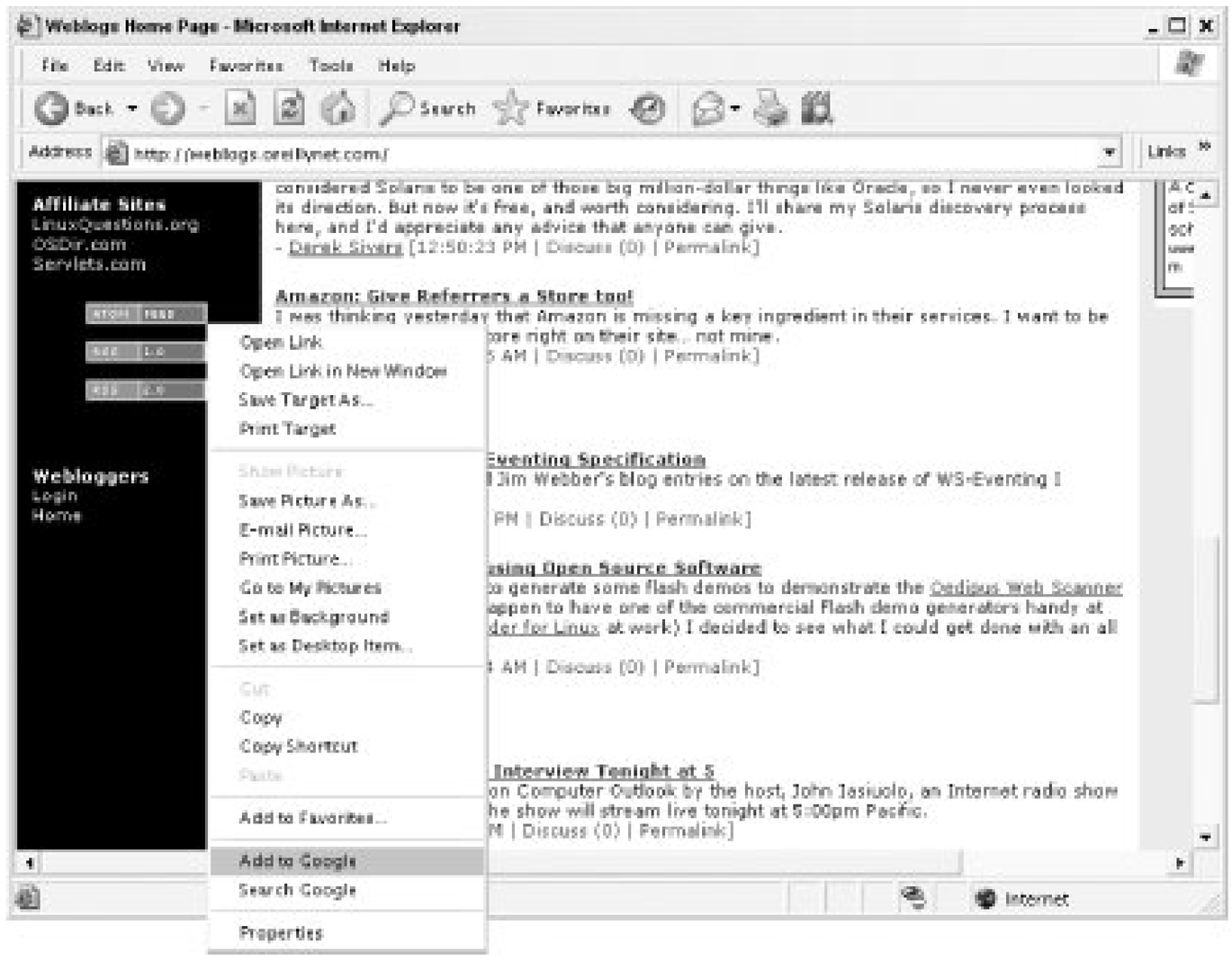
Note that the `contexts` entry ends with `2`, which means the entry will appear only when the user clicks an image. Other values you can use here include `1` (for anywhere), `20` (for text links), or `10` (for text selections).

Save the file, double-click, and confirm that you want to add the new Registry information. You now have a right-click menu entry called Add to Google that appears whenever you right-click an image.

Running the hack

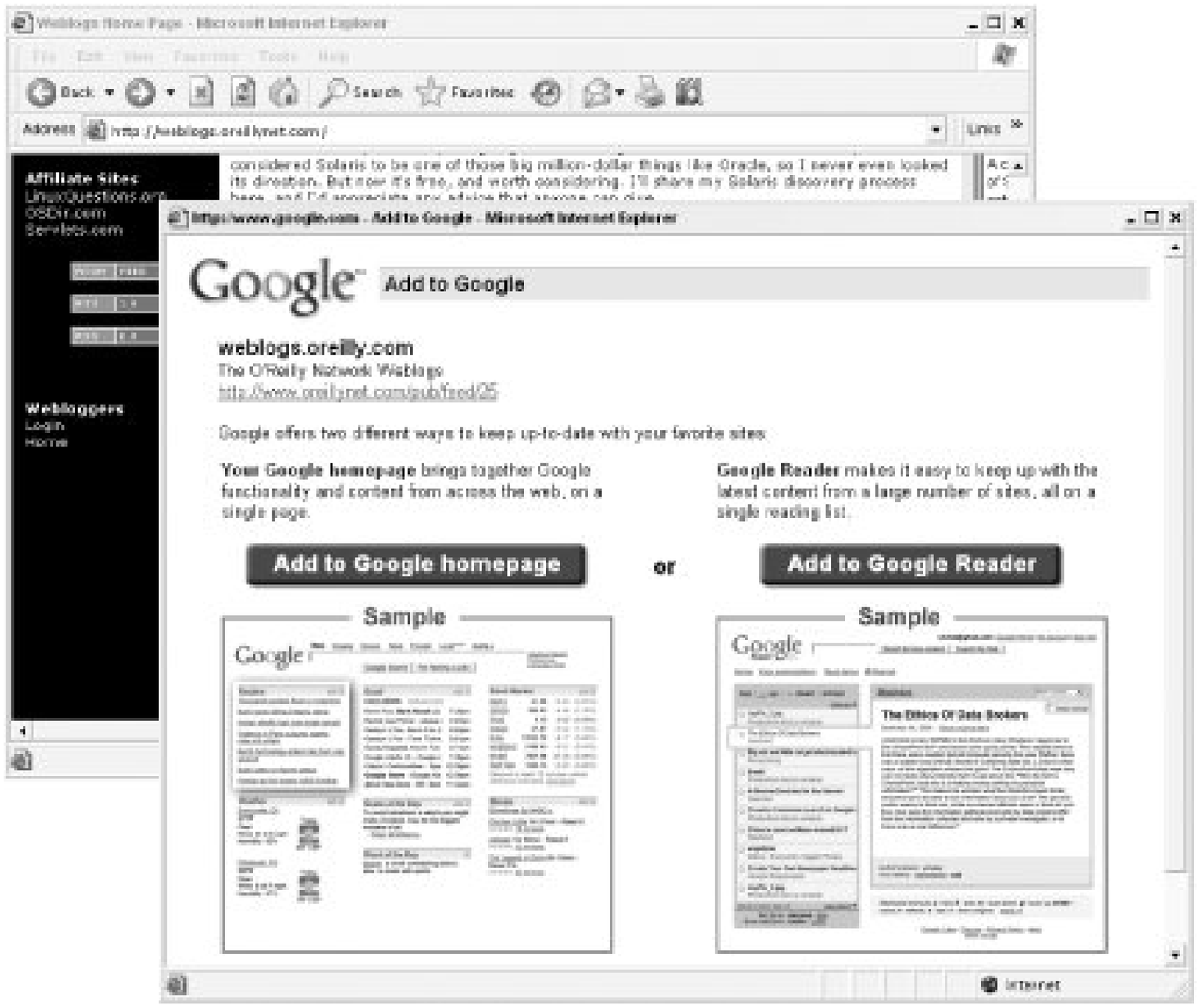
Once the code and Registry settings are in place, restart Internet Explorer. Browse to a site with a feed URL link and take the new context-menu entry for a spin. When you right-click an image, you should see Add to Google, as shown in Figure 4-40.

Figure 4-40. Add to Google context-menu entry



When you click the Add to Google context-menu entry, the Add to Google page appears in a window, such as the one shown in Figure 4-41, where you can choose your preferred reader.

Figure 4-41. Add to Google page in a new window



Keep in mind that the Add to Google context-menu entry is available for*every* image on a web page, regardless of whether it links to an RSS feed. So, you must use your best judgment about when to use the feature. If the image turns out not to be linked to an RSS feed, Add to Google won't be able to show the feed title and description in the upper-right corner of the page, and you'll know you clicked a bad feed link.

Choose your preferred application, and then you can close the pop-up window and go back to reading the site.

Firefox

If you use Firefox, you're probably well aware of the orange Live Bookmark icon that appears at the right end of the address bar when the site you're visiting has an available XML feed. This icon indicates that the site author has embedded a bit of code into the page to let applications know where her XML feed is located [Hack #39].

Normally, you can click the icon to add a Live Bookmark that tracks recent entries to the site in your browser's bookmarks. Michael Koziarski has built an extension for Firefox calledFeed Your Reader that changes the Live Bookmark feature to use the orange icon for adding feeds to your favorite readerincluding Google's offeringsinstead of to your browser's bookmarks.

Browse to the extension page (http://projects.koziarski.net/fyr/), install the extension directly in the page, and restart Firefox. Choose Tools Extensions from the top menu, highlight Feed Your Reader, and click Options. Choose Google Reader from the list of options in the drop-down menu, as shown in Figure 4-42.

Figure 4-42. The newsreader options in Feed Your Reader

Click OK, and the extension is set to go. Browse to any site with an embedded XML feed (such as http://weblogs.oreillynet.com) and click the orange Live Bookmark icon in the address bar. Instead of adding a Live Bookmark, the Add to Google page opens in a new tab in your browser, where you can choose your preferred reader.

Though the two browsers require slightly different approaches, both can be extended to help you add feeds to Google more quickly, saving you the hassle of opening new windows and cutting and pasting URLs.

Hack 58. Tame Long Google URLs

With an eye for URLs and the right tools, you can shorten long Google URLs when you need tc instant message, or on paper.

Most of the time, we're all surfing the Web in virtual isolation. It's just you and the computer, and the la length of a URL at a page you're visiting. But as soon as you want to share the piece of the Web you're v length of a URL becomes important.

Because email programs wrap text at 72 characters for easy reading, any URL that's longer could be brc person on the other end of the message won't be able to see the page you've sentor he'll have to spend URL together in Notepad. And imagine trying to hand-write a note to someone that includes some of the

Trimming Google URLs

Google has a lot of great content to share with others, but some of the URLs are definitely too long to se using the Quick Search box in the Firefox browser [Hack #55]to search for the term `brevity` on Google y

```
http://www.google.com/search?q=brevity&start=0&ie=utf-8&oe=utf-8&client=firefox-a&rls=or
```

Those 112 characters are definitely past the 72-character safe zone. If you look at the URL, you can see contain the relevant information. The characters `?q=brevity` look important, but the rest of the URL looks

> It's important to note that what looks like gibberish is actually useful information to Go to you when you're trying to share links, so you can cut it out.

Cutting out the garbage characters of the URL gives you something more manageable:

```
http://www.google.com/search?q=brevity
```

The 38 characters in this URL are well within the safe zone, and the URL points to exactly the same page Web Search URL without the *www* prefix automatically redirects to the same page, you can trim four mo

```
http://google.com/search?q=brevity
```

This `q=` pattern is repeated throughout Google's services, and you can often use this method to trim URL Search. Here are a few examples:

Service	URL pattern
Google Images	`http://images.google.com/images?q=`*`insert query`*
Google Groups	`http://groups.google.com/groups?q=`*`insert query`*
Google News	`http://news.google.com/news?q=`*`insert query`*
Froogle	`http://froogle.google.com/froogle?q=`*`insert query`*

When you're ready to share a URL, keep an eye out for ways to trim the URL down to size. But there wil
you have is a URL-trimming service.

URL-Trimming Services

The scourge of long URLs is so rampant on the Web that several free services have appeared to help you
long URLs with others. To see how these services can help, here's an example of a Google Maps URL tha
directions from San Francisco, Calif., to the O'Reilly offices in Sebastopol, Calif.:

```
http://maps.google.com/maps?f=d&hl=en&saddr=San+Francisco,+CA&daddr=1005+Gravenstein+Hw
```

As you can see, this 106-character URL is dense with information. There's nothing extraneous we can st
information at Google. This is where TinyURL.com can help. Copy any long URL you want to abbreviate a
the front page at http://tinyurl.com . Click the Make TinyURL! button, and the next page gives you an ab

```
http://tinyurl.com/oorj6
```

These 24 characters are well within the safe zone and definitely won't break in an email. Another service
produces the following URL:

```
http://shorl.com/dikafrekikuru
```

Each of these services stores the long URL on their servers, assigns the URL a random character string,
when someone visits the short address on their servers. Shorl.com even provides some usage statistics,
people have used the shortened URL.

There are some drawbacks to using these third-party services. The person you're sharing the link with w
actually going to visit. This might make for some fun practical jokes, but it's always better to be as direc
URLs with people. Also, the longevity of the link isn't guaranteed. If TinyURL orShorl.com goes out of bu

fail. Using a redirection service such as these isn't the best choice if you're going to print a URL in a boo use, these services are a good way to share long URLs without annoying the person on the other end.

PREV

PREV

Hack 59. Autocomplete Search Terms as You Type

Google can suggest your search terms before you even finish typing them.

It's true: Google is clairvoyant. It can guess what you're going to search for even before you've typed it. Well, maybe that overstates it. But it can certainly take an educated guess, based on the popularity and number of results of certain keywords.

This hack relies on the Greasemonkey Plugin (http://greasemonkey.mozdev.org/) for the Firefox web browser (http://www.mozilla.com/firefox/).

Don't believe me? Visit http://www.google.com/webhp?complete=1 and start typing, and Google will autocomplete your query after you've typed just a few characters. This is insanely cool, and virtually nobody knows about it. And even people "in the know" need to visit a special page to use it. This hack makes this functionality work everywhereeven on the Google home page (http://www.google.com).

The Code

This user script runs on all Google pages, but it works only on pages with a search form. Of course, being Google, this is most pages, including the home page and web search result pages.

This hack doesn't do any of the autocompletion work itself. It relies entirely on Google's own functionality for suggesting completions for partial search terms, defined entirely in http://www.google.com/ac.js. All we need to do is create a `<script>` element pointing to Google's own code and insert it into the page. Then, we tell Google to activate it by adding another `<script>` element that calls Google's own `InstallAC` function.

Save the following user script as *google-autocomplete.user.js*:

```
// ==UserScript==
// @name            Google Autocomplete
// @namespace       http://diveintomark.org/projects/greasemonkey/
// @description     Autocomplete search keywords as you type
// @include         http://*.google.tld/*
// @exclude         http://*/*complete=1*
// ==/UserScript==

function getSearchBox(sFormName) {
```

```
    return document.forms.namedItem(sFormName);
}

function injectAC(sFormName) {
    var elmScript = document.createElement('script');
    elmScript.src = 'http://www.google.com/ac.js';
    document.body.appendChild(elmScript);
    var elmDriver = document.createElement('script');
    elmDriver.innerHTML = 'var elmForm = document.forms.namedItem("' +
    sFormName + '");\\n' +
    'InstallAC(elmForm, elmForm.elements.namedItem("q"),' +
    'elmForm.elements.namedItem("btnG"), "search", "en");';
    document.body.appendChild(elmDriver);
}

var sFormName = 'f';
var elmForm = getSearchBox(sFormName);
if (!elmForm) {
    sFormName = 'gs';
    elmForm = getSearchBox(sFormName);
}
if (!elmForm) { return; }
window.setTimeout(function(  ) { injectAC(sFormName); }, 100);
```

Running the Hack

After installing the user script (Tools Install This User Script), go to http://www.google.com and start typing the word `greasemonkey`. After typing the first three letters, `gre`, you will see a drop-down menu with possible completions, as shown in Figure 4-43.

Figure 4-43. Autocompletion of "gre" search

If you continue typing `greasemonkey` and then type a space, Google will suggest possible multiword searches, as shown in Figure 4-44.

Figure 4-44. Suggestions for multiword "greasemonkey" search

Mark Pilgrim

PREV

PREV

Hack 60. Refine Your Google Search

Google might already know what keywords you should add to your search to find exactly what you're looking for.

As described in "Autocomplete Search Terms as You Type" [Hack #59], you can visit http://www.google.com/webhp?complete=1 and start typing, and Google Suggest will autocomplete your query as you type. By itself, this is wickedly cool. Now, let's make it even cooler by integrating i into the main Google web search. Along with the usual search results, you'll see a list of related queries made up of additional keywords, so you can refine your search.

The Code

Google Suggest works by requesting a specially constructed URL with the characters you've typed so far. The request returns JavaScript code, and Google Suggest evaluates this code and adds the results to its autocomplete menu. If you type a complete keyword, followed by a space, Google Suggest will return a list of popular searches that include your keyword plus one or two other words.

For example, if you type `firefox`, Google Suggest constructs this URL:

```
http://www.google.com/complete/search?js=true&qu=firefox
```

Enter that URL in your location bar and you'll see Google's response:

```
sendRPCDone(frameElement, "firefox", new Array("firefox", "firefox download",
"firefox browser", "firefox extensions", "firefox plugins", "firefox mozilla",
"firefox themes", "firefox.com", "firefox web browser", "firefox 1.0"),
new Array("25,900,000 results", "8,000,000 results", "6,990,000 results",
"1,270,000 results", "1,250,000 results", "8,160,000 results",
"1,950,000 results", "1 result", "5,460,000 results", "6,540,000 results"),
new Array(""));
[end example]
```

In other words, Google is already doing the hard part: tracking billions of queries and ranking them by popularity. Compared to that, constructing the request and parsing the response is easy. You can mimic Google's autocomplete algorithm by constructing the URL yourself, calling `GM_xmlhttpRequest` and parsing the response.

Save the following user script as *refinesearch.user.js*:

```
// ==UserScript==
// @name           Refine Your Search
// @namespace      http://diveintomark.org/projects/greasemonkey/
// @description    adds a "refine your search" list on Google search results
// @include        http://www.google.tld/search*
// ==/UserScript==

function getCurrentSearchText(  ) {
    var elmForm = document.forms.namedItem('gs');
    if (!elmForm) { return; }
    var elmSearchBox = elmForm.elements.namedItem('q');
    if (!elmSearchBox) { return; }
    var usQuery = elmSearchBox.value;
    if (!usQuery) { return; }
    return usQuery;
}

function getFirstSearchResult(  ) {
    var results = document.evaluate("//p[@class='g']", document, null,
        XPathResult.ORDERED_NODE_SNAPSHOT_TYPE, null);
    return results.snapshotLength ? results.snapshotItem(0) : null;
}

function parseRefineYourSearchResults(oResponse) {
    if (oResponse.responseText.indexOf('new Array(') == -1) return;
    var arResults = oResponse.responseText.split(
        'new Array("')[1].split('")')[0].split('", "');
    var usQuery = getCurrentSearchText(  );
    var htmlArResults = new Array(  );
    for (var i = 0; i < arResults.length; i++) {
        if (!arResults[i] || (arResults[i] == usQuery)) continue;
        htmlArResults.push('<a href="http://www.google.com/search?q=' +
                           escape(arResults[i]) + '">' +
                           arResults[i] + '</a>');
    }
    if (!htmlArResults.length) return;
    var elmRefine = document.createElement('div');
    elmRefine.id = 'refineyoursearch';
    elmRefine.style.fontSize = 'small';
    elmRefine.style.paddingTop = elmRefine.style.paddingBottom = '1em';
    var html = 'Refine your search: ' + htmlArResults.join(' &middot; ');
    elmRefine.innerHTML = html;
    var elmFirstResult = getFirstSearchResult(  );
    elmFirstResult.parentNode.insertBefore(elmRefine, elmFirstResult);
}

var usQuery = getCurrentSearchText(  );
if (!usQuery) return;
if (!getFirstSearchResult(  )) return;
GM_xmlhttpRequest({
    method: "GET",
    url:    "http://www.google.com/complete/search?hl=en&js=true&qu=" +
```

```
            escape(usQuery + ' '),
    onload: parseRefineYourSearchResults
});
```

Running the Hack

After installing the user script from Tools → Install This User Script, go to http://www.google.com and search for `firefox`. Before the first search result, you'll see a list of related queries, as shown in Figure 4-45.

Figure 4-45. Google search for "firefox" with suggested refined searches

If you click on one of the suggested refined searches, such as `firefox plugins`, Google displays those search results, which include suggestions for even further refinements, as shown in Figure 4-46. Depending on your keywords, you might be able to *drill down* several levels, until Google finally runs out of suggestions.

Figure 4-46. Google search for "firefox plugins" with suggestions

Google Suggest works only on web searches, and only in English, so this hack inherits those limitations. You can read more about Google Suggest in Google's FAQ (http://labs.google.com/suggestfaq.html).

Mark Pilgrim

Hack 61. Make Google More Accessible for Low-Vision Users

Change Google's layout to make it easier for low-vision users to read.

As a class of disabilities, low-vision users are often ignored by accessibility experts. However, accessibility expert Joe Clark has recently published his research into the needs of web users with limited vision. He pioneered a technique known as the*zoom layout*: a special alternate style applied to a web page that specifically caters to low-vision users.

This hack relies on the Greasemonkey Plugin (http://greasemonkey.mozdev.org/) for the Firefox web browser (http://www.mozilla.com/firefox/).

As I was learning about zoom layouts, it occurred to me that this would be a perfect application of Greasemonkey. (Actually, that thought occurs to me a lot these days.) This hack is my first attempt at transforming a site into a zoom layout.

If you want, you can read more about zoom layouts at http://www.alistapart.com/articles/lowvision/ and http://joeclark.org/atmedia/atmedia-NOTES-2.html.

The Code

This user script runs on several specific Google pages:

- Google's home page at http://www.google.com.
- International versions of Google's home page, such ashttp://www.google.ca.
- Other variations of Google's home page, such ashttp://www.google.com/webhp and http://www.google.com/imghp. You can reach these by clicking one of the navigation links at the top of http://www.google.com.
- Web search results.
- Image search results.

This hack is written to be cross-browser compatible. It works in Firefox with Greasemonkey, in Internet Explorer 6 for Windows with Turnabout, and in Opera 8 with its built-in support for User JavaScript. You can download Turnabout at http://reifysoft.com/turnabout.php, and Opera at http://www.opera.com.

Save the following user script as *zoom-google.user.js*:

```
// ==UserScript==
// @name           Zoom Google
// @namespace      http://diveintomark.org/projects/greasemonkey/
// @description    make Google more accessible to low-vision users
// @include        http://www.google.tld/
// @include        http://www.google.tld/?*
// @include        http://www.google.tld/webhp*
// @include        http://www.google.tld/imghp*
// @include        http://www.google.tld/search*
// @include        http://images.google.tld/
// @include        http://images.google.tld/?*
// @include        http://images.google.tld/images*
// ==/UserScript==

function addGlobalStyle(css) {
    var elmHead, elmStyle;
    elmHead = document.getElementsByTagName('head')[0];
    elmStyle = document.createElement('style');
    elmStyle.type = 'text/css';
    elmHead.appendChild(elmStyle);
    elmStyle.innerHTML = css;
}

function getElementsByClassName(sTag, sClassName) {
    sClassName = sClassName.toLowerCase(  ) + ' ';
    var arElements = document.getElementsByTagName(sTag);
    var iMax = arElements.length;
    var arResults = new Array(  );
    for (var i = 0; i < iMax; i++) {
        var elm = arElements[i];
        var sThisClassName = elm.className;
        if (!sThisClassName) { continue; }
        sThisClassName = sThisClassName.toLowerCase(  ) + ' ';
        if (sThisClassName.indexOf(sClassName) != -1) {
            arResults.push(elm);
        }
    }
    return arResults;
}

function removeFontTags(  ) {
    // remove font tags
```

```
    var arFonts = document.getElementsByTagName('font');
    for (var i = arFonts.length - 1; i >= 0; i--) {
        var elmFont = arFonts[i];
        var elmSpan = document.createElement('span');
        elmSpan.innerHTML = elmFont.innerHTML;
        elmFont.parentNode.replaceChild(elmSpan, elmFont);
    }
}

function zoomStyle(  ) {
    addGlobalStyle('body { margin: 30px; } \\n' +
'body, td { font-size: large ! important; } \\n' +
'html>body, html>body td { font-size: x-large ! important; } \\n' +
'body, div, td { background: navy ! important; ' +
    'color: white ! important; } \\n' +
'a:link { background: transparent ! important; ' +
    'color: yellow ! important; } \\n' +
'a:visited { background: transparent ! important; ' +
    'color: lime ! important; } \\n' +
'a.fl { background: transparent ! important; ' +
    'color: white ! important; } \\n' +
'input { font-size: large ! important; } \\n' +
'html>body input { font-size: x-large ! important; } \\n' +
'.g { width: auto ! important; } \\n' +
'.n a, .n .i { font-size: large ! important; } \\n' +
'html>body .n a, html.body .n .i { font-size: x-large ! important; } \\n' +
'.j { width: auto ! important; }');
}

function accHomePage(  ) {
    // remove personalized header, if any
    var arTable = document.getElementsByTagName('table');
    for (var i = arTable.length - 1; i >= 0; i--) {
        var elmTable = arTable[i];
        var html = elmTable.innerHTML;
        if (/\\/accounts\\/Logout/.test(html)) {
            elmTable.parentNode.removeChild(elmTable);
        }
    }

    // simplify logo
    var arImages = document.getElementsByTagName('img');
    for (var i = arImages.length - 1; i >= 0; i--) {
        var elmLogo = arImages[i];
        if (elmLogo.alt) {
            var elmTextLogo = document.createElement('h1');
            elmTextLogo.style.fontSize = '400%';
            var sAlt = /Firefox/.test(elmLogo.alt) ? '' : elmLogo.alt;
            elmTextLogo.appendChild(document.createTextNode(sAlt));
            elmLogo.parentNode.replaceChild(elmTextLogo, elmLogo);
            var elmLink = elmTextLogo.parentNode;
            while (elmLink.nodeName != 'BODY' &&
```

```
                elmLink.nodeName != 'HTML' &&
                elmLink.nodeName != 'A') {
            elmLink = elmLink.parentNode;
        }
        elmLink.style.textDecoration = 'none';
    } else {
        elmLogo.parentNode.removeChild(elmLogo);
    }
}

// simplify search form
if (document.forms.length) {
    var arTD = document.getElementsByTagName('td');
    for (var i = arTD.length - 1; i >= 0; i--) {
        var elmTD = arTD[i];
        if (/Advanced/.test(elmTD.innerHTML)) {
            elmTD.innerHTML = '';
        }
    }
}

function accSearchResults(  ) {
    // simplify logo
    var elmLogo = document.getElementsByTagName('img')[0];
    var elmTextLogo = document.createElement('h1');
    elmTextLogo.appendChild(document.createTextNode('Google'));
    elmTextLogo.style.marginTop = '0.2em';
    elmTextLogo.style.marginRight = '0.3em';
    elmLogo.parentNode.replaceChild(elmTextLogo, elmLogo);
    elmTextLogo.parentNode.style.textDecoration = 'none';

    // simplify top form
    var elmAdvancedWrapper = document.getElementsByTagName('table')[3];
    var elmAdvanced = elmAdvancedWrapper.getElementsByTagName('td')[1];
    elmAdvanced.parentNode.removeChild(elmAdvanced);

    // remove "tip" if present
    var elmTip = document.getElementsByTagName('table')[7];
    if (/Tip/.test(elmTip.innerHTML)) {
        elmTip.parentNode.removeChild(elmTip);
    }

    // remove ads, if any
    var aw1 = document.getElementById('aw1');
    while (aw1) {
        var table = aw1.parentNode;
        while (table.nodeName != 'TABLE') {
            table = table.parentNode;
        }
        table.parentNode.removeChild(table);
        aw1 = document.getElementById('aw1');
```

```
}
var tpa1 = document.getElementById('tpa1');
if (tpa1) {
    while (tpa1.nodeName != 'DIV' && tpa1.nodeName != 'P') {
        tpa1 = tpa1.parentNode;
    }
    tpa1.parentNode.removeChild(tpa1);
}
var tpa2 = document.getElementById('tpa2');
if (tpa2) {
    while (tpa2.nodeName != 'DIV' && tpa2.nodeName != 'P') {
        tpa2 = tpa2.parentNode;
    }
    tpa2.parentNode.removeChild(tpa2);
}
addGlobalStyle('iframe[name="google_ads_frame"] { ' +
    'display: none ! important }');

// simplify results count
var elmDivider = document.getElementsByTagName('table')[5];
elmDivider.parentNode.removeChild(elmDivider);
var elmResultsContainer = document.getElementsByTagName('table')[5];
var arTD = elmResultsContainer.getElementsByTagName('td');
if (arTD.length > 1) {
    var sResults = arTD[1].textContent;
    var iParen = sResults.indexOf('(');
    if (iParen != -1) {
        sResults = sResults.substring(0, iParen);
    }
    var iDef = sResults.indexOf('[');
    if (iDef != -1) {
        sResults = sResults.substring(0, iDef);
    }
    var elmResults = document.createElement('h2');
    elmResults.appendChild(document.createTextNode(sResults));
    elmResultsContainer.parentNode.replaceChild(elmResults,
        elmResultsContainer);
} else {
    elmResultsContainer.parentNode.removeChild(elmResultsContainer);
}

// make search results use real headers
var arResults = getElementsByClassName('p', 'g');
for (var i = arResults.length - 1; i >= 0; i--) {
    var elmResult = arResults[i];
    var arLink = elmResult.getElementsByTagName('a');
    if (!arLink.length) { continue; }
    var elmLink = arLink[0];
    var elmWrapper = document.createElement('div');
    var elmHeader = document.createElement('h3');
    elmHeader.style.margin = elmHeader.style.padding = 0;
    elmHeader.innerHTML = '<a href="' + elmLink.href + '">' +
```

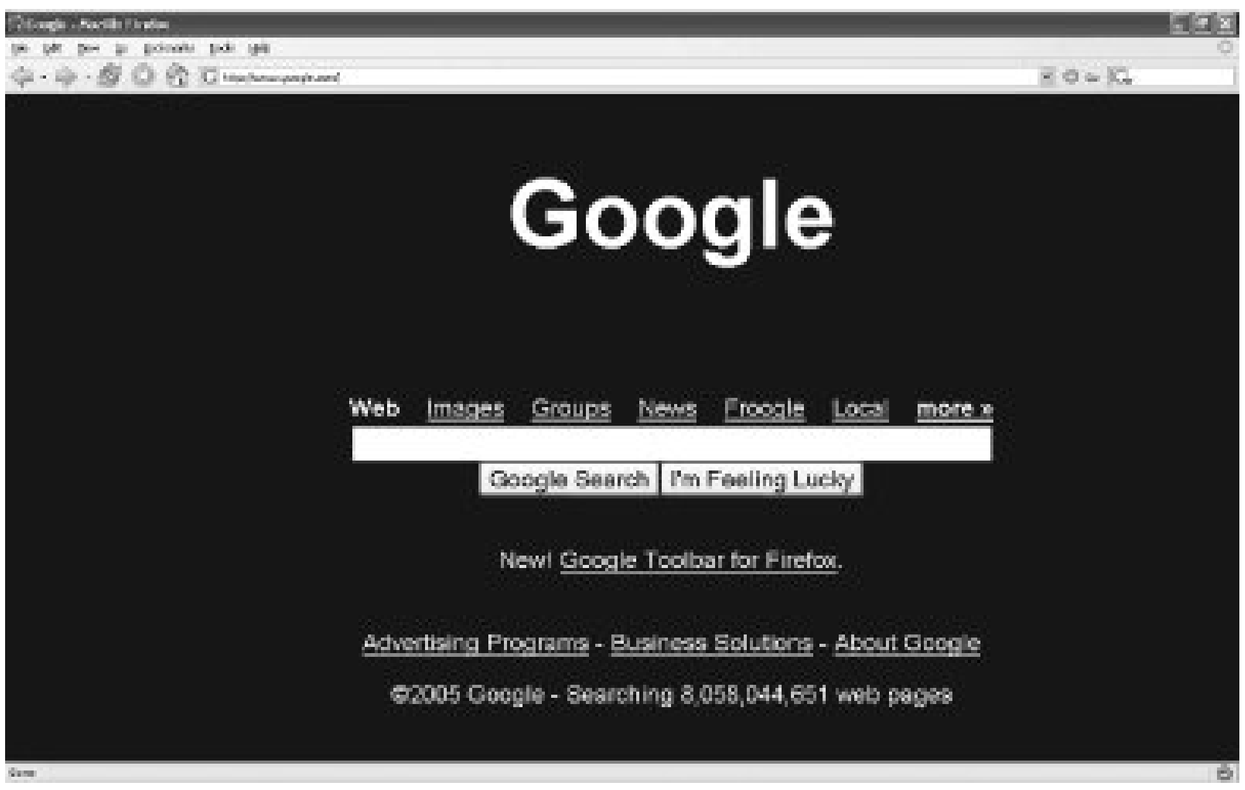
```
            elmLink.innerHTML + '</a>';
        var elmContent = elmResult.cloneNode(true);
        elmContent.innerHTML = elmContent.innerHTML.replace(/<nobr>/g, '');
        arLink = elmContent.getElementsByTagName('a');
        if (!arLink.length) { continue; }
        elmLink = arLink[0];
        elmContent.removeChild(elmLink);
        elmContent.style.marginTop = 0;
        elmWrapper.appendChild(elmHeader);
        elmWrapper.appendChild(elmContent);
        elmResult.parentNode.replaceChild(elmWrapper, elmResult);
    }

    // simplify next page link
    var arFont = document.getElementsByTagName('font');
    for (var i = arFont.length - 1; i >= 0; i--) {
        var elmFont = arFont[i];
        var html = elmFont.innerHTML;
        if (/Result\\ \\;Page\\:/.test(html)) {
            var elmTable = elmFont.parentNode;
            while (elmTable.nodeName != 'TABLE') {
                elmTable = elmTable.parentNode;
            }
            var arTD = elmTable.getElementsByTagName('td');
            if (arTD.length) {
                var elmTD = arTD[arTD.length - 1];
                var arNext = elmTD.getElementsByTagName('a');
                if (arNext.length) {
                    var elmNext = arNext[0];
                    var elmTextNext = document.createElement('center');
                    elmTextNext.innerHTML = '<p style="font-size: ' +
                        'xx-large; margin-bottom: 4em;"><b><a href="' +
                        elmNext.href + '">More Results  ' +
                        '&rarr;</a></b></p>';
                    elmTable.parentNode.replaceChild(elmTextNext,
                        elmTable);
                }
            }
            break;
        }
    }

    // remove bottom ads
    var arCenter = document.getElementsByTagName('center');
    if (arCenter.length > 1) {
        var elmCenter = arCenter[1];
        elmCenter.parentNode.removeChild(elmCenter);
        elmCenter = arCenter[0];
        for (var i = 0; i < 4; i++) {
            elmCenter.innerHTML = elmCenter.innerHTML.replace(/<br>/, '');
        }
    }
```

```
}

document.forms.namedItem('f') && accHomePage(  );
document.forms.namedItem('gs') && accSearchResults(  );
removeFontTags(  );
zoomStyle(  );
```

Running the Hack

After installing the user script (Tools→Install This User Script), go to http://www.google.com. The normally spartan search form has been magnified and simplified even further, as shown in Figure 4-47.

Figure 4-47. Google home page, zoomed



Accessibility studies have shown that low-vision users have an easier time reading light text on a dark background, so therefore the page is displayed as white-on-navy. Unvisited links are displayed in yellow; visited links are displayed in light green. The hack removes several elements from the page, including the Advanced Search link, plus any advertisements for Google services or other messages that occasionally appear below the search box.

When you execute a search, the search results are displayed differently, as shown in Figures 4-48 and 4-49, with the following notable differences:

- The entire page uses the same white-on-navy color scheme we used on the home page.
- The Google logo in the top-left corner is displayed as plain text instead of as an image.
- The top search form no longer includes the Advanced Search option.

- The sponsored links along the top and right are gone.
- The number of results is displayed much larger than before, and in the same white-on-navy color scheme.
- Links to search results pages are displayed in yellow (or green, if you've already visited that page). Other links within each search result, such as the "Cached" and "Similar pages" links, are displayed in white.
- The "Goooooooogle" navigation bar to see more results is replaced by a simple link titled "More results."
- The search box at the bottom of the page is gone.

Figure 4-48. Google search results, zoomed

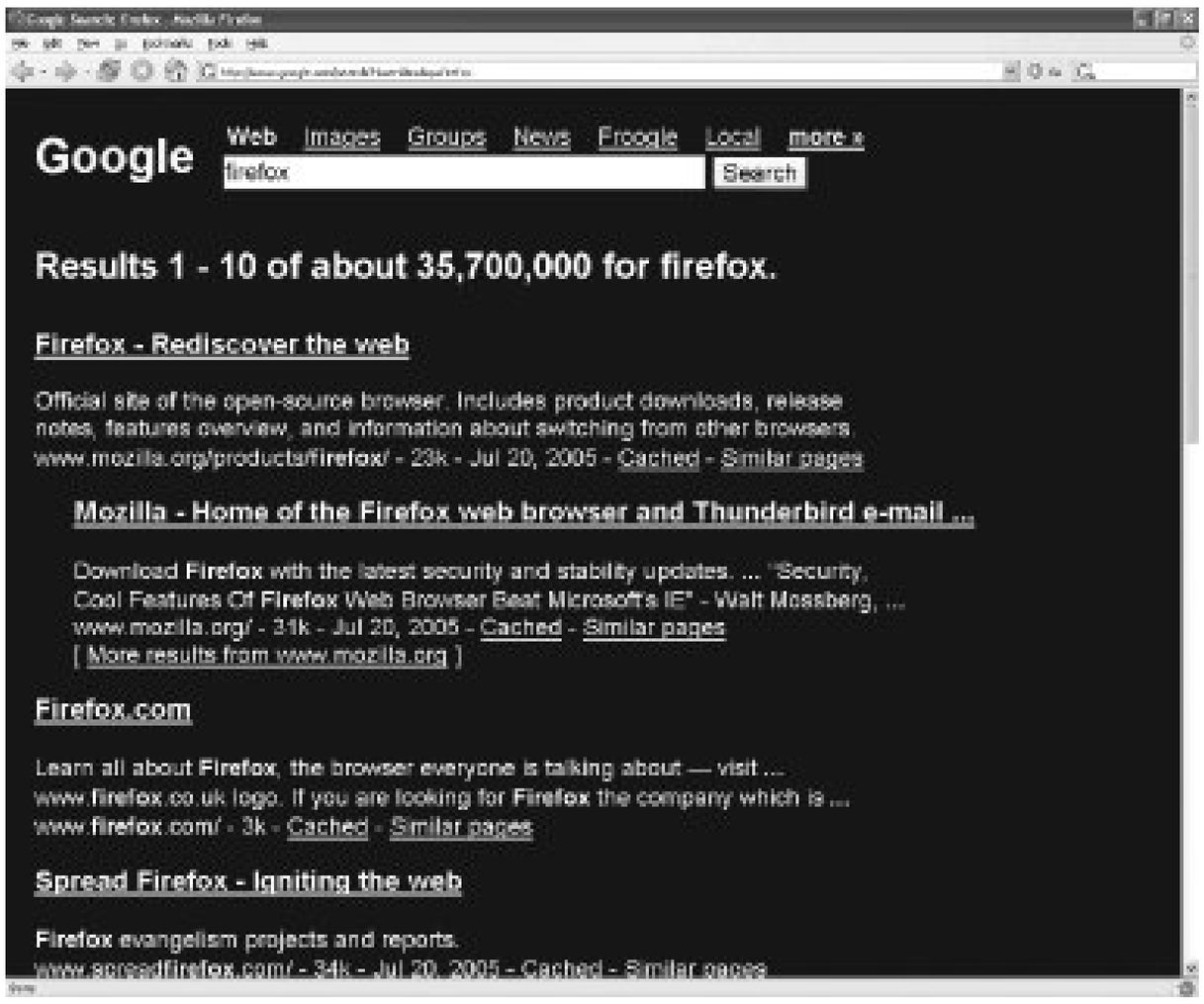
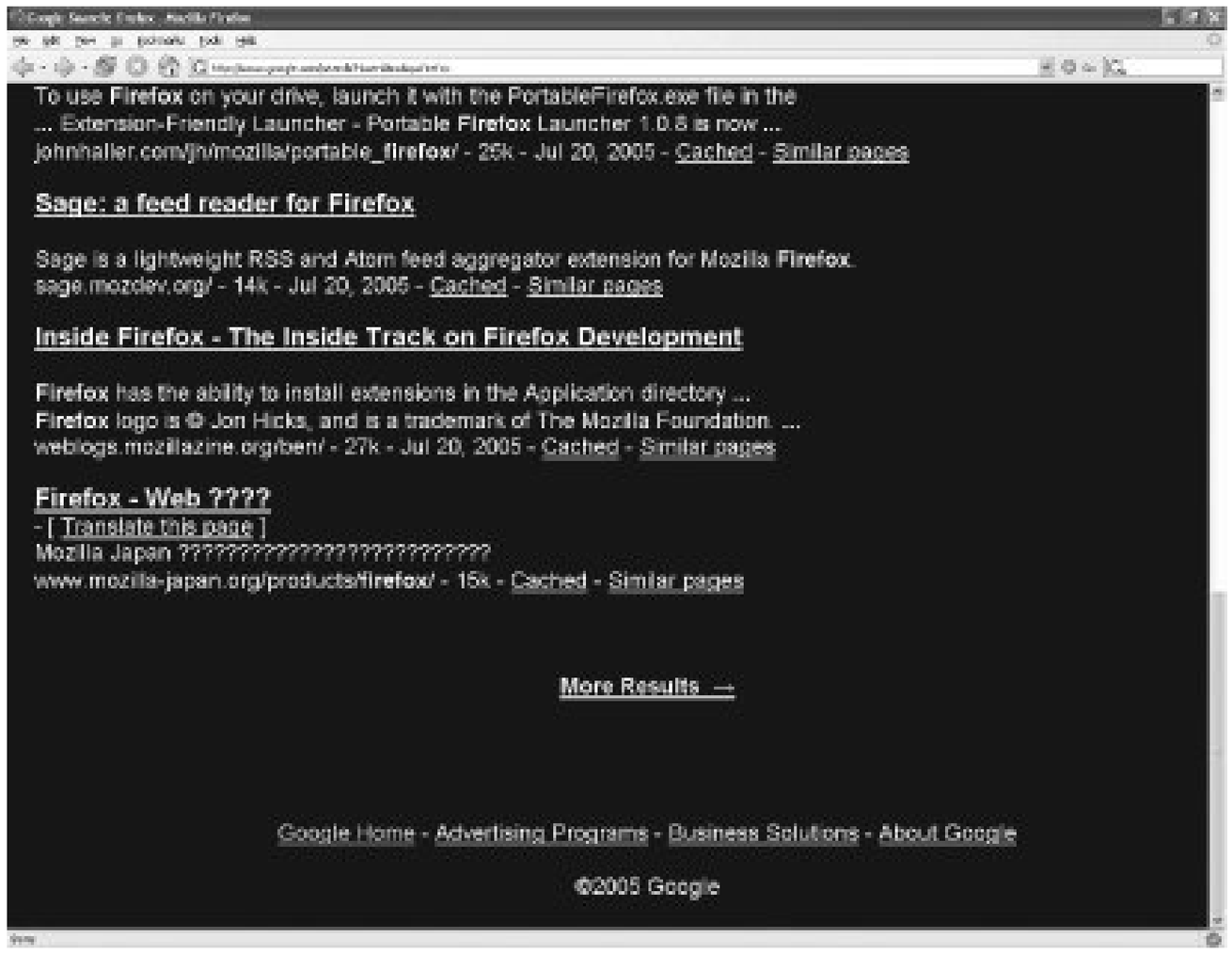


Figure 4-49. Bottom of Google search results, zoomed



If you click the Images link at the top of the page to search for the same keywords in Google Image Search, you will see that the image search results have been similarly hacked, as shown in Figure 4-50.

Figure 4-50. Google image results, zoomed

As with the web search results, the top navigation has been simplified, the number of results is more

prominent, and the "Gooooooooogle" navigation bar has been replaced by a single "More results" link that moves to the next page of images. The image thumbnails themselves cannot be magnified, since Google provides them only in a specific size.

Mark Pilgrim

PREV

Hack 62. Search for Lyrics on Google

Use info from the current or selected track to search for lyrics pages via Google in Safari.

There are tons of lyrics sites on the Web that are ready to provide you with the words to your favorite songs. This hack Googles for the lyrics of the currently playing or selected song in iTunes. You'll be singing along by the second verse.

This hack uses the AppleScript scripting language for Mac OS X.

The Code

This script gets the name and artist of the currently playing track (or, if no track is playing, the selected track), tidies up the text, and incorporates it into a URL that is then used by Safari to search Google for sites containing the lyrics of that particular track:

```
-- base of the URL string, includes the term "lyrics"
property baseURL : "http://www.google.com/search?q=lyrics+"

tell application "iTunes"

  -- get a reference to playing or selected track
  if player state is not stopped then
    set theTrack to current track
  else if selection is not {} then
    set theTrack to (item 1 of selection)
  else
    display dialog "Nothing is playing or selected." buttons {"Cancel"} \xc2
      default button 1 with icon 0
  end if

  -- get the name and artist and replace "bad" characters
  tell theTrack
    set nom to my fixChars(name)
    set art to my fixChars(artist)
  end tell

  -- assemble URL string, replace spaces with "+"
  set theURL to (baseURL & \xc2
```

```
    (my replace_chars((art & "+" & nom), " ", "+"))) as text

  my open_location(theURL)

end tell

on fixChars(a)
  set myDelims to {"!", "@", "#", "$", "%", "^", "&", "*",\xc2
    "(", ")", "-", "-", "+", "=", ":", ";", "'", ",", ".", "/", \xc2
    "<", ">", "?", "{", "}", "[", "]"}
  repeat with curDelim in myDelims
    set AppleScript's text item delimiters to curDelim
    set s to every text item of a
    set AppleScript's text item delimiters to {""}
    set a to s as string
  end repeat
  return a
end fixChars

on replace_chars(txt, srch, repl)
  set AppleScript's text item delimiters to the srch
  set the item_list to every text item of txt
  set AppleScript's text item delimiters to the repl
  set txt to the item_list as string
  set AppleScript's text item delimiters to ""
  return txt
end replace_chars

to open_location(theURL)
  tell application "Safari"
    activate
    -- un-comment if you want to keep from opening windows:
    if name of window 1 does not start with "Google Search:" then
      make new document at end of documents
    end if
    set URL of document 1 to theURL
  end tell
end open_location
```

I've defined a script property (`BaseURL`) at the beginning of the script, setting it to the main portion of the Google search URL, along with the start of the query, the word `lyrics`. The script will add more to the base URL later.

First things first: the script determines whether there is a current track or a selected track and sets the variable `sel` to a reference to one or the other.

Next, the script gets the `artist` and `name` properties of the track referenced by `sel`. It will use these text strings in building the search URL. Additionally, it removes any undesirable characters that migh lead to an incorrect search. The handler `fixChars( )` removes a variety of *bad* characters from the text string sent to it and returns the cleaned-up text.

Now, if the selected/playing song is "Third Uncle" by Brian Eno, the assembled `theURL` variable will contain a string that looks like this:

```
http://www.google.com/search?q=lyrics+Brian+Eno+Third+Uncle
```

All that's left to do is to send the `theURL` string to Safari, using the handler `open_location( )`. If the current front window in Safari doesn't already have a Google search result in it, a new window will be opened; otherwise, the same window is used to load the Google result.

The same effect is used in "AMG EZ Search" [Hack #79 in iPod and iTunes Hacks].

The Google search results page will (hopefully) contain a whole bunch of links to lyric pages for your song, as shown in Figure 4-51.

Figure 4-51. Wow, over 3,000 results!

Running the Hack

Enter the code into Script Editor. Save this script in your iTunes*Scripts* folder as a compiled script (set the File Format in the Save As... dialog to Script) and name it*Google Lyric Search*.

The script will target the currently playing track; if no track is playing, it will target a selected track. Using the Artist and the Song Name, a Google search URL will be constructed and sent to Safari.

If you want to emulate Google's I'm Feeling Lucky button, use the following as your`baseURL` at the start of the script:

```
http://www.google.com/search?btnI=I'm+Feeling+Lucky&q=lyrics+
```

Remember: results are not always fruitful...especially on instrumental tracks. (Da-*dum*!)

Doug Adams

PREV

Chapter 5. Google Maps

Hacks 6369

Google revolutionized online maps with the release of Google Maps (http://maps.google.com) in February 2005, by offering a fast-loading, dynamic, and responsive map that you can zoom into and move around, much like you can with a real map. And unlike a real map, a Google Map can display extended information about points on the map, routes and boundaries, satellite images, or a hybrid of street maps on top of satellite images.

Perhaps the most revolutionary aspect of Google Maps is its API, which allows anyone to build their own maps with their own geographic points and extended information. The API has spawned a series of mapping *mash-ups*, where geographic points from one source are displayed using Google, often at a third site with no relation to Google or the original data source. For example, Chicagocrime.org (http://www.chicagocrime.org) takes data about crimes from the Chicago Police Department, plots those crimes on a map with the Google API, and offers it for public consumption at its web site, givin Chicagoans a map of where crimes are happening in their city.

