Web Crawler

Introduction

Web crawlers, also known as web spiders, are key to the success of search engines. These programs perpetually scan the web, gathering up millions of pages of data and sending it back to search-engine databases. The data is then indexed and processed algorithmically, resulting in faster, more accurate search results. While they are most famously used for search optimization, web crawlers also can be used for automated tasks such as link validation or finding and returning specific data (such as email addresses) in a collection of web pages.

Architecturally, most web crawlers are high-performance multithreaded programs, albeit with relatively simple functionality and requirements. Building a web crawler is therefore an interesting way to practice, as well as compare, multithreaded, or concurrent, programming techniques.

Technology

You will do the same task using the `ExecutorService` and Java 7's `ForkJoinPool` - therefore this assignment rrequires at least JDK 7 and the `HTML Parser` Library: http://htmlparser.sourceforge.net.

Task

Our web crawler's task will be to find and follow links. Its purpose could be link validation, or it could be gathering data. (You might, for instance, instruct the program to search the web for pictures of Angelina Jolie, or Brad Pitt.)

The application architecture consists of the following:

1. An **interface that exposes basic operations to interact with links**; i.e., get the number of visited links, add new links to be visited in queue, mark a link as visited
2. An **implementation for this interface** that will also be the starting point of the application
3. A **thread/recursive action that will hold the business logic to check whether a link has already been visited.** If not, it will gather all the links in the corresponding page, create a new thread/recursive task, and submit it to the ExecutorService or ForkJoinPool
4. An **ExecutorService or ForkJoinPool to handle waiting tasks**

Note that a link is considered "visited" after all links in the corresponding page have been returned.

In addition to comparing ease of development using the concurrency tools available in Java 6 and Java 7, **we'll compare application performance based on two benchmarks:**

- **Search coverage:** Measures the time required to visit 1,500 *distinct* links
- **Processing power:** Measures the time in seconds required to visit 3,000 *non-distinct* links; this is like measuring how many kilobits per second your Internet connection processes.

While relatively simple, these benchmarks will provide at least a small window into the performance of Java concurrency in Java 6 versus Java 7 for certain application requirements.

Assignment

- You will get a Stub-Project. Look for all the `//ToDo` comments. There you have to insert your own implementations!
- Implement some basic logging information (nb links visited, time elapsed)