**CST-235**

**REST, SOAP, and Message Services**

**Project Report**

**11/29/2018**

**Roman Parkhomenko**

The purpose of this application was to become familiarized with the different web services available in the Java EE environment. We did this by implementing JAX-RS, JAX-WS, and Message-Driven EJB’s.

The first service we implemented, and in my opinion the easiest to understand and implement, was the REST service. We used the Java Rest Services API to develop a way to easily access data from our web application. RESTful web services use different HTTP requests to manipulate resources. To implement this, we created a new OrdersRestService class. This class declares the format of data accepted, XML and JSON, and then utilizes the existing OrdersBusinessInterface to access the methods we already had in place. The only thing we had to do was annotate the methods in the REST Service with the appropriate HTTP requests, and annotate the data format of the method. We then utilized Postman to test our new RESTful API and based on the XML or JSON method called, we received data from the server containing our objects. Due to the scalability, flexibility, and ease of implementation, I find it difficult to favor the other web services over the RESTful architecture.

For the second part of this assignment, we created a OrdersSoapService. SOAP is an older protocol for developing an API. It has built-in security, authorization, and error-handling functionality, but it’s significantly more rigid, provides slower load times, and I personally think that configuration was really confusing. However, after going through the difficulty of configuring the SOAP service into the project, we implemented a sayHello method as well as the getOrders method. These methods are registered in a WSDL file, which contains the configuration for the SOAP service. After creating and annotating the methods appropriately, we used the RedHat web services tester to ensure the configuration was working. SOAP is limited to XML, so passing parameters can be more difficult as well since arguments have to be passed through an XML file. After testing it in the RedHat IDE, we used a tool called SoapUI which provides a more streamlined GUI to be able to test our API methods. Overall, I didn’t particularly enjoy configuring SOAP, and while the advantages exist, the XML limit and configuration proved to be cumbersome.

For the last part of the application, we developed a Message-based service with ActiveMQ. I had the most trouble configuring this service and seeing the purpose of it. To utilize the message service, we first had to configure our EAP server with the ActiveMQ extension. After doing so, we created a Message-Driven EJB called OrderMessageService. This service contained only one onMessage() method. This method worked by use creating a session-based connection pool that had a queue where we could add orders. We had to alter the OrderBusinessInterface and OrderBusinessService to get this to work by adding connection pooling, creating a TextMessage with our information, passing that information to the Messaging service, where we had to cast the data to be able to access it. At the end of all those steps, we basically had a console log saying an object had been added to the database. Message-based services by far seem the most impractical in regards to configuration, implementation, and ease of access for external API users.

As I mentioned above, the purpose of this assignment was to familiarize ourselves with the tools available to create an API. From what I’ve researched, the RESTful architecture seems to be the most practical, efficient, and most flexible way to create an API.