Java Programming Laboratory

Java Web Programming

Exercise 1. Using the Java URL class and its open stream method disply the contents of the MMU web page (http://www2.mmu.ac.uk/) on the console.

Exercise 2. Using the Java ServerSocket object write a server called SocketDateServer which listens on localhost port 3000 for a client to attempt to make a connection. Code the server so that once a client has made a connection the current date and time can be sent to the client using a PrintWriter object. Write a client called SocketDateClient which creates a socket object to connect to the DateSocketServer on port 3000 and check that the correct date is returned from the server.

Exercise 3. Using the Java ServerSocket object write a server called SocketCapitalizeSentenceServer which listens on localhost port 3001 for a client to send text data. Code the server so that once a client has made a connection and sent a line of text, the text sent back to the client is capitalized. Write a client called SocketCapitalizeSentenceClient which creates a socket object to connect to the SocketCapitalizeSentenceServer on port 3000 and check that capitalized text is returned from the server.

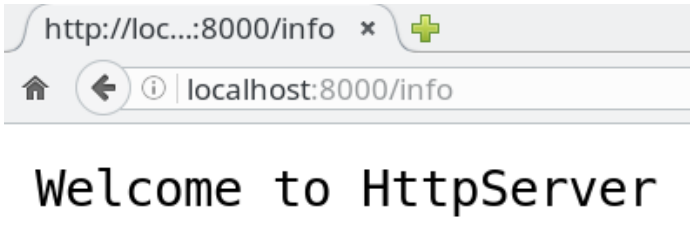
Exercise 4: Using the Java HttpServer class create a server called SimpleHttpServer with the code shown below.

```
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;

import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpServer;

/*
 *  httpserver
*/
public class SimpleHttpServer {

  public static void main(String[] args) throws Exception {
    HttpServer server = HttpServer.create(new InetSocketAddress(8000), 0);
    server.createContext("/info", new MyHandler());
    server.setExecutor(null); // default implementation of threading
    server.start();
    System.out.println("The server is up and running on port 8000");
  }

  static class MyHandler implements HttpHandler {
    public void handle(HttpExchange t) throws IOException {
      String response = "Welcome to HttpServer";
      t.sendResponseHeaders(200, response.length());
      OutputStream os = t.getResponseBody();
      os.write(response.getBytes());
```

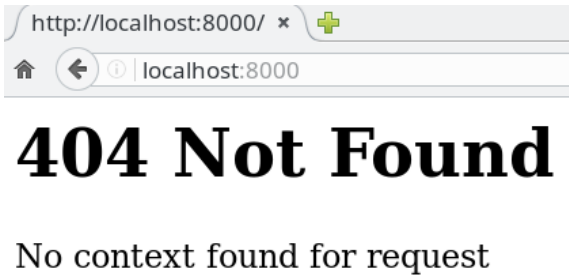
```
      os.close();
    }
  }
}
```

Run the code as a Java application and you should see a message “The server is up and running on port 8000” in the console. Now open a browser and enter the URL address

http://localhost:8000/info



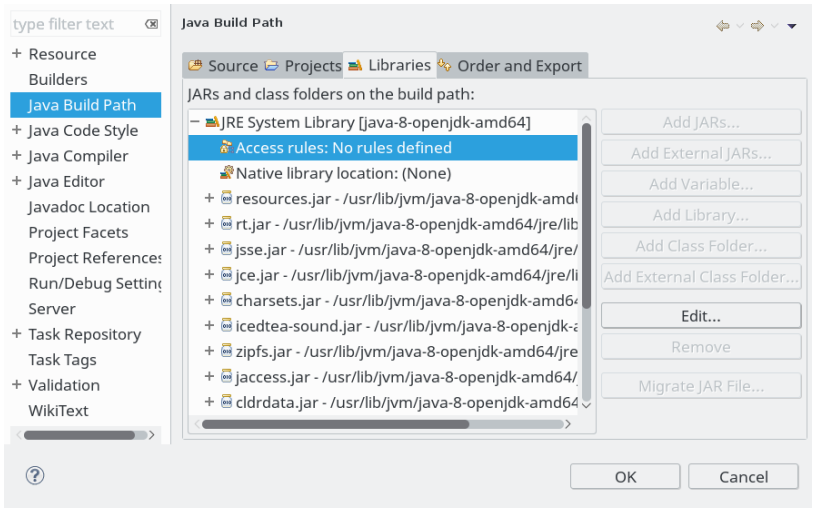
Note if you just use the address http://localhost:8000 without the “/info” context you will get an Http 404 error.



