Oracle Open World 2008 - Hands On Lab

JD Edwards EnterpriseOne
Business Service Development

Table of Contents

1 TUTORIAL OVERVIEW 3
2 CREATING A BUSINESS SERVICE 4
2.1 Startup 4
2.2 Add a Published Business Service Object 5
2.3 Launch JDeveloper from OMW 8
2.4 Add Input Value Object for Published Business Service 11
2.5 Add Output Value Object for Published Business Service 15
2.6 Creating the Published Business Service 20
2.7 Adding a Business Function Call 24
3 TESTING THE PUBLISHED BUSINESS SERVICE 29

1 Tutorial **Overview**

JD Edwards EnterpriseOne Tools release 8.97 has added support for native web service development. This tools release provides the ability to aggregate JDE E1 Business Functions and Database Operations into a “Business Service” that can then be exposed as a web service. This lab demonstrates Business Services development capabilities within the JDEdwards EnterpriseOne product suite

In this tutorial you will create the necessary objects to support a new “Published” Business Service called “CustomAddressManager” that will have an operation that will retrieve and format a customers mailing address.

In this lab you will:

- Create new Business Service records in the Object Management WorkBench (OMW)
- Launch Jdev to edit and test the Business Service.
- Create Input and Output Value Objects using the provided Jdev wizards.
- Create Published Business Service using the provided Jdev wizards.
- Modify the Published Business Service to call a Business Function to retrieve a customer’s formatted address.
- Test the Published Business Service as a web service.

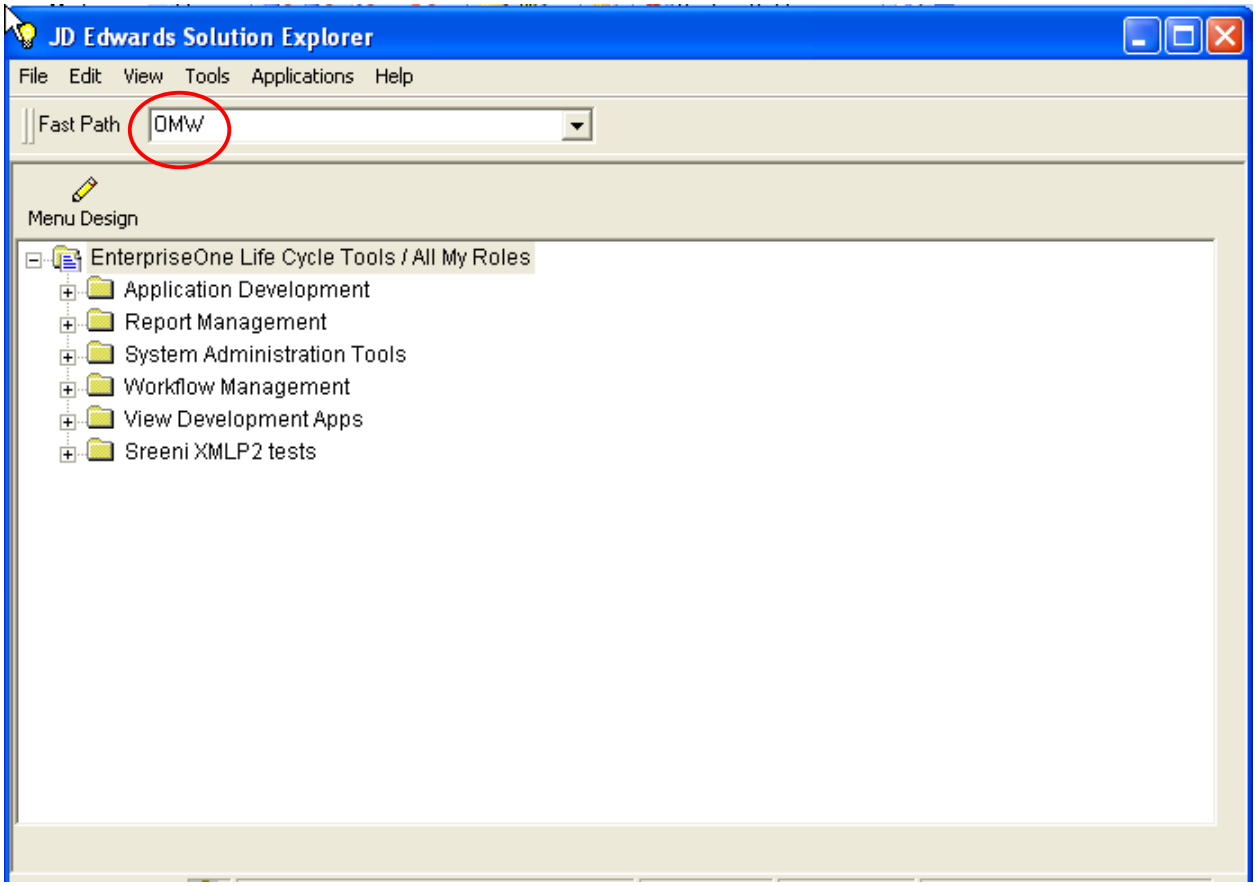
2 Creating a Business Service

2.1 Startup

1. Launch Solution Explorer for the Desktop Icon.

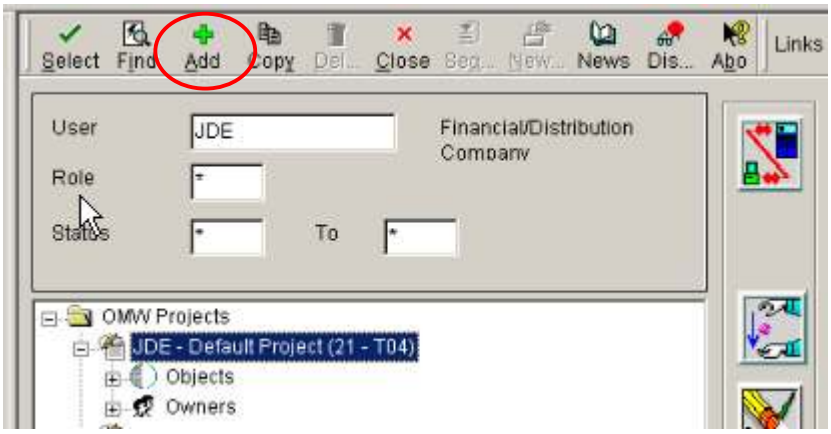
2. Credentials are User:JDE: Password:JDE

3. Type in “OMW” in Fast Path to Launch the “Object Management WorkBench:

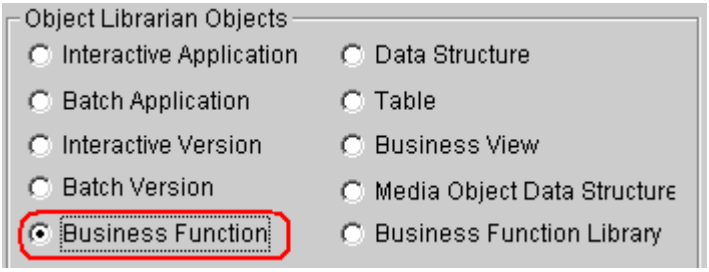


2.2 Add a Published Business Service Object

1. Push the **Find** Button, Highlight the default OMW project (JDE), and click on the Add Button



2. Select the **Business Function** radio button and click **OK**.



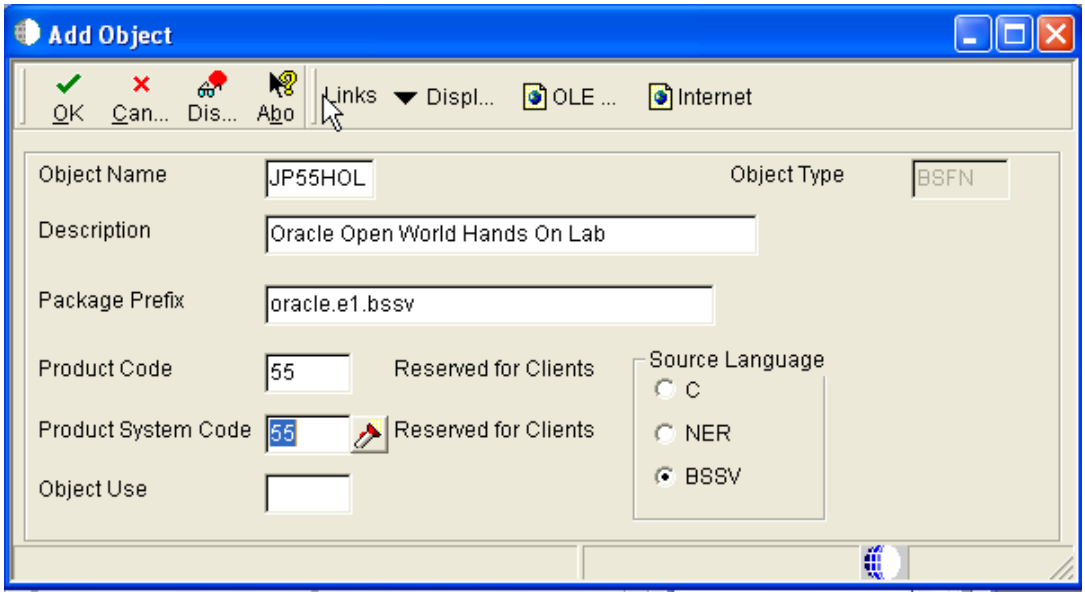
3. On the *Add Object* form, complete the following fields and click the **OK** button:

Object Name: **JP55HOL**
Description: **Oracle Open World Hands On Lab**
Product Code: **55**
System Code: **55**
Object Use: **330**

Source Language: **BSSV**

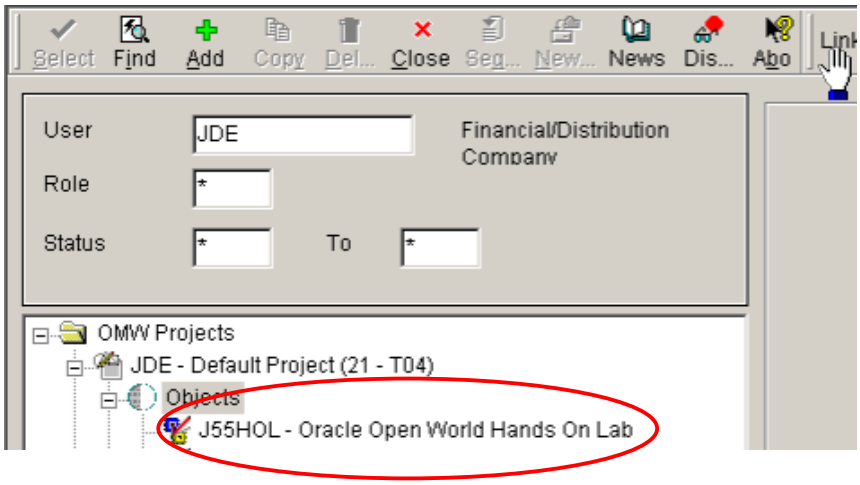
Package Prefix: **oracle.e1.bssv** (**Note:** Use the flash light when the cursor is in the "Package Prefix" field to select this value)

NOTE: ***The Package Prefix field is enabled when you select BSSV as the Source***



4. On the Business Function design form click **OK** to return to OMW.

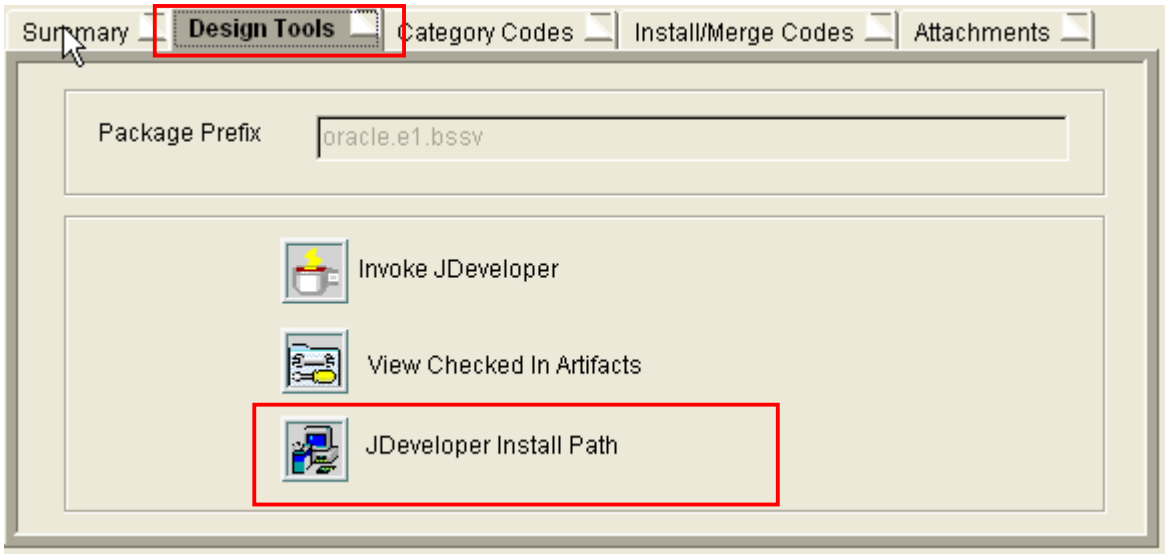
5. You should see the Published Business Service under the **Objects** node of your project.



