Design and Implement Enterprise Java Beans

Daniel R. Cender

Grand Canyon University: CST-235-O500

July 25, 2020

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

**Explanation**

The additions to code in this assignment involved adding Enterprise Java Beans to implement business logic. The business logic classes perform simple tasks using the POJO’s in the “beans” package, like populating a list of orders. A timer service was programmed additionally to log messages on a fixed interval.

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
  + MyTimerService.java
    - Methods: constructor, setTimer, programmicTimer, scheduledTimeout
  + FormController.java (Managed Bean)
    - Methods: onSubmit, onFlash, getService

**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

The very last requirement to complete this assignment was to implement a timer-triggered console logging process, using EJBs. Once injected into the FormController, the timer service would continue to log the message defined in the “@Timeout”-annotated method on a fixed interval. The timeout on this bean was set to 10 seconds, which shows when the timestamps between the last and second-last console lines are compared.

A screenshot of a computer screen

Description automatically generated