What's revolutionary about the site is that developer Adrian Holovaty didn't need to contact the Chicago Police or Google to make it a reality. And no one from either Google or the Chicago Police needed to have a meeting to make the visualization happen. With freely available data and Google's open API, an intrepid developer can help others visualize our physical world in a new way.

A few months after the Google Maps release, Google unveiled Google Earth [Hack #69], a desktop application that visualizes the world in 3-D. Between Google Earth and Google Maps, the sudden boom in geographic visualizations has led to a new geek pastime called *Google Sightseeing*. People scour the maps in both programs, finding unique satellite photos and traveling the globe from their computers. Blogs such as Google Sightseeing (http:// www.googlesightseeing.com) point out the best of the best locations in both programs and are a good place to start your explorations.

In this chapter you'll find hacks to help you navigate Google Maps [Hack #64], find locations in your community [Hack #63], build your own maps with the Google Maps API [Hack #67], and build a mash-up with external data [Hack #68].

This chapter merely scratches the Google Mapping surface. You'll find a complete tour of the best mash-ups, advanced API techniques, and a whole host of fun mapping projects in *Google Maps Hacks* by Rich Gibson and Schuyler Erle (O'Reilly).

Hack 63. Think Global, Google Local

Take web searching to the streetsyour street, in fact. Google Maps narrows down all those zillions of results to within the range of a particular city, state, or postal code.

While the Web and Google have taught us to think global when it comes to looking for information, web searches often fail in the simple task of finding things in our own backyards. Sure, the island of Celebes is the home to Sulawesi Kalossi, but where can I find the finest cup of Sulawesi coffee within walking distance? And even more importantly: do they have free wireless Internet access?

This isn't to say that Google doesn't pay attention to any mention of locale in your queries. If you were, let's say, to search for `coffee san francisco`, you would notice a set of local San Francisco finds ["Quick Links" in Chapter 1] at the top of the results page. As you can see inFigure 5-1, Google also provides addresses, phone numbers, and mileage (from the center of San Francisco, presumably).

Figure 5-1. Local finds appearing as "magic links" at the top of the results page

Google combines its index with data gleaned from theYellow Pages to zero in on local results that

very often prove interesting and useful.

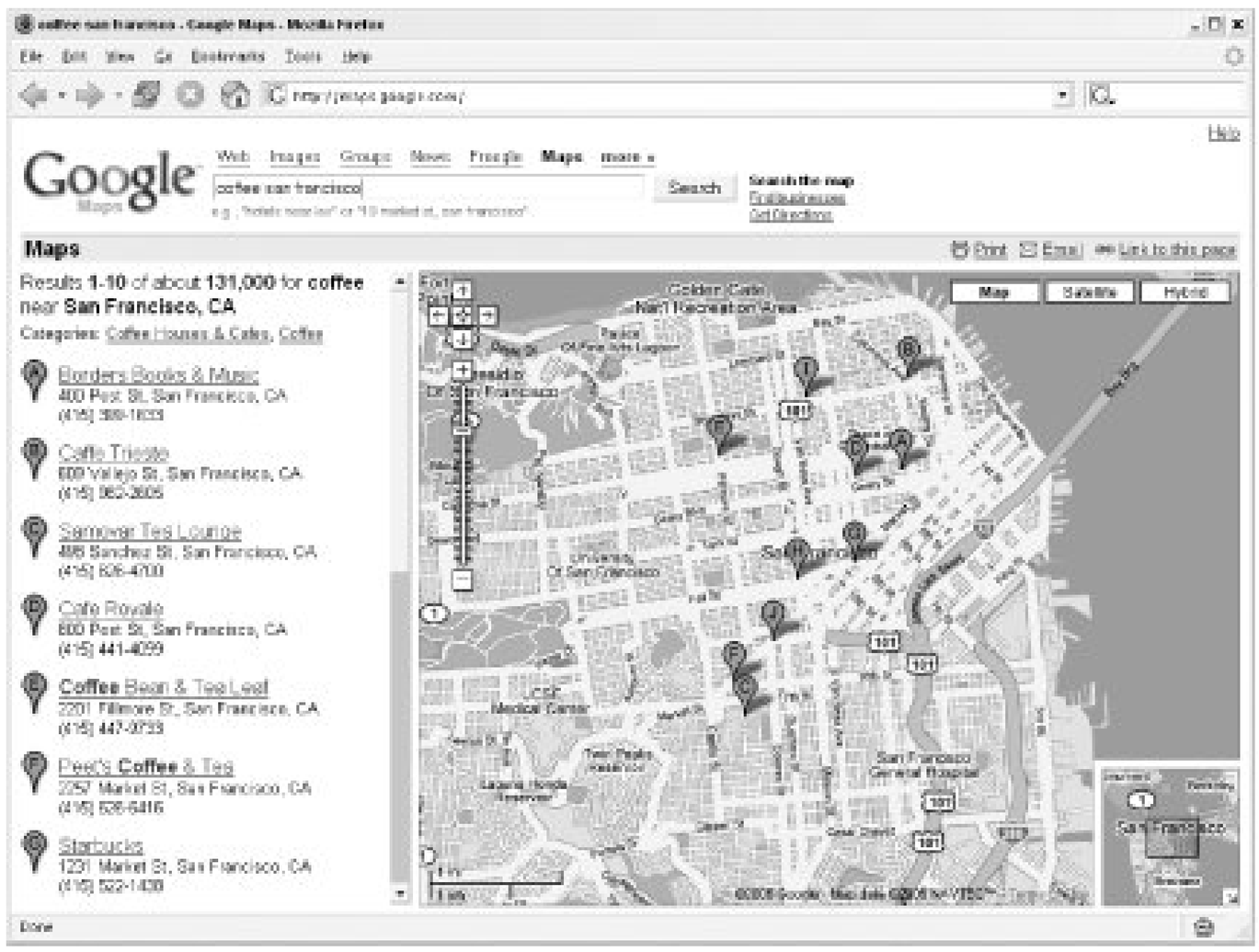
This data is so interesting, in fact, that Google has taken the service beyond that sprinkling of magic links, launching Google Maps (http://maps.google.com), a location-aware frontend to the Google search engine. The Google Maps home page shown in Figure 5-2 has the familiar search field at the top of the page, and it puts a map of the United States front and center. In the search box, you type your search query as usual along with a city (by itself, if the city is unambiguously well known e.g., San Francisco or New York, not Rome or Concord) and a state name or zip code.

Figure 5-2. The Google Maps home page

Before you get too excited about finding that perfect coffee shop on the island of Celebes, you should know that Google Maps doesn't search everywhere. Currently, only the United States, UK, Canada, and Japan are supported. Don't get too used to that limitation, though: Google is planning on expanding.

The query for `coffee san francisco` turned up a nice collection of coffee shops, bookstores, and other places where you can get a cup of coffee in and around San Francisco, as shown in Figure 5-3.

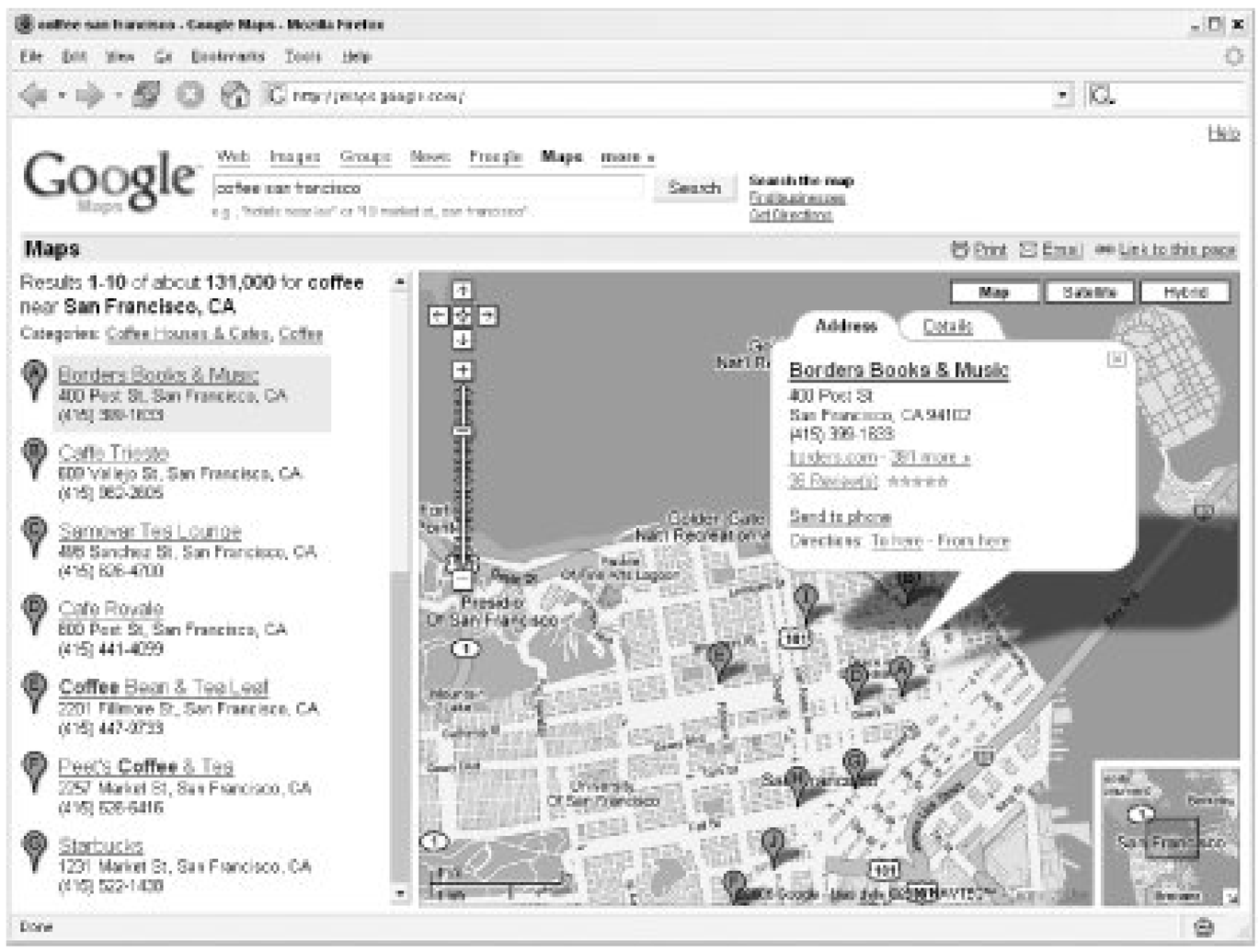
Figure 5-3. Google Maps search results



Notice that each of the results is assigned a letter (e.g., Borders Books & Music is "A") associated with a pin in the map of the area to the right. Each result, as with the magic links, has an associated address and phone number.

Click one of the results, and that particular entry is highlighted on the map, as shown in Figure 5-4.

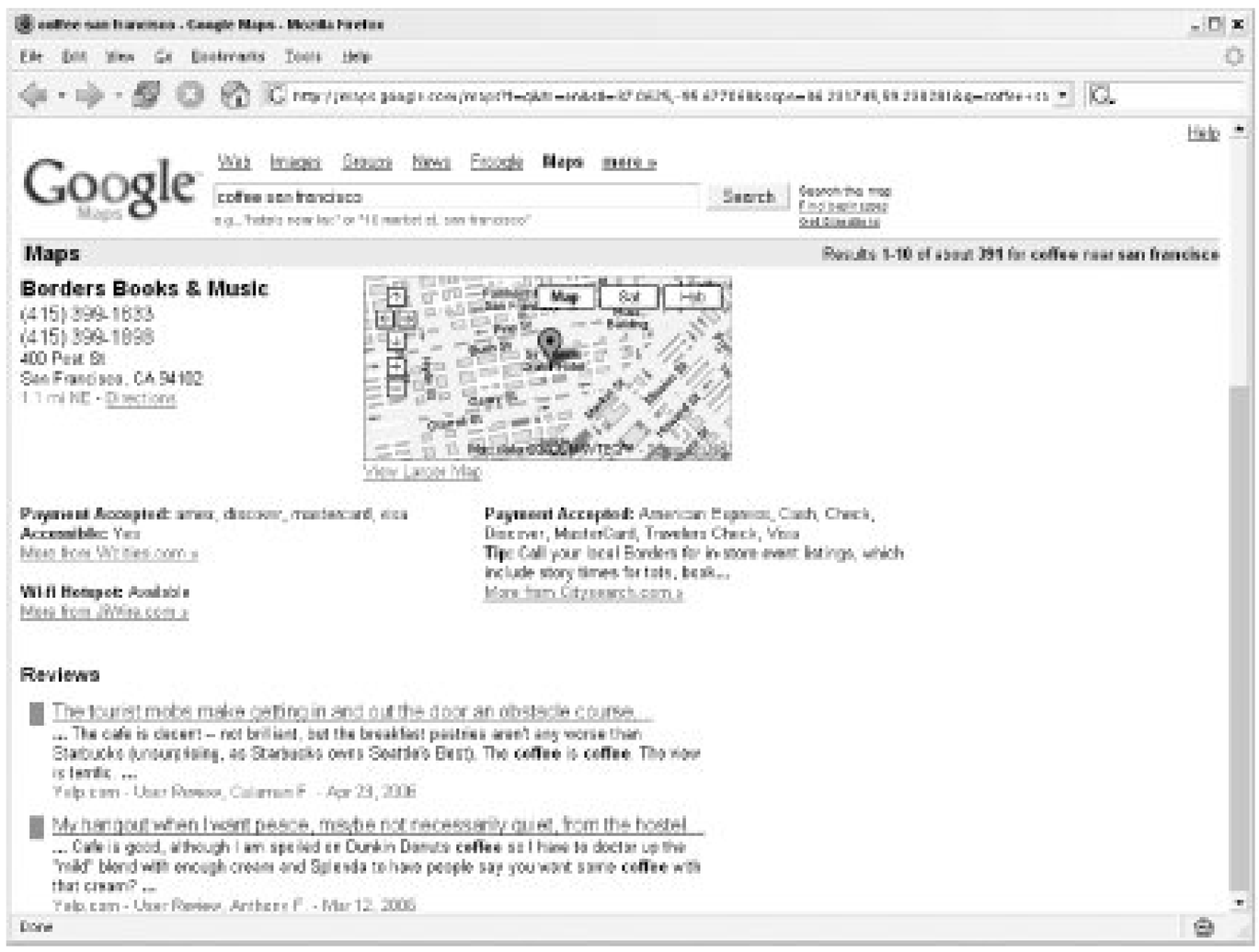
Figure 5-4. Google Maps search result on a map



The window that appears over the point you select provides some more information about your potential destination, including a full address, a star rating from reviews, and quick links to directions to and from this point. Click the Details tab in the window for even more information that can include hours, nearest train, payment types accepted, cuisine, recommended attire, or even tips about that location. For the full details, go back to the Address tab and click the business name to view the business detail page.

As you can see in Figure 5-5, a business detail page includes a map zeroed in on only that one result.

Figure 5-5. A Google Maps business detail page, complete with map and reviews



You'll also find any details about the business, links to reviews at other web sites, and links to sites that refer to (and I don't just mean link to) the business. And yes, Google even lets you know if the location has WiFi available, but you still have to call to find out if it's free.

As with any Google Map, you can pan around the small business location map by clicking the map and dragging it in any direction, or by clicking the arrow buttons. Zoom in or out using the plus or minus buttons, and click the Sat or Hyb buttons to see satellite pictures or streets superimposed on satellite pictures of the area. And if the window is too small for you, click View Larger Map to open a larger map to work with.

With Google Maps, you can make quick, intelligent decisions about where to go and how to get there You also might learn something about what's available in your own backyard.

Hack 64. Get Around http://maps.google.com

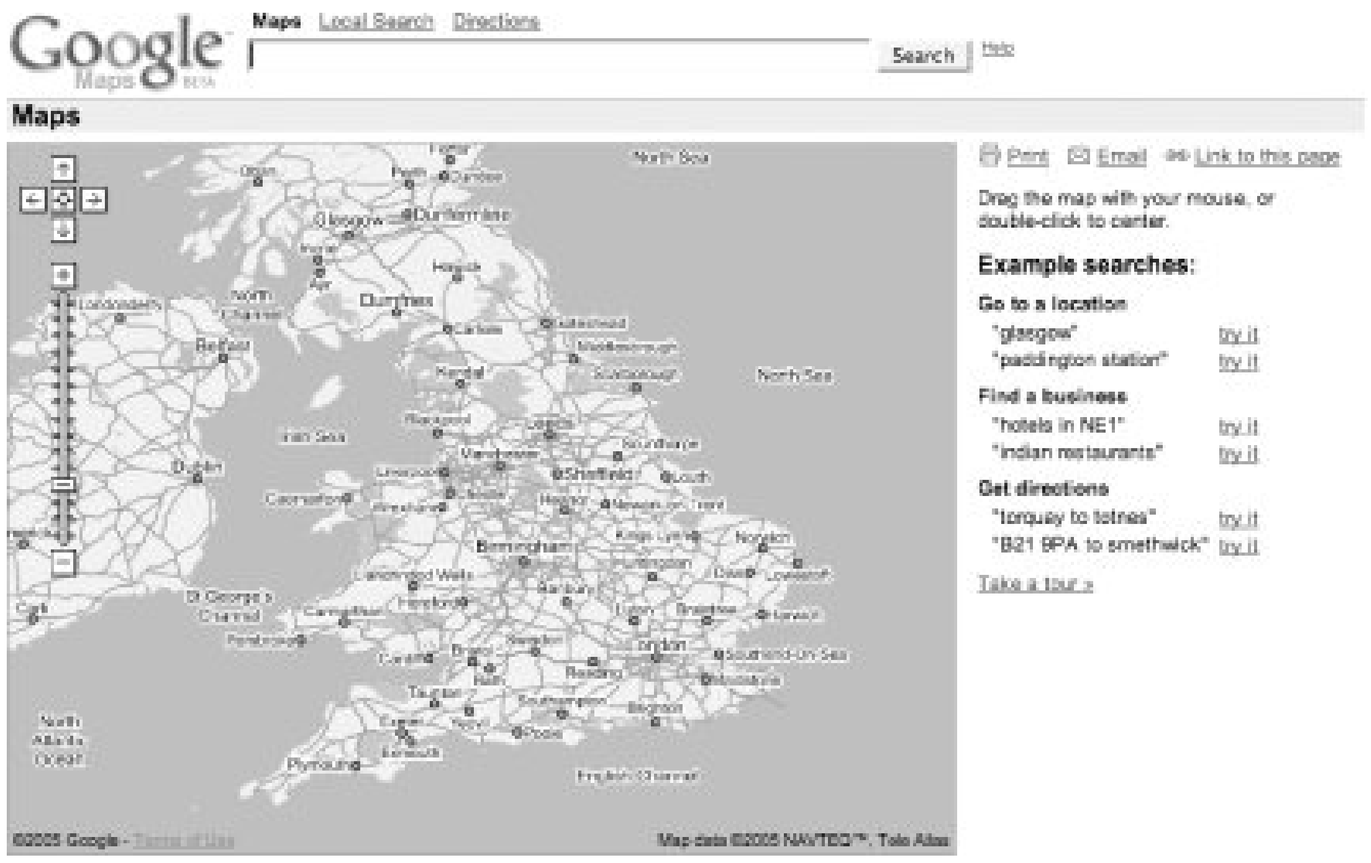
Sometimes you need a map to the map.

Google applied its trademark, carefully designed simplicity, to provide us with its (first) view of place Go to http://maps.google.com/, and you'll get the view shown in Figure 5-6. If you want maps of the United Kingdom, try http://maps.google.co.uk/, and you'll get the view in Figure 5-7. A similar map exists for Japan at http://maps.google.co.jp/.

Figure 5-6. http://maps.google.com/

This shows us what appears as a standard Google search box (called the Location Search box), an overview or orientation map of the country in question, and a results area with instructions and sample searches.

Figure 5-7. http://maps.google.co.uk/



Please remember that this book was written using beta software, and the Google engineers are continually tinkering with the site in order to provide the most compelling possible system. This means that things will change!

What's Different About Google Maps?

Google Maps is a web-mapping service that solves the same old problem of online mapping. So why, 10 years or more into the web revolution, is Google Maps such a big deal? Some of the excitement comes from the Google name and its philosophy. Google states on its corporate philosophy page at http://www.google.com/corporate/tenthings.html that "you can make money without doing evil." However, more of the interest in Google Maps may stem from other ideas stated on the philosophy pagefor example, "The interface is clear and simple" and "Pages load instantly."

Clean pages and fast performance? A commitment to avoid doing evil? Which drives traffic and mind share? Maybe at this moment on the Web you can have it all. In addition to these features, Google Maps also offers a number of innovations in web user interfaces.

Single search box

The first thing that draws attention is that Google uses a single search box for location searches. Do you want to look up an address? Just type it in the box. No more tabbing betweer different fields for street address, city, state, and zip code! (In Internet Explorer, you can even paste multiline addresses into this box, believe it or not.)

Draggable maps

The standard in web mapping is the usual web interface, in which you click on a button to pan the map and see more terrain. What if you wanted to just click and drag to navigate the map?

Well, now you can!

Integrated local search

You can use that single search box to look for the things you want, such as "hotels near Sebastopol," or, for more choices, "hotels near SFO" (SFO is the code for San Francisco International Airport). If you just want to find all the hotels in a given area, zoom into that area, then search for "hotels," and Google Maps will constrain its search to the area shown on the map.

Satellite imagery

With a single click you can flip between viewing a map and viewing satellite or aerial photography. How cool is that?

Keyboard short cuts

You don't need to click and drag your mouse, or strive to hit the little Zoom In and Zoom Out icons: you can use the arrow keys to move around in your map.

Getting Around

Google Maps starts with the overview map shown in Figure 5-6. You can move around that map by clicking and dragging your mouse on the map, by double-clicking your mouse on the map, or by using the arrow keys. Holding the mouse button down and dragging will cause the whole map to move, as if the web browser is providing a small window onto a much larger map. If you double-click on a spot, the map will smoothly pan until the point you clicked on is centered. Using the arrow keys has the same effect as clicking and dragging with the mouse.

Entering a Location

There are many ways to enter locations [Hack #65], but let's start off easy. The conventional way of entering a location is a street address. We've come to accept address lookups in online mapping services as commonplace, but there is a great deal of behind-the-scenes work. In order to display a map of a street address, the system must first find a latitude and longitude that corresponds to this address. The process of linking something (e.g., a street address) with a latitude and longitude is called *geocoding*.

When you enter a query into the search box, Google takes your input and does its best to turn it into a location that can be mapped. So let's start close to home and enter the street address of O'Reilly Media headquarters into the search box:

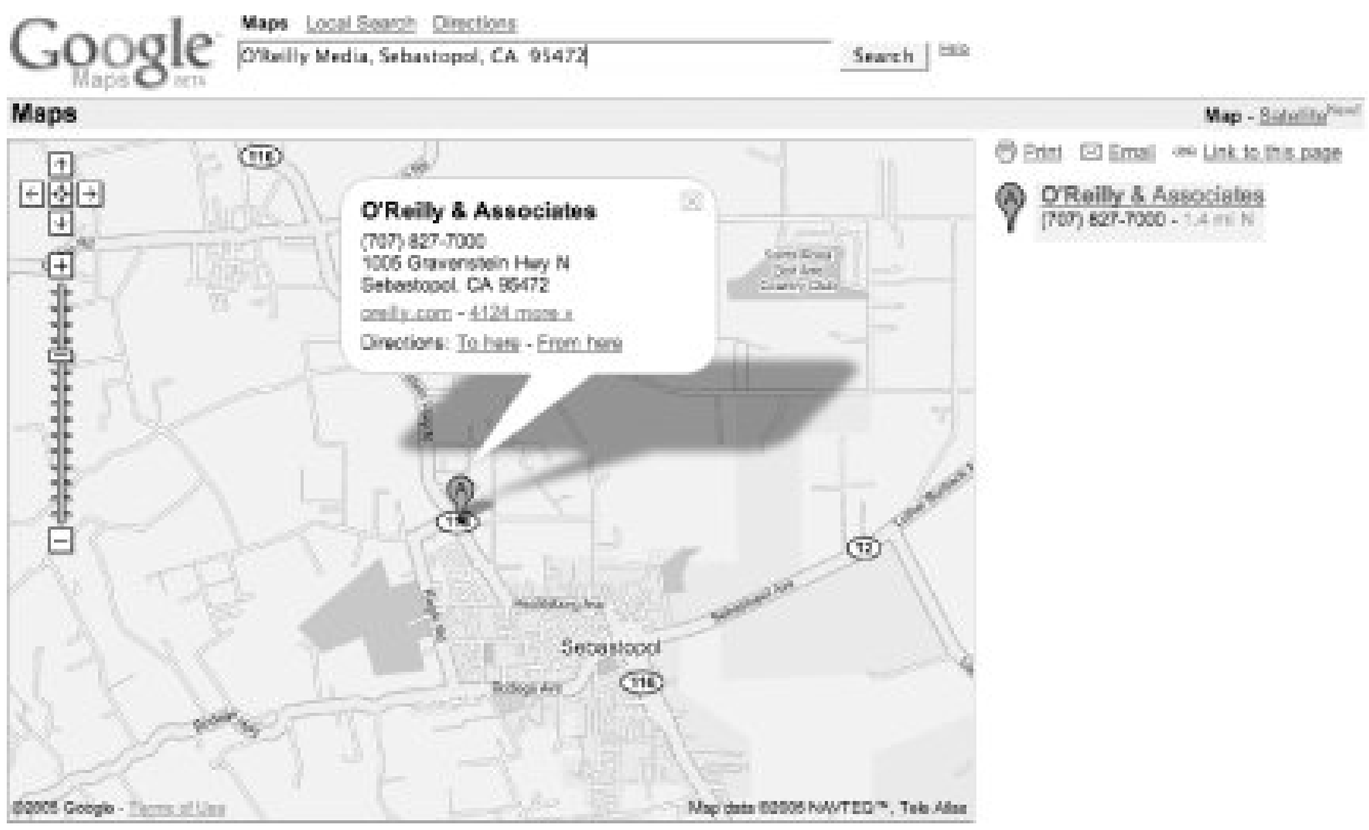
```
1005 Gravenstein Highway North, Sebastopol, CA 95472
```

You could also enter the company name and get the same result:

```
O'Reilly Media, Sebastopol, CA 95472
```

Click Search and you'll get the map shown in Figure 5-8, which shows the address (as best as Google can determine) and hyperlinks to get directions to or from this spot.

Figure 5-8. A Google map of O'Reilly Media's headquarters



The satellite view in Figure 5-9 (zoomed in from the area shown in Figure 5-7) clearly shows that the O'Reilly Empire is centered in a parking lot median strip....

Figure 5-9. O'Reilly Media, apparently located in a median strip of a parking lot

You can also enter a street intersection; for example:

```
Hollywood & Vine St, Hollywood, CA 90068
```

The act of looking up a location has set your search area, or the*extents* of your search area, and you can now use the search box to get more information. For example, if you first zoom in to San Francisco, you can then search for "great sushi" and return results limited to the San Francisco area.

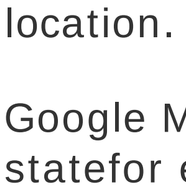
The Google Maps tour suggests that a search on "Great Sushi in New York" is useful. It turns out that "great sushi in San Francisco" also brings up a list of restaurants, but for some reason, "great sushi in Sebastopol" just doesn't work. To be fair, "great sushi in Sebastopol, CA" does bring up our two sushi places. But it also brings up the Larkspur Elementary School District, 37 miles from Sebastopol.

Varying the adjective usedsay from "great" to "mediocre"brings up a new grouping of restaurants. These are not the same places that show up when you do a Google search on "mediocre sushi in San Francisco," so I'm not sure what the qualifications are. "Cheap but filling sushi in San Francisco" might be a more palatable search!

Finding meaningful results for local search is still an unsolved problem. Fortunately, Google is good at searchand getting better all the time.

Rich Gibson and Schyler Erle

Hack 65. Find Yourself (and Others) on Google Maps



Google Maps supports many ways to specify location.

Using addresses to find a place makes a lot of sense for places that have an address, but what do you do when you don't have an address? Fortunately for you, the Google Maps team has supplied a number of additional ways to find yourself.

I suspect that the goal is to create a system in which, if you can imagine a somewhat standard way of representing a location, then Google Maps will support it. The functionality is not quite there yet, but it does support a lot of ways of finding places. As with all of the hacks in this book, and as a general philosophy of life, experimentation is your friend!

The number one rule for finding places using Google Maps is that if there is a way of specifying location that makes sense to you, go ahead and try it! As we saw in "Get Around http://maps.google.com" [Hack #64], standard addresses work, but so does entering a city and state, or a zip code alone. Street intersections also work, as long as you add a city and state.

You can also enter coordinates as latitude and longitude, like `38, -122`, or `38 N, 122 W`. Most modern people don't relate to latitude and longitude directly, but it is a compact and precise way to mark a location.

Google Maps is good at searching by business name. You can search by business name, city, and statefor example, "O'Reilly Media, Sebastopol, CA"with good results. Entering a business name and a city, or a business name and a state, brings up a list of possible matches.

The best Google Maps feature ever is the proximity search, at least for one of my friends, who is a vegetarian and travels a lot. Before Google Maps, he spent a lot of time on other map services planning for trips. A common query was for the closest Whole Foods Market in whatever city he was visiting. Now he can just type his query into the single search box: "whole foods market near Boston MA." As long as he remembers to change *Boston, MA* to his current city, he is set. Table 5-1 shows examples of searches that do and don't work.

Table The limits of Google Maps' understanding

Example	Works?	Description
1005 Gravenstein Highway N, Sebastopol, CA 95472	Yes	Full street address works great.

Example	Works?	Description
79th St and Broadway, NY79th St and Broadway, 10024	Yes	Intersection and city, or intersection and zip code.
Santa Rosa, CASanta Rosa, NM	Yes	City and state works.
San FranciscoMoscow	Yes	The bare city name works absurdly often. If the same city appears in more than one state, it appears to pick the largest. International cities were added recently, but data quality varies.
CA or California	No	State or state abbreviation alone doesn't work.
94305	Yes	Zip code works. Postal codes for other supported countries, such as Canada and the UK work as well.
LAXSFO	Yes	Airport codes work.
Paddington	Yes	In the UK and Japan, subway stations work as locations.
37, 122	Yes	Latitude and longitude expressed as decimal degrees with to express West longitude or South latitude.
37 N, 122 W	Yes	The same, but use `N` and `S` and `E` and `W`.
N 38 24' 08.8" W 122 49' 44.2"	No	Latitude and longitude as degrees-minutes-seconds doesn't seem to work, but perhaps after partaking of the magic syntax elixir....
Range and township	No	Google Maps doesn't seem to do range and township. This would be a great feature for genealogy buffs that get records of their forbears' property transactions.
[location] to [location]	Yes	Any of the above locations that work can be mixed and matched with the word *to* in between them to get driving directions.
[thing] near [location]	Yes	You can use any of the above locations to search for nearby businesses and points of interest.

Odd and Surprising Ways to Find Things

Not everything is documented! Like Google's search, there are a lot of things that just work that are not documented (or at least they are not documented where you are likely to see them). For example, as of April 30, 2005, I could find no mention that entering a latitude and longitude in the search box would have any effect, and yet it works!

The moral is that when you have a wild idea about a way to search for something, try it first, and then if it doesn't work, enjoy that temporary feeling of satisfaction that comes from being ahead of the curve (well, either ahead of the curve, or plumb crazy, but since there is no reliable way to determine which is which, you might as well enjoy it).

When Locations Fail: The Importance of Context

Unless you specify a location in your searche.g., "edible food near King's Cross"Google Maps assumes that the place that you are searching for falls within the area, or*extent*, that is currently shown in the map. As a result, a search that works on the full extent will sometimes fail if you have a local context set. You can reset that context by adding "near [some location]" to your search, or by clicking on the Google Maps logo in the upper left of thepage.

Rich Gibson and Schyler Erle

Hack 66. Build Your Own Google Map

Add a Google Map to your web site in a few quick steps.

One look at the Google Maps API documentation (http://www.google.com/apis/maps/) can send otherw other direction. The Maps API uses JavaScript, a language that can be confusing even for seasoned deve latitude coordinates for every marker you want to add to the map, but the API doesn't provide any type addresses or place names into coordinates.

The Google Maps API was definitely built by engineers for engineers, and it requires a bit of study before However, there are some ways to cheat the system and put together your own map in just a few minute

As an example, suppose you want to share your recent travel destinations with your site readers so they list of cities, but no coordinates, and you're not a JavaScript expert. This hack shows how to assemble a have to touch a GPS.

Google Map Maker

You don't need to know anything about JavaScript or geocoding locations to assemble a map you can pu minutes. UK developer Richard Stephenson built a tool called Google Map Maker that takes the pain out

To get started, browse to Google Map Maker (http://donkeymagic.co.uk/googlemap/), where you'll find the map controls to zoom in or out, or drag the map around until you find the section of the world you w

Zoom in close to the first point you want to map, click the Activate checkbox in the controls next to the the marker to be. The latitude and longitude automatically appear in the control box. Type in a marker n appear when your readers click that particular marker. Click "Add marker," and Google Map Maker reme the map. Repeat the process for different locations, and you end up with a map such as the one shown i

Figure 5-10. Adding points with Google Map Mak

Once you've added all your points to the map, click Generate Code, which gives you the source code for
Copy the code and add it to an HTML file, such as *recent-travels.html* .

Setting Your API Key

The final step is replacing the Google Maps API key in the generated code with your own Google Maps AI
is a bit of JavaScript that looks like this:

```
<script src="http://maps.google.com/maps?file=api&v=1&key= [long string of characters ]
```

You need to replace the existing long string of characters after `key=` with your own key. Browse to the Gc
(http://www.google.com/apis/maps/signup.html) and request a key.

As you register your key, be sure to include the domain where you'll display the map. If you'll share you
http://www.example.com/recent-travels.html , use http://www.example.com as the domain. If you'll dis
http://www.example.com/travels/recent-travels.html , be sure to include the subdirectory. To be associa
API requires the precise location where the map will be published.

Rolling Out Your Map

Once you have the key, edit the file to include your key, upload the file to your server, and open the pag
HTML so it fits in with your site design. In this example, if you add the page heading My Recent Travels,
11 .

Figure 5-11. Custom Google Map generated with Google I

As you click on points, you'll see the pop-up content you included with each point. From here, you can li
share the map with the world.

Hacking the Hack

Google Map Maker gives you the code for an entire HTML page, but with some careful dissection, you ca
your site. Open *recent-travels.html* and take a look at the source code. The page is made up of three dis
top of the page inside of `<script>` tags, a single CSS `<style>` section, and a `<div>` tag in the body of the

To show the map in another page, copy the `<script>` and `<style>` sections of *recent-travels.html* and pas
somewhere between the `<head></head>` tags. Now copy the lone `<div>` tag and place it in your page whe
mind that you might have to adjust the `width` and `height` attributes of the tag to get the map to fit into a
map once it's been added to an existing page.

Figure 5-12. Custom Google Map inside an existing

Though this hack was supposed to keep you from getting your hands dirty withJavaScript, you might ne
the map into an existing page. In this example, the original map had code that added a large map contr

```
map.addControl(new GLargeMapControl(  ));
```

Instead, I changed the code to add a small map control, like so:

```
map.addControl(new GSmallMapControl(  ));
```

And I completely removed the code that adds the Map, Satellite, Hybrid control to the map. You might a
zoom level of the map once the map has been resized. Look for the code that initializes the map, which

```
map.centerAndZoom(new GPoint( -95.9765625, 37.43997405227057),  14);
```

The last number, `14` , is the zoom level for the map when it loads. Change this to any number between `0`
out.

Hack 67. Add a Google Map to Your Web Site

Here's how to get started using the Google Maps API.

At O'Reilly's Where 2.0 converence on June 29, 2005, Google announced an official and documented AP for anyone to add a Google Map to a web page by cutting and pasting a few lines of JavaScript from the

People reacted to the new API in one or more ways. My first act was to scratch my own itch by writing a Google Map. Fortunately, better GPX-to-Google Maps solutions have been created, one of which is docur Maps" [Hack #37 in Google Maps Hacks]. After scratching that itch, I looked to our Geocoder.us site. Sc Census Bureau's public TIGER/Line Map Server API and how to display the resulting map with a neat litt and clunky, but they worked.

The Google Maps API gets rid of the need for that level of head scratching! The march of progress in cor first figuring out ways to do new things, and then progressively making those tasks easier and leaving t programs for their Osborne luggable computer.

I used Google Maps to bring the Geocoder.us site into the protective embrace of the Google Maps API. Ge a free U.S. address geocoder. You can go to the web site and get the latitude and longitude for a U.S. st interface to get the latitude and longitude automatically for a group of addresses [Hack #62 in Google M Maps by scraping their search results, but it's not a part of the official API and doing so violates Google's the Geocoder.us site is based on free data without limited terms of service for noncommercial use.

Figure 5-13 shows the results of geocoding the address of O'Reilly Media's headquarters with the origina location of the address that we just looked up. We'd like to replace this somewhat slow map generated b more attractive, and more easily navigable maps offered by Google Maps. (The original Geocoder.us ma http://geocoder.us/demo_tiger.cgi.)

Figure 5-13. The Census Bureau map originally used by http

Get a Developer Key

The first step in putting a Google Map on your page is to generate a developer's key, which is an alphan Google and helps them track usage of Google Maps. Having to sign up for a developer's key can be some to pay for being able to include free (as in beer) maps on your web site with such relative ease.

You'll need a distinct developer's key for each directory on your site that includes Google Maps. You don script. So if you have several pages that generate calls to Google Maps from the same directory, you on

Fortunately Google has made getting developer's keys as easy as filling in a web form. The Google Maps http://www.google.com/apis/maps/. This includes links to documentation, examples, Terms of Use, and version of the Terms of Use, then the full legalese version. Figure 5-14shows the form with the URL we the Terms of Service, then click Generate API Key.

Figure 5-14. Enter a server and path to generate a dev

In our case, we wanted to enable Google Maps for a single script on our server. If you whole directory, you can leave off the script name and just specify the hostname and Unfortunately, the API key isn't good for directories inside the one you specify, just th

Almost instantly, a key will be generated, along with an example web page that Google refers to this as on your web site, copy the HTML/JavaScript section in Example 5-1and paste it into a new file on your o when you created the developer's key.

Google Maps " Hello World"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script
       src="http://maps.google.com/maps?file=api&v=1&key=[your API key]"
       type="text/javascript"></script>
  </head>
  <body>
    <div id="map"
       style="width: 500px; height: 400px; border: 1px solid  #979797"></div>
    <script type="text/javascript">
    //<![CDATA[

    var map = new GMap(document.getElementById("map"));
    map.addControl(new GSmallMapControl(  ));
    map.centerAndZoom(new GPoint(-122.1419, 37.4419), 4);

    //]]>
    </script>
  </body>
</html>
```

Developer keys work only when they are used on a web page that lives in the server a you created the key. So you can't copy this listing and have it work until you change t own. In general, most of the code examples in this book will require you to substitute for them to work.

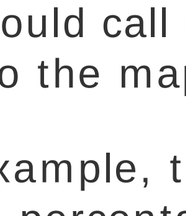
Hello, World!

The "Hello World" page shown in Example 5-1is a standard HTML page, with a bit of JavaScript. The firs element:

```
<script
```

```
src="http://maps.google.com/maps?file=api&v=1&key=[Your API Key]"
type="text/javascript"></script>
```

This imports the Google Maps JavaScript library into our page. A JavaScript-compliant browser will auto
URL. Google can then compare the developer's key and the server name and path that is included in the
records, to see if they match.



The `v=1` parameter in the above URL is important because it specifies the Google Maps
If Google ever changes its API in such a way that backward compatibility is broken, th
continue to function with the original API and give you some breathing room to update
the API.

The next three interesting lines are:

```
var map = new GMap(document.getElementById("map"));
map.addControl(new GSmallMapControl(  ));
map.centerAndZoom(new GPoint(-122.1419, 37.4419), 4);
```

These lines are pretty much self explanatory (for an object-oriented JavaScript programmer). But you d
maps on your own pages!

By default, the size of the map is determined by the size of the HTML element that contains the map. In
to define a division in the page, which provides an area that you can control and format independently f

The first line creates a new `GMap` object and places it within the `div` named `map` . (There's nothing magic a
waywe could call it "Tim," and so long as the JavaScript mentions the same name, it would still work.) T
control to the map, and the third line centers and zooms the map to longitude -122.1419, latitude 37.4

In our example, the `div` element is 500x400 pixels high and has a 1-pixel-wide gray border around the e
height in percentages, such as `style="width: 50%; height: 40%"` . The border itself is totally optional, but
of the page:

```
<div id="map"
style="width: 500px; height: 400px; border: 1px solid  #979797"></div>
```

The *demo.cgi* page at http://geocoder.us/ was already template driven, so to add Google Maps functiona
Google Maps library, and then included these lines in my template:

```
<div id="map" style="width: 500px; height: 300px; border: 1px solid  #979797"></div>
<script type="text/javascript">
//<![CDATA[

var map = new GMap(document.getElementById("map"));
map.addControl(new GSmallMapControl(  ));
map.centerAndZoom(new GPoint([% long %], [% lat %]), 4);

var point =  new GPoint([% long %],[% lat %]);
var marker = new GMarker(point);
```

```
map.addOverlay(marker);

//]]>
</script>
```

The map will automatically size itself to fit within the `<div id="map"...>` tag. In our templating system (P `%]` will be replaced with the contents of the variable `long` , or the longitude. The only differences from the for `lat` and `long` are replaced with variables that will be set in our program, and that a point marker is a looked up.

Getting Outside of Your Head

The "Hello World" example presumes that the HTML `script` element that imports the Google Maps API lib the HTML document's `head` element. Certainly, this is the right place for it to go, but web browsers are p elsewhere in an HTML document. Furthermore, situations will occur where you might want to include the one where you have an HTML templating system that provides a boilerplate header and footer for each p don't want the API library to be imported into every page on the site, because every page outside the di will load up with a developer key error message.

Fortunately, you can indeed import the API library almost anywhere in your document, so long as it app use it. The only thing you really need to know is that some browsersInternet Explorer, in particularwill w rendering the rest of the page, to make sure that the JavaScript itself doesn't modify the page layout. F a bad interaction with the Google Maps API when the script element is used outside the heada JavaScrip The workaround is to add a `defer="defer"` attribute to the script element, which will tell the browser not the page. In that case, our earlier script element example looks like this:

```
<script src="http://maps.google.com/maps?file=api&v=1&key   =[Your API Key] "type="text
```

Getting Right to the Point

Once you've got a Google Map on your page, adding points to it is easy. You'll first create a new `GPoint` o point, and finally add that marker to the map. We'll look more at adding points and lines to Google Maps which shows a pretty Google Map replacing our TIGER map.

Figure 5-15. http://geocoder.us/ with a Google

But is that (always) better? Are there reasons not to use Google Maps? Yes! Google Maps are great, and
the good guys, but it is a profit-making business and its goals might not be your goals. The Google Maps
when you use Google Maps, you are relying on Google. There are restrictions on what you can do with G
be used on a site that is inaccessible to the general public, such as a paid premium content site or a cor
volume, as well: if you expect more than 50,000 hits in a day, Google expects to hear from you first. Yo
Google's images or remove its imprint from its imagery, and it has explicitly reserved the right to put ad
more about the fine details at http://www.google.com/apis/maps/faq.html, but you should also review t
http://www.google.com/apis/maps/terms.html to be on the safe side.

There are (at least currently) limits on the data available from Google. There is far more aerial and satel
Web [Hack #12 in Google Maps Hacks] from publicWeb Mapping Service (WMS) servers than is availabl

See Also

- Google Maps are free-as-in-beer but not free-as-in-speech. So if the power, beauty, and ease of us
  projects such as Geoserver (http://geoserver.sf.net/), Mapserver (http://mapserver.gis.umn.edu/
  Mapserver (http://ka-maps.sf.net/) may fill the bill. The downside, as is often the case with open s
  more of the work yourself! O'Reilly's *Mapping Hacks* and *Web Mapping Illustrated* have much more
  solutions.

Rich Gibson and Schyler Erle

Hack 68. Map Flickr Contacts

By combining the Google Maps API and Flickr API, you can see how Flickr contacts are dispersed throughout the world, and learn how to make Google Maps with limited data.

Flickr (http://www.flickr.com) is a popular site for sharing photos. It's also a hub of online social activity. If you've ever traded photos at Flickr, you know you can connect with many other Flickr members from around the world. Flickr doesn't offer a way to show where all your fellow photographers are on a map, so it's the perfect candidate for a Google Maps mash-up.

As part of its member profiles, Flickr allows any member to include his location. The location can range from something very specific, such as a city and state, to something general, such as a country. Figure 5-16 shows my own Flickr profile, with the location "Corvallis, OR" listed directly under my web site.

Figure 5-16. Flickr profile page with a location set

With this information in hand, you can browse to Google Maps (http://maps.google.com), type in the location, and show a map of the general area of Corvallis, OR. However, this isn't quite enough information to plot a specific point on a Google Map with its API (http://www.google.com/apis/maps/). The jump from a general location to a specific point requires a

bit of geocoding.

Geocoding a Location

The Google Maps API needs information about points on a map in the form of*coordinates*, or listings of latitude and longitude that represent specific geographic points. Most of us don't know our current longitude and latitude, and Flickr doesn't expect us to. This is why the simple city/state/country location information that people enter into Flickr needs to be*geocoded* before you can build your map; that is, a listing such as Corvallis, OR, needs to be translated into its latitude and longitude pair: 44.564722, 123.260833.

Unfortunately, at the time of this writing, Google doesn't explicitly offer a geocoding service. However, you can use Google Maps to get the longitude and latitude of a place name by tweaking a Google Maps URL. A Google Maps URL such as the following returns a bit of JavaScript that's intende to be used in some Google Maps applications:

```
http://maps.google.com/maps?q=insert location &output=js
```

And the URL for your example location would be:

```
http://maps.google.com/maps?q=Corvallis,%20OR&output=js
```

Note that the space in the location has been encoded as`%20`, because spaces aren't allowed in URLs. Inside this page is some code that includes the center coordinates of the location:

```
viewport: {center: {lat: 44.564722,lng: -123.260833}
```

As you can see, you can enter a location and get back coordinates. With some simple scripting, you can automate this process and geocode a large number of place names. With this hurdle out of the way, you can simply assemble your list of places and plot them on a map. However, you must make sure you have what you need before you begin.

What You Need

The most difficult part of assembling this hack is making sure you have the prerequisites in place. You might have to spend some time gathering the pieces you need, but once your environment is set, building the map takes only a few minutes.

To use the Flickr API, you need a Flickr API key. You also need your own numeric Flickr user ID (NSID) so you can request a list of your contacts from the Flickr API. You can get both of these at the Flickr API page (http://www.flickr.com/services/api/).

It's important to note that your Flickr NSID is different from your Flickr username. You can look up your Flickr NSID anytime by logging in and browsing to the Flickr API documentation, as mentioned. You can also paste your photostream URL into a nifty service called idGettr (http://idgettr.com/) to find your Flickr NSID.

You also need a Google Maps API key associated with a public domain. For example, if you're going to view your map at a specific web address such as *http://www.example.com*, you need to register that domain with Google when you request your key. And if you're going to view your map at a subdirectory such as *http://www.example.com/path/to/map/*, you also need to specify the path when you register for a key. The requirement for a specific domain is a limitation of Google Maps, but it's a free service and you need to play by their rules. You can register your domain and get an associated key at http://www.google.com/apis/maps/.

If you're not planning to make your Flickr contacts map public, you can associate your Google Maps API key with the URL http://localhost/. This way, you can access the map on a web server running on your local machine.

The Perl script that writes the JavaScript for this hack requires several Perl modules that aren't usually preinstalled on most systems. `Flickr::API` (http://search.cpan.org/~iamcal/Flickr-API-0.07/lib/Flickr/API.pm) works with the Flickr API, and `XML::Parser::Lite::Tree::XPath` (http://search.cpan.org/~iamcal/XML-Parser-Lite-Tree-XPath-0.02/XPath.pm) helps parse the Flickr responses. You also need `URI::Escape` (http://search.cpan.org/~gaas/URI-1.35/URI/Escape.pm) for encoding locations for use in a URL and `LWP::Simple` (http://search.cpan.org/~gaas/libwww-perl-5.803/lib/LWP/Simple.pm) for passing those locations to Google Maps for geocoding.

The Code

The following Perl script does the heavy lifting for the hack by finding the locations associated with your Flickr contacts, looking up the coordinates for those locations at Google, and writing the necessary JavaScript to display the points on a map. The script also assembles the HTML necessary to display your contacts' buddy icons in the map itself.

Copy the following code to a file called *gmap-contacts.pl*, and be sure to include your own Flickr NSID and API key in the code:

```
#!/usr/bin/perl
# gmap-contacts.pl
# This script creates a JavaScript file that plots points
# on a Google Map for a given Flickr member's contacts.
#
# You can get a Flickr API key, find your NSID, and read the
# full documentation for the Flickr API at:
#
# http://www.flickr.com/services/api/
#
```

```
# You can get a Google Maps API key and read the full
# documentation for the Google Maps API at:
#
# http://www.google.com/apis/maps/

use strict;
use Flickr::API;
use Flickr::API::Response;
use LWP::Simple;
use XML::Parser::Lite::Tree::XPath;
use URI::Escape;

# Set your NSID
my $nsid = 'insert your Flickr NSID ';
my $api_key = 'insert your Flickr API key ';

# Start the API
my $api = new Flickr::API({'key' => $api_key});

my @users;
my @locations;

# Start the JavaScript file, and set the initial center point
# for the Google Map
print <<JSHEADER;
function addMapPoints(  ) {
    map.addControl(new GSmallMapControl(  ));
    map.centerAndZoom(new GPoint(-96.66, 40.817 ), 13);
JSHEADER

# Get a list of contacts
my $response = $api->execute_method('flickr.contacts.getPublicList', {
                'user_id' => $nsid,
});

# Make sure there's a response
if (!$response->{success}) {
    die "Contact list not found! $response->{error_message}";
}

# Parse the response and loop through contacts
my $xpath = new XML::Parser::Lite::Tree::XPath($response->{tree});
my @contacts = $xpath->select_nodes('/contacts/contact');
foreach (@contacts) {
    my ($location,$lat,$lon,$lonlat,$near);
    my $usernsid = $_->{attributes}->{nsid};

    # Find this contact's location
    my $user_res = $api->execute_method('flickr.people.getInfo', {
                'user_id' => $usernsid,
    });
```

```
# Make sure there's a response
if (!$user_res->{success}) {
    warn "\\nLocation for user $usernsid not found! $user_res->{error_message}";
}

# Grab location w/ regex
if ($user_res->{_content} =~ m!<location>(.*?)</location>!gis) {
    $location = $1;
} else {
    warn "\\nNo location found for user $usernsid";
}

# Find Lat/Lon for location at Google
if ($location ne "") {
    my $esc_location = uri_escape($location);
    my $url = "http://maps.google.com/maps?q=$esc_location&output=js";
    my $response = get($url);
    # Note if the location has a related longitude/latitude
    if ($response =~ m!center: {lat: (.*?),lng: (.*?)}!gis) {
       $lat = $1;
       $lon = $2;
       $lonlat = "$lon,$lat";
    # Otherwise, warn the user that the coordinates can't be found
    } else {
        warn "\\nNo coordinates found for location $location";
    }
    # Normalize the name, if possible
    while ($response =~ m!<line>(.*?)</line>!gis) {
       $near = $1;
       $near =~ s! \\d{5}!!gs; #Remove Zips
    }
}

# Save location to the locations array
my $exists = grep $locations[$_]{lonlat} eq $lonlat, 0 .. $#locations;
if (($exists eq 0) && ($lonlat ne "")) {
    my $thisLoc = {
        lonlat => $lonlat,
        lon => $lon,
        lat => $lat,
        near => $near,
    };
    push(@locations, $thisLoc);
}

# Save user to the user array
my ($iconserver,$username,$realname,$photosurl);
if ($user_res->{_content} =~ m!iconserver="(\\d{1,2})"!s) {
    $iconserver = $1;
}
if ($user_res->{_content} =~ m!<username>(.*?)</username>!s) {
    $username = $1;
```

```
        }
        if ($user_res->{_content} =~ m!<realname>(.*?)</realname>!s) {
            $realname = $1;
        }
        if ($user_res->{_content} =~ m!<photosurl>(.*?)</photosurl>!s) {
            $photosurl = $1;
        }
        my $thisUser = {
            nsid => $usernsid,
            iconserver => $iconserver,
            username => $username,
            realname => $realname,
            photosurl => $photosurl,
            lonlat => $lonlat,
        };
        push(@users, $thisUser);
    }

    # Loop through the locations, adding users for that location
    # to info window HTML
    for my $i ( 0 .. $#locations) {
        my $lonlat = $locations[$i]{lonlat};
        my $near = $locations[$i]{near};

        # Start HTML for info window
        my $html = "<b>$near</b><div style=\\"width:200px;\\">";

        # Find users for this location
        for my $j ( 0 .. $#users) {
            my $user_lonlat = $users[$j]{lonlat};
            if ($lonlat eq $user_lonlat) {
                my $nsid = $users[$j]{nsid};
                my $username = $users[$j]{username};
                my $realname = $users[$j]{realname};
                $realname = s!'!\\\\'!g; #Escape apostrophes for JavaScript
                my $photosurl = $users[$j]{photosurl};
                my $iconserver = $users[$j]{iconserver};
                my $iconsrc;
                # If no buddy icon is found, use the standard default
                if ($iconserver eq 0) {
                    $iconsrc = "http://www.flickr.com/images/buddyicon.jpg";
                } else {
                    $iconsrc = "http://photos$iconserver";
                    $iconsrc .= ".flickr.com/buddyicons/$nsid";
                    $iconsrc .= ".jpg";
                }
                $html .= "<div style=\\"float:left;padding:3px;\\">";
                $html .= "<a href=\\"$photosurl\\" title=\\"$realname\\">";
                $html .= "<img src=\\"$iconsrc\\" width=\\"48\\" height=\\"48\\"";
                $html .= " border=\\"0\\" />";
                $html .= "</a>";
                $html .= "</div>";
```

```
        }
    }
    $html .= "</div>";

    # Print out the line of JavaScript for this point
    if ($lonlat ne "") {
        print "\\tcreateMarker(new GPoint($lonlat),'$html');\\n";
    }
}

# Print the last character to close the JavaScript function
print "}\\n";
```

As you can see, the code is fairly complex, but it assembles everything you need for your map quickly. Because many Flickr users can be from the same location, the script stores both locations and Flickr member information in the arrays `@locations` and `@users`. At the end of the script, the last loop runs through the locations, building the HTML necessary to display buddy icons from members in each location.

Running the Hack

Once the script is ready to go, run it from the command line and specify a descriptive name for the JavaScript file, such as *addMapPoints.js*:

```
gmap-contacts.pl > addMapPoints.js
```

The script builds the JavaScript file and notes any locations it couldn't translate into coordinates. Next, you need to put together the standard HTML file that will hold the map. Create a text file called *contacts-map.html* and add the following HTML, being sure to include your Google Maps API key and the JavaScript file you just created in the header:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Flickr Contacts, Mapped</title>
    <script src="addMapPoints.js " type="text/javascript"></script>
    <script src="http://maps.google.com/maps?file=api&v=1&key=insert key "
       type="text/javascript"></script>
    <style type="text/css">
    body {
        font-family:arial,helvetica,sans-serif;
    }
    </style>
</head>
```

```
<body onload="addMapPoints(  ); ">
<h1>Flickr Contacts, Mapped</h1>
<div id="map" style="width:800px;height:600px;border:solid #C3CEAE 1px;);"></div>
    <script type="text/javascript">
    //<![CDATA[
    // Create a "tiny" green marker icon
    var icon = new GIcon(  );
    icon.image = "http://labs.google.com/ridefinder/images/mm_20_green.png";
    icon.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";
    icon.iconSize = new GSize(12, 20);
    icon.shadowSize = new GSize(22, 20);
    icon.iconAnchor = new GPoint(6, 20);
    icon.infoWindowAnchor = new GPoint(5, 1);

    var map = new GMap(document.getElementById("map"));

    // Creates a tiny green marker at the given point
    function createMarker(point,html) {
        var marker = new GMarker(point, icon);
        map.addOverlay(marker);
        GEvent.addListener(marker, "click", function(  ) {
            marker.openInfoWindowHtml(html);
        });
    }

    //]]>
    </script>
</body>
</html>
```

Note in the `<body>` tag that the page runs the `addMapPoints( )` function once the page has loaded. This functiongenerated by *gmaps-contacts.pl* and stored in *addMapPoints.js*initializes the Google Map and adds all the points and contact information.

Also note that, in this example, the initial center of the map is set to 96.66, 40.817, roughly the middle of the United States. If most of your Flickr contacts are in the UK, you might want to change this value so that you don't have to drag the Google Map to a new location to see your contacts' locations. You can change this value by editing the third line of*addMapPoints.js* or by changing the coordinates in the *gmaps-contacts.pl* script that writes that file.

Once both files are in place on your server, bring up*contacts-map.html* in a web browser, and you should see a map such as the one shown in Figure 5-17.

Figure 5-17. Flickr contacts on a Google Map

Click any of the green points on the map, and you'll see an info window containing the Flickr buddy icons that represent your Flickr contacts from that location. You can hover over a buddy icon to see the member's real name, or click the icon to view that member's photos at Flickr.

And, of course, you can zoom or click and drag the map, just as you can with any Google Map. You can even drag the map to other countries and see if you have contacts there. Most importantly, you'l know that with just a bit of geographic information, you can build fairlycomplex Google Maps.

Hack 69. Fly Across the Earth

Google Earth allows you to fly from point to point across the globe, studying its geography, cities, and points of interestall without leaving your computer.

Once you know your way around Google Maps [Hack #64], can find points [Hack #65] on Google Maps, and can even make your own Google Maps [Hack #66], you might think you've charted and visualized the world around you in every possible way. But Google takes mapping to another level with the desktop application Google Earth (http://earth.google.com).

Once you download, install, and start up this free application, you find yourself staring at the Earth from above, as shown in Figure 5-18.

Figure 5-18. Google Earth startup

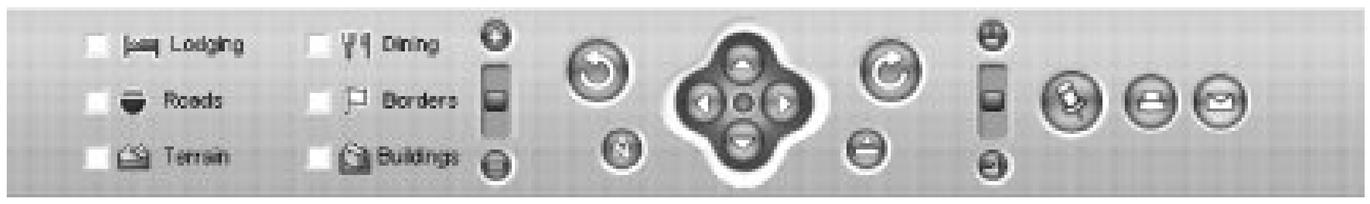
From this omnipotent vantage point, you can spin the earth like a globe, zoom in to see details, and use the program to explore the world around you in a number of different ways.

Navigating Google Earth

Google Earth provides a much more immersive experience than Google Maps because the Google Earth space is in three dimensions. Instead of looking over flat satellite images and maps that are always oriented north, you can spin, tilt, and twist the same data in Google Earth.

Figure 5-19 shows the Google Earth toolbar, and if it looks a bit like a airplane cockpit, that's because the experience of using Google Earth is much like flying.

Figure 5-19. Google Earth navigation bar



The four blue navigation buttons at the center of the bar allow you to move the terrain in front of you in different directions. To the left and right are arrow buttons that allow you to move the orientation of the terrain east or west, which can get confusing. If you ever need to get reoriented, click the N button (or type N on your keyboard) at the lower left of the navigation buttons to automatically orient the view to the north.

The slider at the far left zooms in and out of your current view, and the slider on the far right tilts the current view toward or away from the horizon. Playing with the tilt can also be disorienting. If you find yourself tilted beyond recognition, click the upright orientation button (or type U on your keyboard) at the lower right of the center navigation to automatically go into the more familiar top-down view.

> If you have a mouse with a click-wheel in the center, you can click the wheel and drag the mouse up or down to change the tilt of your current view. You can also spin the mouse wheel to zoom in and out of your current view.

On the far left of the navigation bar, you can choose to include features and points of interest. Here is a look at what each major control adds to your view:

Buildings

> With the Buildings option checked, you can zoom into major metropolitan areas and see 3-D renderings of the buildings in the city. Rotate the view to get a sense of how tall the buildings are in relation to one another, as shown in Figure 5-20.

Roads

> When checked, the Roads option highlights all roads and adds their names to the map. Figure 5-21 shows a view with Roads enabled.

Borders

The Borders option shows lines that divide different parts of the world, including the lines between countries, states, coastlines, and international boundaries.

Terrain

When the Terrain option is checked, you'll see differences in geographic elevation. You might need to tilt the map to see the differences in elevation, as shown in Figure 5-22.

Lodging and Dining

Check either of these options to highlight places to stay or places to eat on the maphandy when you're planning a trip or looking for a nearby place to get dinner.

The 3-D Buildings feature shows the three-dimensional nature of Google Earth. Figure 5-20 shows a close-up of Portland, Oregon, with the Building feature enabled and the view tilted toward the horizon.

Figure 5-20. Downtown Portland, Oregon, in Google Earth with 3-D buildings enabled

At the top left of Google Earth, you can enter locations to view, businesses to plot, and get directions from point to point. As you put in locations, Google Earth saves your searches and directions in a list under the search box. This way, you can easily access past locations. You can also right-click any location and choose Save To My Places to put a location into your permanent Places list, which is right under your recent locations.

Notice that the past-locations box on the left side of Google Earth has aPlay button. This allows you to animate a set of locations, flying to each one. This can help you get a sense of the terrain you'll be driving through, or simply let you feel like you're flying across the world.

For example, imagine you want to go from San Francisco, CA, to Portland, OR. You can get a list of detailed directions in Google Earth as you can with Google Maps. But if you highlight the Route and click Play in Google Earth, you can see the entire route from a bird's-eye view, twisting and turning as the road turns. Figure 5-21 shows the angle and details you see while flying over a route.

Figure 5-21. Flying over a route with directions in Google Earth

Once you've had the flyover tour, those complex merges and freeway maneuvers might not seem so complicated.

In addition to flying over a set of directions, you can highlight points in your saved Places box, click Play, and fly from point to point.

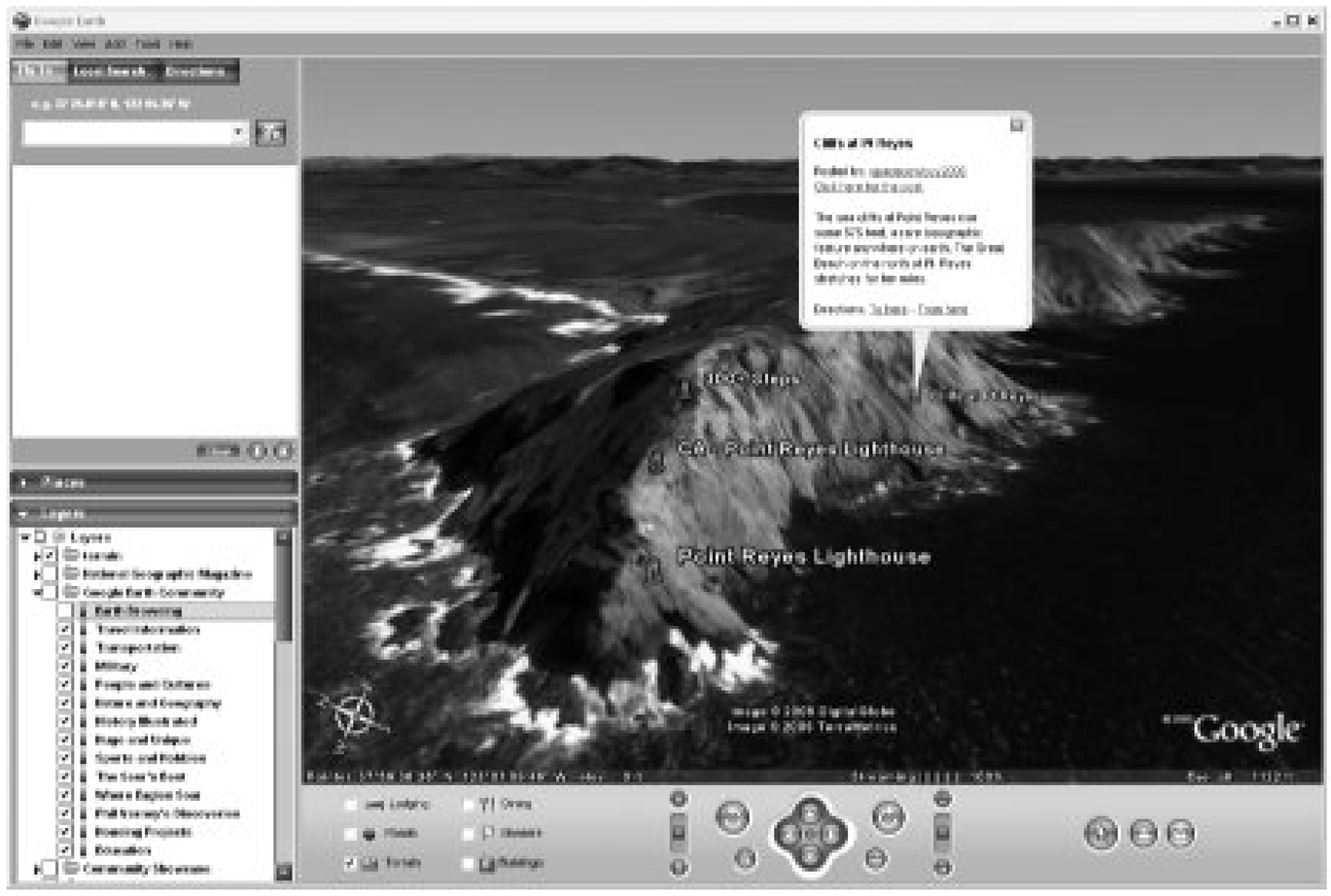
Adding Layers of Information

In addition to the features and points of interest you can enable in the navigation bar, you can add far more layers of data to your Google Earth display. TheLayers box at the bottom left contains dozens of types of data you can overlay on your view: everything from water features, schools, and railroad track locations to crime statistics, census data, and shopping malls. Take a look through the list of available layers, and you're bound to find some of interest.

Even beyond the extra layers, Google Earth provides a way for others to add their own points of interest to Google Earth. Enable the group of layers called Google Earth Community, and you'll find thousands of points of interest, bits of trivia, and even photos contributed by other users.

Figure 5-22 shows an interesting factoid about the cliffs at Point Reyes in California contributed by a Google user.

Figure 5-22. Viewing 3-D terrain in Google Earth, with extended data by the Google Earth Community

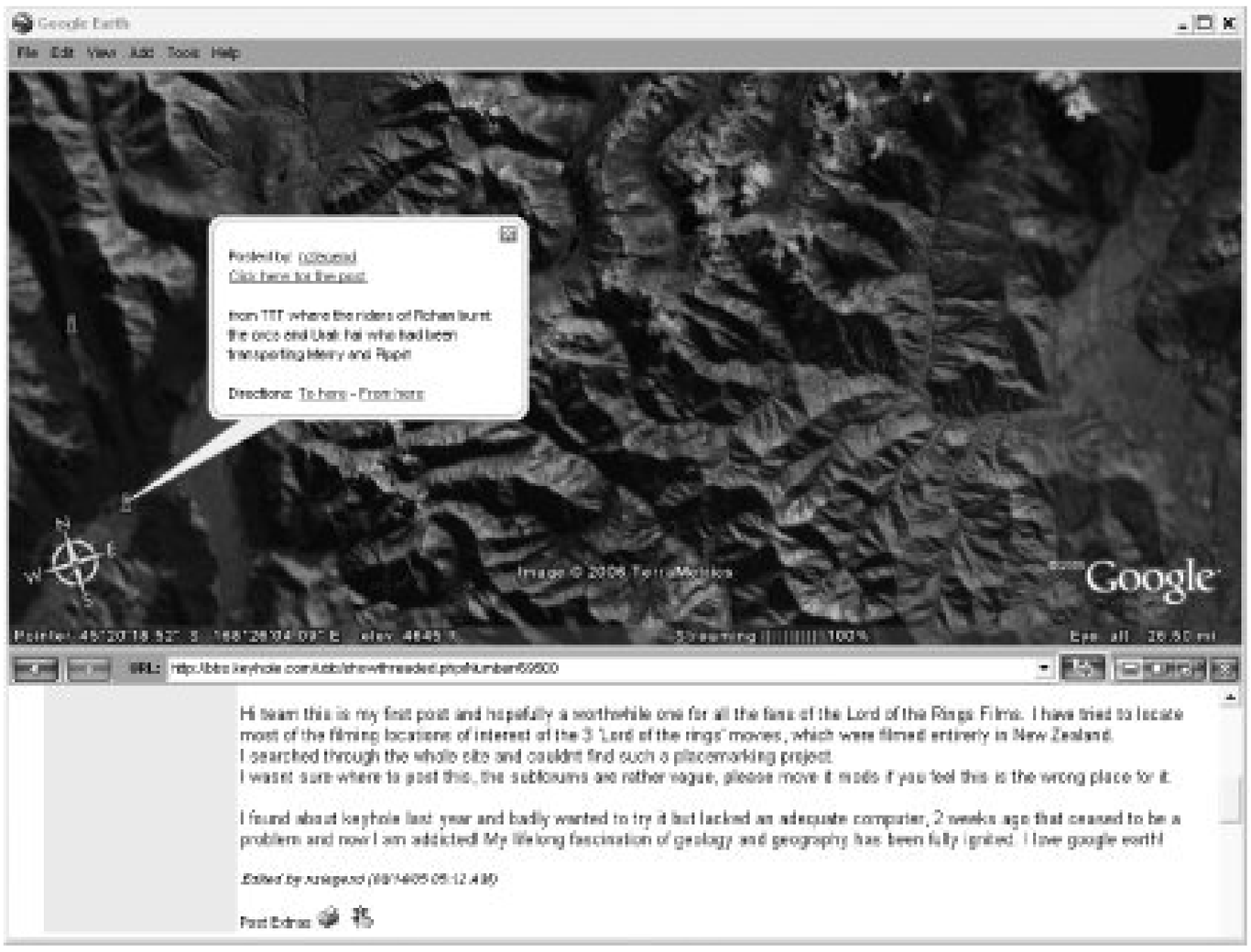


Google members can add arbitrary points for annotating spaces with extended information.Figure 5-23 shows a Google Earth post with a photo of the location to help other users visualize the spot in more detail.

Figure 5-23. Viewing user-contributed data in Google Earth

Some Google Earth users provide a specialized tour of an area through custom annotations.Figure 5-24 shows a single-point tour of New Zealand shooting locations from the*Lord of the Rings* movies.

Figure 5-24. Sightseeing in Google Earth



You can visit the post of a group of points with Google Earth's integrated web browser. Just click "Click here for the post" in the information window, and the Google Earth browser opens the original post about the group of points you're viewing.

Flying around Google Earth with the Google Earth Community enabled can provide hours of entertainment, teach you something new about the world, or simply give you some trivia about your next destination.

Sharing Your Locations

There are several ways to share your own knowledge of locations with others. The ability to share points with the Google Earth Community is built into the application itself. Simply gather a group of points and descriptions in your Places box, right-click the group name, and choose Share With Google Earth Community.

Though the process for sharing is simple, TheGoogle Earth Community site has some guidelines about what can be shared with others. Be sure to provide detailed descriptions about any points you add. A point on a map by itself isn't very valuable, but a point that also tells a story is very useful. And keep in mind that you can post images along with your description, which add another layer of information about a point.

If your points are of interest only to a small community, or even for only a couple of people, you can pass around Keyhole Markup Language (KML) files. These are XML-based files that describe points within Google Earth. To export a list of points as KML, highlight the point or group of points in your Places box, choose Save As..., and select the KML file extension. From there, you can publish the file

on a web site or send it directly to other Google Earth users via email. When you load a KML file in Google Earth, the points in the file are available for you to browse.

If you have a set of geographic data that you want to import into Google Earth, KML files provide a quick way to do so. Take a look at Google Earth's KML tutorial (http://www.keyhole.com/kml/kml_tut.html) for more information about building your own KML files.

Even if you never share a group of points with others, you're bound to learn something new about where we live by flying from point to point inGoogle Earth.

PREV

Chapter 6. Gmail

Google's web-based email service, Gmail (http://www.gmail.com), isn't your ordinary web mail service. Maybe you're attracted to the slick, interactive, real-application-like Ajax interface. Or the command-line jockey in you likes the Pine-like keyboard shortcuts (Pine is a text-only email application, typically found on Unix systems). Or is it the sheer volume of storageover 2 gigabytes, a the time of this writingthat's made you question your relationship with your existing web mail service and its puny 50-megabyte allotment? Most are enticed by the promise (and delivery, mind you) of a Google-like search interface to their email.

There was a day when a simple off-by-one (technically, an off-by-999) error caused quite a stir among early Gmail users. Logging into your Gmail account, you were met with the double takeworthy: "You are currently using 16 MB (0%) of your 1000000 MB." I'll see your gigabyte and raise you a terabyte.

Whatever your reasons for trying, switching to, or lusting after a Gmail account, you're sure to be delighted both by its proper and "improper" usesthe latter being the focus of this chapter.

As with all things Google, the official interface to Gmail is only one of many. Thanks to some clever screen scraping, analysis of the data model and format underlying the candy-coated frontend, and some good old tinkerer's enthusiasm, you can use Gmail for everything, from a filesystem [Hacks #75 and #76] to a backup server [Hack #77] to a mobile email account for Gmail on the go.

Signing Up

When Gmail first launched in 2004, it was available by invitation only, and people scrambled, scrounged, and wheeled-and-dealed to get accounts. Invitations went to the highest bidders on eBay and people swapped accounts for all manner of goods and services. Fortunately, Gmail accounts are much easier to come by today, but you still might need a bit of tenacity to secure one.

Google verifies all new email users by sending an invitation code to a mobile phone. Browse to the Gmail home page (http://www.google.com/gmail) and click "Sign up for Gmail using your mobile phone." Enter your cell phone number, and Google sends an invitation code to your phone. This security measure helps keep spammers out of Google's system, but it also adds a barrier to entry for those without mobile phones. If you don't have a cell phone that can receive SMS messages, you might need to turn to some old-fashioned wheeling-and-dealing to get an account.

Every Gmail user has a number of invitations they can send to others. Chances are, one of your alpha-geek friends already has a Gmail account. Ask nicely and be prepared to offer a latte or three. Also, email acquaintances with Gmail accounts are easy to spot: just look for the*@gmail.com* email address. Set up a filter in your email application to highlight any incoming Gmail and rifle off a response the moment you see one pop up. Your ingenuity and bravado are sure to be admiredand hopefully rewarded. You'll find the extra effort it takes to get an account to be well worth it.

Gmail Search Syntax

Gmail offers a rich search syntax for routing through your email message archiveas if you'd expect, or indeed stand for, any less:

`from:`

Digs through the headers of your email message archive in search of mail sent by someone matching the keyword you provide:

```
from:rael@oreilly.com
```

`to:`

The yang to `from:`'s yin, `to:` finds all messages sent to someone matching a provided keyword. (Don't forget plus-addressing [Hack #70].)

```
to:engineers@example.com
to:raelity+shopping@gmail.com
```

`subject:`

Matches messages with a particular subject:

```
subject:"meeting notes"
```

`label:`

Looks for messages with a particular label applied:

```
label:knitting
```

`has:attachment`

The `has:` syntax has only one possible value (at least at the time of this writing):`attachment`. `has:attachment` in a query returns only messages having one or more attachments:

```
has:attachment
```

`filename:`

Finds messages with an attachment filename that matches a provided pattern. Used with just a file extension (e.g., `pdf` or `txt`), `filename:` turns up all messages with attachments of a particular type:

```
filename:meeting_notes.txt
filename:pdf
```

`in:`

Returns a list of messages in a particular collection (read: folder). Acceptable values for `in:` are `inbox`, `trash`, `spam`, and `anywhere` (trash and spam are not included in searches unless they are explicitly included using `in:trash`, `in:spam`, or `in:anywhere`). Oddly enough, `sent` isn't a usable value for `in:`.

```
in:inbox
in:anywhere
```

`is:`

Acceptable values for `is:` are `starred`, `unread`, and `read`, which return starred, unread, and read messages, respectively:

```
is:read
```

`cc:`

Finds messages carbon copied to particular recipients:

```
cc:tara@example.com
```

`bcc:`

Finds outgoing messages blind carbon copied to particular recipients. Note that `bcc:` doesn't work on any incoming mail because there's no way to tell who's on the bcc line:

```
bcc:tara@example.com
```

`before:`

Matches messages sent or received before a particular date, specified in *`yyyy / mm / dd`* format. Unfortunately, partial datesyear only or year and monthdon't find anything at all:

```
before:2004/10/02
```

`after:`

Matches messages sent or received *on or after* a particular date, specified in *`yyyy / mm / dd`*

format:
`after:2004/11/21`

Phrase Searches

Enclose phrases in double-quotes (") to have the Gmail search treat them as a unit to be matched exactly (case isn't taken into account). The following query finds only accounting department reports

```
Subject:"accounting department report"
```

Basic Boolean

The only Boolean operator supported by Gmail search is `OR` (uppercase is required). In the absence of the `OR` operator, `AND` is implicit.

The Boolean `OR` operator works in Gmail searches just as it does in Google Web Search: specify that any one word or phrase is acceptable by putting an `OR` between each, such as in this query, which finds all messages from the boss or messages with subjects marked as urgent:

```
from:boss@example.com OR subject:urgent
```

Negation

The negation operator () also works as it does in Google Web Search, excluding messages matching the negated keyword or *`operator`* `:` *`keyword`* pair. So the following query turns up all messages to Example Co. *not* sent from the company's special offers department:

```
to:@examplecom -from:offers@
```

Grouping

Parentheses are used a little strangely in Gmail queries. When enclosing a set of words, they specify that each word must be found to be considered a match. So the following matches messages sent to both Sam and Mira:

```
to:(sam mira)
```

Throwing in an `OR` allows optional matches while being explicit about groups of options; while we humans tend to be able to parse precedence without the need of parentheses, search engines require a little more help. The following query finds all messages sent to Sam about rockets or helicopters:

```
to:sam subject:(rockets OR helicopters)
```

Mixing Syntax

Gmail's various search operators tend to play well together. While the tendency is to start out with minimal search criteria and keep whittling down, with a large number of email messages, crafting your searches can result in a lot of work. Take a chance and provide as much information as you can about the message you're after and back off bit by bit if you don't find it. The following query, for instance, is one that I just couldn't pull off in my computer's email client:

```
from:Duncan before:2004/10/01 subject:today "World Cup" lunch
```

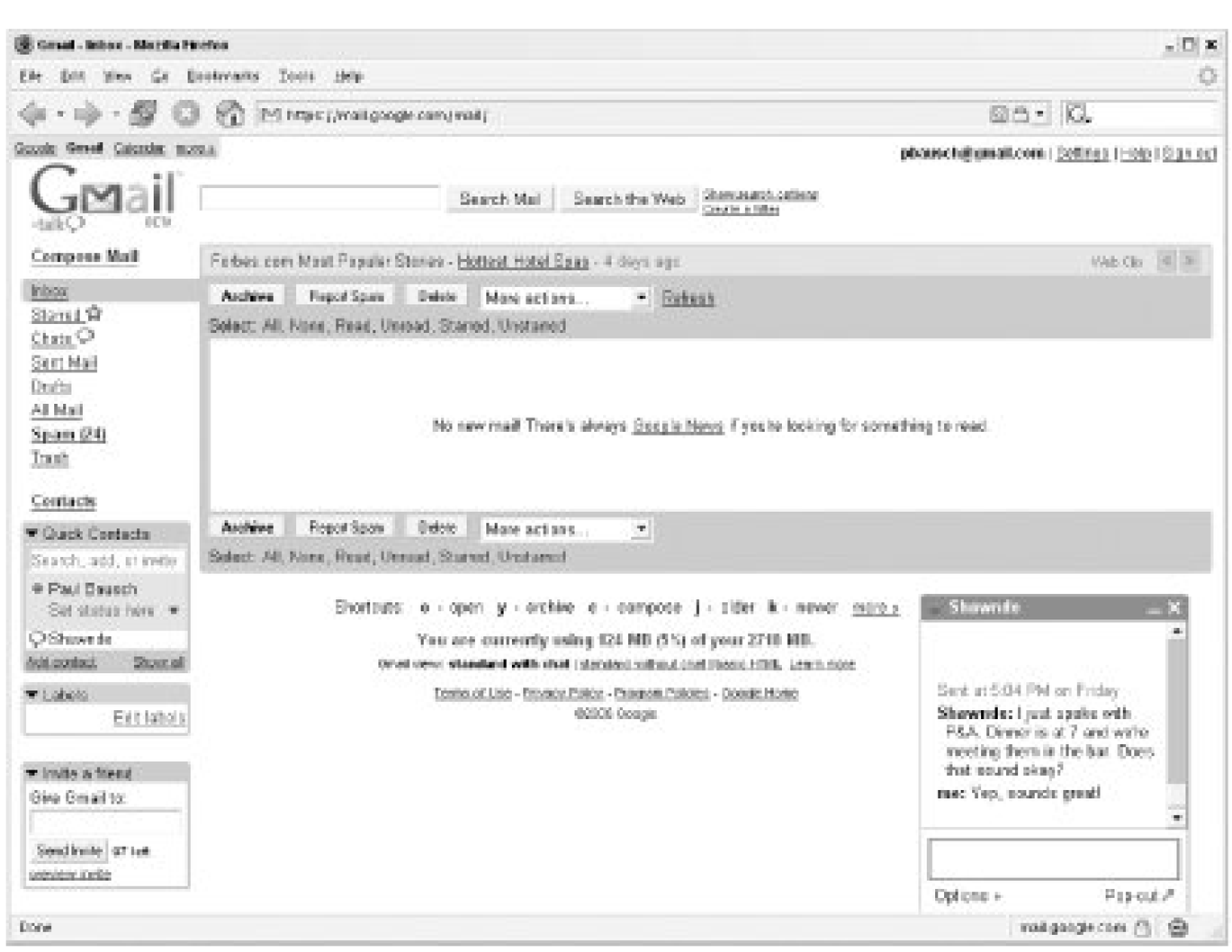
PREV

Gmail Chat

Google understands that some messages are more time-sensitive or chatty than others, and that an instant message is sometimes preferable to an email. This is why it's incorporated a light version of Google Talk [Hack #52] into its Gmail interface. If one of your contacts from your Quick Contacts list is also online, you can send her an instant message, without leaving Gmail.

When you click on a contact to send her an instant message, a new window such as the one shown ir Figure 6-1 pops up on the page.

Figure 6-1. Sending instant messages from Gmail



You can even click the Pop-out button at the bottom right to pop out the chat window from the Gmai page, but be aware that as soon as you navigate away from Gmail in your main browser window, your conversation will end. Gmail chat isn't a robust instant messaging client, but it is a handy way t have an instant conversation with your contacts.

Additional Resources

As with all Hacks books, what you find here is just a taste of what's most likely available by the time this book ends up in your hands. Here are a few more resources you might want to visit:

- The Gmail documentation (http://gmail.google.com/support) is chock-full of tips, tricks, keyboard shortcuts, search syntax, and more.
- Justin Blanton's "Getting More Out of Gmail" (http://justinblanton.com/2004/06/getting-more-out-of-gmail) provided much grist and many pointers for this chapter.
- GmailForums (http://www.gmailforums.com), as the name suggests, is a place to discuss all things Gmail.
- Mark Lyon, author of "Google Email Loader: Import Mail into Gmail," has collected a good list of apps and hacks (http://www.marklyon.org/gmail/gmailapps.htm).
- And, of course, you can always Google for `gmail hacks` and `gmail hacking`.

> Remember that all of these are hacks and, as such, have no quality-of-service guarantees; if they break, they break. About all you can do is go back to the hack's home page and see if there's a new version available.

Hack 70. Create and Use Custom Addresses

Make up an unlimited number of arbitrary email addresses to use when signing up for something, making a purchase online, or tracking a conversation.

Those who've been exposed to the power of a little something called*plus-addressing* never look back, using it anywhere and everywhere they can. And, for something so useful, there's really not much to it.

Simply append a plus sign (+) and some meaningful string of letters or numbers (meaningful to you, that is) to the first part of your email addressthe part before the "at" sign @)and you can tag a particular conversation, use the address to sign up for a service or buy something online, or create a throwaway address you have no intention of paying attention to again.

Say your email address is *raelity@gmail.com*. A plus-addressed version might be *raelity+shopping@gmail.com*. And you don't have to stop there; you can create subtags and sub-subtags such as *raelity+shopping+amazon@gmail.com* and *raelity+shopping+amazon+books@gmail.com* for even more granularity.

And the magic of it is that all plus-addressed email still arrives at the same email address: yours, sans the plus bit. At that point, you can filter, sort, highlight, or trash email sent to that particular address as you see fit.

Plus-addressing means never having to say you only have one email address again.

And you'll be glad to know that Gmail supportsplus-addressing, affording you some rather powerful email-handling, routing, and filtering functionality.

Some of my favorite uses of plus-addressing are:

Tagging a conversation

Keep track of a particular email conversationno matter how long it lastsby copying yourself (i.e., putting yourself in the Cc: field) with a plus-address (e.g.,*raelity+conundrum@gmail.com* or *raelity+tag+conundrum@gmail.com*). That way, as long as you're copied on any ongoing conversation, you'll know just where it all started (and, hopefully, ended).

Inviting people to a party

This is just a variation on the previous theme of tagging a conversation. Invite people to a party and copy yourself with a plus-address (e.g.,*raelity+scavengerhunt@gmail.com* or *raelity+rsvp+scavengerhunt@gmail.com*) to label and track RSVPs.

Signing up for services

Just about every online service requires you to provide an email address before you can sign up. If you never want to receive mail from these people again (aside from the initialand often requiredconfirmation email, that is), assign a plus-address to each service (e.g., *raelity+morningtimes@gmail.com* or *raelity+service+morningtimes@gmail.com*) or, if you've had quite enough of their follow-up messages, announcements, and special offers, set up a filter (http://gmail.google.com/support/bin/answer.py?answer=6579&query=filter&topic=&type=f) to direct them right into the Trash.

Buying things online

Buying things online usually involves some amount of email traffic: purchase confirmation, shipping notification, tracking, and problems. By assigning a plus-address to each vendor (e.g. *raelity+amazon@gmail.com* or *raelity+shopping+amazon@gmail.com*), you can group all your online transactions with that vendor.

While there usually isn't anything you can do about vendors and service providers sharing your email address with others, at the very least, you can keep tabs on the offending party.

Subscribing to mailing lists

There comes a time in any subscriber's life when she wants to disambiguate email pouring in from various mailing lists from more important mail. Give every mailing list its own plus-address (e.g., *raelity+xmlsomething@gmail.com* or *raelity+mailinglist+xmlsomething@gmail.com*) and you can label or siphon incoming mailing list posts into your Archive.

PREV

Hack 71. Import Your Contacts into Gmail

Data entry's a drag. Export your contacts from an existing web mail service, desktop email application, or database, and import them into your Gmail address book.

Possibly the most annoying aspect of moving into any new web mail home is bringing all your family, friends, and business contacts along with you. The average end user has been trained not to expect any sort of import utility, so he instead sighs and settles in for an evening of data entry.

Gmail, as with most post-1990s web mail applications worth their salt, provides a facility that imports all those contacts in just a few clicks; just how many clicks depends on where you're exporting them from. Gmail accepts only one format: comma-separated values (CSV). Thankfully,CSV is about as low a common denominator you could wish for; Yahoo! Address Book, Outlook, Outlook Express, Mac OS X Address Book (with a little help from a free application), Excel, and many other applications, web or otherwise, speak CSV.

> Gmail's Help documentation on the subject of importing contacts is sure to keep up with the needs of its users, so keep an eye on "How do I import addresses into my Contacts list?" (http://gmail.google.com/support/bin/answer.py?answer=8301).

Anatomy of a Contacts CSV

First, a quick tour of a typical contacts CSV file as consumed by Gmail's import tool.

CSV files, as the name suggests, are little more than garden-variety text files in which data is listed one record per line, each field separated by (you guessed it!) a comma. The simplest of all *contacts.csv* files might look something like this:

```
name,email
Rael Dornfest,rael@oreilly.com
Tara Calishain,tara@researchbuzz.com
...
```

The first line lists field namesin this case, name and email address. Each line thereafter is a single person or entity (business, organization, etc.) in your contacts list with a corresponding name and email address.

Gmail accepts various formats of contact entry, recognizing some of the more common fields such as name, email address, phone, birthday, etc. Here's a slightly more detailed*contacts.csv*:

```
first name,last name,email address,phone
Rael,Dornfest,rael@oreilly.com,(212) 555-1212
Tara,Calishain,tara@researchbuzz.com, (212) 555-1213
...
```

Notice that name is split into first and last-name fields, email is called`email address`, and there's a phone field too.

Unless you're going to be using Gmail as your main contacts databaseand I can't quite see why you wouldyou don't need to import more than name and email address (something akin to the first *contacts.csv* example) to find it useful.

At the time of this writing, Gmail does little with fields beyond name and email address but shove them into a Notes field. However, once imported, you can copy information from the Notes field to another field by hand, which could save you a few steps.

Feed CSV to Gmail

Assuming you have a CSV file to work with (if you don't, skip to the next sections for some guidance), importing is a snap.

From the main Gmail screen in your web browser, click the Contacts link (Figure 6-2) at the bottom of the menu on the left side of the page.

Figure 6-2. The Gmail navigation bar with Contacts link

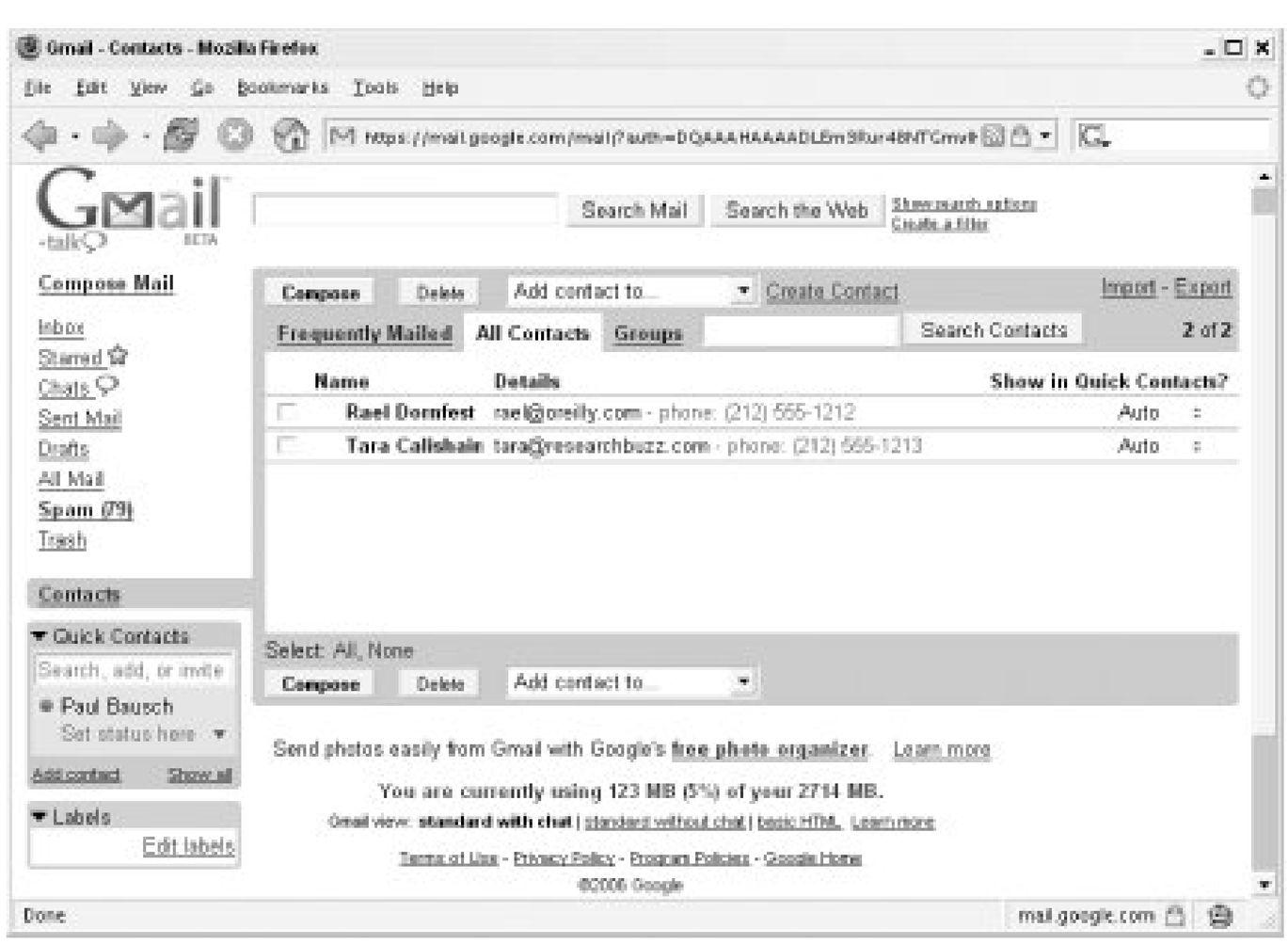
The Contacts page opens, listing all of (or none of, if you don't yet have any) your existing Gmail contacts. These may have been entered by hand, gleaned from incoming and outgoing mail, or imported at some earlier date. Click the Import Contacts link at the top right of the page, and a new window pops up with the Import Contacts dialog.

From the new window, click the Browse... (or equivalent) button when prompted to do so, as shown in Figure 6-3, and find your CSV file on your computer's hard drive. (Just what this looks like depends on your operating system and browser, but essentially you're just choosing a file much like you would from any application.) Click the Import Contacts button andBob's your uncle (that's "Tada!" for my American readers)you should see a confirmation that all went to plan and your contacts have been imported into your Gmail address book.

Figure 6-3. Finding that CSV file

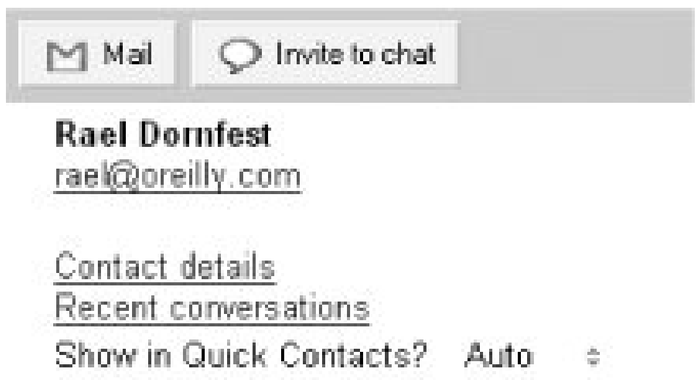
Click the Close link, and you'll see your now fully stocked contacts list. If you're looking at your Frequently Mailed contacts, you might need to click the All Contacts link above your addresses to see the new entries; after all, they're not frequently mailed yet![Figure 6-4](#) shows mine after I imported the second sample CSV at the beginning of this hack.

Figure 6-4. Feeding that CSV file to Gmail



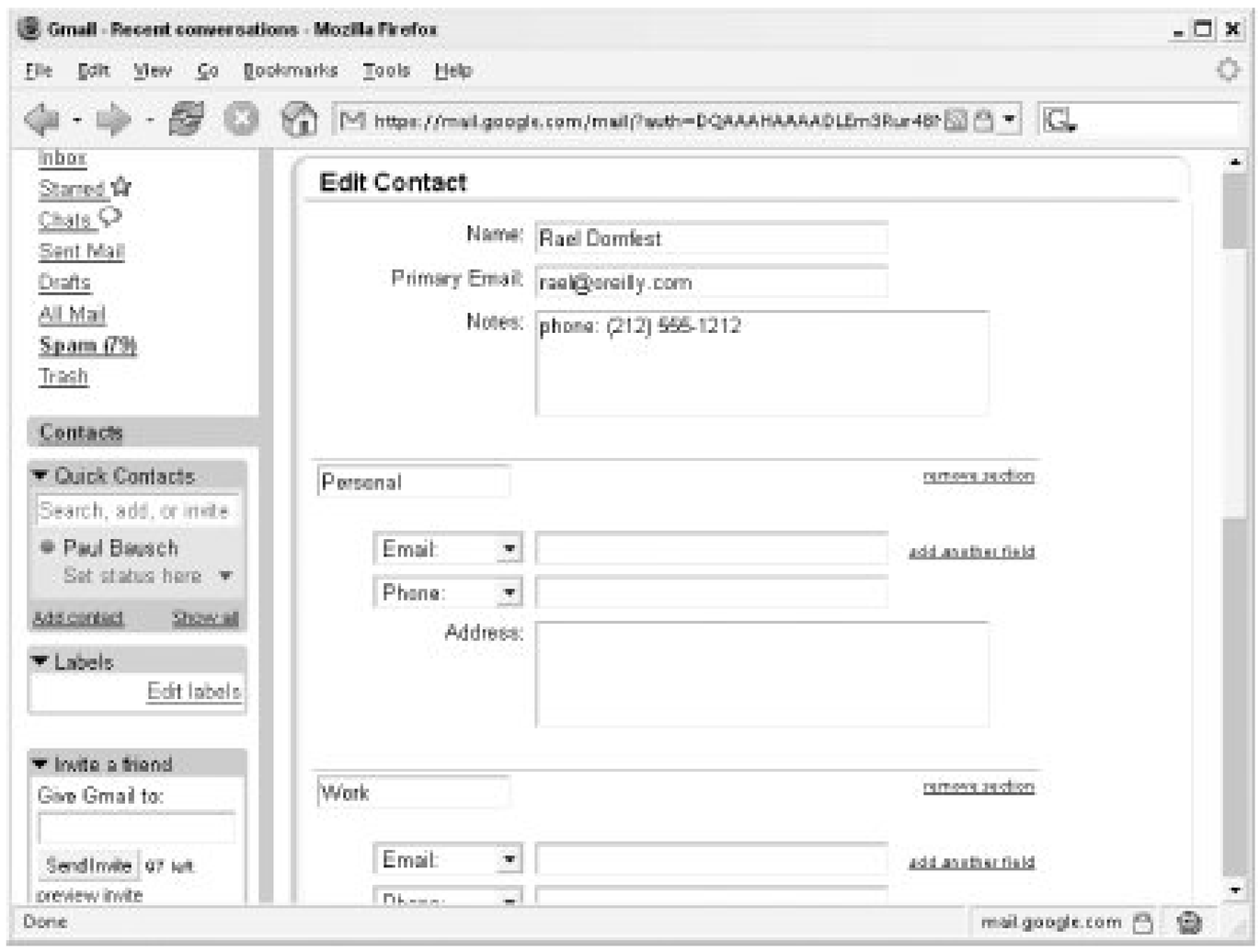
Delete any number of contacts by clicking their associated checkboxes and clicking the Delete button Edit a contact by hovering over the name, which brings up the hovering box shown in[Figure 6-5](#).

Figure 6-5. The hovering contact box



Click Contact Details then the edit contact information link to change anything about the listing. From the Edit Contact page, you can click Add More Contact Info to fill in details about the contact, such as work and home phone, fax, address, or just about any other bit of information you'd like to add. Figure 6-6 shows the Edit Contact page with extended information.

Figure 6-6. Adding extended information to a contact



And you can always type in a contact or three by hand using the Create Contact link on the front Contacts page.

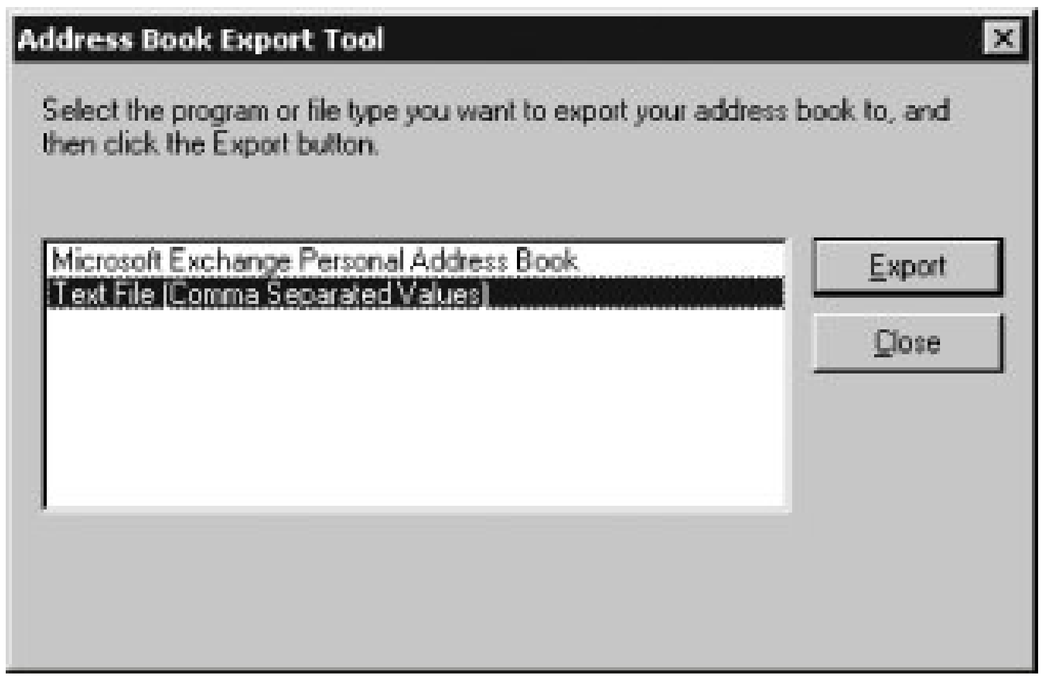
Now whenever you start typing a known contact's name into the To, Cc, or Bcc field of a new message, Gmail autocompletes it for you. No need to remember that cousin Adam is *adamg@ozziesurfers.co.au* or that Auntie Joan is *joan42@tepidmail.com*.

Out of Outlook (Express)

Both Outlook Express and Outlook in Windows can export their address books as CSV.

In Outlook Express, select File→Export→Address Book..., choose Text File (Comma Separated Values) as your output format (see Figure 6-7), and click the Export button.

Figure 6-7. Export your Outlook Express Address Book as CSV



In Outlook, select File→Import and Export..., choose "Export to a file," click Next, select Comma Separated Values (Windows) as your output format, and click Next again. An Export Wizard then guides you the rest of the way to saving your contacts as a CSV file.

Feed either to Gmail as described earlier.

Hopping Out of Hotmail

There are a couple ways to hop out of Hotmail with your contacts in tow. The first way involves Outlook Express or Outlook, and the second uses a touch of copy-and-paste, as suggested by the Gmail team in its online Help documentation.

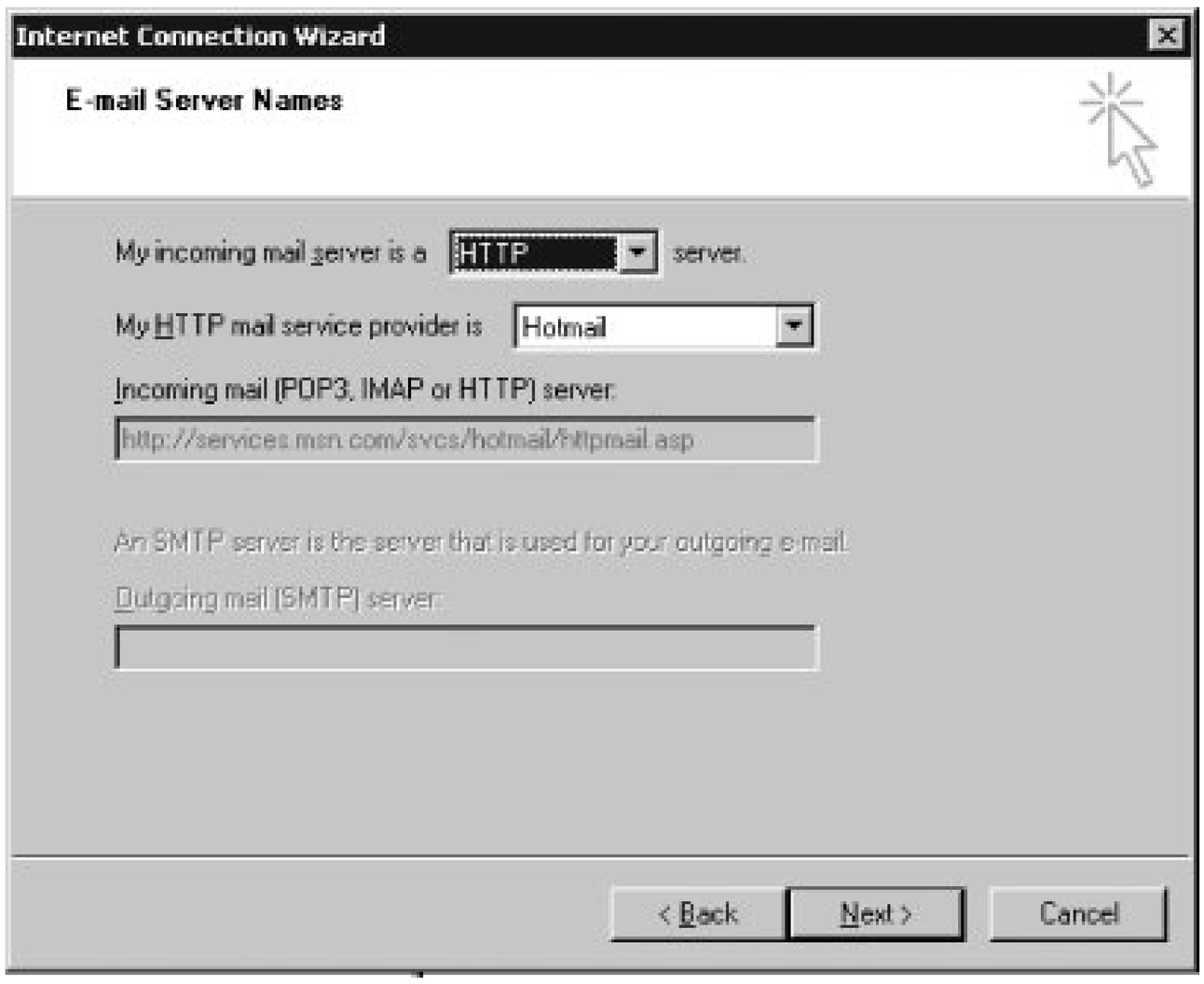
By way of Outlook (Express)

As described earlier, both Outlook Express and Outlook can export to CSV. They can also subscribe to Hotmail accounts and, if you have a Hotmail Plus subscription, synchronize contacts. Putting two and two together, you can use Outlook (Express) as an intermediary as follows.

> To use this method of exporting contacts, you need a subscription to Hotmail Plus (http://join.msn.com/Hotmailplus/overview-std), which costs $19.95/year. If you don't already have a Hotmail Plus subscription, you must go the copy-and-paste route.

Set up a new account in Outlook Express or Outlook, choosing HTTP as the server type and Hotmail as the mail service provider, as shown in Figure 6-8.

Figure 6-8. Setting up a Hotmail Plus email account in Outlook Express



In Outlook Express, click the Addresses icon in the toolbar to open your Address Book. Select Tools Synchronize Now (Figure 6-9) to synchronize your contacts between Outlook Express and Hotmail, thus bringing your Hotmail contacts to your computer.

Figure 6-9. Synchronizing with Hotmail to grab a local copy of your contacts

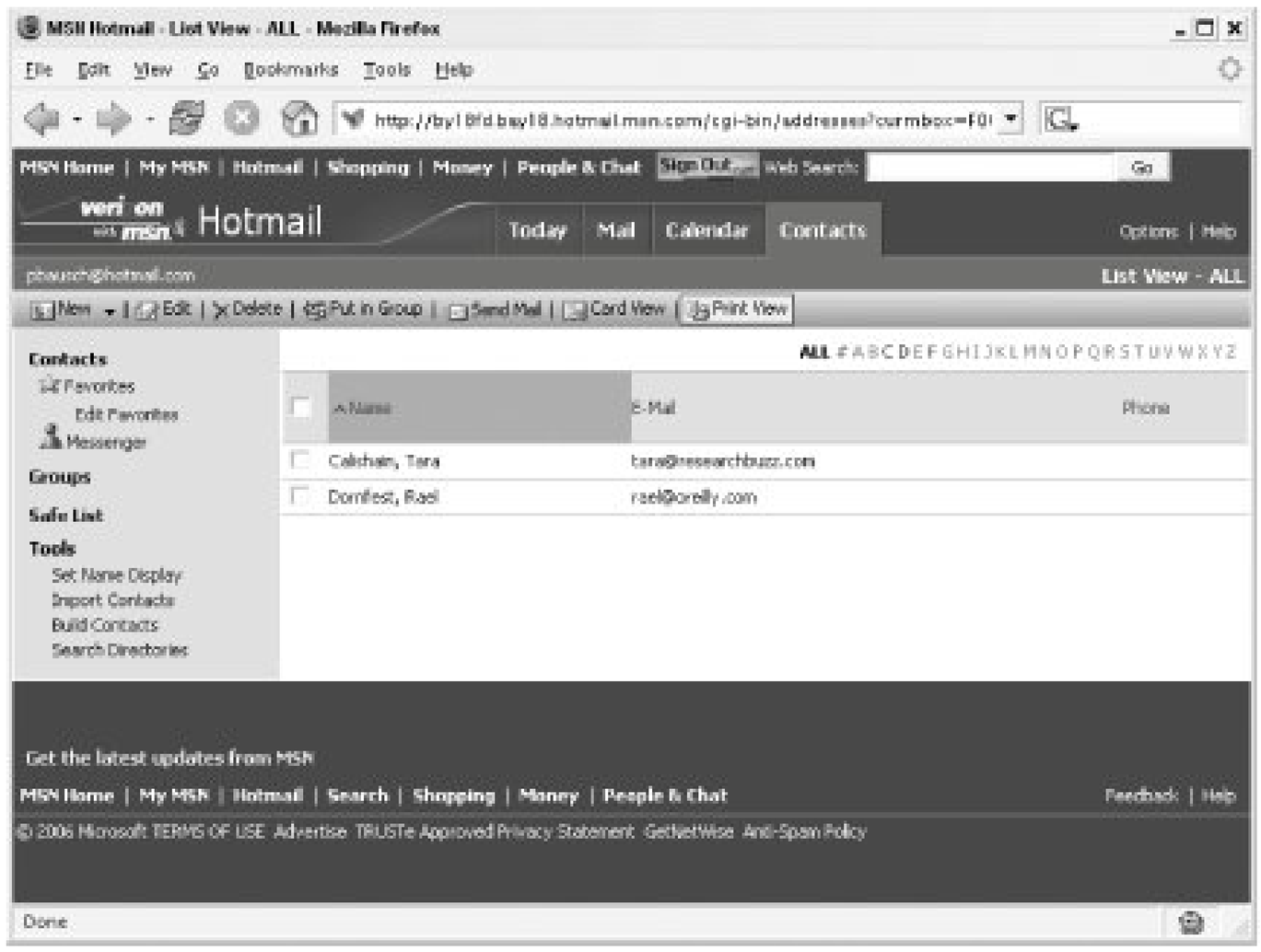
After a few moments of synchronization, your local Address Book will be up to date and you can export those contacts to CSV as described earlier in the "Out of Outlook (Express)" section.

By way of copy-and-paste

This is one of those ugly methods that you can't quite knock because it just plain works.

Log into Hotmail in your web browser of choice and select the Contacts tab, as shown in Figure 6-10. Click the Print View link in the Hotmail toolbar.

Figure 6-10. Clicking the Print View link in the Hotmail Contacts toolbar



In the Print View window that pops up, highlight everything (click and drag your mouse) from Name at the top left to the bottommost row in your list of contacts. Press Ctrl-C or select Edit Copy to copy the contacts, as shown in Figure 6-11.

Figure 6-11. Copying your contacts

Open Microsoft Excel, start a new workbook, select the A1 cell, type Ctrl-V or select Edit Paste Special..., and choose to paste as Text. Your workbook should look something like Figure 6-12.

Figure 6-12. Pasting your contacts into an Excel workbook

Save the workbook as "CSV (Comma delimited)" (ignore the warnings about incompatibilities that Excel throws at you) and give the resulting CSV file to Gmail's import tool.

This turns into an unholy mess under Mac OS X. Contacts are not nicely spread across columns, leaving you with a row of contacts, empty cells, and some odd characters in any CSV file you attempt to create.

Yumping from Yahoo!

Yahoo! Address Book exports directly to CSV.

Log into Yahoo! and visit your Address Book (the Addresses tab). Click the Import/Export link on the top right (Figure 6-13).

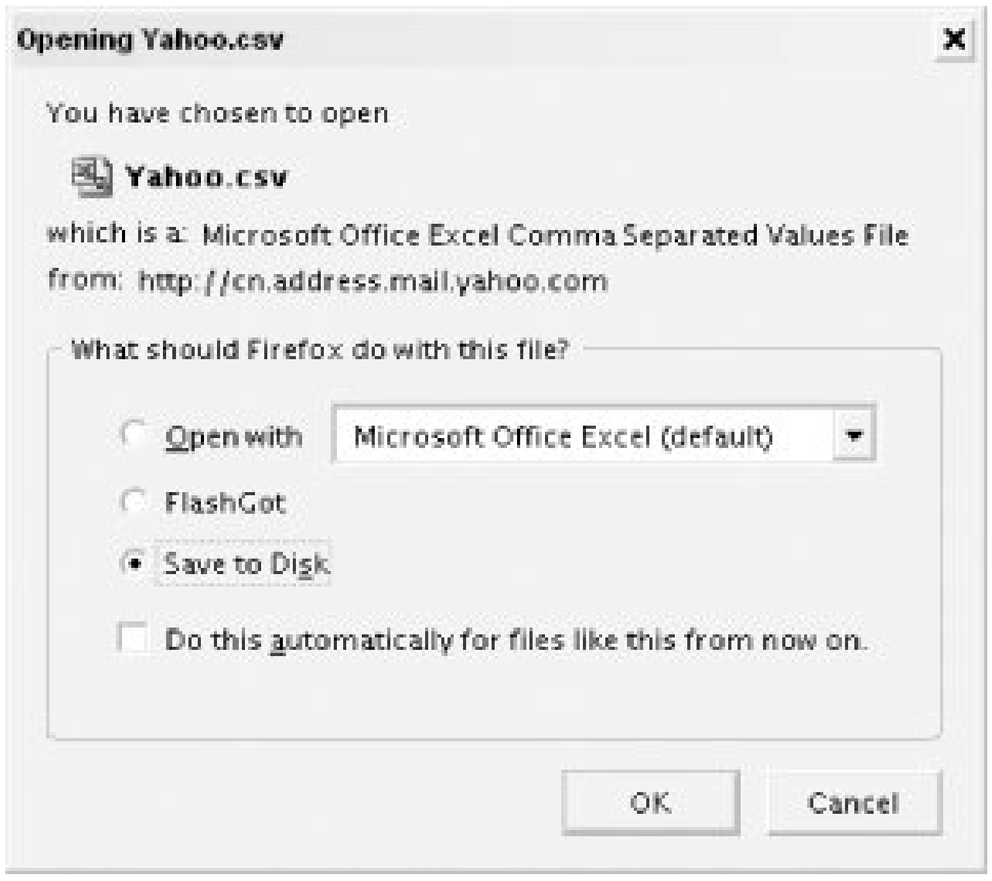
Figure 6-13. Using the Yahoo! Address Book's Import/Export feature

On the Export section of the resulting page, click the Yahoo! CSV Export Now button (Figure 6-14).

Figure 6-14. Exporting as Yahoo! CSV

Your browser will most likely prompt you for a place to save the CSV file on your computer's hard drive, as shown in Figure 6-15.

Figure 6-15. Saving the exported CSV file to your hard drive



Now go ahead and import that CSV using the Gmail import tool, described earlier.

> I do apologize for the bad "Yumping" pun, but "Yahoo!" doesn't leave you much room for alliterated action verbs: yodeling? yanking?

Moving from .Mac

The Mac OS X Address Book exports only to something called vCard, which is understood by many contacts applications, but not by Gmail.

Thankfully, someone's written a magical little app to help.AddressBookToCSV (http://homepage.mac.com/kenferry/software.html#AddressBookToCSV; freeware) slurps up all your contactsname and email address only, which is nicer to my mind than uploading a slew of data unnecessary for your Gmailing needsout of Address Book and spits them into a CSV file that you can feed to Gmail. Download the app, mount the *.dmg* on your desktop, and run it right from there, as shown in Figure 6-16. (If you'll likely use it again and again, go ahead and drag it into your *Applications/Utilities* folder.)

Figure 6-16. AddressBookToCSV exporting Address Book names and email addresses to CSV

When prompted to do so, choose a place to save the*contacts.csv* file and click the Save button. Close the application using Command-Q (it doesn't do so by itself when done).

Feed *contacts.csv* to Gmail as usual.

Hand-Crafting a CSV

If your contacts exist in some form with no obvious path to CSV, you can always export them in any way you can, arriving at some point at either a plain-text file that you can manipulate by handtedious, but possibleor something Excel can read. If you can get to Excel, you can get to CSV; t massage the data into a form similar to that discussed at the beginning of this hack, select File Save As... and save as "CSV (Comma delimited)."

Last-Ditch Effort

If, for whatever reason, you can't massage your contacts into CSV form or use Gmail's Import Contacts tool, there is a (admittedly grotty) way to get all your contacts to Gmail using email itself.

Send out a single email message, preferably one that announces your intention, to (on the To: line) your Gmail account (or one that forwards to your Gmail account), copying all your contacts on the Cc: line.

> You should probably batch these so you have some semblance of privacy; you don't want your family to see all of your business associates' addresses and vice versa. Send a separate message for contacts of a sensitive nature.

When you receive the message at Gmail, open it and choose "Reply to all." Write something explanatory again and send it off.

Gmail automatically adds to your contact list the names and email addresses of the people you send email to from Gmail, so you've just added all of those people to your Gmail address book.

Again, this is a rather annoying way (annoying to your friends, family, and business contacts) to get your contacts list to Gmail, so it should be regarded as a last-ditch effort.

Rael Dornfest and Justin Blanton

PREV

PREV

Hack 72. Import Mail into Gmail

Moving to Gmail doesn't have to mean starting from scratch. Forward mail in bulk from your computer or other web mail service to your Gmail account.

The most enticing feature of Gmail is probably its ability to perform Google-style searches on your own Inbox. Over two gigabytes of free space is intriguing, but it's not much when you consider that you have far more than that available to you on even your most outdated PC. And I'd warrant that not even its snazzy Ajax interface is enough to tear you away from your existing web mail service, which would force you to uproot and start over.

Gmail doesn't currently provide a way to import your existing email archive (web mail service or desktop mailbox). While you might have already considered forwarding all that mail to your Gmail account, just how to do soeven forwarding just the few hundred "important" messagespresents quite a problem.

Not so, thanks to hacks such as the Google Mail Loader for forwarding desktop mail and web mail intermediaries, YPOPs! for Yahoo! Mail and MSN email, and GetMail for Hotmail.

Forward Desktop Mail

The Google Mail Loader (http://www.marklyon.org/gmail ; GNU Public License) is a point-and-click application that reads your existing mail files on your computer and forwards the messages to Gmailone every two seconds, so as not to overload or otherwise annoy the Gmail servers. It does so without deleting mail from your local computer; a copy of each and every message is sent to Gmail. You can even set it to drop uploaded messages into your Gmail Inbox or Sent Mail folder.

GML is cross-platform and understands multiple mailbox formats:

- Mbox (used by Netscape, Mozilla, Thunderbird, and many other email applications)
- MailDir (Qmail and others)
- MMDF (Mutt)
- MH (NMH)
- Babyl (Emacs RMAIL)
- Microsoft Outlook, via a utility such as PST Reader (http://www.mailnavigator.com/reading_ms_outlook_pst_files.html), which converts Outlook's PST files to Mbox format

Installing the hack

Download the Windows or Linux/Mac OS X, source-only version (http://www.marklyon.org/gmail/download.htm). The Windows version is definitely the simplest versior to set up and use, requiring no prerequisites or other bits and pieces.

The source version assumes you have installed the Python scripting language and thePython Mega Widgets (http://pmw.sourceforge.net) toolkit.

The ins and outs of installing GML and all the prerequisites from source is beyond the scope of this book. If you need help, consult the documentation for Python (http://www.python.org) and Python Mega Widgets (http://pmw.sourceforge.net), or ask your local technical guru or system administrator.

If, on the other hand, you have Python on your system and don't much care whether the Google Mail Loader is a desktop or command-line application, skip ahead to the "Hacking the hack" section.

Running the hack

Since Google Mail Loader works directly with your email application's mailboxes, you need to figure out where they live before you can go much further. Consult your email app's preferences or documentatior or just dig aroundboth on your hard drive and by googling for`"outlook express " mailbox files location` , replacing `outlook express` with the name of your email program.

You also need to make sure your mailbox files are in a format that Google Mail Loader can read, as listed in the beginning of this hack. If there's any conversion to do, do it now. For instance, usePST Reader (http://www.mailnavigator.com/reading_ms_outlook_pst_files.html) to turn Outlook and Outlook Express PST files into DBX format.

With mailbox files in hand, launch Google Mail Loader by double-clicking*gmlw.exe* on Windows or typing `python gmlw.py` on the Unix or Mac OS X command line. Figure 6-17 shows Google Mail Loader running under Linux.

Figure 6-17. Google Mail Loader

Work your way down the settings on the left half of the GML window:

1. The default SMTP server (that's the sendmail server, the one used to send your messages to Gmail) of *gsmtp57.google.com* works for most users. If, for some reason, you are required by your local network administrator or Internet service provider to use their outgoing mail server, replace the default with the appropriate address. If your outgoing mail server requires authentication, click the Requires Authentication checkbox and fill in your username and password.

2. Click the Find button and point GML at your mailbox file. If your email application uses MailDir format, select any file inside your MailDir directory.

3. From the File Type pull-down menu, choose your mailbox type (Figure 6-18). There are two versions of Mbox format: one is stricter about the file format and is therefore more accurate, while the other is more lenient and works better on some Mbox files.

Figure 6-18. Selecting your mailbox file type

For some of the history of mailbox file formats, read JamieZawinski's "mail summary files" at http://www.jwz.org/doc/mailsum.html.

1. If you don't know what format your mail application uses, try Googling for`mail format pine` , replacing `pine` with your mail app's name. (Pine uses Mbox, by the way.)

2. GML can upload both your incoming and outgoing mail. Choose Mail I Received from the Message Type pull-down menu, and messages are dropped into your Gmail Inbox and appear to be from th original sender, just as they did in your email application's mailbox. If you choose Mail I Sent, the messages are relabeled as coming from your Gmail address and appear in your Gmail Sent Mail folder.

Gmail automatically labels incoming messages as Inbox. There's no way, unfortunately, for an external application to change this behavior, so messages imported as Mail I Sent are labeled as both Sent Mail and Inbox and appear in both places. Keep in mind that there is only one copy of the message stored, and sent mail is relabeled so as to appear to be from your Gmail address, not your old email address.

If you Archive the copy you see in your Gmail Inbox, it will appear only in Sent Mail (and Archive, of course).

1. Type in your full Gmail address (e.g.,*hank@gmail.com*).

2. Click the Send to Gmail button, and the application will start sending messages, one every two seconds. The delay is necessary to prevent flooding of Google's servers.

If you're interested in the details, click the Save Log button to save the contents of the output window to a file for later review.

There are, as with any hack of this sort, some issues worth noting:

- The timestamp of imported messages in your Gmail Inbox is the same as when the message was received by Google. Inside the message itself, however, the original date is still preserved. You car search for parts of dates to retrieve matching messages: `Aug 94` , for instance, finds all messages from August of 1994.

- The number of messages in your Inbox does not match the number GML reports as sent. This is because the number GML reports is the number of new threads, not of individual messages. Gmail automatically groups related messages as they arrive.

- Some people, especially users of Mozilla or Firefox, report problems with their Mbox files being corrupt. I have tracked down a Python script (http://www.marklyon.org/gmail/cleanmbox.py) that

cleans up most of these problems.

- Importing mail from Outlook is a bit spotty. I recommend one of two things: import your Outlook mail into Outlook Express and then into the open sourceThunderbird mail application (http://www.mozilla.org/products/thunderbird/), or use PST Reader or the like to convert your Outlook mail to Mbox.

Hacking the hack

If you're a command-line jockey or don't particularly relish installing the various prerequisites (Tk, Python Mega Widgets) necessary to run the graphical version of Google Mail Loader, there's a text-only version available at http://www.marklyon.org/gmail/old/default.htm .

The only requirement for the command-line GML is Python (http://www.python.org).

Here's a sample session with the older GML on the Mac OS X command line:

```
$ python gml.py
Mbox & Maildir to Google Mail Loader (GML) by Mark Lyon <mark@marklyon.org>

Usage: gml.exe [mbox or maildir] [mbox file or maildir path] [gmail address] [Optional
SMTP Server]
Exmpl: gml.exe mbox "c:\\mail\\Inbox" marklyon@gmail.com
Exmpl: gml.exe maildir "c:\\mail\\Inbox" marklyon@gmail.com gsmtp171.google.com

$ python gml.py mbox ~/Library/Mail/Mailboxes/1999.mbox/mbox 'hank+gml@gmail.com'

Mbox & Maildir to Google Mail Loader (GML) by Mark Lyon <mark@marklyon.org>

    1 Forwarded a message from  : someone@example.com

Done. Stats: 1 success 0 error 0 skipped.
```

Migrate from an Existing Web Mail Service

Despite attempts by your existing web mail service to entice you to stay, Gmail beckons with its gigabytes of storage, powerful search, rich web interface, and chance of grabbing a better email address than `raelity973@` . That said, you're loath to leave behind the last year or three's email. Well, in some cases, you can indeed take it with you, thanks to some nice donateware Web-to-POP mail utilities. These intermediaries operate in one of two ways:

- The utility sits between your desktop email application and web mail service, allowing you to download all your mail to your computer, after which you can use the Google Mail Loader to feed it

to Gmail.

- The utility combines these two steps into one, grabbing all your web mail and forwarding it in bulk to your Gmail account.

While there are no doubt any number of these utilities, two we stumbled across were GetMail and YPOPs!

If you use Hotmail, you have to pay for POP mail access to your web mail by subscribing to Hotmail Plus. Then you can download all your mail as you would with any other service and use the Google Mail Loader from there. Unfortunately, Hotmail no longer provides POP access for freeperhaps in response to the popularity of competitors such as Gmail.

If you use Yahoo! Mail, you can try out YPOPs! (http://yahoopops.sourceforge.net/ ; donateware), a POP mail proxy that sits between your preferred email application and Yahoo! Mail. It's available for Windows, Mac OS X, Linux, and Solaris. The Windows version self-installs, while the others require that you compile from source code, so they are a little more difficult for the uninitiated to get up and running.

Move any messages you want to download and carry across to Gmail into your Yahoo! Mail Inbox and mark them as unread.

On Windows, run YPOPs! after installation. A little icon appears in your Windows taskbar; double-click it to get to the settings, shown in Figure 6-19.

Figure 6-19. YPOPs! proxies POP mail requests

While you can go ahead and make a few changes in the settings, YPOPs! runs right out of the box without any further configuration.

Now simply set up a POP mail account as you would any other, except that you point to YPOPs! running

locally as your mail serverboth incoming and outgoing. The YPOPs! site has details on configuring most email clients, and you can find them by clicking the Configure Mail Clients link on the left side of the YPOPs! home page (http://www.ypopsemail.com).

Once you have downloaded all your web mail to your computer, use the Google Email Loader to send all the contents of your local inbox to Gmail.

See Also

- GmailerXP (http://gmailerxp.sourceforge.net ; donateware) is the be-all and end-all of Gmail/Windows integration, providing a full-featured frontend to your Gmail email, importing and uploading legacy messages to Gmail, new mail notification, and so on.
- gExodus (http://blog.codefront.net/archives/2004/06/23/gexodus-02-some-new-features-for-gmail-mbox-import-tool) is an Mbox-to-Gmail importer that's similar to Google Mail Loader, with the added feature of custom subject line prefixes for imported mail, which allows you to tag the subject of all imported mail with `[home-import]` , for example, and set up filters and searches for that group of emails.

Mark Lyon, Justin Blanton, and Rael Dornfest

Hack 73. Export Your Gmail

Back up or export your Gmail messages to your computer for safekeeping or offline reading.

You're nicely settled into your new Gmail account and may even have brought over all of your email [Hack #72] since time began. You're mailing up a storm, taking full advantage of the one gigabyte of storage space you're allotted.

So what if you decide Gmail actually isn't for you and want to move out again, either to another web mail service or back to the more traditional email application running on your computer? Or perhaps you just want a local archive of your Gmail for safekeeping, or for offline trawling when you're on a plane and desperately need a copy of that meeting report.

A nifty little archiving script packaged with the libgmail (http://libgmail.sourceforge.net) Python interface to Gmail [Hack #77] is just the ticket. It logs into your Gmail account for you, looks around prompts you to select a collection of messages to archive, and downloads them to your laptop or desktop.

Installing the Hack

There's really nothing to do beyond downloading *libgmail* and *libgmail-docs* (http://sourceforge.net/project/showfiles.php?group_id=113492; or click the Downloads link on the libgmail home page) and unstuffing the archives. In the expanded *libgmail-docs* folder, you'll find a file called *archive.py*; copy it to the *libgmail* program folder.

The only requirement for libgmail is Python (http://www.python.org).

Running the Hack

The important file you moved from libgmail's documentation is *archive.py*, a demo script that logs into Gmail, downloads your email messages, and saves them on your computer's hard drive in a format (Mbox) suitable for importing into many an email program.

On the command line (whether it be the Windows DOS-alike, Mac OS X's Terminal, or Unix shell), run the archive script like so:

```
$ python archive.py
```

You're prompted for your Gmail account name and password, after which libgmail logs you in:

```
Gmail account name:
               raelity

Password:

Please wait, logging in...
Log in successful.
```

There you are. At this point, you can choose to archive just what's in your Inbox (0), starred messages (1), all messages (2), drafts (3), sent messages (4), or a particular set of labeled messages (6 and 7 in my case). Choose the corresponding number and hit Enter:

```
WARNING:root:Live Javascript and constants file versions differ.
Select folder or label to archive: (Ctrl-C to exit)
Note: *All* pages of results will be archived.
  0. inbox
  1. starred
  2. all
  3. drafts
  4. sent
  5. spam
  6. foo
  7. Peeps
Choice: 2
```

Libgmail begins slurping your messages out of Gmail, one by one, and downloads them to an archive file in the current directory on your computer.

> As stated by the program at the outset, "*All* pages of results will be archived," meaning that all messages in the collection you've chosen will be downloaded, not just those that fit on a single page when you're looking at the collection through the standard Gmail web browser interface.

```
ff602fe48d89bc3 1 <b>Hello from Hotmail</b>
   ff602fe48d89bc3 1 Hello from Hotmail

ff5fb9c2829c165 1 Hello Gmail via Gmail Loader
   ff5fb9c2829c165 1 Hello Gmail via Gmail Loader

ff5691f7170cb62 1 Hello Gmail via Gmail Loader
   ff5691f7170cb62 1 Hello Gmail via Gmail Loader

ff3f4310237b607 1 Howdy gmail-lite
```

```
   ff3f4310237b607 1 Howdy gmail-lite

ff39c1fc71abbf1 1 Hello from Gmail mobile
   ff39c1fc71abbf1 1 Hello from Gmail mobile

...

fbd0c388dd1684e 1 Hello, Gmail
   fbd0c388dd1684e 1 Hello, Gmail

fbd0c1db3bcffe2 1 Gmail is different. Here's what you need to know.
   fbd0c1db3bcffe2 1 Gmail is different. Here's what you need to know.

Select folder or label to archive: (Ctrl-C to exit)
Note: *All* pages of results will be archived.
  0. inbox
  1. starred
  2. all
  3. drafts
  4. sent
  5. spam
  6. foo
  7. Peeps
Choice:

Done.
```

And you're done. Choose another collection to download and archive if you wish; otherwise, press Ctrl-C to stop the *archive.py* script.

Now if you look in the directory from which you invoked*archive.py*, you should see a new Mbox-format archive (the one you just created is *archive-all-1096849647.72.mbox*) of your chosen collection of messages, suitable for importing into many an email program:

```
jane:~/Desktop/libgmail-0.0.8 rael$ ls
ANNOUNCE                        constants.pyc
CHANGELOG                       demos
README                          libgmail.py
archive-all-1096849647.72.mbox  lib
```

See Also

- gmail.py (http://www.holovaty.com/blog/archive/2004/06/18/1751) is a simple Python interface to Gmail, focusing on exporting raw messages for backup and import.

Hack 74. Gmail on the Go

Whether you have the latest cell phone technology or an older cell phone with a text-only browser, you'll be able to take Gmail out into the world.

Web mail means never having to say you're sorry that you left your laptop at home. While I can't quite fathom it myselfI keep a lot I need on my laptop beyond basic emailthere are those that wander the world sans the very core of the mobile office. They're happy to use OP's (other people's). "Where there's a web browser, there's a way" is their credo, and for those who can swing it, more power to them.

Where this falls down for me is in the between times: dashing to a meeting without the latest agenda in hand (it's in my email inbox, but my laptop's in my bag and there's no wireless network in sight), meandering through a foreign city and wanting to keep in touch with the folks back home but without having to lug around a laptop, and other moments such as these. There are a few ways to keep up with Gmail on a mobile device, and the choice basically comes down to the type of connection your device supports.

Gmail Mobile

Google offers access to Gmail for small devices via Gmail Mobile. You can use the free service by pointing your phone or PDA to http://m.gmail.com.

You can also read a bit more about the service in your web browser at http://www.google.com/glm/gmail.

Technically, Gmail Mobile is designed for devices that have an XHTML browser.Blackberry's, Treos, and many of the newer mobile phones have XHTML browsers, but you need to check with your device manual to know for sure.

Instead of digging out your cell phone manual, you can browse to the Gmail Mobile URL in your phone or PDA and simply see if it works. If you openhttp://m.google.com on your phone's web browser and see a login screen, go ahead and log in. If you get an error message, it's likely your phone has a WAP browser, which Gmail Mobile doesn't support at the time of this writing.

Once you're logged into Gmail Mobile, you'll see a list of new messages and a menu for more options as shown in Figure 6-20.

Figure 6-20. Gmail Mobile on a Blackberry

From Gmail Mobile, you can do the basics: send and receive email, browse your contacts, and search through your mail. But it's definitely a stripped-down version, and some of the nice touches such as autocomplete when composing messages aren't available.

However, if you want 24/7 access to your mail no matter where you are, the official Gmail Mobile interface is your best bet.

Gmail-mobile PHP

As mentioned earlier, the browser experience on even the smartest of smartphones has a way to go. And most folks don't have any more of the Internet on their phones than a basic text-only WAP view of the world [Hack #51]. While WAP works to some degree, web mail services don't tend to spend much time, if any, providing a WAP interface to your emailthis is definitely true in Google's case.

But there's always a workaround....

The hyphenated Gmail-mobile (http://sourceforge.net/projects/gmail-mobile; GNU Public License) is a PHP (http://www.php.net) application that sits on your web site between your mobile phone's WAP browser and Gmail, brokering requests on your behalf and returning a mobile-appropriate view of your Gmail mail.

This hack assumes you have an account that allows WAP access to the wild, woolly Web from your mobile phone. Check with your mobile operator about your data plan, and don't forget to ask what you're charged per megabyte, because even the lightest of interactions can add up over time.

You can catch a quick status update, read, and even reply to your Gmailand more features are promised. Gmail-mobile predates Google's official mobile offering, and the Gmail-mobile developers are moving forward with the project despite the competition.

Installing the hack

Installing Gmail-mobile is a piece of cake; under both Mac OS X and Linux, I installed it in a matter o seconds.

Gmail-mobile assumes you have PHP installed on a web server running on port 80 (the WAP, and indeed web, default). You also need the curl library (http://www.php.net/curl), which Gmail-mobile uses to talk to Gmail over the Web, and the libgmailer [Hack #77] library, included for your convenience in the Gmail-mobile distribution.

Download Gmail-mobile (http://sourceforge.net/projects/gmail-mobile) and unpack the distribution (1.2 at the time of this writing, but yours is sure to be a later version) somewhere under your web server's document root, where the rest of your web site lives (ask your system administrator or service provider if you're not sure where this is).

You might want to rename the directory something that's easy to type on a mobile phone keypad, such as *gm*.

And you're done. No, really, I was surprised too at just how easy it was.

By default, Gmail-mobile uses browser cookies to maintain state between requests to Gmail's servers. If you have PHP Session (http://www.php.net/session) installed, you can choose to use it instead of cookies. Just comment out the appropriate line in the*config.php* file in your newly unpacked *gmail-mobile* directory. Here, I've left things as they were, using the cookie default:

```
<?php
```

```
    require_once("libgmailer.php");

    /** Session handling method. You must at least choose (uncomment) one. **/

    /**** have PHP Session installed, prefer to use cookie to store session **/
    //$config_session = (GM_USE_PHPSESSION | GM_USE_COOKIE);
    /**** have PHP Session installed, prefer NOT to use cookie **/
    //$config_session = (GM_USE_PHPSESSION | !GM_USE_COOKIE);
    /**** do not have PHP Session installed **/
    $config_session = (!GM_USE_PHPSESSION | GM_USE_COOKIE);

?>
```

Running the hack

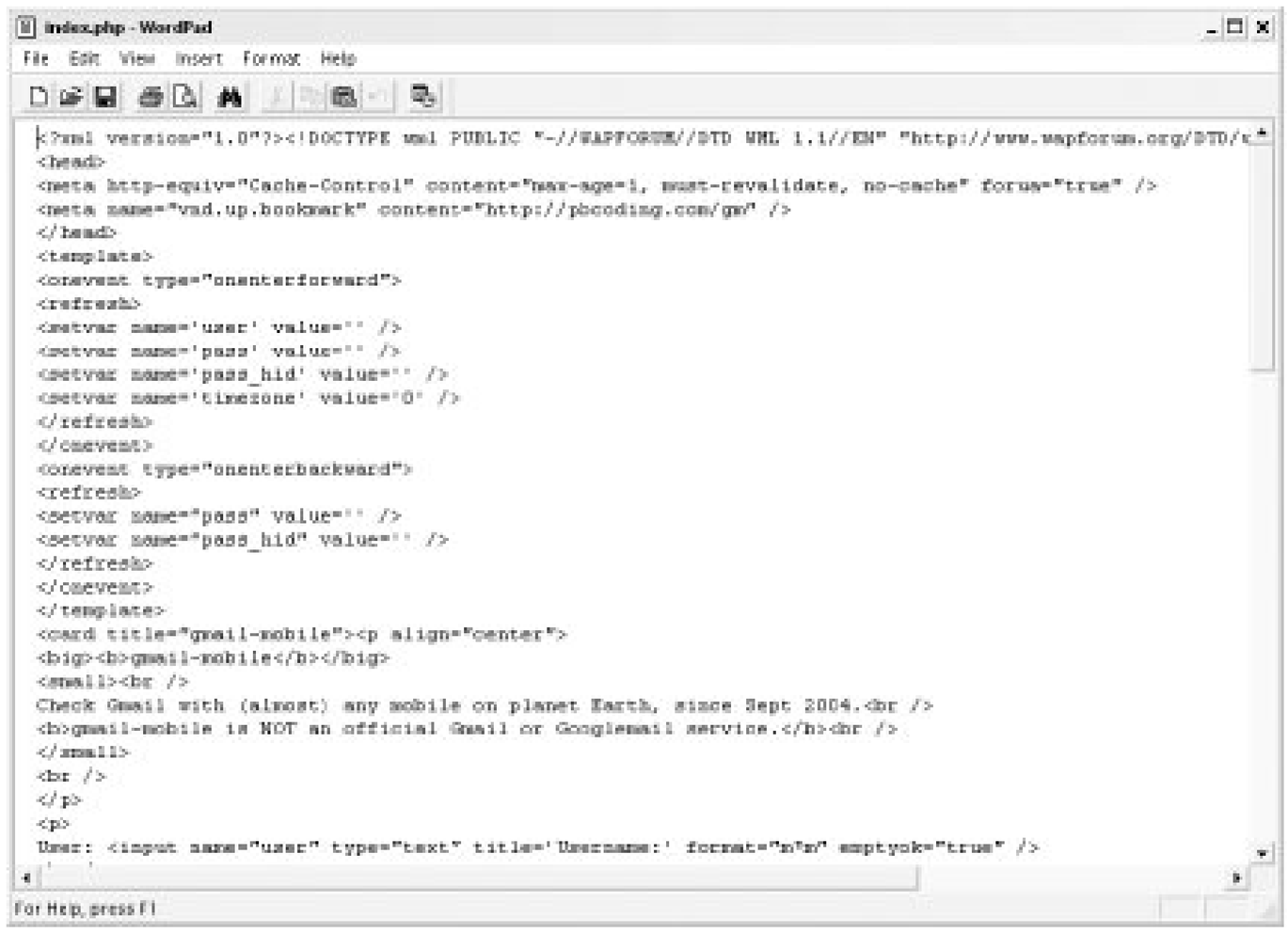
With the easy part out of the way (isn't it wonderful when installation and configuration are the easy part?), you're ready to break out your mobile phone's browser and muddle through typing on that tiny keypad.

Before trying this out from your mobile phone (and to remove one variable in case something doesn't work as expected), point your computer's web browser at a URL corresponding to the *gmail-mobile* directory on your web sitee.g., *http://www.example.com/~rael/gm*.

> You may actually need to tack on */index.php* to that URL, but most PHP-enabled servers know to look for and serve up *index.php* as a default when no filename is specified and there's no static *index.html* in sight. The Gmail-mobile package includes just such an *index.php* file.

Your browser responds in one of two ways. Either it serves up the rawWML source delivered by Gmail-mobile, as shown in Figure 6-21, or it throws up its hands in confusion and prompts you to save the source as a file on your hard drive. If the source (displayed in your browser or saved and opened using something like TextEdit on Mac OS X or Notepad on Windows) looks something like Figure 6-21 and doesn't seem to report any PHP or other errors, you're ready to switch to your mobile phone.

Figure 6-21. Raw Gmail-mobile WAP as viewed through a text editor



Launch your mobile phone's WAP browser and key in the appropriate URL to reach the*gmail-mobile* directory on your web site, as above.

After a few moments of churning (WAP is lightweight, but most mobile bandwidth is on the light side too), you should be greeted with a login screen (Figure 6-22). Key in your Gmail login (*username*`@gmail.com`) and password, alter the time zone if you feel so inclined, and click "Sign in." Just where you find the "Sign in" link varies from phone to phone, WAP browser to WAP browser.

Figure 6-22. Logging into Gmail-mobile from your mobile's WAP browser

A few more moments of churning, and you should see a summary view of your Gmail account (Figure 6-23, left). To visit any of the folders, navigate over the appropriate link and select it, much as you would links in a regular browseralbeit with esoteric keystrokes rather than a mouse.Figure 6-23, right, shows my rather empty Inbox.

Figure 6-23. Taking a gander at a summary of the state of your Gmail (left) and your Inbox (right)

Visit any message in any of your mailboxes by selecting its link. Compose a new message by selecting the Compose link; reply using the Reply link at the bottom of a message.Figure 6-24 shows the composition window in action.

Figure 6-24. Responding to Gmail mail on the go

It's not the spiffy tricked-out Gmail interface that you've come to expect, but it's a great way to take your Gmail with youand, quite frankly, it's better than some of the mobile email applications that I've come across.

See Also

- The Gmail-mobile project has on its to-do list just about anything you could wish for, including search, archive, delete, forward, label, mark as spam, and working with the Gmail address book [Hack #71]. Keep an eye on the project page (http://sourceforge.net/projects/gmail-mobile) for the latest news and distributions.

Other Options

If you cringe at the thought of installing your own PHP application, you might want to see if your phone has POP mail access. Gmail provides free POP access to your account, and if a POP email client is available on your phone, setup should be quite a bit easier than installing Gmail-mobile.

From your main Gmail page, click Settings in the top right and then choose Forwarding and POP from the Settings page. Choose Enable POP for "mail that arrives from now on," unless you want to send all of the email you've ever received to your cell phone. Then choose how you want Gmail to handle mail you've seen via your phone and click Save Changes.

Start up your phone and go into your email settings. If possible, add a new account and then add the following settings:

- Server: `pop.gmail.com`
- SSL: `Yes`
- POP Port: `995`
- SMTP Server: `smtp.gmail.com`
- Authentication: `Yes`
- SMTP Port: `465`
- Account name: *`your account`*`@gmail.com`
- Password: *`your gmail password`*

Each phone has a different method for entering these settings, and you might need to play around with the settings area to find them all.

Once your settings are entered, try checking for new mail. Your phone should contact the Gmail server and download any new messages. You might need to send yourself a test message from another account to verify that everything is working.

This setup process is a good example of the hassle you go through using POP access instead of a WAP or web-based access. Not to mention an inconsistent interface across devices. But it is a handy way to get mail if you don't have the web option.

If you're new to mobile browsing, you might want to take alook at some of Google's other mobile features [Hack #51].

PREV

Hack 75. Use Gmail as a Linux Filesystem

Repurpose your gigs of Gmail as a networked filesystem.

What I wouldn't give for a couple spare gigs of networked filesystem on which to stash a backup of my work in progress or as an intermediary between two firewalled systems (thus, they're not directly reachable from one to the other).

GmailFS, found at:

http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html

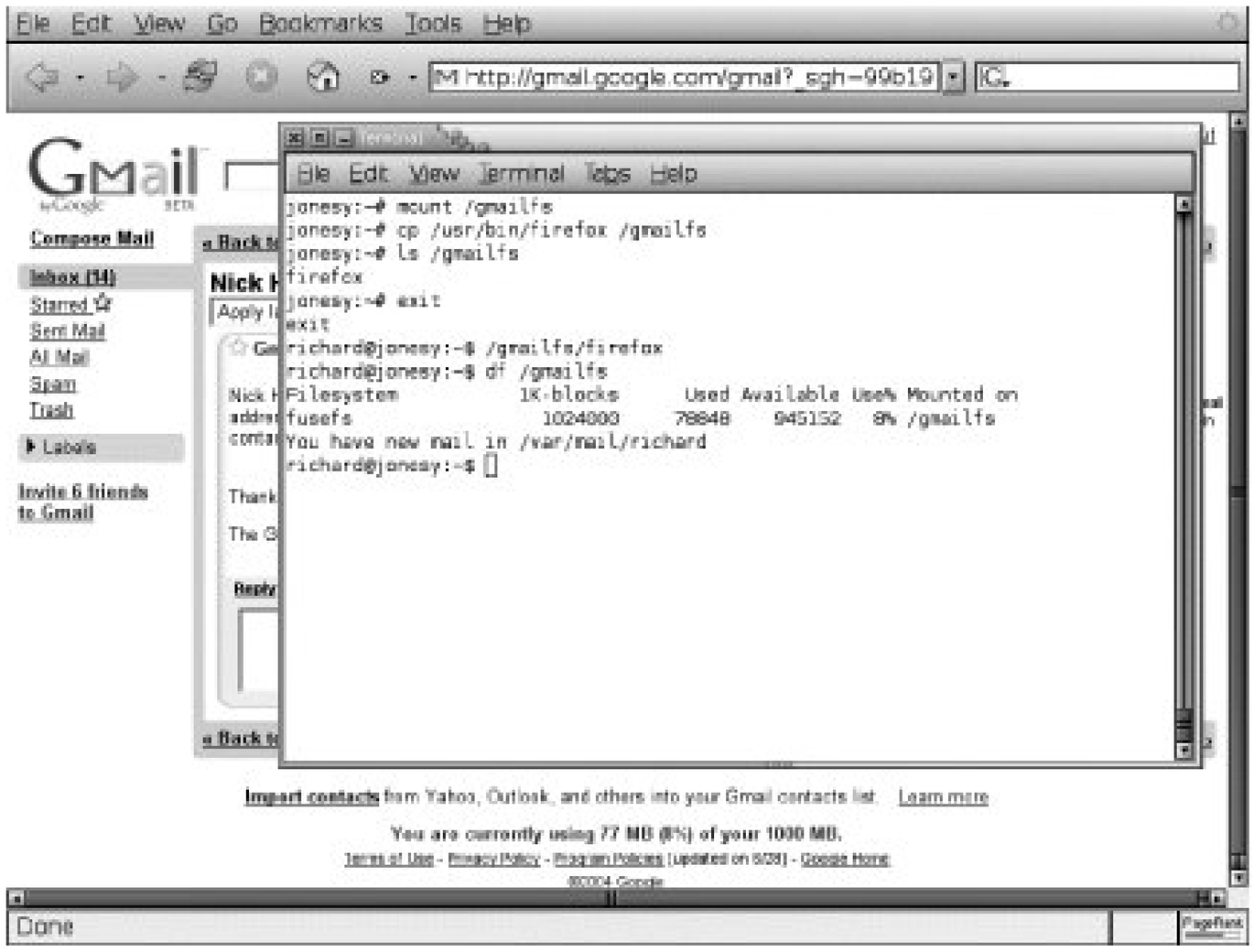
puts your gigs of Gmail storage to work for just such a purpose. It provides a mountable Linux filesystem and repurposes your Gmail account as its storage medium.

GmailFS is a Python application that uses theFUSE (http://sourceforge.net/projects/avf) userland filesystem infrastructure to help provide a filesystem and the libgmail[Hack #77] library to communicate with Gmail.

GmailFS supports most file operations, such as read, write, open, close, stat, symlink, link, unlink, truncate, and rename. This means you can use the lion's share of your favorite Unix command-line tools (`cp`, `ls`, `mv`, `rm`, `ln`, `grep`, et al.) to operate on files stored on Google's Gmail servers.

So, what can you store on the Gmail filesystem, and what can you do with it? About anything you ca with any other (possibly unreliable) networked filesystem built on a cool hack or three.Figure 6-25 shows the Firefox web browser launched from an executable stored as a message in my Gmail account.

Figure 6-25. Reading my Gmail via the Firefox web browser launched from an executable stored on the selfsame Gmail account



This is my first foray into Python, and I'm sure the code is far from elegant. That said, the language has a reputation as an excellent choice for rapid prototypingand this was borne out in my experience. The first working version of GmailFS took about two days of coding with an additional day and a half spent on performance tuning and bug fixing. Given that this includes the learning curve of the language itself, the reputation seems well deserved.

A special mention should go to libgmail and FUSE, as both greatly contributed to the short development time.

(I'm particularly concerned with my attempts to manipulate mutable byte arrays. I'm sure there's a less clumsy way of doing it than the nasty list array string path I'm currently using.)

So, do be careful using the GmailFS and certainly don't use it for anything important.

Implementation Details

All meta-information in the GmailFS is stored in the subjects of emails sent by the Gmail users to themselves.

This was not as good an idea as I first thought. I thought I could speed things up by grabbing the message summary without having to download the entire message, as Gmail elides subjects (abbreviates them and adds ellipses) to fit them on the screen, but it turned out I needed to get the full message anyway. (Yes, the message bodies are empty, but it does add considerable latency to operations such as listing the contents of a large directory.)

The actual file data is stored in attachments. Files can span several attachments, which allows file sizes greater than the maximum Gmail attachment. File size should be limited only by the amount of free space in your Gmail account.

There are three types of important structures in the GmailFS:

Directory and file entry structures

Hold the parent paths and names of files or directories. Symlink information is also kept here. These structures have a reference to the file's or directory's *inode* (a data structure that holds information about where and how the file or directory is stored).

Inode structures

Hold the kind of information usually found in a Unix inode, such as mode, uid, gid, size, etc.

Data block structures

One of three types of messages GmailFS uses to store information related to the filesystem. The subject of the messages holding these structures contains a reference to the file's inode as well as the current block number.

As GmailFS can store files longer than the maximum Gmail attachment size, it uses block numbers to refer to the slice of the original file that this data block message refers to. For example, if you have a blocksize of 5 MB and a file 22 MB long, you will have five blocks (5 MB 5 MB, 5 MB, 5 MB, and 2 MB); the block numbers for these are 0, 1, 2, 3, and 4, respectively.

All subject lines contain an *fsname* (filesystem name) field that serves two purposes.

- Prevents the injection of spurious data into the filesystem by external attackers. As such, the fsname should be chosen with the same care that you would exercise in choosing a password.
- Allows multiple filesystems to be stored on a single Gmail account. By mounting with different fsname options set, the user can create distinct filesystems.

Installing the Hack

This isn't for the uninitiated. I haven't provided newbie-focused, step-by-step installation instructions

because if you aren't able to take care of some of these details yourself, you probably shouldn't be mucking about in this hack. If you're out of your depth, sit back, relax, and read on for edification's sake.

Before you begin, make sure you have Python 2.3 and the python2.3-dev packages installed.

Install version 2 or higher of FUSE (http://sourceforge.net/projects/fuse/). Some Linux distributions (such as Debian) make this available as a package, and most newer Linux kernels include FUSE by default. If your distro doesn't have FUSE, you need to download the source and make and install it manually.

Next, you need the Python FUSE bindings (http://richard.jones.name/google-hacks/gmail-filesystem/fuse-python.tar.gz). Download and extract *fuse-python.tar.gz* and follow the instructions in *fuse-python/INSTALL*.

The Python FUSE bindings are also available from FUSE's CVS page (http://sourceforge.net/cvs/?group_id=21636), but if you grab CVS, remember that the Python bindings don't work with the rest of CVS at the time of this writing; you still need to use FUSE 1.3.

Grab libgmail [Hack #77]. After unarchiving the package, copy *libgmail.py* and *constants.py* to somewhere Python can find them (*/usr/local/lib/python2.3/site-packages* works for Debian; others may vary).

Finally, download GmailFS (http://richard.jones.name/google-hacks/gmail-filesystem/gmailfs.tar.gz) itself and unarchive it. Copy *gmailfs.py* to somewhere easily accessible (*/usr/local/bin/gmailfs.py*, for example), and *mount.gmailfs* (a modified version of mount.fuse distributed with FUSE) to */sbin/mount.gmailfs*.

> If you have an older version of Python that interferes with the running of GmailFS and would rather use a newer version, alter the first line of*gmailfs.py* to point at `#!/path/to/newer/python2.3` rather than the `#!/usr/bin/env python` default.

Take a moment to enjoy just how much you know about such things and move on when you're ready.

Running the Hack

All that remains is to mount your Gmail filesystem.

You can do so via `fstab` or on the command line using `mount`. To use `fstab`, create an */etc/fstab* entry that looks something like this:

```
/usr/local/bin/gmailfs.py /path/of/mount/point
gmailfs \\ noauto,username= gmailuser
```

```
,password= gmailpass
,fsname= zOlRRa
```

Replace *gmailuser* and *gmailpass* with your Gmail username and password, respectively. The value you pass to `fsname` is what you'd like to dub this Gmail filesystem.

It is important to choose a hard-to-guess name here. If others can guess the `fsname`, they can corrupt your Gmail filesystem by injecting spurious messages into your Inbox (read: sending you mail).

To mount the filesystem from the command line, use the following command:

```
# mount -t gmailfs /usr/local/bin/gmailfs.py
    /path/of/mount/point
  \\-o username=
    gmailuser
 ,password=
    gmailpass
 ,fsname=
    zOlRRa
```

Again, replace *gmailuser*, *gmailpass*, and *zOlRRa* with your Gmail username, Gmail password, and preferred filesystem name.

> At the time of this writing, both of these command-line invocations have serious security issues. If you run a multiuser system, others can easily see your Gmail username and password. If this is a problem for you, your only option at present is to modify *gmailfs.py* itself, changing `DefaultUsername`, `DefaultPassword`, and `DefaultFsname` as appropriate.
>
> A future version of GmailFS (perhaps already out by the time you read this) will load these values from configuration files in your home directory.

Refer back to Figure 6-25 to see my mounted *gmailfs* filesystem in action.

Things You Should Know

There are a few things you should know as you begin to stroll about and store things on your Gmail filesystem:

- GmailFS also has a blocksize option, with a default of 5 MB. Files smaller than the minimum blocksize use only the amount of space required to store the file,*not the full blocksize*. Note that any files created during a previous mount with a different blocksize retain their original blocksize until they are deleted. For most applications, you can take full advantage of your bandwidth by keeping the blocksize as large as possible.

- When you delete files, GmailFS places the files in the Trash. The libgmail library does not currently support purging items from the Trash, so you have to do this manually through the regular Gmail web interface.

- To avoid seeing the messages created for your Gmail filesystem in your inbox, you should probably create a filter (http://gmail.google.com/sup-port/bin/answer.py?answer=6579&query=filter&topic=&type=f) to automatically archive GmailFS messages as they arrive in your Inbox. The best approach is probably to search for the `fsname` value; it's in the subject of all your GmailFS messages.

Outstanding Issues

At the time of this writing, there are some outstanding issues with GmailFS that you should be aware of:

- I don't recommend storing your only copy of anything important on GmailFS for the following two reasons:
    - GmailFS is currently a 0.7 release and should be treated as such. You can depend on its being undependable.
    - There's no cryptography involved, so all your files are stored in plain text on Google's Gmail servers. This will no doubt make some of you nervous.
- Performance is acceptable for uploading and downloading very large files (though it is obviously dependent on your having decent bandwidth). However, operations such as listing the contents of a large directory, which requires many round trips, are extremely slow. The poor performance here is largely independent of bandwidth and is related to having to grab entire messages instead of being able to use message summaries.
- I haven't done any testing where GmailFS opens the same file multiple times and performs subsequent operations on the file. I suspect it will behave badly.

If all of this doesn't dissuade you from giving GmailFS a whirl, have at it and enjoy. Just be sure to visit the GmailFS page (http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html) to find out what's new and grab the latest instructions and code.

See Also

- There's a PHP utility (http://ilia.ws/archives/15_Gmail_as_an_online_backup_system.html) to employ Gmail as a backup service.
- Gmail Drive Shell Extension (http://www.viksoe.dk/code/gmail.htm; Windows only) wraps the GmailFS into a virtual filesystem that is visible as just another drive in Windows Explorer.
- Mac OS X (10.3 or above) users should check out the freely available gDisk (http://gdisk.sourceforge.net) that adds a Gmail-powered drive to your desktop.
- "Use Gmail as a Hard Drive" [Hack #76].

Richard Jones

PREV

Hack 76. Use Gmail as a Hard Drive

Drop a couple gigs of Gmail storage on your PC desktop and treat it like any other drive.

If "Use Gmail as a Linux Filesystem" [Hack #75] had you Windows users salivating over the prospect of adding a couple gigabytes of networked storage to your computer, do we have a find for you. GMail Drive (http://www.viksoe.dk/code/gmail.htm) drops the 2+ gigabytes of storage allotted to your Gmail account right onto your desktop. It looks and feels just like a regular hard drivealbeit a tad slower (more than a tad if you're on dial-up) because it's networked rather than local.

And it's as simple to use as one might hope, being a Windows application. There are none of the odd libraries to install or fstab entries (whatever those are) to edit, and none of the fuss of the Linux version you paged through a moment ago.

Point your browser at http://www.viksoe.dk/code/gmail.htm, scroll down to the Download Files section, and grab a copy of GMail Drive. The download should take only a few seconds. Unzip the installer and double-click the Setup icon. A few moments later, you should see a brand-spanking-new GMail Drive under My Computer in Windows Explorer.

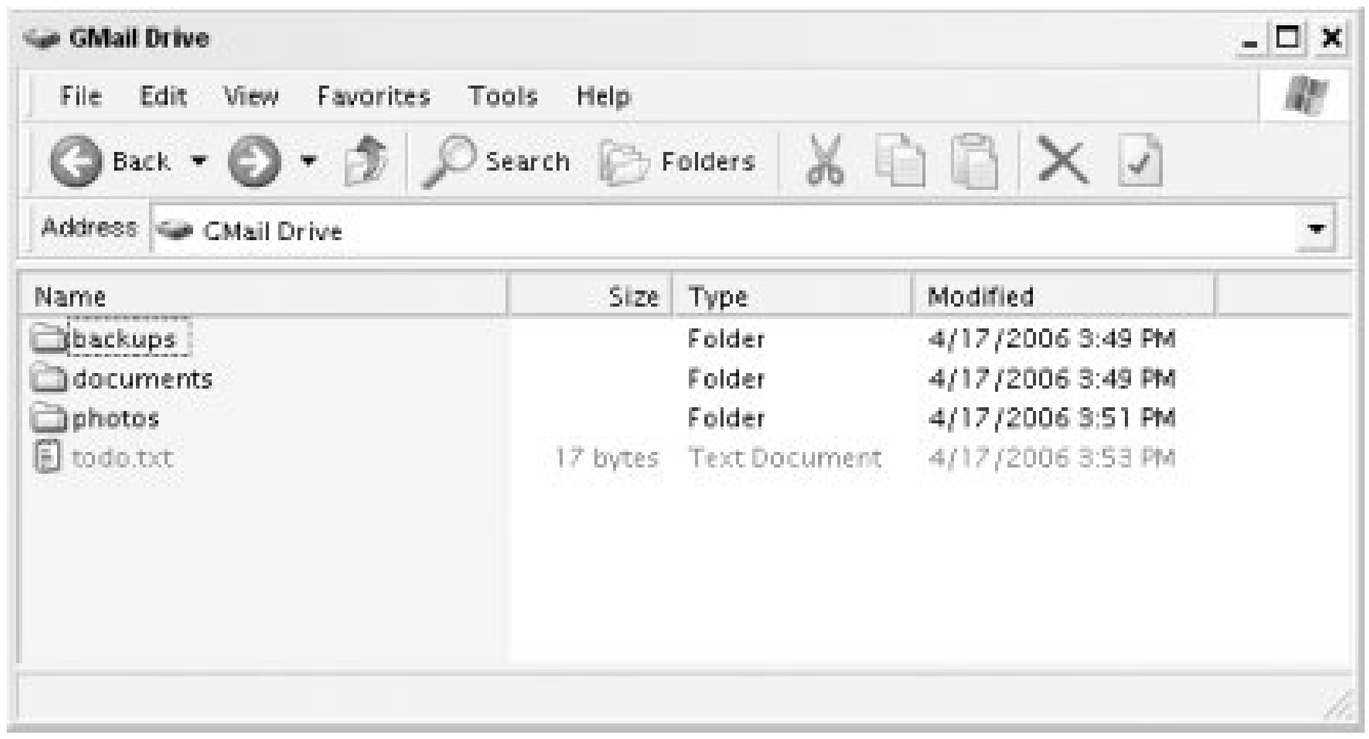
Click the link, and you're prompted to log in, as shown inFigure 6-26.

Figure 6-26. GMail Drive prompting for Gmail login

Notice that GMail Drive provides other options (just click the More button in the Login window), including secure HTTP for encrypted interaction with your remote "drive."

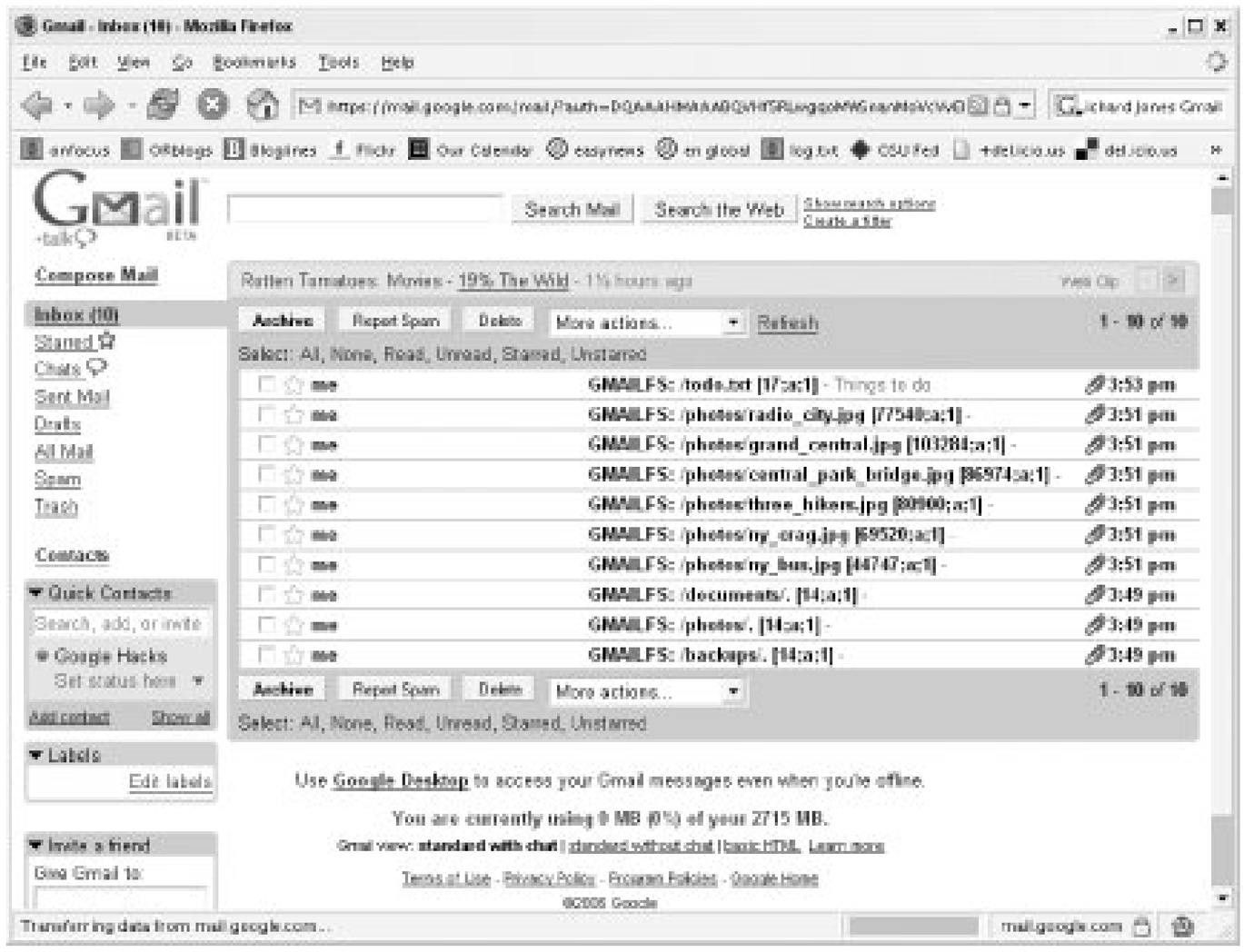
Enter your Gmail username and password and click the OK button to log in. A few seconds later, you drive will be ready to use. Merrily drag and drop files to and fro between your local drive and GMail Drive, as shown in Figure 6-27.

Figure 6-27. Folders and documents on a GMail Drive



As you add folders and documents to the drive, they show up as messages in your Gmail Inbox, as shown in Figure 6-28.

Figure 6-28. GMail Drive files as messages at Gmail



The GMAILFS: prefix appears in the subject of each message, along with the folder, filename, and file size. The file itself is stored as an attachment to the message.

> Keep in mind that if you automatically download a message to your computer via GMail POP access, you mail each file you add to GMail Drive to yourself.

Back in My Computer, right-click the GMail Drive icon to log out, check properties (used space, free space), or log back in.

Notice that the Login option is actually Login As.... This means you can mount the GMail Account of a friend or family member as easily as your own. Transfer that home movie to your grandparents' computer, share your forays into music remixing with your friends, or move files between your home and office computers without toting around an external hard drive or shelling out for a 2 GB USB drive.

See Also

- Mac OS X (10.3 or above) users should check out the freely available gDisk (http://gdisk.sourceforge.net) that adds a Gmail-powered drive to your desktop.
- "Use Gmail as a Linux Filesystem" [Hack #75].

PREV

Hack 77. Program Gmail

Try your hand at writing an alternative interface to Gmail using the freely available Python, P API frameworks.

The relatively simple and lightweight data interface to Gmail stems from the separation between user in model. This has spawned myriad frontends (graphical and otherwise), libraries, and unofficial "API" impl and .NET.

For a glimpse of the Gmail engine and protocol underlying the official Gmail interface and the lion's shar to the service, take a gander at Johnvey Hwang's "About the Gmail engine and protocol" (http://johnvey

Rather than taking you step by step through the same code in each of the five languages and framewor The APIs are all rather similar, which shouldn't come as any great surprise since they are all built on the JavaScript-powered Gmail Web interface.

> Programmatic access to Gmail is accomplished by screen-scraping either the web inter format. While the data format is pretty simple and isn't expected to change dramatica might do that could adversely affect the various programmatic interfaces to its service such hackery comes with no quality-of-service guarantee. In other words, expect brea something's gone wrong, visit the home page of your chosen programmatic interface f news, and further information.

Python

The libgmail (http://libgmail.sourceforge.net ; GNU Public License 2.0/PSF) Python binding for Gmail pro expect from Python) to your Gmail account. Libgmail comes in two pieces: the core program called *libgm* called *libgmail-docs* .

You might be tempted to skip the documentation, but it comes with a lovely set of useful example applic including:

archive.py

Downloads your Gmail messages to your computer for archiving, importing, or moving purposes

gmailsmtp.py

Proxies SMTP requests, allowing you to use Gmail to send email from the comfort of your preferre

sendmsg.py , sends a single email message via Gmail from the command linenot unlike using the

gmailpopd.py

Proxies a standard POP interface to mail from your preferred email application

gmailftpd.py

Pretends to be an FTP server, allowing you to download (only) messages labeled "ftp" via a standa

Installing the hack

Installation is just a matter of downloading and unpacking the libraries (http://sourceforge.net/project/s the Downloads link on the libgmail home page) and putting it someplace where Python can find it.

The code

Libgmail sports much functionality, and each function has its own rather self-explanatory name:`getMessa` `Names` , `getMessagesByLabel` , `geTRawMessage` , and `getUnreadMsgCount` . Leaf through *libgmail.py* for the k love.

Here's a snippet of sample code showing off login, folder selection, and strolling through emailthread by

```
#!/usr/bin/python

# gmail_in_python.py
# A simple example of the libgmail Python binding for Gmail in action
# http://libgmail.sourceforge.net/
#
# Usage: python gmail_in_python.py

import libgmail

# Login
gmail = libgmail.GmailAccount('raelity@gmail.com ', '12bucklemyshoe ')
gmail.login(  )

# Select a folder, label, or starred messages--in this case, the Inbox
folder = gmail.getMessagesByFolder( 'inbox ')

# Stroll through threads in the Inbox
for thread in folder:
  print thread.id, len(thread), thread.subject

  # Stroll through messages in each thread
  for msg in thread:
    print "  ", msg.id, msg.number, msg.subject
```

Replace *raelity@gmail.com* and *12bucklemyshoe* with your Gmail email address and password. Instead o standard folder names or your custom labelse.g., *starred* , *sent* , or *friends* .

Save the code to a file called *gmail_in_python.py* .

Running the hack

Run the *gmail_in_python.py* script on the command line, like so:

```
$ python libgmail_example.pl
WARNING:root:Live Javascript and constants file versions differ.
ff602fe48d89bc3 1 Hello from Hotmail
   ff602fe48d89bc3 1 Hello from Hotmail
ff5fb9c2829c165 1 Hello Gmail via Gmail Loader
   ff5fb9c2829c165 1 Hello Gmail via Gmail Loader
ff5691f7170cb62 1 Hello Gmail via Gmail Loader
   ff5691f7170cb62 1 Hello Gmail via Gmail Loader
ff3f4310237b607 1 Howdy gmail-lite
   ff3f4310237b607 1 Howdy gmail-lite
```

Hacking the hack

Swap in a call to `getMessagesByQuery` , and you now have a command line right to the Gmail search engi

```
                    #folder = gmail.getMessagesByFolder('inbox')folder = gmail.getMessage
```

Here are the results of this little switch:

```
$ python libgmail_example.pl
WARNING:root:Live Javascript and constants file versions differ.
ff3f4310237b607 1 Howdy gmail-lite
   ff3f4310237b607 1 Howdy gmail-lite
```

See Also

- *gmail.py* (http://www.holovaty.com/blog/archive/2004/06/18/1751) is a simple Python interface t messages for backup and import.

Perl

Mail::Webmail::Gmail (http://search.cpan.org/~mincus/Mail-Webmail-Gmail-1.00/lib/Mail/Webmail/Gma programmatic interface to Gmail. You'll find full POD documentation and more sample code than you car at the aforementioned URL.

A Comprehensive Perl Archive Network (CPAN) search for gmail (http://search.cpan.org/search?query=g writing finds many more Perl Gmail libraries including `Mail::Webmail::Gmail` , `WWW::GMail` , and `WWW::Scra` `GMail::Checker` .

PHP

Gmailer, or libgmailer (http://gmail-lite.sourceforge.net ; GNU Public License), is a PHP library for interac (http://www.php.net/curl) with SSL support (http://www.openssl.org). It is the engine underlying the t mobile [Hack #74] , an HTML-only interface to Gmail.

For full libgmailer documentation and plenty of sample code, leaf through the online documentation (htt

Java

G4j, or GMail API for Java (http://g4j.sourceforge.net ; GNU Public License), is a Java interface to Gmail basic GUI frontend to Gmail built on top of G4j.

Full documentation in Javadoc HTML is available online (http://g4j.sourceforge.net/doc).

.NET

The Gmail Agent API (http://johnvey.com/features/gmailapi ; GNU Public License) is a .NET foundation f of source for the API itself; the Gmail Agent Applet, a proof-of-concept Windows frontend to Gmail; and available for download.

There's also full documentation available in both HTML format (http://johnvey.com/features/gmailapi/dc

Hack 78. Force Gmail to Use a Secure Connection

Protect your inbox by automatically redirecting Gmail to an https:// address.

You can use Google's web mail service through an unsecured connection (an *http://* address) or a secure connection (an *https://* address). When I'm out and about and browsing the Web on an untrusted network (such as an Internet cafe), I try to remember to use the *https://* address. But why bother remembering, when Greasemonkey can remember it for me?

This hack relies on the Greasemonkey Plugin (http://greasemonkey.mozdev.org/) for the Firefox web browser (http://www.mozilla.com/firefox/).

The Code

This user script is literally one line of code. The reason it can be so small is that we configure it to run only on http://mail.google.com, the insecure address of Gmail.

Save the following user script as *securewebmail.user.js*:

```
// ==UserScript==
// @name           Secure Webmail
// @namespace      http://diveintomark.org/projects/greasemonkey/
// @description    force webmail to use secure connection
// @include        http://mail.google.com/*
// ==/UserScript==

window.location.href = window.location.href.replace(/^http:/, 'https:');
```

Running the Hack

After installing the user script (Tools Install This User Script), go to http://mail.google.com/mail. Your browser will automatically redirect to https://mail.google.com/mail. Firefox will change the background color of the location bar to pale yellow (as shown in Figure 6-29) to indicate that you are now browsing a secure site.

Figure 6-29. A secure connection to Gmail

Hacking the Hack

Many online applications offer the same service on an *http://* or an *https://* address. This script will work unmodified on any such site. There is nothing Gmail-specific about the code itself; all it does is redirect from an *http://* address to the corresponding *https://* address.

If you use Yahoo! Mail instead of (or in addition to) Gmail, all you need to do is change the script's configuration to tell Greasemonkey to run the script when you visit Yahoo! Mail. Under the Tools menu, select Manage User Scripts. In the list of scripts, select Gmail Secure. You will see the current configuration of where the script should run. Under "Included pages," click Add... and type `http://mail.yahoo.com/*`, as shown in Figure 6-30.

Figure 6-30. Secure Yahoo! Mail configuration

Now, visit Yahoo! Mail at http://mail.yahoo.com. You will immediately be redirected to

https://mail.yahoo.com, and you can sign in to Yahoo! Mail securely.

Mark Pilgrim

PREV

Chapter 7. Webmastering

When the Web was younger, the search engine field was wide open. There were lots of major search engines, including AltaVista, Excite, HotBot, and Webcrawler. This proliferation of search engines had both advantages and disadvantages. One disadvantage was that you had to make sure you submitted your query to several different places, while one advantage was that you had several inflows of traffic spawned from search engines.

As the number of search engines has dwindled, Google's index (and influence) has grown. You no longer have to worry so much about submitting to different places, but you do have to be aware of Google at all times.

PREV

PREV

Google's Importance to Webmasters

But isn't Google just a search engine web site like any other? Actually, its reach is far greater. Google partners with other sites to use the Google index results, including the likes of heavyweight property AOL. Google is also on the multitude of sites out there that use the Google API, advertise through Google, or even display ads for Google. So when you think about potential visitors from Google search results and advertising, you have to think beyond traditional search site borders.

Google's perception of your site has become increasingly more important, which means you have to make sure your site abides by Google's rules or it risks not being picked up. If you're concerned about search engine traffic, you have to make sure that your site is optimized for luring in Google spiders and that it's indexed effectively. And if you don't want Google to index certain parts of your site, you need to understand the ins and outs of configuring your*robots.txt* file to reflect your preferences.

PREV

The Mysterious PageRank

You might hear a lot of talk about Google's PageRank, people bragging about their sites attaining the misty heights of rank 7 or 8, or speaking reverently of sites that have achieved 9 or 10. PageRanks range from 0 (sites that have been penalized or not ranked) to 10 (reserved for only the most popular sites, such as Yahoo! and Google itself). The only place where you can actually see the PageRank of a given URL is in the Google Toolbar [Hack #53], though you can get some idea of its popularity from the Google Directory. Listings in the Google Directory have a green bar next to them which reflects a listing's popularity without giving an exact number.

Google has never provided the entire formula for its PageRank, so all you will find in this book is conjecture. It wouldn't surprise me to learn that the formula is constantly changing; as millions of people try myriad methods to increase their page ranking, Google has to take these efforts into account and (sometimes) react against them.

Why is PageRank so important? Because Google uses it as one aspect of determining how a given URL ranks among millions of possible search results. Still, it's only one aspect. Other aspects are determined via Google's ranking algorithm.

The Equally Mysterious Ranking Algorithm

If you thought Google was tight-lipped about how it determines PageRank, it's an absolute oyster when it comes to the ranking algorithm, which is how Google determines the order of search results. This book can give you some ideas about how the algorithm works, but again, these ideas are conjecture, and the algorithm is constantly changing. Your best bet is to create a content-rich web site and update it often. Google appreciates good content.

Of course, being listed in Google's index is not the only way to tell visitors about your site. You also have the option to advertise on Google.

PREV

Tools for Webmasters

Google doesn't simply leave you to fend for yourself in the Wild Wide Web. Google offers a number o tools that can help you understand how Google sees your site, visualize traffic at your site, advertise with Google, and make money with your site by placing ads for Google. Here is a quick look at the tools that can help you in your quest for top Google placement.

Google Sitemaps

The primary tool that Google offers for webmasters isGoogle Sitemaps (http://www.google.com/webmasters/sitemaps/). Sign up and register your site, and you'll receive detailed reports about when Google last crawled your site, and any errors Google encountered in the process. Figure 7-1 shows a typical site summary at Google Sitemaps.

Figure 7-1. Google Sitemaps Summary page

In addition to detailed information about Google's last visit to your site, you'll find reports about your site's ranking in Google's index for various keywords. Sitemaps also features a tool to analyze your *robots.txt* file to verify that you're keeping the Google bot out of your site's private sections.

To register a site with Google Sitemaps, you must be able to place a text file in your root directory. This is a security measure that lets Google know that you really do have control of the site you want to track.

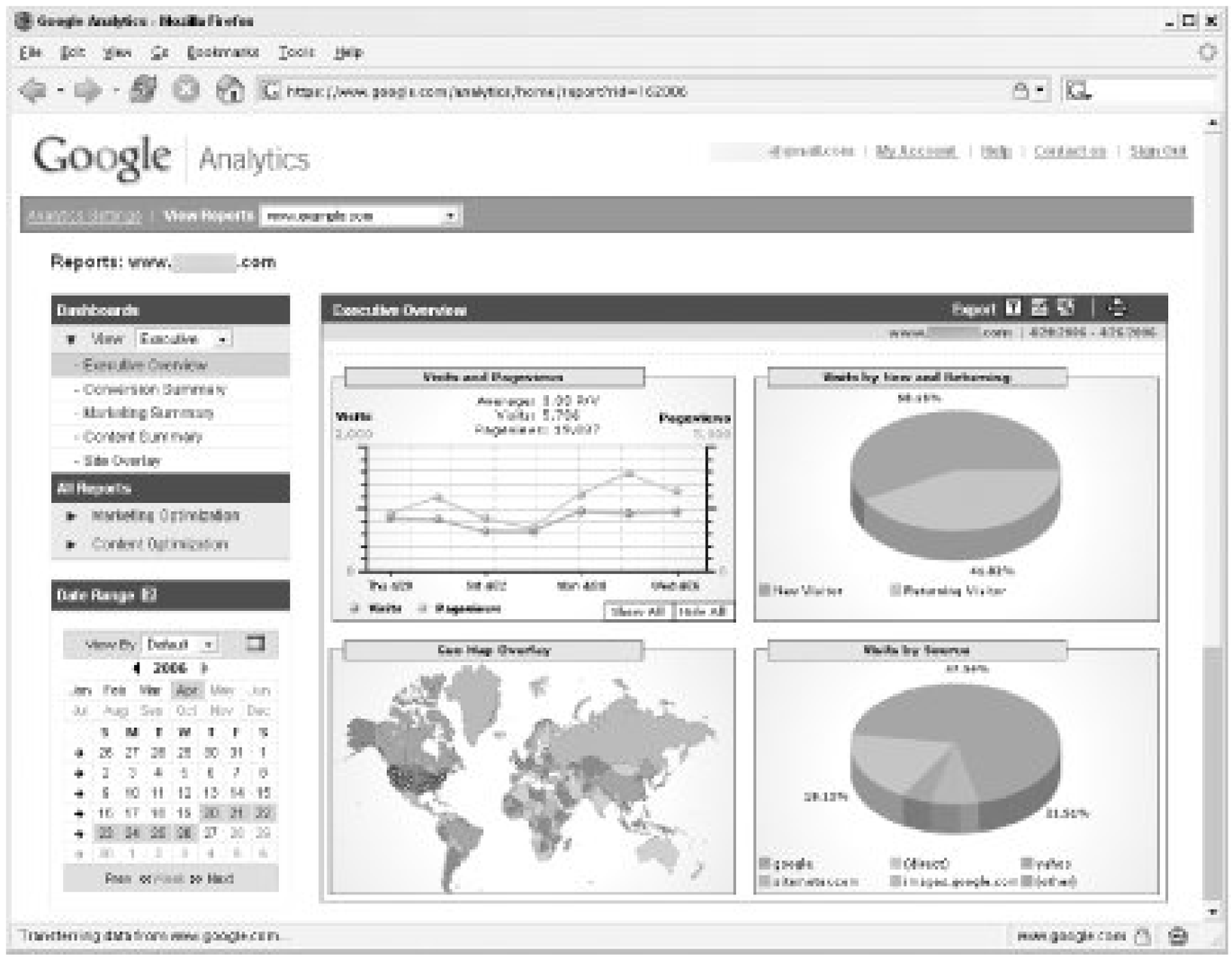
The tool's namesake is the *Sitemap*, a text file you can place on your site that directs the Google bot to recently updated content. Instead of randomly checking your pages, a Google Sitemap tells the Google bot exactly where to go when it visits your site. The Sitemap itself is an XML file that uses Google's Sitemap format (https://www.google.com/webmasters/sitemaps/docs/en/protocol.html) to describe pages at a web site. Google even offers a free Python script called *google-sitemap_gen* (http://sourceforge.net/projects/goog-sitemapgen/) to help you generate a Sitemap automatically. There are a number of third-party plug-ins (http://code.google.com/sm_thirdparty.html) that can help you automatically generate a Sitemap and keep the Google bot informed of changes to your site

Google Analytics

Google Analytics (http://www.google.com/analytics/) is a free web traffic analysis tool that helps you visualize your site's traffic. Instead of relying on web logs generated by your server, you can place a few lines of JavaScript on every page of your site and let Google track your traffic. Google Analytics rivals most of the web log analysis tools available, and it can break down your traffic into a number o segments for review.

Figure 7-2 shows the traffic overview you're greeted with when you log in. It includes a weekly summary of visits and pageviews, a geographic summary of where visitors came from, graphs with the source of the visit, and a look at new versus returning visitors.

Figure 7-2. Google Analytics executive summary of site traffic



Google Analytics is integrated with Google AdWords to help you track the success of advertising campaigns. But you don't need to be an AdWords customer to take advantage of the tool.

> If you want to track your traffic but don't need to see the data summarized in hundreds of different ways, you could try the more user-friendlyMeasure Map (http://www.measuremap.com/), also run by Google. Like Google Analytics, the service is accessible by invitation only as of this writing, but a wider audience is undoubtedly in its future. A hybrid of the two tools will likely be developed.

Google AdWords

Google has built its financial empire on its ability to provide relevant ads to a receptive audience. Google knows that simply grabbing eyeballs isn't enough. It's the*click-through*clicking an ad and following it to the advertiser and its productsthat counts. This is where Google'sAdWords really shine. They're not simply rotating, flip-of-the-coin ads; they're every bit as relevant as the results of your search.

Query Google for `"volvo safety"`, and Car Safety Report ads from Edmunds (http://www.edmunds.com) and insurance quotes from auto insurance providers appear alongside the Volvo safety reports and crash tests. Try `pirates`, and you'll be served (at least at the time of this writing) a Major League Baseball ad. What does MLB have to do with pirates, you ask? Well, Major League Baseball purchased an ad for the keyword because it thinks you might be looking for information about the Pittsburgh Pirates baseball team. As of this writing, you'll also see an ad for Disney's *Pirates of the Caribbean*. If Google has nothing relevant to show, it shows no ads at all.

As an advertiser base, AdWords is hundreds of thousands strong. Mom-and-pops to Fortune 500s are all looking to make their presence known and their wares available alongside Google search results and on thousands of sites across the Web.

This book, which is focused on cool hacks, tools, and techniques, doesn't attempt to describe Google's advertising programs in detail. For a comprehensive introduction to and a detailed treatment of these programs, pick up a copy of *Google Advertising Tools* by Harold Davis (O'Reilly).

In true Google style, AdWords is different from just about every advertising service you've ever seen There's virtually no price barrier; anyone with a few marketing dollars in their pocket can buy a few keywords. Everything is handled through the AdWords site; you don't have to speak to a Google advertising executive to start your campaign. It's so simple that even the most inexperienced marketer can get a leg up. That said, there's a lot to AdWords, and its simplicity can be deceptive.

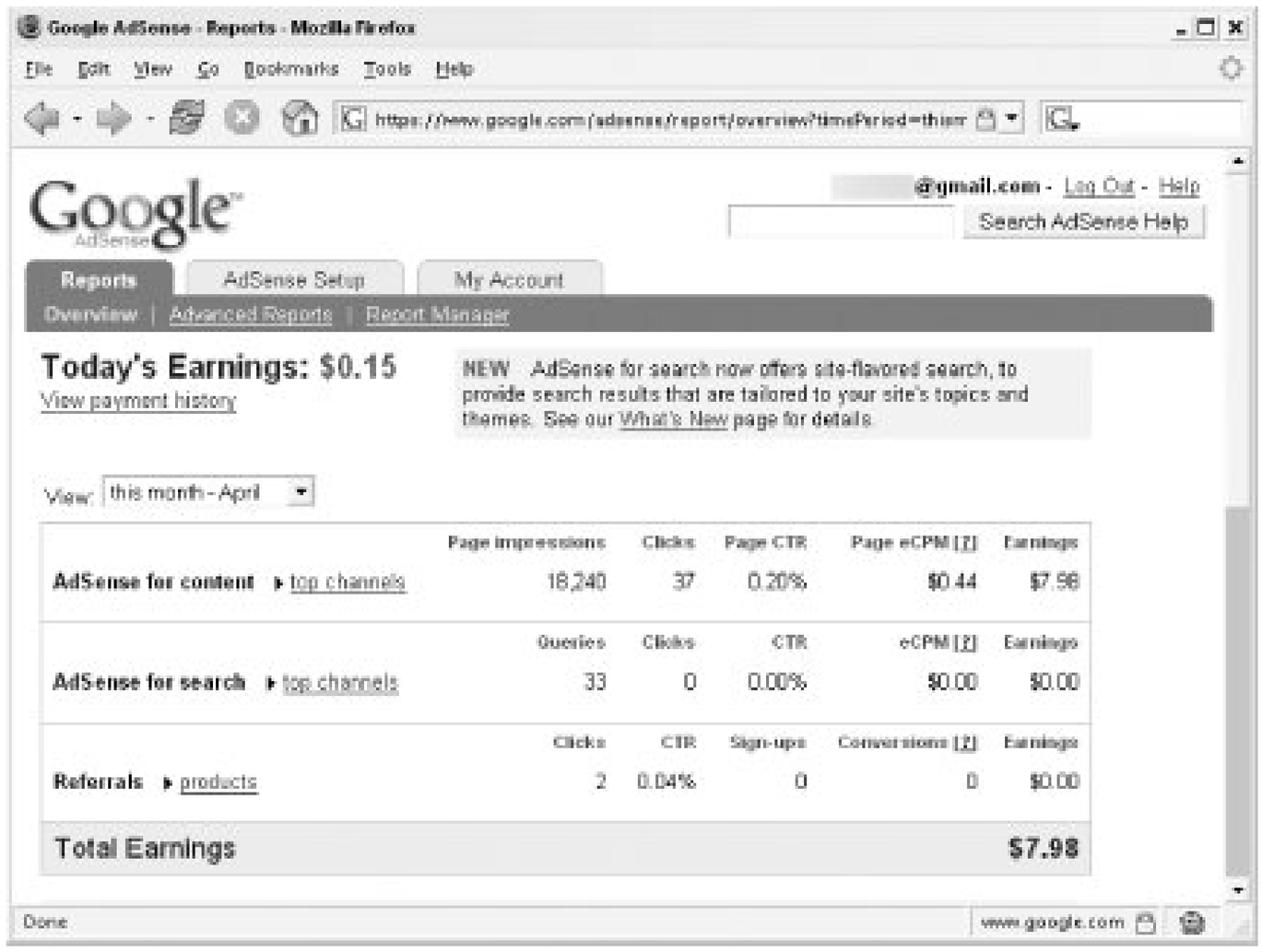
Google AdSense

Google AdSense (http://www.google.com/adsense) is Google's advertising service, designed to deliver advertising magic to your web site. With hundreds of thousands of advertisers signed up, there are sure to be ads that target your readers, whether your site is about baseball, computers, or rare-spoon collecting.

Sign up, choose the shape and size of the ads you want to display, copy some code, paste it into your site, and as your readers click the ads, earn money. Of course, it's not quite that simple; you need to focus on gathering readers and keeping them coming back for more.

The Google AdSense site provides detailed statistics of the number of ads shown and the number of ads clicked. Figure 7-3 shows a monthly AdSense report.

Figure 7-3. Google AdSense earnings report



There are a number of types of ads you can show on your site. In addition to the standard banner and text ads, you can provide Referral buttons that point people to Google products. You can also provide a Google Search box [Hack #88] for your readers with the option to earn money in the process.

Keeping Up with Google's Changes

With Google having such a leading position in the search engine world and so many webmasters looking to Google for traffic, you might guess that there's a lot of discussion about Google in various places around the Web. And you'd be right! My favorite place for Google news and gossip is Webmaster World (http://www.webmasterworld.com). It's not often that the terms "civilized" and "online forums" go together, but they do in this case. Discourse on this site is friendly, informative, and generally flame-free. I have learned a lot from this site.

There are also a few blogs devoted to Google and searching in general:

- Google Blog (http://googleblog.blogspot.com) is the official Google blog and features announcements, pointers, and behind-the-scenes commentary from the Googleplex.

- Googler Matt Cutts maintains a blog called Gadgets, Google, and SEO (http://www.mattcutts.com/blog/). While he doesn't always speak on behalf of Google, he provides insights into Google you won't find anywhere else. Matt discusses sites that have been recently banned for rule violations, new Google features relevant to search engine tuners, and tips for webmasters who want to play nice with Google.

- John Battelle's Searchblog (http://battellemedia.com) covers every conceivable kind of search.

- Google Webmaster Help Center (http://www.google.com/support/webmasters/) should be your first stop to look up official Google policies, find frequently asked questions, and read the latest about Google Sitemaps.

You'll find a complete list of resources for staying on top of the latest Google news in the Appendix.

PREV

In a Word: Relax

One of the things I've learned is that a lot of people spend a lot of time worrying about how Google works and about how they can get the highest possible ranking.

I can appreciate their concern because search engine traffic means a lot to an online business. But the rest of us should just relax. As long as we concentrate on content that's good for visitors (and no just spiders), Google's ranking algorithms will appreciate our sites.

PREV

PREV

Hack 79. A Webmaster's Introduction to Google

Steps to take for optimal Google indexing of your site.

The cornerstone of any good search engine is highly relevant results. Google's unprecedented success has been due to its uncanny ability to match quality information with a user's search terms. The core of Google's search results is based on a patented algorithm called PageRank.

There is an entire industry focused on getting sites listed near the top of search engines. Google has proven to be the toughest search engine for a site to do well on. Even so, it isn't all that difficult to get a new web site listed and begin receiving traffic from Google.

Learning the ins and outs of getting your site listed by a search engine can be a daunting task. There is a vast array of information about search engines on the Web, and not all of it is useful or proper. This discussion of getting your site into the Google database focuses on long-term techniques for successfully promoting your site through Google, helping you avoid some of the common misconceptions and problems that a new site owner might face.

Search Engine Basics

When you type a term into a search site, the engine looks up potential matches in its database and presents the most relevant web page matches first. How those web pages get into the database and, consequently, how you can get yours in there as well, is a three-step process:

1. A search engine visits a site with an automated program called a*spider* (sometimes called a *robot*). A spider is a program similar to a web browser that downloads a site's pages. It doesn't actually display the page anywhere; it just downloads the page data.

2. After the spider has acquired the page, the search engine passes the page to a program called an indexer, which is another robotic program that extracts most of the visible portions of the page. The indexer also analyzes the page for keywords, the title, links, and other important information contained in the code.

3. The search engine adds your site to its database and makes it available to searchers. The greatest difference between search engines is in this final step where ranking or result position for a particular keyword is determined.

Submitting Your Site to Google

The first step is to get your pages listed in the database, and there are two ways to go about this. The first is direct submission of your site's URL to Google via its "Add your URL to Google" page. To counter programmed robots, search engines routinely move submission pages around on their sites. You can find Google's submission page linked from its Help pages orWebmaster Info pages (http://www.google.com/addurl.html).

Visit Google's add URL page, enter the main index page for your site into the submission form, and press Submit. Google's spider (calledGoogleBot) will visit your page, usually within four weeks. The spider will traverse all the pages on your site and add them to its index. Within eight weeks, you should be able to find your site listed in Google.

The second way to get your site listed is to let Google find you based on links that point to your site. Once GoogleBot finds a link to your site from a page that is already in its index, it will visit your site.

Google has been updating its database on a monthly basis for three years. It sends its spider out in crawler mode once a month, as well. Crawler mode is a special mode in which a spider traverses, or *crawls*, the entire Web. As it runs into page links, it indexes those pages in a never-ending attempt t download all the pages it can. Once your pages are listed in Google, they are revisited and updated on a monthly basis. If you frequently update your content, Google may index your search terms more often.

Once you are indexed and listed in Google, the next question for a site owner naturally is, "How can I rank better under my applicable search terms?"

The Search Engine Optimization Template

This is my general recipe for the ubiquitous Google. It is generic enough that it works well everywhere and is as close as I have come to a "one-size-fits-all" SEO (Search Engine Optimization) template.

Use your targeted keyword phrase:

- In `META` keywords. It's not necessary for Google, but it is still a good habit. Keep your`META` keywords short (128 characters max, or 10 keywords).
- In a `META` description. Keep your keywords near the left but as part of a full sentence.
- In the title at the far left, but not as the first word.
- In the top portion of the page in the first sentence of the first full paragraph (plain text: no bold no italic, no style).
- In an H3 or larger heading.
- In bold (second paragraph if possible and anywhere except in the first usage on the page).
- In italic (anywhere except in the first usage).
- In a subscript/superscript.
- In a URL (directory name, filename, or domain name). Do not duplicate the keyword in the URL

- In an image filename used on the page.
- In the `ALT` tag of the image.
- In the `title` attribute of the image.
- In link text to another site.
- In an internal link's text.
- In the `title` attribute of all the targeted links in and out of the page.
- In the filename of your external CSS (Cascading Style Sheet) or JavaScript file.
- In an inbound link on the site (preferably from your home page).
- In an inbound link from off the site (if possible).
- In a link to a site that has a PageRank of 8 or better.

Other search engine optimization issues to consider include:

- Use "last modified" headers if you can.
- Validate the HTML. Some feel that Google's parser has become stricter at parsing instead of milder. It often misses an entire page because of a few simple errors; we have tested this thoroughly.
- Use an HTML template throughout your site. Google can spot the template and parse it off. (Of course, this also means it is pretty good at spotting duplicate content.)
- Keep the page as an *.html* or *.htm* extension. Any dynamic extension is a risk.
- Keep the HTML below 20 KB; 5 to 15 KB is the ideal range.
- Keep the ratio of text to HTML very high. Text should outweigh HTML by a significant amount.
- Double-check your page in Netscape, Opera, and Internet Explorer. Use Lynx if you have it.
- Use only raw `hrEF`s for links. Keep JavaScript far, far away from links. The simpler the link code the better.
- More traffic will come once you realize that 1 referral a day to 10 pages is better than 10 referrals a day to 1 page.
- Don't assume that keywords in your site's navigation template are worth anything at all. Google looks for full sentences and paragraphs. Keywords just lying around orphaned on the page are not worth as much as when they are used in a sentence.

Brett Tabke

Hack 80. Get Inside the PageRank Algorithm

Delve into the inner workings of the Google PageRank algorithm and learn how it affects results.

PageRank, the algorithm used by the Google search engine, was originally formulated by Sergey Brin and Larry Page in their paper "The Anatomy of a Large-Scale Hypertextual Web Search Engine" (http://www-db.stanford.edu/~backrub/google.html).

PageRank is based on the premise, prevalent in the world of academia, that the importance of a research paper can be judged by the number of citations it receives from other research papers. Brin and Page simply transferred this premise to its web equivalent: the importance of a web page can be judged by the number of hyperlinks that point to it from other web pages.

What's the Algorithm?

It might look daunting to nonmathematicians, but thePageRank algorithm is in fact elegantly simple and is calculated as follows:

- PR(A) is the PageRank of a page A.
- PR(T1) is the PageRank of a page T1.
- C(T1) is the number of outgoing links from the page T1.
- d is a damping factor in the range 0 < d < 1; usually set to 0.85.

The PageRank of a web page is therefore calculated as a sum of the PageRanks of all the pages that link to it (its incoming links), divided by the number of links on each of those pages (its outgoing links).

What Does It Mean?

From a search engine marketer's point of view, this means there are two ways in which PageRank can affect the position of your page on Google:

The number of incoming links

> Obviously, the more of these, the better. But there is another thing the algorithm tells you: no incoming link can have a negative effect on the PageRank of the page it points to. At worst, it can have no effect at all.

The number of outgoing links on the page that points to your page

> The fewer of these, the better. This is interesting: given two pages of equal PageRank that link to you, one with 5 outgoing links and the other with 10, you receive twice the increase in PageRank from the page with only 5 outgoing links.

At this point, take a step back and ask yourself just how important PageRank is to the position of your page in the Google Search results.

Note that the PageRank algorithm is that it has nothing whatsoever to do with relevance to the search terms queried. It is simply a single (admittedly important) part of the entire Google relevance ranking algorithm.

Perhaps a good way to look at PageRank is as a multiplying factor applied to the Google Search results after all other computations have been completed. The Google algorithm calculates the relevance of pages in its index to the search terms, and then multiplies this relevance by the PageRank to produce a final list. The higher your PageRank, therefore, the higher up the result list you will be. However, there are still many other factors related to the positioning of words on the page that must be considered.

What's the Use of the PageRank Calculator?

If no incoming link has a negative effect, surely you should just get as many as possible, regardless of the number of outgoing links on its page?

Well, not entirely. The PageRank algorithm is cleverly balanced. Just like the conservation of energy in every physical reaction, PageRank is also conserved with every calculation. For instance, if a page with a starting PageRank of 4 has two outgoing links on it, you know that the amount of PageRank it passes is divided equally between each of its outgoing links. In this case, 4 / 2 = 2 units of PageRank are passed on to each of 2 separate pages, and 2 + 2 = 4so the total PageRank is preserved!

> There are scenarios in which you may find that total PageRank is not conserved after a calculation. PageRank itself is supposed to represent a probability distribution, with the individual PageRank of a page representing the likelihood of a *random surfer* chancing upon it.

On a much larger scale, supposing Google's index contains a billion pages, each with a PageRank of 1, the total PageRank across all pages is equal to a billion. Moreover, each time you recalculate PageRank, no matter what changes in PageRank occur between individual pages, the total PageRank across all one billion pages still adds up to a billion.

This means that although you may not be able to change the total PageRank across all pages, by strategically linking pages within your site, you can affect the distribution of PageRank between pages. For instance, you may want most of your visitors to enter the site through your home page. You would therefore want your home page to have a higher PageRank relative to other pages within the site. Also recall that all the PageRank of a page is passed on and is divided equally between each outgoing link on a page. You would therefore want to keep as much combined PageRank as possible within your own site without passing it to external sites and losing its benefit. This means you would want any page with lots of external links (i.e., links to other people's web sites) to have a lower PageRank relative to other pages within the site to minimize the amount of PageRank that is*leaked* to external sites. Also, bear in mind the earlier statement that PageRank is simply a multiplying facto applied once Google's other calculations regarding relevance have already been done. You would therefore want your more keyword-rich pages to also have a higher relative PageRank.

Also, assuming that every new page in Google's index begins its life with a PageRank of 1, there is a way to increase the combined PageRank of pages within your site: increase the number of pages! A site with 10 pages starts life with a combined PageRank of 10, which is then redistributed through its hyperlinks. A site with 12 pages therefore starts with a combined PageRank of 12. You can thus improve the PageRank of your site as a whole by creating new content (i.e., more pages), and then controlling the distribution of that combined PageRank through strategic interlinking between the pages.

And this is the purpose of the PageRank Calculator: to create a model of the site on a small scale, including the links between pages, and see what effect the model has on the distribution of PageRank.

How Does the PageRank Calculator Work?

To get a better idea of the realities of PageRank, visit thePageRank Calculator (http://www.markhorrell.com/seo/pagerank.asp).

It's simple, really. Start by typing in the number of interlinking pages you want to analyze and hit Submit. I have confined this number to just 20 pages to ease server resources. Even so, this should give a reasonable indication of how strategic linking can affect the PageRank distribution.

Next, for ease of reference once the calculation has been performed, provide a label for each page (e.g., Home Page, Links Page, Contact Us Page, etc.), and again hit Submit.

Finally, use the list boxes to select which pages each page links to. You can use Ctrl and Shift to highlight multiple selections.

You can also use this screen to change the initial PageRanks of each page. For instance, if one of you pages is supposed to represent Yahoo!, you may want to raise its initial PageRank to, say, 3. However, in actuality, the initial PageRank is irrelevant to its final computed value. In other words, even if one page were to start with a PageRank of 100, after many iterations of the equation, the final computed PageRank would converge to the same value as if it had started with a PageRank of only 1!

You can play around with the damping factor *d*, which defaults to 0.85, as this is the value quoted in Brin and Page's research paper.

Mark Horrell

PREV

Hack 81. 26 Steps to 15 KB a Day

Hot and cold running content is what draws visitors to your web site.

Too often, getting visitors from search engines is boiled down to a succession of tweaks that may or may not work. But, as I show in this hack, solid content thoughtfully put together can make more of an impact than a decade's worth of fiddling with `META` tags and building the perfect title page.

Following these 26 steps from A to Z will guarantee a successful site, bringing in plenty of visitors from Google.

A. Prep Work

Prepare work and begin to build content. Long before the domain name is settled on, start putting together notes to build a site of at least 100 pages. That's 100 pages of "real content," not including link, resource, about, and copyright pages, which are necessary, but not content-rich, pages.

Can't think of 100 pages' worth of content? Consider articles about your business or industry, Q&A pages, or back issues of an online newsletter.

B. Choose a Brandable Domain Name

Choose a domain name that's easily brandable. For example, choose something like Google.com and not *<Mykeyword>*.com.

Keyword domains are out; branding and name recognition are in. Big time in. Keywords in a domain name have never meant less to search engines. Consider Goto.com becoming Overture.com, and understand why it was changed. It's one of the most powerful gut check calls I've ever seen on the Internet. It took resolve and nerve to blow away several years of branding. (That's a whole 'nother article, but learn the lesson as it applies to all of us.)

C. Site Design

The simpler your site design, the better. As a rule, text content should outweigh HTML content. The pages should be validated and usable in everything from Lynx to leading browsers. In other words, keep it close to HTML 3.2 if you can. Spiders do not yet like eating HTML 4.0 and the mess that it car bring. Stay away from heavy Flash, Java, or JavaScript.

Go external with scripting languages if you must have them, though there's little reason to have

them that I can see. They rarely help a site and can actually hurt it greatly due to many factors that most people don't appreciate (the search engines' distaste for JavaScript is just one of them). Arrange the site in a logical manner with directory names hitting the top keywords that you want to emphasize. You can also go the other route and just throw everything in the top level of the directory (this is rather controversial, but it's produced good long-term results across many engines). Don't clutter or spam your site with frivolous links such as "best viewed in...", or other things such as counters. Keep it clean and professional to the best of your ability.

Learn the lesson of Google itself: simple is retro cool. Simple is what surfers want.

Speed isn't everything; it's the only thing. Your site should respond almost instantly to a request. If your site has three to four seconds' delay until "something happens" in the browser, you're in trouble. That three to four seconds of response time may vary in sites viewed in countries other than your native one. The site should respond locally within three to four seconds (maximum) to any request. Longer than that, and you'll lose 10 percent of your audience for each additional second. That 10 percent could be the difference between success and failure.

D. Page Size

The smaller the page size, the better. Keep it under 15 KB, including images, if you can. The smaller the better. Keep it under 12 KB if you can. The smaller the better. Keep it under 10 KB if you can. I trust you are getting the idea here. Over 5 KB and under 10 KB. It's tough to do, but it's worth the effort. Remember, 80 percent of your surfers will be at 56 KB or less.

E. Content

Build one page of content (between 200 and 500 words) per day and put it online.

If you aren't sure what you need for content, start with the Overture keyword suggestor (http://inventory.overture.com/d/searchinventory/suggestion/) and find the core set of keywords for your topic area. Those are your subject starters.

F. Keyword Density and Keyword Positioning

This is simple, old-fashioned Search Engine Optimization (SEO) from the ground up.

Use the keyword once in the title, once in the description tag, once in a heading, once in the URL, once in bold, once in italic, and once high on the page, and make sure the density is between 5 and 20 percent (don't fret about it). Use well-written sentences and spellcheck them! Spellchecking is becoming more important as search engines are moving toward autocorrection during searches. There is no longer a reason to look like you can't spell.

G. Outbound Links

From every page, link to one or two high-ranking sites under the keyword you're trying to emphasize. Use your keyword in the link text (this is ultra-important for the future).

H. Cross-Links

Cross-links are links *within* the same site.

Link to on-topic quality content across your site. If a page is about food, make sure it links to you apples page and your veggies page. With Google, on-topic cross-linking is important for sharing your PageRank value across your site. You do not want an *all-star* page that outperforms the rest of your site. You want 50 pages that produce 1 referral each a day, not 1 page that produces 50 referrals each day. If you find a page that drastically outproduces the rest of the site with Google, you need to offload some of that PageRank value to other pages by cross-linking heavily. It's that old share-the-wealth thing.

I. Put It Online

Don't go with virtual hosting; go with a standalone IP address.

Make sure the site is *crawlable* by a spider. All pages should be linked to more than one other page on your site, and not more than two levels deep from the top directory. Link the topic vertically as much as possible back to the top directory. A menu that is present on every page should link to your site's main *topic index* pages (the doorways and logical navigation system that lead to real content). Don't put your site online before it is ready. It's worse to put a*nothing* site online than no site at all. You want it to be fleshed out from the start.

Go for a listing in the Open Directory Project (ODP) (http://dmoz.org/add.html). Getting accepted to the ODP will probably get your pages listed in the Google Directory.

J. Submit

Submit your main URL to Google, F*, AltaVista, WiseNut, Teoma, DirectHit, and Hotbot. Now comes the hard part: forget about submissions for the next six months. That's right,submit and forget.

K. Logging and Tracking

Get a quality logger/tracker that can do justice to inbound referrals based on logfiles. Don't use a graphic counter; you need a program that can provide much more information than that. If your host doesn't support referrers, back up and get a new host. You can't run a modern site without full referrals available 24/7/365 in real time.

L. Spiderings

Watch for spiders from search engines (one reason you need a good logger and tracker!). Make sure that spiders crawling the full site can do so easily. If not, double-check your linking system to make sure the spider can find its way throughout the site. Don't fret if it takes two spiderings to complete

your whole site for Google or F*. Other search engines are potluck; if you haven't been added within six months, it's doubtful you'll be added at all.

M. Topic Directories

Almost every keyword sector has an authority hub on itstopic. Find it (Google Directory can be very helpful here because you can view sites based on how popular they are) and submit within the guidelines.

N. Links

Look around your keyword section in the Google Directory; this is best done*after* getting an Open Directory Project listingor two. Find sites that have link pages or freely exchange links. Simply request a swap. Put a page of on-topic, in-context links on your site as a collection spot. Don't worry if you can't get people to swap links; move on. Try to swaplinks with one fresh site a day. A simple personal email is enough. Stay low-key about it and don't worry if site Z doesn't link to you. Eventually it will.

O. Content

Add one page of quality content per day. Timely, topical articles are always best. Try to stay away from too much blogging of personal material and look more for*article* topics that a general audience will like. Hone your writing skills and read up on the right style of*web speak* that tends to work with the fast-and-furious web crowd: lots of text breaksshort sentenceslots of dashessomething that reads quickly.

Most web users don't actually read; they scan. This is why it is so important to keep key pages to a minimum. If people see a huge overblown page, a portion of them will hit the Back button before trying to decipher it. They have better things to do than waste 15 seconds (a stretch) at understanding your whizbang menu system. Just because some big support site can run Flash-heavy pages, this does not mean that you can. You don't have the pull factor that they do.

Use headers and bold standout text liberally on your pages as logical separators. I call them*scanner stoppers* because the eye logically comes to rest on the page.

P. Gimmicks

Stay far away from *fads of the day* or anything that appears spammy, unethical, or tricky. Plant yourself firmly on the high ground in the middle of the road.

Q. Linkbacks

When *you* receive requests for links, check out the sites before linking back to them.Check them through Google for their PageRank value. Look for directory listings. Don't link back to junk just

because you were asked. Make sure they're sites similar to yours and on-topic. Linking to*bad neighborhoods*, as Google calls them, can actually cost you PageRank points.

R. Rounding Out Your Offerings

Use options such as "email a friend," forums, and mailing lists to round out your site's offerings. Hit the top forums in your market and read, read, read until your eyes hurt. Stay away from*affiliate fades* that insert content onto your site such as banners and pop-up windows.

S. Beware of Flyer and Brochure Syndrome

If you have an economical site or online version of bricks and mortar, be careful not to turn your site into a brochure. These don't work at all. Think about what people want. They don't come to your site to view *your content*, they come to your site looking for *their content*. Talk as little about your products and yourself as possible in articles (sounds counterintuitive, doesn't it?).

T. Keep Building One Page of Content Per Day

Head back to the Overture suggestion tool (http://inventory.overture.com/d/searchinventory/suggestion/) to get ideas for fresh pages.

U. Study Those Logs

After a month or two, you will start to see a few referrals from places you were able to get listed. Look for the keywords people are using. See any bizarre combinations? Why are people using them to find your site? If there is something you have overlooked, then build a page around that topic. Engineer your site to feed the search engine what it wants. If your site is about oranges, but your referrals are about orange citrus fruit, then get busy building articles around citrus and fruit instead of the generic oranges. The search engines tell you exactly what they want to be fed. Listen closely! There is gold in referral logs; it's just a matter of panning for it.

V. Timely Topics

Nothing breeds success like success. Stay abreast of developments in your topic of interest. If big sit Z is coming out with product A at the end of the year, build a page and have it ready in October so that search engines get it by December.

W. Friends and Family

Networking is critical to the success of a site. This is where all that time you spend in forums pays of Here's the catch-22 about forums: lurking is almost useless. The value of a forum is in the interactio with your colleagues and cohorts. You learn from the interaction, not just by reading. Networking

pays off in linkbacks, tips, and email exchanges, and generally puts you*in the loop* of your keyword sector.

X. Notes, Notes, Notes

If you build one page per day, you will find that brainstorm-like inspiration will hit you in the head at some magic point. Whether you are in the shower (dry off first), driving (please pull over), or just parked at your desk, write it down! If you don't, then 10 minutes later, you will have forgotten all about that great idea. Write it down and get specific about what you are thinking. When the inspirational juices are no longer flowing, come back to those content ideas. It sounds simple, but it's a lifesaver when the ideas stop coming.

Y. Submission Check at Six Months

After six months, walk back through your submissions and see if you have been listed in all the search engines you submitted to. If not, resubmit and forget again. Try those freebie directories again, too.

Z. Keep Building Those Pages of Quality Content!

Starting to see a theme here? Google loves content, lots of quality content. The content you generate should be based on a variety of keywords. After a year, you should have around 400 pages of content. This will get you good placement under a wide range of keywords, generate reciprocal links, and position your site to stand on its own two feet.

Do these 26 things, and I guarantee you that in one year's time, you will call your site a success. It will draw between 500 and 2,000 referrals a day from search engines. If you build a good site and achieve an average of 4 to 5 page views per visitor, you should be in the range of 1015 KB page views per day in one year's time. What you do with that traffic is up to you!

Brett Tabke

Hack 82. Be a Good Search Engine Citizen

Five don'ts and one do for getting your site indexed by Google.

A high ranking in Google can mean a great deal of traffic. Because of that, there are lots of people spending lots of time trying to figure out an infallible way to get a high ranking from Google. Add this Remove that. Get a link from this. Don't post a link to that.

Submitting your site to Google to be indexed is simple enough. Google has a site submission form (http://www.google.com/addurl.html), though it says that if your site has at least a few inbound links (other sites that link to you), it should find you that way. In fact, Google encourages URL submitters to get listed on The Open Directory Project (ODP, http://www.dmoz.org) or Yahoo! (http://www.yahoo.com).

Nobody knows the secret of achieving high PageRank without effort. Google uses a variety of elements, including page popularity, to determine PageRank. PageRank is one of the factors determining how high up a page appears in search results. But there are several things that you should not do and one big thing that you absolutely should.

Does breaking one of these rules mean that you will automatically be thrown out of Google's index? No. There are over four billion pages in Google's index as of this writing, and it's unlikely that Google will immediately find out about your violation. But there's a good chance it'll find out eventually. Is it worth having your site removed from the most popular search engine on the Internet?

Thou Shalt Not:

Cloak

Cloaking is when your web site is set up such that search engine spiders get different pages than those that human surfers get. How does the web site know which are the spiders and which are the humans? By identifying the spider's User Agent or IPthe latter being the more reliable method.

An Internet Protocol (IP) address is the computer address from which a spider comes. Everything that connects to the Internet has an IP address. Sometimes the IP address is always the same, as with web sites. Sometimes the IP address changes, in which case it's called a *dynamic address*. (If you use a dial-up modem, chances are that every time you log onto the Internet your IP address is different. That's a dynamic IP address.)

A *User Agent* is a way for a program that surfs the Web to identify itself. Internet browsers such as Mozilla use User Agents, as do search engine spiders. There are literally dozens of different kinds of User Agents; see the Web Robots Database (http://www.robotstxt.org/wc/active.html) for an

extensive list.

Advocates of cloaking claim that cloaking is useful to absolutely optimize content for spiders. Anti-cloaking critics claim that cloaking is an easy way to misrepresent site contentfor example, feeding a spider a page designed to get site hits for pudding cups when the site is actually about baseball bats. You can get more details about cloaking and different perspectives on it at http://pandecta.com/, http://www.apromotionguide.com/cloaking.html, and http://www.webopedia.com/TERM/C/cloaking.html.

Hide text

Text is hidden by putting words or links in a web page that are the same color as the page's backgroundputting white words on a white background, for example. This is also called*fontmatching*. Why would you do this? Because a search engine spider can read the words you've hidden on the page while a human visitor can't. Again, getting caught doing this could get you banned from Google's index, so don't do it.

This goes for other page content tricks too, such as*title stacking* (putting multiple copies of a title tag on one page), putting keywords in comment tags,*keyword stuffing* (putting multiple copies of keywords in a very small font on the page), putting keywords not relevant to your site in your`META` tags, and so on. Google doesn't provide an exhaustive list of these types of tricks on its site, but any attempt to circumvent or fool its ranking system is likely to be frowned upon. Its attitude is more like, "You can do anything you want to with your pages, and we can do anything we want to with our indexsuch as excluding your pages."

Use doorway pages

Doorway pages (sometimes called *gateway pages*) are pages aimed specifically at one topic. They don't have a lot of original content and lead to the main page of a site (thus the name doorway pages).

For example, say you have a page devoted to cooking. You create doorway pages for several types o cookingFrench cooking, Chinese cooking, vegetarian cooking, etc. The pages contain terms and`META` tags relevant to each type, but most of the text is a copy of all the other doorway pages, and all it does is point to your main site.

Doorway pages are illegal in Google and annoying to the Google user, so don't use them. You can learn more about doorway pages at:

- http://searchenginewatch.com/webmasters/bridge.html
- http://www.searchengineguide.com/whalen/2002/0530_jw1.html

Check your link rank with automated queries

Using automated queries (except for the sanctioned Google API) is against Google's Terms of Service Using an automated query to check your PageRank every 12 seconds is triple-bad: it's not what the

search engine was built for, and Google probably considers it a waste of its time and resources.

Link to "bad neighborhoods."

Bad neighborhoods are sites that exist only to propagate links. Because link popularity is one aspect of how Google determines PageRank, some of these sites have set up *link farms*, which are sites that exist only for the purpose of building site popularity with bunches of links. The links are not topical, like a specialty subject index, and they're not well-reviewed, like Yahoo!; they're just a pile of links. Another example of a bad neighborhood is a general FFA (*free for all*) page, where anyone can add their link. Linking to pages in this way is grounds for a penalty from Google.

Now, what happens if one of these pages links to *you*? Will Google penalize your page? No. Google accepts that you have no control over who links to your site.

Thou Shalt:

Create great content

All the HTML contortions in the world will do you little good if you have lousy, old, or limited content. If you create great content and promote it without playing search engine games, you will get noticed and you will get links. Remember Sturgeon's Law: "Ninety percent of everything is crud." Why not make your web site an exception?

What Happens If You Reform?

Maybe your site is not exactly the work of a good search engine citizen. Maybe you have 500 doorway pages, 10 `title` tags per page, and enough hidden text to make an O'Reilly Pocket Guide. But maybe now you want to reform. You want to have a clean, lovely site and leave the doorway pages to *Better Homes and Gardens*. Are you doomed? Will Google ban your site for life?

No. The first thing you need to do is clean up your site. Remove all traces of rule breaking. Next, send a note about your site changes and the URL to help@google.com. Note that Google really doesn't have the resources to answer every email about why it did or didn't index a site. Otherwise, it'd be answering emails all dayand there's no guarantee it will reindex your kinder, gentler site. But it will look at your message.

What Happens If You Spot Google Abusers in the Index?

What if some other site that you come across in your Google searching is abusing Google's spider an PageRank mechanism? You have two options. You can send an email to spamreport@google.com or fill out the form at http://www.google.com/contact/spamreport.html. (I'd fill out the form; it reports the abuse in a standard format that Google is used to seeing.)

Hack 83. Clean Up for a Google Visit

Before you submit your site to Google, make sure you've cleaned it up to make the most of your indexing.

You clean up your house when you have important guests over, right? If you want visitors, Google's crawler is one of the most important guests your site will ever have. A high Google ranking can lead to incredible numbers of referrals, both from Google's main site and from sites with searches powered by Google.

To make the most of your listing, step back and look at your site. By making some adjustments, you can make your site both more Google-friendly and more visitor-friendly:

If you must use a splash page, have a text link from it.

> If I had a dollar for every time I went to the front page of a site and saw no way to navigate besides a Flash movie, I'd be able to nap for a living. Google doesn't index Flash files, so unless you have some kind of text link on your splash page (a "Skip This Movie" link, for example, that leads into the heart of your site), you're not giving Google's crawler anything to work with You're also making it difficult for surfers who don't have Flash or are visually impaired.

Make sure your internal links work.

> Sounds like a no-brainer, doesn't it? Make sure your internal page links work so the Google crawler can get to all your site's pages. You should also make sure that your visitors can navigate.

Check your title tags.

> There are few things sadder than getting a page of search results and finding "Insert Your Title Here" as the title for some of them, although this is not quite as bad as getting results for a domain and seeing the *exact same* `title` tag over and over and over and over.
>
> Look. Google makes it possible to search just the `title` tags in its index. Further, the `title` tags are easy to read on Google's search results and are an easy way for a surfer to quickly ge an idea of what a page is about. If you're not making the most of your `title` tag, you're missing out on a lot of attention to your site.
>
> The perfect `title` tag, to me, says something specific about the page it heads and is readable to both spiders and surfers. This means you shouldn't stuff it with as many keywords as you can. Make it a readable sentence, orand I've found this to be useful for some pagesa question.

Check your `META` tags.

> Google sometimes relies on `META` tags for a site description when there's a lot of navigation code that wouldn't make sense to a human searcher. I'm not crazy about `META` tags, but I'd make sure that at least the front page of my web site has a description and keyword `META` tag set, especially if my site relies heavily on code-based navigation (like from JavaScript).

Check your `ALT` tags.

> Do you use a lot of graphics on your pages? Do you have `ALT` tags for them so that visually impaired surfers and the Google spider can figure out what those graphics are? If you have a splash page with nothing but graphics on it, do you have `ALT` tags on all those graphics so that a Google spider can get some idea of the content? `ALT` tags are perhaps the most neglected aspect of a web site. Make sure yours are set up.
>
> By the way, just because `ALT` tags are a good idea, don't go crazy. You don't have to explain in your `ALT` tags that a list bullet is a list bullet. You can just mark it with an asterisk.

Check your frames.

> If you use frames, you might be missing out on some indexing. Google recommends you read Danny Sullivan's article "Search Engines and Frames" at http://www.searchenginewatch.com/webmasters/frames.html. Be sure that Google can either handle your frame setup or that you've created an alternative way for Google to visit, such as using the `NOFRAMES` tag.

Consider your dynamic pages.

> Google says they "limit the number and amount of dynamic pages" they index. Are you using dynamic pages? Do you have to?

Consider how often you update your content.

> There is some evidence that Google indexes popular pages with frequently updated content more often. How often do you update the content on your front page?

Make sure you have a *robots.txt* file if you need one.

> If you want Google to index your site in a particular way, make sure you have a *robots.txt* file for the Google spider to refer to. You can learn more about *robots.txt* at http://www.robotstxt.org/wc/norobots.html.

If you don't want Google to cache your pages, you can add a line to every page that you don't want cached.

> Add this line to the `<HEAD>` section of your page:

```
<META NAME="ROBOTS" CONTENT="NOARCHIVE">
```

This tells all robots that archive content, including engines such as Daypop and Gigablast, not to cache your page. If you want to exclude just the Google spider from caching your page, use this line:

```
<META NAME="GOOGLEBOT" CONTENT="NOARCHIVE">
```

PREV

Hack 84. Remove Your Materials from Google

Remove your content from Google's various web properties.

Some people are more than thrilled to have Google index their sites. Other folks don't want the GoogleBot anywhere near them. If you fall into the latter category, and the bot's already done its worst, there are several things you can do to remove your materials from Google's index. Each part of GoogleWeb Search, Google Images, and Google Groupshas its own set of methodologies.

Google Web Search

Here are several tips to avoid being listed.

Making sure your pages never get there to begin with

While you can take steps to remove your content from the Google index after the fact, it's always much easier to make sure the content is never found and indexed in the first place.

Google's crawler obeys the *robot exclusion protocol*, a set of instructions you put on your web site that tells the crawler how to behave when it reaches your content. You can implement these instructions in two ways: via a `META` tag that you put on each page (handy when you want to restrict access to only certain pages or certain types of content) or via a *robots.txt* file that you insert in your root directory (handy when you want to block some spiders completely or want to restrict access to specific content or directories). You can get more information about the robot exclusion protocol and how to implement it at http://www.robotstxt.org/.

Removing your pages after they're indexed

There are several things you can remove from Google's results.

> These instructions are only for keeping your site out of Google's index. For information on keeping your site out of all major search engines, you have to work with the robots exclusion protocol.

The whole site

Use the robots exclusion protocol, probably with *robots.txt*.

Individual pages

Use the following `META` tag in the `HEAD` section of each page you want to remove:

```
<META NAME="GOOGLEBOT" CONTENT="NOINDEX, NOFOLLOW">
```

Snippets

A *snippet* is the little excerpt of a page that Google displays on its search result page. To remove snippets, use the following `META` tag in the `HEAD` section of each page for which you want to prevent snippets:

```
<META NAME="GOOGLEBOT" CONTENT="NOSNIPPET">
```

Cached pages

To prevent Google from keeping cached versions of your pages in its index, use the following `META` tag in the `HEAD` section of each page for which you want to prevent caching:

```
<META NAME="GOOGLEBOT" CONTENT="NOARCHIVE">
```

Removing that content now

Once you implement these changes, the next time GoogleBot crawls your web site (usually within a few weeks), it will remove or limit your content according to your `META` tags and *robots.txt* file. If you want your materials removed right away, you can use the automatic remover at http://services.google.com:8882/urlconsole/controller. You have to sign in with an account (requires an email address and a password). Using the remover, you can request that Google crawl your newly created *robots.txt* file, or you can enter the URL of a page that contains exclusionary `META` tags.

> Make sure all your exclusion tags are set up before you use this service. Going to all the trouble of getting Google to pay attention to a *robots.txt* file or exclusion rules that you haven't set up is simply a waste of your time.

Reporting pages with inappropriate content

While you may like your own content fine, you might find that, even if you have filtering activated, you're getting search results with explicit content. Or you might find a site with a misleading title tag and content completely unrelated to your search.

You have two options for reporting these sites to Google. Bear in mind that there's no guarantee that Google will remove the sites from the index, but it will investigate them. At the bottom of each page of search results is a "Dissatisfied? Help Us Improve" link; follow it to a form for reporting inappropriate sites. You can also send the URL of explicit sites that show up on a SafeSearch but

probably shouldn't to safesearch@google.com. If you have more general complaints about a search result, send an email to search-quality@google.com.

Google Images

Google's Image database of materials is separate from that of the main search index. To remove items from Google Images, use *robots.txt* to specify that the GoogleBot Image crawler should stay away from your site. Add these lines to your *robots.txt* file:

```
User-agent: Googlebot-Image
Disallow: /
```

You can use the automatic remover mentioned previously in "Removing that content now" to have Google remove the images from its index database quickly.

There may be cases where someone has put images on his server for which you own the copyright. In other words, you don't have access to his server to add a*robots.txt* file, but you need to stop Google from indexing your content there. In this case, you need to contact Google directly. Google has instructions for situations just like thisat http://www.google.com/remove.html; choose Option 2, "If you do not have any access to the server that hosts your image."

Google Groups

As with the Google Web Index, you have the option to both prevent material from being archived on Google and to remove it after the fact.

Preventing your material from being archived

To prevent your material from being archived on Google, add the following line to the headers of you Usenet posts:

```
X-No-Archive: yes
```

If you do not have the option to edit the headers of your post, make that line the first line in your post itself.

Removing materials after the fact

If you want to remove materials after the fact, you need to gather a couple pieces of information. First, find the email address that you used to post the messages; even if you don't have access to the address anymore, you can still remove your materials. You also need the Message IDs of the pos or posts you want to remove.

Here's how you find a Message ID for a particular Google Groups post. First, bring up the message

detail and click the "Show options" link at the top of the post. Then, click "Show original" and copy the text after the Message-ID header. The ID itself is typically a long string of numbers followed by the domain name you posted from. A typical Message ID looks like this: <12345678900.0ABC123@example.com>.

Browse to the automatic-removal tool at http://groups.google.com/groups/msgs_remove. Enter the email and Message IDs, and proceed to the validation step.

Google asks you to include the statement:

I swear under penalty of civil or criminal laws that I am the person who posted each of the foregoing messages or am authorized to request removal by the person who posted those messages.

Google wants to be sure that you're serious about removing the materials. You also need to provide your full contact information and your reason for deleting the messages.

Someone at Google Groups will review your request, and you'll usually find out if your request has been successful in a few days.

Google Phonebook

You might not want to have your contact information made available via the Phonebook searches on Google. You'll have to follow one of two procedures, depending on whether the listing you want removed is for a business or for a residential number.

If you want to remove a business phone number, you need to send a request on your business letterhead to:

> Google Phonebook Removal
> 1600 Amphitheatre Parkway
> Mountain View, CA 94043

Be sure to include a phone number so that Google can reach you to verify your request.

Removing a residential phone number is much simpler. Fill out theform at http://www.google.com/help/pbremoval.html. The form asks for your name, city and state, phone number, email address, and reason for removal in the form of a multiple-choice question: incorrect number, privacy issue, or "other."

Hack 85. Get the Most Out of AdWords

Here's some guest commentary and advice by Andrew Goodman of Page Zero Media on how advertisers can better cope with the increasingly complex Google AdWords program.

AdWords (https://adwords.google.com) is just the sort of advertising program that you might expect to roll out of the big brains at Google. The designers of the advertising system have innovated thoroughly to provide precise targeting at low cost with less work; it really is a completely new way of looking at advertising.

The "less work" part is something I take seriously. Focus on structuring a campaign in a robust way so that you can understand and adjust it. Don't beat yourself up by trying to get everything 100 percent perfect. A "robust" campaign has a well-planned category structure, or "ontology" (basically, themed campaigns and ad groups). And a robust ongoing advertising project includes easy-to-follow metrics such as cost per order, and an easy interface for reading those results. Remember, you might wind up handing this work off to someone else. Don't make it convoluted.

> I've taken to nicknaming the latest version of AdWords "2.5." In the previous edition of this book, I wrote about AdWords 2.0. A lot has changed. Let's get up to speed.

In the early days (AdWords 1.0), the platform offered few features and charged a fixed rate (in cost-per-thousand impressions, or CPM, format) for ads that would show up near search results, triggered by a user query on a keyword or phrase you placed in your AdWords account.

In 2002, Google came out with version 2.0. The pricing was based on a cost-per-click auction that allowed advertisers to bid, but also incorporated click-through rates (CTR) into the ad rank formula. The idea was to push more relevant ads higher on the page.

In August 2005 (with experiments and refinements taking place prior to and following that), Google changed the system fairly significantly. Instead ofCTR, the new ad rank formula became Max CPC (your bid on a keyword or a group of keywords) multiplied byQuality Score (QS). QS is multidimensional. Google states that CTR is predominant in QS.

But other factors, including landing page relevancy and ad copy relevancy, can play a significant role in where your ad now ranks. It's largely a black box, but if you've followed Google closely, you may be aware of the types of issues it focuses on. On the organic search side, certain kinds of deceptiveness and poor user experiences are judged as "evil," and you could wind up paying the penalty. It's safe to say that some of Google's main pet peevessuch as pop-ups, deceptive redirects, and pages with nothing but graphicsare included in the list of potential QS criteria.

If a keyword in your AdWords account has a low quality score, you might be asked to bid very high$5

per click isn't uncommonjust to show up at all. Some advertisers are relatively unaffected by these changes. Others have been virtually wiped off the map.

Between this and the increasingly competitive environment, advertisers must become quite systematic with how they test and refine campaigns. Those with large accounts now demand more features to help them stay organized, to control ad delivery, and to report on their results. Irrespective of the fundamental "version" of AdWords that I'm calling 2.5, the past couple of years have seen Google add dozens of small features and refinements to the platform, sometimes seemingly on a weekly basis. Let me draw your attention to a few notable ones. You need to look elsewhere if you seek in-depth programming advice vis-\x88 -vis the AdWords API and the like. To borrow a phrase from Jim Sterne, "I'm a marketing guy."

Dynamic Keyword Insertion: Beware

Most of the time, users click on ads with titles and text that closely match what they're looking for. Lazy advertisers who want to take advantage of the ability to match the user's query in their ad title or ad text can use dynamic keyword insertion. The format is`{KeyWord: Alternate text }`yes, you need to include those squiggly brackets (called*braces*). So, if this is placed in your ad title and the user types`Red Cactus`, Red Cactus will be your ad title.

A typical use of this format is a large list of products that you don't want to create separate ads for; you want users to see something better than a generic title. Here's the problem with dynamic keyword insertion. Even factoring in the ad rank boost you get from the higher CTR that generally comes with matching the user's query, the return on investment of ads using such matching is ofteneven usuallylower than if you use a manual ad title that is slightly less Pavlovian in its appeal to searchers. You do pay a penalty in terms of ad rank when you write ad copy that filters out some prospects, but it's often worth it to pay that penalty.

Run Better Tests

Go into the "edit campaign settings" area of your account, for any given campaign. The default method of rotating ads is set to Google's advantage. Google will start crowding out the ad that gets fewer clicks (a lower CTR) and start showing your high-CTR ad more often. That puts more money in its pocket, but it doesn't allow you to run the test evenly to measure through to your revenue stream to be sure which ad is truly the "better performer."

To ensure even ad rotation, uncheck the "show better-performing ads more often" box. Incidentally, the even ad rotation continues to be an underused feature of AdWords. You can use it to test more than ads. If you have two or three alternate landing pages, you can send your AdWords traffic to them equally from a given group of keywords just by setting up more than one ad for an ad group, with identical ad copy, and changing only the destination URLs associated with the ad.

If you have tagged each ad URL with unique tracking code so your analytics package knows what's going on (or are using Google's Conversion Tracker), you should be able to compare the conversion rates on these landing pages without having to use any fancy content management techniques on your site. An example would be testing the home page against a tailored landing page, or a category page showing a selection of choices against a page describing a single product in depth. Don't guess, test.

Run Smarter Reports

You don't need to sign up for Google's Analytics service, or buy Webtrends Enterprise, Omniture SiteCatalyst, or other high-powered analytics services, to get useful tracking of your clicks right through to a sale. Google's entry-level Conversion Tracker gives you plenty of information if you set i up right. It's fairly similar to most analytics software in that you have to install the correct JavaScript code on your site and sometimes customize things to pass through revenue data, if desired.

But what if you want to determine the return on investment on*ad copy* you're testing, as opposed to which *keywords or groups of keywords* are performing better? Many users aren't aware of how powerful the advanced reporting is in AdWords. Go to the Reports tab and, once on the Create Repor page, select Text Ad Report. You can then select a time frame and limit the report to the campaigns you're interested in.

Finally, you need to customize the stats shown in the report. Conversion-related data isn't included by default, so you need to bring up the whole range of available report elements (under Advanced Options: Columns, further down the Create Report page) and then click some additional boxes to adc them.

To keep the reports compact, I usually deselect "impressions," but that's me.

Run the report and view it either online or in your preferred downloadable format. Quite simply, if you've made enough sales off the groups of keywords in question, you should see comparative data showing exact cost-per-conversion numbers for ads you've been running head to head (no fancy tracking codes required on your destination URLs). If you determine that one ad is significantly bette than another in its ultimate revenue generation effect, you've made the most important discovery you can make about ad performance, and next, you'll want to delete the nonperforming ad and perhaps run new tests. You'll also want to attempt to learn lessons from the test, which is easier if you've been testing certain elements of your copy based on industry theories and your own hypothesis.

Sometimes, these tests are amazing. I ran a test on one producta new ad competing with the oldfor the month of April 2006. Both generated about 600 clicks. The old ad generated 11 unique new customers at a cost of $9.73 per customer. The new ad generated zero transactions, so we're still waiting on that cost-per-acquisition figure as it is currently at "infinity"! 110...on 600 clicks! That's statistically significant to say the least. Goodbye to the new ad.

This helped us confirm that our call to action in the old ad, relating to free shipping and available inventory, was working well. Unfortunately, AdWords doesn't give me suggestions for how to explain to a client that I wasted his cash testing a "great new ad" that turned out to suck. Maybe we can chalk it up to "branding."

Control Your Ad Positions

You can now tell AdWordskeyword by keywordto show your ads only in certain positions in the

advertising area of the search engine results page. This is available in Advanced Options under Edit Campaign Settings.

Let's say you want your ad to show up only if you can be no higher on the page than ad position 3 and no lower than 6. Enabling the ad position control provides a drop-down box in "keyword settings for every keyword. The default is "any position," but if you say to show your ad no higher than third position, and no lower than sixth, Google simply doesn't show your ad if your ad rank forces you outside these positions. Be careful with this one.

Essentially, what it's telling the system is to keep your ads turned off unless your positioning criteria are met, so you could see sharply reduced ad impressions if you use the feature. This feature won't adjust your bid for you, so this is not "classic" bid-to-position functionality that is available with some third-party bid management tools. It looks like Google is testing the waters on this feature, including the revenue impact, in order to decide whether to implement a more robust set of bid management options.

For now, Google still has no bid-to-position feature and no day-parting feature. Advertisers seeking such exotica need to investigate third-party options or custom programming on their own through the AdWords API. Keep in mind, though, that the existing bid system is easy to use because it has an automatic bid discounter, and it subjects your ad delivery to whatever budgetary parameters you specify. Frequent (several times daily) bid changes are overrated, especially given that your ad rank is not solely dependent on your bid.

Don't Be a Slave to Automation

Google has been brilliant about automating editorial functions. If you're new to some of the legal and policy issues, the process can be daunting. One example is the automated ad copy checker that kicks in when you enter a new ad. Sometimes, the system identifies a misspelling when in fact you're using a niche term that's relevant to your region or industry.

Other times, a potential trademark violation comes up, such as using the word "enterprise" in your ad copy (which is the name of a rental car company). If you're not in the rental car business, it's highly unlikely that your use of a common word constitutes any kind of violation. You should politely appeal to Google in the box provided. Usually, your ad is approved within 48 hours.

Control Your Content Bidding

Some advanced advertisers might want to use custom programming to separate clicks from Google's "content partners" from clicks on the same keywords emanating from the search network. I prefer to use established analytics packages that do this. At the very least, you need to understand that most "content targeting" clicks are worth significantly less to the average advertiser than most clicks originating from a web search.

In Edit Campaign settings, make sure you enable "content bidding" if you have "content targeting" enabled. Then set different bids on content, ad group by ad group. If you're bidding around $.50 on a set of keywords, I'd recommend bidding $.15$.20 on content, or even less. The only way to know for sure is to use sophisticated analytics.

Control Your Geography

During campaign setup, you're offered "location targeting options." For existing campaigns, you can edit these settings. Specify countries and territories, regions and cities, or custom. Whatever you do, don't show your ad to the whole world unless that's your intention. Using custom geography, you car even target a radius of a few hundred miles, or a geographic "shape," if you're handy with the tools, which are now easier to use with a WYSIWYG-type interface on a map.

Boo-yah! This functionality works fairly well subject to the limitations of mapping users' IP addresses to exact locations. It's currently available in about 15 countries. Even national advertisers may find this highly useful, running campaigns in several dozen major metropolitan areas and adjusting their bids upwards in some of them to reflect the cities that are most responsive in terms of sales conversions.

A lot of these features can be a downright blast to use, especially for junkies. Don't forget your goals acquiring new customers and communicating with them effectively and consistently. Ultimately, it's about customers, growth, and profitwhatever those concepts mean to you and your business. Many of the most advanced features have the effectwhen boiled downof causing you to buy too much media, or too little. Keep that in mind as you build and refine a campaign that runs consistently in an optimal range.

Andrew Goodman

Hack 86. Generate Google AdWords

You've written the copy and you've planned the budget. Now, what keywords are you going to use for your ad?

You've read about it and you've thought about it and you're ready to buy one of Google'sAdWords. You've even got your copy together, and you feel pretty confident about it. You have only one problem now: figuring out your keywords (the search words that will trigger your AdWord to appear)

You're probably buying into the AdWords program on a budget, and you definitely want to make every penny count. Choosing the right keywords means your ad will have a higher click-through rate Thankfully, the Google AdWords program allows you to do a lot of tweaking, so if your first choices don't work, experiment, test, and test some more!

Choosing AdWords

So where do you get the search keywords for your ad? There are four places that might help you finc them:

Logfiles

> Examine your site's logfiles. How are people finding your site now? What words are they using? What search engines are they using? Are the words they're using too general to be used for AdWords? If you look at your logfiles, you can get an idea of how people who are interested in your content are finding your site. (If they aren't interested in your content, why would they visit?)
>
> In fact, if you use Google Analytics (https://www.google.com/analytics/) to measure your site traffic, you'll find a category called Keyword Considerations that automatically tabulates the keywords that people have used to find your site via Google and other search engines.

Examine your site

> If you have an internal search engine, check its logs. What are people searching for once they get to your site? Are there any common misspellings you could use as an AdWord? Are there any common phrases you could use?

Brainstorm

What do people think of when they look at your site? What keywords do you want them to think of? Brainstorm about the product that's most closely associated with your site. What words come up?

Imagine someone goes to a store and asks about your products. How would they ask? What words would they use? Consider the different ways someone could look for or ask about your product or service, and if there's a set of words or a phrase that pops up over and over again.

Glossaries

If you've brainstormed until wax dribbles out your ears but are no closer to coming up with words relevant to your site or product, visit some online glossaries to jog your brain. The Glossarist (http://www.glossarist.com) links to hundreds of glossaries on hundreds of different subjects. Check and see if it has a glossary relevant to your product or service, and see if you can pull some words from there.

Keyword tools

Google's competitor Yahoo! has its own advertising network called Overture, and it offers a too to test potential advertising keywords. Browse to the Keyword Select Tool (http://inventory.overture.com/d/searchinventory/suggestion/) and take a few of your ideas for a spin. You'll find Yahoo! search counts for your word, and related terms.

Keep in mind that the most popular keywords might not always be the best choice for AdWords ads because the top terms are often very general.

You might also try Google's similar words features [Hack #8] to find variations of your core keywords.

Exploring Your Competitors' AdWords

Once you have a reasonable list of potential keywords for your ad, run them in the Google search engine. Google rotates advertisements based on the spending cap for each campaign, so even after running a search three or four times, you may see different advertisements each time. Use the AdWords scraper [Hack #87] to save these ads to a file and review them later.

If you find a potential keyword that apparently contains no advertisements, make a note of it. When you're ready to buy an AdWord, you'll have to check its frequency; it might not be searched often enough to be a lucrative keyword for you. But if it is, you've found a potential advertising spot with no other ads competing for searchers' attentions.

Hack 87. Scrape Google AdWords

Scrape the AdWords from a saved Google results page into a form suitable for importing into

Google's AdWordsthe text ads that appear to the right of the regular search resultsare delivered on a co
means that, even if you run a search for the same query word multiple times, you won't necessarily get

If you're considering using Google AdWords to run ads, you might want to gather up and save the ads t
you have to do a little scraping to get at that data.

Be sure to read "A Note on Spidering and Scraping" in Chapter 8 for some unders

This hack lets you scrape the AdWords from a saved Google results page andexport them to a comma-s

This hack requires a Perl module called `HTML::TokeParser` , which can be found at

http://search.cpan.org/search?query=htmL%3A%3Atokeparser&mode=all

You need to install it before the hack will run.

The Code

Save this code to a text file named *adwords.pl* :

```
#!/usr/bin/perl

# usage: perl adwords.pl results.html
#
use strict;
use HTML::TokeParser;

die "I need at least one file: $!\\n"
unless @ARGV;

my @Ads;
for my $file (@ARGV){
    # skip if the file doesn't exist
```

```
# you could add more file testing here.
# errors go to STDERR so they won't
# pollute our csv file

unless (-e $file) {
warn "What??: $file -- $! \\n-- skipping --\\n";
next;
}

# now parse the file
my $p = HTML::TokeParser->new($file);
while(my $token = $p->get_token) {
next unless $token->[0] eq 'S'
    and $token->[1] eq 'a'
    and $token->[2]{id} =~ /^aw\\d$/;
my $link = $token->[2]{href};
my $ad;
if($link =~ /pagead/) {
    my($url) = $link =~ /adurl=([^\\&]+)/;
    $ad->{href} = $url;
} elsif($link =~ m{^/url\\?}) {
    my($url) = $link =~ /\\&q=([^&]+)/;
    $url =~ s/%3F/\\?/;
    $url =~ s/%3D/=/g;
    $url =~ s/%25/%/g;
    $ad->{href} = $url;
}
$ad->{adwords} = $p->get_trimmed_text('/a');
$ad->{desc} = $p->get_trimmed_text('/font');
($ad->{url}) =  $ad->{desc} =~ /([\\S]+)$/;
push(@Ads,$ad);

}
}

print quoted( qw( AdWords HREF Description URL ) );
for my $ad (@Ads) {
    print quoted( @$ad{qw( adwords href desc url )} );
}


sub quoted {
    return join( ",", map { "\\"$_\\"" } @_ )."\\n";
}
```

Running the Hack

Call this script on the command line ["How to Run the Hacks" in the Preface], providing the name of the

```
% perl adwords.pl
```

```
                input.html

          >

                output.csv
```

input.html is the name of the Google results page that you've saved.*output.csv* is the name of the comn

```
% perl adwords.pl
                input.html
```

```
                input2.html

          >

                output.csv
```

The results appear in a comma-delimited format, as in these results for the query`new car` :

```
"AdWords","HREF","Description","URL"
"New Car","http://clickserve.dartsearch.net/link/click?lid=43000000003748919","Official
"Top New Car Quotes","http://www.carpricesecrets.com/L.php?x=7207976","Find out our Lowe
"Shop for New Cars","http://www.automotive.com/redir/newcar.asp?src_id=1611%26kw_sqid=12
...
```

Some lines are prematurely broken in this code listing for the purposes of publica

The hack returns the AdWords headline, the link URL, the description in the ad, and the URL on the ad (and see which companies are using which headlines and descriptions. Scraping AdWords is a quick way

Tim Allwine and Tara Calishain

Hack 88. Add Search to Your Site

Offer your readers a way to search your sitewithout hiring a team of developers to build it.

If you've used the `site:` advanced operator, you've already experienced Google's ability to search a single corner of the Web. Because Google indexes everything it can find on the public Web, you can use Google's `site:` operator to search your own web site. For example, if you want to look for every instance of the word *podcast* on your own web site, you could use the query `site:` *`www.example.com`* `podcast`, replacing *`www.example.com`* with your own domain.

People are used to using search features to find what they're looking for at web sites, and you can give your readers the power of Google with just a few lines of HTML. By putting a Google search form on your site, you can provide a convenient way to search through your pages, without hiring someone to develop a search feature for you.

The simplest way to add a Google form for searching your site is with a bit of HTML. Copy the following code into your site's HTML:

```
<form action="http://www.google.com/custom">
<input type="text" name="q" size="20" />
<input type="submit" value="Search" />
<div>
<input type="radio" name="sitesearch" value="" />
   Search the web
<input type="radio" name="sitesearch"
   value="insert your domain" checked="checked" />
   Search this site
</div>
</form>
```

Once you've added the code, you'll see a form on your site such as the one in Figure 7-4.

Figure 7-4. Google Free site search on a blog

Readers can type in a term, choose to search the entire Web or just your site, and click Search. They're taken to a standard Google results page. To limit your readers to searching only your site, you can remove the first radio button input tag and change the second tag to a hidden field. The new code looks like this:

```
<form action="http://www.google.com/custom">
<input type="text" name="q" size="20" />
<input type="submit" value="Search" />
<input type="hidden" name="sitesearch"
   value="insert your domain" />
</form>
```

This HTML presents your readers with a text field and search button, and results are limited to your site.

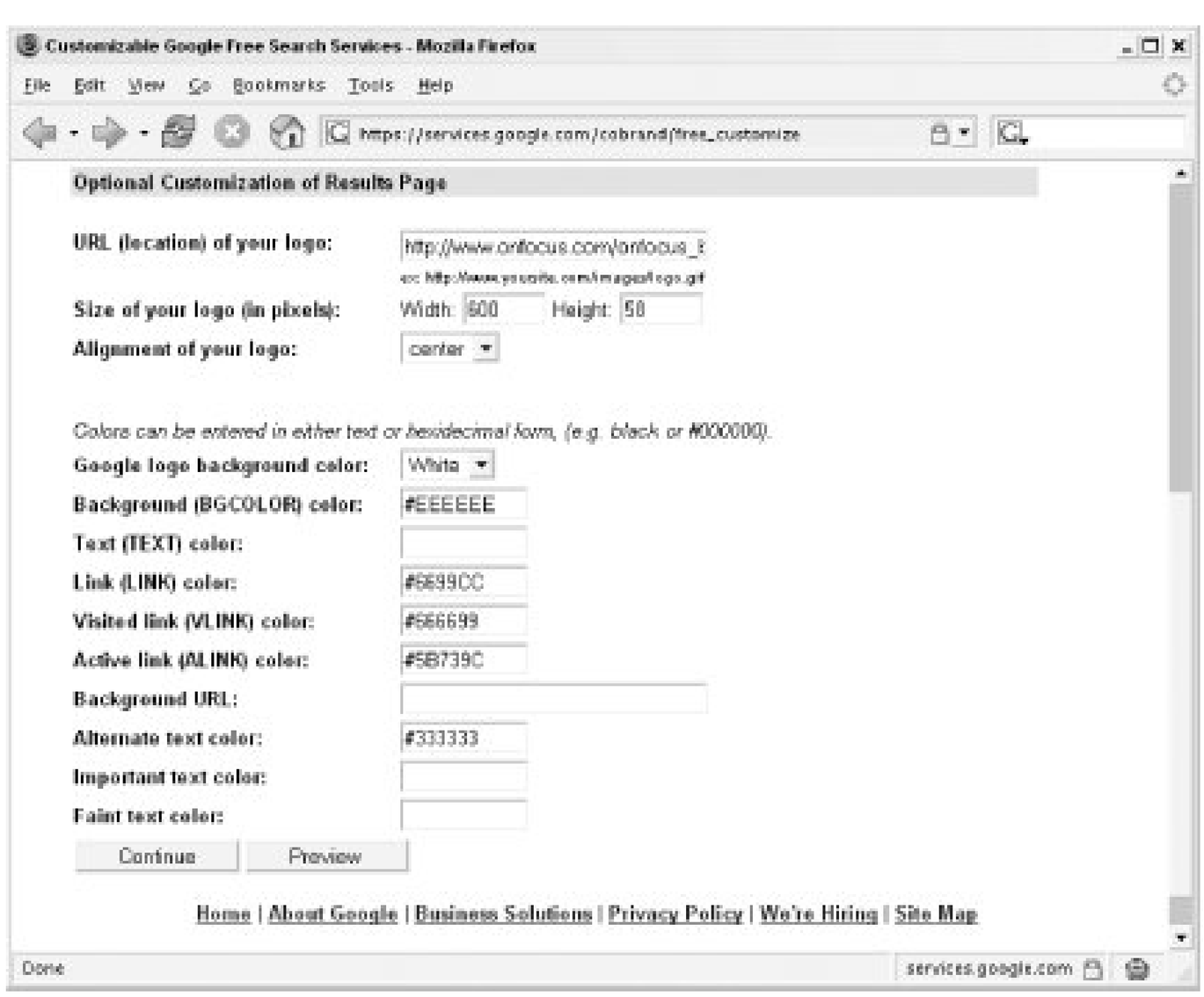
Google Free

The idea of sending your readers away from your site might not be appealing. After all, you want your readers to experience your brandingyour look and feelrather than Google's. You can reach a compromise with a feature called Google Free. This service lets you customize the Google results page with your own look and feel and a logo. The Google logo is still present, and you can't control the layout, but this process of *cobranding* can help your readers understand that they haven't strayed too far from your site.

Google Free has a Terms of Use policy (http://www.google.com/services/terms_free.html) that you should review before signing up. Basically, if you use the service as it is intended to be used, there won't be any problems. If you send the results to a pop-up window or interfere with the results page in any way, Google can terminate your use of the service.

To customize a Google results page, browse to the Google Free web site (http://www.google.com/services/free.html) and click "Customize Google for your site." From there, enter your site's domain and click Continue. Figure 7-5 shows the customization form that allows you to change several features on the Google results page.

Figure 7-5. Customizing a Google Free search



As you can see, you can add your own logo, which is an image file that resides on your server. You can also change several colors on the page, including background, link, and text colors. As you're adjusting settings, click the Preview button to see an example of your design. When your results page design is set, click Continue. Register for the service, and remember your login so you can change your Google Free design at any point. Click Continue, and Google provides the HTML necessary to add the form to your site.

Unlike the previous HTML example, the code Google provides is linked to your Google Free account s the results page is displayed with your design. Then, as your readers use the form, they go to your cobranded results page instead of the standard Google results page. Figure 7-6 shows an example of

a results page with a custom logo at the top and custom background and link colors.

Figure 7-6. A custom Google Free results page

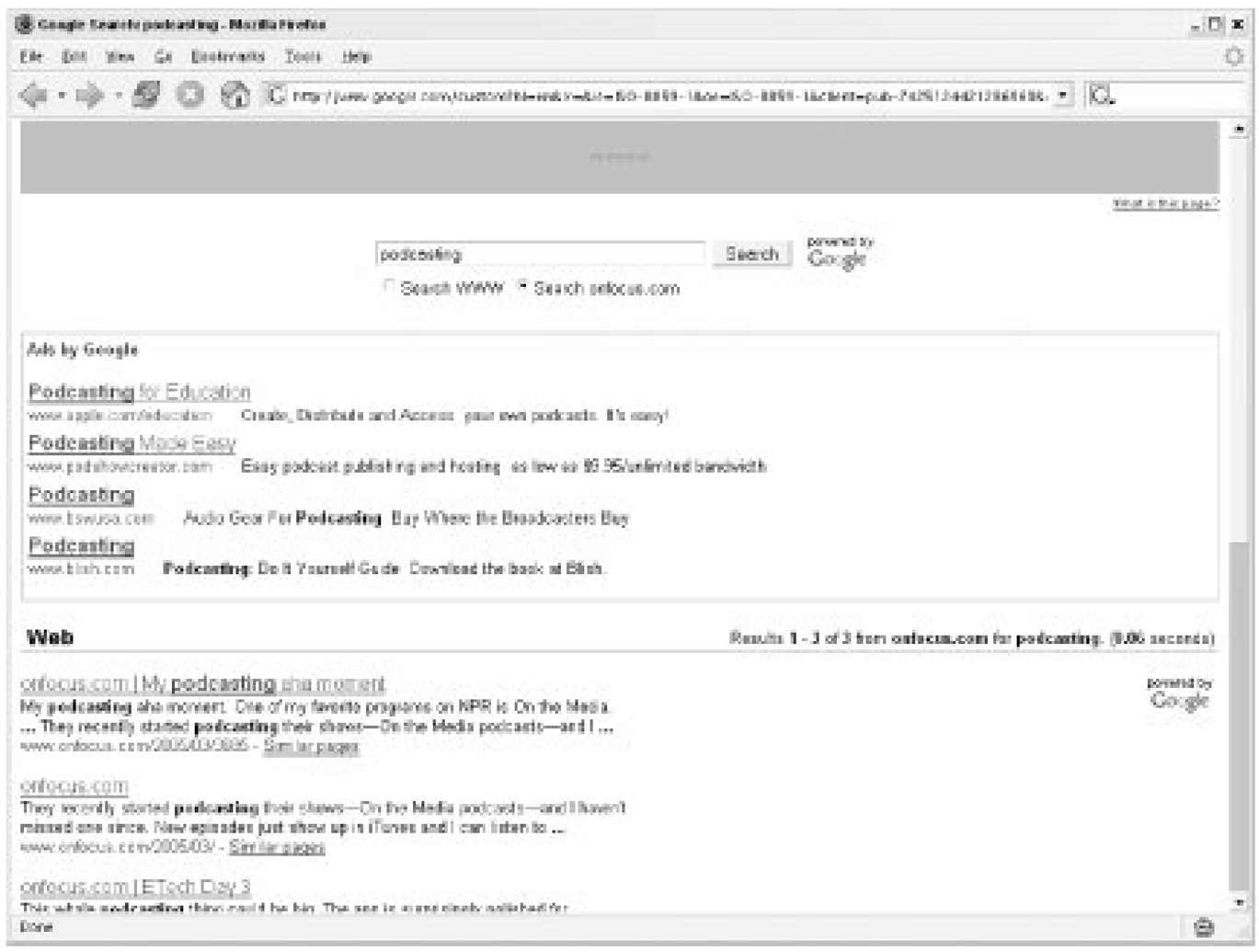
As you can see in Figure 7-6, Google adds advertising to the Google Free results page, just as it does on its standard search page.

AdSense for Search

If you think you should be paid for sending your readers to Google, take a look at "Tools for Webmasters" earlier in this chapter, specifically AdSense for Search (http://www.google.com/adsense/). Setting up AdSense for Search is similar to setting up a Google Free results page, and you might make some money in the process.

There are a few differences between Google Free and AdSense for Search that you should be aware of. First, the ads are much more prominent on AdSense results pages, as you can see in Figure 7-7.

Figure 7-7. A custom Google AdSense for Search results page



With AdSense results, the ads are displayed prominently on the page before the search results, instead of in a sidebar to the right of the search results. Also, the maximum custom logo size with AdSense for Search is 50x50 pixels, which is quite small. Google Free doesn't have that limitation. What makes AdSense for Search attractive is that in exchange for these limitations, you get a small cut of the advertising revenue.

No matter which route you choose, giving your readers the ability to search your site can help them find what they're after. And you can outsource the work of adding a search feature to Google for no cost beyond the time of copying and pasting some HTML.

Hack 89. Feed News to Your Web Site

Use RSS to display headlines about any topic on a remote web site.

Really Simple Syndication (RSS) and Atom are both standard XML formats designed for syndicating cont there are so many tools available that can work with RSS and Atom means that they really are simple tc news headlines in either format, and you can create a feed for any Google News search.

Imagine you run a site about gardening in Oregon and you want to keep your readers up to date on wha could point your readers to a Google News search results page, but you'd be sending them away from y the headlines of news stories about Oregon gardening on your own web site. This hack shows how these to an existing site with just a bit of scripting.

Finding a Feed

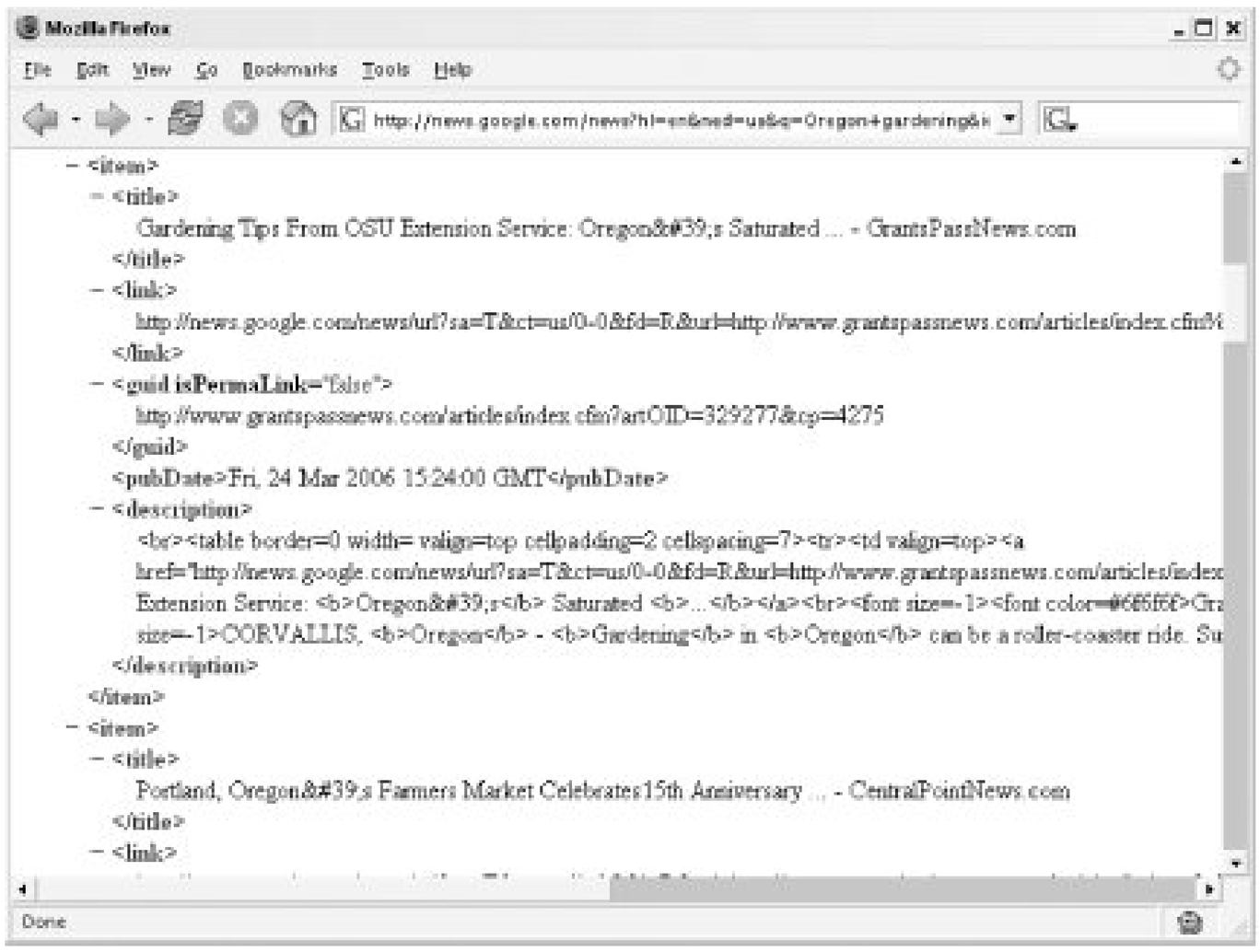
The first step to displaying a news feed on a remote site is finding the feed URL for the news topic you'r News (http://news.google.com), type in a query for your topic, and click Search News. For this example find relevant stories. You should see a list of stories such as the one shown in Figure 7-8.

Figure 7-8. Google News search results with feeds

On the left side of the page are links to the XML feeds in both RSS and Atom formats. This hack uses th
and choose Copy Link Location or Copy Shortcut. The feed URL is now on your virtual clipboard, ready f

If you click the icon instead of copying the URL (in a browser that displays XML), you'll get a sense of wl
7-9 shows what a Google News RSS feed looks like in Firefox.

Figure 7-9. A Google News RSS feed in Firefox



Looking at the feed, you can see that the basic tags of each RSS`<item>` that represent a news story are
and `<pubDate>` . The first two tags hold what you'd expect: the title of the story and a link to the story. T
HTML that includes a description and potentially related stories, and`<pubDate>` holds the date the story w
for a script to grab this information and display it on another site.

The Code

This script uses a module tailor-made for working with RSS called, appropriately enough,`XML::RSS` . The
shortcuts when parsing RSS, and it makes working with the format easy. To install`XML::RSS` on your syst
command line like this:

```
perl MCPAN e shell
cpan> install XML::RSS
```

The other module you'll need is`LWP::Simple` , which fetches the RSS from Yahoo!.

Add the following code to a file called *write_feed.pl* :

```
#!/usr/bin/perl
# write_feed.pl
# Accepts an RSS feed URL and prints as HTML.
# Usage: write_feed.pl <RSS Feed URL>

use strict;
use LWP;
use XML::RSS;

# Grab the incoming feed URL
my $url = @ARGV[0] or die "Usage: write_feed.pl <RSS Feed URL>\\n";

# Set the client for fetching pages
my $browser = LWP::UserAgent->new;
$browser->agent("Mozilla/5.01 (windows; U; NT4.0; en-us) Gecko/25250101");

# Fetch the Google RSS Feed for the query
my $rss = $browser->get($url);

# Parse the RSS
my $xmlrss = new XML::RSS(  );
$xmlrss->parse($rss->content);

# Print the feed Header with title
print "<h3>" . $xmlrss->channel('title') . "</h3>";
print "<ul>";

# Loop through the items returned, printing them out
foreach my $item (@{$xmlrss->{'items'}}) {
        my $title = $item->{'title'};
        my $url = $item->{'link'};
        my $description = $item->{'description'};
        print "<li><a href=\\"$url\\">$title</a></li>\\n";
}

# Print the feed Footer
print "</ul>";
```

The `XML::RSS parse( )` function turns the RSS from Yahoo! into an object that Perl can work with. From each item in the feed and prints the results with a bit of HTML formatting.

Running the Hack

To run the script, call it from a command line, supplying a feed URL:

```
perl write_feed.pl "
                    insert feed URL
                "
```

This prints the HTML to the command line, which probably isn't the effect you're after. To pipe the outpu end like this:

```
perl write_feed.pl "
                    insert feed URL
                " >
                    insert text filename
```

And here's a look at writing a Google News feed as HTML directly:

```
perl write_feed.pl "http://news.google.com/news?hl=en&ned=us&q=Oregon+gardening&ie=UTF-8
```

Once run, the newly created *or-garden.html* file contains the script's output, which looks something like

Figure 7-10. A converted Google News RSS feed at anothe

The final step to including this file in an existing web site is to drop it in as a *server-side include* . This ca an existing dynamically generated web page, like so:

```
<!--#include virtual="/or-garden.html" -->
```

Be sure to replace the name of the file with the name of your generated HTML file. This hack shows very can change the code toward the end of *write-feed.pl* to match the look and feel of your web site.

You can run this script manually once in a while, but it's much handier to let the server handle it. You ca run every few hours, and you won't ever have to touch it again. Any new stories about the topic also sh a quick way to keep visitors to your site up to date with stories about a topic that's important to them, a

PREV

Chapter 8. Programming Google

When search engines first appeared on the scene, they were more open to being spidered, scraped, and aggregated. Sites such as Excite and AltaVista didn't worry too much about the odd surfer using Perl to grab a slice of a page or meta-search engines including their results in aggregated search results. Sure, egregious data suckers might get shut out, but the search engines weren't worried about sharing their information on asmaller scale.

Google never took that stance. Instead, it has regularly prohibited meta-search engines from using its content without a license, and it tries its best to block unidentified web agents such as Perl's `LWP::Simple` module or even `wget` on the command line. Google has even been known to block IP address ranges for running automated queries.

Google had every right to do this; after all, it was its search technology, database, and computer power. Unfortunately, however, these policies meant that casual researchers and Google nuts, like you and I, couldn't play with its rich dataset in any automated way.

Google changed all that with the release of theGoogle Web API (http://api.google.com) in the spring of 2002. The Google Web API doesn't allow you to do every kind of search possiblefor example, it doesn't support the `phonebook:` syntaxbut it does make available Google's eight-billion-page web database so that developers can create their own interfaces and use Google search results to their liking.

> API stands for "Application Programming Interface," a doorway for programmatic access to a particular resource or applicationin this case, the Google index.

So how can you participate in all this Google API goodness?

You have to register for a *developer's key*, a login of sorts to the Google API. Each key affords its owner 1,000 Google Web API queries per day, after which you're out of luck until the next day. Even if you don't plan on writing any applications, having a key at your disposal is still useful. There are various third-party applications built on the Google API that you might want to visit and try out; some of these ask that you use your own key and allotted 1,000 queries.

Signing Up and Google's Terms

Signing up for a Google Web API developer's key is simple. First, create a Google account. The only requirements are a valid email address and a made-up password.

Not only can you use the Google Web API, but you can also use Personalized Search [Hack #12], web management of Google Alerts [Hack #47], post access to Google Groups [Hack #40], and use other services that require an account.

Of course, you must agree to Google's Terms and Conditions (http://www.google.com/apis/download.html) before you can proceed. In broad strokes, this says:

- Google exercises no editorial control over the sites that appear in its index. The Google API might return some results that you might find offensive.
- The Google API may be used for personal, noncommercial use only. It may not be used to sell a product or service or to drive traffic to a site for the sake of advertising sales.
- You can't noodle with Google's intellectual property marks that appear within the API.
- Google does not accept any liability for the use of its API. This is a beta program.
- You may indicate that the program you create uses the Google API, but not if the application(s) "(1) tarnish, infringe, or dilute Google's trademarks, (2) violate any applicable law, and (3) infringe any third-party rights." Any other use of Google's trademark or logo requires written consent.

Once you've entered your email address, created a password, and agreed to the Terms of Service, Google sends you an email message to confirm the legitimacy of your email address. The message includes a link for final activation of the account. Click the link to activate your account, and Google emails you your very own license key.

You've signed in and generated a key; you're all set! What now? If you don't intend to do any programming, just stop here. Put your key in a safe place and keep it on hand to use with any cool third-party Google API-based services you come across.

The Google Web APIs Developer's Kit

If you are interested in doing some programming, download the Google Web APIsDeveloper's Kit (http://www.google.com/apis/download.html). While not strictly necessary to any Google API programming that you might do, the kit contains much that is useful:

- A cross-platform WSDL file (see the section "What's WSDL?" later in this chapter)
- A Java wrapper library abstracting away some of the SOAP plumbing
- A sample .NET application
- Documentation, including JavaDoc and SOAP XML samples

Simply click the download link, unzip the file, and take a look at the*README.txt* file to get underway.

Using Your Google API Key

Whenever you send a request to the Google server in a program, you have to send your key along with it. Google checks the key and determines whether it's valid and you're still within your daily 1,000 query limit; if so, Google processes the request.

All the programs in this book, regardless of language and platform, provide a place to plug in your key. The key itself is just a string of random-looking characters (e.g., `12BuCK13mY5h0E/34KN0cK@ttH3Do0R`).

If you plan to make your hack available online for others to use, you might want to consider asking visitors to sign up for and use their own Google API keyat least optionally. A thousand queries per day really isn't that much, and should your hack become popular, you'll more than likely have a few unhappy visitors for whom it just doesn't work once you've used up your quota. You can see an example of this in action on Tara's GoogleJack! Page (http://www.researchbuzz.org/archives/001418.shtml); notice the spot in the GoogleJack! form for Google API Key.

A Perl hack usually includes a line such as the following:

```
...
# Your Google API developer's key.
my $google_key='insert key here';
...
```

The Java *GoogleAPIDemo* included in the Google Web APIs Developer's Kit is invoked on the command line, like so:

```
% java -cp googleapi.jar com.google.soap.search.GoogleAPIDemo
            insert_key_here  search ostrich
```

In both cases, *insert key here* or *insert_key_here* should be substituted with your own Google Web API key. For example, I would plug my made-up key into the Perl script as follows:

```
...
# Your Google API developer's key.
my $google_key='12BuCK13mY5h0E/34KN0cK@ttH3Do0R';
...
```

What's WSDL?

Pronounced "whiz-dill," WSDL stands for Web Services Description Language, an XML format for describing web services. The most useful bit of the Google Web API Developer's Kit is *GoogleSearch.wsdl*, a WSDL file that describes the Google API's available services, method names, and expected arguments to your programming language of choice.

Most of the hacks in this book assume that the *GoogleSearch.wsdl* file is in the same directory as the scripts you're writing, since this is probably the simplest setup. If you prefer to keep it elsewhere, be sure to alter the path in the script at hand. A Perl hack usually specifies the location of the WSDL file like so:

```
...
# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";
...
```

I like to keep such files together in a *library* directory, so I would make the following adjustment to the previous code snippet:

```
...
# Location of the GoogleSearch WSDL file.
my $google_wsdl = "/home/me/lib/GoogleSearch.wsdl";
...
```

Understanding the Google API Query

The core of a Google application is the query.Without the query, there's no Google data, and without that, you don't have much of an application. Because of its importance, it's worth taking a little time to look into the anatomy of a typical query.

Query Essentials

The command in a typical Perl-based Google API application that sends a query to Google looks like this:

```
my $results = $google_search ->
  doGoogleSearch(
    key, query, start, maxResults,
    filter, restrict, safeSearch, lr,
    ie, oe
  );
```

Usually, the items within the parentheses are variables, numbers, or Boolean values (`1`, `true`, or `false`). In the previous example, I've included the names of the arguments themselves rather than sample values so that you can see their definitions here:

key

This is where you put your Google API developer's key. Without a key, the query won't go very far.

query

This is your query, composed of keywords, phrases, and special syntaxes.

start

Also known as the *offset*, this integer value specifies at what result to start counting when determining which 10 results to return. If this number were`16`, the Google API would return results 1625; if `300`, results 300309 (assuming, of course, that your query found that many results). This is known as a *zero-based index*, since counting starts at 0, not 1. The first result is result 0, and the 999th result is 998. Admittedly, it's a little odd, but you get used to it quicklyespecially if you go on to do a lot of programming. Acceptable values are`0` to `999` because Google returns only up to 1,000 results per query.

maxResults

This integer specifies the number of results that you want the API to return. The API returns results in batches of up to 10, so acceptable values are `1` tHRough `10`.

filter

You might think that the *`filter`* option has something to do with the SafeSearch filter for adult content. It doesn't. This Boolean value (`TRue` or `false`) specifies whether your results go through automatic query filtering, removing near-duplicate content (titles and snippets that are very similar) and multiple (more than two) results from the same host or site. With filtering enabled, only the first two results from each host are included in the result set.

restrict

No, *`restrict`* doesn't have anything to do with SafeSearch either. It allows you to restrict your search to one of Google's topical searches or to a specific country. Google has four topic restricts: U.S. Government (`unclesam`), Linux (`linux`), Macintosh (`mac`), and FreeBSD (`bsd`). The complete country list is in the Google Web API documentation. To leave your search unrestricted, leave this option blank (usually signified by empty quotation marks:`""`).

safeSearch

Now here's the SafeSearch filtering option. This Boolean (`true` or `false`) specifies whether results returned are filtered for questionable (read: adult) content.

lr

This stands for *language restrict*, and it's a bit tricky. Google has a list of languages in its API documentation to which you can restrict search results, or you can simply leave this option blank and have no language restrictions.

There are several ways you can restrict results to a particular language. First, you can simply include a language code. If you want to restrict results to English, for example, use`lang_en`. But you can also restrict results to more than one language, separating each language code with a `|` (pipe), signifying `OR`. `lang_en|lang_de`, then, constrains results to only those "in English or German."

You can omit languages from results by prepending them with a`-`(minus sign). `lang_en` returns all results but those in English.

ie

This stands for *input encoding* and allows you to specify the character encoding used in the query you're feeding the API. Google's documentation says, "Clients should encode all request data in UTF-8 and should expect results to be in UTF-8." In the first iteration of Google's API program, the Google API documentation offered a table of encoding options (`latin1`, `cyrillic`, etc.), but now everything is `UTF-8`. In fact, specifying anything other than UTF-8 is summarily

ignored.

oe

This stands for *output encoding*. As with input encoding, everything's `UTF-8`.

A Sample

Enough with the placeholders; what does an actual query look like?

Take, for example, a query that uses variables for the key and the query, requests 10 results starting at result number 100 (actually the 101st result), and specifies that filtering and SafeSearch be turned on. That query in Perl would look like this:

```
my $results = $google_search ->
  doGoogleSearch(
    $google_key, $query, 100, 10,
    "true", "", "true", "",
    "utf8", "utf8"
  );
```

Note that the key and query could just as easily have been passed along as quote-delimited strings:

```
my $results = $google_search ->
  doGoogleSearch(
    "12BuCK13mY5h0E/34KN0cK@ttH3Do0R", "+paloentology +dentistry" , 100, 10,
    "true", "", "true", "",
    "utf8", "utf8"
  );
```

While things appear a little more complex when you start fiddling with the language and topic restrictions, the core query remains mostly unchanged; only the values of the options change.

Intersecting Country, Language, and Topic Restrictions

Sometimes you want to restrict your results to a particular language in a particular country, or to a particular language, particular country, and particular topic. Now here's where things start to look a little on the odd side.

The rules are as follows:

- Omit something by prepending it with a `-` (minus sign).
- Separate restrictions with a `.` (period, or full stop); spaces are not allowed.
- Specify an `OR` relationship between two restrictions with a `|` (pipe).

- Group restrictions in parentheses. You can have parentheses within parentheses (*nested parentheses*) for fine-grained control over grouping in your queries.

Let's say you want a query to return results in French, draw only from Canadian sites, and focus only within the Linux topic. Your query would look something like this:

```
my $results = $google_search ->
  doGoogleSearch(
    $google_key, $query, 100, 10,
    "true", "linux.countryCA", "true", "lang_fr",
    "utf8", "utf8"
  );
```

For results from Canada or from France, you would use:

```
"linux.(countryCA|countryFR)"
```

Or maybe you want results in French, but from anywhere but France:

```
"linux.(-countryFR)"
```

For a comprehensive list of restricts, see Section 2.4, "Restricts," of*APIs_Reference.html*, part of the Google API documentation.

Putting Query Elements to Use

You can use the different elements of the query as follows:

Using SafeSearch

If you're building a program for family-friendly use, you'll probably want SafeSearch turned on as a matter of course. But you can also use it to compare safe and unsafe results. "SafeSearch Certify URLs" [Hack #24] does just that. You can create a program that takes a word from a web form and checks its counts in filtered and unfiltered searches, providing a*naughty rating* for the word based on the counts.

Setting search result numbers

Whether you request 1 or 10 results, you still use one of your developer key's daily dose of 1,000 Google Web API queries. Wouldn't it then make sense to always request 10? Not necessarily; if you're using only the top resultto bounce the browser to another page, generate a random query string for a password, or whateveryou might as well add even the minutest amount of speed to your application by not requesting results you're just going to throw out or ignore.

Searching different topics

With four different specialty topics available for searching through the Google API, dozens of different languages, and dozens of different countries, there are thousands of combinations of topic/language/country restrictions that you can work through.

Consider an *open source country* application. You can create a list of keywords very specific to open source (such as `linux`, `perl`, etc.) and create a program that cycles through a series of queries that restricts your search to an open source topic (such as `linux`) and a particular country. So, you might discover that `perl` was mentioned in France in the `linux` topic 15 times, in Germany 20 times, and so on.

You can also concentrate less on the program itself and more on an interface to access these variables. How about a form with pull-down menus that allows you to restrict your searches by continent (instead of country)? You can specify which continent in a variable that's passed to the query. Or how about an interface that lets the user specify a topic and cycles through a list of countries and languages, pulling result counts for each one?

PREV

Understanding the Google API Response

While the Google API grants you programmatic access to Google's Web index, it doesn't provide all the functionality available through the Google.com web site's search interface.

Can Do

The Google API, in addition to simple keyword queries,supports the following special syntaxes ["Special Syntax" in Chapter 1]:

- `site:`
- `daterange:`
- `intitle:`
- `inurl:`
- `allintext:`
- `allinlinks:`
- `filetype:`
- `info:`
- `link:`
- `related:`
- `cache:`

Can't Do

The Google API does not support these special syntaxes:

- `phonebook:`
- `rphonebook:`
- `bphonebook:`

While queries of this sort provide no individual results, aggregate result data is sometimes returned and can prove rather useful. googly.php [Hack #99], for instance, displays the number of results

(`estimatedTotalResultsCount`).

The 10-Result Limit

While searches through the standard Google.com home page can be tuned ["Setting Preferences" in Chapter 1] to return 10, 20, 30, 50, or 100 results per page, the Google Web API limits the number to 10 per query. This doesn't mean, mind you, that the rest are not available to you, but it takes a wee bit of creative programming that entails looping through results, 10 at a time[Hack #93].

What's in the Results

The Google API provides both aggregate and per-result data in its result set.

Aggregate data

The aggregate data (information on the query itself and on the kinds and number of results that query turned up) consists of:

`<documentFiltering>`

A Boolean (`true`/`false`) value specifying whether results were filtered for very similar results or for those that come from the same web host.

`<searchComments>`

Any commentary (e.g., a note about stop words being removed) Google might throw in that would usually be displayed just beneath the search box on a typical Google results page.

`<estimatedTotalResultsCount>`

An estimate of how many results might be found for your search in the Google index. This number may vary from invocation to invocation, moment to momentthus the "estimated" proviso.

`<estimateIsExact>`

Google may sometimes be sure of its `estimatedTotalResultsCount`, in which case `estimateIsExact` is set to `true`.

`<resultElements>`

The individual results themselves, returned as an array.

`<searchQuery>`

Your Google query, right back at you.

`<startIndex>`

The index of the first result in the current array of results. Assuming your query asked for a `start` of `0`, the first result would have a `startIndex` of `1`. If you asked for a `start` of `25`, `startIndex` would be `26`. Yes, I know it's confusing that `start` is 0-based, while `startIndex` is 1-based, but that's the way the cookie crumbles, I'm afraid.

`<endIndex>`

The index of the last result in the current array of results. This is always whatever you set as `start` + `maxResults` in your query, unless the total is greater than the number of `estimatedTotalResultsCount`, in which case it is simply `estimatedTotalResultsCount`.

`<searchTips>`

Provides suggestions on how to better use Google; suitable for displaying to the end user.

`<directoryCategories>`

A list of directory categories, if any, associated with the query.

`<searchTime>`

The time spent by the Google server (in seconds) on your search.

Individual search result data

The "guts" of a search resultthe URLs, page titles, and snippetsare returned in a`<resultElements>` list. Each result consists of the following elements:

`<summary>`

The Google Directory summary, if available.

`<URL>`

The search result's URL; consistently starts with`http://`.

`<snippet>`

A brief excerpt of the page with query terms highlighted in bold (HTML`<b>` `</b>` tags).

`<title>`

The page title in HTML.

`<cachedSize>`

The size in kilobytes (KB) of the Google-cached version of the page, if available.

`<relatedInformationPresent>`

If set to `1`, means that a `related:` search on the current result's URL will turn up something of use.

`<hostName>`

When you set `filter` to `TRue` in your query, only two results from the same hostname are included in your set of results. In the second of these results,`hostName` is set to the host from which the result came.

`<directoryTitle>`

The title under which this result appears in the Google Directory (http://directory.google.com, a.k.a. the Open Directory Project), if it is in the directory at all.

`<directoryCategory>`

The Google Directory category, if any, in which this result can be found.`<directoryCategory>` consists of `<fullViewableName>`, the name given to the category itself, and `<specialEncoding>`, which is any special encoding assigned to the directory category at hand.

You no doubt notice the conspicuous absence of PageRank. Google does not make PageRank available through anything but the official Google Toolbar[Hack #53]. You can get a general idea of a page's popularity by looking over the *popularity bars* in the Google Directory.

PREV

Beyond Web APIs

This chapter focuses on the Google Web API, but Google does make other data available through separate APIs. To find a current list of other ways to incorporate Google data into your applications, take a look at the Google APIs list (http://code.google.com/apis.html). At the time of this writing, some of the other APIs include programmatic access to AdWords [Hack #85], Blogger [Hack #46], Google Calendar (http://code.google.com/apis/gdata/calendar.html), Google Maps [Hack #67], and Google Homepage [Hack #44].

PREV

A Note on Spidering and Scraping

A small percentage of the hacks in this book involve*spidering*, or meandering through sites and scraping data from their web pages to be used outside of their intended context. Given that you have the Google API at your disposal, why then would you resort to spidering and scraping?

The main reason is simply that you can't gain access to everything Google through the API. While it nicely serves the purposes of searching the Web programmatically, the API (at the time of this writing) doesn't go any further than Google's main Web Search index. And it's even limited in what you can pull from the index. You can't do a Google PhoneBook search, trawl Google News, leaf through Google Catalogs, or interact in any way with most of Google's other specialty search properties.

So, while Google provides a good start in its API, there are more often than not situations in which you can't get to the Google data you're most interested in. Not to mention being unable to combine what you can get through the Google API with data from other sites without such a convenience. This is where spidering and scraping come in.

There are a few things that you need to keep in mind when resorting to scraping:

Scrapers are brittle

> The life of a scraper lasts only as long as the page it is scraping remains formatted in the same manner. When the page changes, your scraper canand most likely willbreak.

Tread lightly

> Tread lightly, taking only as much as you need and no more. If all you need is the data from the page that is already open in your browser, save the source and scrape that.

Maximize your effectiveness

> Make the most out of every page you scrape. Rather than hitting Google again and again for the next 10 results, and then the next 10, set your preferences ["Setting Preferences" in Chapter 1] so that you get all you can on a single page. For instance, set your preferred number of results to 100 rather than the default of 10.

Mind the terms of service

> It might be tempting to go one step further and create programs that automate retrieving and scraping, but you're more likely to tread on the toes of the site owner (Google or otherwise) and be asked to leave or simply be locked out.

So use the API whenever you can, scrape only when you absolutely must, and mind your Ps and Qs when fiddling about with other people's data.

PREV

Hack 90. Program Google in Perl

This simple script illustrates the basics of programming the Google Web API with Perl and lays the groundwork for the lion's share of hacks to come.

The vast majority of hacks in this book are written in Perl. While the specifics vary from hack to hack much of the busywork of querying the Google API and looping over the results remain essentially the same.

This hack is utterly basic, providing a foundation on which to build more complex and interesting applications. If you haven't done anything of this sort before, this hack is a good starting point for experimentation. It simply submits a query to Google and prints out the results.

The Code

In addition to the Google API Developer's Kit, you need to install the SOAP::Lite Perl module[Hack #91] before running this hack.

Type the following code into your preferred plain-text editorbe it Notepad, TextEdit, or a command-line editor such as vi or Emacsand save it to a file named*googly.pl*, replacing *`insert key here`* with your Google developer's key:

```
#!/usr/local/bin/perl
# googly.pl
# A typical Google Web API Perl script.
# Usage: perl googly.pl <query>

# Your Google API developer's key.
my $google_key='insert key here';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

use strict;

# Use the SOAP::Lite Perl module.
use SOAP::Lite;

# Take the query from the command line.
my $query = shift @ARGV or die "Usage: perl googly.pl <query>\\n";

# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
```

```
my $google_search = SOAP::Lite->service("file:$google_wsdl");

# Query Google.
my $results = $google_search ->
    doGoogleSearch(
      $google_key, $query, 0, 10, "false", "",  "false",
      "", "latin1", "latin1"
    );

# No results?
@{$results->{resultElements}} or exit;

# Loop through the results.
foreach my $result (@{$results->{resultElements}}) {
 # Set the results as variables
  my $title = $result->{title} || "no title";
  my $url = $result->{URL};
  my $snippet = $result->{snippet} || 'no snippet';

 # Strip HTML from the results
  $title =~ s!<[^>]+>!!gis;
  $snippet =~ s!<[^>]+>!!gis;

 # Print out the main bits of each result
 print
  join "\\n",
  $title,
  $url,
  $snippet,
  "\\n";
}
```

Running the Hack

Run this script from the command line ["How to Run the Hacks" in the Preface], passing it any Google search you want to run, like so:

```
$ perl googly.pl  "
                  query keywords
                "
```

Here's a sample run:

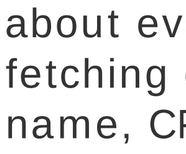
```
% perl googly.pl
Usage: perl googly.pl <query>
% perl googly.pl "
```

```
                    learning perl

                  "

...
learn.perl.org: Online Perl Library
http://learn.perl.org/library/
It's divided into three primary sections: learning about programming, learni
ng about Perl, and advanced topics. The latter includes brief sections on ...

oreilly.com -- Online Catalog: Learning Perl, Third Edition
http://www.oreilly.com/catalog/lperl3/
Learning Perl is the quintessential tutorial for the Perl programming language.
The third edition has not only been updated to Perl Version 5.6, ...
...
```

The first attempt doesn't specify a query, so it triggers a usage message and doesn't go any further. The second searches for `learning perl` and prints out the results.

PREV

Hack 91. Install the SOAP::Lite Perl Module



Install the SOAP::Lite Perl module, the backbone of the vast majority of hacks in this book.

The `SOAP::Lite` (http://www.soaplite.com) Perl module is the de facto standard for interfacing with SOAP-based web services from Perl. As such, it is used extensively throughout this book and in hacks you might stumble across online.

While teaching you how to install Perl modules is beyond the scope of this book, we've included these instructions to bootstrap your Google hacking so you don't need to wander off in search of a Perl book.

Unfortunately, it's rather common for Internet service providers (ISPs) not to make `SOAP::Lite` available to their users. In many cases, ISPs are rather restrictive in general about what modules they make available and scripts they allow users to execute. Others are more accommodating and more than willing to install Perl modules on request. Before expending time and brainpower installing `SOAP::Lite` yourself, ask your service provider if it's already there or if it can be installed for you.

Probably the easiest way to install `SOAP::Lite` is via another Perl module, CPAN, included with just about every modern Perl distribution. The CPAN module automates the installation of Perl modules, fetching components and any prerequisites from the Comprehensive Perl Archive Network (thus the name, CPAN) and building the whole kit and caboodle on the fly.

> CPAN installs modules into standard system-wide locations and, therefore, assumes you're running as the root user. If you have no more than regular user access, you have to install `SOAP::Lite` and its prerequisites by hand (see "Unix Installation by Hand" later in this chapter).

Unix and Mac OS X Installation via CPAN

Assuming you have the CPAN module, have root access, and are connected to the Internet, installation should be no more complicated than this:

```
% su
Password:
# perl -MCPAN -e shell
cpan shell -- CPAN exploration and modules installation (v1.52)
ReadLine support available (try \Q\Qinstall Bundle::CPAN'')
cpan> install SOAP::Lite
```

Or, if you prefer one-liners:

```
% sudo perl -MCPAN -e 'install SOAP::Lite'
```

In either case, go grab yourself a cup of coffee, meander in the garden, read the paper, and check back once in a while. Your terminal's sure to be riddled with incomprehensible gobbledygook that you can, for the most part, summarily ignore. You may be asked a question or three; in most cases, simply hitting Enter to accept the default answer does the trick.

Unix Installation by Hand

If CPAN installation didn't quite work as expected, you can of course install `SOAP::Lite` by hand. Download the latest version from SOAPLite.com (http://www.soaplite.com/), unpack, and build it like so:

```
% tar xvzf SOAP-Lite-latest.tar.gz
SOAP-Lite-0.55
SOAP-Lite-0.55/Changes
...
SOAP-Lite-0.55/t/37-mod_xmlrpc.t
SOAP-Lite-0.55/t/TEST.pl
% cd SOAP-Lite-
               0.XX

% perl Makefile.PL
We are about to install SOAP::Lite and for your convenience will
provide you with list of modules and prerequisites, so you'll be able
to choose only modules you need for your configuration.
XMLRPC::Lite, UDDI::Lite, and XML::Parser::Lite are included by default.
Installed transports can be used for both SOAP::Lite and XMLRPC::Lite.
Client HTTP support (SOAP::Transport::HTTP::Client)        [yes]
Client HTTPS support (SOAP::Transport::HTTPS::Client...    [no]
...
SSL support for TCP transport (SOAP::Transport::TCP)       [no]
Compression support for HTTP transport (SOAP::Transport... [no]
Do you want to proceed with this configuration? [yes]
During "make test" phase we may run tests with several SOAP servers
that may take long and may fail due to server/connectivity problems.
Do you want to perform these tests in addition to core tests? [no]
Checking if your kit is complete...
Looks good
...
% make
mkdir blib
mkdir blib/lib
...
```

```
% make test
PERL_DL_NONLAZY=1 /usr/bin/perl -Iblib/arch -Iblib/lib
-I/System/Library/Perl/darwin -I/System/Library/Perl -e 'use
Test::Harness qw(&runtests $verbose); $verbose=0; runtests @ARGV;'
t/01-core.t t/02-payload.t t/03-server.t t/04-attach.t t/05-customxml.t
t/06-modules.t t/07-xmlrpc_payload.t t/08-schema.t t/01-core.....
...
# sudo make install
Password:
Installing /Library/Perl/XMLRPC/Lite.pm
Installing /Library/Perl/XMLRPC/Test.pm
...
```

If, during the `perl Makefile.PL` phase, you run into any warnings about installing prerequisites, instal each in turn before attempting to install `SOAP::Lite` again. A typical prerequisite warning looks something like this:

```
Checking if your kit is complete...
Looks good
Warning:
               prerequisite HTTP::Daemon
 failed to load: Can't locate
HTTP/Daemon.pm in @INC (@INC contains: /System/Library/Perl/darwin
/System/Library/Perl /Library/Perl/darwin /Library/Perl /Library/Perl
/Network/Library/Perl/darwin /Network/Library/Perl
/Network/Library/Perl .) at (eval 8) line 3.
```

If you have little more than user access to the system and still insist on installing`SOAP::Lite` yourself, you have to install it and all its prerequisites somewhere in your home directory.*~/lib*, a *lib* directory in your home directory, is as good a place as any. Inform Perl of your preference like so:

```
% perl Makefile.PL LIB=
               /home/login/lib
```

Replace */home/login/lib* with an appropriate path.

Windows Installation via PPM

If you're running Perl under Windows, chances are it's ActiveState'sActivePerl (http://www.activestate.com/Products/ActivePerl/). Thankfully, ActivePerl's outfitted with a CPAN-like module installation utility. TheProgrammer's Package Manager (PPM, http://aspn.activestate.com/ASPN/Downloads/ActivePerl/PPM/) grabs nicely packaged module bundles from the ActiveState archive and drops them into place on your Windows system with little need of help from you.

Simply launch PPM from inside a DOS terminal window and tell it to install the`SOAP::Lite` bundle:

```
C:>ppm
PPM interactive shell (2.1.6) - type 'help' for available commands.
PPM> install SOAP-Lite
```

If you're running a reasonably recent build, you're probably in for a pleasant surprise:

```
C:>ppm
PPM interactive shell (2.1.6) - type 'help' for available commands.
PPM> install SOAP-Lite
Note: Package 'SOAP-Lite' is already installed.
```

A Note About Expat

There's a little something called Expat (http://expat.sourceforge.net) that, more often than not, is the one hiccup in the installation processparticularly when you install with the CPAN module or by hand. Expat is an XML parser library written in the C programming language and underlying many of the XML modules you might use. Fortunately, it is probably installed by default on the system you're using, but if it isn't there, you won't get very far.

The easiest way to install Expat under Mac OS X or Unix/Linux goes a little something like this:

```
$ curl -O http://easynews.dl.sourceforge.net/sourceforge/expat/expat-
                  X
                 .
                  XX
                 .
                  X
                 .tar.gz
$ tar -xvzf xpat-
                  X
                 .
                  XX
                 .
                  X
                 .tar.gz
...
```

```
$ cd expat-
                  X
               .
                  XX
               .
                  X

$ ./configure
$ make
$ sudo make install
```

PREV

Hack 92. Program Google with the Net::Google Perl Module

Here's a crisp, clean, object-oriented alternative to programming Google with Perl and the SOAP::Lite module.

An alternative, more object-oriented Perl interface to the Google API is Aaron Straup Cope's `Net::Google` (http://search.cpan.org/search?query=net+google&mode=module). While not fundamentally different from using SOAP::Lite [Hack #91] (as we do throughout this book), constructing Google API queries and dealing with the results is a little cleaner.

There are three main Google API interfaces defined by the module: `search( )`, `spelling( )`, and `cache( )`. These talk to the Google Web Search engine, spellchecker, and Google cache, respectively.

To provide a side-by-side comparison to googly.pl [Hack #90], the typical `SOAP::Lite`-based way to talk to the Google API, we've provided a script identical in function and almost identical in structure.

The Code

You'll still need `SOAP::Lite` and a couple other prerequisites to use `Net::Google`.

> For a step-by-step guide to installing `Net::Google` on Windows, see John Bokma's excellent tutorial "Net-Google with ActiveState Perl" (http://johnbokma.com/perl/net-google.html).

Save the following script as *net_googly.pl*, replacing *`insert key here`* with your Google developer's key:

```
#!/usr/local/bin/perl
# net_googly.pl
# A typical Google API script using the Net::Google Perl module.
# Usage: perl net_googly.pl <query>

use strict;

# Use the Net::Google Perl module.
use Net::Google;

# Your Google API developer's key.
```

```
use constant GOOGLE_API_KEY => '1BcTFcWyRzzIb/dggoXyAB5KjOFYUtjE';

# Take the query from the command line.
my $query = shift @ARGV or die "Usage: perl net_googly.pl <query>\\n";

# Create a new Net::Google instance.
my $google = Net::Google->new(key => GOOGLE_API_KEY);

# And create a new Net::Google search instance.
my $search = $google->search(  );

# Build a Google query.
$search->query($query);
$search->starts_at(0);
$search->max_results(10);
$search->filter(0);

# Query Google.
$search->results(  );

# Loop through the results.
foreach my $result ( @{$search->results(  )} ) {
 # Set the results as variables
  my $title = $result->title(  ) || "no title";
  my $url = $result->URL(  );
  my $snippet = $result->snippet(  ) || 'no snippet';

 # Strip HTML from the results
  $title =~ s!<[^>]+>!!gis;
  $snippet =~ s!<[^>]+>!!gis;

 # Print out the main bits of each result.
 print
  join "\\n",
  $title,
  $url,
  $snippet,
  "\\n";
}
}
```

Notice that the code is almost identical to that of*googly.pl*. The only real changes are cleaner object-oriented method calls for setting query parameters and dealing with the results. So, rather than passing a set of parameters to a `SOAP::Lite` service call like this:

```
doGoogleSearch(
      $google_key, $query, 0, 10, "false", "",  "false",
      "", "latin1", "latin1"
);
```

set these parameters individually like this:

```
$search->query($query);
$search->starts_at(0);
$search->max_results(10);
$search->filter(0);
```

Not much difference, but definitely cleaner.

Running the Hack

Invoke the hack on the command line in the same manner you did in "Program Google in Perl" [Hack #90]:

```
$ perl net_googly.pl  "
                    query keywords
                "
```

The results will be the same.

Hack 93. Loop Around the 10-Result Limit

If you want more than 10 results, you'll have to loop.

The Google API returns only 10 results per query, which is plenty for some queries, but, for most applica have to loop, querying for the next set of 10 each time. The first query returns the top 10; the next, 11

This hack builds on the basic Perl script *googly.pl* , introduced in "Program Google in Perl" [Hack #90] . using, you have to create a loop.

Bear in mind that each and every query counts against your daily allotment. Loop 3 doesn't seem like much given your quota of 1,000 queries a day, you'd be surprised where they all went.

The Code

In addition to the Google API Developer's Kit, you need to install the `SOAP::Lite` Perl module [Hack #91]

Save the following code to a text file named *looply.pl* . Again, remember to replace *`insert key here`* with are needed to support looping through more than the first 10 results are in bold:

```
#!/usr/local/bin/perl
# looply.pl
# A typical Google Web API Perl script.
# Usage: perl looply.pl <query>

# Your Google API developer's key.
my $google_key='insert key here ';

# Location of the GoogleSearch WSDL file.
my $google_wsdl = "./GoogleSearch.wsdl";

# Number of times to loop, retrieving 10 results at a time
my $loops = 3; # 3 loops x 10 results per loop = top 30 results

use strict;

# Use the SOAP::Lite Perl module.
use SOAP::Lite;

# Take the query from the command line.
```

```
my $query = shift @ARGV or die "Usage: perl looply.pl  <query>\\n";

# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
my $google_search = SOAP::Lite->service("file:$google_wsdl");

# Keep track of result number.
my $number = 0;

for (my $offset = 0; $offset <= ($loops-1)*10; $offset += 10) {

# Query Google.
my $results = $google_search ->
    doGoogleSearch(
      $google_key, $query, 0, 10, "false", "",  "false",
      "", "latin1", "latin1"
    );

# No sense continuing unless there are more results
last unless @{$results->{resultElements}};

# Loop through the results.
foreach my $result (@{$results->{resultElements}}) {
 # Set the results as variables
  my $title = $result->{title} || "no title";
  my $url = $result->{URL};
  my $snippet = $result->{snippet} || 'no snippet';

 # Strip HTML from the results
  $title =~ s!<[^>]+>!!gis;
  $snippet =~ s!<[^>]+>!!gis;

 # Print out the main bits of each result
 print
  join "\\n",
  ++$number,
  $title,
  $url,
  $snippet,
  "\\n";
}}
```

Notice that the script tells Google which set of 10 results it's after by passing an offset (`$offset`). The of

Running the Hack

Run this script from the command line ["How to Run the Hacks" in the Preface], passing it your Google

```
$ perl looply.pl   "query keywords "
```

Here's a sample run:

```
$ perl looply.pl
Usage: perl looply.pl <query>% perl looply.pl "perl loops"
1
Perl Basics: Loops
http://www.pageresource.com/cgirec/ptut9.htm
An introduction to the basics of using Perl loops.
...
29
comp.lang.perl: Perl loops should use break, not last
http://coding.derkeiler.com/Archive/Perl/comp.lang.perl/2005-01/0040.html
when inside a Whileor Do loop, ... (sci.math.symbolic); Perl loops should use  break, nc

30
comp.lang.perl: Re: Perl loops should use break, not last
http://coding.derkeiler.com/Archive/Perl/comp.lang.perl/2005-02/0001.html
in a tight loop. ... Perl is simply not suited to this ... (comp.lang.perl.modules);  Re
```

The first attempt doesn't specify a query, so it triggers a usage message and doesn't go any further. Th
the *googly.pl* script, but now the number of results you net is limited only by your specified loop count (i

Hacking the Hack

Alter the value assigned to the `$loops` variable in the *looply.pl* script to change the number of results. Fo

```
# Number of times to loop, retrieving 10 results at a time.
my $loops = 9; # 9 loops x 10 results per loop = top 90 results
```

Hack 94. Program Google in Java

Programming the Google Web API in Java is a snap, thanks to the functionality packed into th

Thanks to the Java Archive (JAR) file included in the Google Web API Developer's Kit, programming the *com.google.soap.search* , a nice, clean wrapper around the underlying Google SOAP, along with the Apa parser and Apache SOAP (http://xml.apache.org/soap/) stack, among others.

In addition to the *googleapi.jar* file included in the Google API Developer's Kit, you n http://java.sun.com/downloads/) to compile and run this hack.

The Code

Save the following code to a file called *Googly.java* , replacing *insert key here* with your Google develo

```
// Googly.java
// Bring in the Google SOAP wrapper.
import com.google.soap.search.*;
import java.io.*;

public class Googly {
  // Your Google API developer's key.
  private static String googleKey = "insert key here";
  public static void main(String[] args) {
    // Make sure there's a Google query on the command line.
    if (args.length != 1) {
      System.err.println("Usage: java [-classpath classpath] Googly <query>");
      System.exit(1);
    }
    // Create a new GoogleSearch object.
    GoogleSearch s = new GoogleSearch(  );
    try {

      s.setKey(googleKey);
      s.setQueryString(args[0]); // Google query from the command-line
      s.setMaxResults(10);
      // Query Google.
      GoogleSearchResult r = s.doSearch(  );
      // Gather the results.
      GoogleSearchResultElement[] re = r.getResultElements(  );
```

```
    // Output.
for ( int i = 0; i < re.length; i++ ) {
  // set the results as variables
  String title = re[i].getTitle(  );
  String URL = re[i].getURL(  );
  String snippet = re[i].getSnippet(  );

  // strip HTML from the results
  title = title.replaceAll("<[^>]+>", "");
  snippet = snippet.replaceAll("<[^>]+>", "");

  // print out the main bits of each result
  System.out.println(title);
  System.out.println(URL);
  System.out.println(snippet + "\\n");
}
```

```
    // Anything go wrong?
    } catch (GoogleSearchFault f) {
      System.out.println("GoogleSearchFault: " + f.toString(  ));
    }
  }

}
```

Compiling the Code

To successfully compile the Googly application, you need that*googleapi.jar* archive. I chose to keep it in path after `-classpath` accordingly:

```
% javac -classpath googleapi.jar Googly.java
```

This should leave you with a brand new*Googly.class* file, ready to run.

Running the Hack

Run Googly on the command line ["How to Run the Hacks" in the Preface], passing it your Google query

```
% java -classpath .:googleapi.jar Googly  "
                    query words
                "
```

and like so under Windows (notice the `;` replaces the `:` in the classpath):

```
java -classpath .;googleapi.jar Googly "
   query words
"
```

Here's a sample run:

```
% java -classpath .:googleapi.jar Googly "Learning Java"
Learning the Java Language
http://java.sun.com/docs/books/tutorial/java/index.html
Trail: Learning the Java Language. This trail covers the fundamentals of programming  ir

Learning the Java Language: Table of Contents
http://java.sun.com/docs/books/tutorial/java/TOC.html
Trail: Learning the Java Language: Table of Contents. Object-Oriented Programming  Conce

Java Programming FAQs and Tutorials: Learning Java
http://www.apl.jhu.edu/~hall/java/FAQs-and-Tutorials.html
A collection of FAQs and Tutorials for learning Java Programming.
...
```

PREV

Hack 95. Program Google in Python

Programming the Google Web API with Python is simple and clean, as these scripts and inter examples demonstrate.

Programming the Google Web API with Python is a piece of cake, thanks to MarkPilgrim's PyGoogle wrap module (http://pygoogle.sourceforge.net/)now maintained by Brian Landers. PyGoogle abstracts away r the underlying SOAP, XML, and request/response layers, leaving you free to spend time with the data its

PyGoogle Installation

Download a copy of PyGoogle (http://sourceforge.net/project/showfiles.php?group_id=99616) and follo installation instructions (http://pygoogle.sourceforge.net/dist/readme.txt). Assuming all goes to plan, d should be nothing more complex than this:

```
% python setup.py install
```

Alternatively, if you want to give this a whirl without installing PyGoogle or don't have permissions to ins globally on your system, simply put the included *SOAP.py* and *google.py* files into the same directory as *googly.py* script itself.

The Code

Save this code to a text file called *googly.py* , replacing *insert key here* with your Google developer's k

```
#!/usr/bin/python
# googly.py
# A typical Google Web API Python script using Mark Pilgrim's
# PyGoogle Google Web API wrapper
# [http://diveintomark.org/projects/pygoogle/].
# Usage: python googly.py <query>

import sys, string, codecs, re

# Use the PyGoogle module.
import google

# Grab the query from the command line
if sys.argv[1:]:
  query = sys.argv[1]
```

```
else:
  sys.exit('Usage: python googly.py <query>')

# Your Google API developer's key.
google.LICENSE_KEY = 'insert key here'

# Query Google.
data = google.doGoogleSearch(query)

# Teach standard output to deal with utf-8 encoding in the results.
sys.stdout = codecs.lookup('utf-8')[-1](sys.stdout)

# Output.
for result in data.results:
  # set the results as variables
  title = result.title
  URL = result.URL
  snippet = result.snippet

  # Strip HTML
  regex = re.compile('<[^>]+>')
  title = regex.sub(r'',title)
  snippet = regex.sub(r'',snippet)

  # print out the main bits of each result
  print string.join( (title, URL, snippet), "\\n"), "\\n"
```

Running the Hack

Invoke the script on the command line ["How to Run the Hacks" in the Preface] as follows:

```
% python googly.py  "
                    query words
                "
```

Here's a sample run that searches for `python` :

```
% python googly.py "python"
Python Programming Language
http://www.python.org/
Home page for Python, an interpreted, interactive, object-oriented, extensible
programming language. It provides an extraordinary combination of clarity and ...

Dive Into Python
http://diveintopython.org/
This book lives at . If you're reading it somewhere else, you may not have the latest ve
```

```
...
```

Hacking the Hack

Python has a marvelous interface for working interactively with the interpreter. It's a good place to expe
with modules such as PyGoogle, querying the Google API on the fly and digging through the data structu
returns.

Here's a sample interactive PyGoogle session demonstrating the use of the`doGoogleSearch` , `doGetCachec`
and `doSpellingSuggestion` functions:

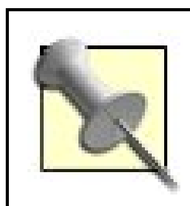
```
% python
ActivePython 2.4 Build 244 (ActiveState Corp.) based on
Python 2.4 (#60, Feb  9 2005, 19:03:27) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import google
>>> google.LICENSE_KEY = '
               insert key here
               '
>>> data = google.doGoogleSearch("Python")
>>> dir(data.meta)
['__doc__', '__init__', '_  _module_  _', 'directoryCategories', 'documentFiltering'
, 'endIndex', 'estimateIsExact', 'estimatedTotalResultsCount', 'searchComments',
 'searchQuery', 'searchTime', 'searchTips', 'startIndex']
>>> data.meta.estimatedTotalResultsCount
18800000
>>> data.meta.directoryCategories
[<SOAPpy.Types.structType item at 13216224>: {'fullViewableName': 'Top/Computers
/Programming/Languages/Python', 'specialEncoding': ''}]
>>> dir(data.results[2])
['URL', '__doc__', '__init__', '_  _module_  _', 'cachedSize', 'directoryCategory',
'directoryTitle', 'hostName', 'relatedInformationPresent', 'snippet', 'summary',
 'title']
>>> data.results[1].title
'Dive Into <b>Python</b>'
>>> data.results[1].URL
'http://diveintopython.org/'
>>> google.doGetCachedPage(data.results[1].URL)
'<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">\\n
<BASE HREF="http://diveintopython.org/"><table border=1
...
>>> google.doSpellingSuggestion('piethon')
'python'
```

Hack 96. Program Google in C# and .NET

Create GUI and console Google search applications with C# and the .NET framework.

The Google Web APIs Developer's Kit includes a sample C# Visual Studio .NET (http://msdn.microsoft.cc the *Form1.cs* code.

This hack provides basic code for a simple console Google search application similar in function (and, in



Compiling and running this hack requires that you have the .NET Framework (h

The Code

Type this code and save it to a text file called *googly.cs* , replacing *insert key here* with your Google de

```
// googly.cs
// A Google Web API C# console application.
// Usage: googly.exe <query>
// Copyright (c) 2002, Chris Sells.
// No warranties extended. Use at your own risk.
using System;
class Googly {
  static void Main(string[] args) {
    // Your Google API developer's key.
    string googleKey = "insert key here";
    // Take the query from the command line.
    if( args.Length != 1 ) {
      Console.WriteLine("Usage: google.exe <query>");
      return;
    }
    string query = args[0];
    // Create a Google SOAP client proxy, generated by:
    // c:> wsdl.exe http://api.google.com/GoogleSearch.wsdl
    GoogleSearchService googleSearch = new GoogleSearchService(  );
    // Query Google.
    GoogleSearchResult results = googleSearch.doGoogleSearch(googleKey,
query, 0, 10, false, "", false, "", "latin1", "latin1");
    // No results?
    if( results.resultElements == null ) return;
```

```
    // Loop through results.
    foreach( ResultElement result in results.resultElements ) {
      Console.WriteLine(  );
      Console.WriteLine(result.title);
      Console.WriteLine(result.URL);
      Console.WriteLine(result.snippet);
      Console.WriteLine(  );
    }
  }
}
```

Compiling the Code

Before compiling the C# code itself, you must create a Google SOAP client proxy. The proxy is a wodge
and return values. Fortunately, you don't have to do this by hand; the .NET Framework kit includes an a

This is a remarkable bit of magic if you think about it: the lion's share of code t

Call *wsdl.exe* with the location of your *GoogleSearch.wsdl* file, like so:

```
C:\\GOOGLY.NET>wsdl.exe GoogleSearch.wsdl
```

If you don't have the WSDL file handy, don't fret. You can point*wsdl.exe* at its location on Google's web

```
C:\\GOOGLY.NET\\CS>wsdl.exe http://api.google.com/GoogleSearch.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'C:\\GOOGLY.NET\\CS\\GoogleSearchService.cs'.
```

The end result is a *GoogleSearchService.cs* file that looks something like this:

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.42
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
```

```
using System.ComponentModel;
using System.Diagnostics;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Serialization;

//
// This source code was auto-generated by wsdl, Version=2.0.50727.42.
//
...
    public System.IAsyncResult BegindoGoogleSearch(string key, string q, int start, int
        return this.BeginInvoke("doGoogleSearch", new object[] {
                    key,
                    q,
                    start,
                    maxResults,
                    filter,
                    restrict,
                    safeSearch,
                    lr,
                    ie,
                    oe}, callback, asyncState);
    }
...
```

Now on to *googly.cs* itself:

```
C:\\GOOGLY.NET\\CS>csc /out:googly.exe *.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.
```

Running the Hack

Run Googly on the command line ["How to Run the Hacks" in the Preface], passing it your Google query

```
C:\\GOOGLY.NET\\CS>googly.exe  "
                    query words
                "
```

The DOS command window isn't the best when it comes to displaying and allow

Here's a sample run:

```
% googly.exe  "
                    WSDL while you work
                 "

Using the SOAPscope Workspace
http://www.mindreef.com/support/soapscope/4.1/help/workspace.html
<b>You</b> can invoke from <b>WSDL</b> or resend messages, and <b>work</b> with the resu


<b>Working</b> with <b>WSDL</b>
http://www.mindreef.com/support/soapscope/4.1/help/serviceview.html
This allows <b>you</b> to perform <b>WSDL</b> analysis on <b>work</b> in progress to det
...
```

Chris Sells and Rael Dornfest

Hack 97. Program Google in VB.NET

Create GUI and console Google search applications with Visual Basic and the .NET framework

Along with the functionally identical C# version [Hack #96], the Google Web APIs Developer's Kit (in the a simple console Google search application without the possible opacity of a full-blown Visual Studio .NE

Compiling and running this hack requires that you have the .NET Framework (

The Code

Save the following code to a text file called *googly.vb* , replacing *insert key here* with your Google deve

```
' googly.vb
' A Google Web API VB.NET console application.
' Usage: googly.exe <query>
' Copyright (c) 2002, Chris Sells.
' No warranties extended. Use at your own risk.
Imports System
Module Googly
  Sub Main(ByVal args As String(  ))
    ' Your Google API developer's key.
    Dim googleKey As String = "insert key here"
    ' Take the query from the command line.
    If args.Length <> 1 Then
      Console.WriteLine("Usage: google.exe <query>")
      Return
    End If
    Dim query As String = args(0)
    ' Create a Google SOAP client proxy, generated by:
    ' c:> wsdl.exe /l:vb http://api.google.com/GoogleSearch.wsdl
    Dim googleSearch As GoogleSearchService = New GoogleSearchService(  )
    ' Query Google.
    Dim results As GoogleSearchResult = googleSearch.
doGoogleSearch(googleKey, query, 0, 10, False, "", False, "", "latin1",
"latin1")
    ' No results?
    If results.resultElements Is Nothing Then Return
    ' Loop through results.
```

```
    Dim result As ResultElement
    For Each result In results.resultElements
      Console.WriteLine(  )
      Console.WriteLine(result.title)
      Console.WriteLine(result.URL)
      Console.WriteLine(result.snippet)
      Console.WriteLine(  )
    Next
  End Sub
End Module
```

Compiling the Code

Not surprisingly, compiling the code for the VB and .NET application is similar to compiling the code in C

Before compiling the VB application code itself, you must create a Google SOAP client proxy. The proxy Fortunately, you don't have to do this by hand; the .NET Framework kit includes an application,*wsdl.exe*

Call *wsdl.exe* with the location of your *GoogleSearch.wsdl* file and specify that you'd like VB proxy code:

```
C:\\GOOGLY.NET\\VB>wsdl.exe /l:vb GoogleSearch.wsdl
```

If you don't have the WSDL file handy, don't fret. You can point*wsdl.exe* at its location on Google's web

```
C:\\GOOGLY.NET\\VB>wsdl.exe /l:vb http://api.google.com/GoogleSearch.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'C:\\GOOGLY.NET\\VB\\GoogleSearchService.vb'.
```

What you get is a *GoogleSearchService.vb* file with all that underlying Google SOAP-handling ready to g

```
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:2.0.50727.42
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict Off
Option Explicit On

Imports System
Imports System.ComponentModel
```

```
Imports System.Diagnostics
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Xml.Serialization
...
    Public Function BegindoGoogleSearch(ByVal key As String, ByVal q As String, ByVal st
        Return Me.BeginInvoke("doGoogleSearch", New Object(  ) {key, q, start, maxResult
    End Function

    '''<remarks/>
    Public Function EnddoGoogleSearch(ByVal asyncResult As System.IAsyncResult) As Googl
        Dim results(  ) As Object = Me.EndInvoke(asyncResult)
        Return CType(results(0),GoogleSearchResult)
    End Function
...
```

Now, to compile that *googly.vb* :

```
C:\\GOOGLY.NET\\VB>vbc /out:googly.exe *.vb
Microsoft (R) Visual Basic Compiler version 8.0.50727.42
for Microsoft (R) .NET Framework version 2.0.50727.42
Copyright (c) Microsoft Corporation.  All rights reserved.
```

Running the Hack

Run Googly on the command line ["How to Run the Hacks" in the Preface], passing it your Google query

```
C:\\GOOGLY.NET\\VB>googly.exe "
                    query words
                "
```

The DOS command window isn't the best when it comes to displaying and allo

Functionally identical to its C# counterpart [Hack #96], the VB hack should turn up the same resultsGo

Chris Sells and Rael Dornfest

Hack 98. Program Google with ColdFusion

ColdFusion MX includes all the tools necessary to work with the Google API.

ColdFusion is a development platform for creating web applications. Its tag-based template structure is a popular choice for HTML developers who want to move to more dynamic content and are already familiar with using tags. In fact, if you were to glance at a ColdFusion script, you might think the cod was standard HTML. Putting together a ColdFusion template is a lot like putting together an HTML page, but you can draw on resources such as databases and web services to bring in dynamic content.

You need a version of ColdFusion MX or later to use this hack, because it relies on the `<cfinvoke>` tag that was added with the MX release. This tag takes the heavy lifting out of consuming web services.

The Code

This hack shows how you can quickly use the ColdFusion Markup Language (CFML) to bring in Google content. The script assembles the proper SOAP request based on a querystring variable and gets a response from Google with the `<cfinvoke>` tag. Then the `<cfloop>` tag goes through each bit of data, adding it to the page.

Add the following code to a file called *googly.cfm* and upload it to your server:

```
<!---
googly.cfm
Accepts a search term and shows the top results.
Usage: googly.cfm?q=<Query>
--->
<html>
<body>
<h2>Google Search Results</h2>
<ol>
<!--- Set your unique Google Key --->
<cfset google_key = "insert your key">

<!--- Grab the incoming search query --->
<cfset query = "#URL.q#">

<!--- Construct a Google API request with required options --->
<cfinvoke
  webservice ="http://api.google.com/GoogleSearch.wsdl"
  method ="doGoogleSearch"
```

```
  returnVariable = "results" >
    <cfinvokeargument name="key" value="#google_key#">
    <cfinvokeargument name="q" value="#query#">
    <cfinvokeargument name="start" value="0">
    <cfinvokeargument name="maxResults" value="10">
    <cfinvokeargument name="filter" value="False">
    <cfinvokeargument name="restrict" value="">
    <cfinvokeargument name="safeSearch" value="False">
    <cfinvokeargument name="lr" value="">
    <cfinvokeargument name="ie" value="">
    <cfinvokeargument name="oe" value="">
</cfinvoke>

<!--- Loop Through Response --->
<cfoutput>
<cfloop from="1" to="#ArrayLen(results.ResultElements)#" index="i">
    <cfset title = "#results.ResultElements[i].title#">
    <cfset aURL = "#results.ResultElements[i].URL#">
    <cfset snippet = "#results.ResultElements[i].snippet#">

    <li><div style="margin-bottom:15px;">
    <a href=\\"#aURL#\\">#Replace(title,"<br>","")#</a>
    #Replace(snippet,"<br>","")#<br />
    <cite>#aURL#</cite></div></li>
</cfloop>
</cfoutput>

</ol>
</body>
</html>
```

Take a look at the `<cfinvoke>` tag in the script. Note that the `<cfinvokeargument>` tag sets all the required parameters for the `doGoogleSearch` method. The `<cfloop>` section at the bottom of the script goes through each search result, printing out the title, URL, and snippet to the page.

Running the Hack

Bring up the page in a browser to see it in action:

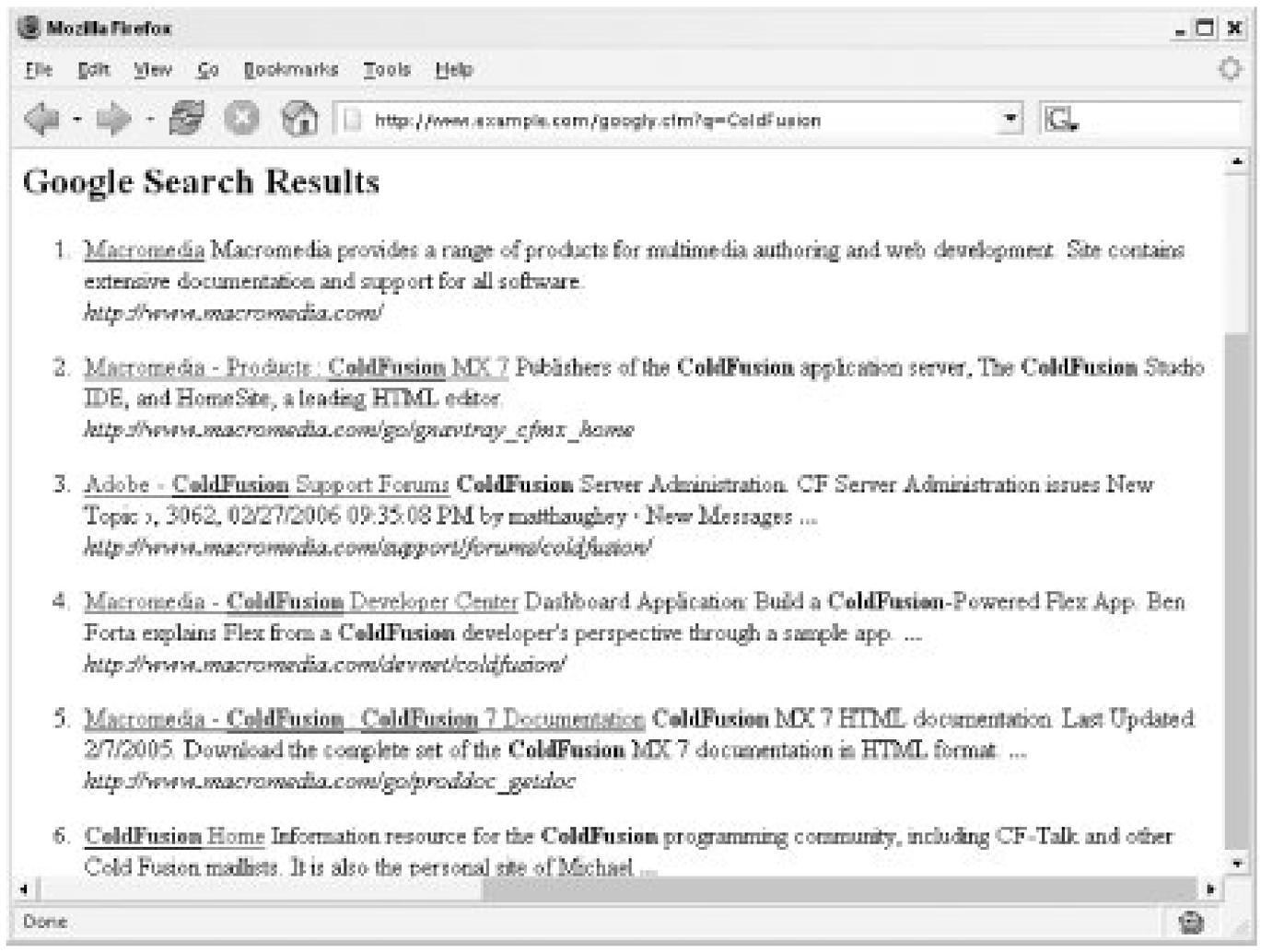
`http://example.com/googly.cfm?q=insert word`

Separate multiple words with URL-encoded spaces, as in this search for `"ColdFusion MX"`:

`http://example.com/googly.cfm?p=ColdFusion%20MX`

Figure 8-1 shows the top Google search results for `ColdFusion`.

Figure 8-1. Google search results for "ColdFusion"



As you can see, ColdFusion MX includes all the tools you need to make Google API requests and work with the responses, and integrating Google data with existing ColdFusion applications can be accomplished with just a few lines of code.

PREV

Hack 99. Program Google with PHP 5

Take advantage of some of the latest features in PHP to quickly add Google data to PHP-powered pages.

The recursively named PHP Hypertext Processing language (PHP) is a popular choice for building dynamic web applications. The PHP platform is continually evolving, and the latest version (5) includes a built-in SOAP extension (http://www.php.net/soap) you can use to consume services such as the Google API. This hack shows how easy it is to request and parse web services data with PHP.

If you're running PHP on Windows, you might need to enable the SOAP extension with a quick edit to your *php.ini* file, which is usually located in your *c:\\windows* directory. Open the file and add the following line:

```
extension=php_soap.dll
```

Once you've made the change, restart your web server; your PHP scripts should have access to the SOAP extension.

The Code

Save the following code to your web server in a file called *googly.php*, and be sure to replace *insert key here* with your own Google API key:

```
<?php
// googly.php
// Accepts a search term and shows the top results.
// Usage: googly.php?q=<Query>

// Your Google API developer's key
$google_key = "insert key here";

// Grab the incoming search query
$query = $_GET['q'];

if ($query == "") {
    print "usage: googly.php?q=&lt;Query&gt;";
    die;
}

// Initiate a SOAP client
```

```
$client = new SoapClient("GoogleSearch.wsdl");

// Construct a Google SOAP request
$results = $client->doGoogleSearch(
   $google_key, $query, 0, 10, "false", "",  "false",
   "", "latin1", "latin1"
);
?>
<html>
<body>
<h2>Google Search Results</h2>
<ol>
<?php
// Loop through the results returned, printing them out
foreach($results->resultElements as $result) {
    $title = $result->title;
    $url = $result->URL;
    $snippet = $result->snippet;
    print "<li><div style=\\"margin-bottom:15px;\\">";
    print "<a href=\\"$url\\">$title</a><br />";
    print "$snippet<br />";
    print "<cite>$url</cite></div></li>\\n";
}
?>
</ol>
</body>
</html>
```

This script uses the value of the querystring variable `q` to build a Google API SOAP request and loop through the results.

Keep in mind that the `new SoapClient` method needs to know where the Google WSDL file is located. You can either place it in the same directory as your script or include a full path to the file in the method's argument. And, of course, you can pick up your own copy of the WSDL file in the Google API developer's kit (http://www.google.com/apis/).

Running the Hack

To run the script, point your web browser to the location of the script on your server and add the querystring variable `q`:

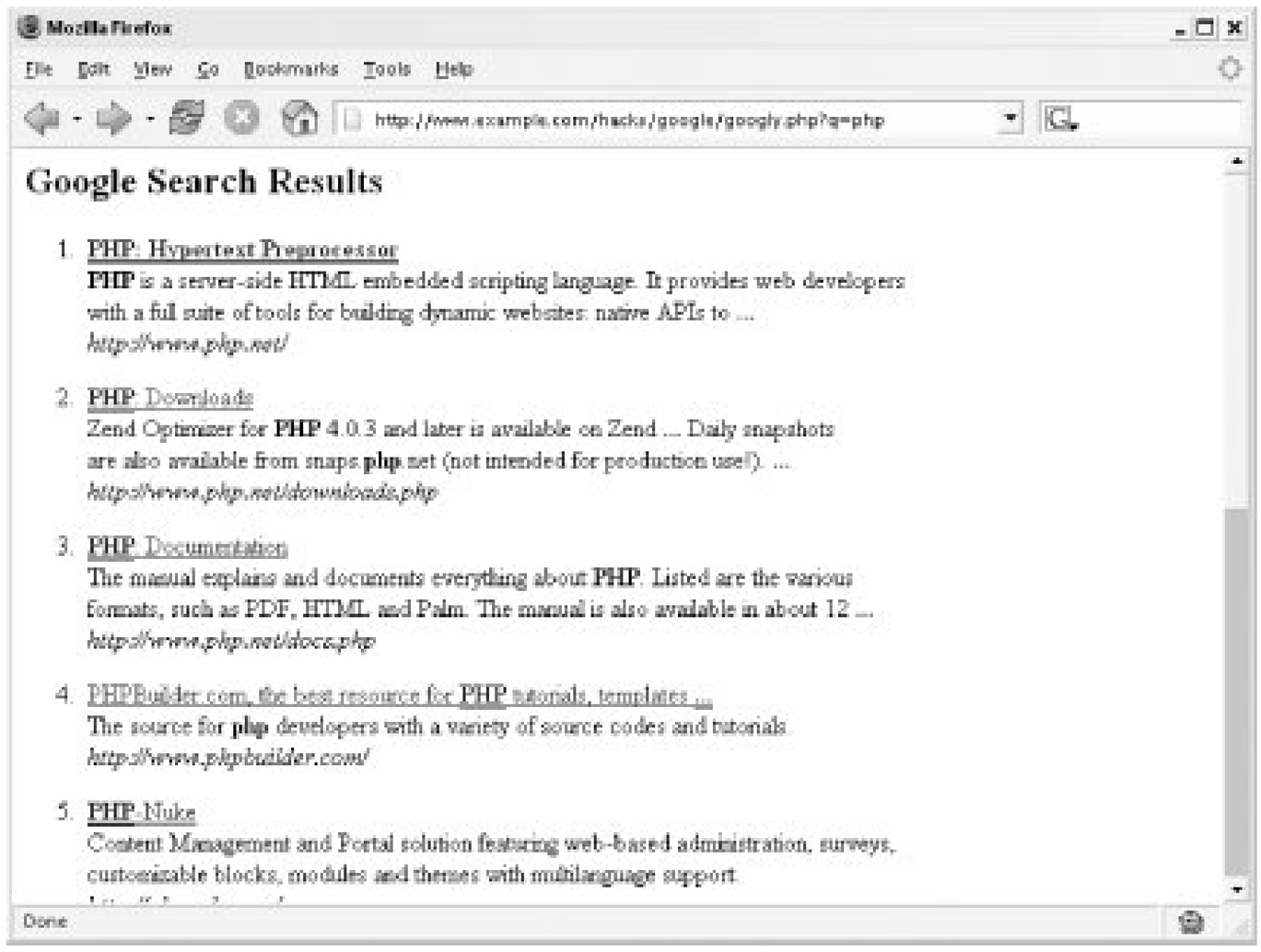
```
http://example.com/googly.php?q=insert word
```

You can add multiple words by encoding spaces for URLs. For example, here's the search string for `"PHP encoding"`:

```
http://example.com/googly.php?q=PHP%20encoding
```

Figure 8-2 shows the results of a search for PHP.

Figure 8-2. Google Search results for "PHP"



With PHP's built-in SOAP extension, working with Google API data is much more intuitive than with earlier versions of PHP.

Hack 100. Program Google with VBScript

Build Google searches into Windows programs or ASP pages with VBScript.

VBScript is a general-purpose scripting language for Windows that gets its name from its big brother of here and there, the code in this hack can add Google searching to Office applications or an ASP-powered Windows Script, and it provides just the basics for requesting and presenting Google results.

Microsoft Windows Script is built into the fabric of the Windows operating system and is used primarily b withyou guessed itsystem administration. But Microsoft Windows Scripts can also be used to automate a programs.

What You Need

If your Windows installation is up to date, you shouldn't need to install anything extra to run this hack. I you might want to grab the latest version of Microsoft Windows Script at http://www.microsoft.com/scri Microsoft Windows Script 5.6 or later for your version of Windows.

This hack also relies on the Microsoft SOAP object to communicate with Google. Your system should alre into trouble, you can always download the latest version at http://msdn.microsoft.com/webservices. The the latest version from the Downloads section and install it.

The Code

Like any other script, the code is simply plain text in a standard text file. You can even use Notepad to a

```
' googly.vbs
' Accepts a search term and shows the top results.
' Usage: cscript googly.vbs <Query> //I

'Set your unique Google Key
Const GOOGLE_KEY = "insert your key"

'Grab the incoming search query or ask for one
If WScript.Arguments.Length = 0 Then
    strQuery = InputBox("Enter a search Term")
Else
    strQuery = WScript.Arguments(0)
End If

'Initialize the SOAP object
Set SoapClient = WScript.CreateObject("MSSOAP.SoapClient")
```

```
SoapClient.mssoapinit "GoogleSearch.wsdl"

'Send the request
Set Results = SoapClient.doGoogleSearch(GOOGLE_KEY, strQuery, 0, 10, False, "", False, '

'Loop through the entire SOAP response
For Each Result In Results

  'Find the resulst node
  If Result.nodeName = "resultElements" Then

    'Go through every child node
    For i = 0 To Result.childNodes.length - 1

      'If the current node is an item, proceed
      If Result.childNodes.Item(i).nodeName = "item" Then

        'Go through each result detail and set variables
        Set rsDetail = Result.childNodes.Item(i)
        For j = 0 To rsDetail.childNodes.length - 1
          Select Case rsDetail.childNodes.Item(j).nodeName
            Case "title"
              strTitle = rsDetail.childNodes.Item(j).text
            Case "URL"
              strURL = rsDetail.childNodes.Item(j).text
            Case "snippet"
              strSnippet = rsDetail.childNodes.Item(j).text
          End Select
        Next

        'The variables are set, so print them out
        WScript.Echo stripHTML(strTitle) & VbCr
        WScript.Echo strURL & VbCr
        WScript.Echo stripHTML(strSnippet) & VbCr & vbCrLf

        'Destroy the detail object
        Set rsDetail = Nothing
      End If

      'Reset the variables until the next round
      strTitle = ""
      strURL = ""
      strSnippet = ""
    Next
  End If
Next

'Unload the SOAP object
Set SoapClient = Nothing

'A function to strip HTML
Function stripHTML(str)
```

```
    Set objRegExpr = New regexp
    objRegExpr.Pattern = "<[^>]+>"
    objRegExpr.Global = True
    objRegExpr.IgnoreCase = True
    stripHTML = objRegExpr.Replace(str,"")
    Set objRegExpr = Nothing
End Function
```

This code accepts a query on the command line when the script runs, or it asks the user for a query witl the Google API using the SOAP object. The code then picks through the SOAP response to find the prope user.

Note that the last bit of code is a function that strips HTML tags from the responsehandy when you're si a web page.

Running the Hack

There are a couple different ways to run the code. The most useful way to see the results is to run the s command prompt and type the following command:

```
cscript googly.vbs
                  insert word
                //I
```

Be sure to include multiword searches in quotes:

```
cscript googly.vbs "
                  insert multiword phrase
              " //I
```

The `//I` switch tells Microsoft Windows Script to output anything to the command line. Here's a look at t

```
% cscript googly.vbs vbscript //I
Scripting
http://msdn.microsoft.com/scripting/
VBScript &middot; JScript &middot; Script Runtime &middot; Windows Script Host &
middot; WMI &middot; ADSI. Sample Code.  Script Repository &middot; Misc. Code Samples.

VBScript Tutorial
http://www.w3schools.com/vbscript/default.asp
Free HTML, XHTML, CSS, JavaScript, DHTML, XML, DOM, XSL, XSLT, RSS, ASP, ADO,  PHP, SQL
...
```

If you want to, though, you can just double-click the*googly.vbs* file as you would any other program. Yo shown one at a timeall 10 of themin window prompts such as the one in Figure 8-3.

Figure 8-3. A Google result for "vbscript" in a wi

While this isn't the handiest way to view search results, it shows that adding Google search results to VE

PREV

PREV

Appendix 1. Track News About Google

Google is a big company, and there are many ways to watch Google in action. Whether you're a potential employee, investor, competitor, or just a fan of the site, you'll want to keep tabs on what Google is doing and where it might be headed. The news sources in this appendix should give you a starting point for watching the company, and you can add the RSS or Atom feeds for the sources directly to your favorite Google newsreader. Once you subscribe to a few Google-related feeds, you won't have any trouble keeping up with the latest news.

PREV

Google Sources

You can use Google to find information about anything in the world, even Google itself:

Google Blog

This is the official source for news and commentary straight from Google. Be sure to check the links listed under More Google Blogs for links to blogs dealing with specific Google products and locales.

Web

http://googleblog.blogspot.com/

Feed

http://googleblog.blogspot.com/atom.xml

Google News Search

Use the power of Google News to track stories about Google from thousands of news sources around the world.

Web

http://news.google.com/news?q=Google

Feed

http://news.google.com/news?q=Google&output=atom

Google Finance

Track the progress of Google's stock and find financial news and analysis.

Stock Performance

http://finance.google.com/finance?q=GOOG

Financial News

http://finance.google.com/finance?cid=694653&morenews=10

Financial Discussions

http://finance.google.com/group/google.finance.694653

Google Jobs

Find out what positions are open at Google.

Web

http://www.google.com/jobs/

Google Labs

Google Labs is where Google unveils its newest technology, experiments, and bits of code that might not be ready for the main Google site. Browse around the lab for a while, and you'll get a sense of where Google is headed.

Web

http://labs.google.com/

Google Code

Google Code is a site devoted to all the ways you can integrate Google into your own applications. Built for developers, it features the latest news about coding for Google and offers links to Google APIs and open source projects.

Web

http://code.google.com/

Featured Products Feed

http://code.google.com/feeds/featured.xml

Google Code Blog Feed

http://code.google.com/feeds/updates.xml

Google Press Center

The Press Center is an official spot for news that Google wants to release to the public. It's very stuffy compared with the Google Blog, but it's a valuable source of official press releases and company financial statements.

Web

http://www.google.com/press/

Press Release Feed

http://googlepress.blogspot.com/atom.xml

Google Products

This page lists most of the products and services that Google offers in one place. Visit periodically to find new product offerings, or to quickly find a link to an old favorite.

Web

http://www.google.com/options/

Outside News Sources

There are many outside news sources that are dedicated to Google and the world of search engines. Here are a few news sources that frequently have information about Google:

Google Blogoscoped

Philipp Lenssen is an independent blogger in Stuttgart, Germany who covers every move that Google makes.

Web

http://blog.outer-court.com/

Feed

http://blog.outer-court.com/rss.xml

Search Engine Watch Blog

This is a Jupitermedia blog that follows all major search engines, including Google.

Web

http://blog.searchenginewatch.com/blog/

Feed

http://feeds.searchenginewatch.com/sewblog

John Battelle's Searchblog

John wrote a book about Google called *The Search*, and he writes about the search engine industry at his blog.

Web

http://battellemedia.com/

Feed

http://feeds.feedburner.com/JohnBattellesSearchblog

ResearchBuzz

Tara Calishain (coauthor of this book) covers all aspects of Internet research, and Google as it relates to research.

Web

http://researchbuzz.com/

Feed

http://www.researchbuzz.org/researchbuzz.rss

Slashdot Topic: Google

The original "news for nerds" has a category devoted toGoogle.

Web

http://slashdot.org/search.pl?tid=217

Slashdot doesn't offer news feeds for individual topics, but if you want to build your own feed of Google News, you can always build your own feed by screen-scraping the site. A post called Slashdot Topic Feeds (http://www.onfocus.com/2006/03/3789) at Paul Bausch's blog shows how to scrape together a Slashdot feed.

The Unofficial Google Weblog

As the name implies, this a blog devoted to Google. It's run by Google, Inc./AOL and often picks up some obscure bits of news that you might not see otherwise.

Web

http://google.weblogsinc.com/

Feed

http://google.weblogsinc.com/rss.xml

Simply Google

While not technically a news source, this reimagination of theGoogle home page is a convenient one-stop page for most of the search features that Google offers. In addition to dozens of search forms, there are links to Google sites, software, blogs, and independent sources of Google information.

Web

http://www.usabilityviews.com/simply_google.htm

PREV

Google Employee Blogs

Google has over 5,000 employees, so it makes sense that some keep blogs. Mostemployee blogs aren't connected with the company in any way, and you might not learn much about Google from them. But if you tune into a few for a while, you'll definitely see the company in a new way. Here are a few of Google's employees who have a blog:

Matt Cutts

Matt Cutts's blog Gadgets, Google, and SEO covers Google's battle with search engine spam and life inside Google. If you want an inside take on Google, this should be your first stop.

Web

http://www.mattcutts.com/blog/

Feed

http://www.mattcutts.com/blog/feed/atom/

Chris Wetherell

Chris is a software engineer at Google by day, and a musician with the indie band Dealership by night.

Web

http://www.massless.org/

Feed

http://massless.org/atom.xml

Jason Shellen

Jason is the Blogger Program Manager at Google, and, as you'd expect, he has a blog.

Web

http://www.shellen.com/

Feed

http://feeds.feedburner.com/shellendotcom

Xooglers

Xooglers is a group blog by ex-employees of Google. They give their take on the search industry and Google from their unique perspective.

Web

http://xooglers.blogspot.com/

Feed

http://xooglers.blogspot.com/atom.xml

You can probably spot many more employee blogs by browsing to Google and typing in the phrase`"I work (at | for) Google" weblog`.

Grassroots Sources

Though these sources don't pass through an editor, and the content isn't produced by professionals, they can provide a unique perspective on the company, point to interesting personal opinions, or refer you to obscure bits of information about Google.

Flickr Photos

Public photos tagged with `Google` at Flickr are often photos taken by employees, pictures from Google campuses, pictures from Google events, or screenshots of Google software.

Web

http://www.flickr.com/photos/tags/google

Feed

http://www.flickr.com/services/feeds/photos_public.gne?tags=google&format=rss_200

del.icio.us

Hundreds of people swap links on the social bookmarks servicedel.icio.us, and every day several dozen links tagged with `Google` are added. The "Popular Google URLs" list shows which Google-related links are bookmarked most often.

Web

http://del.icio.us/tag/google

Feed

http://del.icio.us/rss/tag/google

Popular Google URLs

http://del.icio.us/popular/google

Popular Google URLs Feed

http://del.icio.us/rss/popular/google

Technorati

Use Technorati to find blog posts that mention Google. Its Google tag page also pulls in conten from Flickr, del.icio.us, and Furl.

Web

http://www.technorati.com/tag/Google

Feed

http://feeds.technorati.com/feed/posts/tag/Google

Google-related Google Groups

Try a search for Google at Google Groups to find some official and unofficial gathering places tc discuss Google. Also try plugging in specific Google product names to find groups devoted to discussing certain aspects of Google.

Web

http://groups.google.com/groups/dir?lnk=srgmt&q=Google

It would be impossible to track everything Google is doing on a daily basis, but subscribing to a combination of these sources should help keep you in the Google loop.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The tool on the cover of Google Hacks, Third Edition, is a a pair of locking pliers. Locking pliers are very versatile tools. They can be used for turning, twisting, cutting wire, tightening screws and bolts, and clamping. Locking pliers are specially designed to put pressure on a bolt or nut in such a way that the user can approach the nut or bolt from any angle. A simple squeeze can put up to a ton of pressure between the pliers' jaws, enabling them to lock onto even oddly shaped pieces. Locking pliers include a guarded release, which prevents accidental release or pinching, and a trigger, which unlocks the pliers.

The cover image is an original photograph by Edie Freedman. The cover font is Adobe ITC Garamond The text font is Linotype Birka; the heading font is Adobe Helvetica Neue Condensed; and the code font is LucasFont's TheSans Mono Condensed.

PREV

PREV

Index

PREV

PREV

Index

[**SYMBOL**] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z]

PREV

Index

automated queries

PREV

Index

[SYMBOL] [A] [**B**] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z]

PREV

Index

PREV

Index

Index

PREV

Index

Index

Index

Index

Index

Index

Index

PREV

Index

Index

Index

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z]

Index

Index

Index

Index

Index

Index

PREV

Index

WMS (Web Mapping Service), data and map imagery available from WSDL

PREV

Index

Index

Index

PREV