web scraping con python

**When is web scraping useful?**

Suppose I have a shop selling shoes and want to keep track of my competitor's prices. I could go to my competitor's website each day to compare each shoe's price with my own, however this would take a lot of time and would not scale if I sold thousands of shoes or needed to check price changes more frequently. Or maybe I just want to buy a shoe when it is on sale. I could come back and check the shoe website each day until I get lucky, but the shoe I want might not be on sale for months. Both of these repetitive manual processes could instead be replaced with an automated solution using the web scraping techniques covered in this book.

In an ideal world, web scraping would not be necessary and each website would provide an API to share their data in a structured format. Indeed, some websites do provide APIs, but they are typically restricted by what data is available and how frequently it can be accessed. Additionally, the main priority for a website developer will always be to maintain the frontend interface over the backend API. In short, we cannot rely on APIs to access the online data we may want and therefore, need to learn about web scraping techniques.

1

**Background research**

Before diving into crawling a website, we should develop an understanding about the scale and structure of our target website. The website itself can help us through their robots.txt and Sitemap files, and there are also external tools available to provide further details such as Google Search and WHOIS.

**Estimating the size of a website**

The size of the target website will affect how we crawl it. If the website is just a few hundred URLs, such as our example website, efficiency is not important. However, if the website has over a million web pages, downloading each sequentially would take months. This problem is addressed later in *Chapter 4*, *Concurrent Downloading*, on distributed downloading.

A quick way to estimate the size of a website is to check the results of Google's crawler, which has quite likely already crawled the website we are interested in. We can access this information through a Google search with the site keyword to filter the results to our domain. An interface to this and other advanced search parameters are available at <http://www.google.com/advanced_search>.

**Identifying the technology used by a website**

The type of technology used to build a website will effect how we crawl it. A useful tool to check the kind of technologies a website is built with is the builtwith module, which can be installed with:

**pip install builtwith**

This module will take a URL, download and analyze it, and then return the technologies used by the website. Here is an example:

**>>> import builtwith**

**>>> builtwith.parse('http://example.webscraping.com')**

**{u'javascript-frameworks': [u'jQuery', u'Modernizr', u'jQuery UI'],**

**u'programming-languages': [u'Python'],**

**u'web-frameworks': [u'Web2py', u'Twitter Bootstrap'],**

**u'web-servers': [u'Nginx']}**

We can see here that the example website uses the Web2py Python web framework alongside with some common JavaScript libraries, so its content is likely embedded in the HTML and be relatively straightforward to scrape. If the website was instead built with AngularJS, then its content would likely be loaded dynamically. Or, if the website used ASP.NET, then it would be necessary to use sessions and form submissions to crawl web pages. Working with these more difficult cases will be