Application Design And Implementation

Daniel R. Cender

Grand Canyon University: CST-235-O500

July 12, 2020

**All Source Code Retrievable At:** [**https://github.com/DanielCender/CST-235**](https://github.com/DanielCender/CST-235)

**Explanation**

The applications built in this assignment show the bare-bones implementation of Managed Beans with JSF pages or web servlets with JSP pages. The first assignment section illustrates how Servlet controllers can dynamically determine which parameters or views to present to the user based on the access route and other conditions. The second section uses a simple Controller (also a Managed Bean) to bind logic and a simple data model to XHTML JSF pages. The server itself seems to extract all classes marked with Bean annotations and inject them into the XHTML views where specified by their model name.

Below is a simple breakdown of all the classes/views in the assignment.

**Classes**

* assignment2a
  + TestServlet.java (Web Servlet)
    - Methods: init, doGet, doPost, destroy
* assignment2b
  + User.java (Managed Bean)
    - Methods: constructor, overloaded constructor, getters/setters
  + FormController.java (Managed Bean)
    - Methods: onSubmit, onFlash

**Pages**

* assignment2a
  + TestForm.jsp
  + TestResponse.jsp
* assignment2b
  + TestForm.xhtml
  + TestResponse.xhtml
  + TestResponse2.xhtml

**assignment2a**

This simple application evolved over the course of this guide. The first screenshot is of the original TestRequest servlet which logs messages from lifecycle methods to the server console.

A screenshot of a computer screen

Description automatically generated

The second screenshot of this section shows the HTTP parameters logged to the server console.

A screenshot of a computer screen

Description automatically generated

The next screenshot shows the result of forwarding the HTTP request and parameters to a TestResponse.jsp page. That page renders the HTTP parameters to a simple HTML body.

A screenshot of a computer screen

Description automatically generated

This last portion of the assignment section required implementing a simple HTML form and processing form submission with the TestServlet.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**assignment2b**

One of the first steps of this section was to deploy a facelet-enabled XHTML page to the JBoss server and retrieve data from a Java Bean. This page displayed the “firstName” and “lastName” properties of the User managed bean.

A screenshot of a computer screen

Description automatically generated

The instructions then called for creating a JSF response page for a simple JSF login form. The form sets the “firstName” and “lastName” values on the managed User bean in the FacesContext request parameters. The form and the ResponsePage.xhtml results are below.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

The next step in this segment was to implement an “onFlash” handler method in the FormController. This redirects the browser to the “TestResponse” page without updating the state of the User bean. It’s clear from the Network tab that the application redirected the browser to another page and displayed the stale User bean.

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

The final step in this assignment was to fix the Flash redirect so the managed bean received updates properly. I duplicated the TestResponse view to a new file and prefixed the EL template calls with “flash.” before setting the onFlash redirect to use the new view. The screenshots below show the new view in the project directory and the resulting view from firing the “Flash” action on the form.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated