

Cognizant Technology Solution

Exercises – Java Service

WebMethods Integration Workshop

WebMethods CoE

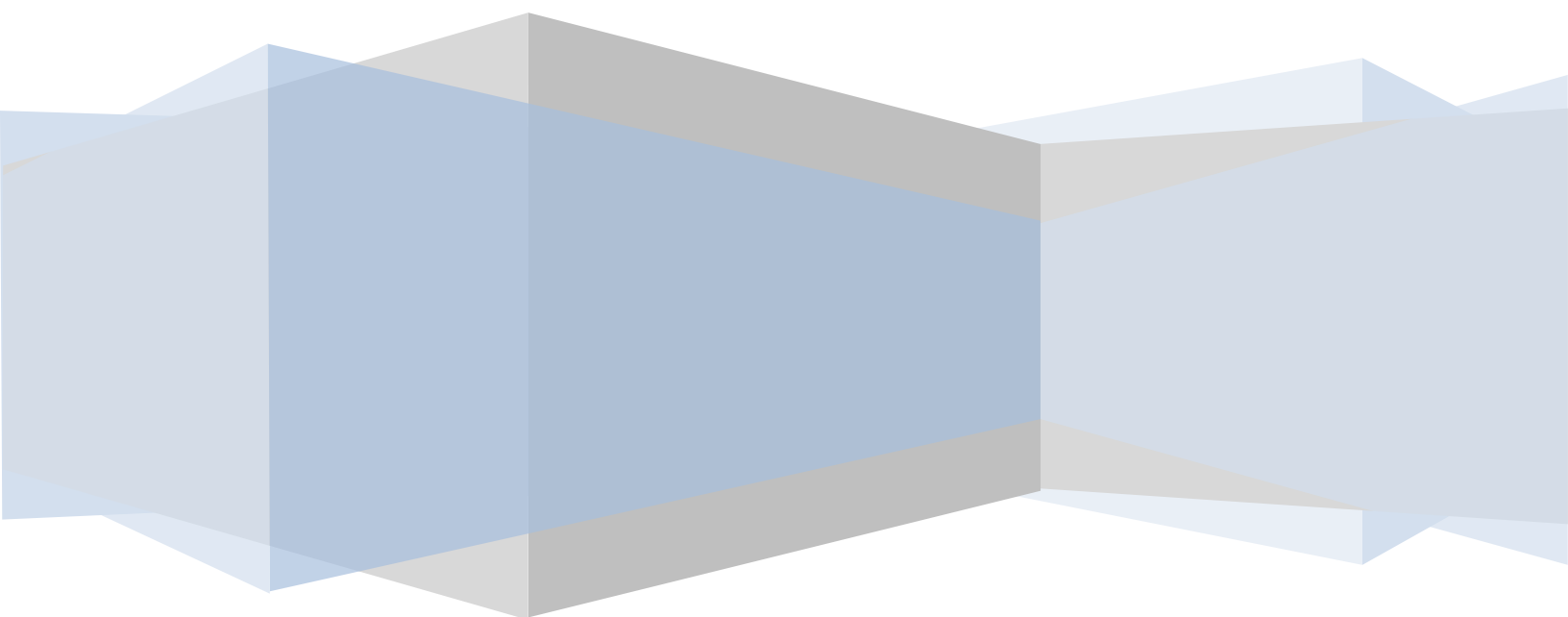


Table of Contents

Introduction	3
The IData Object.....	3
Theory.....	3
Practical	4
Java Service Creation	4
Steps to create java service.....	4

Java Service

Introduction

Since Java is the native language of services, it is the easiest language in which to build a service.

WebMethods Developer provides an Integrated Development Environment (IDE) that you can use to create, compile, and publish Java services. The IDE automatically generates an appropriately structured source file that you simply “fill in” using the builtin editor. When you save the source file, the IDE automatically compiles it and registers it on the server.

The IData Object

The **IData object** is the universal container that services use to receive input from and deliver output to other programs. It contains an ordered collection of **key/value pairs** on which a service operates. An IData object can contain any number of key/values pairs (elements). The keys in an IData object must be Strings. The values can be any Java objects (including IData objects).

IDataFactory is used to create IData Object. It is a class which extends java.lang.Object. Getting data from and putting data into IData elements takes two steps. First, you must position the cursor at the IData element. Next, you get or set the data in that element. The class that you can use to position a cursor in an IData object is **IDataCursor**. The IDataCursor class contains methods for performing basic cursor operations such as placing the cursor at the first, last, or next element in the object.

After you position the cursor on the element with which you want to work, you can use the **getValue** or **setValue** methods to read or write the value of that element, respectively. This class also provides methods for inserting new elements, getting key names, and deleting elements.

Theory

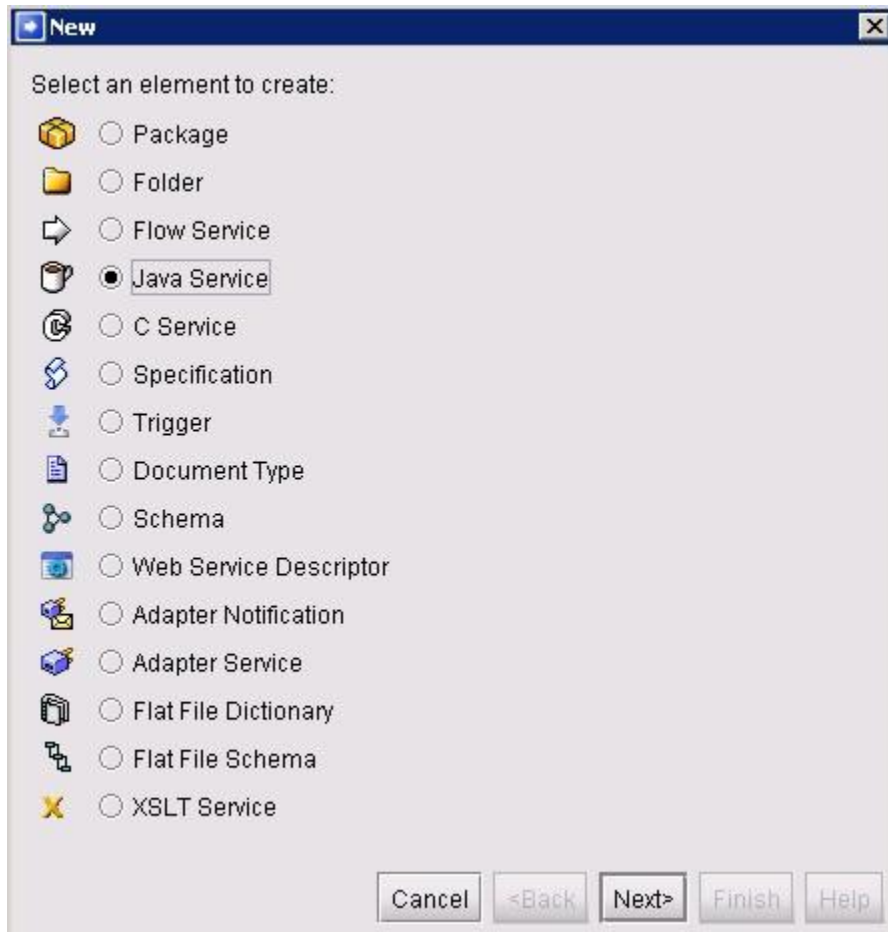
- 8-2-SP2_Developer_Users_Guide (Chapter 15)
- 8-2-SP1_Service_Development_Help (Chapter 12)

Practical

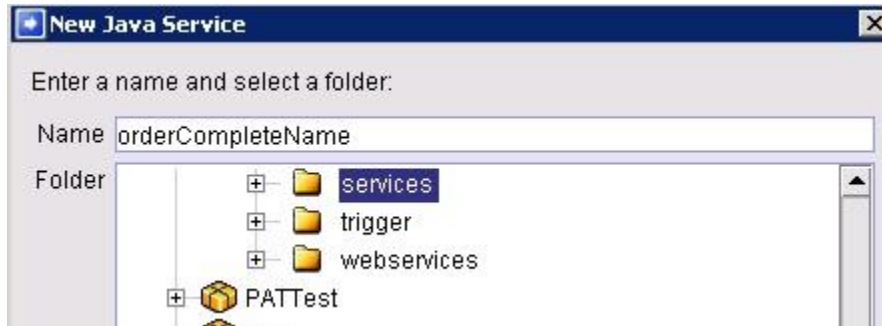
Java Service Creation

Steps to create java service

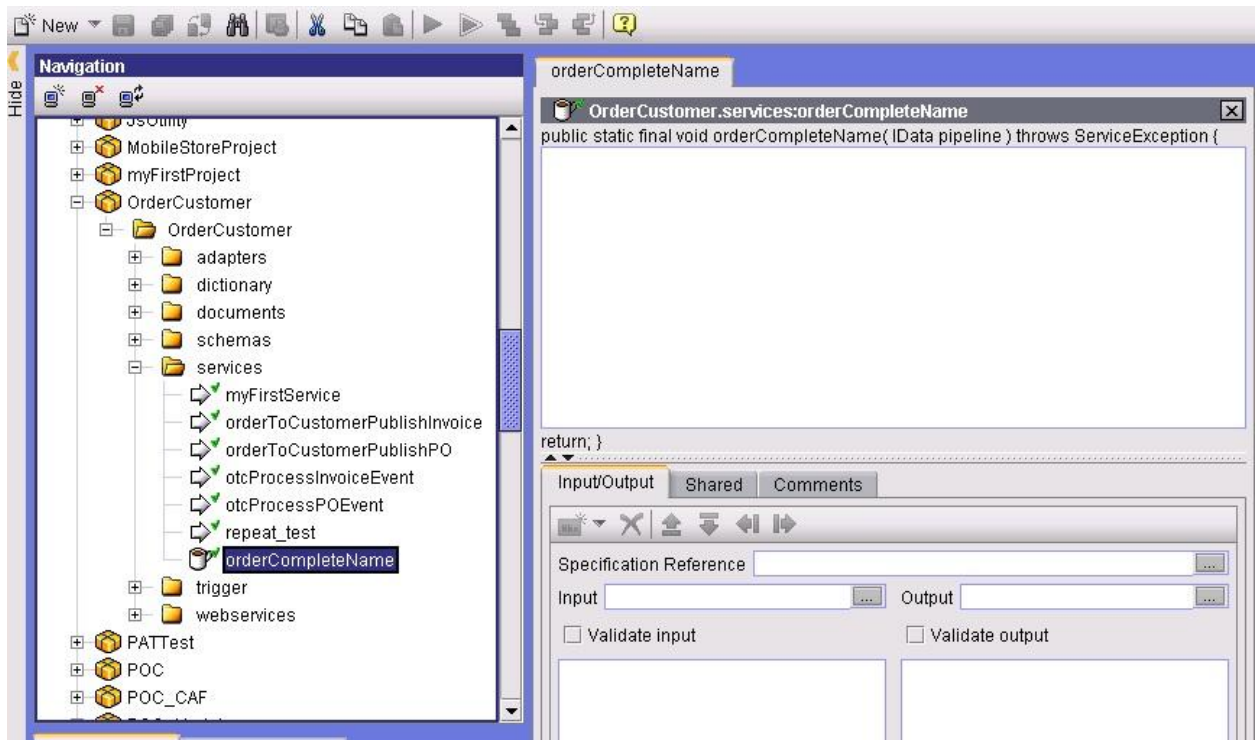
- 1) Right click on services folder select **New** -> **All Choices**. Developer opens **New** wizard.
- 2) Select **Java Service** from the list of elements. Click **Next**.



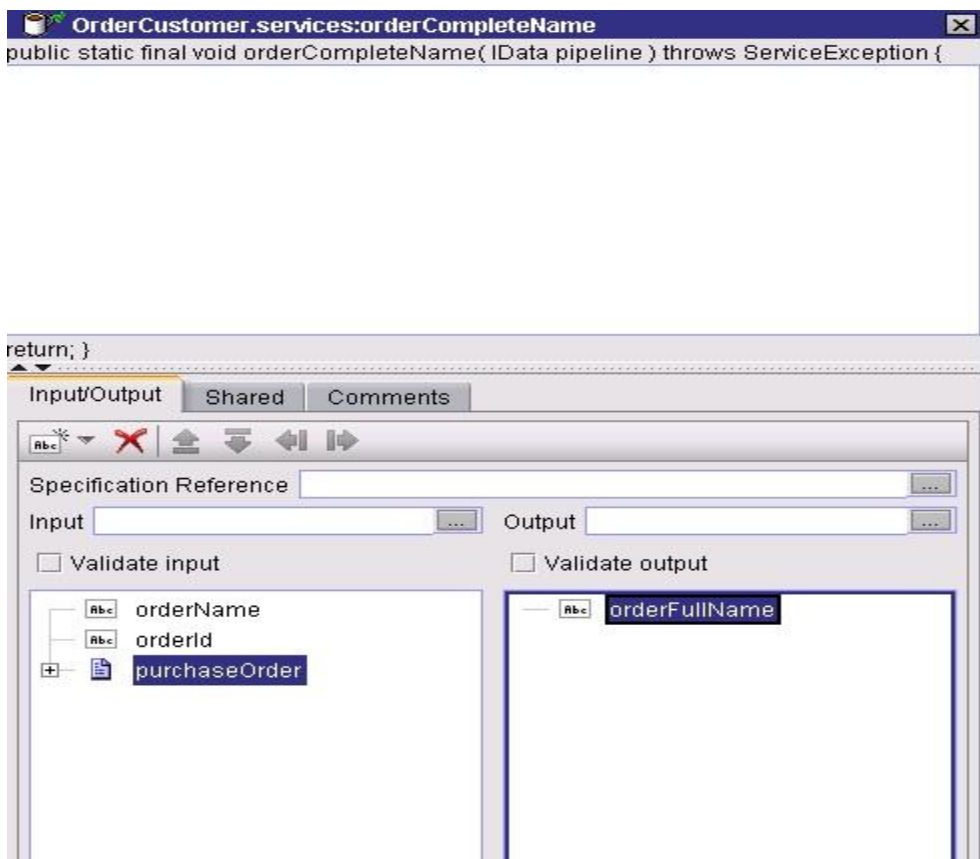
- 3) Type a unique name for the java service and select the appropriate folder. Click **Finish**.