With some versions of Eclipse you have to add an access rule for `com.sun.net.httpserver.HttpServer`. This is done by right clicking on the project and selecting build path and then configure build path.

Right Click Project→ Build Path→ Configure Build Path

Then go to libraries and select Access rules and Edit.



Enter the Access rule as shown below making sure the resolution is set to “Accessible”.

Exercise 5: Using the Java HttpServer class create a date web service which returns current date to a client. The server should use two HTTP context paths as shown below.

```
import java.io.IOException;
import java.net.InetSocketAddress;
import com.sun.net.httpserver.HttpServer;
 /**
  * Date RESTful web service at http://localhost:3000/get-date
  * @author Alan Crispin
  *
  */
public class ControllerHttpServer {

public static void main(String[] args) {

try {
      HttpServer server = HttpServer.create(new InetSocketAddress(3000), 0);
      server.createContext("/", new GetHomeHandler());
      server.createContext("/get-date", new GetDateHandler());
      server.setExecutor(null); //default implementation of threading
      // start the server
      server.start();
      System.out.println("Server running on port 3000");
      }
      catch (IOException io) {
            System.out.println("Connection problem: "+io);
      }
}
}
```

This means that the URL pattern http://localhost:3000/ is handled by the GetHomeHandler class which shows the home page while the URL pattern which uses the path (route) /get-date i.e. http://localhost:3000/get-date returns the current date.

Each of the Handlers implement HttpHandler interface which has one method called "handle" which has to be overridden. This provides an HttpExchange object which is used to communicate with the client. The HttpExchange class encapsulates an HTTP request received and a response to be generated in one exchange. It provides methods for examining the request from the client, and for building and sending the response. The code for the GetDateHandler is shown below. The first step is to create a BufferedWriter object called "out" using the HttpExchange object method called the getResponseBody(). The getResponseBody() method is used to get an OutputStream to send the response. The HttpExchange object is to send the response headers (i.e. 200 OK) using the method called sendResponseHeaders(int,long).When the response has been written, the BufferedWriter stream must be closed to terminate the exchange.

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.Date;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;

public class GetDateHandler  implements HttpHandler {
	@Override
public void handle(HttpExchange he) throws IOException {

BufferedWriter out = new BufferedWriter(new OutputStreamWriter(he.getResponseBody()));
he.sendResponseHeaders(200, 0);
out.write(new Date().toString());
out.close();
	}
}
```

For more information about the HttpExchange class and how it encapsulates an HTTP request received and a response to be generated in one exchange see:-

https://docs.oracle.com/javase/7/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpExchange.html

Exercise 6: Write a web service tester class to test the date web service as shown below.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;

public class WebServiceTester {

	public static void main(String[] args) {
		System.out.println("Date = " + getDateTest());
	}

	private static StringBuffer getDateTest() {
	StringBuffer response = new StringBuffer();
	try {
	URL url = new URL("http://localhost:3000/get-date");
	BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
	String output;
	while ((output = reader.readLine()) != null) {
			response.append(output);
	}
	reader.close();
	} catch (Exception e) {
```

```
        System.out.println(e.getMessage());
        }
        return response;
        }
}
```

Exercise 7: Use a REST client such as Firefox RESTClient or Google postman to test the date web service of exercise 5.

Alan Crispin (14/10/17)