SENG2050 – Lab JSTL

This laboratory will apply JSP EL + JSTL in our Java web applications. We will utilise our lab tasks from the JDBC and MVC modules, and refactor them to use EL + JSTL.

Make sure you complete the JDBC & JDBC modules before continuing. Also, watch the lecture & demo videos to gain an understanding of how we utilise EL + JSTL.

1 Basic EL + JSTL

EL can be used almost anywhere in a JSP page. It is best used in JSP tag attributes. Before we start using JSTL tags, make sure you place the associated taglib jars in the `WEB-INF/lib` directory.

The includes for our JSTL libraries are:

- `<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>`
- `<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>`
- `<%@ taglib prefix = "fn" uri = "http://java.sun.com/jsp/jstl/function" %>`
- `<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>`

To reference a bean:
`${beanId}`

To reference a bean property:
`${beanId.property}`

To access a collection (array, collection, map):
`${beanId.collection[index]}`

Method can be called:
`${beanId.methodName(arg1, arg2, ...)}`

Remember, we should never use EL to print values out raw to the page. Always sanitise it with:
`<c:out value="${value}" />`

We should always try to use JSTL + EL in JSP pages. Avoid using Java - when we use Java we risk putting controller or model logic in our views.

2 JDBC

In the JDBC lab, we developed an application which can query a MSSQL database, and display a list of records. In this section, we will refactor this application to utilise EL + JSTL.

Start by refactoring you application to hide the JSP page displaying the list of movies behind a Servlet, as per good MVC design. In the servlet setup the page by querying the database, and injecting the records into the request object, as below:

```
    request.setAttribute("movies", Movie.getAllMovies());
```

This servlet acts as the controller. It sets up the environment such that the JSP can render the view data. To display the results, forward the request to the page movies.jsp:

```
RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/path/to/movies.jsp");
dispatcher.forward(request, response);
```

Refactor your JSP such that it requires no JSP scriptlets - i.e. no Java code. Only utilise the JSP EL + JSTL. Your JSP will need the following:

```
    <%-- Import the Core taglib--%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <%--create the HTML table using foreach tag--%>
    <table>
        <tr><th>Name</th><th>Year</th><th>URL</th></tr>
        <c:foreach var="movie" items="${movies}">
            <tr>
                <td><c:out value="${movie.name}"/></td>
                <td><c:out value="${movie.year}"/></td>
                <td><c:out value="${movie.url}"/></td>
            </tr>
        </c:foreach>
    </table>
```

Because your JSP isn't calling the getAllMovies() method, it can be reused by different controllers that call different methods to create a list of movies. For example, your controller might run a method that queries the database for all movies from a certain year. If you have time you should do this.

3 Java Beans, Sessions, Cookies & MVC

In lab 3, 4 & 5, we developed basic applications which utilised JSP pages. Refactor these applications to remove **ALL** Java code from our JSPs. This means, we will have **NO** scriptlets, whatsoever. In the MVC workshop, we refactored our applications to have distinct Models, Views and Controllers. We will continue this refactoring in this lab. In particular, we will remove all Java from our JSPs, and place all setup code and controlling code in a servlet - as per MVC.

From lab 3, refactor the table creator to use a JSTL `<c:forEach ... >` loop. Also, replace all `<jsp:useBean ... />` tags to use EL + `<c:out ... />`.

From lab 4 & 5, refactor the shop and quiz application to hide all JSPs behind Servlets (using the `RequestDispatcher`). Remove all the bean setup code from the JSPs with a `<jsp:useBean ... />` tag. Refactor bean mutations (setProperty or setter calls, adding an item to the cart, setting a quiz anwser) such that it is performed in a Servlet in a `doPost`, rather than in the JSP page. Replace all scriptlets with their JSTL equivelents.

Watch the module instructional videos to gain insight on how we can do this.

4 Further Tasks

Further apply your knowledge of JSP EL + JSTL by refactoring the directed learning tasks from the JDBC module. In general, all requests should be serviced by a Servlet, which queries a Bean, before being forwarded on to a JSP page. Make sure there is no Java in any of our JSPs!