Design and Implement Enterprise Java Beans

Daniel R. Cender

Grand Canyon University: CST-235-O500

July 26, 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 build on the assignment code from last week. Validation is added to the Java Managed Beans and JSF forms to enforce input data restrictions. I replaced the bare XHTML with use of a general layout, comprised of smaller ui:composition JSF view components. Now, every page has the same header/footer layout. Furthermore, the use and styling flexibility of JSF data grids was explored on the TestResponse.xhtml page.

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

**Classes**

assignment4

* + User.java (Managed Bean)
    - Methods: constructor, overloaded constructor, getters/setters
  + Order.java (POJO)
    - Methods: constructor, getters/setters
  + Orders.java (Managed Bean)
    - Methods: constructor, getters/setters
  + OrdersBusinessInterface.java (EJB interface)
    - Methods: test
  + OrdersBusinessService.java (EJB Implementation)
    - Methods: constructor, implements test
  + AnotherOrdersBusinessService.java (alternative EJB Implementation)
    - Methods: constructor, implements test
  + FormController.java (Managed Bean)
    - Methods: onSubmit, onFlash

**Pages**

assignment3

* + TestForm.xhtml
  + TestResponse.xhtml
  + TestResponse2.xhtml

**assignment4**

This assignment sees the first use of Enterprise Java Beans (EJBs) for this course. The first step is to create a simple service interface and EJB implementation, which should be injected into the FormController when instantiated. As can be seen in the screenshot below, the “test” method prints a simple message when it runs from the FormController’s “onSubmit” method.

A screenshot of a computer screen

Description automatically generated

After changing the list of “alternatives” classes in the beans.xml file, The AnotherOrdersBusinessService gets injected into the application runtime. The printed string from the other business service can be seen in this second image:

A screenshot of a computer screen

Description automatically generated

The next phase of this assignment required adding class-scoped lists of orders to these beans. Once a public getService method was made available to the JSF pages via the FormController, the TestResponse page can now display data directly from the EJBs. The first screenshot below shows the results of tying the AnotherOrdersBusinessService bean into the controller. The second shows the list of orders associated with the OrdersBusinessService EJB. You can see the applicable strings from each Bean still printing in the console from their respective “test” methods.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated