Advanced Programming

Lab Sheet 22 – Creating Web Services

Introduction

This lab sheet follows on from the previous set of exercises, in which you created an application that consumes a web service. In this set of exercises, you will create a simple web service using the Spark Java framework.

Web services are an important topic; they're currently the dominant architecture used for large scale systems in industry. You can test web services that are built to respond to requests using the HTTP GET method using your web browser. As with all software that opens up a network port for listening, you can only have one program using a port at once, so remember to quit old versions of your server before running new ones!

Task 1: A Simple Web Service

Create a web service that responds to "/hello" requests with a simple HTML "Hello World" response. This is very similar to the example in the lecture. You will need to download the Spark Java library from moodle and add it as a dependency to your eclipse project in the normal manner. Remember the import static in the example code! Eclipse will not be able to add that automatically using the ctrl+shift+O shortcut. Verify that you have everything set up and running by navigating to http://localhost:8080/hello in your browser, and remember to kill the server when done testing.

Extend your web service so that it can also respond to "/stats" requests. When responding to these requests, your service should provide some basic information about the running JVM, such as the amount of memory currently in use and free, and the number of CPU cores being used by the JVM. To find this information, you will need to read the documentation for the Runtime class.

Extend your web service further so that it can also respond to "/dir" requests. When responding to these requests, your service should provide a listing of the files and directories inside its current working directory. If you were developing a real-world application, this could pose a security risk, but for our purposes this week, it is OK.

Task 2: Parameter Passing

Create a HTML file inside your Java project, and place inside it a simple HTML form. The form should use the GET method, and submit its data to http://localhost:8080/colour/ for processing. The form should contain two fields, a name and a favourite colour.

Create a Spark Java web service that will run on port 8080 and respond to "/colour" GET requests. In response to these requests the service should output a simple greeting for the user, coloured in their favourite colour using CSS (in-line is fine for this). This means that your web service will need to retrieve the name and favourite colour from the query portion of its URL, and that the service will only work with valid CSS colours.

Create a second version of the HTML and service that is capable of passing the data via the POST method, instead of GET.

Task 3: Re-Creating the Hygiene Web Service

This is quite a large task, so it might be that you don't get round to this before starting work on the assignment. Even if you don't get round to this task now, it might be worth spending some time on over the break, or even over the summer to keep your skills sharp for next year. Remember, next year you will be working on a final year project, which will involve you designing and building a fairly large system from scratch.

In the previous set of exercises, you accessed a web service to look up the hygiene rating of restaurants based on their postcode. The service that I gave you was written in PHP, but you could easily accomplish the same task using Spark Java. If you build the service correctly, you should be able to swap the URL of the service from sandbox.kriswelsh.com to localhost and your app should continue working.

There a number of challenges in creating the hygiene rating web service:

1. The data (available here) is in a truly awful XML-based format, with one file created by each council in the UK, containing the results of their inspections. You will need to write a program that saves the results into a mySQL or SQLite database.

2. Users of the web service might make silly mistakes with the parameters. What if someone forgets to tell you what search type to perform? What if they forget the postcode?

3. The web service that I gave you access to also supports searches based on establishment name, and using the latitude/longitudes stored for each restaurant to allow searches for places nearest to the specified location. How can you implement this functionality such that the load on the server from processing requests from many users remains manageable?