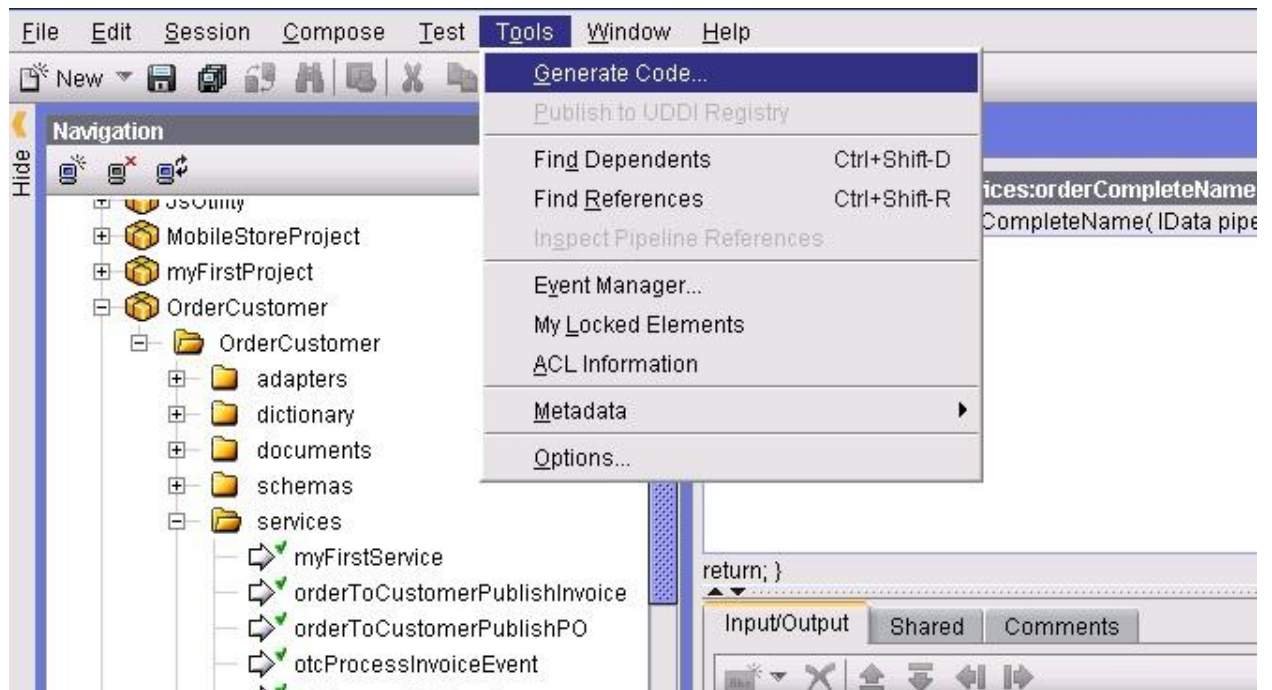
- 4) A java service is created with specified name.



- 5) Click on **Input/Output** tab. Add variables in input and output fields which are required for service.



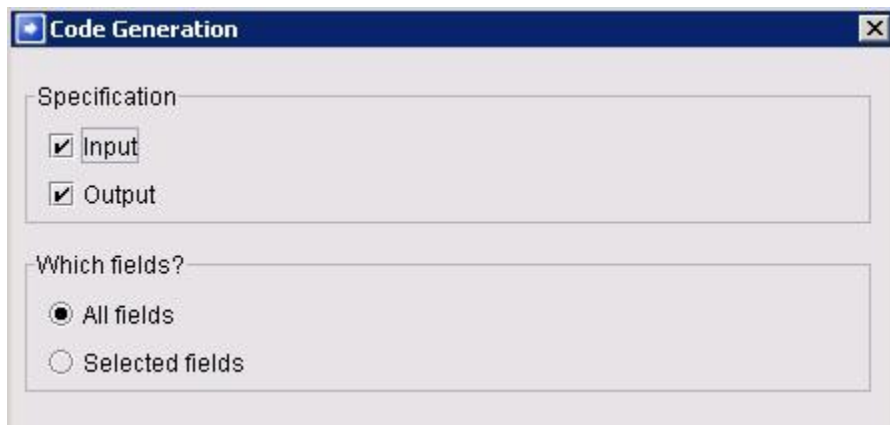
- 6) Click on **Tools** and select **Generate Code**. Developer opens **Code Generation Wizard**.



- 7) Select **For implementing this service** from the list of elements. Click **Next**.

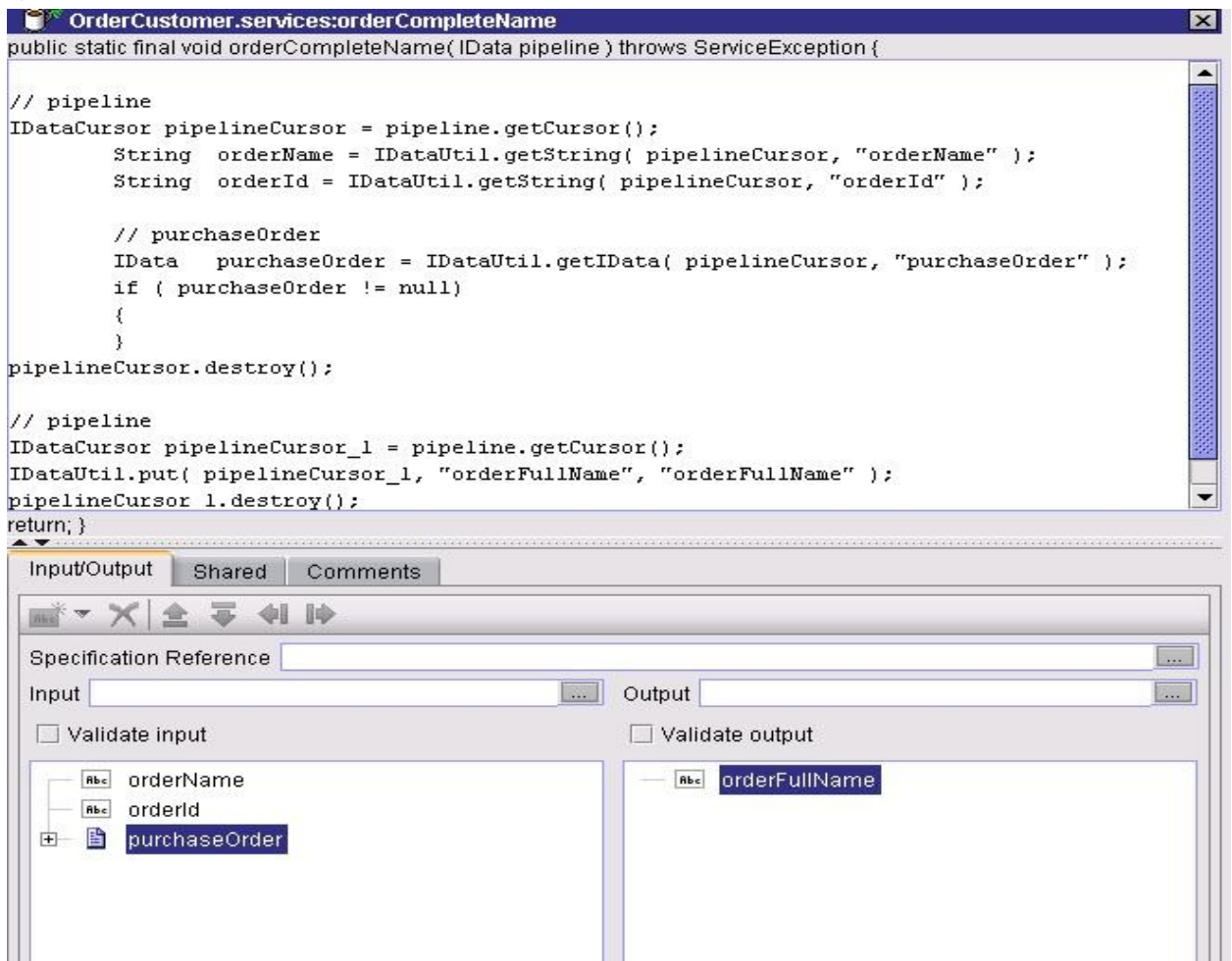


- 8) Under **Specification** select both **Input** and **Output** and under **Which field?** Select **All fields**. Click **Finish**.



The 'Code Generation' dialog box has two sections. The 'Specification' section contains two checked checkboxes: 'Input' and 'Output'. The 'Which fields?' section contains two radio buttons: 'All fields' (which is selected) and 'Selected fields'.

- 9) Developer generates code and places it on the Clipboard.
- 10) Paste (Ctrl+v) the contents of the Clipboard onto the editor (Highlight the Editor by clicking on it).



The Eclipse IDE shows a Java class named `OrderCustomer.services.orderCompleteName` with the following code:

```
public static final void orderCompleteName( IData pipeline ) throws ServiceException {  
  
    // pipeline  
    IDataCursor pipelineCursor = pipeline.getCursor();  
        String  orderName = IDataUtil.getString( pipelineCursor, "orderName" );  
        String  orderId  = IDataUtil.getString( pipelineCursor, "orderId" );  
  
        // purchaseOrder  
        IData  purchaseOrder = IDataUtil.getIData( pipelineCursor, "purchaseOrder" );  
        if ( purchaseOrder != null )  
        {  
        }  
    pipelineCursor.destroy();  
  
    // pipeline  
    IDataCursor pipelineCursor_1 = pipeline.getCursor();  
    IDataUtil.put( pipelineCursor_1, "orderFullName", "orderFullName" );  
    pipelineCursor_1.destroy();  
    return; }  
}
```

Below the code editor, the 'Input/Output' tab is active. It displays a 'Specification Reference' field and two input/output fields. The 'Input' field is empty, and the 'Output' field contains 'orderFullName'. There are checkboxes for 'Validate input' and 'Validate output', both of which are unchecked. A tree view on the left shows a hierarchy with 'orderName', 'orderId', and 'purchaseOrder' as children of a parent node.

11) Type the code to perform the logic.

The screenshot shows a code editor window titled "OrderCustomer.services:orderCompleteName". The code is as follows:

```
public static final void orderCompleteName( IData pipeline ) throws ServiceException {  
  
    // pipeline  
    IDataCursor pipelineCursor = pipeline.getCursor();  
    String  orderName = IDataUtil.getString( pipelineCursor, "orderName" );  
    String  orderId = IDataUtil.getString( pipelineCursor, "orderId" );  
  
    // purchaseOrder  
    IData   purchaseOrder = IDataUtil.getIData( pipelineCursor, "purchaseOrder" );  
    if ( purchaseOrder != null )  
    {  
    }  
  
    pipelineCursor.destroy();  
  
    String orderFullName = orderName.concat(orderId);  
  
    // pipeline  
    IDataCursor pipelineCursor_1 = pipeline.getCursor();  
    IDataUtil.put( pipelineCursor_1, "orderFullName", orderFullName );  
    pipelineCursor_1.destroy();  
  
    return; }  
}
```

Below the code editor is the "Input/Output" configuration panel. It includes tabs for "Input/Output", "Shared", and "Comments". The "Input/Output" tab is active, showing a "Specification Reference" field, "Input" and "Output" fields, and checkboxes for "Validate input" and "Validate output". The "Input" field is populated with a tree structure showing "orderName", "orderId", and "purchaseOrder". The "Output" field is populated with "orderFullName".

- 12) Click on **Shared** tab and enter values in **Extends**, **Implements**, **Imports** and **Source** fields as per requirement.

Input/Output **Shared** Comments

Note: these fields are shared by all Java services in this folder.

Extends WmSampleAdapter

Implements

Imports java.util.Date

Imported Java packages

Source

```
static IData addrDoc
static
(
    addrDoc = IDataFactory.create ();

    IDataCursor addrCursor = addrDoc.getCursor ();
    addrCursor.last ();
    addrCursor.insertAfter ("companyName", "Jones, Ltd.");
```

Shared, private source