SENG2050 Week 3 Part B

March 2022

In this laboratory we will learn how to:

- Use request parameters;
- Link servlets together; and
- Validate pages with Javascript.

Place this weeks content in a new folder called 'lab02' in the Tomcat webapps directory. These tasks build on last week's, you way want to copy last week's lab content into this week's lab folder.

Remember, never copy-and-paste code out of this lab sheet. It can cause errors.

1 The Request Object

You can pass parameters to your servlet via request parameters like so:
`http://localhost:8080/lab02/YOUR_SERVLET?PARAMETER_NAME=VALUE`

For example, to pass my name to a servlet with the URL "/DisplayMyName" I would use:
`http://localhost:8080/lab02/DisplayMyName?name=Hayden`

If you want to pass more than one parameter to a servlet separate the parameters with "&", i.e.
`http://localhost:8080/lab02/DisplayMyName?firstName=Hayden&lastName=Cheers`

Passing parameters into a servlet is meaningless unless something is done with the parameters. To access the parameters from your servlet you must get them from the HttpServletRequest like so:

```
String value = request.getParameter("PARAMETER_NAME");
if(value==null){
        //the parameter wasn't passed
}else if(value.equals("someValue")){
        //the parameter was an expected value
        //sometimes you may like to check against a regex
}else{
        //some other value was found
}
```

You can also pass multiple values as a single named parameter. This is exposed as an array of values. For example with the following URL:

`http://localhost:8080/lab02/LuckyNumbers?number=1&number=3&number=6&number=4`

We can retreive the array of 'number' values with:

```
        String[] numbers = request.getParameterValues(``number'');
```

1.1 Task 1 - Reading Parameters

With this knowledge, create a servlet that reads your first and last name from the request parameters. Use these values to generate a valid HTML page that presents a welcome message, containing your first and last names.

2 Multiple Classes

Writing all this HTML code as Java strings must be getting exhausting and doesn't look very maintainable. Create a new class to offload all the HTML generation, this class should contain methods that return properly formatted HTML tags such as:

```
public class HtmlGen{
        public static String doctype(){
                return "<!DOCTYPE html>\n<html lang=\"en\">";
        }

        public static String head(String title){
                return "<head><title>" + title + "</head></title>";
        }

        public static String h1(String heading){
                return "<h1>" + heading + "</h1>";
        }

        //Other methods to generate frequently used tags
        //maybe generate a table
        //...
        //...
}
```

2.1 Task 2 - HTML Generation Helpers

Create a HtmlGen class such the one listed above. Make sure it contains methods to generate the following tags:

- HTML doctype
- title
- headings (1, 2, 3)
- script & css

You can always add more later. Create a servlet to test these out.

2.2 Task 3 - Packages

It is good practice to package your Java classes. This is a standard Java convention that we all need to follow. So far we have always used the 'root' java package. i.e. the unnamed package. A Java package is similar to a C++ namespace, but it enforces where you can store you .java files. For example, if I want to store my Sevlets in a package named 'servlets', they have to be in a folder also named 'servlets' and contain the following package declaration on the first line of the .java file:

```
package servlets;
```

Now that we have more than one class, lets go ahead and move our classes to a package. Move the servlets from task 1 & 2 into a Java package. Make sure to put both classes into a Java package, you can easily compile these classes at the same time with:

```
javac *.java
```

3 Linking Servlets

In task 1 we had to manually type our request parameters into the address bar. This is quite tedious, and the average user would not know what to do.

Instead of having to type in the URL request parameters in the address bar let's create a new servlet that displays a form to capture this data. The HTML for such a form follows:

```
<form action="DisplayMyName" method="get" onsubmit="return true;">
        <label for="firsName">First Name</label>
        <input type="text" name="firstName" id="firstName" /><br>
        <label for="lastName">Last Name</label>
        <input type="text" name="lastName" id="lastName" /><br>
        <input type="reset" value="Clear" />
        <input type="submit" value="Submit" />
</form>
```

Note: The 'action' attribute should match the destination servlet's URL. The method should match the method you have implemented in the servlet (i.e. method="get" would execute the "doGet" method, method="post" would execute the "doPost" method).

3.1 Task 4 - Submitting forms

Create a servlet that outputs this form. Once it is working change the value of the "method" attribute from "get" to "post" and see what happens.

Remember, the two most common HTTP methods are 'get' and 'post'. Our servlets contain methods for handling both of these request types. How can we change the DisplayMyName servlet to handle a 'post' form?

4 JavaScript Validation

JavaScript is a language that runs on the client (the web browser). It may look like Java but it is not. Javascript and your Java servlets do not communicate directly. You cannot reference your servlets from Javascript. It can only operate upon the HTML document the servlets generate. To find out more about JavaScript visit http://www.w3schools.com/js/.

It is always a good idea to validate any user input. This next exercise adds input validation to the form using JavaScript. Link to the following JavaScript file from your form servlet:

```
function validate(){
        var firstName = document.getElementById("firstName");
        var lastName = document.getElementById("lastName");
        var resultStatus = true;
        var messageStr = "The following errors were detected:\n";
        if (firstName == null || firstName.value == "") {
                resultStatus = false;
                messageStr += "- First name is empty or blank\n";
        }
        if (lastName == null || lastName.value == "") {
                resultStatus = false;
                messageStr += "- Last name is empty or blank\n";
        }
        if (!resultStatus){
                alert(messageStr);
        }
        return resultStatus;
}
```

4.1 Task 5 - Using Javascript

Store this Javascript in a file at 'webapps/lab02/js/validate.js', and load it with:

```
<script src=''/lab02/js/validate.js''></script>
```

Lastly, to run the script when the submit button is clicked you must change the form's "onsubmit" attribute from "return true" to "return validate()"