ACME Web Shop

JOURNAL- LAB2

Vineet Choudhury
Syed Wahaj Ali
Alex Rudenko

IMSE 2012

CONTENTS

Chapter1: Shipment Service

1.1 Overview

What is the role of Shipment Service in ACME web shop implementation?

The Shipment Service provides shipping service to an ACME web shop customer. During the ordering process, the customer is given the possibility of choosing either express or regular shipping type. The customer is also given information about cost of shipping and number of days required for each kind of shipping type.

What are the criteria used by shipment service to calculate available shipment types and cost. In other words what is the business logic used by the shipping service?

The shipment service calculates available shipment types along with their respective cost and days required to deliver based on two parameters - destination country and total weight.

The business logic used is:-

```
IF country =Germany AND weight<=100Kgs
shipment_types = (express_shipment, regular_shipment);
calculate express shipment cost;
 calculate regular shipment cost;

ELSE
shipment_types = (regular_shipment)
calculate regular shipment cost;

/*calculate express shipment cost and days*/
IF total_weight <=20Kgs, cost =10, days=2
ELSE IF total_weight <=50Kgs, cost =30, days=2
ELSE total_weight <=100Kgs, cost =50, days=2

/*calculate regular shipment cost and days*/
IF total_weight <=50Kgs, cost =5, days=5
ELSE IF total_weight <=100Kgs, cost =10, days=5
ELSE IF total_weight <=500Kgs, cost =30, days=5
ELSE cost =50, days=5

/*recalculate cost and days when destination country is not Germany*/
IF country != Germany THEN cost = cost +20, days = 10
```

1.2 Implementation

What approach was used to develop the Shipment Service?

We developed the Shipment Service as a SOAP web service using the top-down approach. In the top-down approach, we first write the WSDL file describing our web service. Next, with the help of available tools like wsimport or WSDL2Java we create the skeleton Java classes and finally we write the business logic in the generated Java classes.

What tools were used to develop the Shipment Service?

We used the following tools for developing the Shipment Service:-

- Eclipse IDE 4.2
- WSo2 Developer Studio 2.1.0
- WSo2 Application Server 5.0.0

Describe the main steps taken to implement the Shipment Service?

Step#1 *To decide whether a database should be used?*
After reading and discussing about the requirements, we decided not to use a database for Shipment Service. Instead, we decided to pass the entire order generated from the web shop to the Shipment Service. Among other details, the order would contain the list of all ordered items including its weight and quantity ordered as well as the recipient's country.

Step#2 *To write the XML schema that defines the data type to be used by the WSDL document*
We used two complex data types – one for sending the order from ACME web shop to shipment service and another to get the response back from the shipment service to the web shop.

Step#3 *To write the WSDL document*
We used the inbuilt function in Eclipse IDE to write the WSDL document to describe the service offered by the shipment service. Our shipment service offers the service called "CalculateShipping".

Step#4 *To generate skeletons of Java classes (stubs) & write the business logic in the generated Java classes*
We used the AXIS2 tool provided by WSo2 Developer Studio to generate the skeletons of the JAVA classes, otherwise known as stubs. Then we implemented the business logic for the "CalculateShipping" service.

Step#5 *To deploy the service in WSo2 Application server*
We created a JAR file and uploaded it in the WSo2 Application Server together with the WSDL file to expose our shipment service.

1.3 Testing

What tool was used to test the Shipment Service?

We used the SOAPUI tool (http://www.soapui.org) to test the developed Shipment Service.

What kinds of tests were performed?

We performed functional testing of the shipment web service. We tested if the implementation of the business logic was correct and the shipment service was providing correct responses to the requests.

Describe briefly the various test cases written?

The availability of different shipment types as well as the cost and delivery time for shipping depends on destination country and total weight. If we further delve into our business logic, we find that there can be nine different cases in which the available shipping types or cost or delivery time might vary. So we wrote nine different test cases to cover the entire range of business logic.

How the test cases were written?

The test cases were written in SOAPUI tool. We used XPath for asserting the responses of our test cases.

Chapter2: Inventory Service

2.1 Overview

<u>What is the role of Inventory Service in ACME web shop implementation?</u>

Inventory Service provides several operations for managing inventory items. It would be called by 3rd parties as well as by web shop application.

<u>What operations were implemented by the Inventory Service?</u>

The following operations were implemented:

- getLowQtyItems()
- search(productName)
- get(productTypeId)
- create(productName)
- checkAvailabilityForSingle(productTypeId)
- checkAvailabilityForBatch(productTypeId[])

2.2 Implementation

<u>What approach was used to develop the Inventory Service?</u>

We developed the Inventory Service as a SOAP web service using the bottom-up approach. In the bottom-up approach, we first create the JAVA classes and later generate the WSDL using the JAVA classes.

<u>What tools were used to develop the Inventory Service?</u>

We used the following tools for developing the Inventory Service:-

- Netbeans IDE 7.2
- Glassfish Application Server 3.1

<u>Describe the main steps taken to implement the Inventory Service?</u>

- Creating of the Inventory Service class
- Enabling persistence
- Annotating with JAX-WS annotations
- Deploying and testing

2.3 Testing

What tool was used to test the Inventory Service?

We used the SOAPUI tool (http://www.soapui.org) to test the developed Inventory Service.

What kinds of tests were performed?

We performed functional testing of the inventory web service. We tested if the implementation of the business logic was correct and the inventory service was providing correct responses to the requests.

Any major point to note regarding Inventory Service testing?

Since inventory service use database we needed to restore database state before and after some test cases. For that we used JDBC requests which execute SQL statements

2.4 Difficulties Faced

Firstly, we decided to use WSo2 application server for deploying Inventory Service. But we encountered problems with JPA running inside WSo2 application server. The first problem was that WSo2 development studio does not include Jar files to the build file. Also WSo2 development studio is not able to integrate with WSo2 application server for the deployment process. Therefore, we used War or Aar files that IDE exports to deploy them to the application server. Because of missing Jar files deployed application was not able to find needed class (we tried to use different libraries - Hibernate and EclipseLink). After we solved this problem by adding Jar files to the War/Aar files manually we encountered problems with configuration of persistence. After that, when everything seemed to be ok, we got an error like “Could not build Entity Manager” and we were not able to solve it.

Also, in Linux version of WSo2 IDE, there is no good support for MySQL. For example, IDE does not see tables inside MySQL database. We tried to deploy our project on Windows machine as well but encountered pretty much the same problems.

Finally, we decided to switch to Netbeans and Glassfish and use Hibernate as JPA implementation because we didn’t have enough time for digging into WSo2 problems. We decided to code our Web Services first, and then, if we are lucky to solve WSo2 problems, to deploy code there.

We decided to use SoapUI for testing and there were some problems with JDBC connectivity. In order to connect SoapUI and database we needed to put mysql connector jar file inside jre/lib/ext directory.

Chapter3: Payment Merchant Service

3.1 Overview

What is the role of Payment Merchant Service in ACME web shop implementation?
The Payment Merchant Service is a third party web service whose role in the Acme Web Shop is to process the payments once the end user has completed the order. The user will be redirected to the landing page of the payment service on completion of the order and can then make the payment, through this payment service.

What is the scope of implementation of Payment *Merchant* Service?

Since the payment service is a third party service, the scope of this task was to implement a REST based web service. The actual internals of how the payment is made is not covered. On the contrary, the implementation serves to highlight the interaction that will take place between ACME web shop and the payment service. We have however implemented basic level validations (such as credit card expiry can't be less than the current year).

3.2 Workflow

Once the user has completed the order, she will be redirected to the landing page of the payment service. The landing page contains a form that requires the user to input the credit card data to make the payment. This page will also contain the amount that the user is about to pay(which needs to be passed to the payment service from the acme webshop). Furthermore, a reference token is also passed along with the amount to track which order this payment belongs to. This form page is independent of the payment service, i.e. to say that it is not part of the REST API. The form however serves the purpose of calling the makePayment() of the API. In case user enters some incorrect information, or leaves a field blank, he will be prompted to re-enter this information and would be redirected back to this form.

3.3 Implementation

What tools were used to develop the Payment Service?
- Eclipse IDE 4.2
- WSo2 Developer Studio 2.1.0
- WSo2 Application Server 5.0.0
- Apache CXF
- ActiveMQ

Describe the main steps taken to implement the Shipment Service?

Step#1	We had to decide on an implementation of JAX-RS. There were two options that we considered for JAX-RS: Jersery and Apache CXF. Since WSo2 supports Apache CXF by default, we went with that.
Step#2	Secondly we had to decide on the type of project we are going to create for this service: Dynamic Web Project or Jax-WS Service Project. Initially we went with Jax-WS Service Project . However, later we wanted to create a landing page for the payment service, the Jax-WS project wouldn't run out of the box. Hence we created a Dynamic Web Project, in which we could have the landing page and the web service in a single project.
Step#3	The next step was to write the actual Rest Web Service that would be called to complete the payment process.
Step#4	Once we had the API ready, it was time to create the form/landing page that the user would use to communicate with our web service. We went with JSP over servlet, since it didn't require any further configurations.
Step#5	The next step was to set up the messaging infrastructure. To do that we had to decide upon the different providers, and we went with ActiveMQ. We set up a queue that we were going to use to communicate between the Payment Merchant Service and the Acme Web Shop.
Step#6	Now that our application was sending messaging to the queue on successful processing of payments, the last step was to create a consumer for these messages. We created a simple java object (POJO) to fetch these messages. Later these will be used to call BPEL processes.
Step#7	Finally we deployed this application on WSO2 server, as a JAX-RS/JAX-WS application.

3.4 Testing

What tool was used to test the Payment Merchant Service?

We used the SOAPUI tool (http://www.soapui.org) to test the developed Payment Merchant Service. To test the JMS messages, we had to integrate soapui with HermesJMS, which is a tool that connects with ActiveMQ.

What kinds of tests were performed?

We performed functional testing of the payment web service. Since we were not implementing any real business logic, we did not need to test it. However, we did do some validation testing to make sure that erroneous data did not result in a valid payment. Secondly, since this task was using a queue for communication we needed to test the messaging part of the service as well.

Describe briefly the various test cases written?

The majority of test cases deal with checking the validity of input data. They are used to check that missing input values should cause an error and not a valid payment. Likewise, an expiry date in the past should cause an error. Lastly, we needed to check whether messages that were posted to the ActiveMQ queue were actually being transferred to the queue.

How the test cases were written?

The test cases were written in SOAPUI tool. HermesJMS were used as a bridge between ActiveMQ and SoupUI.

3.5 Challenges

The first major challenge in developing the REST web service was to work out how to integrate the different components of the application together. That is to say how to set up the testing frame work, how to interact between the queue and the testing framework and between the application and the queue. Setting up CXF correctly itself turned out to be a bit of a challenge. Once the application was up, creating a new servlet in the same JAX-WS service was not working correctly due to some routing error. To solve this issue we created a Dynamic Web Project in its place. Regarding the creation of the EJB, we realized that it was something that could not be deployed on the WSo2 server. Hence we went with just creating a Java Application for to fetch the messages from the queue.

Chapter4: Service Proxy

4.1 Overview

What is the role of Service Proxy?

The Service Proxy provides a single endpoint for all three services implemented earlier namely Shipment Service, Inventory Service and Payment Merchant Service. This is currently being use to test all the three services.

4.2 Implementation

How the Service Proxy was implemented?

We used WSo2 ESB and we created custom proxy with conditional routing for our web services. Now the test cases invoke this single proxy service and the request are routed according to the header fields.