CS 320: Lab 2A: Web Applications II

Your Task

This lab builds upon the **AddNumbers** and **GuessingGame** examples from CS320_Lab02. It will "force" you to investigate, explore, and learn the structure of a WebApp, from the server (**main()**), through **Web.xml**, the servlets, the controllers, the model, and the JSPs, as well as how to refactor project components. You will need to have this kind of familiarity with the WebApp architecture in order to get started on your team project, as well as to be able to make significant technical contributions to any and all aspects of your team project.

NOTE: You must work individually on this lab. It is imperative that everyone on your team be able to full comprehend, design, implement, toubleshoot, and debug the complete architecture of a standard client-server application. This **WILL** come up on the exam.

NOTE: This lab will require a significant effort, and you should start on it as soon as possible.

You have six tasks:

- First, make a copy of the **CS320_Lab02_username** project, and then refactor that copy to **CS320_Lab02a_username**. YOU MUST DO THIS BEFORE YOU EXPORT YOUR SOLUTION AND SUBMIT IT TO MARMOSET. **Submissions that DO NOT adhere to that guideline will not be graded until they are refactored and resubmitted.**

Example: Professor Zeller would refactor the copy of his **CS320_Lab02_dzeller** project to **CS320_Lab02a_dzeller** first thing after he imported the project into Eclipse.

- Add a third number to the **AddNumbers** example, such that a third number field shows up on the **AddNumbers** web page, and the returned result is the sum of all three numbers. Your code should check for the same errors as the existing **AddNumbers** example does. This problem will require changes to the **AddNumbers** servlet, controller, and JSP.

- Create a new WebApp based on the **AddNumbers** example, called **MultiplyNumbers**. This WebApp should accept two numbers from the user and return the product of those numbers. It should check for all of the same errors that the existing **AddNumbers** example does. This problem will require changes to **Web.xml**, and the controller (modify the existing **AddNumbersController** and refactor it to **NumbersController**, rather than creating a new controller). You will also need to create a new **MultiplyNumbers** servlet and new **MultiplyNumbers** JSP file, both of which can be based off the **AddNumbers** versions of those files.

- After you have modified **AddNumbers** and created **MultiplyNumbers** and have them both completely working, modify the existing **Index** JSP file so that you can access all three of the servlets (**AddNumbers**, **MultiplyNumbers**, and **GuessingGame**) from the **Index** page. Implement this is a collection of three "submit" buttons, each labelled with one of the three functions. Each button will result in **IndexServlet** issuing a **doGet** to the appropriate JSP.

- After all of the above are working, create a common model class (**Numbers.java**) for **AddNumbers** and **MultiplyNumbers**, such that the numbers submitted through the HTTP request are stored in the model, along with the result. You will need **get** and **set** accessor methods for all model fields, along with a constructor for the model class. Look at the model for the **GuessingGame** WebApp as a template - and how that model is instantiated, initialized, and accessed. The corresponding JSP's should pull the data

directly from the **Numbers** model, rather than from the HTTP request. Again you can look at the **GuessingGame** implementation to see how this is done.

- As part of this exercise, implement a complete set of test cases for the controller and model classes. Implement those test cases as you develop each of the above tasks. A significant portion of the effort on your team project will be creating an extensive and comprehensive set of JUnit test cases - **WHILE YOU ARE DEVELOPING YOUR TEAM PROJECT**.

- **NOTE: We HIGHLY recommend that you save your work after you have each of the above steps working.** To that point, we also highly recommend that you use this lab as a means for learning how to use **git** to store your changes on **GitHub**. At a minimum, archive the project at each stage, and submit it to Marmoset. That will provide a back-up in case you lose your work, or if you need to revert to a previous version of your work.

NOTE: You must work individually on this lab. It is imperative that everyone on your team be able to fully comprehend, design, implement, toubleshoot, and debug the complete architecture of a standard client-server application. This **WILL** come up on the exam.

Submitting

NOTE: You must also get this lab signed off by a tutor/mentor or your instructor in order to get credit for it. Simply submitting the lab to Marmoset without a lab sign-off will **NOT** result in any credit. **You must get a lab sign-off**.

Here is the complete set of tests that will be used to sign-off on your lab.

Here is a PDF version of the complete sign-off criteria that you can **download, print, and use as a checkoff list when you get the sign-off for the lab.**

Although you should strive to satisfy all of the above tests, a minimum passing submission must have the following:

- An **operational Index page** that links to all 3 applications
- The **Add Numbers page** must allow entry of three numbers
- A **Numbers.java** model file
- **AddNumbers.JSP** and **MultiplyNumbers.JSP** must pull their field data directly from **Numbers.java** and **NOT** from the parameter values passed in through the HTTP POST received by the the Servlet.
- There will be at least **some test cases** for the methods in **MultiplyNumbersController.java** and **Numbers.java**

When you are done, submit the lab to the Marmoset server using the method below.

From a web browser

Save your project (**CS320_Lab02a_username**) to a zip file by right-clicking it and choosing

Export…→Archive File

Upload the saved zip file to the **Lab02a** project on the Marmoset server. The server URL is

https://cs.ycp.edu/marmoset/