2.3 Launch JDeveloper from OMW

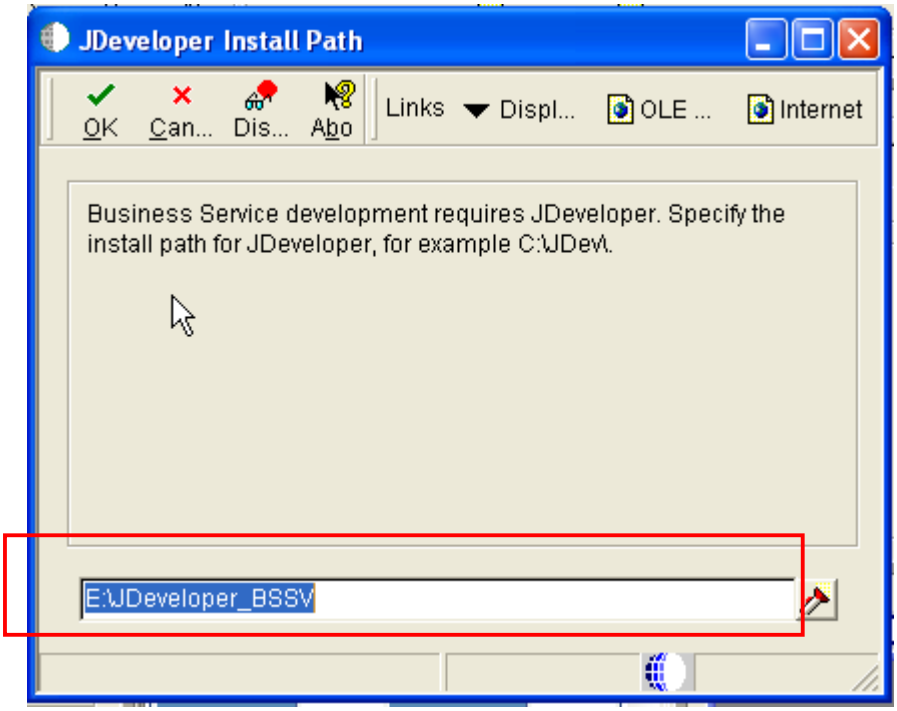
1. In OMW, Select the Published Business Service (**JP55HOL**) and click on the **Design** button in the center column.

2. Click on the **Design Tools** tab and then click on **JDeveloper Install Path**.



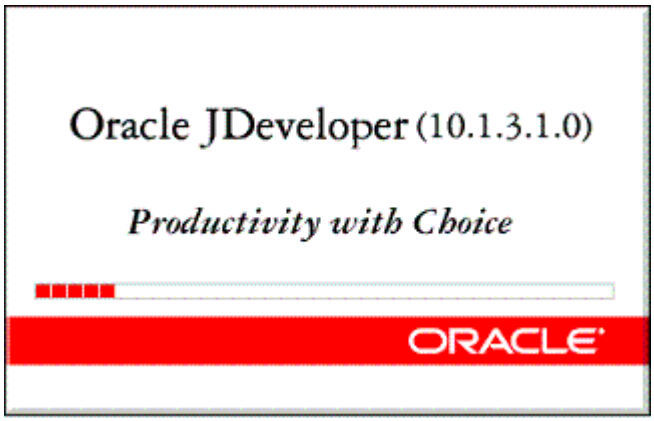
3. Ensure that the Jdeveloper Install Path is set to e**:\\Jdeveloper_BSSV** and press **OK.**

   (Note: use the icon to select the Jdev install path)



4. Click on the **Design Tools** tab and then click on **Invoke JDeveloper**.

5. This will launch Oracle JDeveloper



6. If you are prompted to migrate for older version of Jdev – Select **No**:

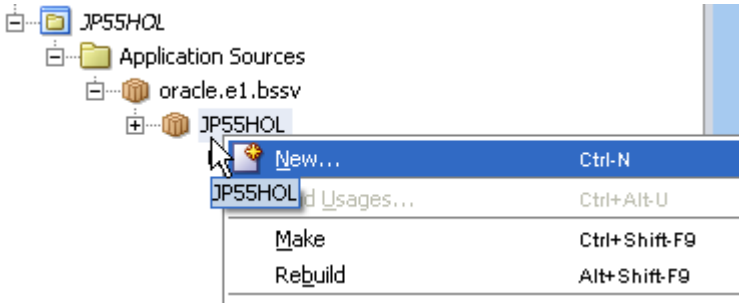
7. If you are prompted to save files, select **OK**.

Note: Your screen may not be exactly the same as the screenshot above.

8. When JDeveloper opens, you will see a **JDeveloper project** for the Published Business Service (**JP55HOL**).

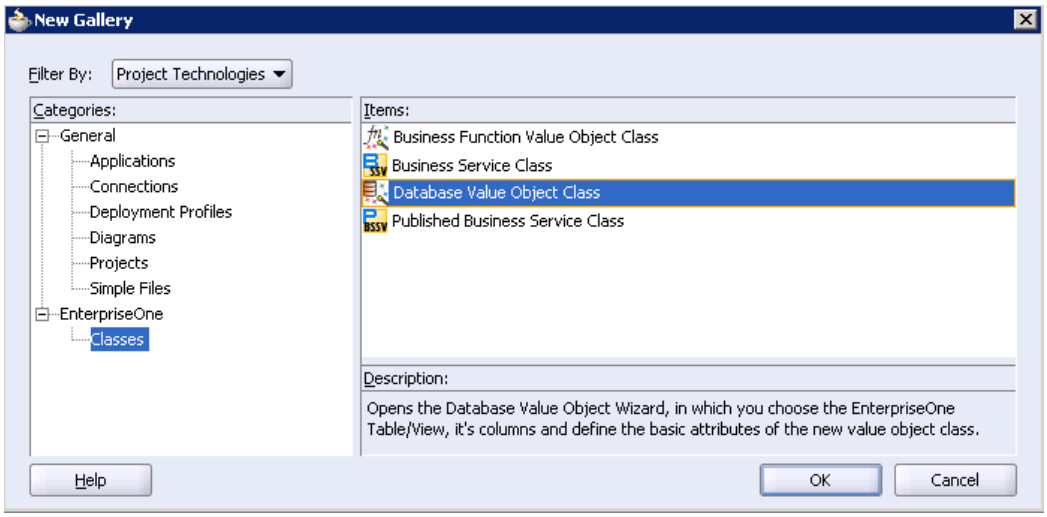
2.4 Add Input Value Object for Published Business Service

1. Select the Business Service (**JP55HOL**) and right click. Select **New** from the menu.

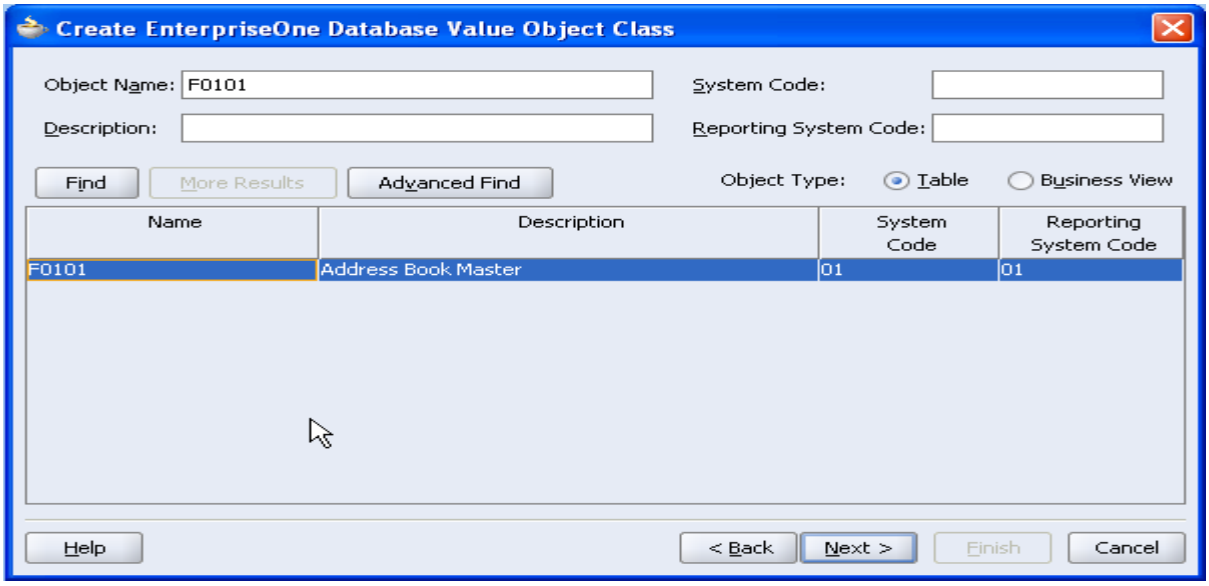


2. Under Categories, select **EnterpriseOne->Classes**.

   Select **Database Value Object Class** and press the **OK** button.

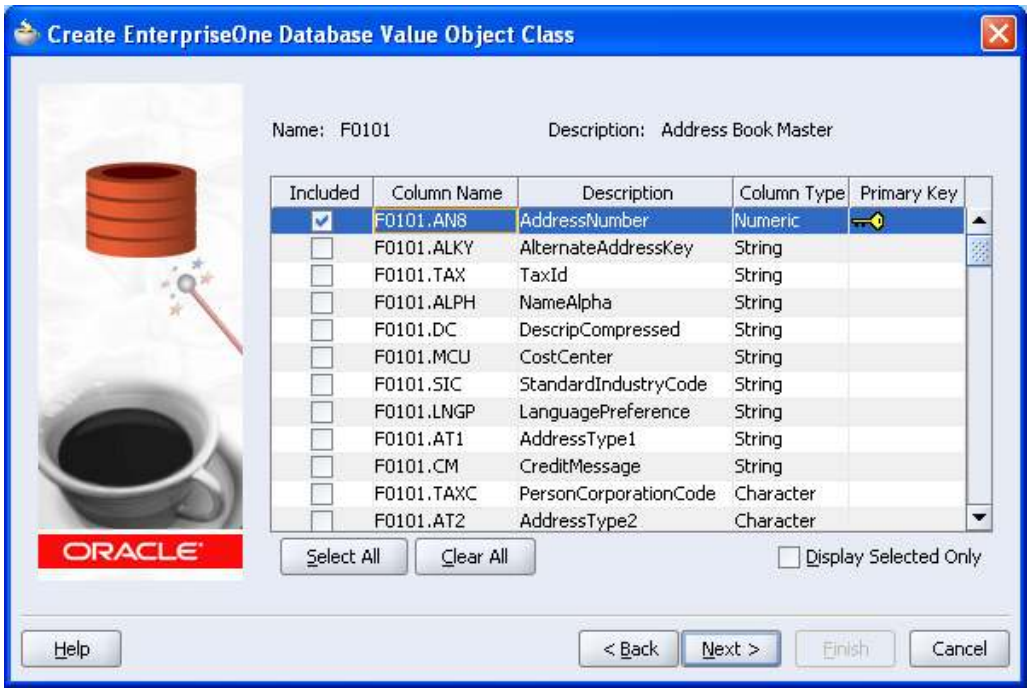


3. In the *Create EnterpriseOne Database Value Object* window, enter **F0101** as the Object Name and press the **Find** button. Select the table from the result and press the **Next** button.

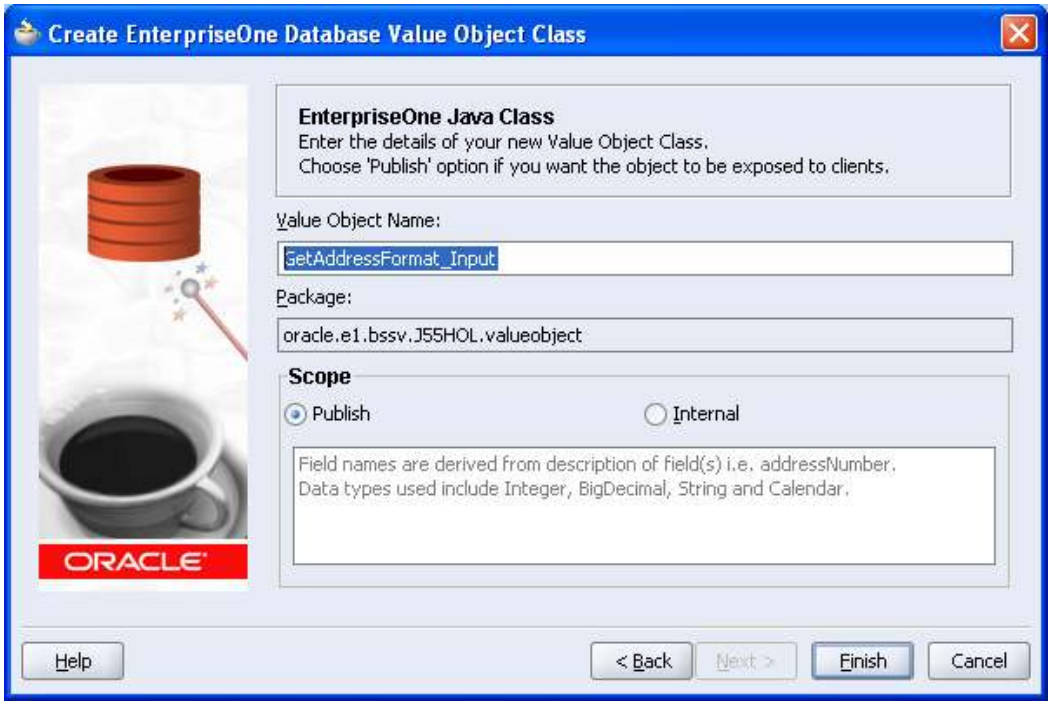


4. Select the following fields and press the **Next** button.

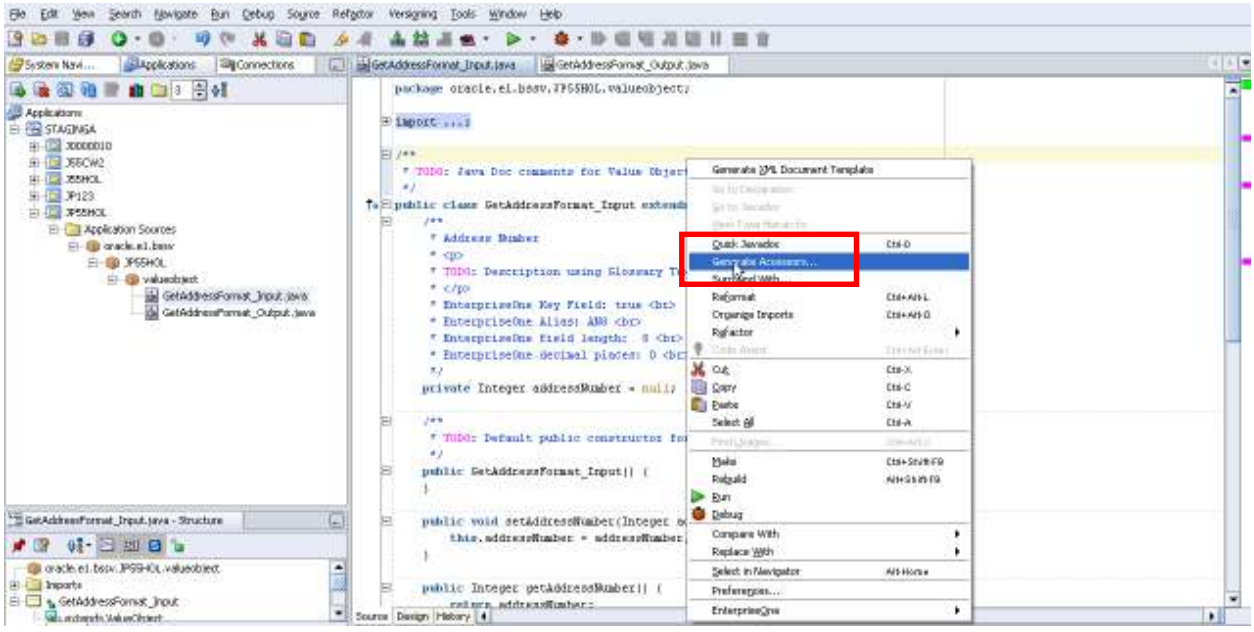
Column Name	Description
F0101.AN8	Address Number



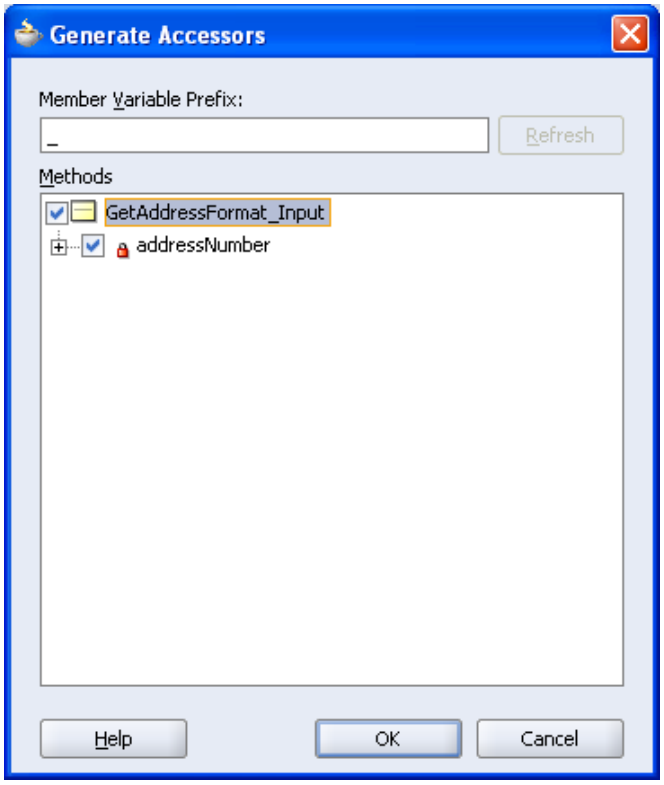
5. Set the *Value Object Name* to **GetAddressFormat_Input** and the *Scope* as **Publish**. Click the **Finish** button to create the input Value Object.



6. The GetAddressFormat_Input Value Object should now be displayed in JDeveloper. Right-click on any whitespace in the value object class and click on **Generate Accessors**.



7. Select all the fields and press the **OK** button.



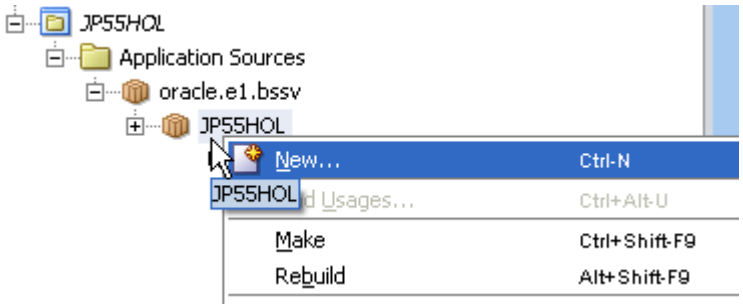
8. The **get** and **set** methods for the value object elements have been created. You can scroll down to view these methods.

```
    /**
     * TODO: Default public constructor for instantiating: GetAddressFormat_Input
     */
    public GetAddressFormat_Input() {
    }

    public void setAddressNumber(Integer addressNumber) {
        this.addressNumber = addressNumber;
    }

    public Integer getAddressNumber() {
        return addressNumber;
    }
}
```

9. Save the File.

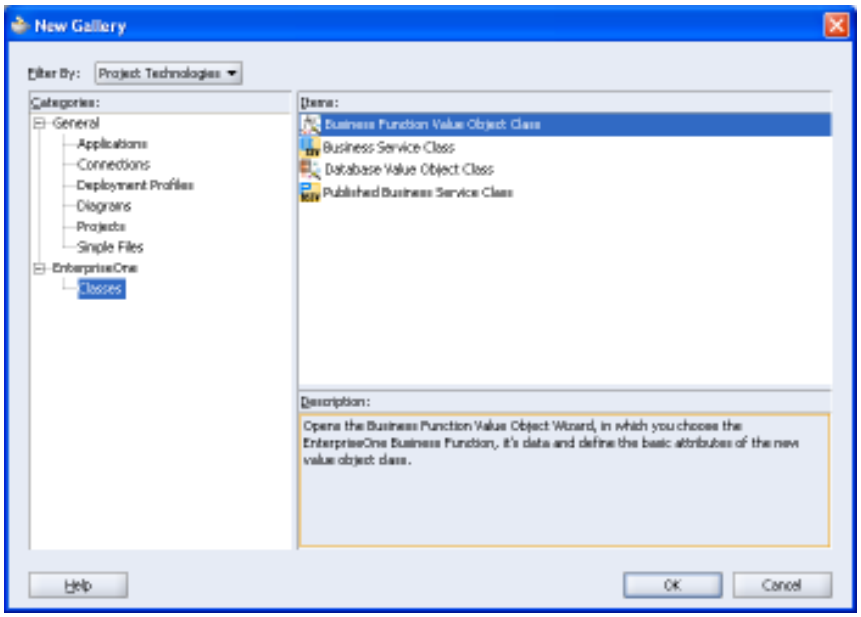
2.5 Add Output Value Object for Published Business Service

1. Select the Business Service (**JP55HOL**) and right click. Select **New** from the menu.

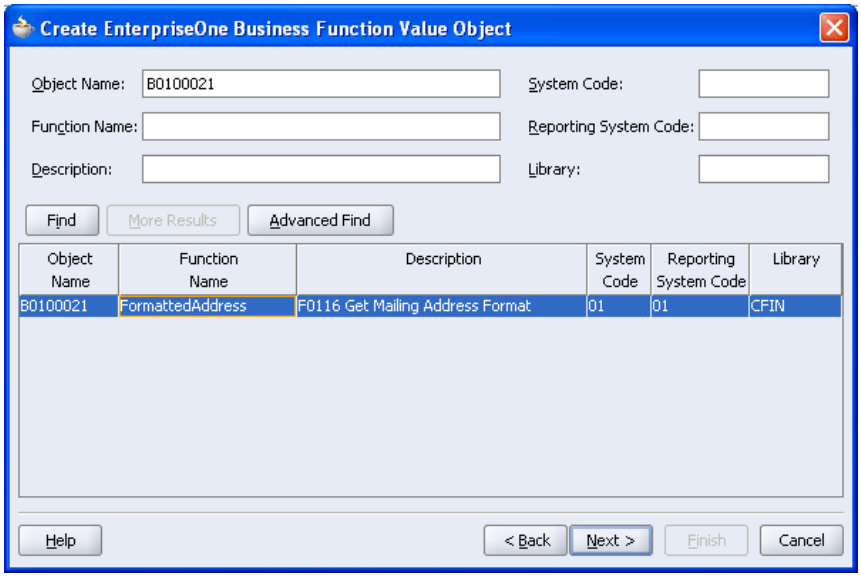


2. Under Categories, select **EnterpriseOne->Classes**.

   Select **Business Function Value Object Class** and press the **OK** button.

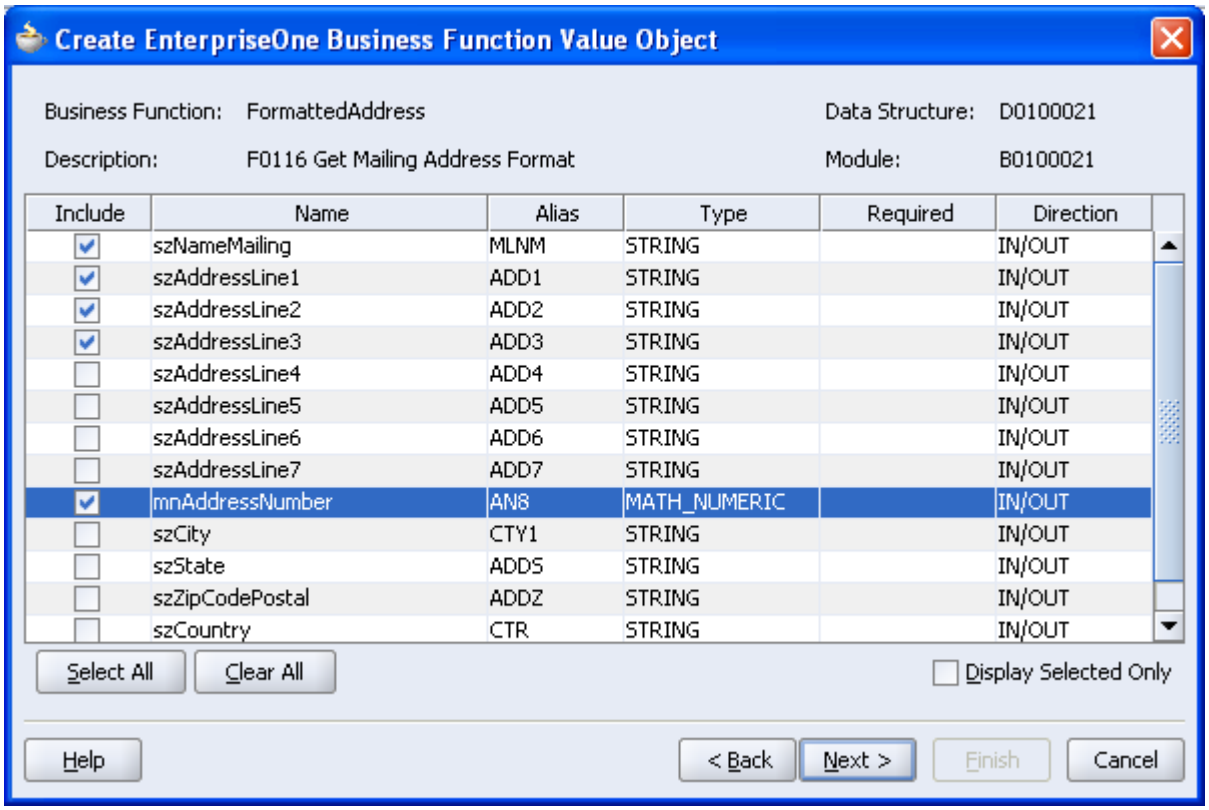


3. In the *Create EnterpriseOne Business Function Value Object* window, enter **B0100021** as the Object Name and press the **Find** button. Select the Business Function from the result and press the **Next** button.

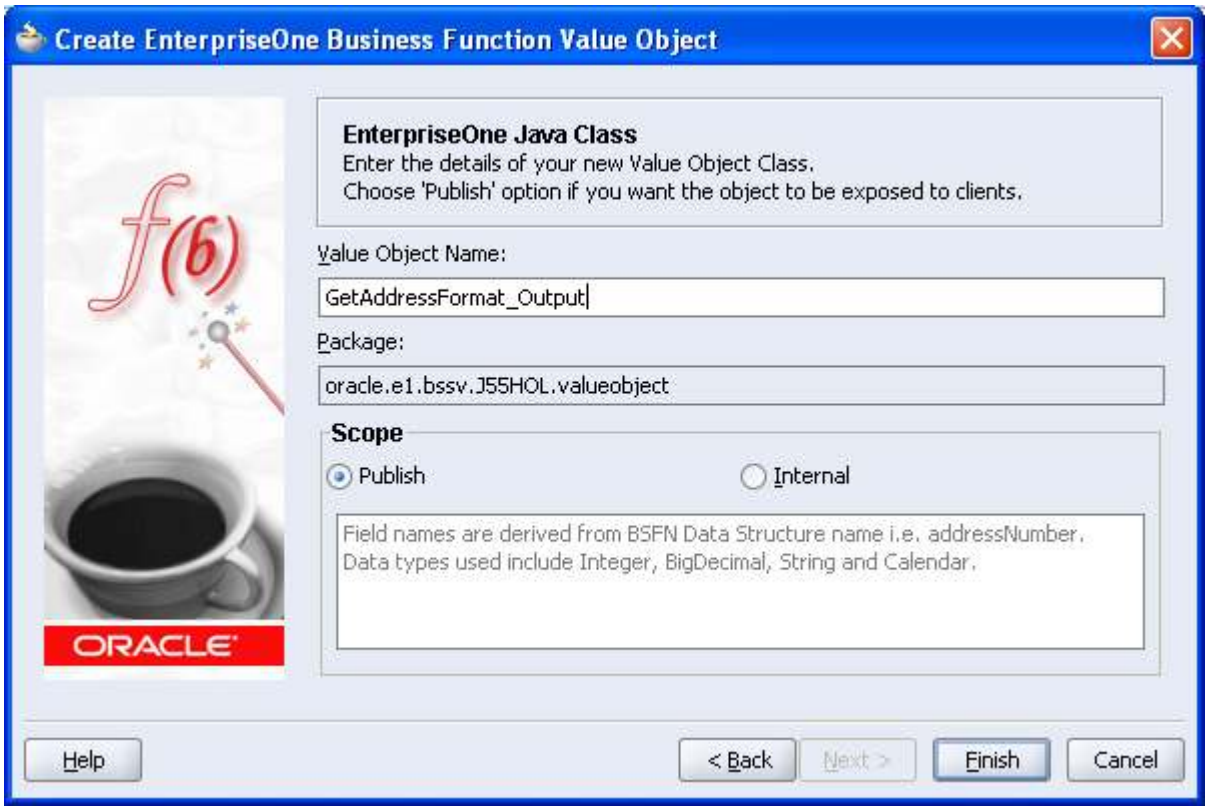


4. Select the following fields

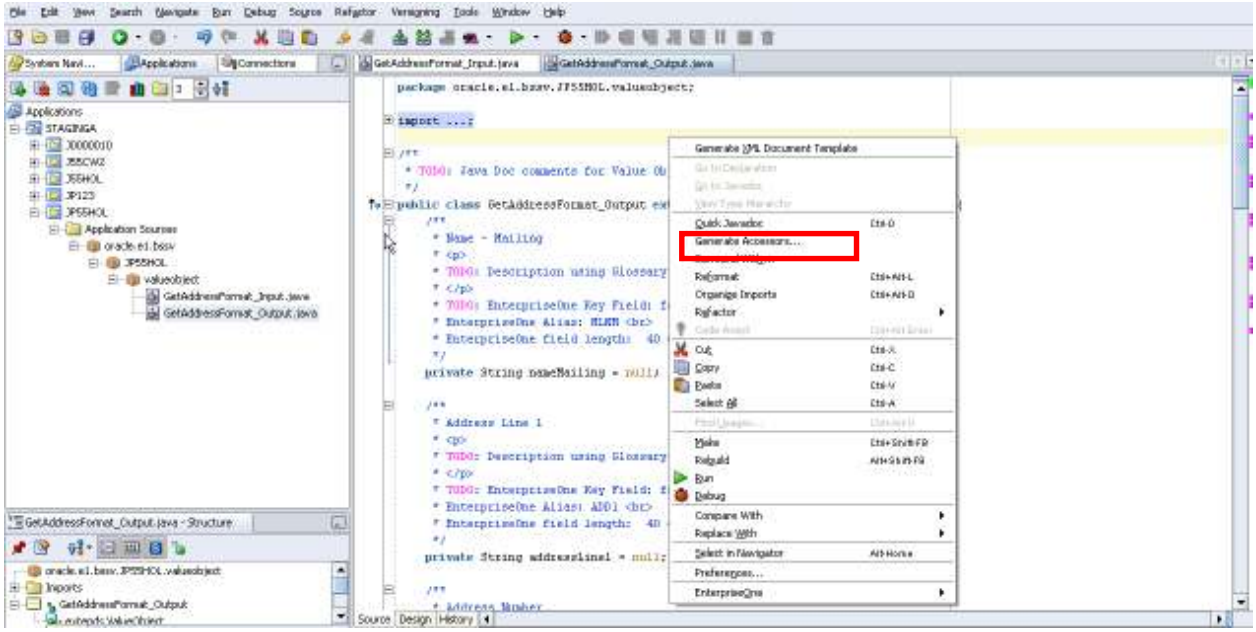
Column Name	Alias
szNameMailing	MLNM
szAddressLine1	ADD1
szAddressLine2	ADD2
szAddressLine3	ADD3
mnAddressNumber	AN8



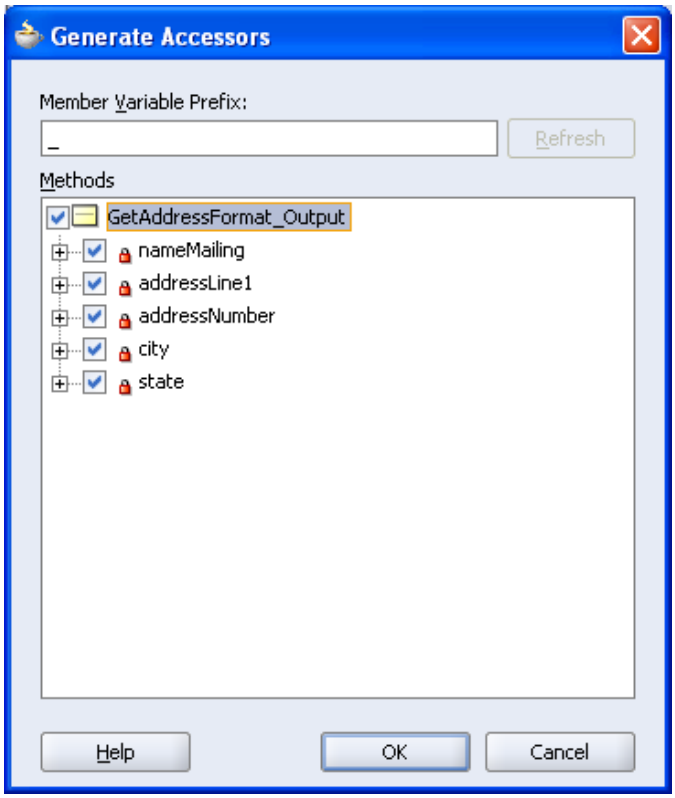
5. Set the *Value Object Name* to **GetAddressFormat_Output** and the *Scope* as **Publish**. Click the **Finish** button to create the output Value Object.



6. The GetAddressFormat_Output Value Object should now be displayed in JDeveloper. Right-click on any whitespace in the value object class and click on **Generate Accessors**.



7. Select all the fields and press the **OK** button.



8. The **get** and **set** methods for the value object elements have been created. You can scroll down to view these methods. Example code is shown below:

```
    /**
     * TODO: Default public constructor for instantiating: GetAddressFormat_Output
     */
    public GetAddressFormat_Output() {
    }

    public void setNameMailing(String nameMailing) {
        this.nameMailing = nameMailing;
    }

    public String getNameMailing() {
        return nameMailing;
    }

    public void setAddressLine1(String addressLine1) {
        this.addressLine1 = addressLine1;
    }

    public String getAddressLine1() {
        return addressLine1;
    }

    public void setAddressNumber(Integer addressNumber) {
        this.addressNumber = addressNumber;
    }
```

9. At the beginning of the code, add the Import statements highlighted in ***bold*** below:

```
import java.io.Serializable;
import oracle.e1.bssvfoundation.base.ValueObject;
import oracle.e1.bssvfoundation.base.MessageValueObject;
import oracle.e1.bssvfoundation.util.E1MessageList;

/**
 * TODO: Java Doc comments for Value Object here
 */
```

10. Make the following change to the :

public class GetAddressFormat_Output extends *ValueObject* implements Serializable {
- to -
public class GetAddressFormat_Output extends ***MessageValueObject*** implements Serializable {

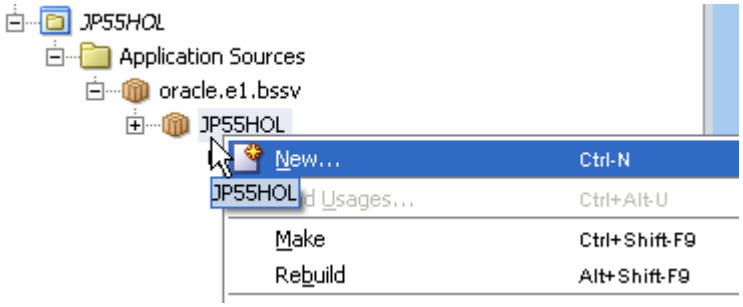
After these changes the code should look like the below:

```
package oracle.el.bssv.JP55HOL.valueobject;

import java.io.Serializable;

import oracle.el.bssvfoundation.base.ValueObject;
import oracle.el.bssvfoundation.base.MessageValueObject;
import oracle.el.bssvfoundation.util.ElMessageList;


/**
 * TODO: Java Doc comments for Value Object here
 */
public class GetAddressFormat_Output extends MessageValueObject implements Serializable {
    /**
     * Name - Mailing
```

11. Save the File.

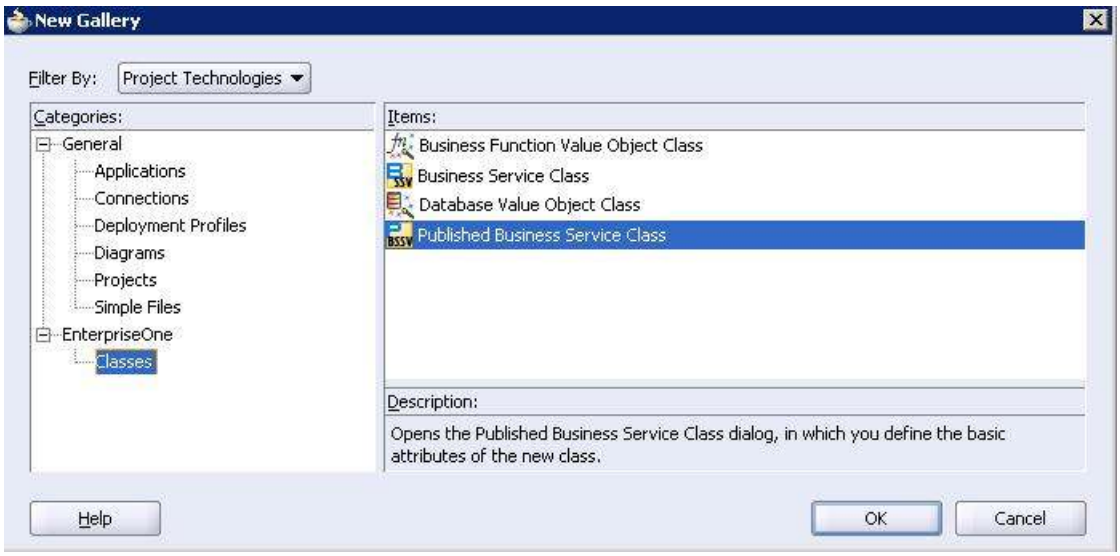
2.6 Creating the Published Business Service

1. Select the Business Service (**JP55HOL**) and right click. Select **New** from the menu.



2. Under Categories, select **EnterpriseOne->Classes**.

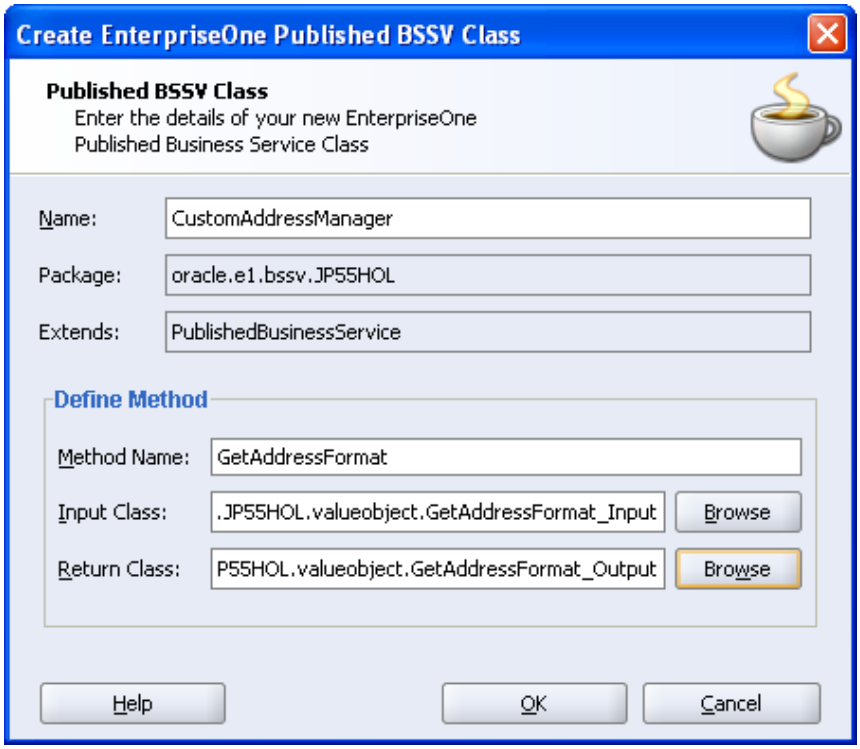
   Select **Published Business Service Class** and press the **OK** button. This will create the template for the Published Business Service class.



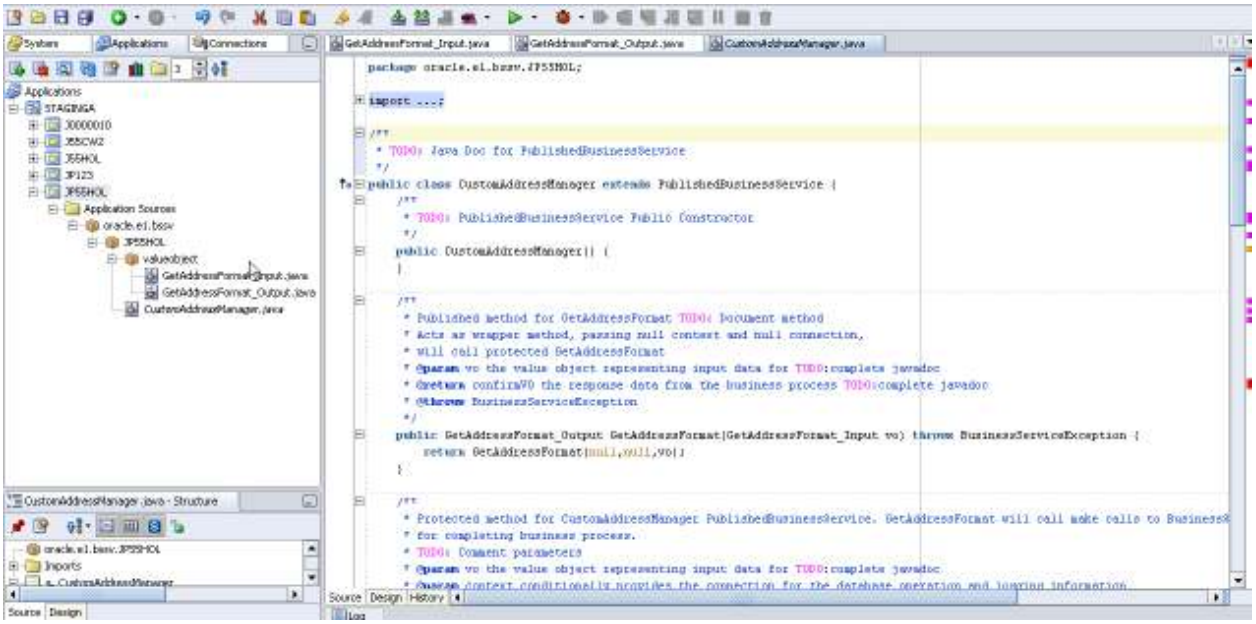
3. In the *Create EnterpriseOne Published BSSV Class* window, enter the following information

   Name: **CustomAddressManager**
   Method Name: **GetAddressFormat**
   Input Class: **oracle.e1.bssv.JP550009.valueobject.GetAddressFormat_Input**
   Output Class: **oracle.e1.bssv.JP550009.valueobject.GetAddressFormat_Output**

   (**Note**: Use the **Browse** button to select the Input and Output Class)



4. Press the **OK** button. The Published Business Service class template is created.



5. Add the following Import Statement at the beginning of the code:

   import oracle.e1.bssvfoundation.util.MathNumeric;

   After inserting the code should look like the below:

```
package oracle.el.bssv.JP55H0L;

import oracle.el.bssv.JP55H0L.valueobject.GetAddressFormat_Input;
import oracle.el.bssv.JP55H0L.valueobject.GetAddressFormat_Output;
import oracle.el.bssvfoundation.base.BSFNParameters;
import oracle.el.bssvfoundation.base.IContext;
import oracle.el.bssvfoundation.base.PublishedBusinessService;
import oracle.el.bssvfoundation.connection.IConnection;
import oracle.el.bssvfoundation.exception.BSFNServiceInvalidArgException;
import oracle.el.bssvfoundation.exception.BSFNServiceSystemException;
import oracle.el.bssvfoundation.exception.BusinessServiceException;
import oracle.el.bssvfoundation.services.IBSFNService;
import oracle.el.bssvfoundation.util.ElMessage;
import oracle.el.bssvfoundation.util.ElMessageList;
import oracle.el.bssvfoundation.util.MathNumeric;
```

6. Within the GetAddressFormat method there is a “TODO” comment for creating the internal value object. Type in the below code:

   GetAddressFormat_Output internalVO = new GetAddressFormat_Output();
   internalVO.setAddressNumber(vo.getAddressNumber());

   After inserting the code should look like the below:

```
protected GetAddressFormat_Output GetAddressFormat(IContext context, IConnection connection, GetAddressFormat_Input vo) thr
    //perform all work within try block, finally will clean up any connections.
    try {
        //Call start published method, passing context of null
        //will return context object so BSFN or DB operation can be called later.
        //Context will be used to indicate default transaction boundary, as well as access
        //to formatting and logging operations.
        context = startPublishedMethod(context, "GetAddressFormat", vo);

        //Create new PublishedBusinessService messages object for holding errors and warnings that occur during processing.
        ElMessageList messages = new ElMessageList();

        //TODO: Create a new internal value object.
         GetAddressFormat_Output internalVO = new GetAddressFormat_Output();
         internalVO.setAddressNumber(vo.getAddressNumber());
```

7. At the end of the GetAddressFormat method modify the code as shown below.

   - Delete the follwing line of code:

     GetAddressFormat_Output confirmVO = new GetAddressFormat_Output(internalVO);

   - Change all references to "*confirmVO*" to "*internalVO*".

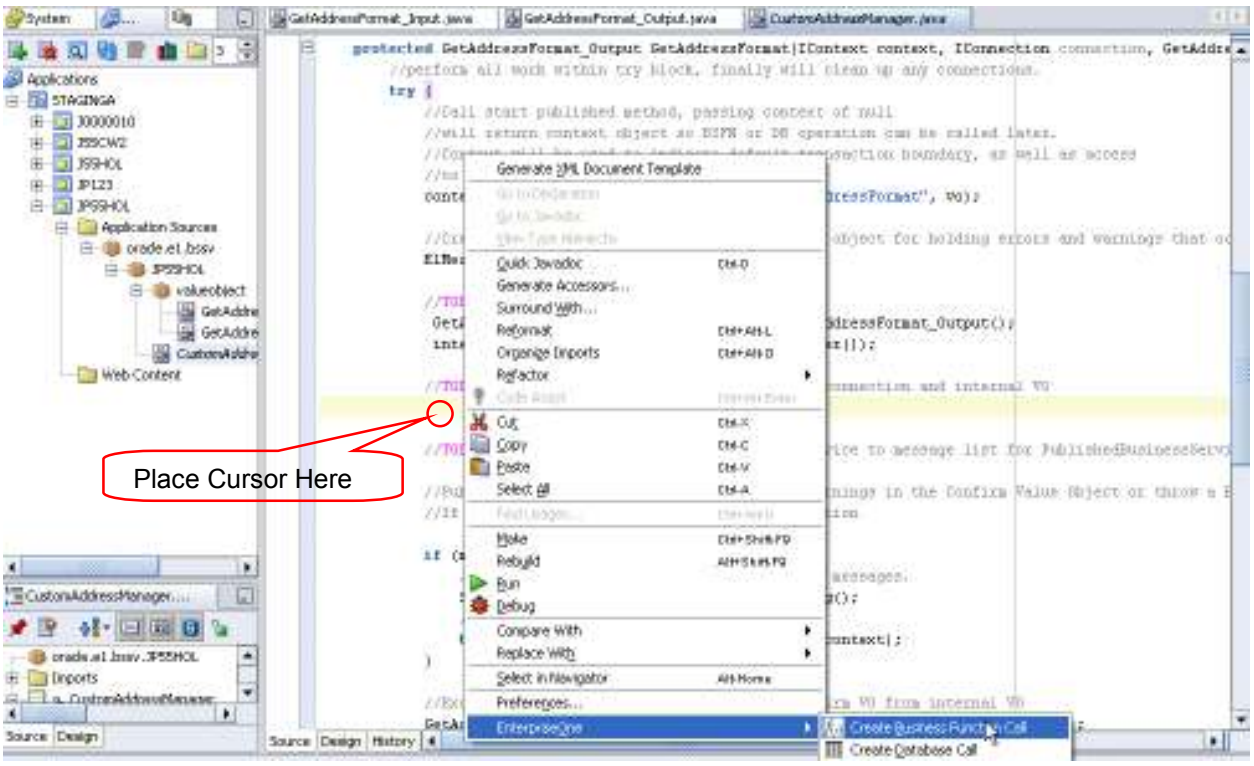
8. Once these changes have been made – the code should look like the below:

```
        //Throw new BusinessServiceException
        throw new BusinessServiceException(error, context);
      }

      //Exception was not thrown, so create the confirm VO from internal VO

      internalVO.setElMessageList(messages);
      finishPublishedMethod(context, "GetAddressFormat");
      //return outVO, filled with return values and messages
      return internalVO;
    } finally {
      //Call close to clean up all remaining connections and resources.
      close(context, "GetAddressFormat");
    }
```

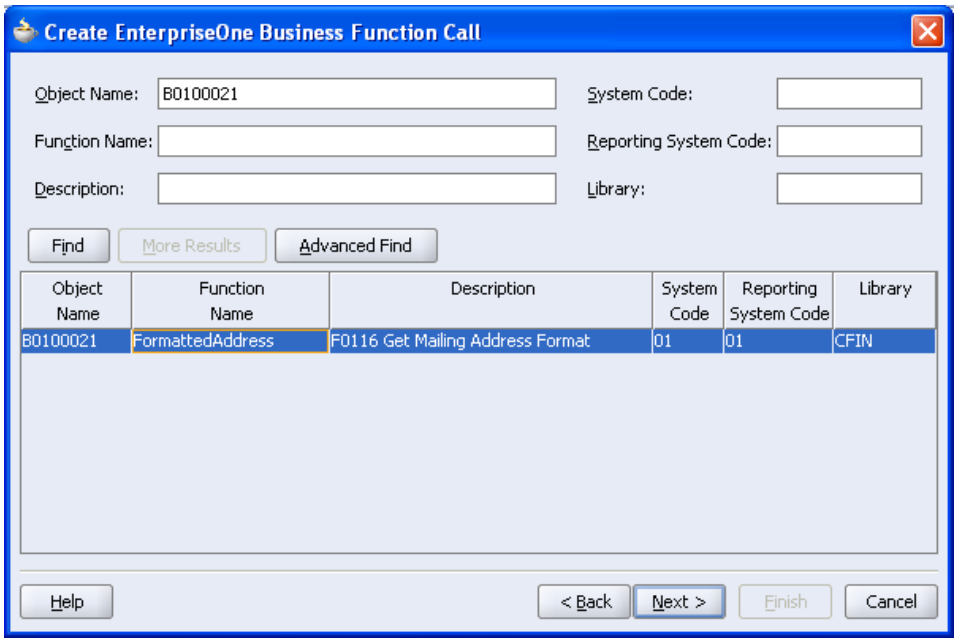
9. Save the Code.

2.7 Adding a Business Function Call

1. In the “GetAddressFormat” methcd, after the **“//TODO: Call BusinessService passing context, connection and internal VO**” comment, place the cursor, right click, and select the EnterpriseOne→Create Business Function Call wizard.



2. In the *Create EnterpriseOne Business Function Value Call* window, enter **B0100021** as the Object Name and press the **Find** button. Select the Business Function from the result and press the **Next** button.



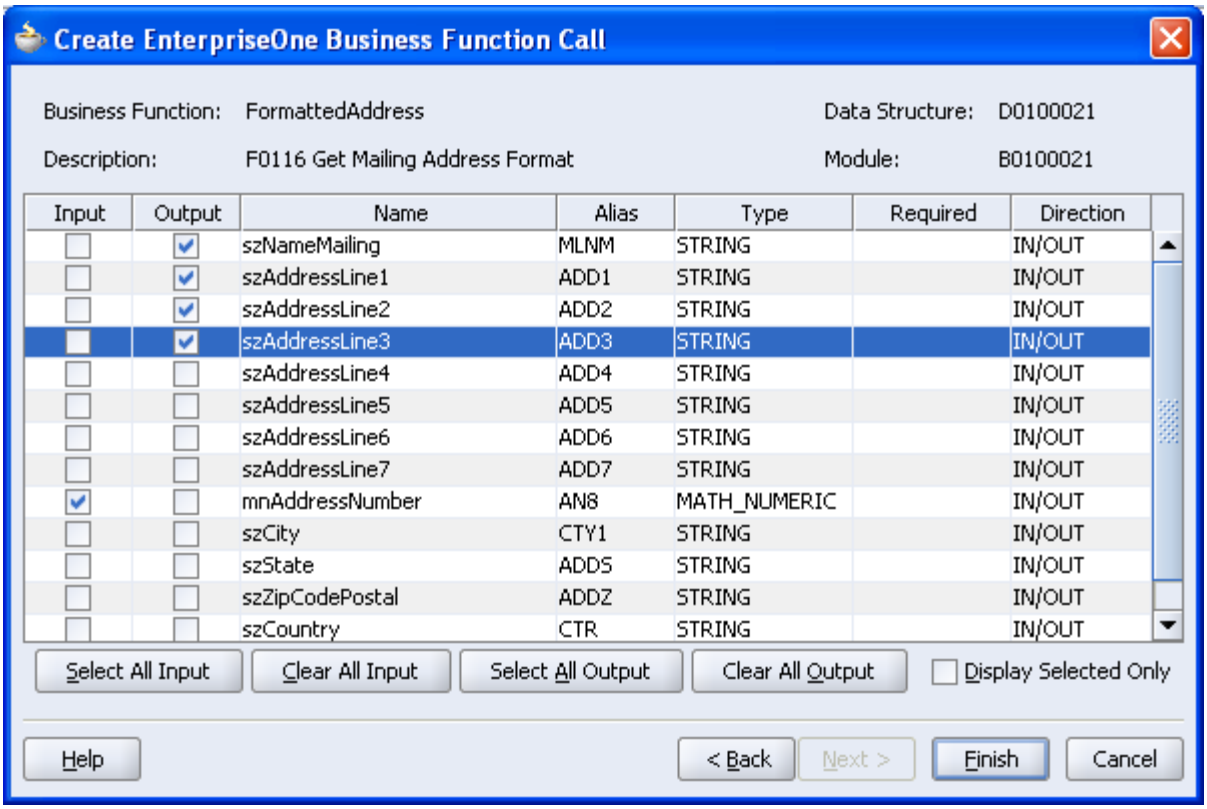
3. Select the following fields and press the **Finish** button.

Input:

Column Name	Alias
mnAddressNumber	AN8

Output:

Column Name	Alias
szNameMailing	MLNM
szAddressLine1	ADD1
szAddressLine2	ADD2
szAddressLine3	ADD3



4. A call to a new method will be generated where the cursor was positioned and should look like the below:

```
//TODO: Call BusinessService passing context, connection and internal V0

callFormattedAddress(context, connection, internalV0);
```

5. The “callFormattedAddress” method will also be created at the bottom of the code. Make to coding changes below so that the correct value object is passed in and the address number is loaded into the business function parameters.

```
/**
 * Calls the FormattedAddress(B0100021) business function which has the D0100021 datastructure.
 * @param context conditionally provides the connection for the business function call and logging information
 * @param connection can either be an explicit connection or null. If null the default connection is used.
 * @param TODO document input parameters
 * @return A list of messages if there were application errors, warnings, or informational
 * messages. Returns null if there were no messages.
 */
private static ElMessageList callFormattedAddress(IContext context, IConnection connection, GetAddressFormat_Output intern
    BSFNParameters bsfnParams = new BSFNParameters();

    // map input parameters from input value object
    if (internalV0.getAddressNumber() != null)
        bsfnParams.setValue("mnAddressNumber", new MathNumeric(internalV0.getAddressNumber()));
```

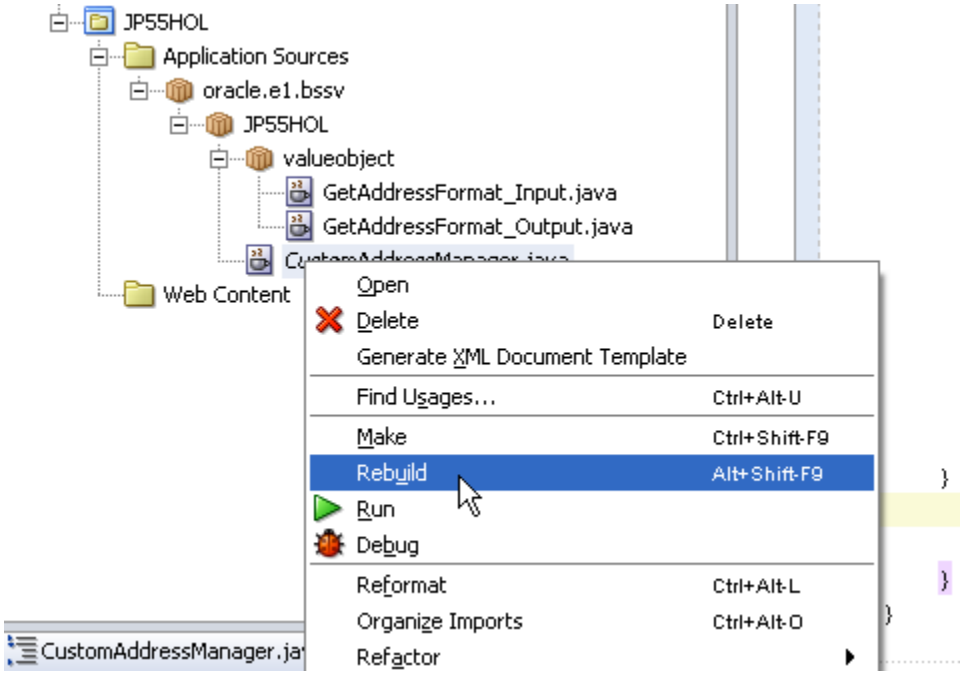
6. At the bottom of the “callFormattedAddress” method you will need to modify the “map output..” section. This should just removing the “sz” from the set methods – for example change “setszNameMailing” to “setNameMailing”:

   The code should loook like the below when finished.

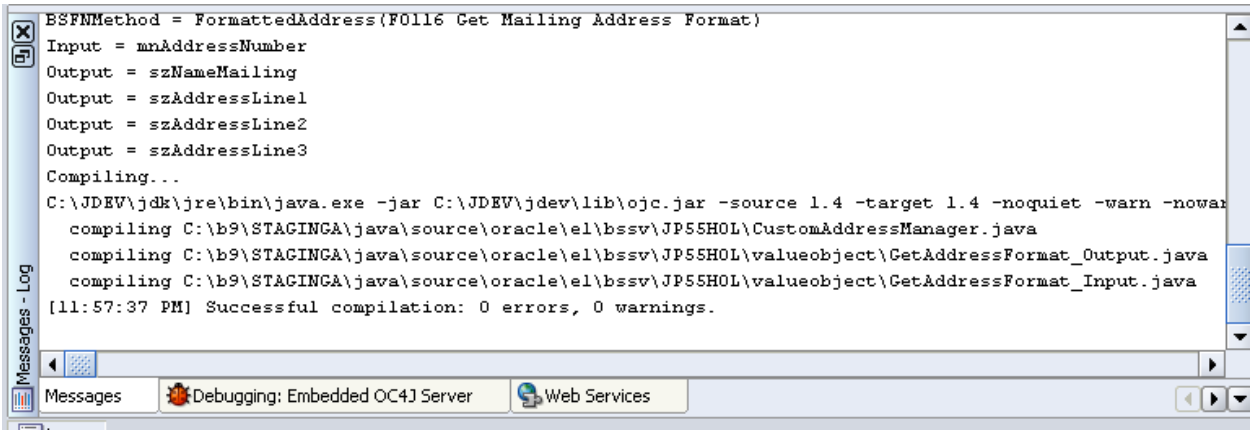
```
        //map output parameters to output value object
        internalVO.setNameMailing((String)bsfnParams.getValue("szNameMailing"));
        internalVO.setAddressLine1((String)bsfnParams.getValue("szAddressLine1"));
        internalVO.setAddressLine2((String)bsfnParams.getValue("szAddressLine2"));
        internalVO.setAddressLine3((String)bsfnParams.getValue("szAddressLine3"));

        //return any errors, warnings, or informational messages to the caller
        return bsfnParams.getElMessageList();
    }
}
```

7. Save the code.

8. Build the code by highlighting “CustomAddressManager” in the navigation pane, right click, and select **Rebuild**.

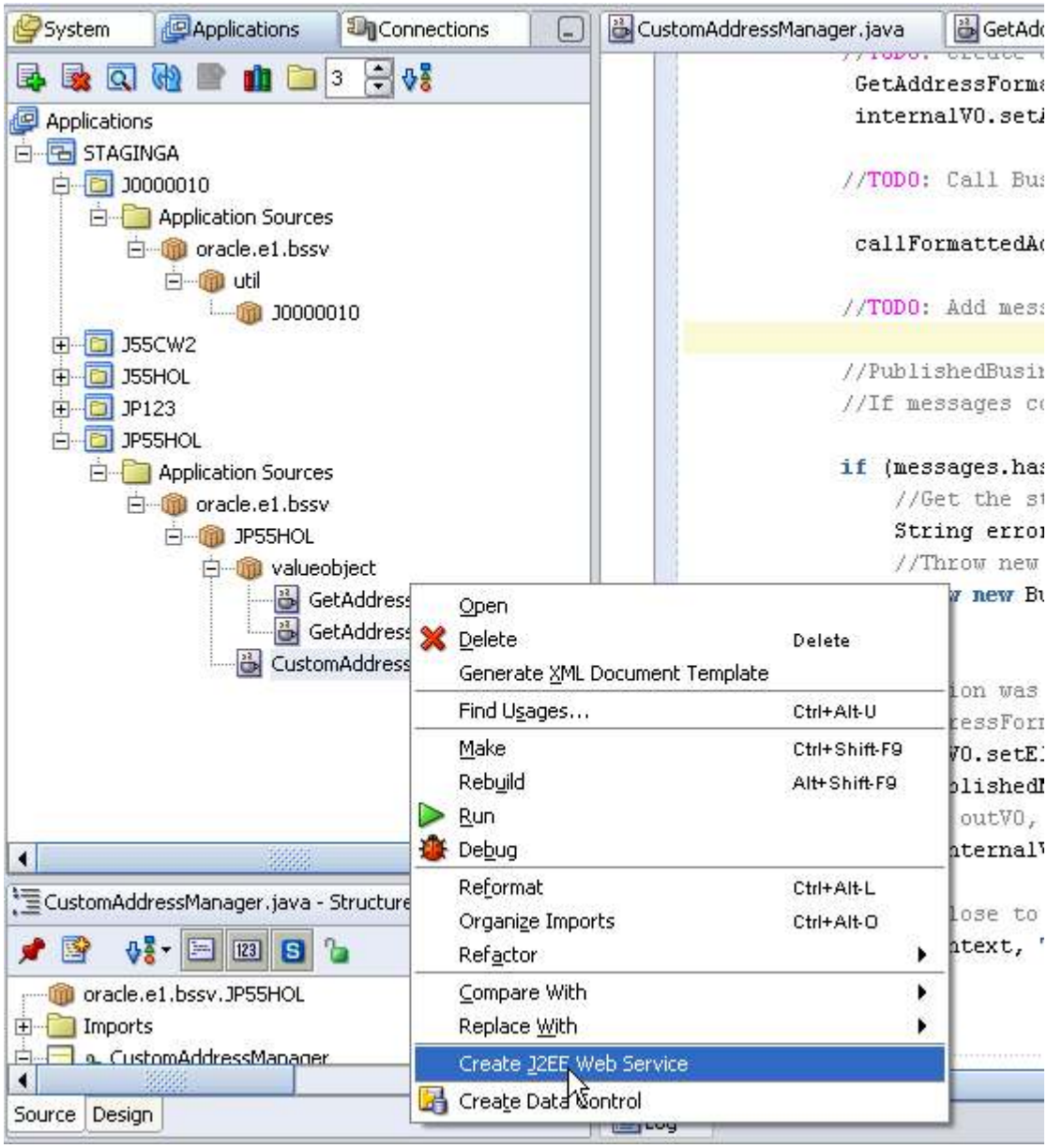


9. The code should successfully build with a log that looks like the below:


BSFNMethod = FormattedAddress(F0116 Get Mailing Address Format)
Input = mnAddressNumber
Output = szNameMailing
Output = szAddressLine1
Output = szAddressLine2
Output = szAddressLine3
Compiling...
C:\JDEV\jdk\jre\bin\java.exe -jar C:\JDEV\jdev\lib\ojc.jar -source 1.4 -target 1.4 -noquiet -warn -nowar
compiling C:\b9\STAGINGA\java\source\oracle\el\bssv\JP55H0L\CustomAddressManager.java
compiling C:\b9\STAGINGA\java\source\oracle\el\bssv\JP55H0L\valueobject\GetAddressFormat_Output.java
compiling C:\b9\STAGINGA\java\source\oracle\el\bssv\JP55H0L\valueobject\GetAddressFormat_Input.java
[11:57:37 PM] Successful compilation: 0 errors, 0 warnings.
Messages - Log
Messages
Debugging: Embedded OC4J Server
Web Services

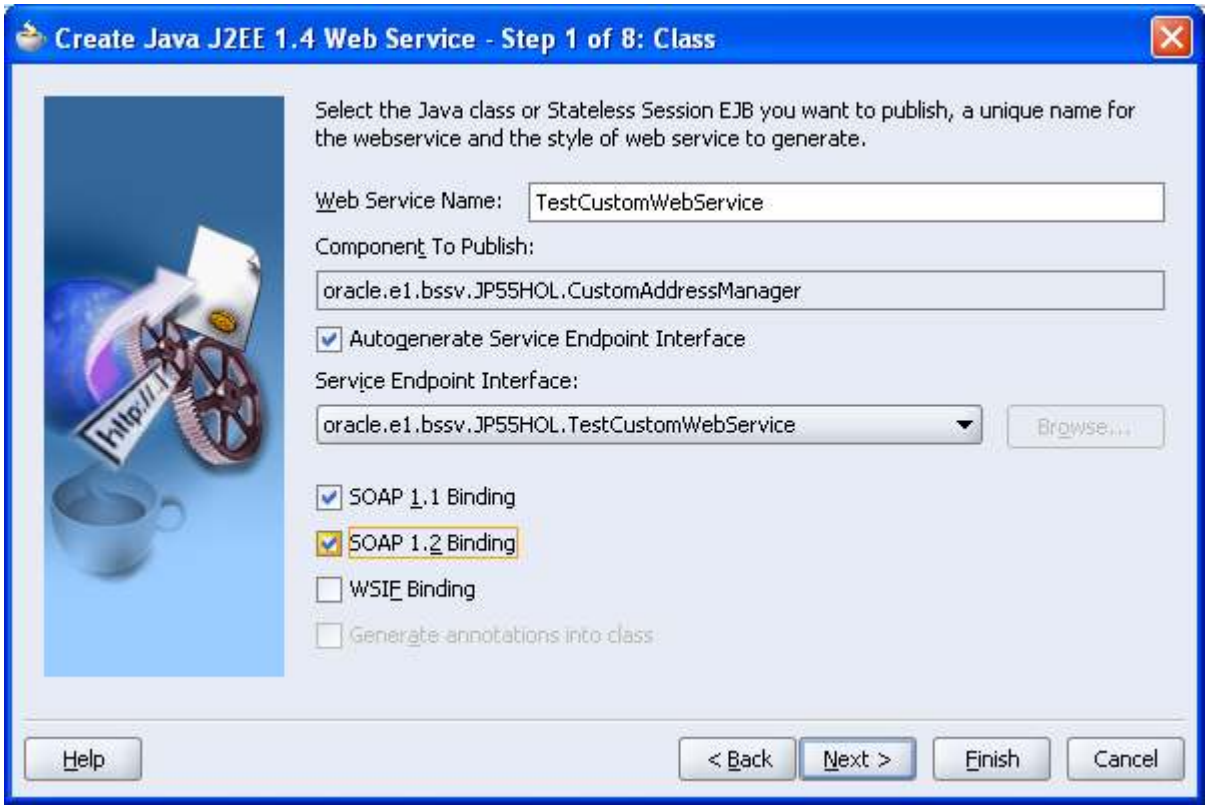
3 Testing the Published Business Service

1. Highlight the CustomerAddressManager class in the Navigation pane, right click, and select the "**Create J2EE Web Service**" option.

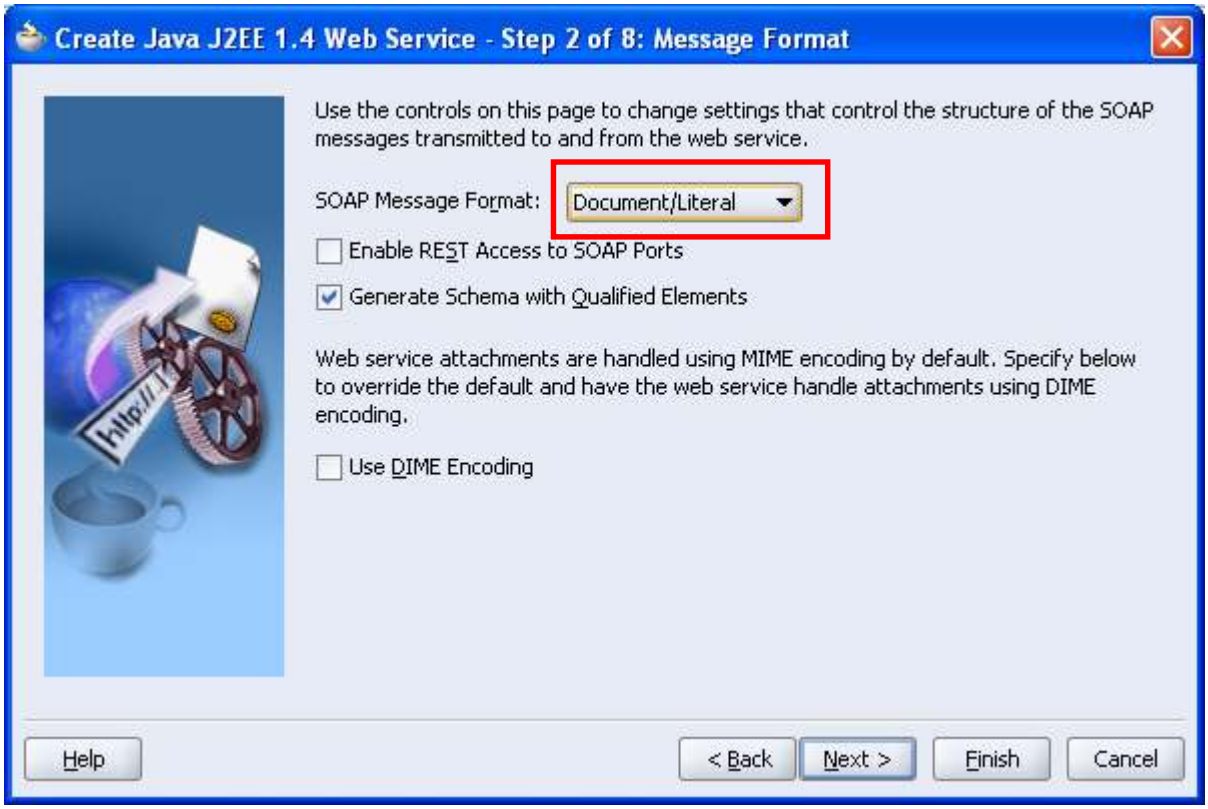


2. On the Select J2EE Web Service Version screen, select J2EE 1.4 and press **OK**.

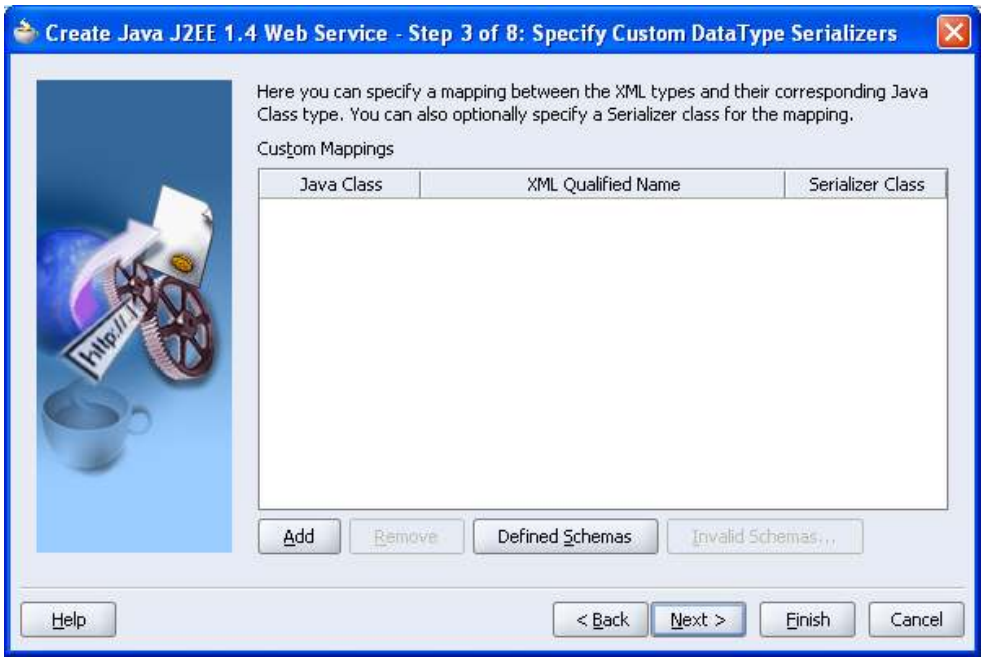
3. On the "Create Java J2EE 1.4 Web Service Step 1" screen type in "TestCustomWebService" for the Web Service Name. Press **Next**:



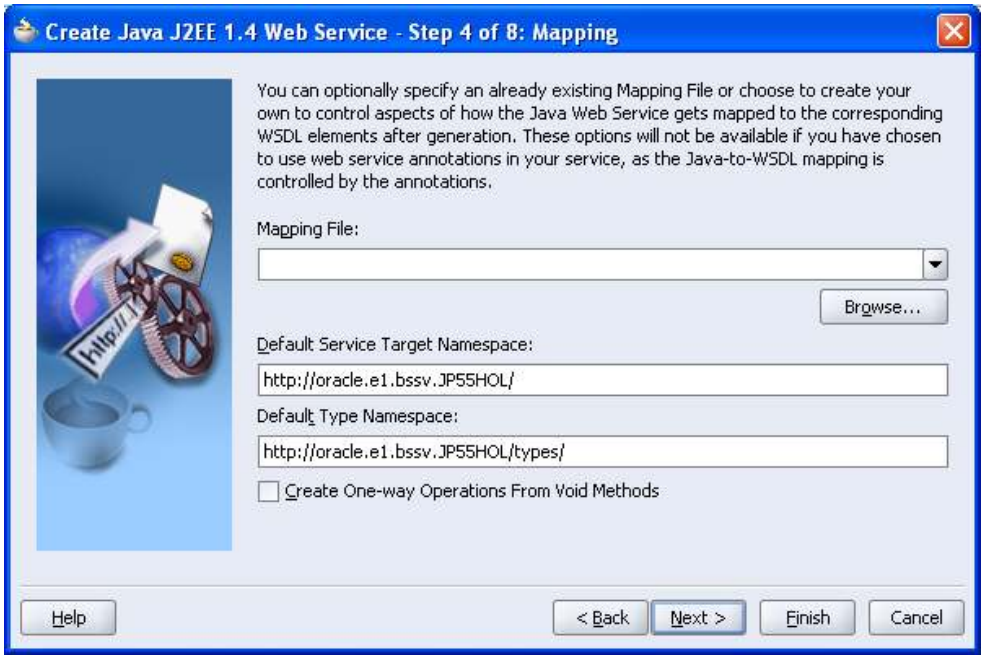
4. On the “Create Java J2EE 1.4 Web Service Step 2” screen – Change the “SOAP Message Format” to “**Document/Literal**”: Press **Next**:



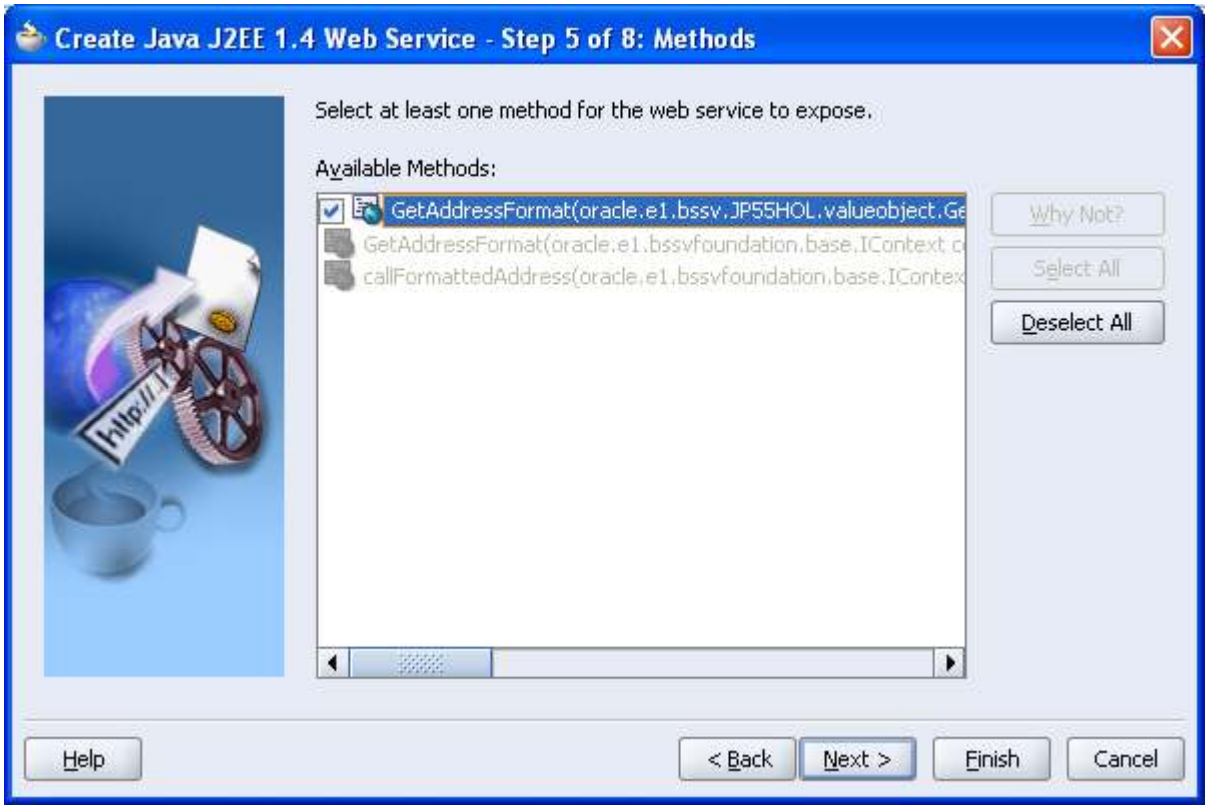
5. On the “Create Java J2EE 1.4 Web Service Step 3” screen – Make no changes and Press **Next**.



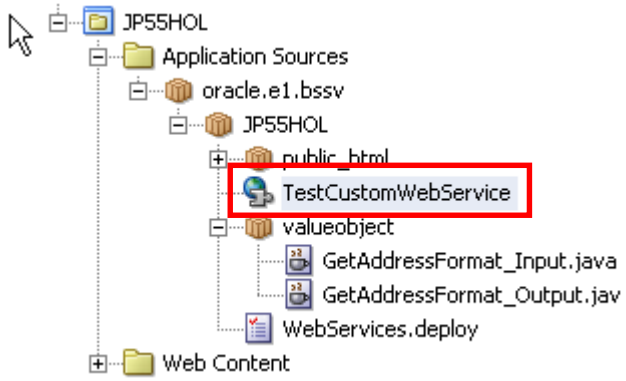
6. On the “Create Java J2EE 1.4 Web Service Step 4” screen – Make no changes and Press **Next**.



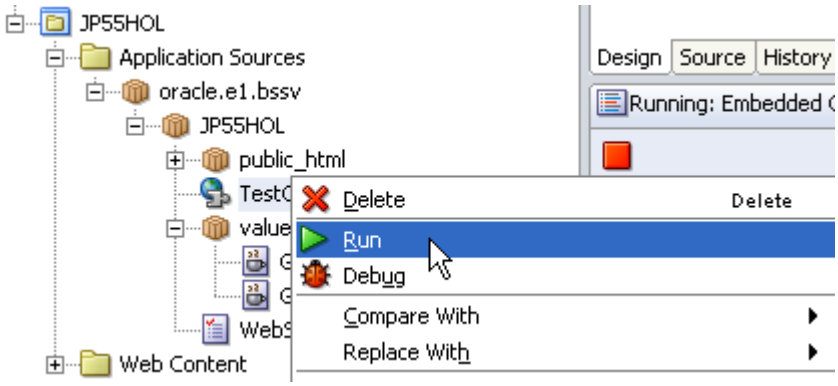
7. On the “Create Java J2EE 1.4 Web Service Step 5” screen – Make sure that the “GetAddressFormat” method is selected. Press **Finish**.



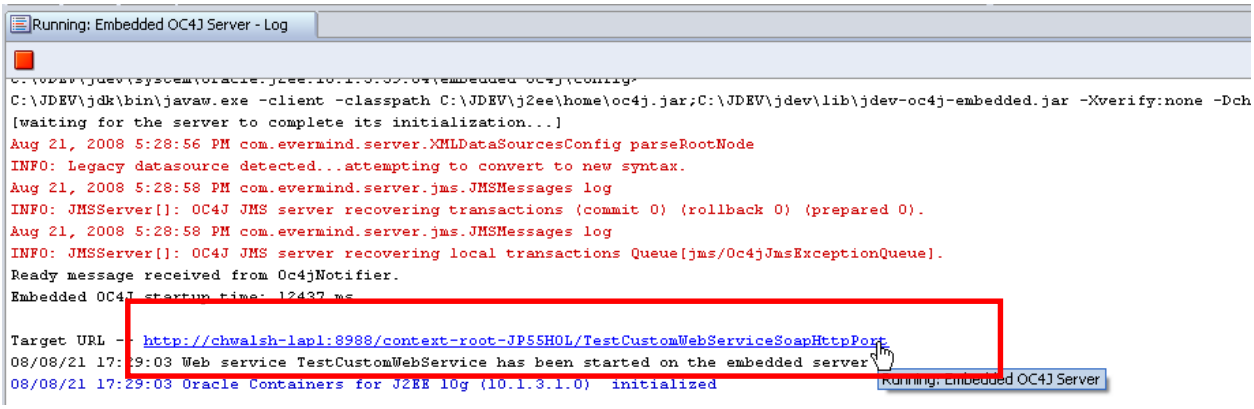
8. A set of new objects should be created – including “TestCustomWebService” .



9. Select the “TestCustomWebService”, Right Click, and select the “Run” option.

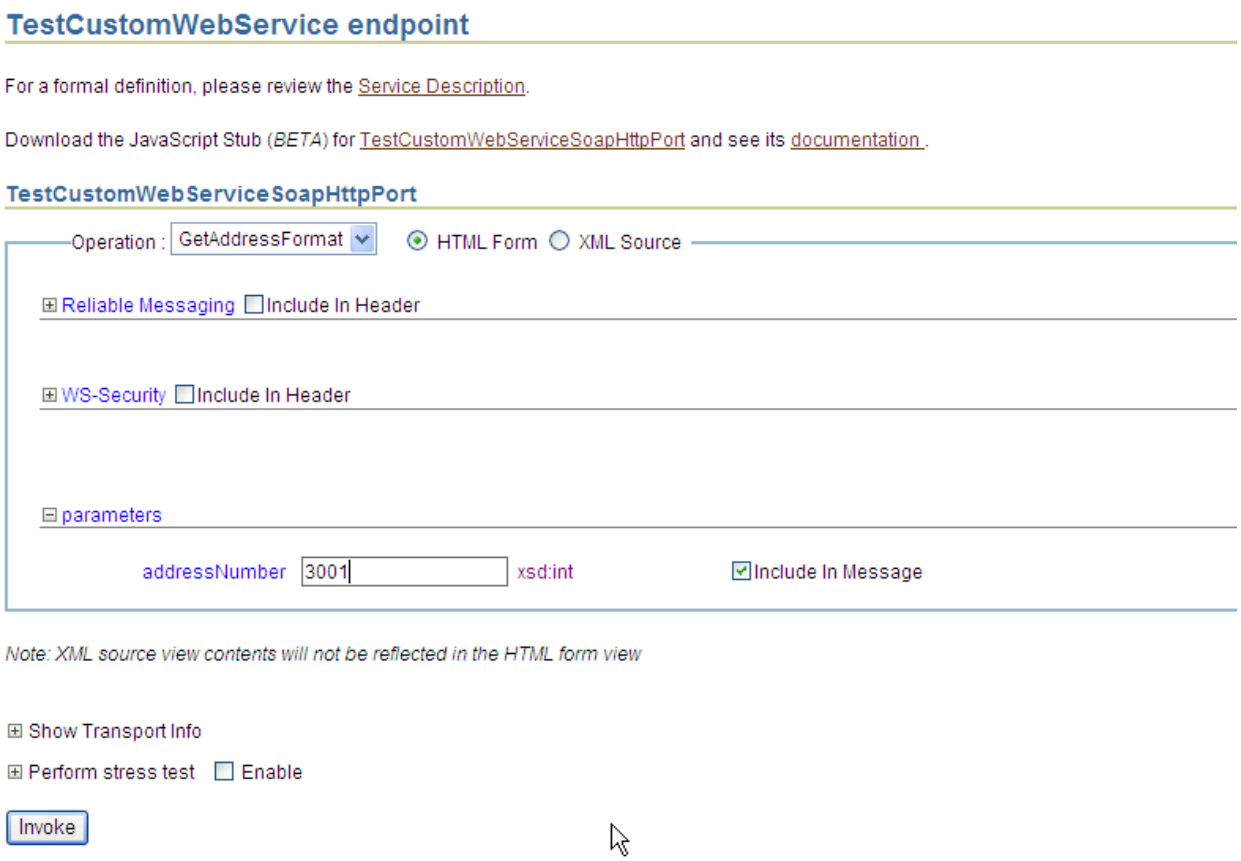


10. An embedded OC4J container will be launched and the log window will display a URL that can be selected to test the service. Place the cursor on the URL and left click.



11. A browser should be launched with a page that looks like the below. If there is an error loading the page then make sure that the url is correct. There is a known bug in Jdev that the context root is sometimes held from previous sessions. The url should look like the below – if it does not the type this url into the browser:

    http://oracle-jde:8988/context-root-**JP55HOL**/TestCustomerWebServiceSoapHttpPort



12. Enter a valid address book number (i.e. 3001) and press **Invoke**:

13. The Test Result page should display with the mailing address information. To make the xml easier to read – select the "Formatted XML" link:

14. After selecting the Formatted XML link the page should look like: