Creating a RESTful Web Service for IMS-Transaction

Author: Ivy Ho (ivyho@ca.ibm.com), Advisory Software Engineer, J2C Team Lead, IBM
Editor: Virginia Lovering (lovering@ca.ibm.com), Information Specialist, IBM

Introduction

The Java™ API for RESTful Web Services (JAX-RS), also known as JSR-311, is a programming model that allows you to create Representational State Transfer (REST) services quickly. In this tutorial, we will guide you through the steps to create a RESTful Web Services to retrieve phone book information based on a last name from the IMS server. Any JAX-RS v1.1 implementation can be used, but for this tutorial we will be using the IBM implementation which is installed when you install a v7 WebSphere Test Environment.

Software Requirements

- Rational Application Developer V7.5.5.x or later.
- Optional Features required: Java EE Connector (J2C) Development Tools
- WebSphere Application Server V7.0 Test Environment
- Web 2.0 V1.0.1 feature pack

Other Requirements

The phone book COBOL book sample is installed on the IMS system.
Learning Objectives

In this tutorial, you will learn to:

- Create a J2C application using the J2C Bean wizard for processing a IMS transaction.
- Enable the JAX-RS in the J2C bean project for a IMS transaction.
- Deploy and Test the JAX-RS web service to retrieve phone book addresses.

Create a J2C bean application for an IMS transaction

The following describes how to use the J2C Java™ bean wizard to build a JAX-RS enabled dynamic web application that processes an IMS IM® transaction and returns a phone book record. We are using the non-managed connection and we are connecting directly to the IMS server.
**Select a Resource Adapter**

1. Import Resource Adapter and Create a connector project.
2. Select File -> J2C -> J2C Bean.
3. Select the IMS TM Resource adapter:

   ![Screen shot of Resource Adapter Selection window](image)

   3. Specify the connector project name.
4. Select a target server, and click Next:

   ![Screen shot of Connector Import window](image)
5. On the Adapter Style Page, click Outbound:

![Adapter Style Page](image1.png)

6. On the Scenario Selection page, click IMS COBOL, PL/1 or C-based applications:

![Scenario Selection Page](image2.png)
7. On the Connection Properties page
   a. clear Managed Connection
   b. select Non-Managed Connection
   c. and enter the connection information:

![Connection Properties](image1)

**Create a JAX-RS enabled Dynamic Web Project**

The following steps describe how to create the J2C bean in a JAX_RS enabled dynamic web project. The steps are very similar to the steps to create a typical J2C bean. The only difference is the extra configuration when you create the dynamic Web Project.

1. Enter Web Project Name and click New:

![New J2C Bean](image2)
2. On the New Source Project Creation page, select Web Project and click Next:

3. On the New Dynamic Web Project page,
   a. Select the target runtime server.
   b. Accept the default web module version.
   c. Under Configuration, select IBM JAX-RS Configuration. This will add the JAX-RS 1.1, Ajax Proxy, and Server-side technologies facets. The facet adds the library, servlet information, and support for JAX-RS annotations processing and JAX-RS quick-fixes.
   d. Select add Project to an EAR in the EAR membership, and click Next:
4 On the Web Module page, click Generate web.xml deployment descriptor, and click Next:
5. On the Web 2.0 Server-Side Technologies page, accept the default, and click Next:

![Web 2.0 Server-Side Technologies](image)

6. On the JAX-RS Capabilities page, in the Library field select the library for your server version, for example IBM WebSphere JAX-RS Library for WAS 7. Click Include library with this application and select to include it as a Shared Library. Click Finish:

![JAX-RS Capabilities](image)
7. On the J2C bean Output Properties page, specify Package name, Interface class name and Implementation name for the J2C bean. In the Implementation name field, by default, the name appears as `<interface name>Impl`.

8. On the Add Java method page, beside the Input type field, click New:
9. On the Data Import page, click Browse and select the COBOL copy book. Click Next:

![Data Import Page](image1)

10. In the Saving properties page, select the following values for input type:
   a. Select Default for Generation Style.
   b. Click Browse beside the Project Name and choose the Web project IMSWebPhoneBook.
   c. In the Package Name field, type sample.ims.data.
   d. In the Class Name field, accept the default INPUTMSG. Click Finish.

![Saving Properties Page](image2)
11. In the COBOL Importer page, click Show Advanced. Select these options:

   a. Select the options in the following table:

   **COBOL Importer Parameter Settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform Name</td>
<td>Z/OS</td>
</tr>
<tr>
<td>Codepage</td>
<td>IBM-037</td>
</tr>
<tr>
<td>Floating point format name</td>
<td>IBM Hexadecimal</td>
</tr>
<tr>
<td>External decimal sign</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Endian name</td>
<td>Big</td>
</tr>
<tr>
<td>Remote integer endian name</td>
<td>Big</td>
</tr>
<tr>
<td>Quote name</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>Trunc name</td>
<td>STD</td>
</tr>
<tr>
<td>Nsymbol name</td>
<td>DBCS</td>
</tr>
</tbody>
</table>

   b. Click Query to load the data.

   c. A list of data structures from the Ex01.cbl file is shown. Select OUTPUT-MSG in the Data structures field, and click Next:
12. In the Saving Properties page,
   a. Select Default for Generation Style.
   b. Click Browse beside the Project Name and choose the Web project IMSWebPhoneBook.
   c. In the Package Name field, type sample.ims.data.
   d. In the Class Name field, accept the default OUTPUTMSG. Click Finish.
Java Method
Specify the name of the new method.

Name: *getPhoneBook
Input type: INPUTMSG

Use the input type for output
Output type: OUTPUTMSG

User defined InteractionSpec or ConnectionSpec parameters

Method summary:
getPhoneBook (INPUTMSG arg) : OUTPUTMSG
12. Click Finish to complete the J2C bean creation and generate all the artifacts.

You will see the following artifacts in the Web Project:

**J2C bean interface**
phone_book.java

**J2C bean Implementation**
phone_bookImpl.java

**Data Binding files**
INPUTMSG.java
OUTPUTMSG.java
Introspecting the phone book Imp.java

The connection information is generated as values for the @j2c.managedConnectionFactory doclets

```java
import javax.resource.cci.ResourceAdapterMetaData;

/**<*
 * @j2c.managedConnectionFactory class="com.ibm.connector2.ims.ico.IMSMangedConnectionFactory"
 * @j2c.managedConnectionFactory-property name="dataStoreName" value="IMS1"
 * @j2c.managedConnectionFactory-property name="hostName" value="myIMSHost"
 * @j2c.managedConnectionFactory-property name="portNumber" value="9999"
 * @j2c.connectionSpec class="com.ibm.connector2.ims.ico.IMSConnectionSpec"
 * @generated
 */

public class PhoneBookImp implements sample.ims.PhoneBook {
```

The connection values will be used in the initializeBinding method to connection to the IMS system directly.
invokedInteractionSpec = is;
    }
    return;
  
  /**
   * @generated
   */
  protected void initializeBinding() throws ResourceException {
    ConnectionFactory cf = null;
    com.ibm.connector2.imc.IMSClusterConnectionPool cf = new com.ibm.connector2.imc.IMSClusterConnectionPool();
    cf.setPSName("IMS");
    cf.setPSName("IMSHost");
    cf.setPortNumber(Integer.valueOf("9999"));
    catch (Exception e) {
      throw new ResourceException(e.getMessage());
    }
    cf = (ConnectionFactory) mcf.createConnectionFactory();
    setConnectionFactory(cf);
  
  typeLevelConnectionSpec = new com.ibm.connector2.imc.IMSClusterConnectionSpec();
  
  /**
   * @generated
   */
  public PhoneBookImpl() throws ResourceException {
    initializeBinding();
  
  
    }
Create Sample Applications

Create the Sample PhonebookRS.java and PhonebookRSApplication.java into the sample.ims package of project IMSWebRSProject.

PhonebookRSApplication.java

```java
package sample.ims;

import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.core.Application;

public class PhoneBookRSApplication extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> classes = new HashSet<Class<?>>();
        classes.add(PhoneBookRS.class);
        return classes;
    }
}
```

PhoneBookRS.java -

Define a list which is an array of LAST names.
The Last Name is the key to the PhoneBook record
In the getList method, it will invoke the getPhoneBook method in the J2C bean to retrieve the phone book record from the backend IMS system.
The output phonebook record contains extensions, zips, name1 and name2.

PhonebookRS.java

```java
package sample.ims;
```
import javax.resource.ResourceException;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rsPathParam;
import javax.ws.rs.Produces;
import sample.ims.data.*;

/**
 * A sample resource that provides access to
 * server configuration properties
 */
@Path(value="/addresses")
public class PhoneBookRS {
    public PhoneBookRS() {
    }

    private static String[] list = new String[] {
            "LAST1",
            "LAST2",
            "LAST3"
    };

    @GET
    @Produces(value="text/plain")
    @Path(value="/{id}")
    public String getPropety(@PathParam("id") int id) throws ResourceException {

        INPUTMSG input = new INPUTMSG();
        input.setIn__ll((short)59);
        input.setIn__zz((short)0);
        input.setIn__trcd("IVTNO");
        input.setIn__cmd("DISPLAY1");
        input.setIn__name1(list[id]);
        input.setIn__name2(""");
        input.setIn__extn(""");
        input.setIn__zip(""");

        PhoneBookImpl proxy = new PhoneBookImpl();
        OUTPUTMSG output = proxy.getPhoneBook(input);
        System.out.println("\nName1:"+output.getOut__name1()+"\nName2:"+output.getOut__name2()+"\nextension:"+output.getOut__extn()+"\nzip:"+output.getOut__zip());

        buffer.append("\nName1:"+output.getOut__name1()+"\nName2:"+output.getOut__name2()+"\nextension:"+output.getOut__extn()+"\nzip:"+output.getOut__zip());

        return buffer.toString();
    }

    @GET
    @Produces(value="text/plain")
    public String getList() throws ResourceException {
        StringBuffer buffer = new StringBuffer();

        for (int i = 0; i < list.length; ++i) {
            System.out.println("i="+i+" Name1="+list[i]);
            if (i != 0) {
                buffer.append("   ");
                buffer.append("   ");
            }
            buffer.append(" ");
            buffer.append(" ");
        }

        INPUTMSG input = new INPUTMSG();
        input.setIn__ll((short)59);
        input.setIn__zz((short)0);
        input.setIn__trcd("IVTNO");
        input.setIn__cmd("DISPLAY1");
        input.setIn__name1(list[i]);
        input.setIn__name2(""");
        input.setIn__extn(""");
        input.setIn__zip(""");

        PhoneBookImpl proxy = new PhoneBookImpl();
        OUTPUTMSG output = proxy.getPhoneBook(input);
        System.out.println("\nName1:"+output.getOut__name1()+"\nName2:"+output.getOut__name2()+"\nextension:"+output.getOut__extn()+"\nzip:"+output.getOut__zip());

        buffer.append("\n Name1:"+output.getOut__name1()+"\nName2:"+output.getOut__name2()+" \nextension:"+output.getOut__extn()+"\nzip:"+output.getOut__zip());

        return buffer.toString();
    }
}
System.out.println("getProperty.id="+id);
StringBuffer buffer = new StringBuffer();

if (id > -1 && id < list.length ) {
    INPUTMSG input = new INPUTMSG();
    input.setIn__name1(list[id]);
    input.setIn__ll((short)59);
    input.setIn__zz((short)0);
    input.setIn__trcd("IVTN0");
    input.setIn__cmd("DISPLAY1");
    input.setIn__name2("");
    input.setIn__extn("");
    input.setIn__zip(""薮");
    PhoneBookImpl proxy = new PhoneBookImpl();
    OUTPUTMSG output = proxy.getPhoneBook(input);
    buffer.append("\n Name1:"+output.getOut__name1()+"\n Name2:"+output.getOut__name2()+"\n extension:"+output.getOut__extn()+"\n zip:"+output.getOut__zip());
    return buffer.toString();
} else {
    return "Customer Number Not Found";
}

@GET
@Produces(value="text/html")
@Path(value="html")
public String getHTMLList()
{
    return "<html><body><p><b>Hello</b></body></html>";
}

**Inspect Project Path**

Inspect the properties for the IMSWebRSProject

In the Project Facets, you will see that JAX-RS (REST Web Services) is selected
In the Project References, ensure that the connector project and the EAR project is being selected.
Configure deployment descriptor web.xml

1. Open WebContent/WEB-INF/web.xml. In the Design view, select the Servlet JAX-RS Servlet and click Add and add an Initialization parameter to the JAX-RS servlet,

Enter `javax.ws.rs.Application` as param name
and `sample.ims.phone bookRSApplication` as param value.

Save the web.xml.
2. Right click on the EAR project and select Properties. Ensure that the Project References are referencing the connector project and the JAX-RS dynamic web project.

3. Now with everything configured and setup, we are ready to deploy and test the J2C JAX-RS web service. Right click the WebSphere Application Server and select Properties. Find out the HTTP port number, in this sample, it is 9080. The port number will be used when you invoke the J2C JAX-RS web service.
Test the JAX-RS Web service

1. In the Servers view, right-click your server and select Add and Remove Projects, and add the JAX-RS EAR to the server. Restart the server.

2. Open a Web browser and enter the following URL: http://localhost:<default_host_port>/IMSWebRSProject/jaxrs/addresses/<address index> The address assigned to that index value will display.

3. To retrieve all addresses in the Address Book application, enter the following URL: http://localhost:<default_host_port>/IMSWebRSProject/jaxrs/addresses
By specifying the second index to the list that contains an array of last name, it will return the second phone book record:

Name1: LAST1
Name2: FIRST2
extension: 8-111-2222
zip: D02/R02
If you do not specify the index, all the phone book records will be displayed:

```json
{
    Name1:LAST1
    Name2:SHIRLEY
    extension:111
    zip:111
    Name1:LAST2
    Name2:FIRST2
    extension:0-111-2222
    zip:002/R02
    Name1:LAST3
    Name2:FIRST3
    extension:0-111-3333
    zip:003/R03
}
```