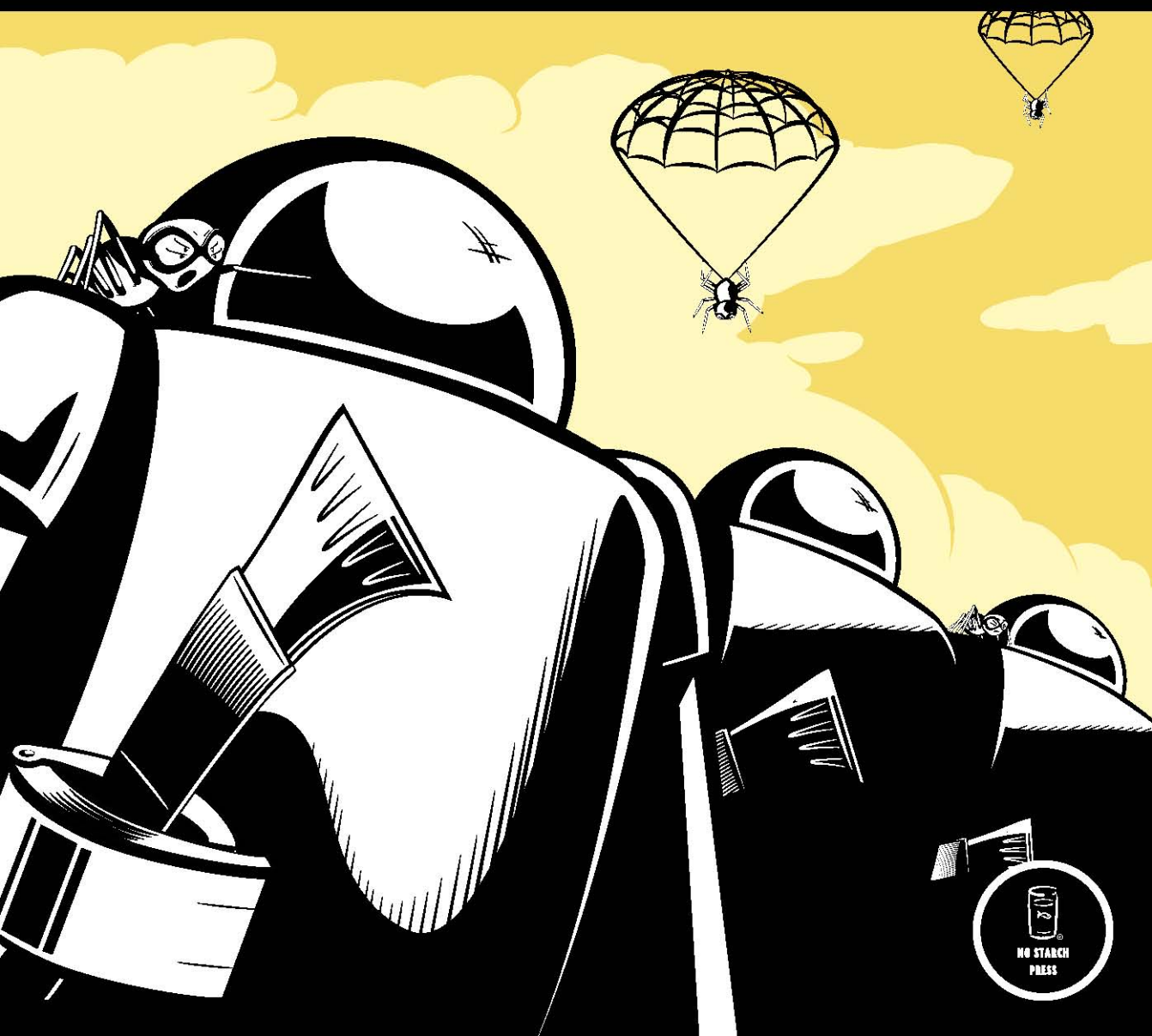
WEBBOTS, SPIDERS, AND SCREEN SCRAPERS

A GUIDE TO DEVELOPING INTERNET AGENTS WITH PHP/CURL

MICHAEL SCHRENK



NO STARCH PRESS

2

IDEAS FOR WEBBOT PROJECTS

It's often more difficult to find applications for new technology than it is to learn the technology itself. Therefore, this chapter focuses on encouraging you to generate ideas for things that you can do with webbots. We'll explore how webbots capitalize on browser limitations, and we'll see a few examples of what people are currently doing with webbots. We'll wrap up by throwing out some wild ideas that might help you expand your expectations of what can be done online.

Inspiration from Browser Limitations

A useful method for generating ideas for webbot projects is to study what *cannot* be done by simply pointing a browser at a typical website. You know that browsers, used in traditional ways, cannot automate your Internet experience. For example, they have these limitations:

- Browsers cannot aggregate and filter information for relevance
- Browsers cannot interpret what they find online
- Browsers cannot act on your behalf

However, a browser may leverage the power of a webbot to do many things that it could not do alone. Let's look at some real-life examples of how browser limitations were leveraged into actual webbot projects.

Webbots That Aggregate and Filter Information for Relevance

TrackRates.com (http://www.trackrates.com, shown in Figure 2-1) is a website that deploys an army of webbots to aggregate and filter hotel room prices from travel websites. By identifying room prices for specific hotels for specific dates, it determines the actual market value for rooms up to three months into the future. This information helps hotel managers intelligently price rooms by specifically knowing what the competition is charging for similar rooms. TrackRates.com also reveals market trends by performing statistical analysis on room prices, and it tries to determine periods of high demand by indicating dates on which hotels have booked all of their rooms.

Figure 2-1: TrackRates.com

I wrote TrackRates.com to help hotel managers analyze local markets and provide facts for setting room prices. Without the TrackRates.com webbot, hotel managers either need to guess what their rooms are worth, rely on less current information about their local hotel market, or go through the arduous task of manually collecting this data.

Webbots That Interpret What They Find Online

WebSiteOptimization.com (http://www.websiteoptimization.com) uses a webbot to help web developers create websites that use resources effectively. This webbot accepts a web page's URL (as shown in Figure 2-2) and analyzes how each graphic, CSS, and JavaScript file is used by the web page. In the interest of full disclosure, I should mention that I wrote the back end for this web page analyzer.

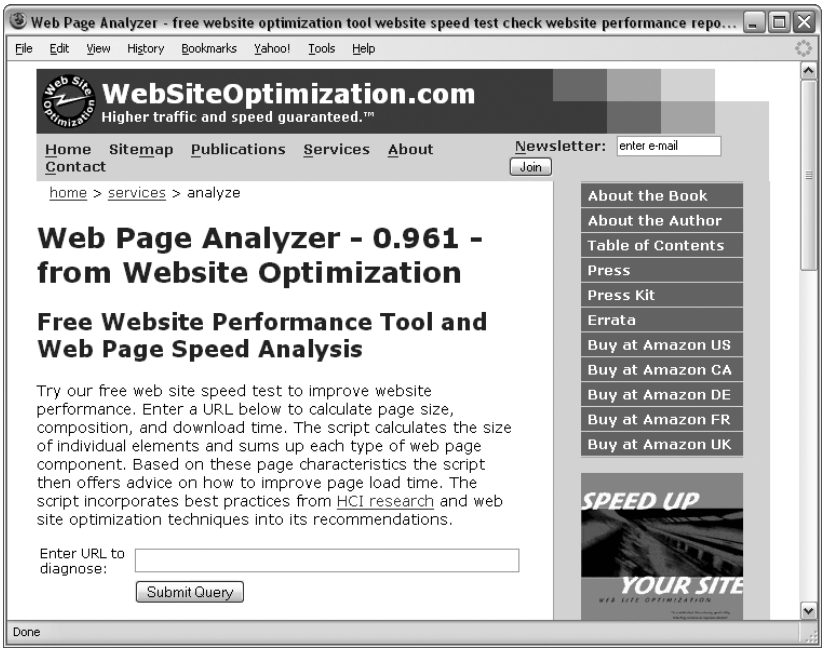


Figure 2-2: A website-analyzing webbot

The WebSiteOptimization.com webbot analyzes the data it collects and offers suggestions for optimizing website performance. Without this tool, developers would have to manually parse through their HTML code to determine which files are required by web pages, how much bandwidth they are using, and how the organization of the web page affects its performance.

Webbots That Act on Your Behalf

Pokerbots, webbots that play online poker, are a response to the recent growth in online gambling sites, particularly gaming sites with live poker rooms. While the action in these pokers sites is live, not all the players are. Some online poker players are webbots, like Poker Robot, shown in Figure 2-3.

Webbots designed to play online poker not only know the rules of Texas hold 'em but use predetermined business rules to expertly read how others play. They use this information to hold, fold, or bet appropriately. Reportedly, these automated players can very effectively pick the pockets of new and inexperienced poker players. Some *collusion webbots* even allow one virtual

player to play multiple hands at the same table, while making it look like a separate person is playing each hand. Imagine playing against a group of people who not only know each other's cards, but hold, fold, and bet against you as a team!

Obviously, such webbots that play expert poker (and cheat) provide a tremendous advantage. Nobody knows exactly how prevalent pokerbots are, but they have created a market for anti-pokerbot software like Poker BodyGuard, distributed by StopPokerCheaters.com.

Figure 2-3: An example pokerbot

A Few Crazy Ideas to Get You Started

One of the goals of this book is to encourage you to write new and experimental webbots of your own design. A way to jumpstart this process is to brainstorm and generate some ideas for potential projects. I've taken this opportunity to list a few ideas to get you started. These ideas are not here necessarily because they have commercial value. Instead, they should act as inspiration for your own webbots and what you want to accomplish online.

When designing a webbot, remember that the more specifically you can define the task, the more useful your webbot will be. What can you do with a webbot? Let's look at a few scenarios.

Help Out a Busy Executive

Suppose you're a busy executive type and you like to start your day reading your online industry publication. Time is limited, however, and you only let yourself read industry news until you've finished your first cup of coffee. Therefore, you don't want to be bothered with stories that you've read before or that you know are not relevant to your business. You ask your developer to create a specialized webbot that consolidates articles from your favorite industry news sources and only displays links to stories that it has not shown you before.

The webbot could ignore articles that contain certain key phrases you previously entered in an *exclusion list*[1] and highlight articles that contain references to you or your competitors. With such an application, you could quickly scan what's happening in your industry and only spend time reading relevant articles. You might even have more time to enjoy your coffee.

Save Money by Automating Tasks

It's possible to design a webbot that automatically buys inventory for a store, given a predetermined set of buying criteria. For example, assume you own a store that sells used travel gear. Some of your sources for inventory are online auction websites.[2] Say you are interested in bidding on under-priced Tumi suitcases during the closing minute of their auctions. If you don't use a webbot of some sort, you will have to use a web browser to check each auction site periodically.

Without a webbot, it can be expensive to use the Internet in a business setting, because repetitive tasks (like procuring inventory) are time consuming without automation. Additionally, the more mundane the task, the greater the opportunity for human error. Checking online auctions for products to resell could easily consume one or two hours a day—up to 25 percent of a 40-hour work week. At that rate, someone with an annual salary of $80,000 would cost a company $20,000 a year to procure inventory (without a webbot). That cost does not include the cost of opportunities lost while the employee manually surfs auction sites. In scenarios like this, it's easy to see how product acquisition with a webbot saves a lot of money—even for a small business with small requirements. Additionally, a webbot may uncover bargains missed by someone manually searching the auction site.

Protect Intellectual Property

You can write a webbot to protect your online intellectual property. For example, suppose you spent many hours writing a JavaScript program. It has commercial value, and you license the script for others to use for a fee. You've been selling the program for a few months and have learned that some people

[1] An *exclusion list* is a list of keywords or phrases that are ignored by a webbot.

[2] Some online auctions actually provide tools to help you write webbots that manage auctions. If you're interested in automating online auctions, check out eBay's Developers Program (http://developer.ebay.com).

are downloading and using your program without paying for it. You write a webbot to find websites that are using your JavaScript program without your permission. This webbot searches the Internet and makes a list of URLs that reference your JavaScript file. In a separate step, the webbot does a *whois* lookup on the domain to determine the owner from the domain registrar.[3] If the domain is not one of your registered users, the webbot compiles contact information from the domain registrar so you can contact the parties who are using unlicensed copies of your code.

Monitor Opportunities

You can also write webbots that alert you when particular opportunities arise. For example, let's say that you have an interest in acquiring a Jack Russell Terrier.[4] Instead of devoting part of each day to searching for your new dog, you decide to write a webbot to search for you and notify you when it finds a dog meeting your requirements. Your webbot performs a daily search of the websites of local animal shelters and dog rescue organizations. It parses the contents of the sites, looking for your dog. When the webbot finds a Jack Russell Terrier, it sends you an email notification describing the dog and its location. The webbot also records this specific dog in its database, so it doesn't send additional notifications for the same dog in the future. This is a fairly common webbot task, which could be modified to automatically discover job listings, sports scores, or any other timely information.

Verify Access Rights on a Website

Webbots may prevent the potentially nightmarish situation that exists for any web developer who mistakenly gives one user access to another user's data. To avoid this situation, you could commission a webbot to verify that all users receive the correct access to your site. This webbot logs in to the site with every viable username and password. While acting on each user's behalf, the webbot accesses every available page and compares those pages to a list of appropriate pages for each user. If the webbot finds a user is inadvertently able to access something he or she shouldn't, that account is temporarily suspended until the problem is fixed. Every morning before you arrive at your office, the webbot emails a report of any irregularities it found the night before.

Create an Online Clipping Service

Suppose you're very vain, and you'd like a webbot to send an email to your mother every time a major news service mentions your name. However, since you're not vain enough to check all the main news websites on a regular basis, you write a webbot that accomplishes the task for you. This webbot accesses a collection of websites, including CNN, Forbes, and Fortune.

---

[3] *whois* is a service that returns information about the owner of a website. You can do the equivalent of a whois from a shell script or from an online service.

[4] I actually met my dog online.

You design your webbot to look only for articles that mention your name, and you employ an exclusion list to ignore all articles that contain words or phrases like *shakedown*, *corruption*, or *money laundering*. When the webbot finds an appropriate article, it automatically sends your mother an email with a link to the article. Your webbot also blind copies you on all emails it sends so you know what she's talking about when she calls.

Plot Unauthorized Wi-Fi Networks

You could write a webbot that aids in maintaining network security on a large corporate campus. For example, suppose that you recently discovered that you have a problem with employees attaching unauthorized wireless access points to your network. Since these unauthorized access points occur inside your firewalls and proxies, you recognize that these unauthorized Wi-Fi networks pose a security risk that you need to control. Therefore, in addition to a new security policy, you decide to create a webbot that automatically finds and records the location of all wireless networks on your corporate campus.

You notice that your mail room uses a small metal cart to deliver mail. Because this cart reaches every corner of the corporate campus on a daily basis, you seek and obtain permission to attach a small laptop computer with a webbot and Global Positioning System (GPS) card to the cart. As your webbot hitches a ride through the campus, it looks for open wireless network connections. When it finds a wireless network, it uses the open network to send its GPS location to a special website. This website logs the GPS coordinates, IP address, and date of uplink in a database. If you did your homework correctly, in a few days your webbot should create a map of all open Wi-Fi networks, authorized and unauthorized, in your entire corporate campus.

Track Web Technologies

You could write webbots that use *web page headers*, the information that servers send to browsers so they may correctly render websites, to maintain a list of web technologies used by major corporations. Headers typically indicate the type of webserver (and often the operating system) that websites use, as shown in Figure 2-4.

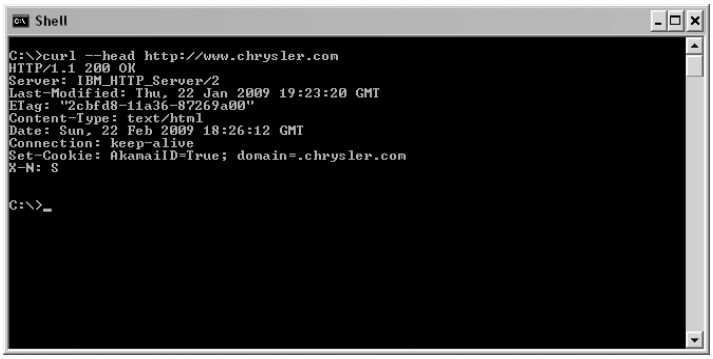


Figure 2-4: A web page header showing server technology

Your webbot starts by accessing the headers of each website from a list that you keep in a database. It then parses web technology information from the header. Finally, the webbot stores that information in a database that is used by a graphing program to plot how server technology choices change over time.

Allow Incompatible Systems to Communicate

In addition to creating human-readable output, you could design a webbot that only talks to other computers. For example, let's say that you want to synchronize two databases, one on a local private network and one that's behind a public website. In this case, *synchronization* (ensuring that both databases contain the same information) is difficult because the systems use different technologies with incompatible synchronization techniques. Given the circumstances, you could write a webbot that runs on your private network and, for example, analyzes the public database through a password-protected web service every morning. The webbot uses the Internet as a common protocol between these databases, analyzes data on both systems, and exchanges the appropriate data to synchronize the two databases.

Final Thoughts

Studying browser limitations is one way to uncover ideas for new webbot designs. You've seen some real-world examples of webbots in use and read some descriptions of conceptual webbot designs. But, enough with theory—let's head to the lab!

The next four chapters describe the basics of webbot development: downloading pages, parsing data, emulating form submission, and managing large amounts of data. Once you master these concepts, you can move on to actual webbot projects.