SENG2050 - Lab 02

February 26, 2018

Note: You may need to set your CLASSPATH and JAVA_HOME on the lab computer each time you log in.
Note: Create a webapps/lab02/ directory to complete these tasks. These tasks build on last week's, you way want to copy last week's lab content into this week's lab folder.
Note: When working in a UON lab, you will not be able to access any pages with Chrome or Firefox. On Edge you can access pages via http://localhost:8080. Do not forget to add http://.
Note: Remember to **NEVER** copy text straight out of this lab sheet.

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

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 (check blackboard for source):

```
String value = request.getparameter("parameterName");
if(value==null){
        //the parameter wasn't passed
}else if(parameter.equals("someValue")){
        //the parameter was an expected value
}else{
        //some other value was found
}
```

With this knowledge, create a servlet that reads your first and last name from the request parameters and displays a welcome message in a valid HTML5 page.
Note: To pass multiple values as a single parameter (an array of values) reuse the parameter's name i.e.
`http://localhost:8080/lab02/LuckyNumbers?number=1&number=3&number=6&number=4`
Then retrieve it with

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

2 Multiple Classes

Writing all this HTML code within 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 (check blackboard for source):

```
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
        //...
        //...
}
```

It is good practice to package your Java classes, now that we have more than 1 class let's go ahead and do it. Recreate the above servlet using this new helper class. Make sure to put both classes into a Java package, you can easily compile these classes at the same time with:

```
javac *.java
```

Note: Classes with the "package" statement must reside within a folder of the same name, i.e. a class with "package myJavaPackage;" would need to be within a folder called "myJavaPackage".

3 Linking Servlets

Instead of having to type in the URL request parameters in the address bar let's create a new servlet with a form to capture this data. The HTML for one such form follows (see blackboard for source):

```
<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>
```

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.
Note: The action 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).

4 JavaScript Validation

JavaScript is a language that runs on the client (the web browser), it may look like Java but it is not. 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 (see blackboard for source):

```
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){
                resultStatus = false;
                messageStr += "- Could not find input with id 'firstName'\n";
        }
        if(!lastName){
                resultStatus = false;
                messageStr += "- Could not find input with id 'lastName'\n";
        }
        if(resultStatus){
                var firstNameValue = firstName.value;
                var lastNameValue = lastName.value;

                if(firstNameValue == ""){
                        resultStatus = false;
                        messageStr += "First name is blank\n";
                }
                if(lastNameValue == ""){
                        resultStatus = false;
                        messageStr += "Last name is blank\n";
                }
        }
        if (!resultStatus){
                alert(messageStr);
        }
        return resultStatus;
}
```

Hint: Save the JavaScript file to webapps/lab02/js/validate.js
To load it use:

```
<script src=``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()"