REST-Based Services and Message-Based Services

Daniel R. Cender

Grand Canyon University: CST-235-O500

August 5, 2020

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

**Explanation**

This milestone enables the sample web application to serve XML and JSON-formatted data to REST client requests.

**Classes**

assignment6a

* + 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, getOrders, setOrders, sendOrder
  + OrdersBusinessService.java (EJB Implementation)
    - Methods: constructor, implements test, implements getOrders, implements setOrders, implements sendOrder
  + OrdersRestService (JAX-RS REST Route)
    - Methods: getOrdersAsJson, getOrdersAsXml
  + OrderMessageService (JMS Queue)
    - Methods: constructor, onMessage
  + AnotherOrdersBusinessService.java (alternative EJB Implementation)
    - Methods: constructor, implements test, implements getOrders, implements setOrders
  + MyTimerService.java
    - Methods: constructor, setTimer, programmicTimer, scheduledTimeout
  + FormController.java (Managed Bean)
    - Methods: onSubmit, onFlash, getService, getAllOrders, insertOrder, onSendOrder
  + OrdersDataInterface.java (Managed Interface)
    - Methods: getConnection, get, getAll, save, update, delete
  + OrdersDataService.java (Managed Bean)
    - Methods: get, getAll, save, update, delete

**Pages**

assignment6a

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

**assignment6a**

The first phase of this assignment involved implementing two routes where a list of orders could be retreived in either JSON or XML format. The first two screenshots below show the prettified result data from the default CDI business service.

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated

The screenshots below are from accessing the same routes via a GET request in Postman:

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

The next portion of the assignment required implementing a simple message queue with JMS. Using a JSF button, the FormController submitted a TextMessage to the queue for processing by the OrdersMessageService class. After processing, a response message is shown to the user on a redirect page, as shown below.

A screenshot of a cell phone

Description automatically generated

In the last step of this assignment, I had to send a second message (an ObjectMessage) that would be parsed back to an Order when received and stored in the PostgreSQL database. I added an OrdersDataInterface and an OrdersDataService to clean up the service injections. Once the message was properly type-casted to an Order, I called the “service.save(order)” operation. After quickly updating the OrdersBusinessService to use the data service’s “.getAll()” query method, the TestResponse.xhtml page showed a full list of the testapp.Orders table’s records. The first screenshot shows the logged results of the save process, and the second shows the visual updates to the response page after inserting a plethora of new Orders.

A screenshot of a computer screen

Description automatically generated

A screenshot of a cell phone

Description automatically generated