Container Link Support in Rational Application Developer for WebSphere Software v7.5

Tips to create J2C Java beans and ant scripts using J2C Bean wizard from a Cobol file that contains CICS Channels and Containers Commands

Laszlo Benedek & Ivy Ho

February 27, 2009

Requirements

You should have Rational Application Developer for WebSphere Software v7.5 with J2EE Connector (J2C) Tools installed if you want to follow the steps.

The article assumes the reader has some basic knowledge of ant scripts. We will not explain the basics of how the ant script works. J2C ant scripts are already supported in Rational Application Developer for WebSphere Software v7, we will only explore the specifics for the CICS Container Link support in the generated ant script.

If you are interested in the basics of J2C ant script, this paper Working with J2C Ant Scripts in Rational Application Developer v7 will give you some basic knowledge.

Introduction

Level: Intermediate

In IBM Rational Application Developer for WebSphere Software version 7.5, one of the key features in J2C Connector Tools is the CICS Channel Support. You can generate J2C Java Beans and Data binding files from importing Cobol source that contains CICS Channel commands. You can save the settings into an Ant script when you go through the J2C Bean wizard or the CICS/IMS Data Binding wizard.

This article is divided into two major sections.

1. The first section is for readers who are interested in using the tooling for CICS Container Link Support in Rational Application Developer v7.5.

2. The second section is for users who like to explore the generated ant scripts. The sample ant scripts included in this article were created using these J2C Tools.

CICS Channel Introduction

CICS Channel

In the past, CICS application programs use communication area COMMAREA to exchange data. The new CICS Channel support enhances how data is transferred between programs.

Containers are named blocks of data designed for passing information between programs. You can think of them as named communication areas (COMMAREAs). Programs can pass any number of containers between each other. Containers are grouped together in sets called channels; a channel is analogous to a parameter list.

Here are some of the benefits of Channel/Container model over COMMAREA:

- Unlike COMMAREA, channels are not limited in size to 32KB. There is no limit to the number of containers that can be added to a channel, and the size of individual containers is limited only by the amount of storage that you have available.
- Because a channel can comprise multiple containers, it can be used to pass data in a more structured way. In contrast, a COMMAREA is a monolithic block of data.
- Unlike COMMAREAs, channels don't require the programs that use them to know the exact size of the data returned.

CICS Sample Cobol Program EC03

This sample server program listing1 below demonstrates the use of channels and containers in a CICS program. It queries the length of data in a container and places the length, the current date and the current time into containers InputDataLength, CurrentDate and CurrentTime respectively. A message indicating success or failure is returned in container OutputMessage. If no channel is passed to the program it abends with abend code NOCH.

EC03 is a Basic CICS Server Cobol sample program which is used with the frontend sample programs to demonstrate the use of External Call Interface (ECI) with channels and containers.

The following code is sample code created by IBM Corporation. This sample code is not part of any standard IBM product and is provided to you solely for the purpose of assisting you in the development of your applications. The code is provided 'as is', without warranty or condition of any kind. IBM shall not be liable for any damages arising out of your use of the sample code, even if IBM has been advised of the possibility of such damages.

Listing 1.Samples CICS program with Channels

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. EC03.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

      *   Container names
       01 DATECONTAINER       PIC X(16) VALUE 'CurrentDate'.
       01 TIMECONTAINER       PIC X(16) VALUE 'CurrentTime'.
       01 INPUTCONTAINER      PIC X(16) VALUE 'InputData'.
       01 OUTPUTCONTAINER     PIC X(16) VALUE 'OutputMessage'.
       01 LENGTHCONTAINER     PIC X(16) VALUE 'InputDataLength'.

      *   Data fields used by the program
       01 INPUTLENGTH         PIC S9(8) COMP-5.
       01 CURRENTTIME         PIC S9(15) COMP-3.
       01 CHANNELNAME         PIC X(16) VALUE SPACES.
       01 OUTPUTSTRING        PIC X(72) VALUE SPACES.
       01 DATESTRING          PIC X(16) VALUE SPACES.
       01 TIMESTRING          PIC X(16) VALUE SPACES.
       01 RESPCODE            PIC S9(8) COMP-5.
       01 RESPCODE2           PIC S9(8) COMP-5.

       PROCEDURE DIVISION.
       MAIN-PROCESSING SECTION.

      *   Get name of channel
           EXEC CICS ASSIGN CHANNEL(CHANNELNAME)
                            END-EXEC.
      *   If no channel passed in, terminate with abend code NOCH
            IF CHANNELNAME = SPACES THEN
                 EXEC CICS ABEND ABCODE('NOCH') NODUMP
                             END-EXEC
            END-IF.

      *   Read content length of container InputData
            EXEC CICS GET CONTAINER(INPUTCONTAINER)
                             CHANNEL(CHANNELNAME)
                             NODATA FLENGTH(INPUTLENGTH)
                             INTOCCSID(037)
                             RESP(RESPCODE)
                             RESP2(RESPCODE2)
                             END-EXEC.

      *   Check response code
            EVALUATE RESPCODE

      *   Container not passed in
                WHEN DFHRESP(CONTAINERERR)
                  STRING INPUTCONTAINER
                             DELIMITED BY SPACE
                       ' container was not passed to the program'
                             DELIMITED BY SIZE
                             INTO OUTPUTSTRING END-STRING

      *   Container is BIT not CHAR
                WHEN DFHRESP(CCSIDERR)
                  IF RESPCODE2 = 3
                    STRING 'Container '
                             DELIMITED BY SIZE
                         INPUTCONTAINER
                             DELIMITED BY SPACE
                         ' type is BIT, not CHAR'
                             DELIMITED BY SIZE
                             INTO OUTPUTSTRING END-STRING
                  END-IF

      *   Read from container OK
                WHEN DFHRESP(NORMAL)
                  STRING 'Read from '
                             DELIMITED BY SIZE
                       INPUTCONTAINER
                             DELIMITED BY SPACE
                       ' container successfully'
                             DELIMITED BY SIZE
                             INTO OUTPUTSTRING END-STRING

                  EXEC CICS PUT CONTAINER(LENGTHCONTAINER)
                             FROM(INPUTLENGTH)
                             FLENGTH(LENGTH OF INPUTLENGTH)
                             BIT NOHANDLE END-EXEC

      *   Other response code
                WHEN OTHER
                  STRING 'The GET CONTAINER command returned an',
                     ' unexpected response code'
                           DELIMITED BY SIZE
                           INTO OUTPUTSTRING END-STRING

           END-EVALUATE.

      *   Place output string in container OutputData
           EXEC CICS PUT CONTAINER(OUTPUTCONTAINER)
                           FROM(OUTPUTSTRING)
                           FLENGTH(LENGTH OF OUTPUTSTRING)
                           CHAR FROMCCSID(037)
                           END-EXEC.

      *   Get the current time and date
           EXEC CICS ASKTIME ABSTIME(CURRENTTIME)
                           END-EXEC.

      *   Format date and time
           EXEC CICS FORMATTIME ABSTIME(CURRENTTIME)
                           DDMMYYYY(DATESTRING) DATESEP('/')
                           TIME(TIMESTRING) TIMESEP(':')
                           END-EXEC.

      *   Place current date in container CurrentDate
           EXEC CICS PUT CONTAINER(DATECONTAINER)
                           FROM(DATESTRING)
                           FLENGTH(LENGTH OF DATESTRING)
                           CHAR FROMCCSID(037)
                           END-EXEC.

      *   Place current time in container CurrentTime
           EXEC CICS PUT CONTAINER(TIMECONTAINER)
                           FROM(TIMESTRING)
                           FLENGTH(LENGTH OF TIMESTRING)
                           CHAR FROMCCSID(037)
                           END-EXEC.

      *   Finish
           EXEC CICS RETURN END-EXEC.

           EXIT.
```

J2C tools for CICS Channel Support

The J2C Tools in Rational Application Developer for WebSphere Software v7.5 provide CICS Channel Support in different ways.

Wizards Support

You can create the J2C generated artifacts for CICS Channel Support using J2C wizards. The two wizards also support ant script generation. It enables you to go through the wizard once, choose the **CICS Channel to Java** Mapping and record the user options and data in an Ant script file.

The two wizards are:

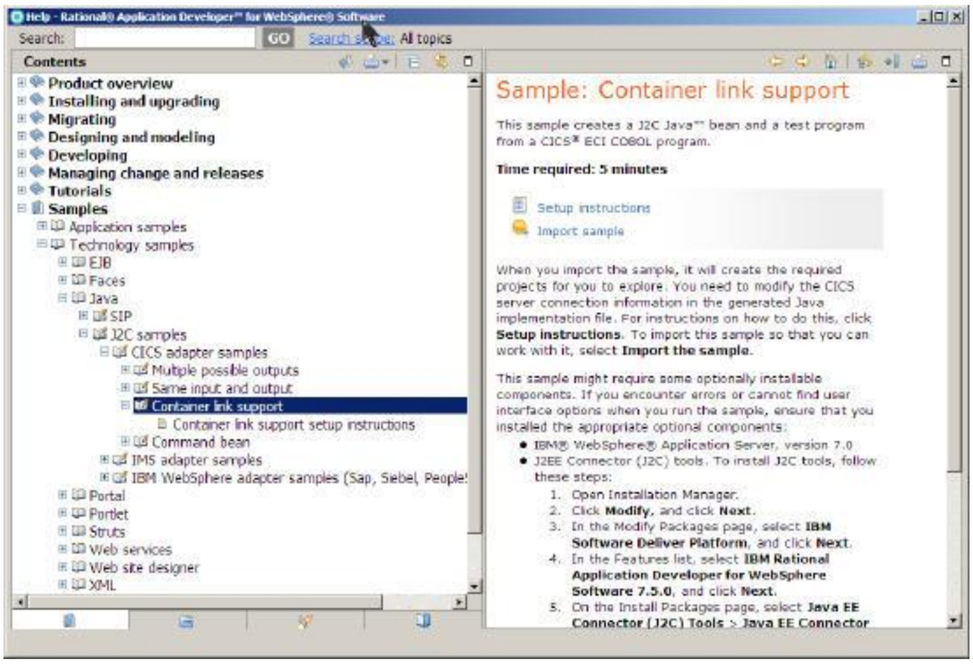
1. CICS IMS Data Binding wizard: It will generate one or more data binding files.
2. J2C Java Bean wizard: It will generate J2C beans, import resource adapters as well as data binding files.

Samples Support

There are prebuilt samples where you can import into your workspace and explore the generated code. You can execute the generated code if you install the Cobol CICS channel copy book to your CICS system.

The sample is located in **Help > Samples > Technology samples > Java > J2C Samples > CICS Adapter samples > Container link support**

Figure 0. CICS container Sample


Help - Rational® Application Developer™ for WebSphere® Software
Search:
GO
Search scope: All topics
Contents

- Product overview
- Installing and upgrading
- Migrating
- Designing and modeling
- Developing
- Managing change and releases
- Tutorials
- Samples
  - Application samples
  - Technology samples
    - EJB
    - Faces
    - Java
      - SIP
      - J2C samples
        - CICS adapter samples
          - Multiple possible outputs
          - Same input and output
          - Container link support
            - Container link support setup instructions
          - Command bean
        - IMS adapter samples
        - IBM WebSphere adapter samples (Sap, Siebel, People!
    - Portal
    - Portlet
    - Struts
    - Web services
    - Web site designer
    - XML

Sample: Container link support

This sample creates a J2C Java™ bean and a test program from a CICS® ECI COBOL program.

Time required: 5 minutes

Setup instructions
Import sample

When you import the sample, it will create the required projects for you to explore. You need to modify the CICS server connection information in the generated Java implementation file. For instructions on how to do this, click **Setup instructions**. To import this sample so that you can work with it, select **Import the sample**.

This sample might require some optionally installable components. If you encounter errors or cannot find user interface options when you run the sample, ensure that you installed the appropriate optional components:

- IBM® WebSphere® Application Server, version 7.0
- J2EE Connector (J2C) tools. To install J2C tools, follow these steps:
  1. Open Installation Manager.
  2. Click **Modify**, and click **Next**.
  3. In the Modify Packages page, select **IBM Software Deliver Platform**, and click **Next**.
  4. In the Features list, select **IBM Rational Application Developer for WebSphere Software 7.5.0**, and click **Next**.
  5. On the Install Packages page, select **Java EE Connector (J2C) Tools > Java EE Connector**

Deployment Support

The deployment support is not specific to the CICS Container Link. It is a generic support for all generated J2C Java beans since Rational Application Developer for WebSphere Software v7. There is a deployment wizard in J2C Tools for you to deploy the generated J2C Java bean into faces JSP, simple JSP, Web Service and EJBs. You can just use simple JSP to unit test your generated code without writing any client code.

There are two flows for deployment:

- You can go through the deployment as part of the flow when you create J2C Java beans.
- You can create J2C Java beans and invoke the deployment wizard separately.
- The J2C ant script generated during wizard flow does not include deployment information. It only records options for code generation.

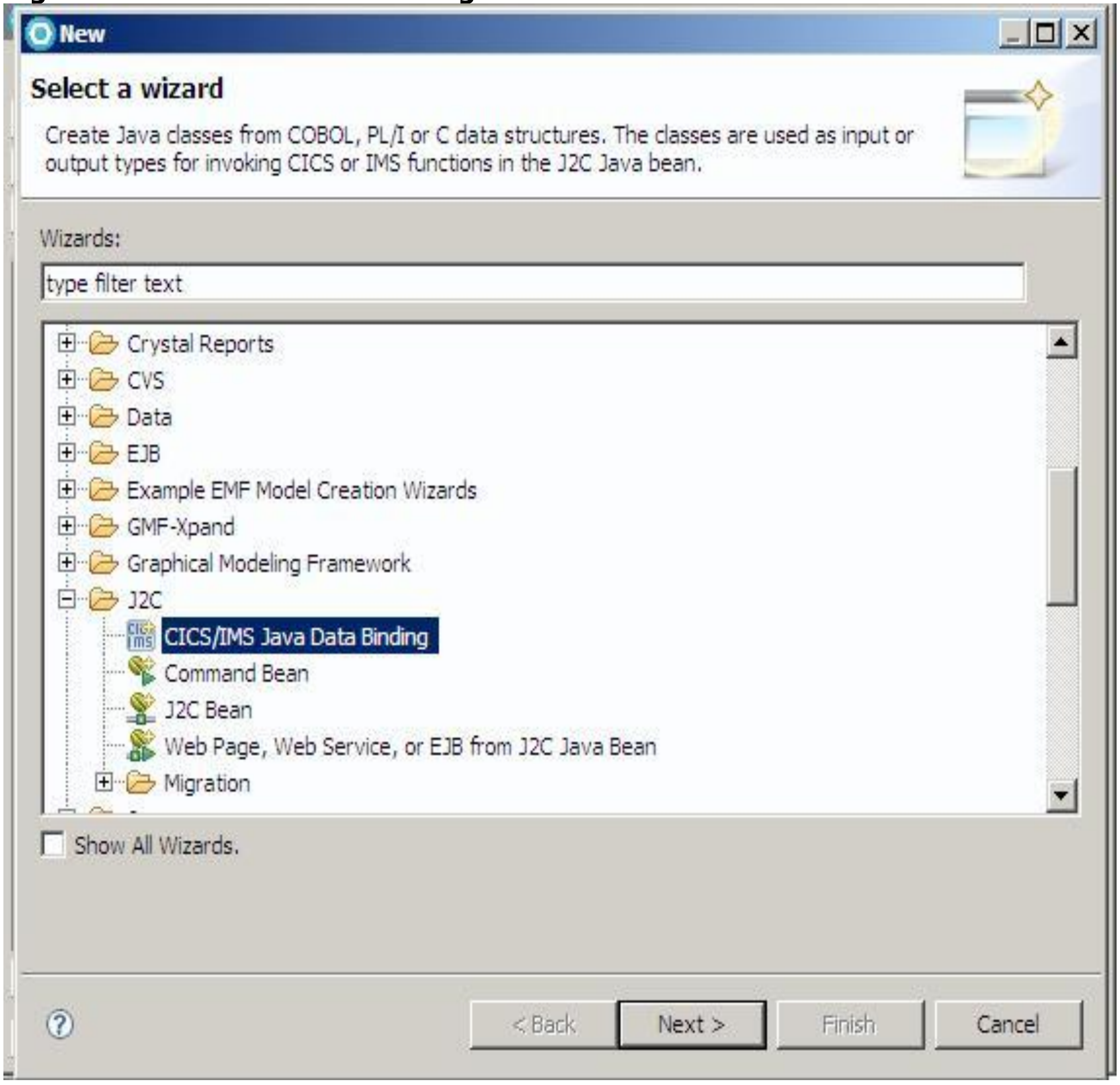
Data Binding generation in CICS/IMS Data Binding wizard

The CICS IMS Data Binding wizard allows you to generate data binding files based on the Cobol, C or PL/1 source. When you go through the wizard, you have to select different options in the wizard before you can generate the data binding files. The Ant script support will save all the settings in the data binding Ant script file.

Figure 1 shows you how to launch the CICS IMS Data Binding wizard.

1. Select **File > New > Other**.
2. Select **J2C > CICS IMS data Binding**.

Figure 1. CICS IMS Data Binding wizard



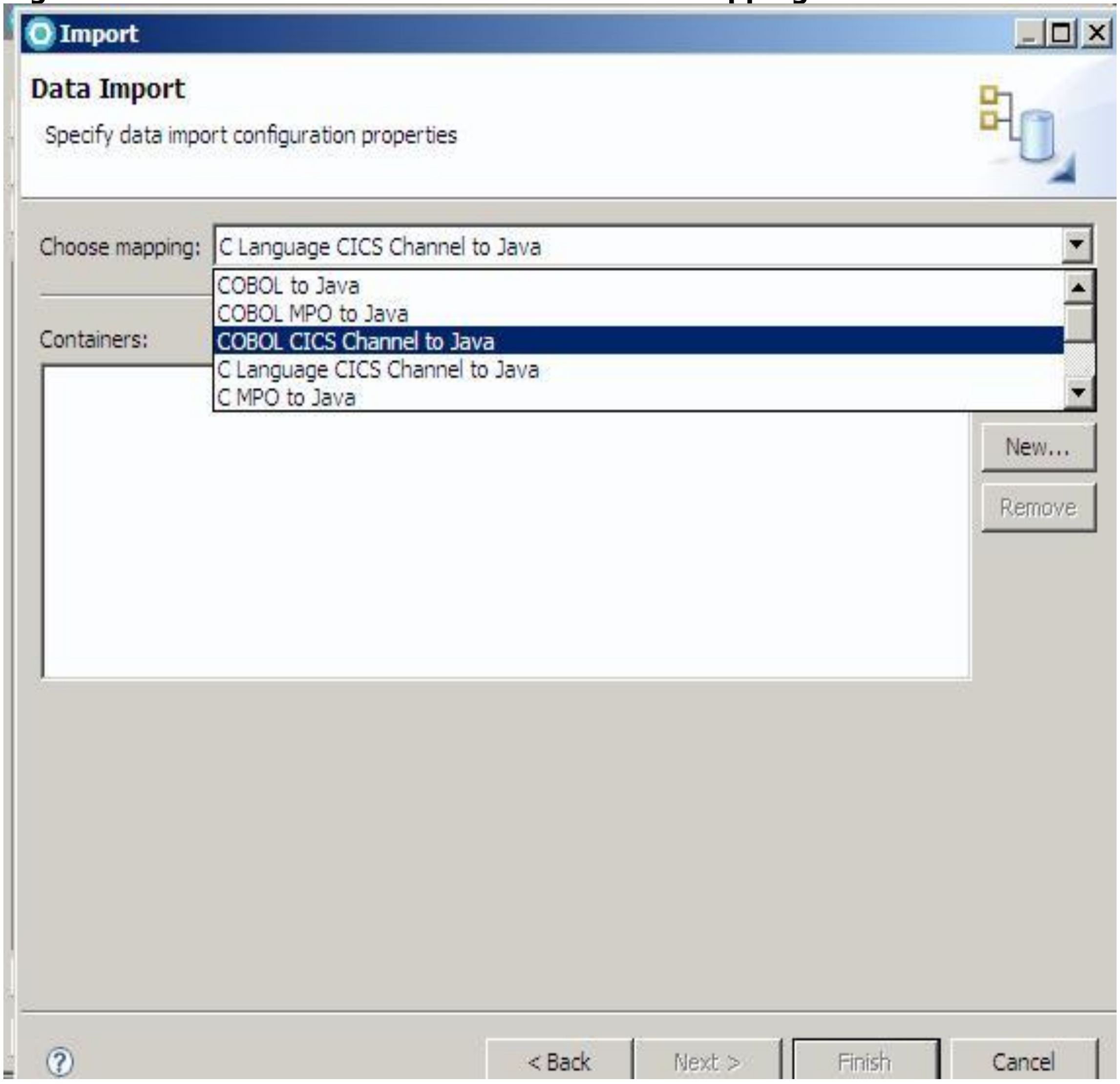
Select a wizard

Create Java classes from COBOL, PL/I or C data structures. The classes are used as input or output types for invoking CICS or IMS functions in the J2C Java bean.

New
Wizards:
type filter text
Crystal Reports
CVS
Data
EJB
Example EMF Model Creation Wizards
GMF-Xpand
Graphical Modeling Framework
J2C
CICS/IMS Java Data Binding
Command Bean
J2C Bean
Web Page, Web Service, or EJB from J2C Java Bean
Migration
Show All Wizards.
< Back
Next >
Finish
Cancel

1. Click **Next** in the CICS IMS Data Binding wizard to launch the Data Import page.
2. Choose **COBOL CICS Channel to Java**, **C Language CICS Channel to Java** or **PL1 CICS Channel to Java** mapping.

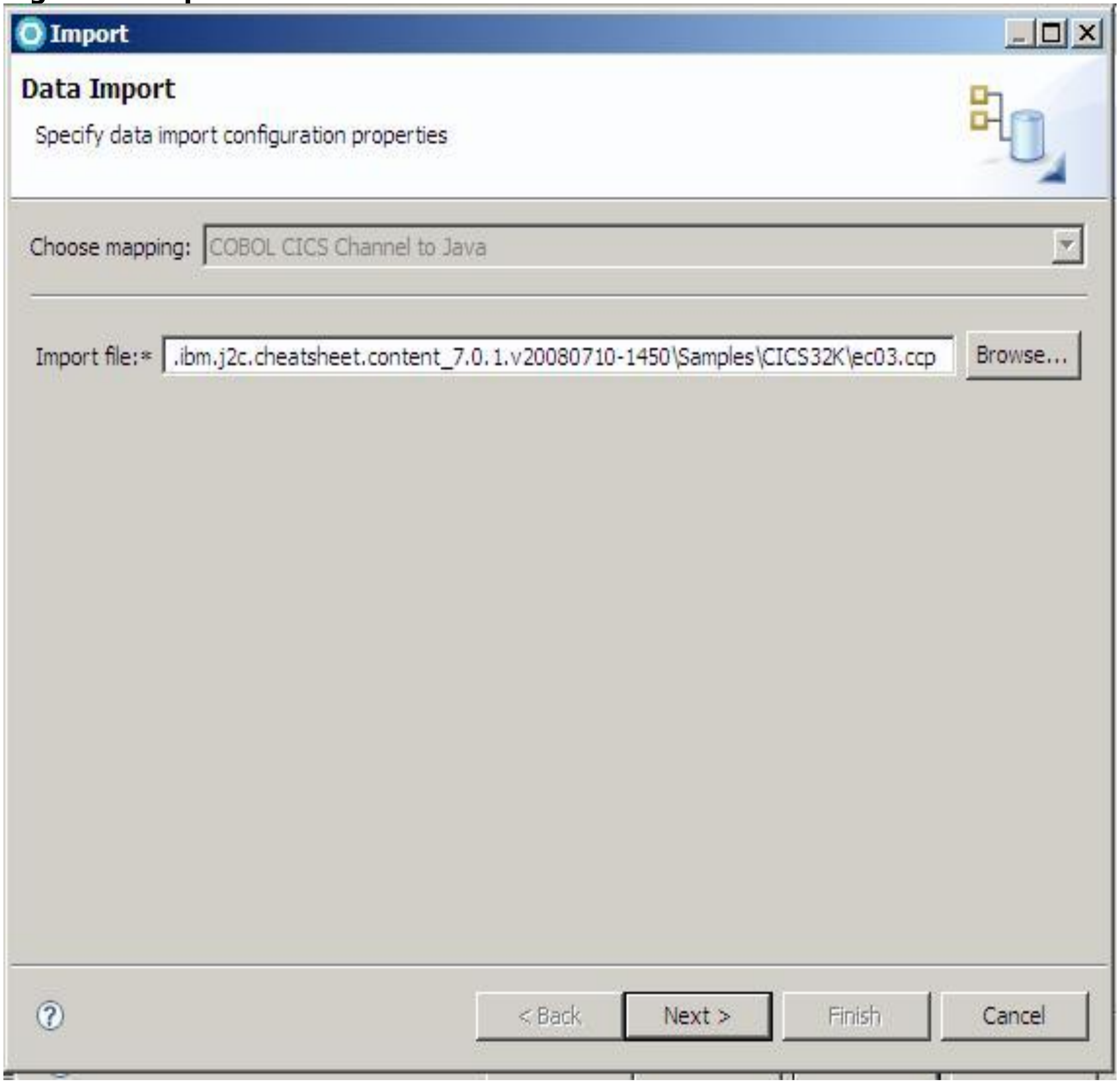
Figure 2. Choose COBOL CICS Channel to Java Mapping


Import
Data Import
Specify data import configuration properties
Choose mapping:
C Language CICS Channel to Java
COBOL to Java
COBOL MPO to Java
COBOL CICS Channel to Java
C Language CICS Channel to Java
C MPO to Java
Containers:
New...
Remove
< Back
Next >
Finish
Cancel

3. Select the Cobol, PL1, or C source file.

   There is a sample CICS Channel Cobol source ec03.cpp located in the Rational Application Developer for WebSphere Software install shared folder under the plug-in com.ibm.j2c.cheatsheet.content.v7.0.1.xxxx.

Figure 3. Import Cobol source



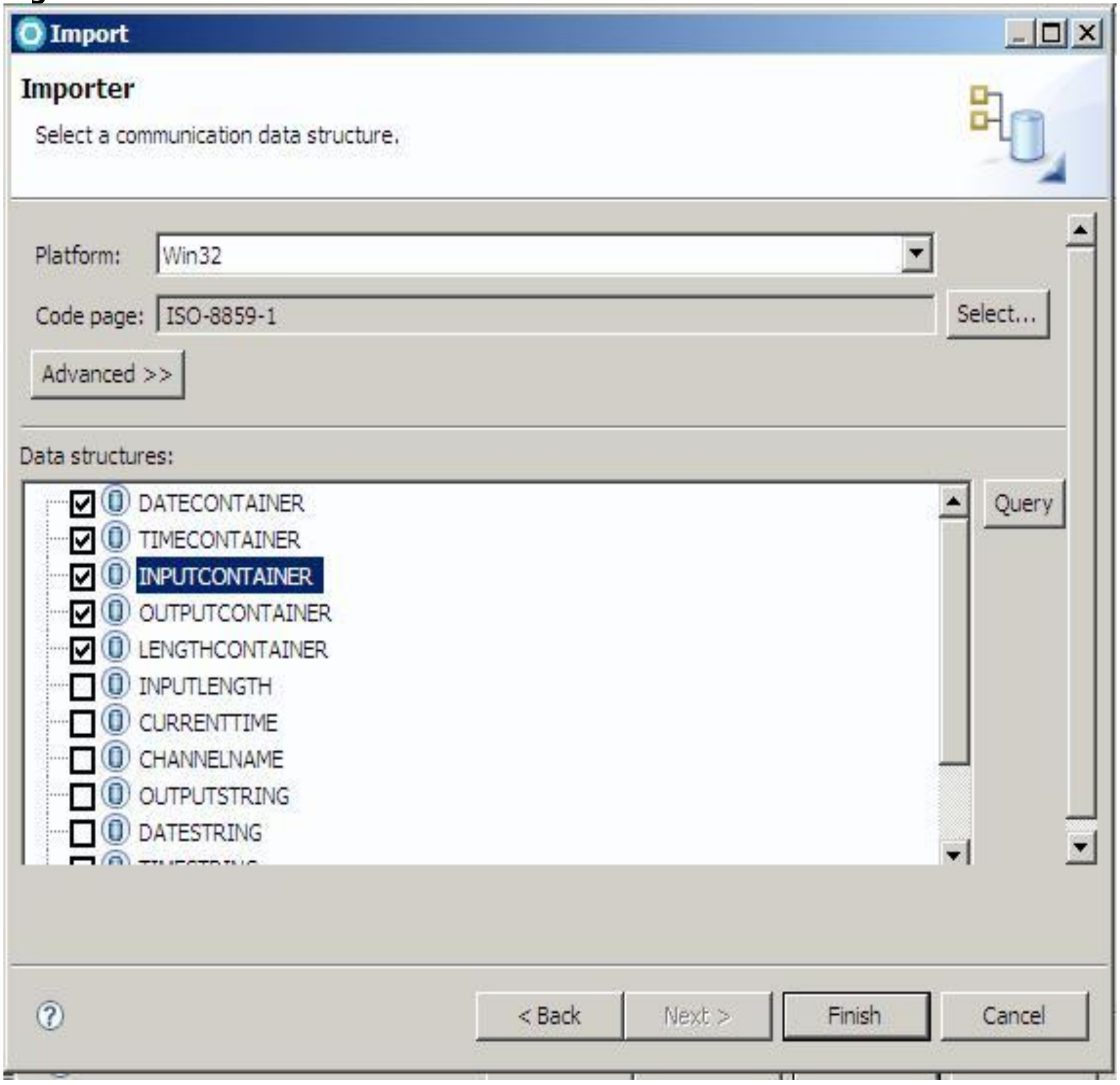
Choose mapping: COBOL CICS Channel to Java

Import file:* .ibm.j2c.cheatsheet.content_7.0.1.v20080710-1450\Samples\CICS32K\ec03.ccp

4. Click **Next** to launch the Importer page.
5. Select the importer settings and the data structure to be imported.

   For the sample ec03.cpp, select *DATECONTAINER*,*INPUTCONTAINER*,*TIMECONTAINER*,*OUTPUTCONTAINER* and *LENGTHCONTAINER* from the data structures section.

Figure 4. Select data structure from Cobol source



Import

Importer

Select a communication data structure.

Platform: Win32

Code page: ISO-8859-1

Data structures:

- [✓] DATECONTAINER
- [✓] TIMECONTAINER
- [✓] INPUTCONTAINER
- [✓] OUTPUTCONTAINER
- [✓] LENGTHCONTAINER
- [ ] INPUTLENGTH
- [ ] CURRENTTIME
- [ ] CHANNELNAME
- [ ] OUTPUTSTRING
- [ ] DATESTRING

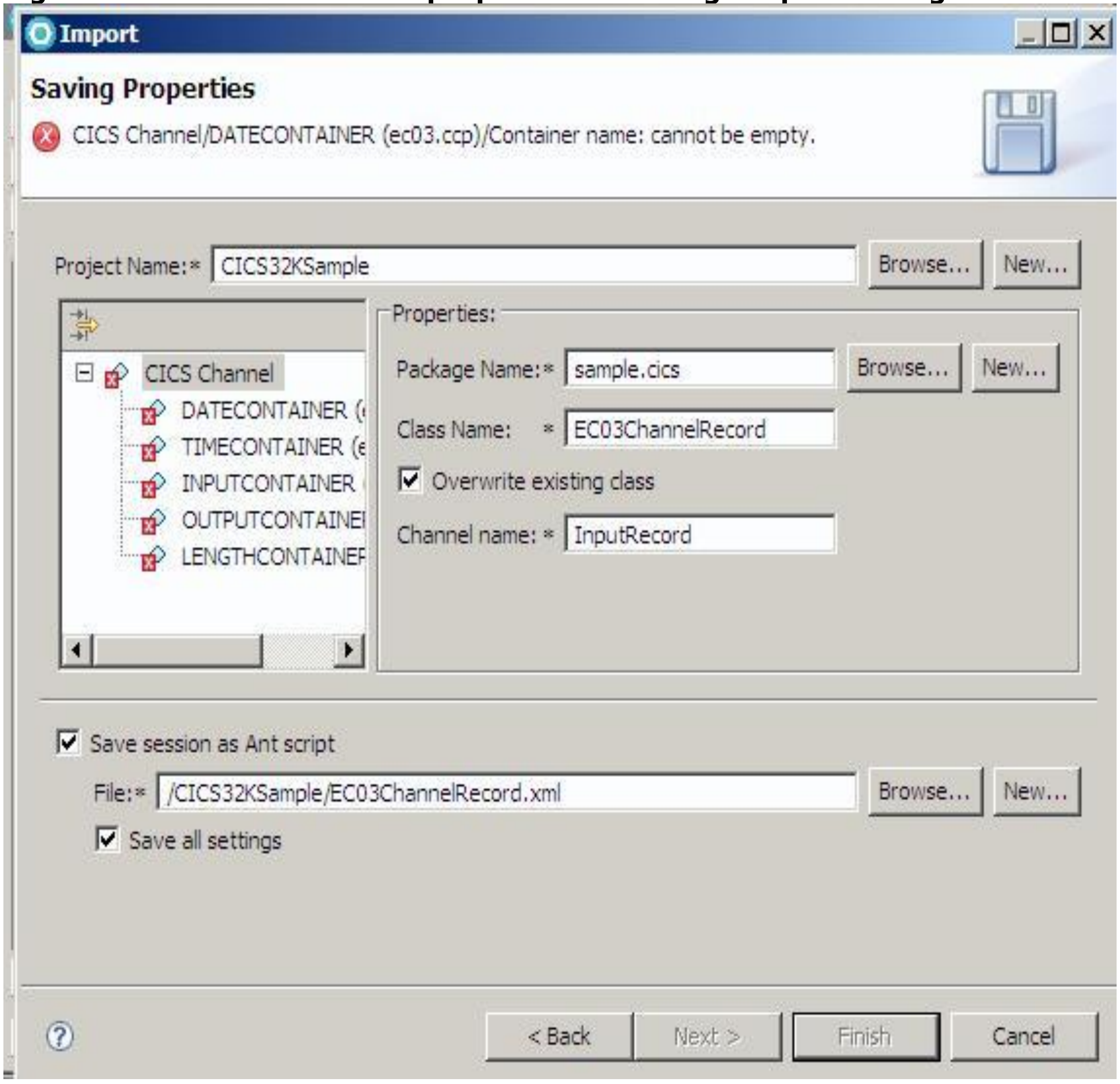
6. Click **Next** to launch the Saving Properties Page shown in Figure 5.

   Click on the CICS Channel, enter the Project name, Package Name, Class Name and Channel Name. Enter **EC03ChannelRecord** as CICS channel java class name and **InputRecord** as channel Name.

   You will select the **Save session as Ant script** checkbox to use the Ant script feature. The default Data Binding Ant Script file name is the name of the Channel Data binding class name. The default location of Ant script will be the current Project where the Channel Data Binding is generated. You can change the name and location for the Ant script generation.

   The **Save All Settings** option is useful if you would like to generate the values for all the settings when you go through the wizard. In this way, it is like you have generated a template for all the properties. If you do not select the **Save All Settings** option, only the option that you modify or enter when you go through the CICS IMS Data Binding wizard will be generated. For example, the default value for **Generation Style** is **Default**. The **Generation Style** will be generated in the Ant script only if you select another style like **Shorten names**.

Figure 5.Enter CICS Channel properties in Saving Properties Page



Import

Saving Properties

CICS Channel/DATECONTAINER (ec03.ccp)/Container name: cannot be empty.

Project Name:* CICS32KSample

CICS Channel
- DATECONTAINER (
- TIMECONTAINER (
- INPUTCONTAINER
- OUTPUTCONTAINE
- LENGTHCONTAINER

Properties:

Package Name:* sample.cics

Class Name: * EC03ChannelRecord

☑ Overwrite existing class

Channel name: * InputRecord

☑ Save session as Ant script

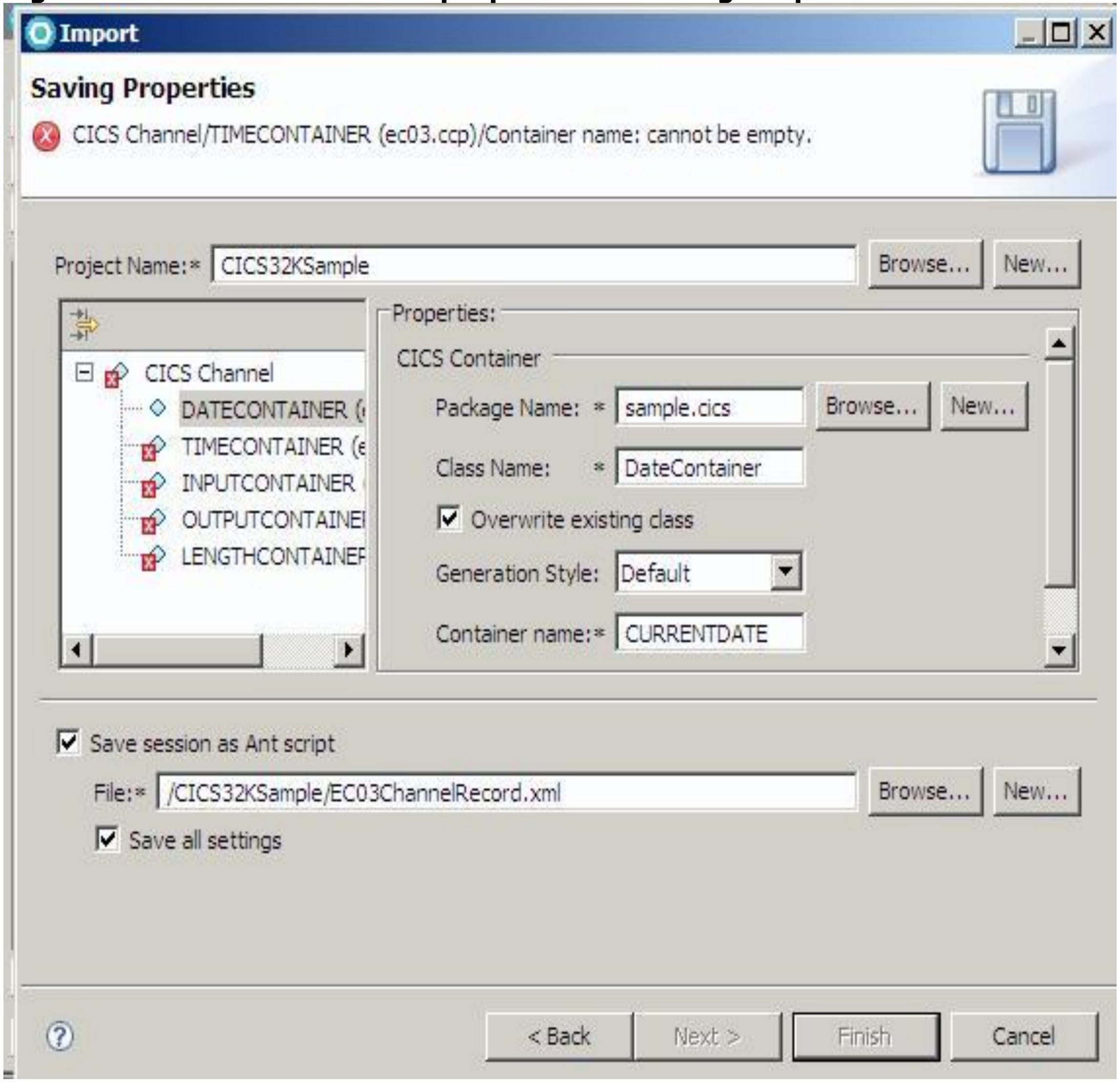
File:* /CICS32KSample/EC03ChannelRecord.xml

☑ Save all settings

For each of the data structures (Container) under the CICS channel, enter the corresponding Package Name, Class Name, Channel Name and Container names. Select the Container Type for each of the Container as indicated below.

Data Structure	Container Name	Container Class Name	Container Type
DATECONTAINER	CURRENTDATE	DateContainer	CHAR
TIMECONTAINER	CURRENTTIME	TimeContainer	CHAR
OUTPUTCONTAINER	OUTPUTMESSAGE	OutputContainer	CHAR
INPUTCONTAINER	INPUTDATA	InputContainer	CHAR
LENGTHCONTAINER	INPUTDATALENGTH	LengthContainer	BIT

Figure 6.Enter DateContainer properties in Saving Properties



CICS Channel/TIMECONTAINER (ec03.ccp)/Container name: cannot be empty.

Project Name:* CICS32KSample

Package Name: * sample.cics

Class Name: * DateContainer

☑ Overwrite existing class

Generation Style: Default

Container name:* CURRENTDATE

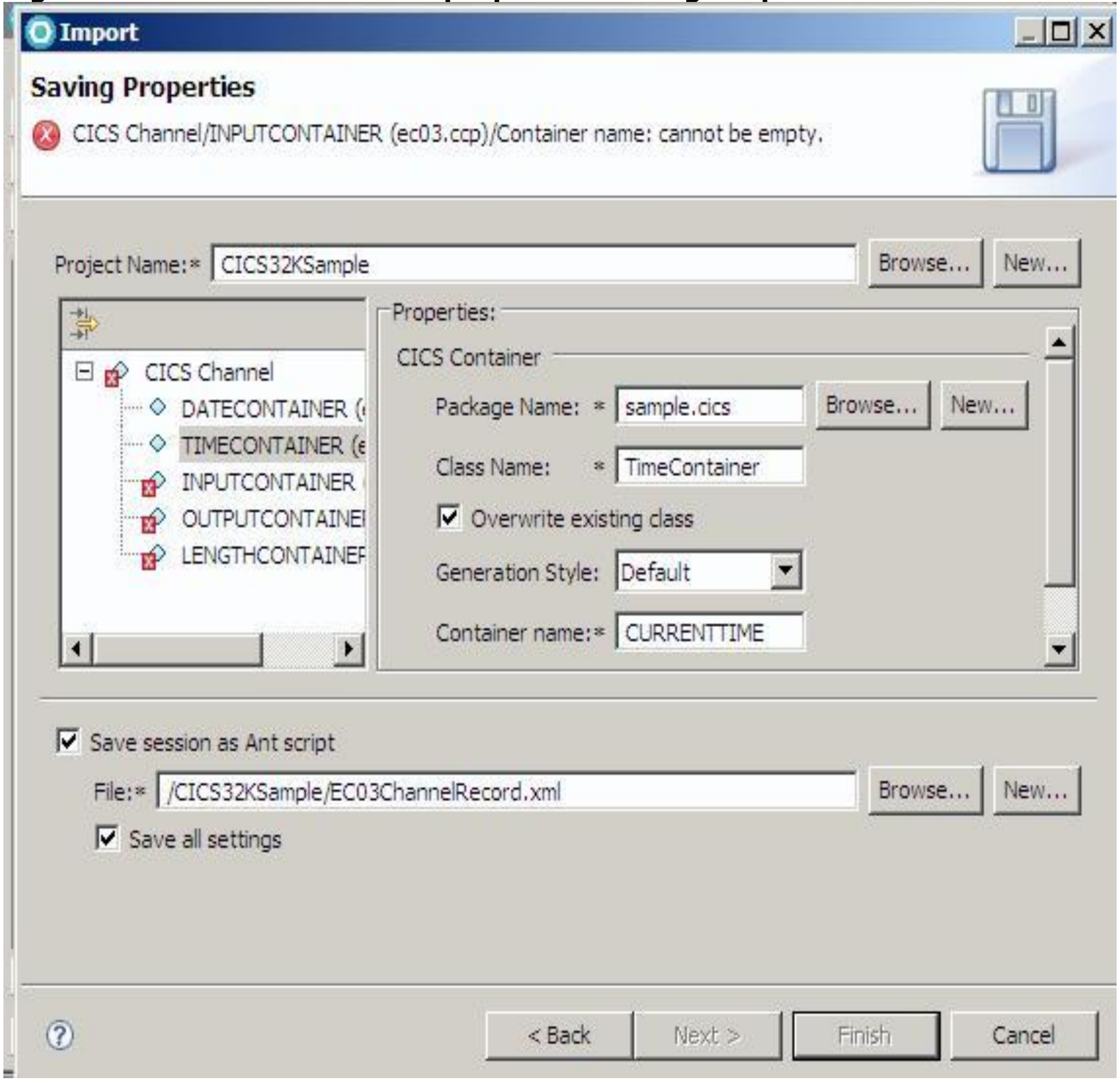
☑ Save session as Ant script

File:* /CICS32KSample/EC03ChannelRecord.xml

☑ Save all settings

Enter **TimeContainer** as Java class Name

Figure 7.Enter TimeContainer properties Saving Properties



Import

Saving Properties

CICS Channel/INPUTCONTAINER (ec03.ccp)/Container name: cannot be empty.

Project Name:* CICS32KSample

CICS Channel
- DATECONTAINER (
- TIMECONTAINER (e
- INPUTCONTAINER
- OUTPUTCONTAINE
- LENGTHCONTAINEF

Properties:

CICS Container

Package Name: * sample.cics

Class Name: * TimeContainer

[✓] Overwrite existing class

Generation Style: Default

Container name:* CURRENTTIME

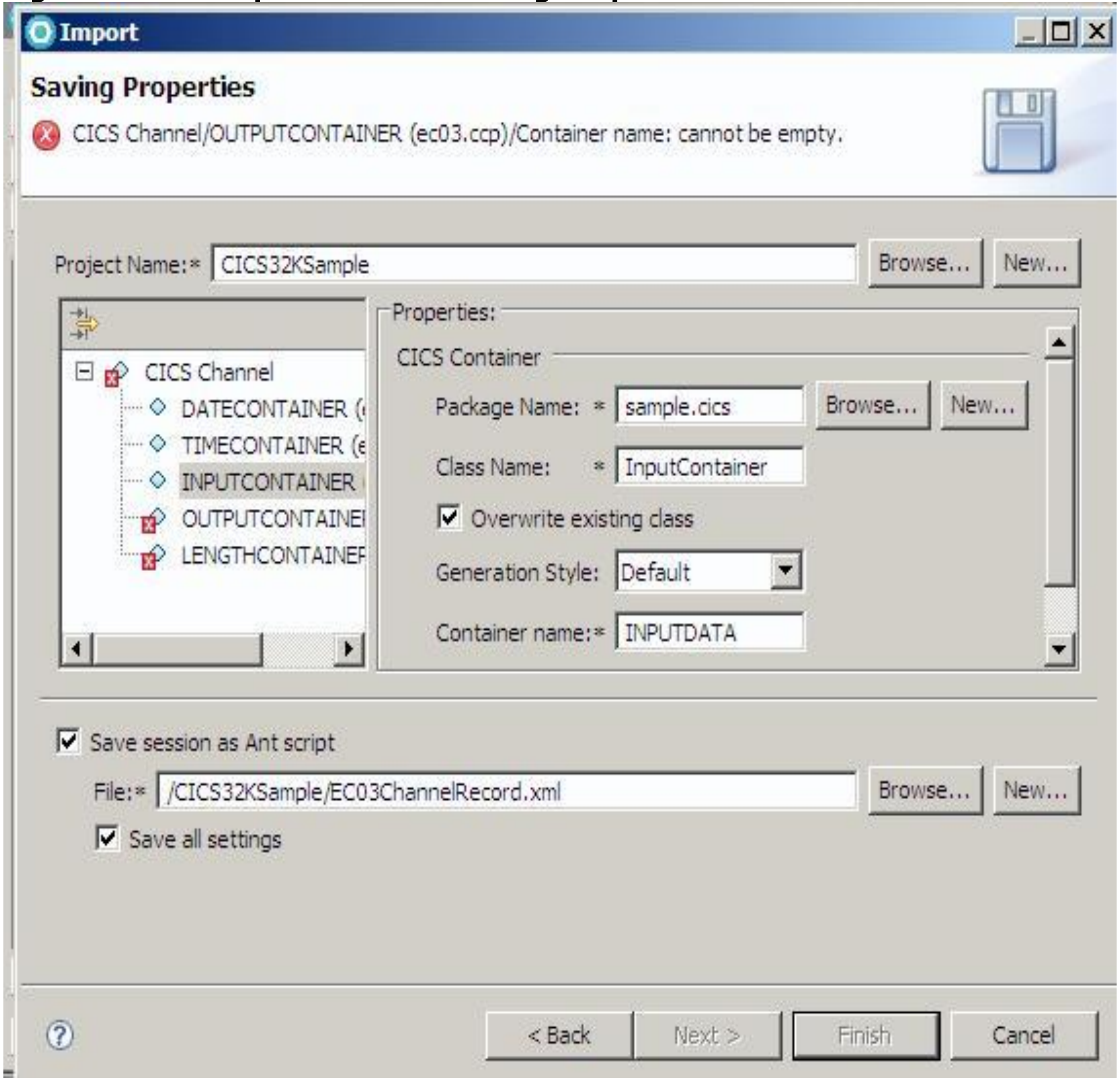
[✓] Save session as Ant script

File:* /CICS32KSample/EC03ChannelRecord.xml

[✓] Save all settings

Enter **InputContainer** as Container Java class Name

Figure 8.Enter InputContainer Saving Properties



Import

Saving Properties

CICS Channel/OUTPUTCONTAINER (ec03.ccp)/Container name: cannot be empty.

Project Name:* CICS32KSample

Package Name: * sample.cics

Class Name: * InputContainer

[✓] Overwrite existing class

Generation Style: Default

Container name:* INPUTDATA

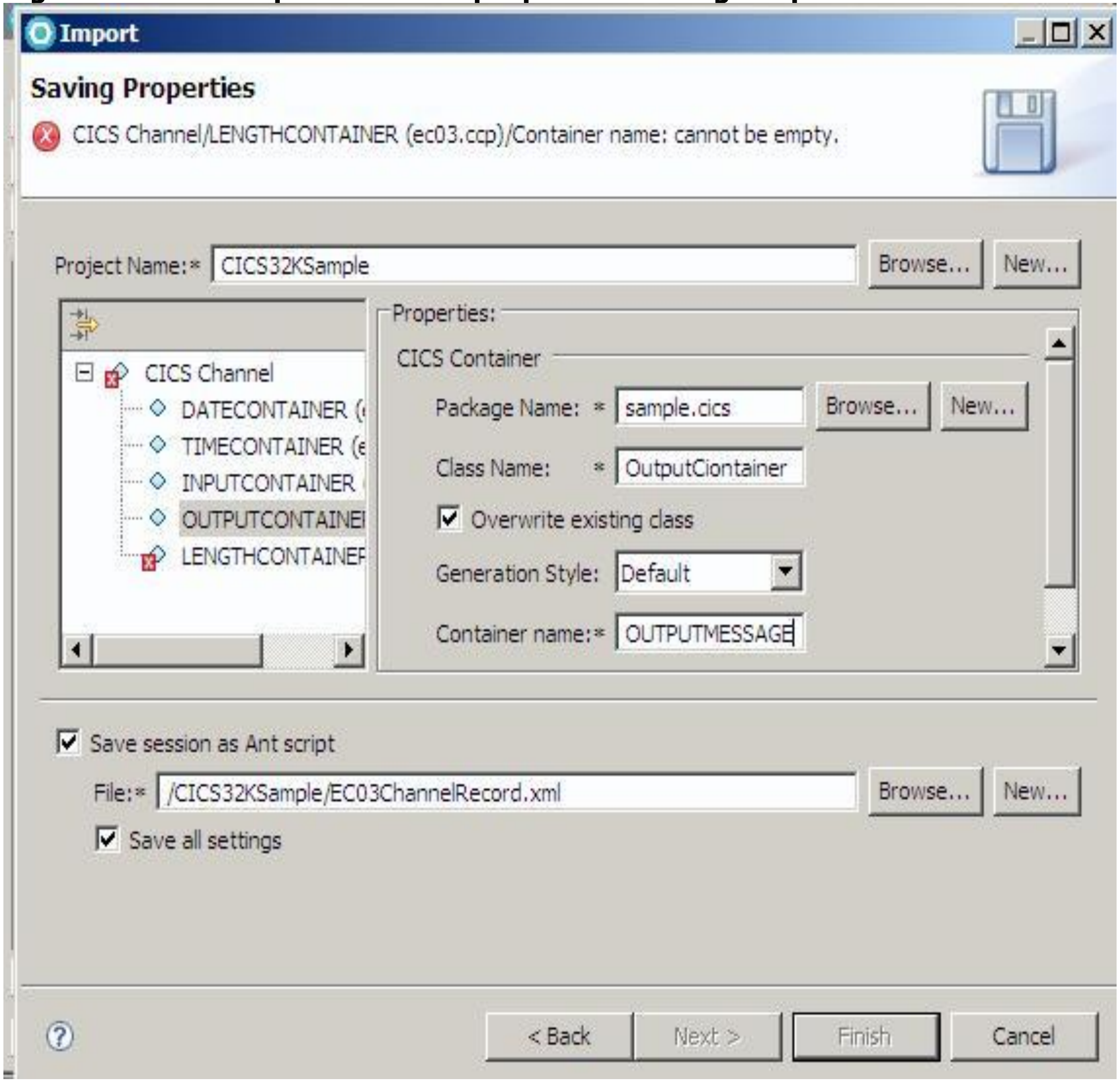
[✓] Save session as Ant script

File:* /CICS32KSample/EC03ChannelRecord.xml

[✓] Save all settings

Enter **OutputContainer** as Container Java class Name

Figure 9.Enter OutputContainer properties Saving Properties



Import

Saving Properties

CICS Channel/LENGTHCONTAINER (ec03.ccp)/Container name: cannot be empty.

Project Name:* CICS32KSample

CICS Channel
- DATECONTAINER (
- TIMECONTAINER (
- INPUTCONTAINER
- OUTPUTCONTAINE
- LENGTHCONTAINEF

Properties:

CICS Container

Package Name: * sample.cics

Class Name: * OutputCiontainer

☑ Overwrite existing class

Generation Style: Default

Container name:* OUTPUTMESSAGE

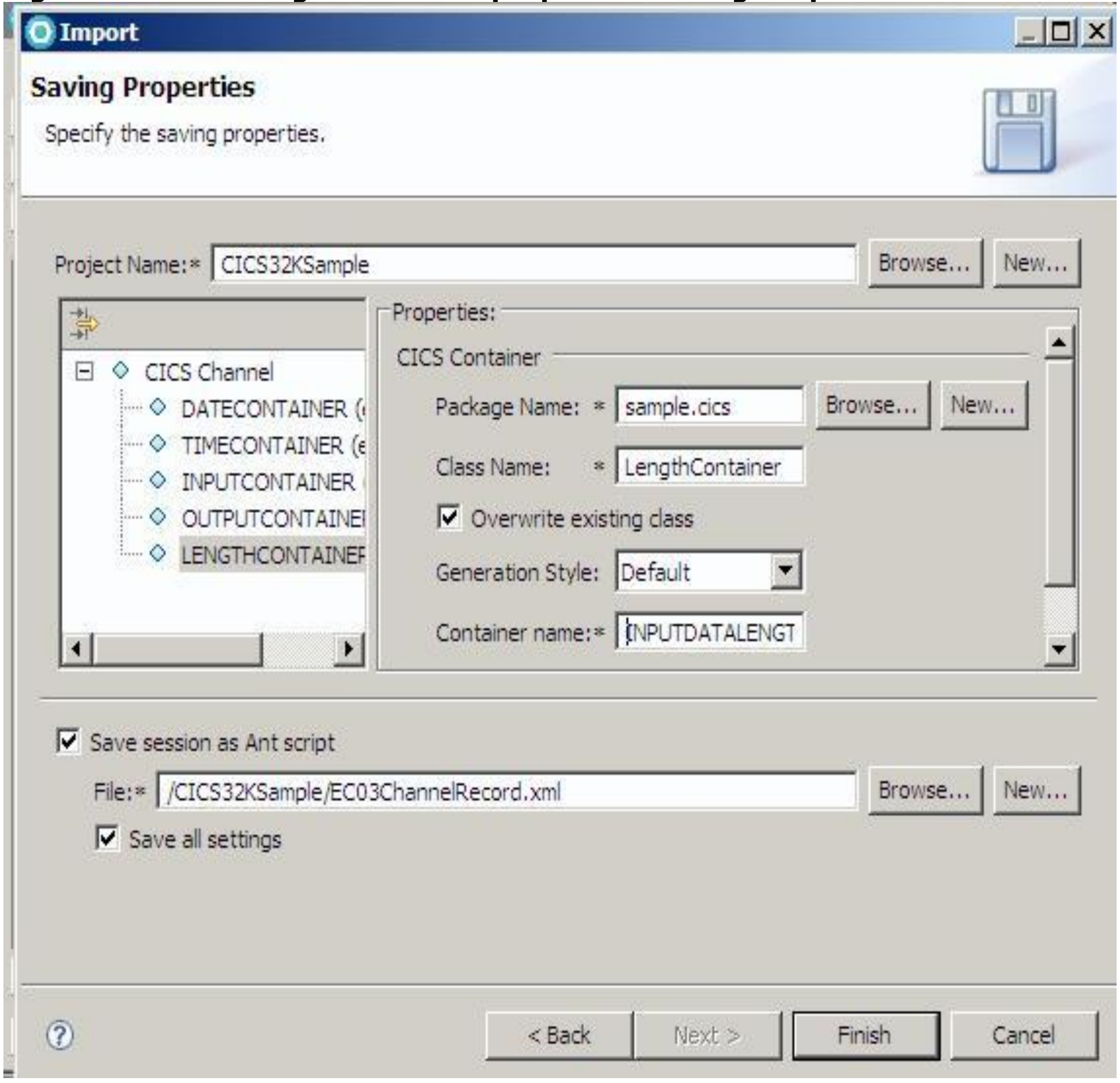
☑ Save session as Ant script

File:* /CICS32KSample/EC03ChannelRecord.xml

☑ Save all settings

Enter **LengthContainer** as Container Java class Name

Figure 10.Enter LengthContainer properties Saving Properties



Import

Saving Properties

Specify the saving properties.

Project Name:* CICS32KSample

CICS Channel
- DATECONTAINER (
- TIMECONTAINER (e
- INPUTCONTAINER
- OUTPUTCONTAINEI
- LENGTHCONTAINEF

Properties:

CICS Container

Package Name: * sample.cics

Class Name: * LengthContainer

[✓] Overwrite existing class

Generation Style: Default

Container name:* INPUTDATALENGT

[✓] Save session as Ant script

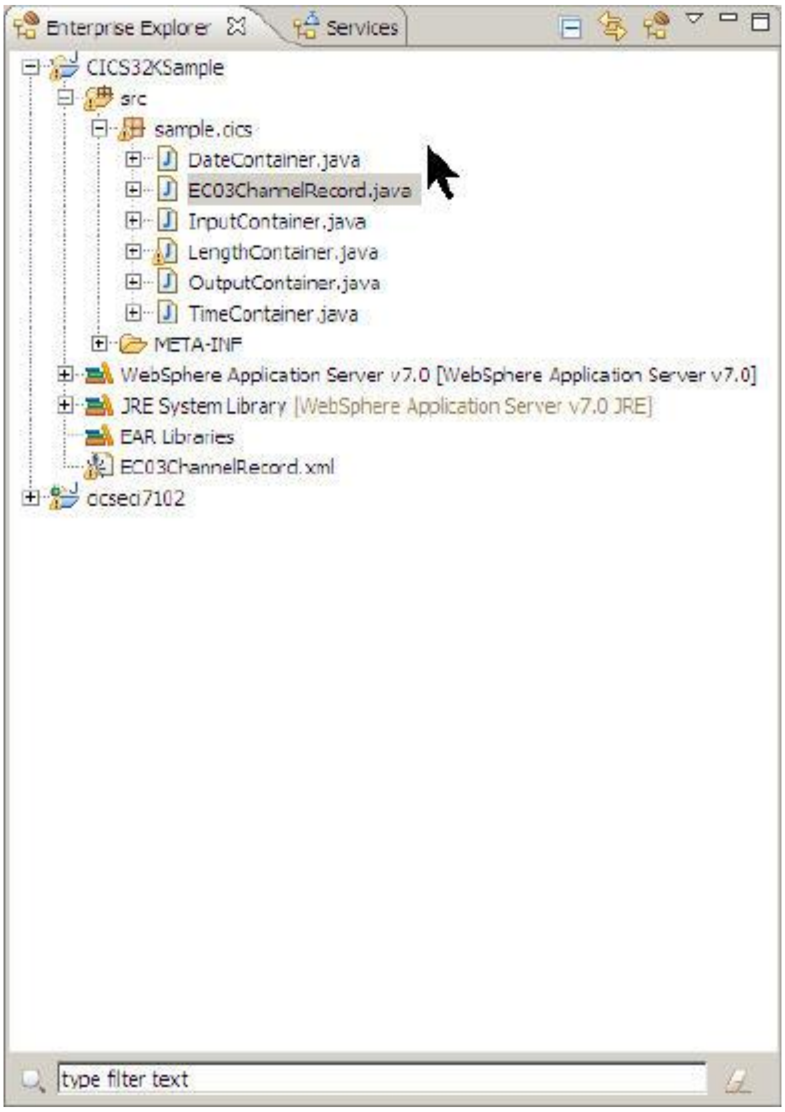
File:* /CICS32KSample/EC03ChannelRecord.xml

[✓] Save all settings

7. Click the **Finish** button to generate the Data Binding files as well as the ant script.

Generated Code

- For each of the container, there is one Java data binding generated.
- The generated CICS Channel class **EC03ChannelRecord.java** acts as a wrapper class for all the containers. It has getter and setter for each of the container class.
- The CICS Channel class will be used as input and output data type.



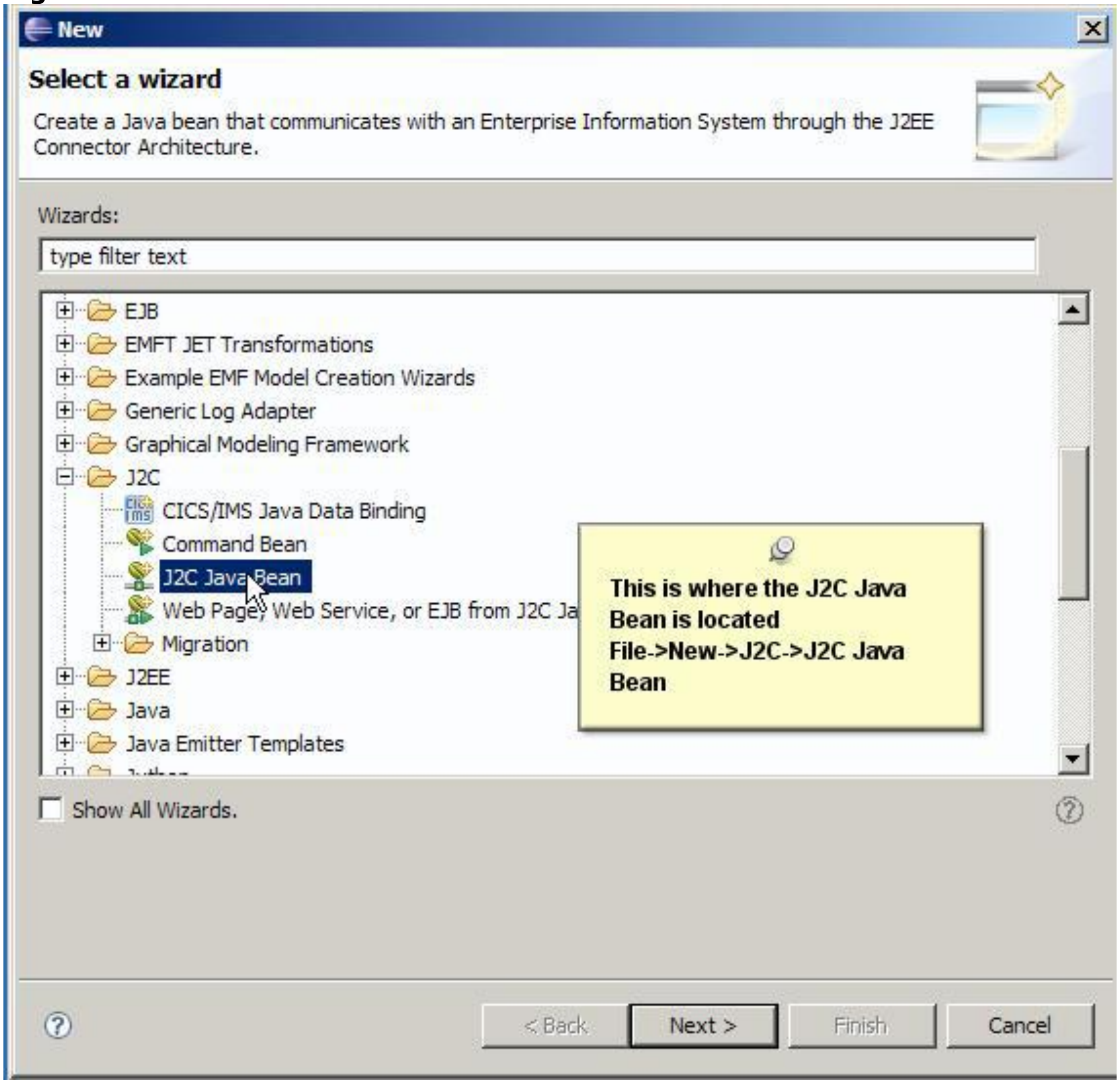
J2C Java bean generation that supports CICS Channel commands

The J2C Java Bean wizard creates a bean that communicates with an Enterprise Information System. Figure 4 shows where the J2C Java Bean wizard is under the J2C folder.

To find the J2C Java Bean wizard shown in Figure 11:

1. Select **File > New > Other**.
2. Select **J2C > J2C Java Bean wizard**.

Figure 11. J2C Java Bean Wizard



New

Select a wizard

Create a Java bean that communicates with an Enterprise Information System through the J2EE Connector Architecture.

Wizards:

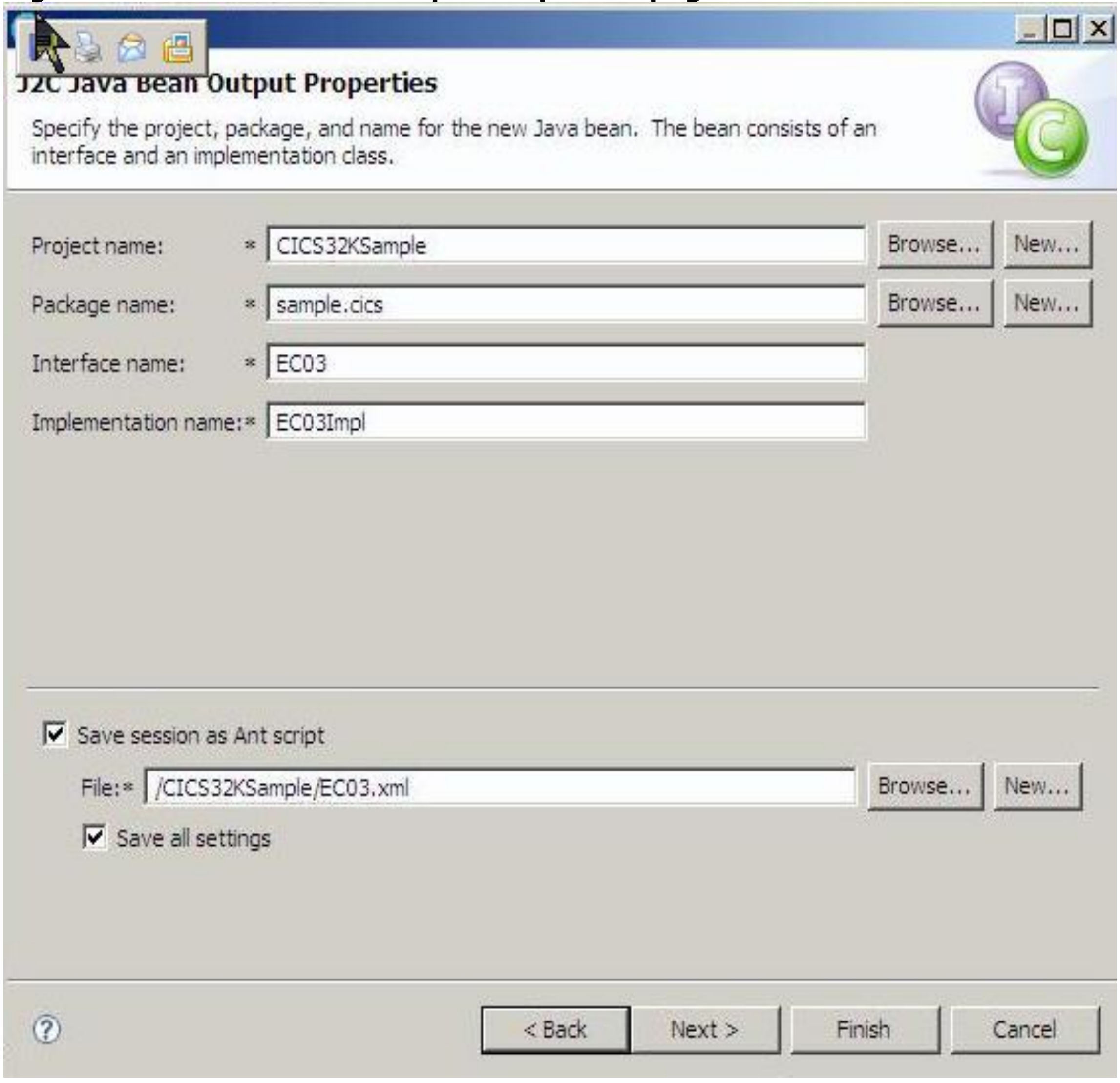
type filter text

This is where the J2C Java Bean is located File->New->J2C->J2C Java Bean

Figure 12 shows the **Save session as Ant Script** support in the J2C Java Bean Output Properties page. To go to this page:

1. Select **J2C Java Bean wizard** and click Next.
2. In the Resource Adapter Selection page, select the resource adapter **ECIResourceAdapter v7.1.0.2** and click **Next**. The **ECIResourceAdapter v7.1.0.2** provides CICS Channel Support.
3. The Connection Properties page launches. Enter the connection information and click **Next**.
4. You will now go to the J2C Java Bean Output Properties page.

Figure 12. J2C Java Bean Output Properties page



J2C Java Bean Output Properties

Specify the project, package, and name for the new Java bean. The bean consists of an interface and an implementation class.

Project name: * CICS32KSample

Package name: * sample.cics

Interface name: * EC03

Implementation name: * EC03Impl

☑ Save session as Ant script

File: * /CICS32KSample/EC03.xml

☑ Save all settings

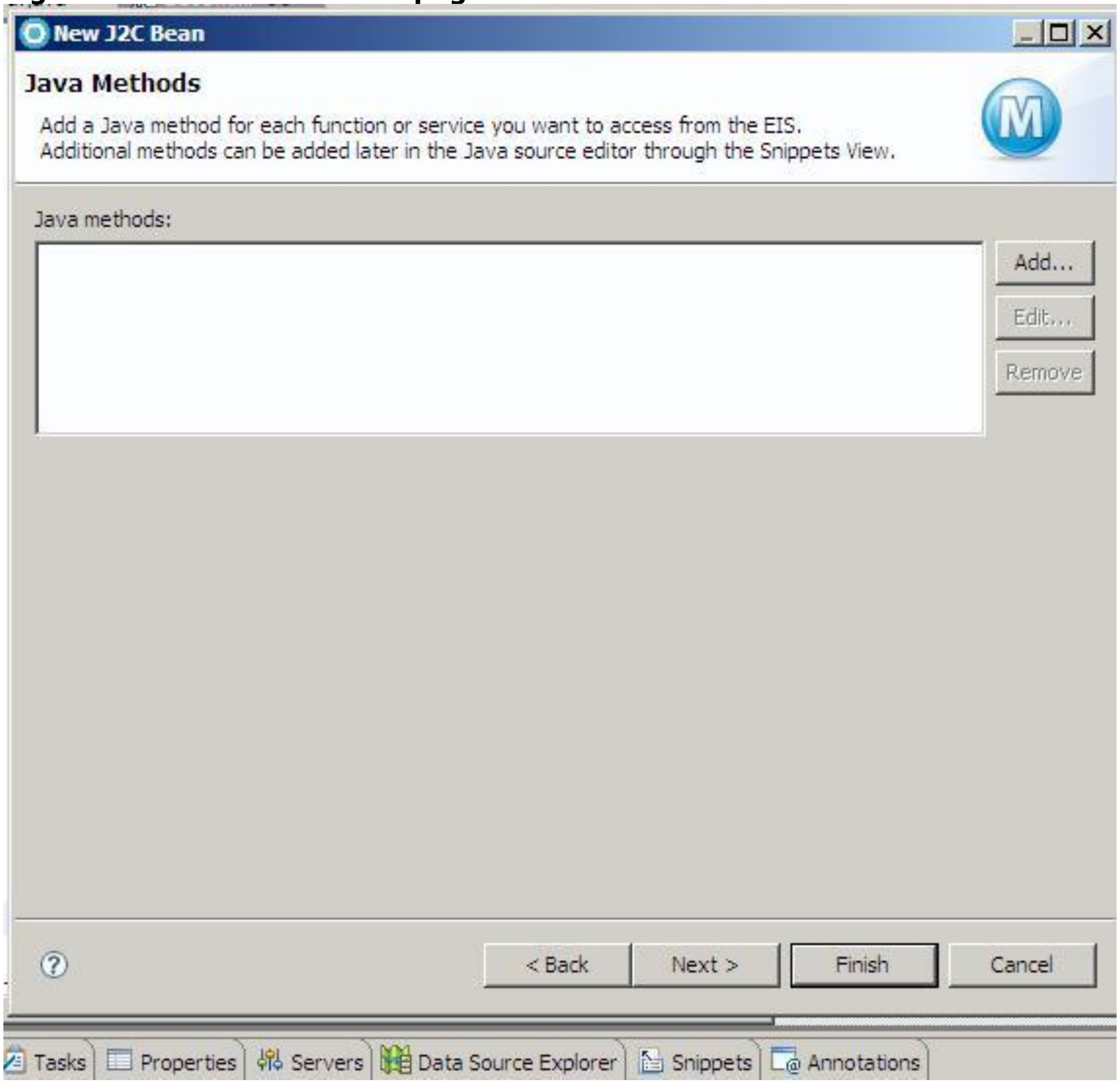
In the J2C Java Bean Output Properties page, specify the name of your J2C Java Bean Interface and Implementation file, as well as the project name and package name of where they will be generated.

Select the **Save session as Ant script** checkbox to use the Ant script feature.

The default J2C Java Bean Ant Script file name is the name of the J2C Java Bean interface. The default location of the Ant script will be the current Project where the J2C bean is generated. You can change the name and location for the Ant script generation. The **Save All Settings** option is useful if you would like to generate the values for all the settings when you go through the wizard. This is a good way to capture for informational purposes all of the values that were used, and it makes it easier for you to change a particular value in the future. If you do not select the **Save All Settings** option, only the option that you modify or enter when you go through the J2C Java Bean wizard will be generated.

After you select the **Save session as Ant script** option, click **Next** to go to the Java Methods Page.

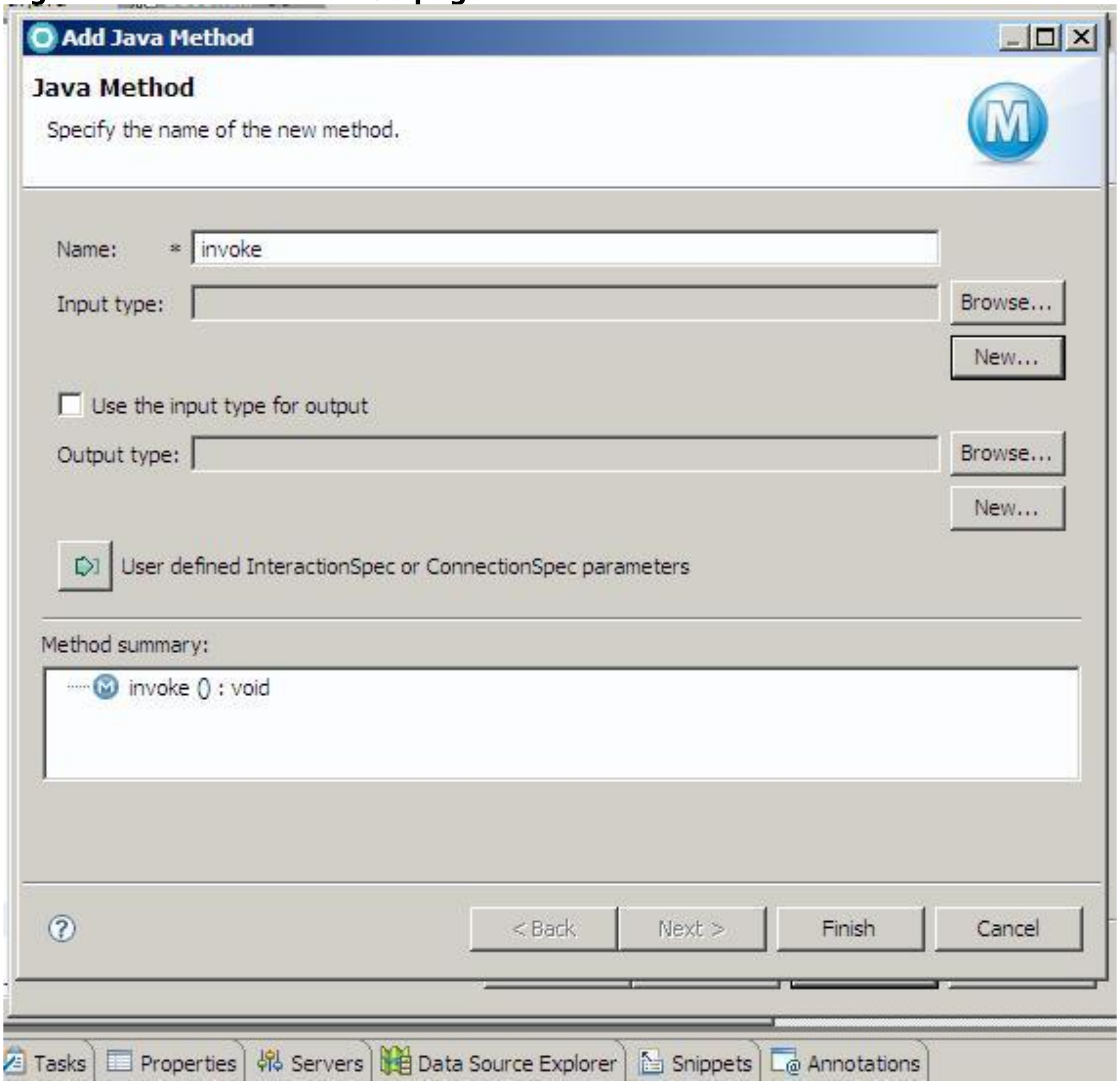
Figure 13. J2C Java Method page


New J2C Bean
Java Methods
Add a Java method for each function or service you want to access from the EIS.
Additional methods can be added later in the Java source editor through the Snippets View.
Java methods:
Add...
Edit...
Remove
< Back
Next >
Finish
Cancel
Tasks
Properties
Servers
Data Source Explorer
Snippets
Annotations

Click the **Add** button in the Java Methods page to create the Java Method.

In the Java Method Page show in figure 14, enter **invoke** as the method Name.

Figure 14. J2C Java Method page



Name: * invoke

Input type:

☐ Use the input type for output

Output type:

User defined InteractionSpec or ConnectionSpec parameters

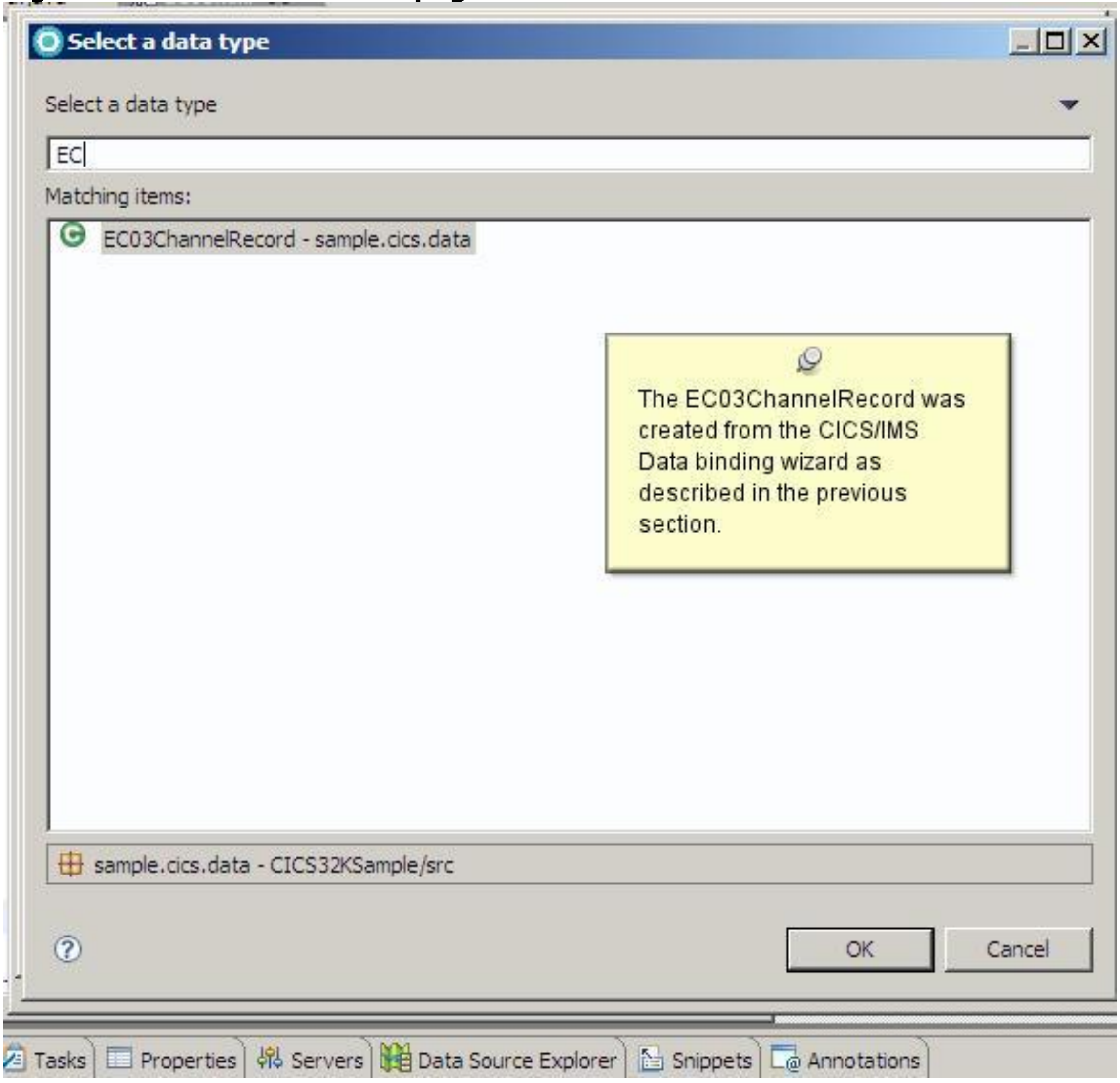
Method summary:

invoke () : void

Click the **Browse** button to select the data type to be used for the Input Data Type.

We will select **EC03ChannelRecord** which is created in the CICS/IMS Data Binding as shown in previous section.

Figure 15. J2C Java Method page

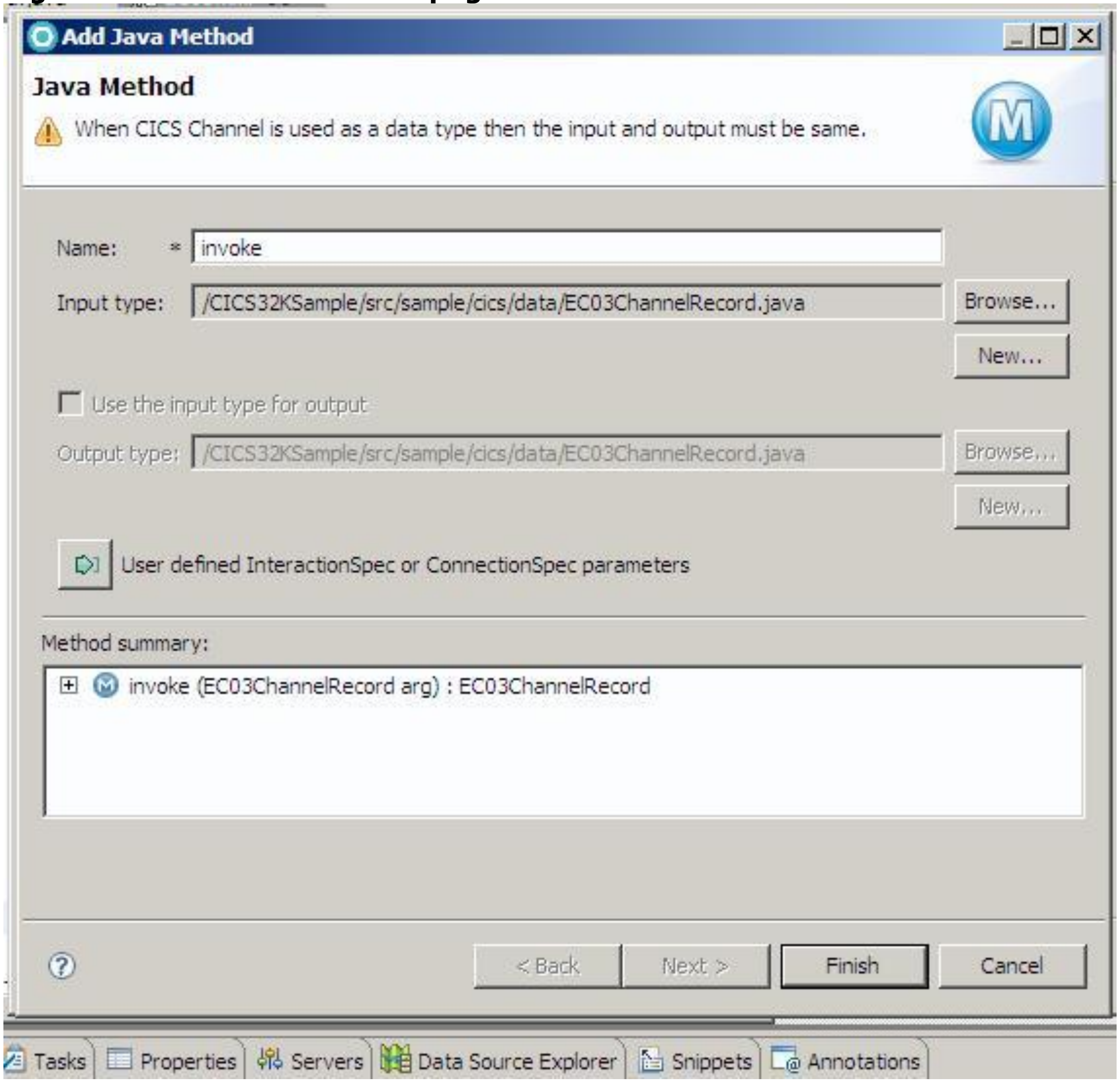


The EC03ChannelRecord was created from the CICS/IMS Data binding wizard as described in the previous section.

Note that when the CICS Channel type is used as Input Data type, the Output data type is required to be the same.

Click the **Finish** button to continue.

Figure 16. J2C Java Methods page



Java Method

When CICS Channel is used as a data type then the input and output must be same.

Name: * invoke

Input type: /CICS32KSample/src/sample/cics/data/EC03ChannelRecord.java

Use the input type for output

Output type: /CICS32KSample/src/sample/cics/data/EC03ChannelRecord.java

User defined InteractionSpec or ConnectionSpec parameters

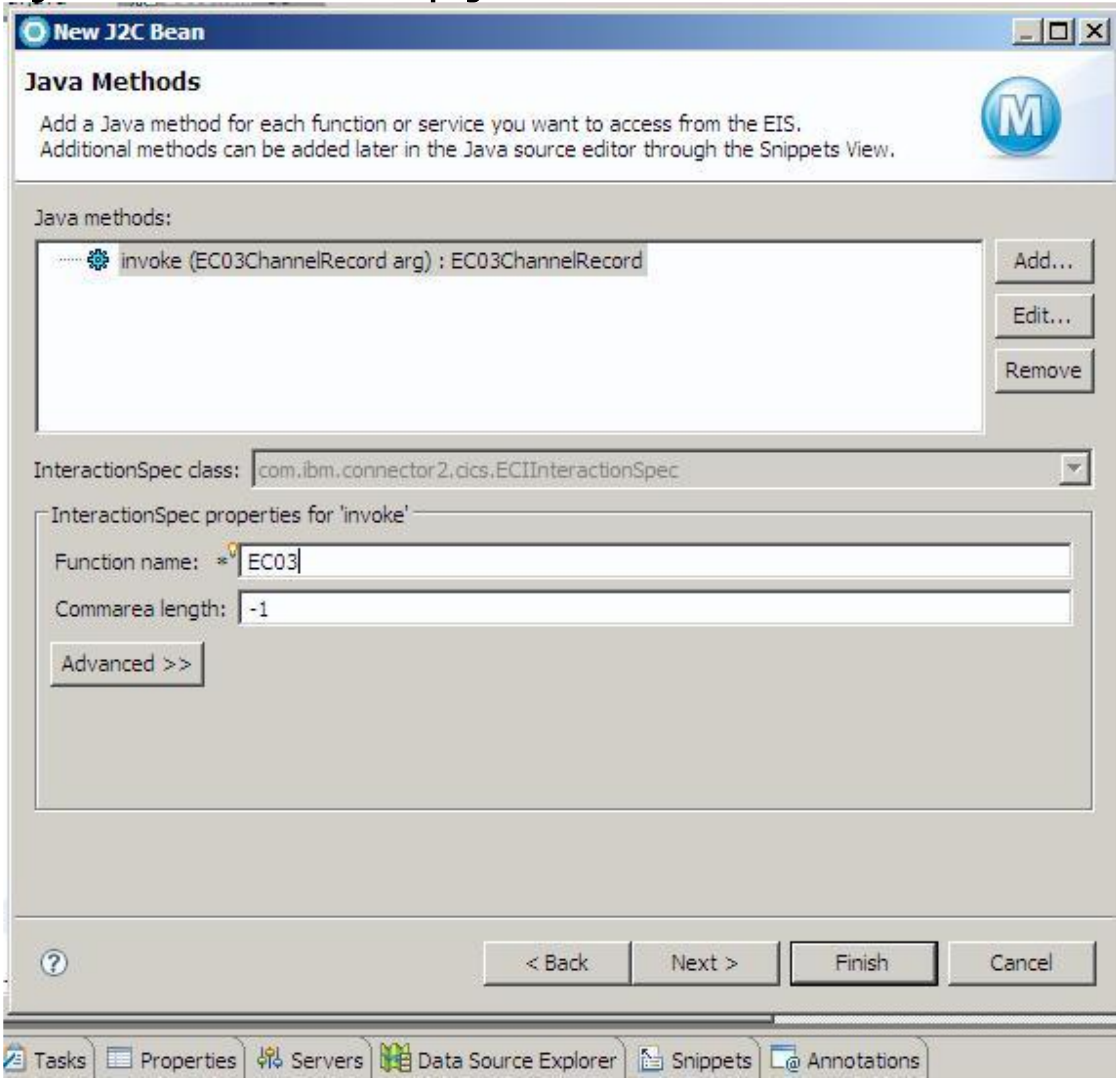
Method summary:

invoke (EC03ChannelRecord arg) : EC03ChannelRecord

In the InteractionSpec properties for **invoke** method, enter **EC03** as the FunctionName as shown in Figure 17. This functionName has to match your CICS Cobol server Program id. For the sample EC03.cpp, the Program id is **EC03**.

Click the **Finish** button to continue generating the J2C Java Bean interface and Implementation code.

Figure 17. J2C Java Methods page Function Name


New J2C Bean

Java Methods

Add a Java method for each function or service you want to access from the EIS.
Additional methods can be added later in the Java source editor through the Snippets View.

Java methods:

invoke (EC03ChannelRecord arg) : EC03ChannelRecord

InteractionSpec class: com.ibm.connector2.cics.ECIInteractionSpec

InteractionSpec properties for 'invoke'

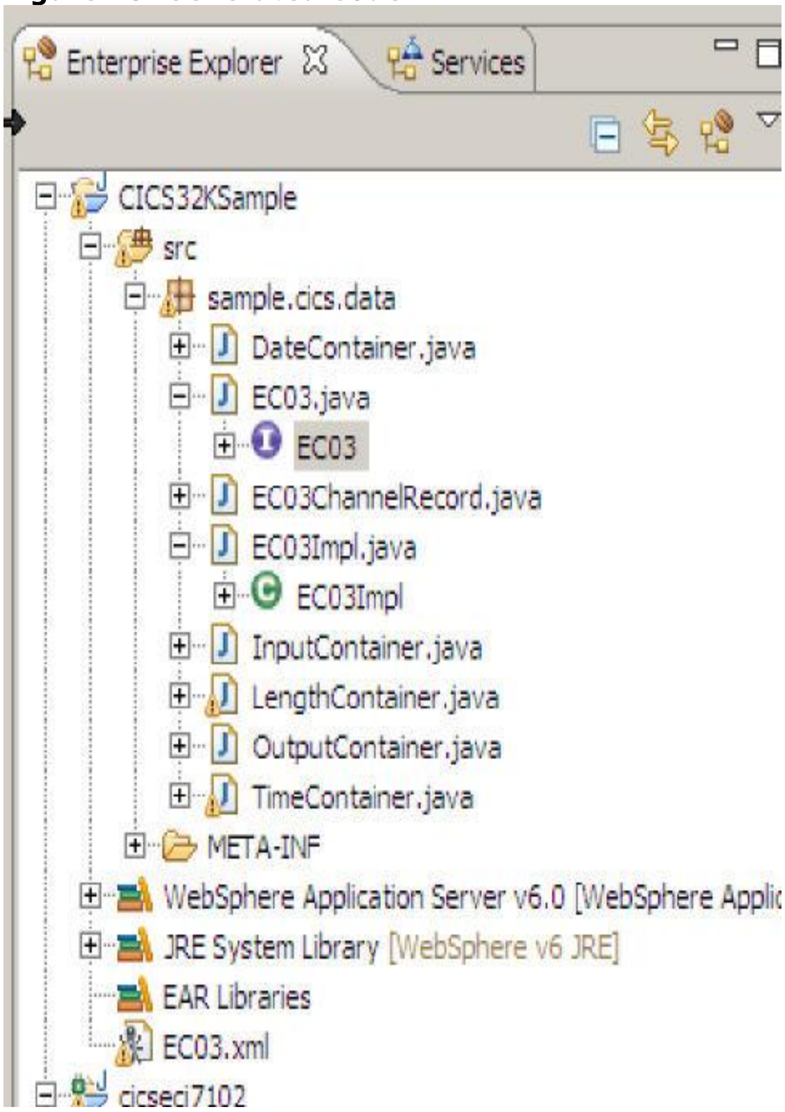
Function name: * EC03

Commarea length: -1

Generated Code

- There will be a J2C Java Bean Interface and Implementation class generated.
- The method in the class will have CICS Channel class **EC03ChannelRecord.java** as the input and output data type.

Figure 18. Generated Code



Doclets

J2C doclet tags enable you to specify particular characteristics of your J2C Java bean. The J2C Java bean consists of an interface and an implementation class (Impl.java). The implementation section of the Java bean contains J2C doclets that you can edit using the tags listed in the following pages.

You can modify the doclets property value e.g. the jndi-name or value of the functionName, the method body will be regenerated automatically. It is not recommended to add any user code to the method body; it will be overwritten once the regeneration of the method body occurs.

Listing 15. Doclets in the generated code

```
/**
 * @j2c.connectionFactory jndi-name="MyCICS32KJNDIName"
 * @j2c.connectionSpec
class="com.ibm.connector2.cics.ECIConnectionSpec"
 * @generated
 */
public class EC03Impl implements sample.cics.data.EC03 {

        private ConnectionSpec typeLevelConnectionSpec;
        private InteractionSpec invokedInteractionSpec;
        private InteractionSpec interactionSpec;
        private ConnectionSpec connectionSpec;
        private Connection connection;
        private ConnectionFactory connectionFactory;

/**
* @j2c.interactionSpec
class="com.ibm.connector2.cics.ECIInteractionSpec"
* @j2c.interactionSpec-property name="functionName" value="EC03"
* @generated
*/
public sample.cics.data.EC03ChannelRecord invoke(
        sample.cics.data.EC03ChannelRecord arg)
```

Note that in listing16, there are special tags to the Java source code for the CICS Container Link support. Some of them are:

- @type-descriptor.CICSChannel
- @type-descriptor.CICSContainer

Listing 16. Doclets in the generated Channel code

```
/**
 * @generated
 * Generated Class: EC03ChannelRecord
 * @type-descriptor.CICSChannel channel-name="EC03ChannelRecord"
 * @type-descriptor.CICSContainer class-
name="sample.cics.data.OutputContainer"
 *   container-name="OUTPUTDATA" container-type="CHAR"
 * @type-descriptor.CICSContainer class-
name="sample.cics.data.DateContainer"
 *   container-name="CURRENTDATE" container-type="CHAR"
 * @type-descriptor.CICSContainer class-
name="sample.cics.data.TimeContainer"
 *   container-name="CURRENTTIME" container-type="BIT"
 * @type-descriptor.CICSContainer class-
```

```
name="sample.cics.data.InputContainer"
 *   container-name="INPUTMESSAGE" container-type="CHAR"
 * @type-descriptor.CICSContainer class-
name="sample.cics.data.LengthContainer"
 *   container-name="INPUTDATALENGTH" container-type="BIT"
 */
```

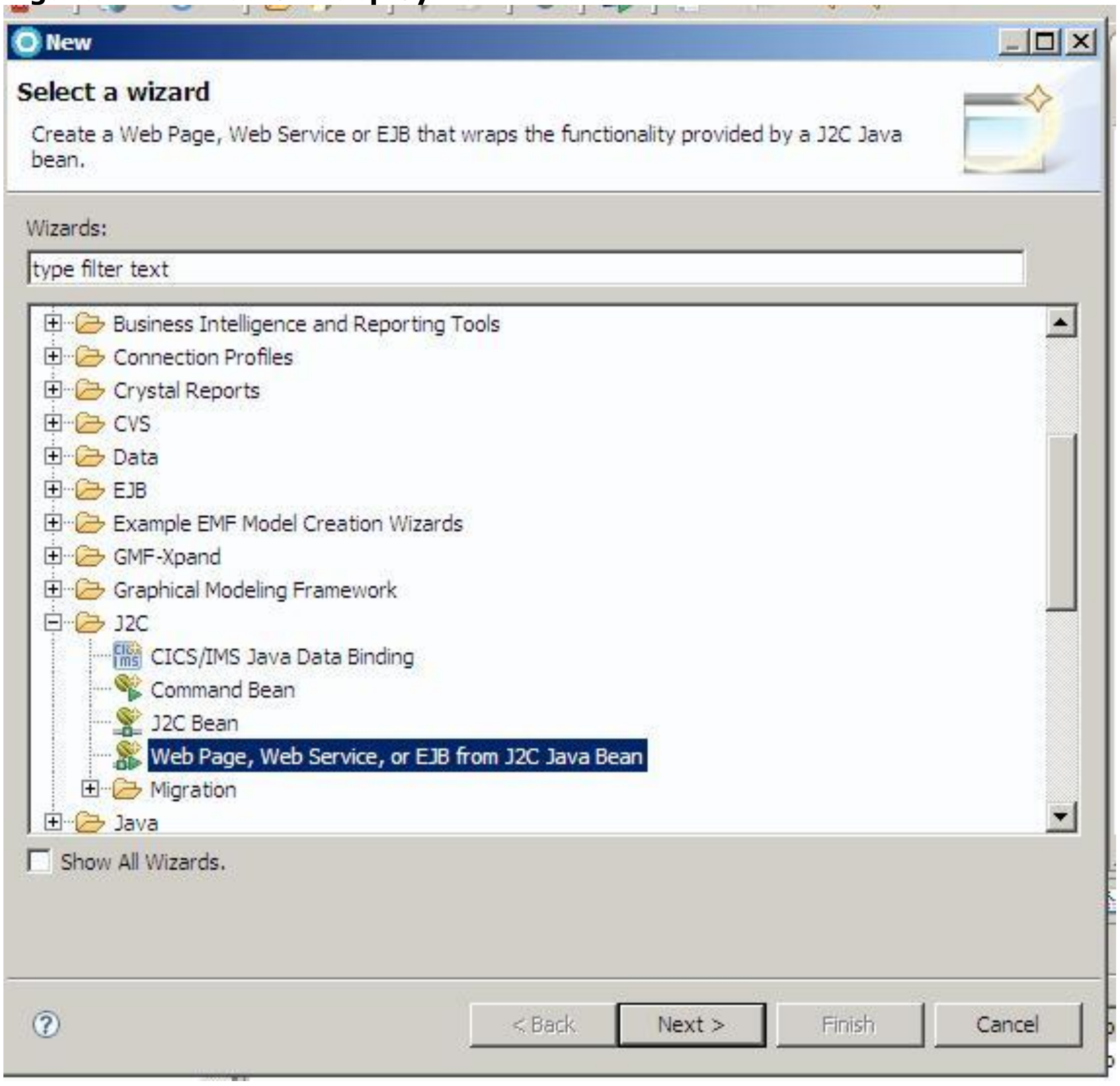
Deploying J2C Java Bean

Deploying J2C Java Bean using the standalone deployment wizard

After you have generated the J2C Java Bean Interface and Implementation, you can use the Web Page, Web Service, or EJB from J2C Java Bean to deploy as Simple JSP.

1. Select **J2C > Web Page, Web Service, or EJB from J2C Java Bean**.

Figure 19. Standalone deployment wizard



New

Select a wizard

Create a Web Page, Web Service or EJB that wraps the functionality provided by a J2C Java bean.

Wizards:

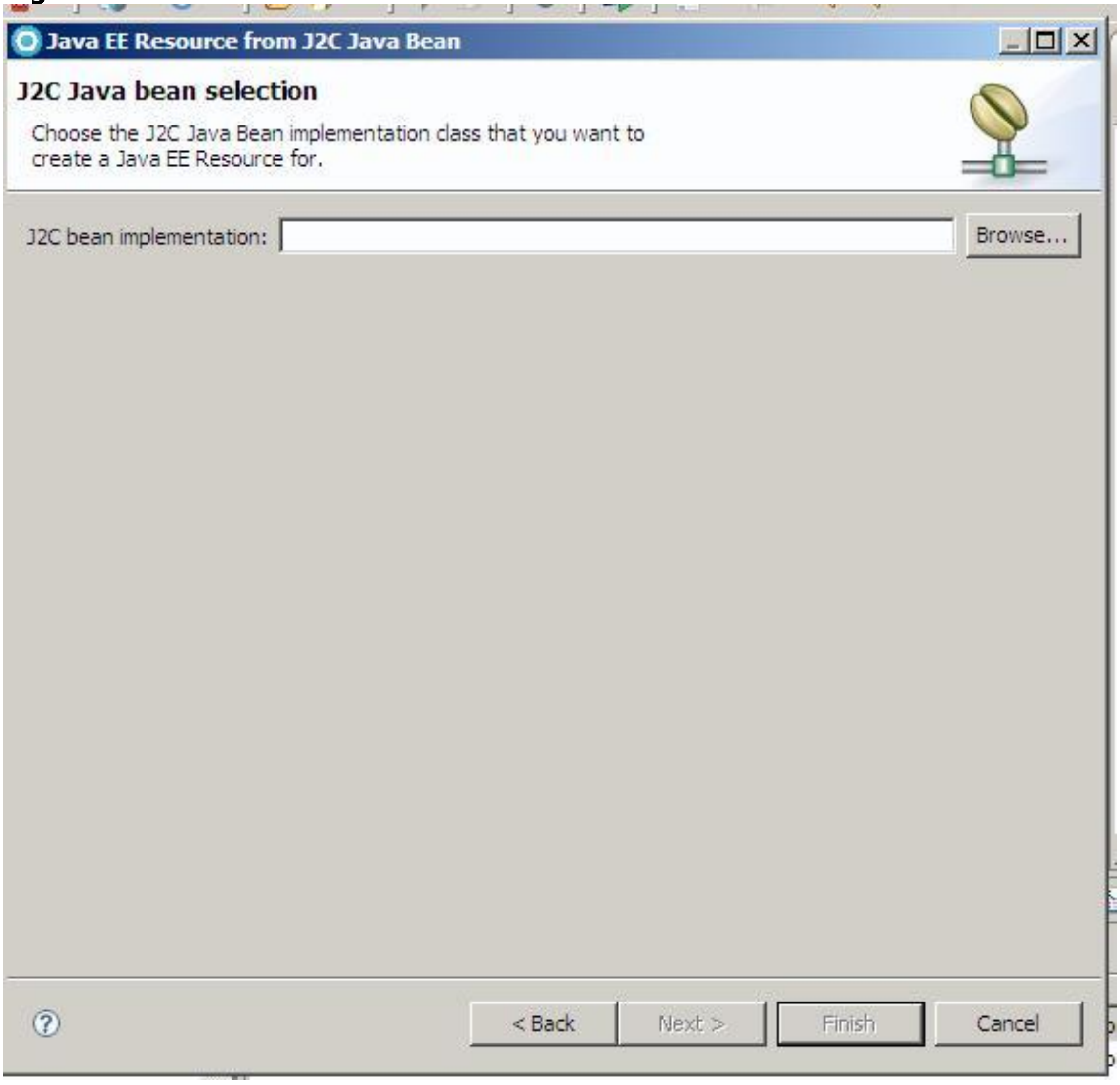
type filter text

- Business Intelligence and Reporting Tools
- Connection Profiles
- Crystal Reports
- CVS
- Data
- EJB
- Example EMF Model Creation Wizards
- GMF-Xpand
- Graphical Modeling Framework
- J2C
  - CICS/IMS Java Data Binding
  - Command Bean
  - J2C Bean
  - Web Page, Web Service, or EJB from J2C Java Bean
  - Migration
- Java

Show All Wizards.

2. Select the **Browse** button to locate the J2C Java Bean

Figure 20. Browse for J2C Java Bean



Java EE Resource from J2C Java Bean

J2C Java bean selection

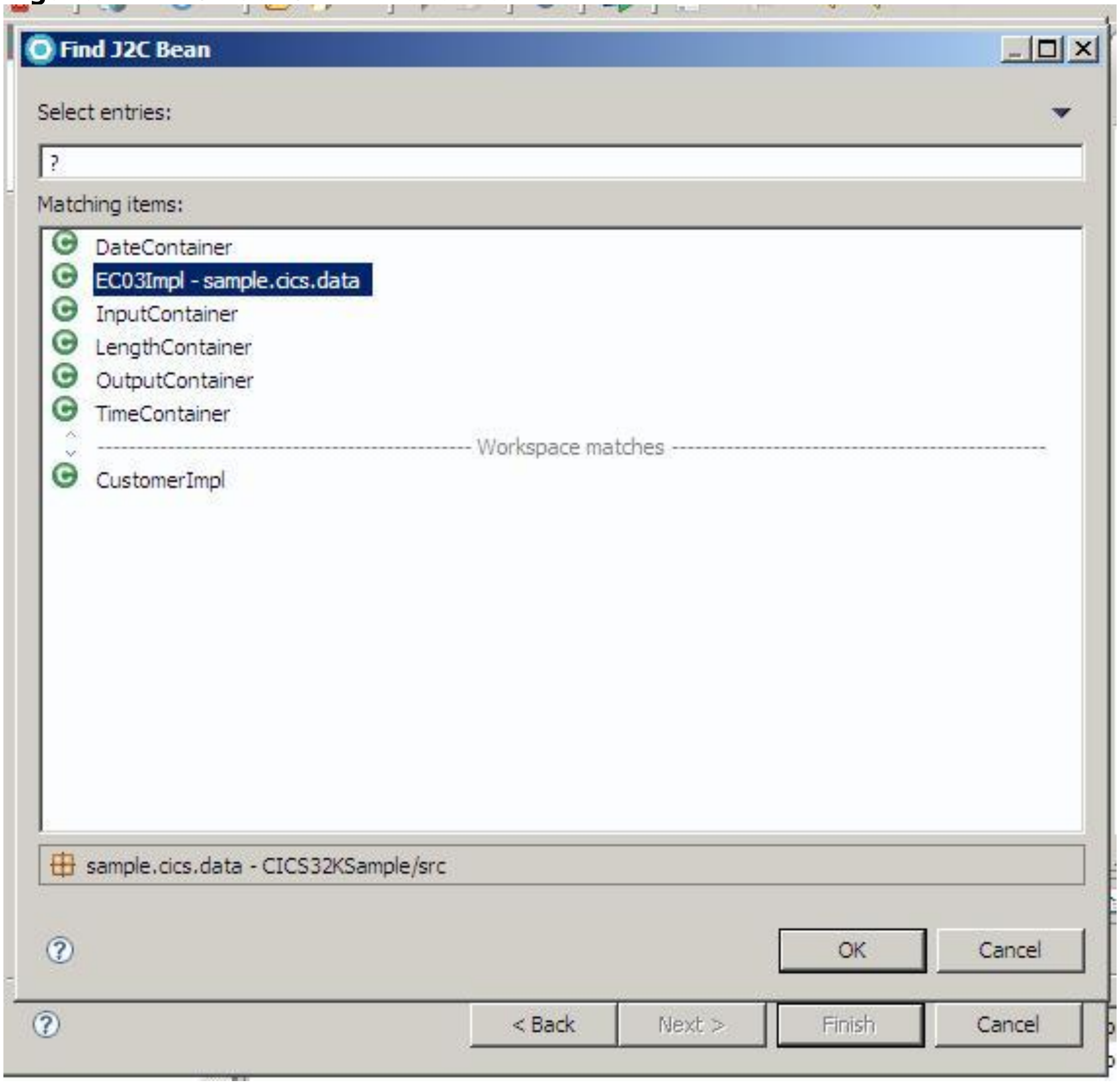
Choose the J2C Java Bean implementation class that you want to create a Java EE Resource for.

J2C bean implementation:

3. Enter **?** in the Find J2C Bean wizard.

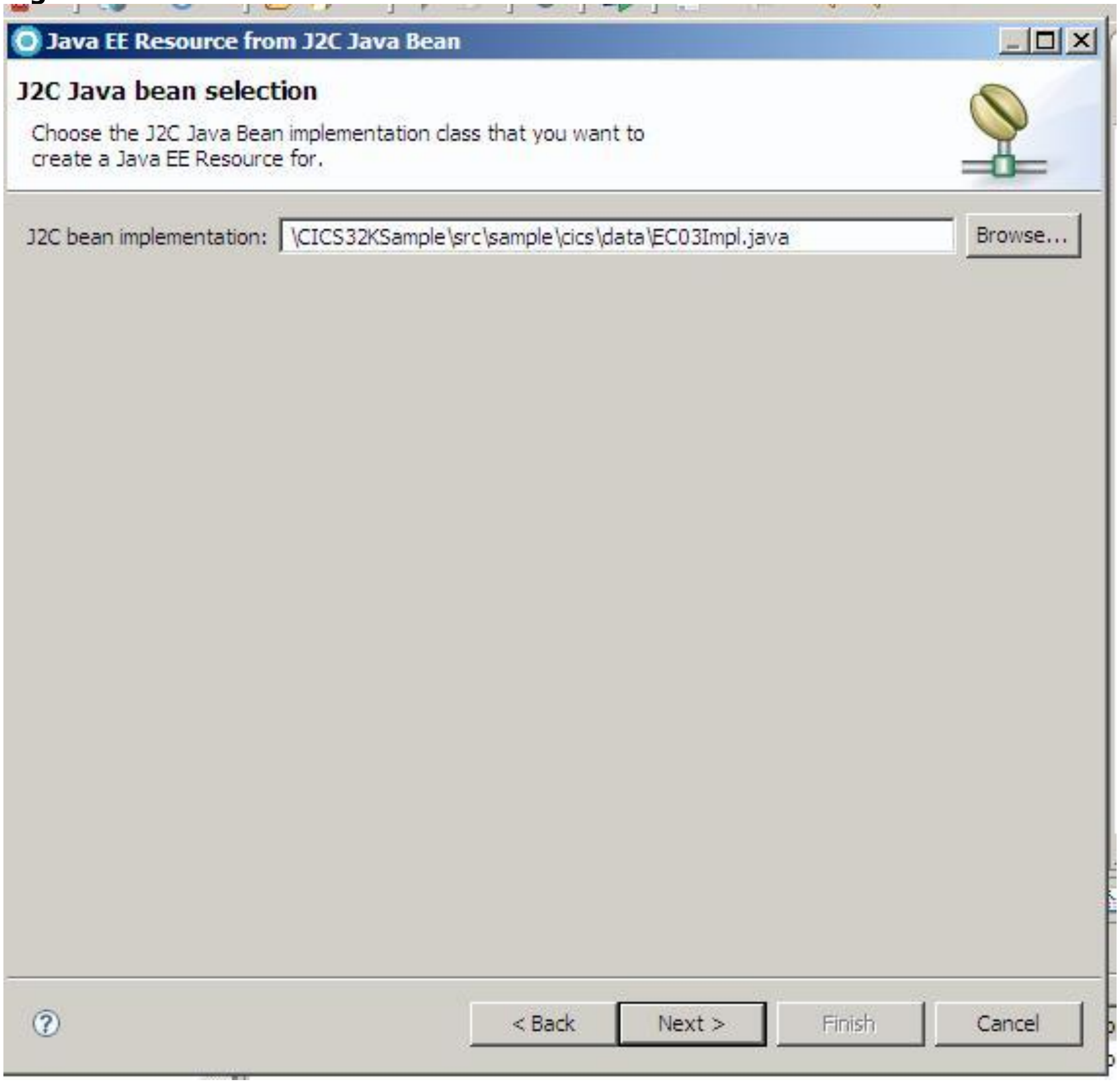
   A list of files will be displayed for your selection. Select EC03Impl from the list.

Figure 21. Browse for J2C Java Bean



4. Click **Next** in the J2C Java Bean Selection page.

Figure 22. J2C Java Bean selection

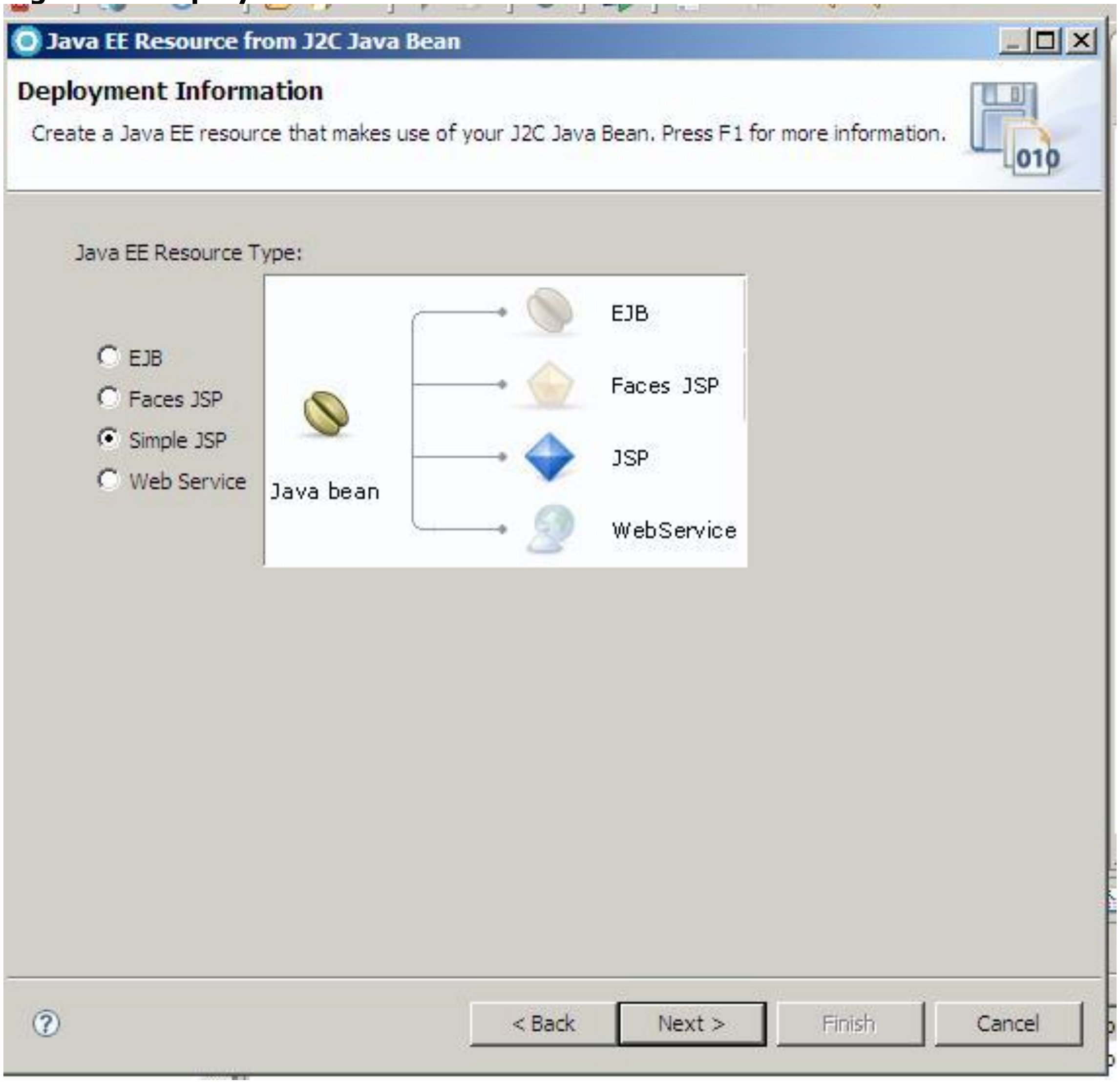


Choose the J2C Java Bean implementation class that you want to create a Java EE Resource for.

J2C bean implementation: \CICS32KSample\src\sample\cics\data\EC03Impl.java

5. In the Deployment Page, select **Simple JSP**. Press **Next**

Figure 23.Deployment wizard



Java EE Resource from J2C Java Bean

Deployment Information

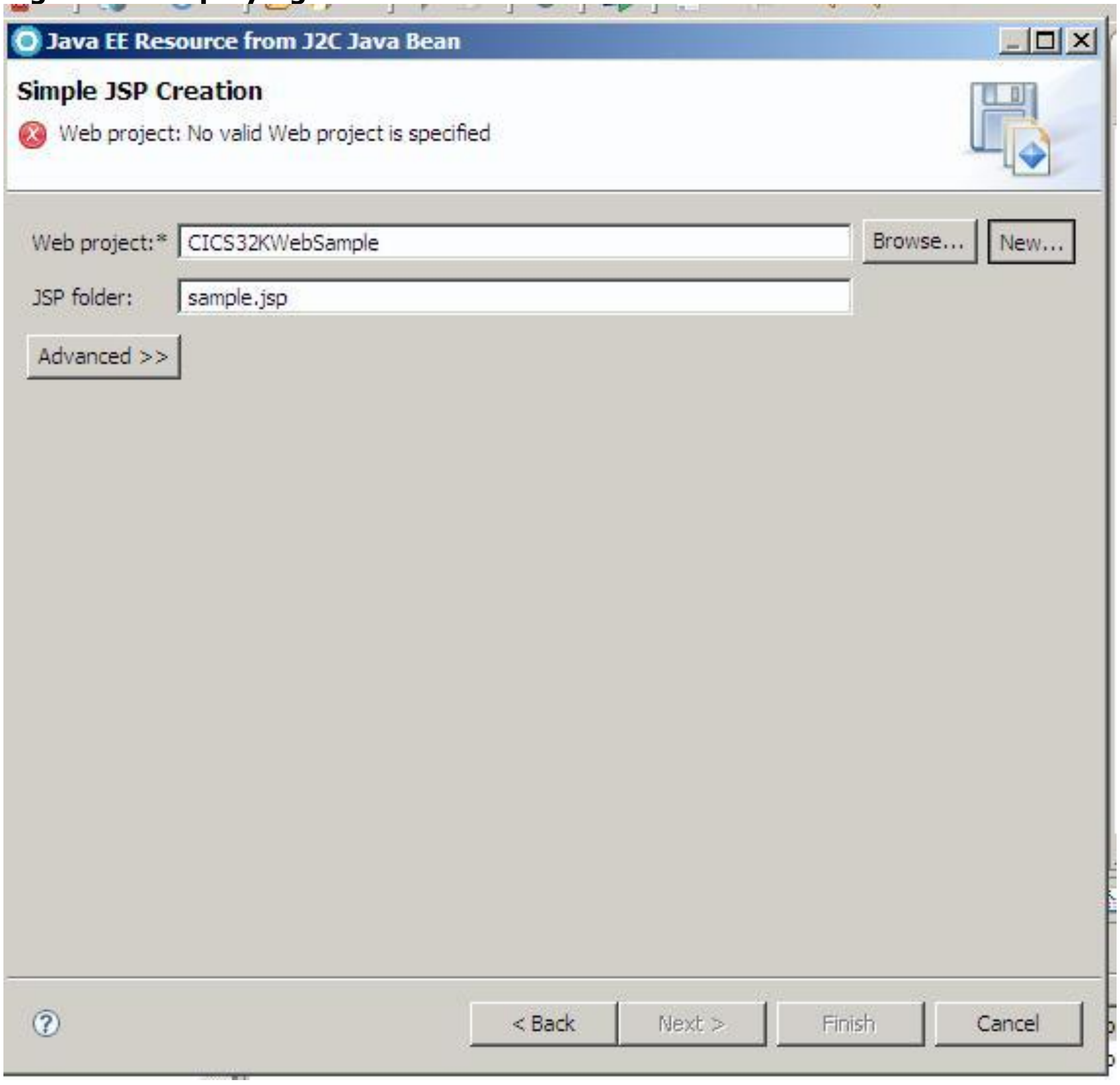
Create a Java EE resource that makes use of your J2C Java Bean. Press F1 for more information.

Java EE Resource Type:

( ) EJB
( ) Faces JSP
(●) Simple JSP
( ) Web Service

6. Enter Web Project Name and select **New** button.

Figure 24. Deploying



Java EE Resource from J2C Java Bean

Simple JSP Creation

Web project: No valid Web project is specified

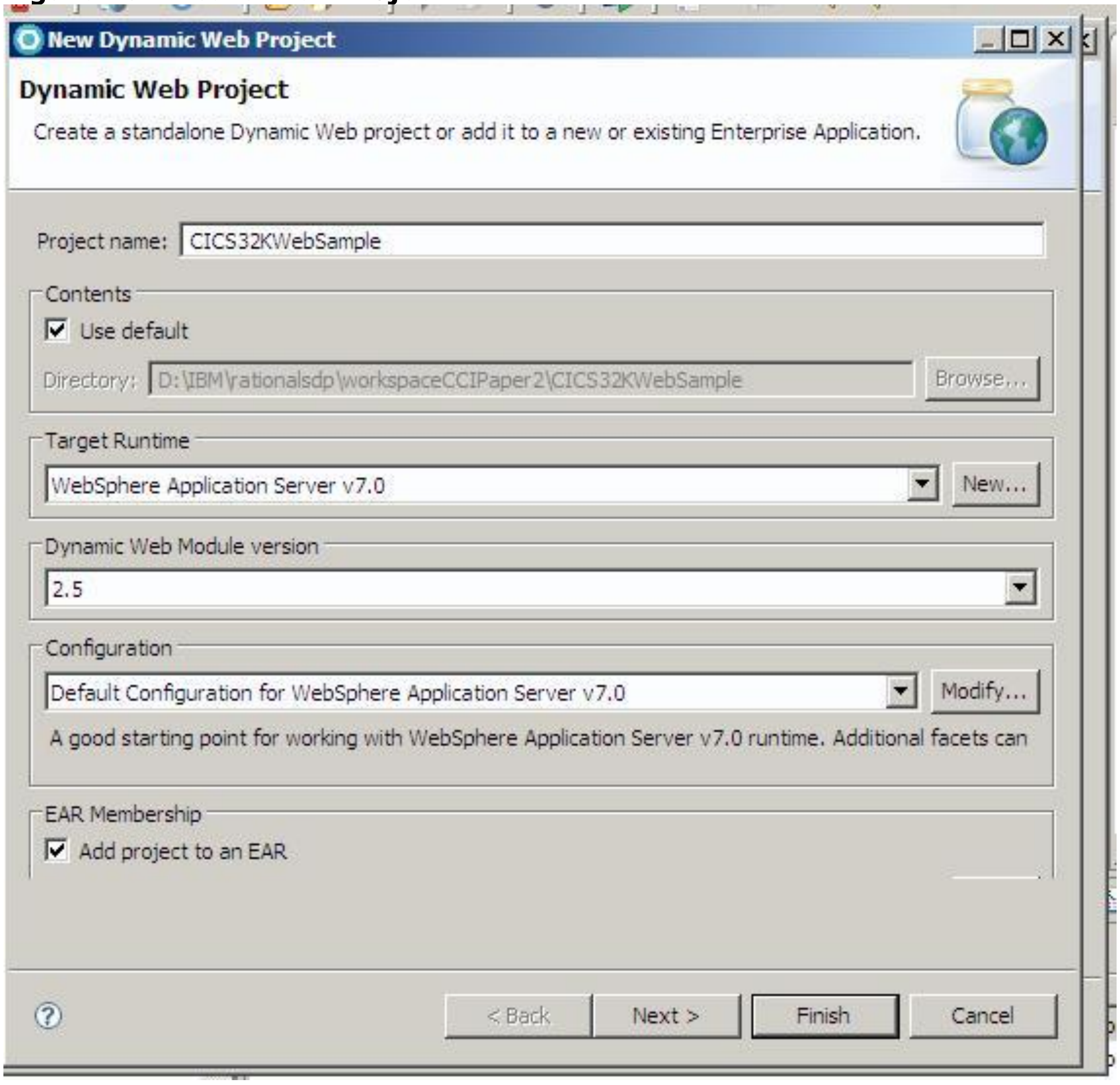
Web project:* CICS32KWebSample

JSP folder: sample.jsp

7. Press **Finish** button in the New Dynamic Web Project wizard to create the web project.

   Press **Finish** again in the Deployment page to generate the JSP.

Figure 25. Create Web Project



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: CICS32KWebSample

Contents

☑ Use default

Directory: D:\IBM\rationalsdp\workspaceCCIPaper2\CICS32KWebSample

Target Runtime: WebSphere Application Server v7.0

Dynamic Web Module version: 2.5

Configuration: Default Configuration for WebSphere Application Server v7.0

A good starting point for working with WebSphere Application Server v7.0 runtime. Additional facets can

EAR Membership

☑ Add project to an EAR

Deploying J2C Java Bean as part of the J2C Java Bean creation

You will be able to deploy as part of the J2C Java Bean creation. In figure 17 J2C Java Method Page Function Name, instead of press **Finish** to generate the J2C Java Bean, just select the **Next** button.

Then it will get you to the deployment wizard page in figure 22. Just follow the same steps (step 4 to step 7) in the above section to deploy as Simple JSP.

Congratulations, you have learned how to generate a CICS Container Link J2C application using the J2C Tools. If you want to learn about the J2C Ant script support, you can continue to the next section.

J2C Ant scripts support

Ant session recording for CICS Channel support

In section 1, we show you how to use the wizard to generated J2C Java Beans and also record a wizard session. The ant session recording enables you to go through the wizard once, choose the CICS Channel to Java Mapping and record the user options and data in an Ant script file. The two wizards where the Ant script generation is supported are:

- CICS IMS Data Binding wizard: The Ant script will generate one or more data binding files.
- J2C Java Bean wizard: The Ant script will generate J2C beans, import resource adapters as well as data binding files.

Layout of Data Binding Ant scripts

Layout of Data Binding Ant scripts. See Listing 2 below;

The generated data binding script is composed of targets, tasks, types and properties. To find out more about how Ant works, go to http://Ant.apache.org/manual/.

The script has one custom task, *discover*. The rest of the tags are custom types. Data discovery occurs in two distinct steps, which can be easily identified in the sample script with the *performImport* and *writeToWorkspace* complex elements (Ant custom types).

Both custom types contain the necessary data required for language (COBOL, PL1, or C) file import and (data binding bean) artifact generation.

The import and generation is performed internally, using a Discovery Agent and a Workspace Resource Writer respectively. Depending on the language file being imported and the generated artifact type, various implementations can be used. Each implementation is identified by its ID (javax.xml.namespace.QName). Language files such as COBOL, C, and PL/1 are some examples for which an agent implementation exists. In this example, the imported type is a COBOL file with CICS Channel support. Processing this type of COBOL file uses the same Discovery Agent as for the MPO case, hence the ID of *{com/ibm/adapter}MPOCobolDiscoveryAgent* is used. The same logic applies for writers as well. The writer ID for COBOL CICS Channel to Java is *"{com/ibm/adapter/cobol/cicschannel/writer}JAVA_WRITER"*

The file import (COBOL) data and the data necessary to write the artifact (Java) is stored using custom property types like *propertyElement* and *propertyGroup*. This is the format the agents and writer can handle.

Note that at the end of the script, we have done a refreshLocal and incrementalBuild. This is to allow the Annotation builder to regenerate the code in method body based on all the doclet Tags in the J2C bean and data binding file, and to synchronize the project.

Listing 2.Samples Data Binding Ant scripts with CICS Channel Support

```
<?xml version="1.0" encoding="UTF-8"? >
<project default="DataBinding1"
            name="/CICS32KSample/EC03ChannelRecord.xml"
            xmlns:adapter="http://com.ibm.adapter" >
  <property name="debug" value="true"/ >
  <property name="project1" value="CICS32KSample"/ >
  <target name="DataBinding1" >
    <adapter:createProject projectName="${project1}"
                           projectType="Java"
                           sourceFolder="src"/ >
    <adapter:discover >
      <adapter:performImport
agent="{com/ibm/adapter}MPOCobolDiscoveryAgent" >
        <adapter:importResource >
          <adapter:propertyGroup name="CobolFileGroup" >
            <adapter:propertyElement name="CobolFile"

value="D:\IBM\SDP70Shared\plugins\

com.ibm.j2c.cheatsheet.content_7.0.1.
                                     v20080710-
1450\Samples\CICS32K\ec03.ccp"/ >
          </adapter:propertyGroup >
        </adapter:importResource >
        <adapter:queryProperties >
          <adapter:propertyGroup name="ImportProperties" >
            <adapter:propertyElement name="Platform" value="Win32"/ >
            <adapter:propertyElement name="Codepage" value="ISO-8859-
1"/ >
            <adapter:propertyElement name="Numproc" value="PFD"/ >
            <adapter:propertyElement name="FloatingPointFormat"
value="IEEE 754"/ >
              <adapter:propertyGroup name="ExternalDecimalSignGroup"
>
                <adapter:propertyElement name="ExternalDecimalSign"
value="ASCII"/ >
              </adapter:propertyGroup >
            <adapter:propertyGroup name="EndianGroup" >
              <adapter:propertyElement name="Endian" value="Little"/
>
              <adapter:propertyElement name="RemoteEndian"
value="Little"/ >
            </adapter:propertyGroup >
            <adapter:propertyGroup name="CompileOptions" >
              <adapter:propertyElement name="Quote" value="DOUBLE"/ >
              <adapter:propertyElement name="Trunc" value="STD"/ >
              <adapter:propertyElement name="Nsymbol" value="DBCS"/ >
            </adapter:propertyGroup >
          </adapter:propertyGroup >
        </adapter:queryProperties >
        <adapter:queryResult >
          <adapter:selectElement name="DATECONTAINER"/ >
          <adapter:selectElement name="TIMECONTAINER"/ >
          <adapter:selectElement name="INPUTCONTAINER"/ >
          <adapter:selectElement name="OUTPUTCONTAINER"/ >
          <adapter:selectElement name="LENGTHCONTAINER"/ >
        </adapter:queryResult >
      </adapter:performImport >
      <adapter:writeToWorkspace

writer="{com/ibm/adapter/cobol/cicschannel/writer}JAVA_WRITER" >
        <adapter:propertyGroup name="MPO_PG">
          <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
          <adapter:propertyTree name="MPO_TP" >
            <adapter:propertyNode name="COBOLMPOToJavaResourceWriter"
>
              <adapter:propertyGroup name="Java Type Name" >
                <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                <adapter:propertyElement name="Class Name"
value="EC03ChannelRecord"/ >
                <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                <adapter:propertyElement name="CHANNEL_NAME"
value="InputRecord"/ >
              </adapter:propertyGroup >
            </adapter:propertyNode >
            <adapter:propertyNode name="COBOLImportResult0" >
              <adapter:propertyGroup name="COBOLToJavaResourceWriter"
>
                <adapter:propertyGroup name="Java Type Name" >
                  <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                  <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                  <adapter:propertyElement name="Class Name"
value="DateContainer"/ >
                  <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                  <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
                  <adapter:propertyElement name="CONTAINER_NAME"
value="CURRENTDATE"/ >
                  <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
                </adapter:propertyGroup >
              </adapter:propertyGroup >
            </adapter:propertyNode >
            <adapter:propertyNode name="COBOLImportResult1" >
              <adapter:propertyGroup name="COBOLToJavaResourceWriter"
>
                <adapter:propertyGroup name="Java Type Name" >
                  <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                  <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                  <adapter:propertyElement name="Class Name"
```

```
value="TimeContainer"/ >
                  <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                  <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
                  <adapter:propertyElement name="CONTAINER_NAME"
value="CURRENTTIME"/ >
                  <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
                </adapter:propertyGroup >
              </adapter:propertyGroup >
            </adapter:propertyNode >
            <adapter:propertyNode name="COBOLImportResult2" >
              <adapter:propertyGroup name="COBOLToJavaResourceWriter"
>
                <adapter:propertyGroup name="Java Type Name" >
                  <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                  <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                  <adapter:propertyElement name="Class Name"
value="InputContainer"/ >
                  <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                  <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
                  <adapter:propertyElement name="CONTAINER_NAME"
value="INPUTDATA"/ >
                  <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
                </adapter:propertyGroup >
              </adapter:propertyGroup >
            </adapter:propertyNode >
            <adapter:propertyNode name="COBOLImportResult3" >
              <adapter:propertyGroup name="COBOLToJavaResourceWriter"
>
                <adapter:propertyGroup name="Java Type Name" >
                  <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                  <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                  <adapter:propertyElement name="Class Name"
value="OutputCiontainer"/ >
                  <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                  <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
                  <adapter:propertyElement name="CONTAINER_NAME"
value="OUTPUTMESSAGE"/ >
                  <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
                </adapter:propertyGroup >
              </adapter:propertyGroup >
            </adapter:propertyNode >
            <adapter:propertyNode name="COBOLImportResult4" >
              <adapter:propertyGroup name="COBOLToJavaResourceWriter"
>
```

```
                <adapter:propertyGroup name="Java Type Name" >
                  <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
                  <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
                  <adapter:propertyElement name="Class Name"
value="LengthContainer"/ >
                  <adapter:propertyElement name="Overwrite existing
class" value="true"/ >
                  <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
                  <adapter:propertyElement name="CONTAINER_NAME"
value="INPUTDATALENGTH"/>
                  <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
                </adapter:propertyGroup >
              </adapter:propertyGroup >
            </adapter:propertyNode >
          </adapter:propertyTree >
        </adapter:propertyGroup >
      </adapter:writeToWorkspace >
    </adapter:discover >
    <eclipse.refreshLocal depth="infinite" resource="${project1}"/ >
    <eclipse.incrementalBuild project="${project1}"/ >
  </target >
</project >
```

Layout of J2C Java Bean Ant scripts

The following sample (shown in Listing 3) describes the J2C bean generation portion of the Ant file generated as a result of J2C Java Bean session recording.

The script is organized into different targets that depend on each other. The different targets will be executed sequentially according to the dependency.

The script has one task, generateService. The two distinct sections are buildService and writeToWorkspace. The first section is different from the previous example, since the service is being built, not discovered. It assumes that the *EC03ChannelRecord.java* file and all container Classes exist under the *CICS32KSample* project and the *sample.cics.data* package.

The second section is identical to the writeToWorkspace from CICS/IMS Data Binding, with the difference in the values being used.

Among the described tasks, the Ant file contains two additional helper tasks. The createProject task, as its name implies, it is used for project creation. The importResourceAdapter task is used for importing resource adapter and creating a connector project cicseci7102

Layout of J2C Java Bean Ant scripts. See Listing 3 below;

Listing 3. Sample Code for J2C Java Bean Ant script

```
<?xml version="1.0" encoding="UTF-8"? >
<project default="J2CBeanGeneration1"
            name="/CICS32KSample/EC03.xml"
            xmlns:adapter="http://com.ibm.adapter"
            xmlns:j2c="http://com.ibm.adapter.j2c" >
  <target name="Init1" >
    <property name="debug" value="true"/ >
    <property name="project1" value="CICS32KSample"/ >
    <property name="ra.project" value="cicseci7102"/ >
    <property name="ra.runtime" value="WebSphere Application Server
v7.0 stub"/ >
    <property name="ra.file"

value="D:\IBM\SDP751\ResourceAdapters\cics15\cicseci7102.rar"/ >
  </target >
  <target depends="Init1" name="DataBinding1" >
    <adapter:createProject projectName="${project1}"
                           projectType="Java"
                           sourceFolder="src"/ >
    <eclipse.refreshLocal depth="infinite" resource="${project1}"/ >
    <eclipse.incrementalBuild project="${project1}"/ >
  </target >
  <target depends="DataBinding1" name="importResourceAdapter1" >
    <j2c:importResourceAdapter addToEAR="no"
                               connectorFile="${ra.file}"
                               connectorModule="${ra.project}"
```

```
                                targetRuntime="${ra.runtime}"/ >
  </target >
  <target depends="importResourceAdapter1" name="J2CBeanGeneration1"
>
    <adapter:createProject projectName="${project1}"
                           projectType="Java"
                           sourceFolder="src"/ >
    <j2c:generateService >
      <j2c:buildService class="EC03" package="sample.cics" >
        <j2c:method >
          <j2c:methodName value="invoke"/ >
          <j2c:methodInput

value="/CICS32KSample/src/sample/cics/data/EC03ChannelRecord.java"/ >
          <j2c:methodOutput

value="/CICS32KSample/src/sample/cics/data/EC03ChannelRecord.java"/ >
          <j2c:interactionSpec
class="com.ibm.connector2.cics.ECIInteractionSpec" >
            <adapter:propertyGroup
name="INTERACTION_SPEC_PROPERTY_PG" >
              <adapter:propertyElement name="functionName"
value="EC03"/ >
              <adapter:propertyElement name="commareaLength" value="-
1"/ >
              <adapter:propertyElement name="replyLength" value="-1"/
>
              <adapter:propertyElement name="executeTimeout"
value="0"/ >
              <adapter:propertyElement name="interactionVerb"
value="1"/ >
            </adapter:propertyGroup >
          </j2c:interactionSpec >
        </j2c:method >
        <j2c:managedConnectionFactory

class="com.ibm.connector2.cics.ECIManagedConnectionFactory"
                  target="MyDefaultJNDIName" >
          <adapter:propertyGroup
name="MANAGED_CONNECTION_FACTORY_CLASS_PROPERTIES" >
            <adapter:propertyGroup name="Server" >
              <adapter:propertyElement name="ConnectionURL"
value="myURL"/ >
              <adapter:propertyElement name="ServerName"
value="myServer"/ >
            </adapter:propertyGroup >
            <adapter:propertyGroup name="UserVerification" >
              <adapter:propertyElement name="UserName"
value="myName"/ >
              <adapter:propertyElement name="Password"
value="myPassword"/ >
            </adapter:propertyGroup >
            <adapter:propertyElement name="TraceLevel" value="2"/ >
            <adapter:propertyGroup name="Security"/ >
          </adapter:propertyGroup >
        </j2c:managedConnectionFactory >
        <j2c:connectionSpec
```

```
class="com.ibm.connector2.cics.ECIConnectionSpec"/ >
        <j2c:resourceAdapter project="${ra.project}"/ >
      </j2c:buildService >
      <adapter:writeToWorkspace

writer="{com/ibm/adapter/j2c/codegen/writer}J2CAnnotationWorkspaceRes
ourceWriter">
        <adapter:propertyGroup name="J2C Java Bean Writer Properties"
>
          <adapter:propertyElement name="Project"
value="CICS32KSample"/ >
          <adapter:propertyElement name="PackageName"
value="sample.cics"/ >
          <adapter:propertyElement name="InterfaceName" value="EC03"/
>
        </adapter:propertyGroup >
      </adapter:writeToWorkspace >
    </j2c:generateService >
    <eclipse.refreshLocal depth="infinite" resource="${project1}"/ >
    <eclipse.incrementalBuild project="${project1}"/ >
  </target >
</project >
```

Customization

In order to customize the J2C Ant scripts, you need to know the following:

- What property to change
- What property value to set
- Where can you find the lists of valid property values

Some of the properties can be easily identified from the J2C UI, and you can view the list of possible values in a drop-down list.

Create Project Type

- **Basic Information**

  There are three project types that you can set:

  - Java
  - Web
  - EJB

  The *sourceFolder* represents the source folder in the project. The Java project creation wizard allows users to change the source folder default name or create multiple source folders. The Web and EJB projects require the *runtimeName* to be specified. The *runtimeName* is a string that can be determined from the preference page in the Runtime Environments under the Server option.
  Note: The ${project1} value for the *projectName* is an Ant variable defined through <property name="project1" value="CICS32KSample"/>.

- **Information in Ant Script**

Listing 4. Create Java project in Ant script

```
<adapter:createProject projectName="${project1}"
                       projectType="Java"
                       sourceFolder="src"/>
```

The code snippet in Listing 5 shows how to create a Web project.

1. In addition to the projectType, you need to specify the runtimeName.

Listing 5. Sample code for create Web Project in Ant script

```
<adapter:createProject projectName="${project1}"
                       projectType="Web"
                       runtimeName="WebSphere Application Server
v7.0">
```

If **add To EAR** is specified when you create the Web project, you need to specify the **EARProjectName** and **addToEAR** option besides the runtimeName.

Listing 6. Sample code for create Web Project with EAR in Ant script

```
 <adapter:createProject EARProjectName="MyWebEAR"
                        addToEAR="yes"
                        projectName="${project1}"
                        projectType="Web"
                        runtimeName="WebSphere Application Server
v7.0"/>
```

The code snippet in Listing 7 shows how to create an EJB project.

2. In addition to the projectType, you need to specify the runtimeName.

Listing 7. Sample code for create EJB Project in Ant script

```
<adapter:createProject projectName="${project1}"
                       projectType="EJB"
                       runtimeName="WebSphere Application Server
v7.0"/>
```

If add To EAR is specified when you create the EJB Project, you need to specify the **EARProjectName** and **addToEAR** option besides the runtimeName.

Listing 8. Sample code for create EJB Project with addToEAR enabled in Ant script

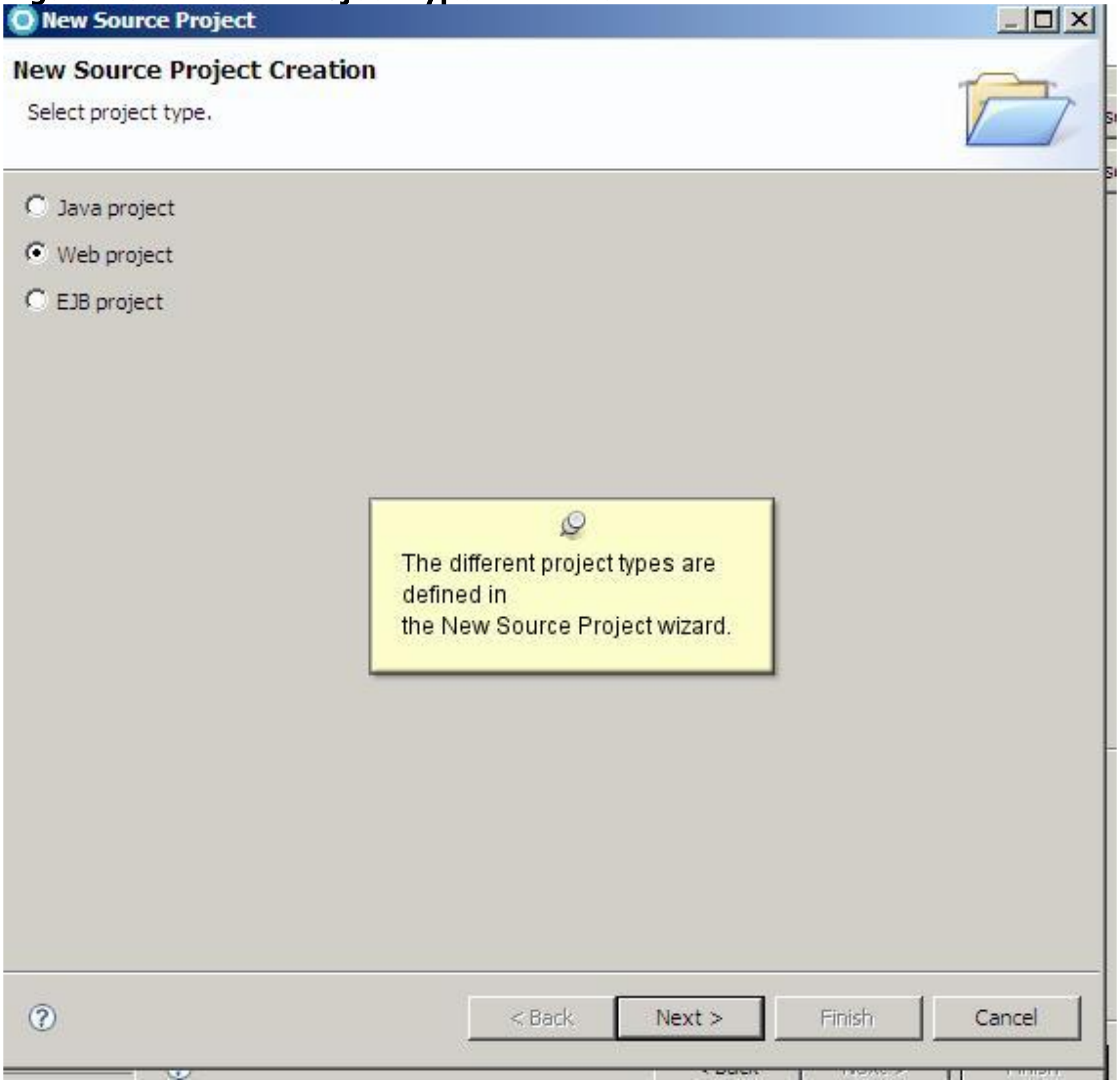
```
<adapter:createProject EARProjectName="MyEJBEAR"
                       addToEAR="yes"
                       projectName="${project1}"
                       projectType="EJB"
```

```
                        runtimeName="WebSphere Application Server
v7.0"/>
```

Information in wizard

The Project Type is located in the **New Source Project Creation** wizard as shown in figure 26.

Figure 26.Different Project Type in wizard


New Source Project

New Source Project Creation

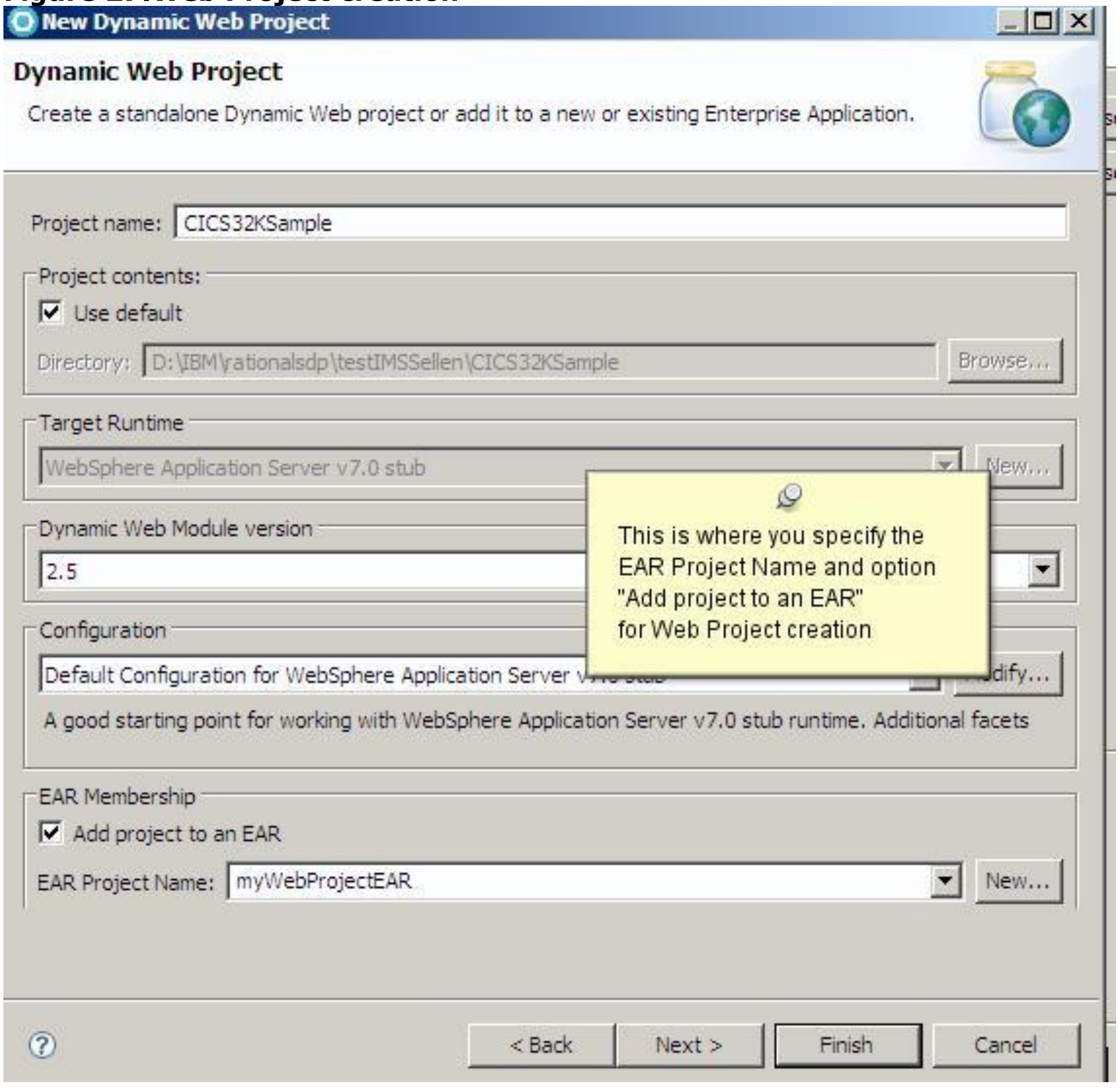
Select project type.

- ( ) Java project
- (●) Web project
- ( ) EJB project

The different project types are defined in the New Source Project wizard.

The Web Project Type EAR Project Name and Option "Add EAR to Project" is located in **New Dynamic Web Project** wizard as shown in figure 27.

Figure 27.Web Project creation



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: CICS32KSample

Project contents:

[✓] Use default

Directory: D:\IBM\rationalsdp\testIMSSellen\CICS32KSample

Target Runtime: WebSphere Application Server v7.0 stub

Dynamic Web Module version: 2.5

Configuration: Default Configuration for WebSphere Application Server v7.0 stub

A good starting point for working with WebSphere Application Server v7.0 stub runtime. Additional facets

EAR Membership

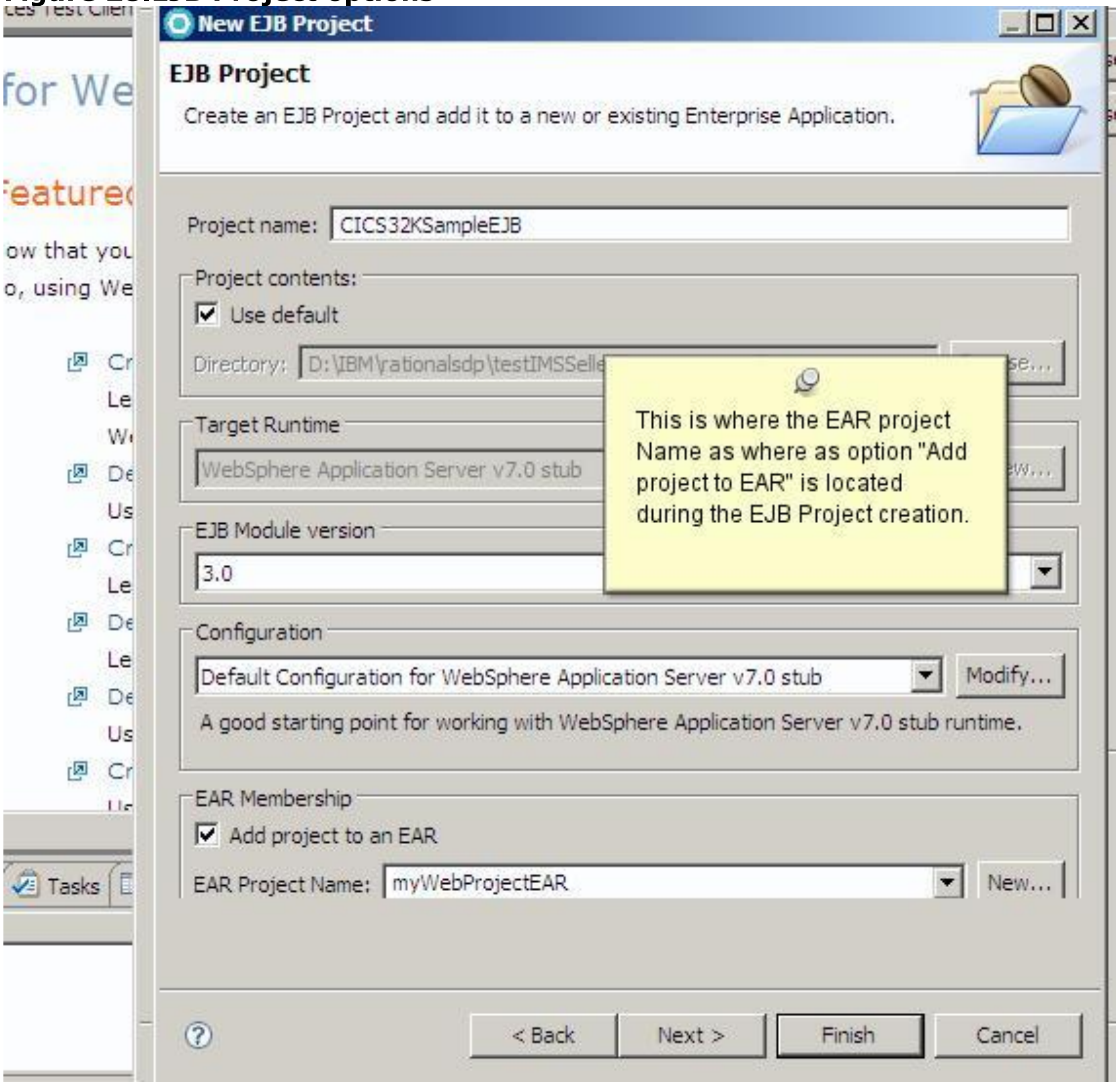
[✓] Add project to an EAR

EAR Project Name: myWebProjectEAR

This is where you specify the EAR Project Name and option "Add project to an EAR" for Web Project creation

The EJB Project Type EAR Project Name and Option "Add EAR to Project" is located in **New EJB Project** wizard as shown in figure 28.

Figure 28.EJB Project options


New EJB Project

EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: CICS32KSampleEJB

Project contents:

☑ Use default

Directory: D:\IBM\rationalsdp\testIMSSelle

Target Runtime

WebSphere Application Server v7.0 stub

EJB Module version

3.0

Configuration

Default Configuration for WebSphere Application Server v7.0 stub

A good starting point for working with WebSphere Application Server v7.0 stub runtime.

EAR Membership

☑ Add project to an EAR

EAR Project Name: myWebProjectEAR

This is where the EAR project Name as where as option "Add project to EAR" is located during the EJB Project creation.

queryResult

- **Basic Information**

  The *queryResult* represents the COBOL structures to be selected in the resulted query. When the COBOL file is analyzed a query is executed and the returned result can be a flat or nested structure of COBOL elements.

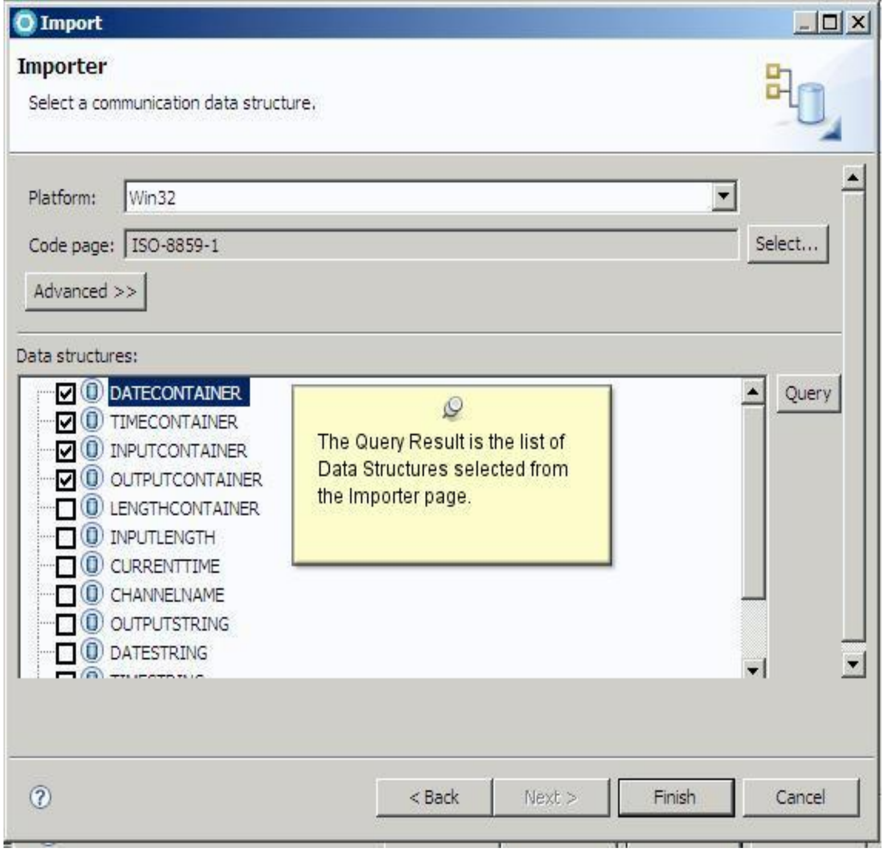
- **Information in Ant Script**

Listing 9. Sample queryResult section in J2C Java Bean Ant script

```
<adapter:queryResult>
    <adapter:selectElement name="DATECONTAINER"/>
    <adapter:selectElement name="TIMECONTAINER"/>
    <adapter:selectElement name="INPUTCONTAINER"/>
    <adapter:selectElement name="OUTPUTCONTAINER"/>
    <adapter:selectElement name="LENGTHCONTAINER"/>
 </adapter:queryResult>
```

- **Information in wizard**

Figure 29.queryResult and the corresponding page in wizard



Import

Importer

Select a communication data structure.

Platform: Win32

Code page: ISO-8859-1

Data structures:

- [✓] DATECONTAINER
- [✓] TIMECONTAINER
- [✓] INPUTCONTAINER
- [✓] OUTPUTCONTAINER
- [ ] LENGTHCONTAINER
- [ ] INPUTLENGTH
- [ ] CURRENTTIME
- [ ] CHANNELNAME
- [ ] OUTPUTSTRING
- [ ] DATESTRING

The Query Result is the list of Data Structures selected from the Importer page.

Discovery Agent

- **Basic Information**

  Some of the properties are implicit and do not show up in the J2C UI. One example is the Discovery Agent ID.

  You will not see any of the agent name "{com/ibm/adapter}MPOCobolDiscoveryAgent" in the UI. However, if you select the language mapping to be **COBOL CICS Channel to JAVA** or **COBOL MPO to Java**, then you will see the agent name generated in the Ant script.

  COBOL CICS Channel to Java is using the same pattern as MPO COBOL to Java, so they are using the same discovery agent.

- **Information in Ant Script**

  Listing 4 shows the agent name when you select **COBOL_CICS_CHANNEL_TO_JAVA** or **COBOL_MPO_TO_JAVA** mapping.

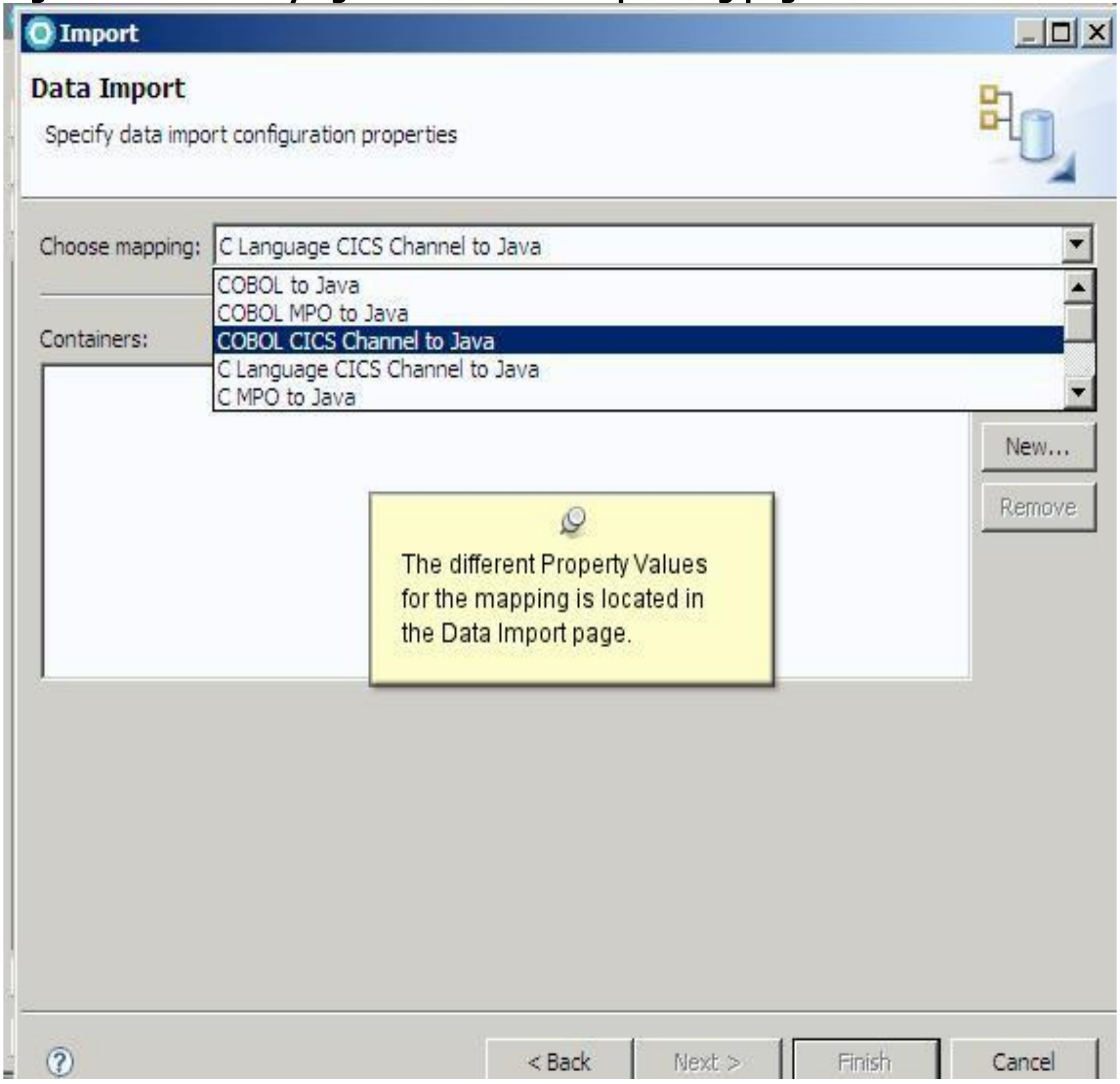
Listing 10. Sample Discovery Agent section in J2C Java Bean Ant script

```
     <adapter:discover>
            <adapter:performImport
agent="{com/ibm/adapter}MPOCobolDiscoveryAgent">
                <adapter:importResource>
                    <adapter:propertyGroup name="CobolFileGroup">
                        <adapter:propertyElement name="CobolFile"
value=
"D:\70Shared\plugins\com.ibm.j2c.cheatsheet.content_7.0.1\Samples\CIC
S32K\ec03.ccp"/>
                    </adapter:propertyGroup>
```

- **Information in wizard**

  The **choose Mapping** is located in the Data Import page inside the CICS/IMS Data binding wizard and J2C Bean wizard.

Figure 30. Discovery agent and the corresponding page in wizard



Import

Data Import

Specify data import configuration properties

Choose mapping: C Language CICS Channel to Java

- COBOL to Java
- COBOL MPO to Java
- COBOL CICS Channel to Java
- C Language CICS Channel to Java
- C MPO to Java

Containers:

The different Property Values for the mapping is located in the Data Import page.

Resource Writer

- **Basic Information**

  While a Discovery Agent reads the language files to generate an in memory model of the artifacts to be generated, the Workspace Resource Writer takes the model and generates the artifacts using the properties specified by the user. And the same as for the agent, the writer is transparent to the user throughout the wizard. Its ID surfaces in the Ant script only.

- **Information in Ant Script**

Listing 11. writer and and write properties in Data Binding Ant script

```
<adapter:writeToWorkspace

writer="{com/ibm/adapter/cobol/cicschannel/writer}JAVA_WRITER" >
  <adapter:propertyGroup name="MPO_PG">
    <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
    <adapter:propertyTree name="MPO_TP" >
      <adapter:propertyNode name="COBOLMPOToJavaResourceWriter" >
        <adapter:propertyGroup name="Java Type Name" >
          <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
          <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
          <adapter:propertyElement name="Class Name"
value="EC03ChannelRecord"/ >
          <adapter:propertyElement name="Overwrite existing class"
value="true"/ >
          <adapter:propertyElement name="CHANNEL_NAME"
value="InputRecord"/ >
        </adapter:propertyGroup >
      </adapter:propertyNode >
      <adapter:propertyNode name="COBOLImportResult0" >
        <adapter:propertyGroup name="COBOLToJavaResourceWriter" >
          <adapter:propertyGroup name="Java Type Name" >
            <adapter:propertyElement name="Project Name"
value="CICS32KSample"/ >
            <adapter:propertyElement name="Package Name"
value="sample.cics"/ >
            <adapter:propertyElement name="Class Name"
value="DateContainer"/ >
            <adapter:propertyElement name="Overwrite existing class"
value="true"/ >
            <adapter:propertyElement name="GenerationStyle"
value="Default"/ >
            <adapter:propertyElement name="CONTAINER_NAME"
value="CURRENTDATE"/ >
            <adapter:propertyElement name="CONTAINER_TYPE"
value="BIT"/ >
```

```
          </adapter:propertyGroup >
        </adapter:propertyGroup >
      </adapter:propertyNode >
```

Listing 12. writer and and write properties in J2C Bean Ant script

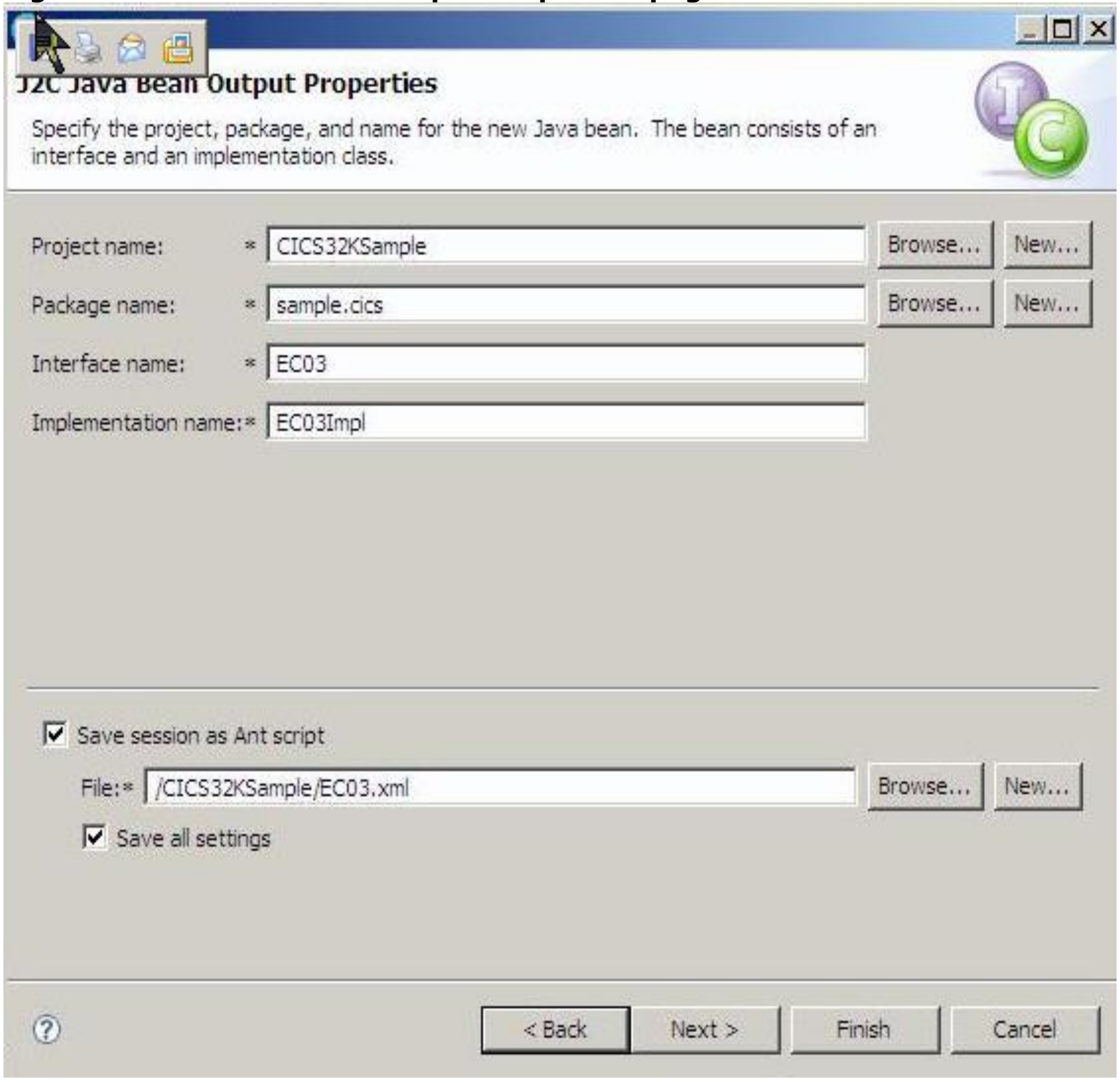
```
<adapter:writeToWorkspace

writer="{com/ibm/adapter/j2c/codegen/writer}J2CAnnotationWorkspaceRes
ourceWriter">
  <adapter:propertyGroup name="J2C Java Bean Writer Properties">
        <adapter:propertyElement name="Project"
value="CICS32KSample"/ >
        <adapter:propertyElement name="PackageName"
value="sample.cics"/>
        <adapter:propertyElement name="InterfaceName" value="EC03"/>
    </adapter:propertyGroup >
</adapter:writeToWorkspace>
```

- **Information in wizard**

  The write properties for the COBOLTOJavaResourceWriter is located in the Saving Properties page of the CICS/IMS Data Binding as shown in previous section from figure 5 through figure 10. This includes the Project, PackageName and InterfaceName.

  The write properties for the J2CAnnotationWorkspaceResourceWriter is located in J2C Java bean Output Properties page as shown below in figure 22. This includes the Project, PackageName and InterfaceName.

Figure 31. J2C Java Bean Output Properties page



J2C Java Bean Output Properties

Specify the project, package, and name for the new Java bean. The bean consists of an interface and an implementation class.

Project name: *	CICS32KSample
Package name: *	sample.cics
Interface name: *	EC03
Implementation name: *	EC03Impl

☑ Save session as Ant script

File: * /CICS32KSample/EC03.xml

☑ Save all settings

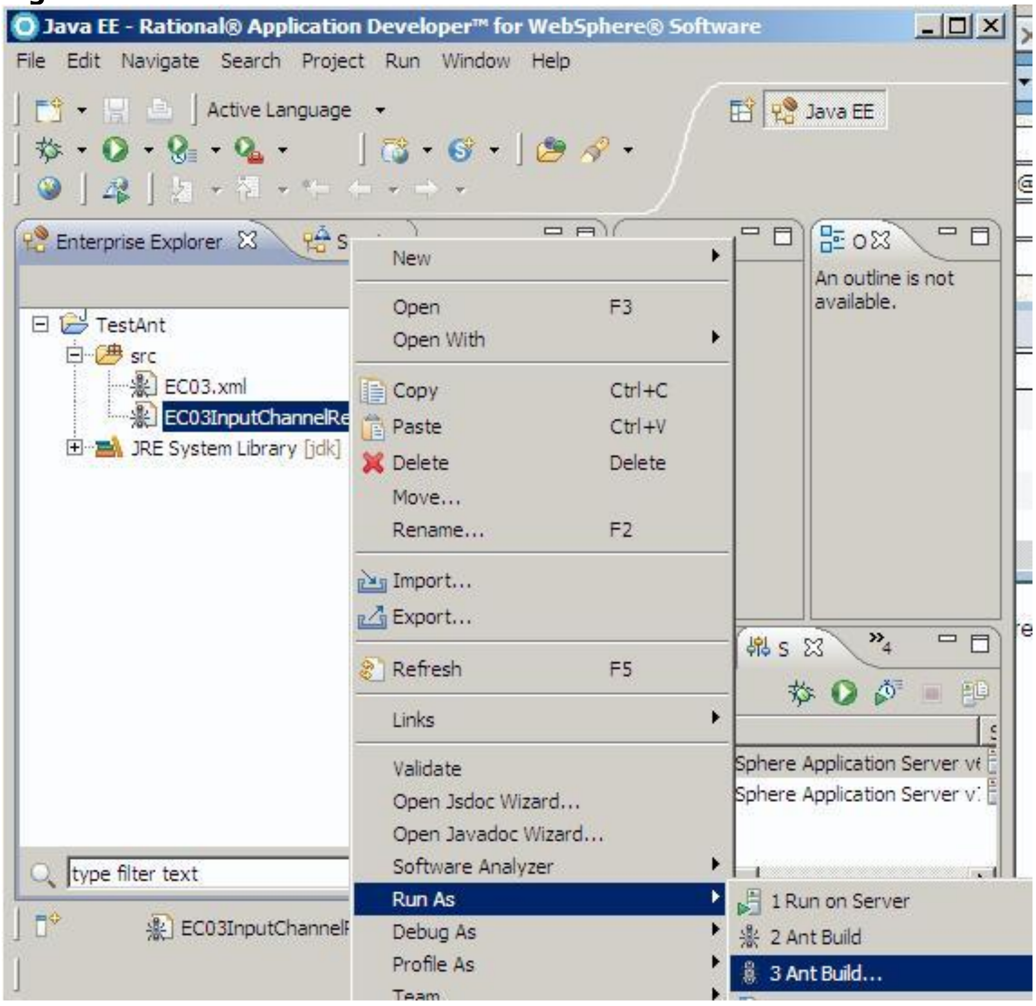
Running Ant Script

Running Ant Script in the workspace

Here are the steps to run the ant script inside the Rational Application Developer for WebSphere Software v7.5

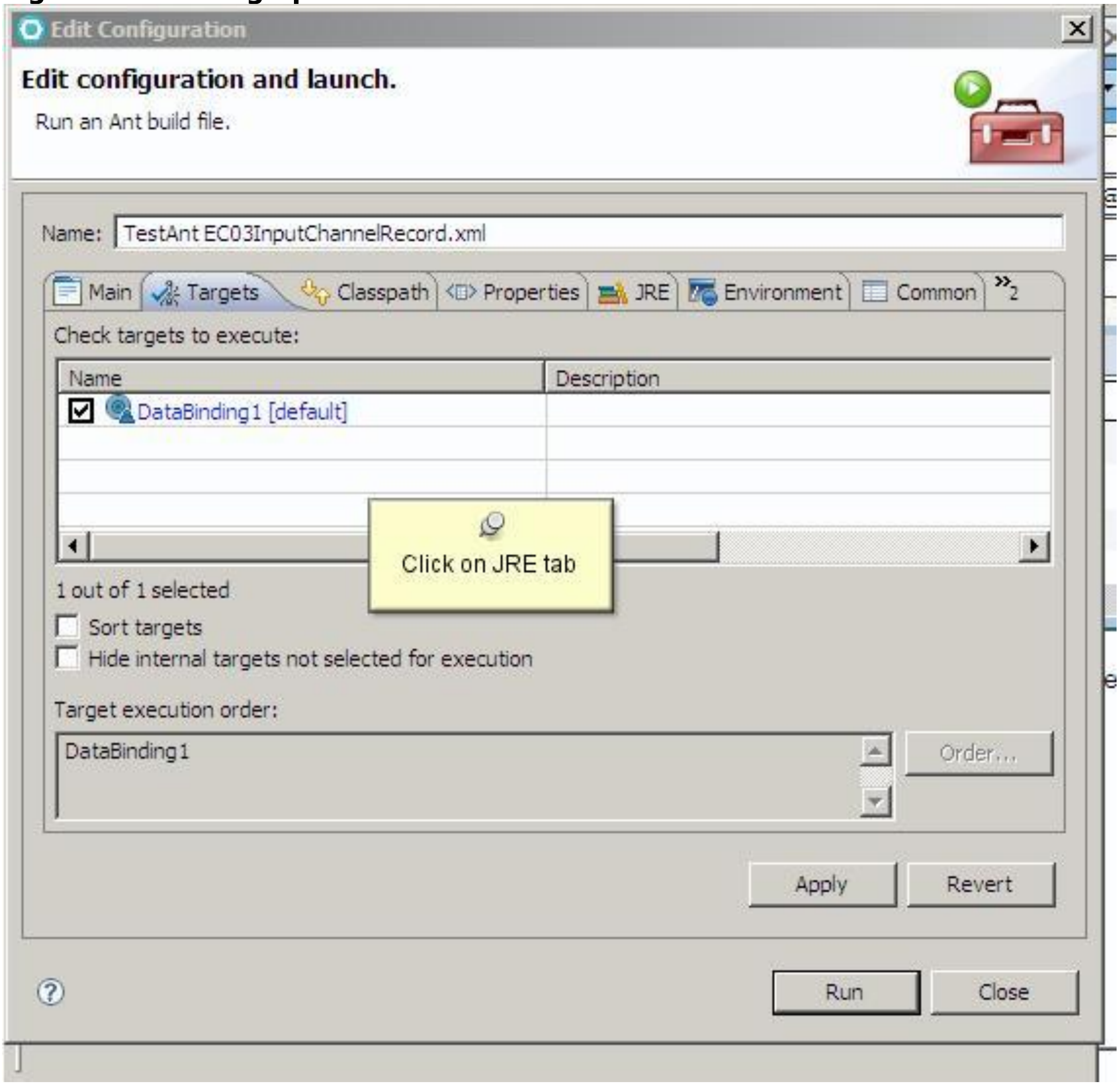
- Bring up a clean workspace.
- Create a Project and copy the generated ant script into the project. In this example, we will copy the EC03InputChannelRecord.xml which will generate the Channel data binding files and the container data binding files.
- Right Click on the Ant script file and select **Run As > 3 Ant Build..** as shown in figure 32. You only need to do this once in each workspace to set up the JRE environment. The next time you can select **Run As > 2 Ant Build**

Figure 32. Run As > 3 AntBuild



- In the Edit Configuration and launch page, click on the JRE tab.

Figure 33. Setting up JRE



Edit Configuration

Edit configuration and launch.

Run an Ant build file.

Name: TestAnt EC03InputChannelRecord.xml

Check targets to execute:

Name	Description
☑ DataBinding1 [default]	

1 out of 1 selected

☐ Sort targets

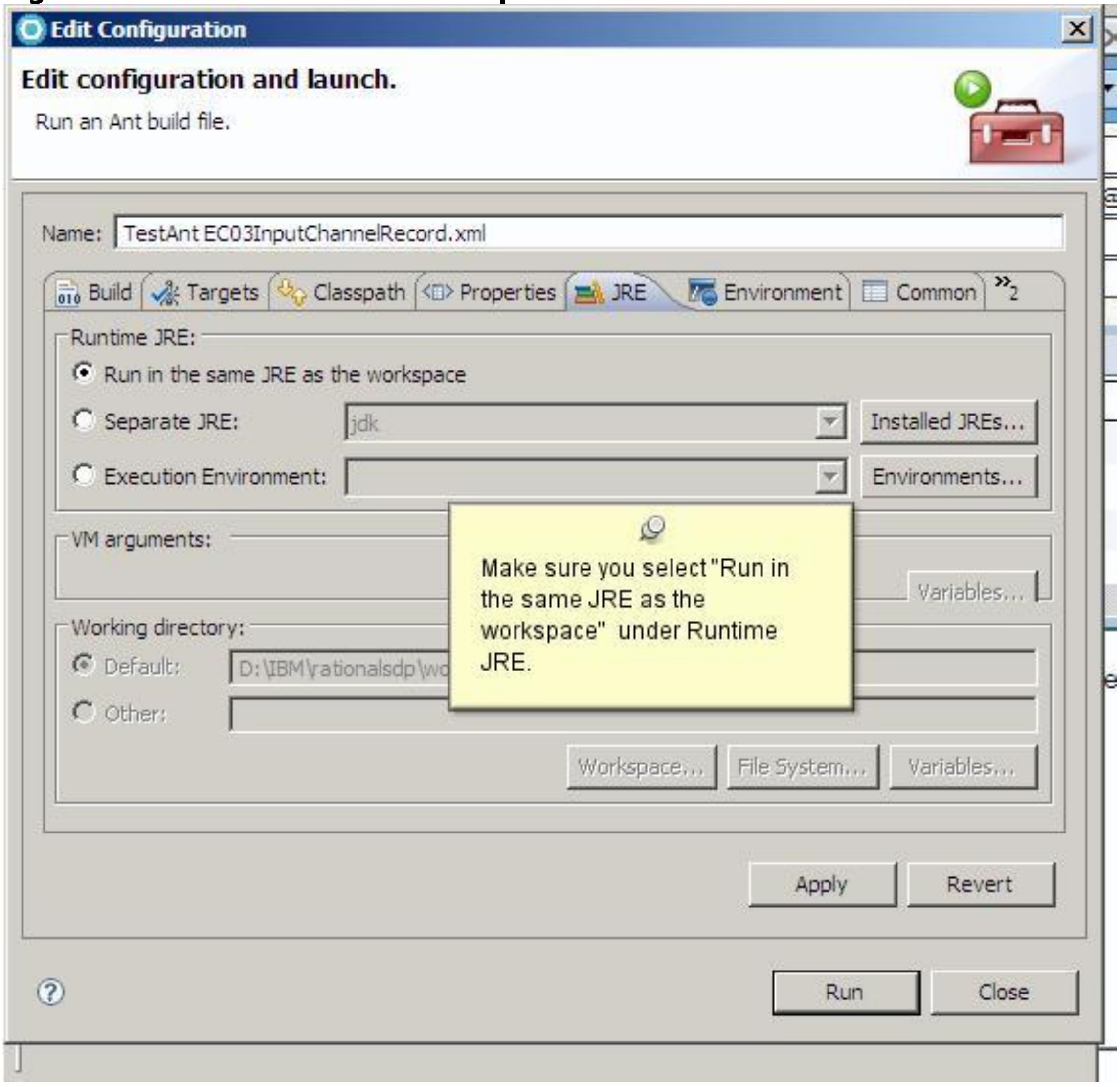
☐ Hide internal targets not selected for execution

Target execution order:

DataBinding1

- Select **run in the same JRE as the workspace.**

Figure 34. Same JRE as the workspace



Edit Configuration

Edit configuration and launch.

Run an Ant build file.

Name: TestAnt EC03InputChannelRecord.xml

Runtime JRE:

(● Run in the same JRE as the workspace)

○ Separate JRE: jdk

○ Execution Environment:

VM arguments:

Working directory:

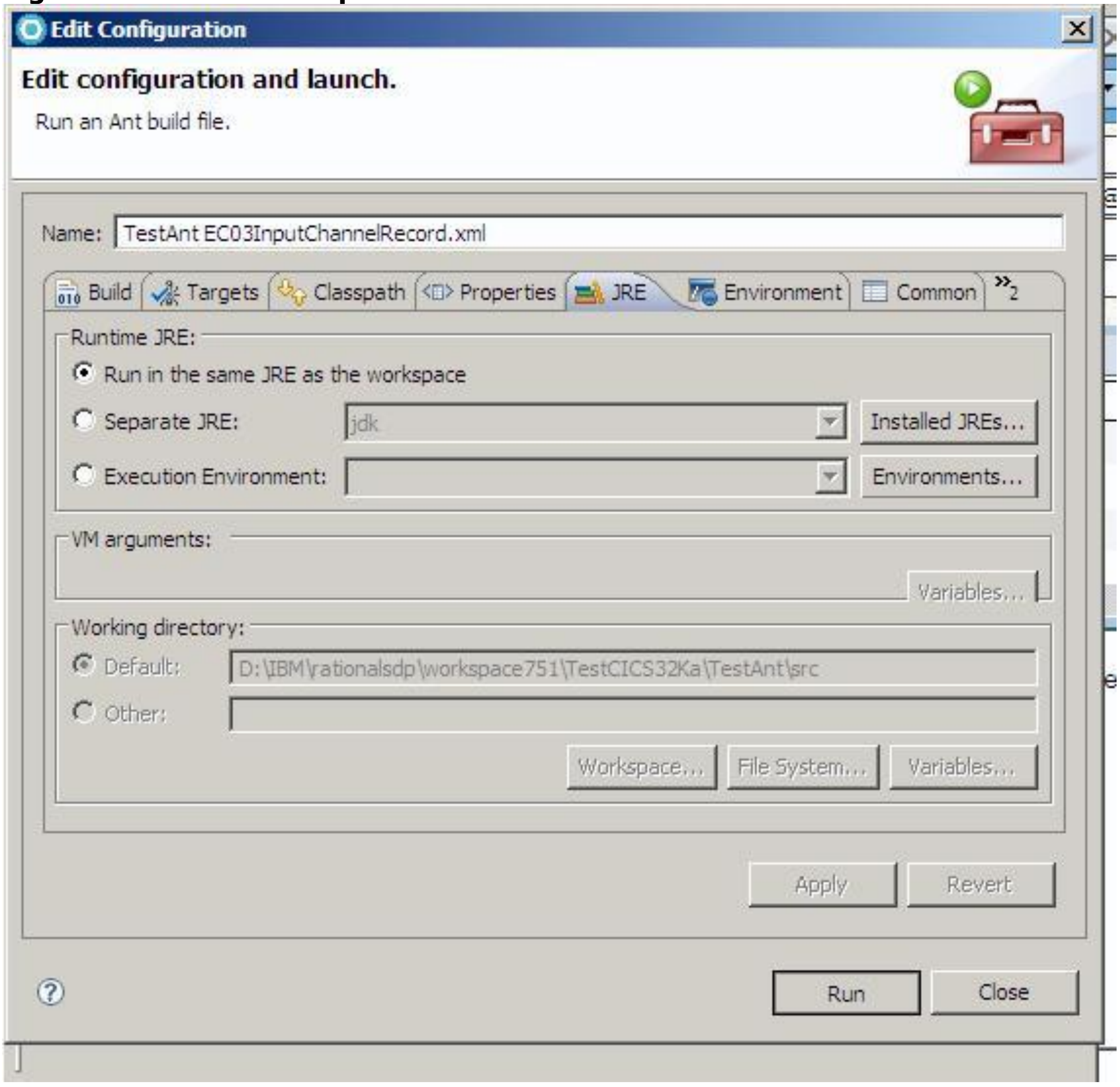
● Default: D:\IBM\rationalsdp\wo

○ Other:

Make sure you select "Run in the same JRE as the workspace" under Runtime JRE.

- Click on the **Run** button to execute the ant script

Figure 35. Run Ant script



Edit Configuration

Edit configuration and launch.

Run an Ant build file.

Name: TestAnt EC03InputChannelRecord.xml

Runtime JRE:

- [✓ Run in the same JRE as the workspace]
- Separate JRE: jdk
- Execution Environment:

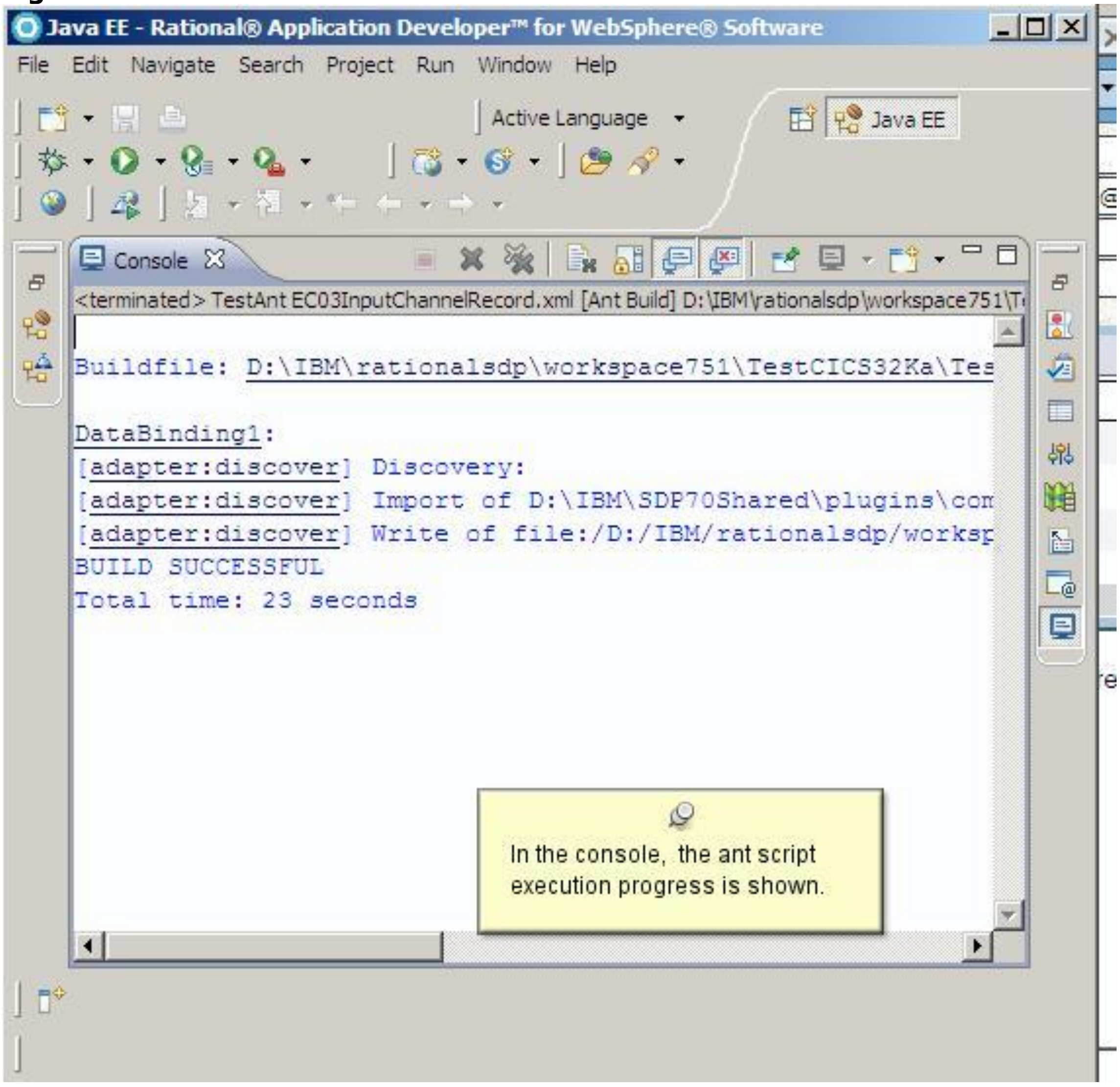
VM arguments:

Working directory:

- [✓ Default:] D:\IBM\rationalsdp\workspace751\TestCICS32Ka\TestAnt\src
- Other:

- In the Console, you will see the execution progress of the ant script. if you see "Build Success", the ant script is executed successfully

Figure 36. Console

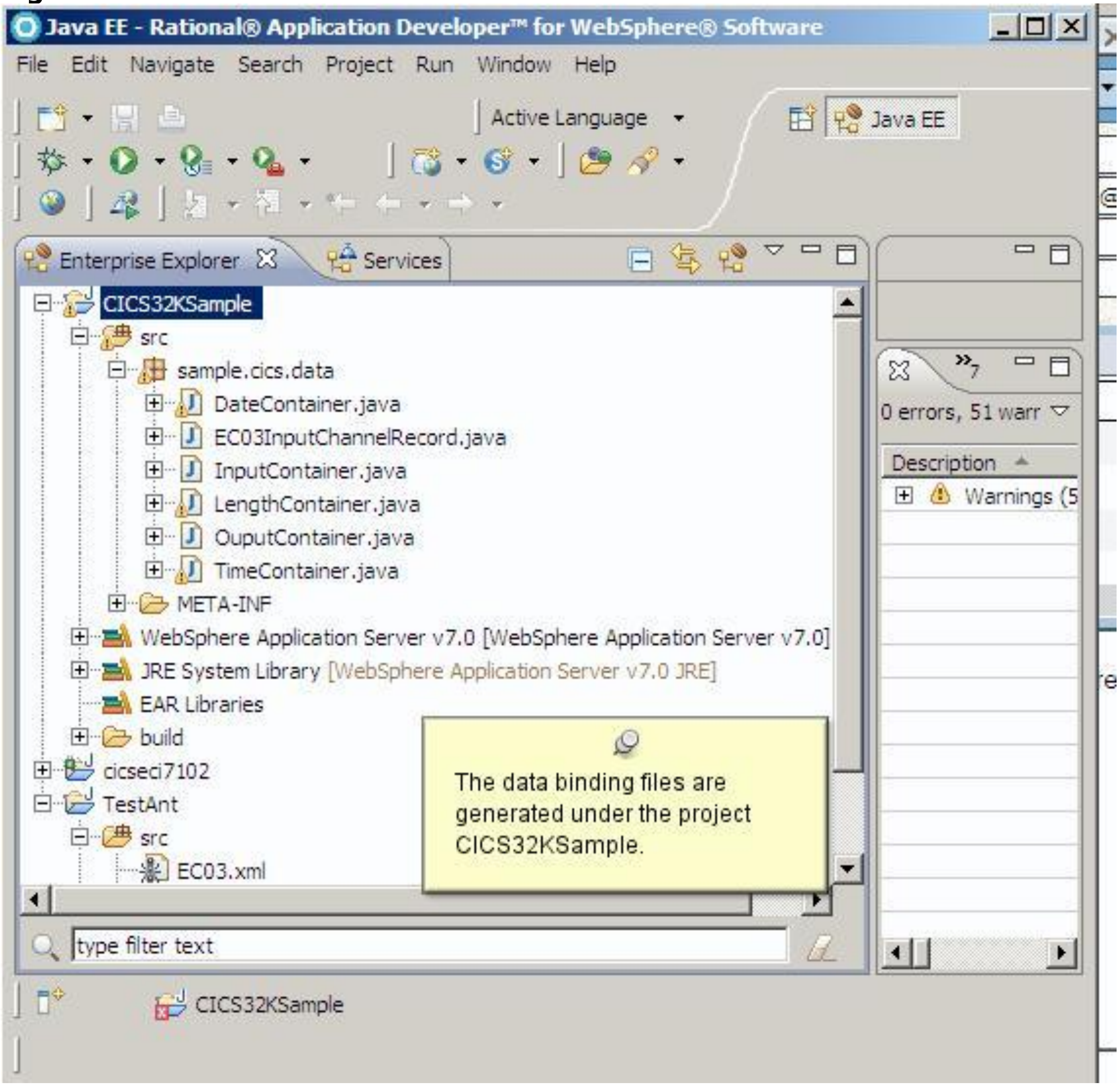

Java EE - Rational® Application Developer™ for WebSphere® Software
File Edit Navigate Search Project Run Window Help
Active Language
Java EE
Console

```
<terminated> TestAnt EC03InputChannelRecord.xml [Ant Build] D:\IBM\rationalsdp\workspace751\T
Buildfile: D:\IBM\rationalsdp\workspace751\TestCICS32Ka\Tes

DataBinding1:
[adapter:discover] Discovery:
[adapter:discover] Import of D:\IBM\SDP70Shared\plugins\com
[adapter:discover] Write of file:/D:/IBM/rationalsdp/worksp
BUILD SUCCESSFUL
Total time: 23 seconds
```

In the console, the ant script execution progress is shown.

- In the Enterprise Explorer view, you will see the generated data binding files.

Figure 37. Generated files



The data binding files are generated under the project CICS32KSample.

Running Ant Script from command line

The generated Data binding Ant scripts can be executed without launching the eclipse workbench. Once generated or modified, you can execute a script from the Microsoft Windows command line without bringing up the IDE. Running Rational Application Developer for WebSphere Software in this way is called headless mode. To run the Ant script, invoke the antRunner application passing the Ant file as argument. A simple batch file for running the generated Ant scripts in headless mode is shown in Listing 14, following:

Listing 14. Batch file to invoke data binding Ant script

```
echo on
setlocal

set ECLIPSE="D:\IBM\SDP750"
set WORKSPACE=D:\workspace\testAntCICS32K
set BUILDFILE=D:\workspace\EC03InputChannelRecord.xml


set JAVA_HOME=%ECLIPSE%\jdk
SET
EQUINOXJAR=%ECLIPSE%\plugins\org.eclipse.equinox.launcher_1.0.100.v20
080509-1800.jar
set PATH=%JAVA_HOME%\bin;%PATH%
set CLASSPATH=%JAVA_HOME%\lib;%CLASSPATH%

echo "Generating the databinding file.."
java -cp %EQUINOXJAR% org.eclipse.core.launcher.Main -clean -data
%WORKSPACE%
-application org.eclipse.ant.core.antRunner -buildfile %BUILDFILE%
```

Where:

- ECLIPSE environment variable defines the path to the eclipse folder within Rational Application Developer for WebSphere Software
- WORKSPACE defines the path where the workspace will be created
- BUILDFILE is the path to your generated Ant script you would like to run.

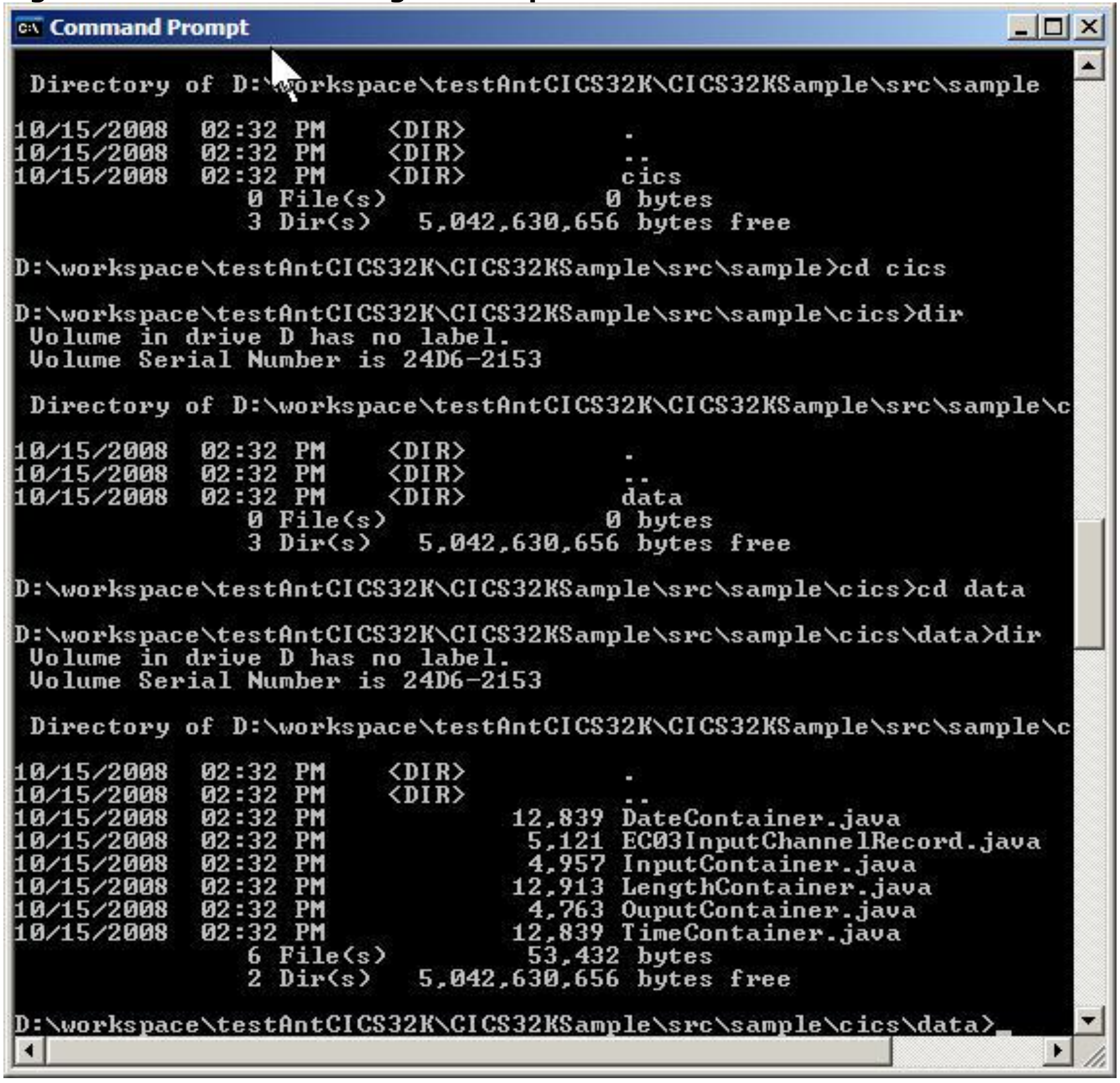
To run the Ant file generated from the CICS/IMS Data Binding session, modify the values of the following variables in the Test.bat batch file to fit your environment.

- Replace the value for *ECLIPSE* with your eclipse root directory (where eclipse.exe is located)
- Replace the value for *WORKSPACE* with the workspace name and location
- Replace the value for *BUILDFILE* with your J2C Ant script
- Make sure that the %EQUIOXJAR% is correct, check if that jar file exists. The name of the org.eclipse.equinox.launcher_xxx.jar may be different, then you need to make modifications

If the debug property is set to True, you should see progress messages and a **BUILD SUCCESSFUL** message at the end. You can also browse to the target workspace to see the generated files.

To run the Data binding ant script from the command line, you can invoke Test.bat as show in figure 38.

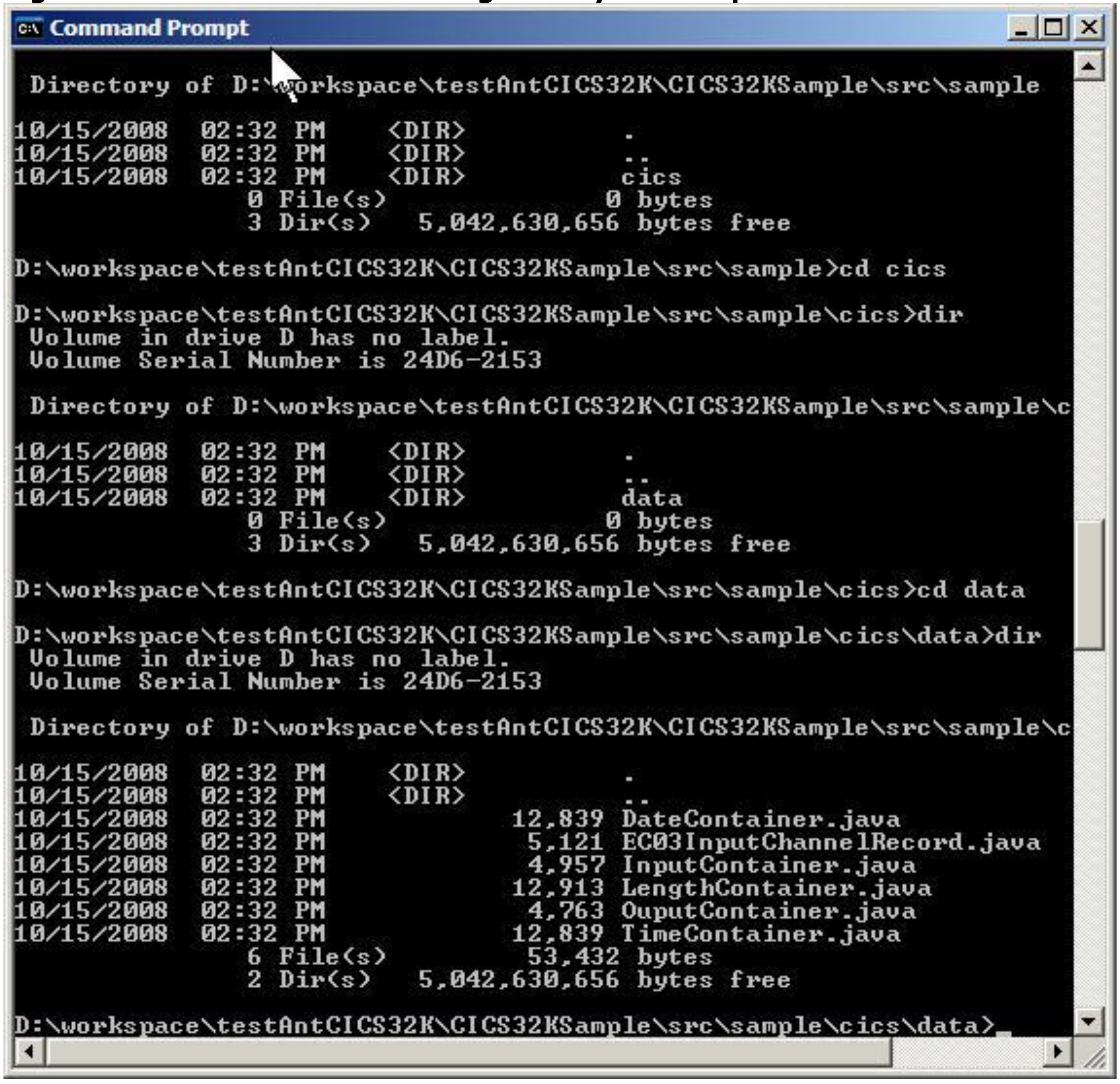
Figure 38. Console showing Ant script execution



```
 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM    <DIR>          cics
               0 File(s)              0 bytes
               3 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample>cd cics

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics>dir
 Volume in drive D has no label.
 Volume Serial Number is 24D6-2153

 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample\c

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM    <DIR>          data
               0 File(s)              0 bytes
               3 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics>cd data

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics\data>dir
 Volume in drive D has no label.
 Volume Serial Number is 24D6-2153

 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample\c

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM            12,839 DateContainer.java
10/15/2008  02:32 PM             5,121 EC03InputChannelRecord.java
10/15/2008  02:32 PM             4,957 InputContainer.java
10/15/2008  02:32 PM            12,913 LengthContainer.java
10/15/2008  02:32 PM             4,763 OuputContainer.java
10/15/2008  02:32 PM            12,839 TimeContainer.java
               6 File(s)         53,432 bytes
               2 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics\data>_
```

After the test.bat completes successfully, you can go to the workspace and all the projects and data binding files are created, as shown in figure 37.

Figure 39 Generated data binding files by Ant script



Command Prompt

```
 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM    <DIR>          cics
               0 File(s)              0 bytes
               3 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample>cd cics

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics>dir
 Volume in drive D has no label.
 Volume Serial Number is 24D6-2153

 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample\c

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM    <DIR>          data
               0 File(s)              0 bytes
               3 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics>cd data

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics\data>dir
 Volume in drive D has no label.
 Volume Serial Number is 24D6-2153

 Directory of D:\workspace\testAntCICS32K\CICS32KSample\src\sample\c

10/15/2008  02:32 PM    <DIR>          .
10/15/2008  02:32 PM    <DIR>          ..
10/15/2008  02:32 PM            12,839 DateContainer.java
10/15/2008  02:32 PM             5,121 EC03InputChannelRecord.java
10/15/2008  02:32 PM             4,957 InputContainer.java
10/15/2008  02:32 PM            12,913 LengthContainer.java
10/15/2008  02:32 PM             4,763 OuputContainer.java
10/15/2008  02:32 PM            12,839 TimeContainer.java
               6 File(s)         53,432 bytes
               2 Dir(s)   5,042,630,656 bytes free

D:\workspace\testAntCICS32K\CICS32KSample\src\sample\cics\data>_
```

Tips

The default WebSphere Application Server runtime is v7.0 in Rational Application Developer for WebSphere Software v7.5.x.

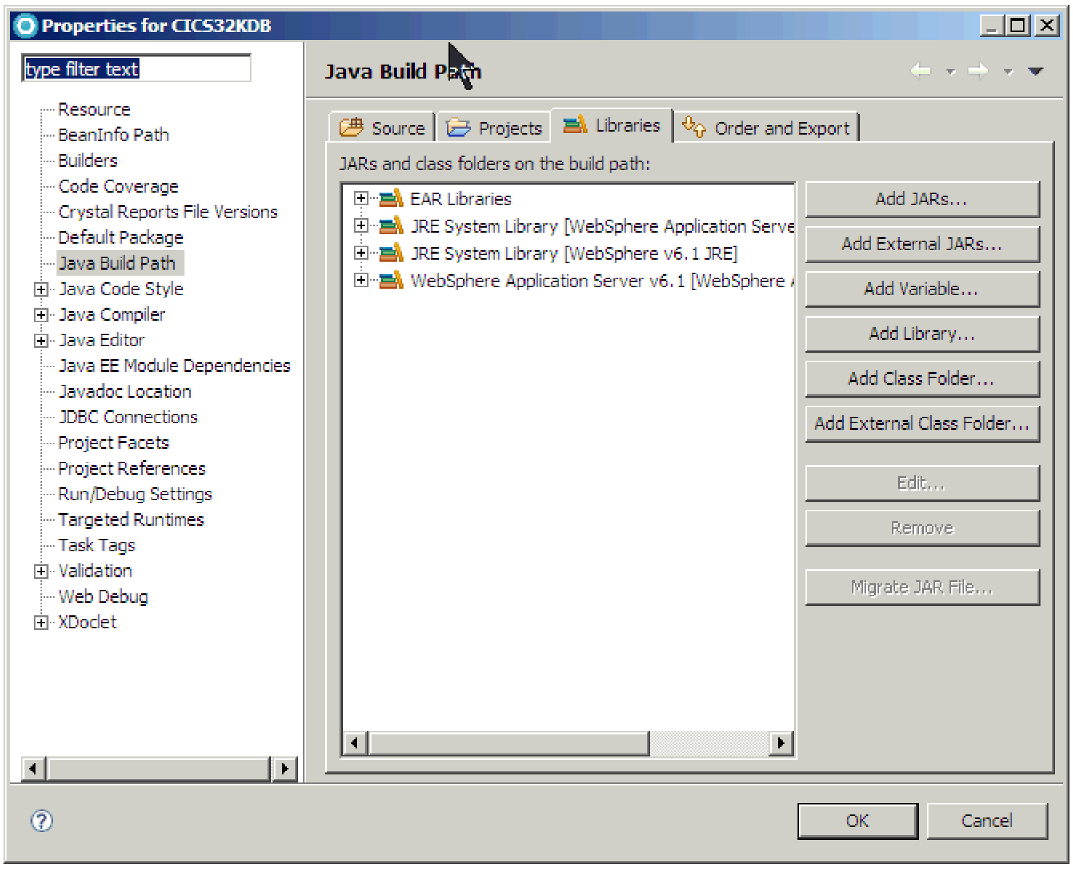
For Java project, this is the default runtime always selected when you execute the ant script.

Always ensure the following projects are consistent in their WebSphere Application Server runtime and JRE level to avoid undesired behavior.

- Connector project
- Project for Data binding generation
- Project for J2C Java Beans generation (can be same project as Data binding generation)

To modify the WebSphere Application Server runtime in the project, do one of the following to bring up the Java Build Path

- Click on the project and select Properties and click Java Build Path
  or
- Click on the project, select Build Path > Configure Built Path

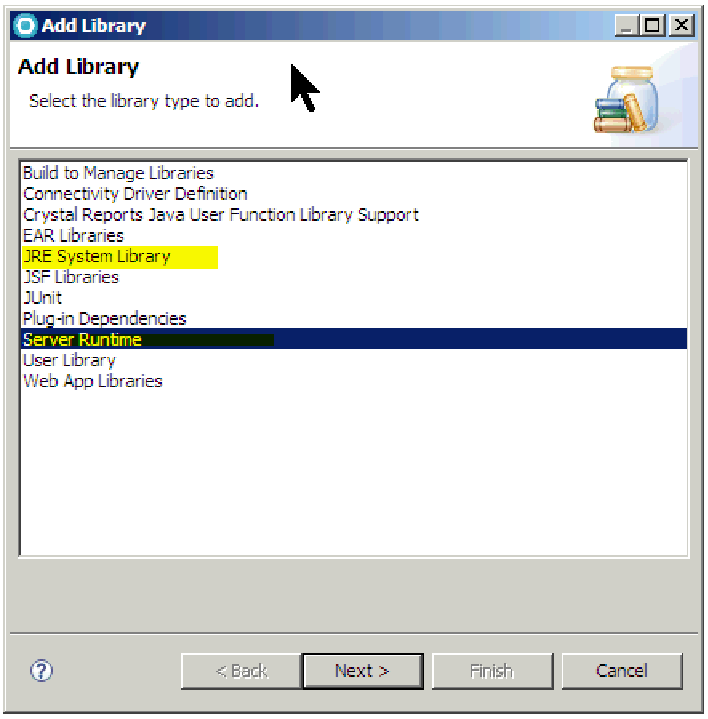


JARs and class folders on the build path:

- EAR Libraries
- JRE System Library [WebSphere Application Serve
- JRE System Library [WebSphere v6.1 JRE]
- WebSphere Application Server v6.1 [WebSphere /

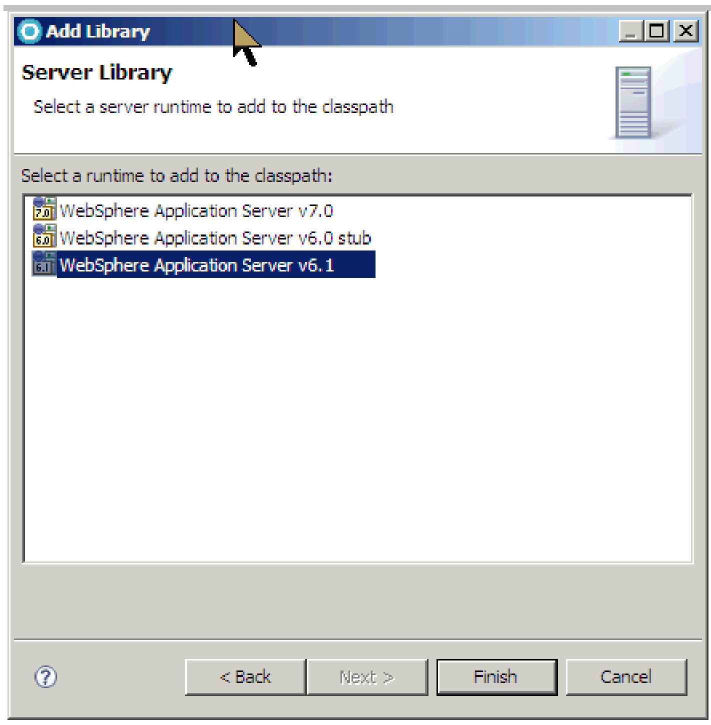
You can select **Add Library** to select the right Server runtime version and JRE version you want.
To change the Server Runtime.

Select Server runtime in the Add Library



Build to Manage Libraries
Connectivity Driver Definition
Crystal Reports Java User Function Library Support
EAR Libraries
JRE System Library
JSF Libraries
JUnit
Plug-in Dependencies
Server Runtime
User Library
Web App Libraries

Select the server runtime you desired, press Finish.



Add Library

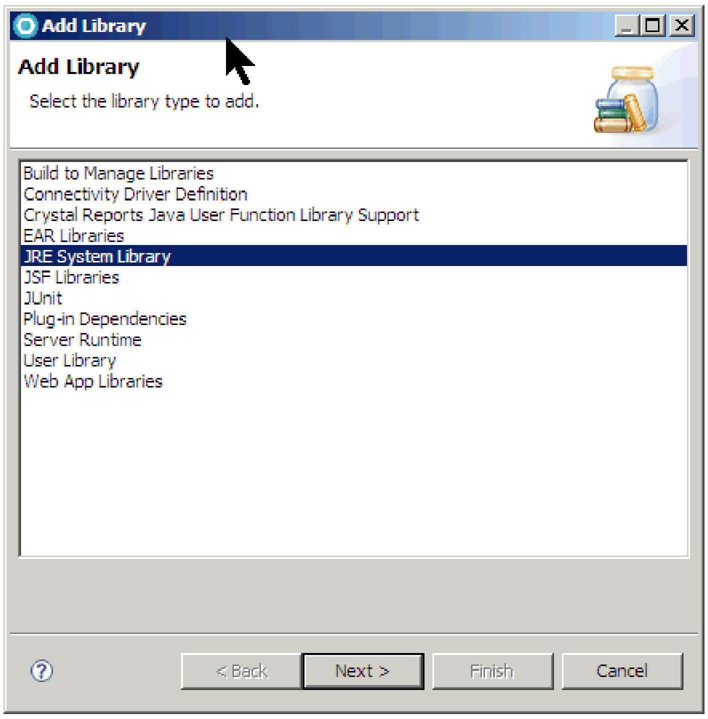
Server Library

Select a server runtime to add to the classpath

Select a runtime to add to the classpath:

- WebSphere Application Server v7.0
- WebSphere Application Server v6.0 stub
- WebSphere Application Server v6.1

To change JRE version, in Add Library.



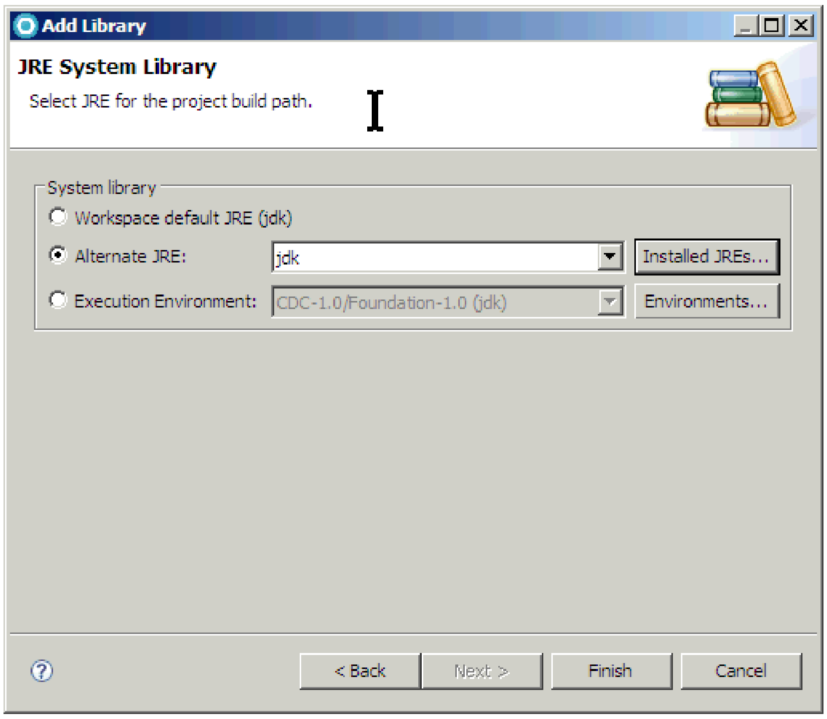
Add Library

Add Library

Select the library type to add.

- Build to Manage Libraries
- Connectivity Driver Definition
- Crystal Reports Java User Function Library Support
- EAR Libraries
- JRE System Library
- JSF Libraries
- JUnit
- Plug-in Dependencies
- Server Runtime
- User Library
- Web App Libraries

Select Alternate JRE, click on Installed JREs.. button


Add Library

JRE System Library

Select JRE for the project build path.

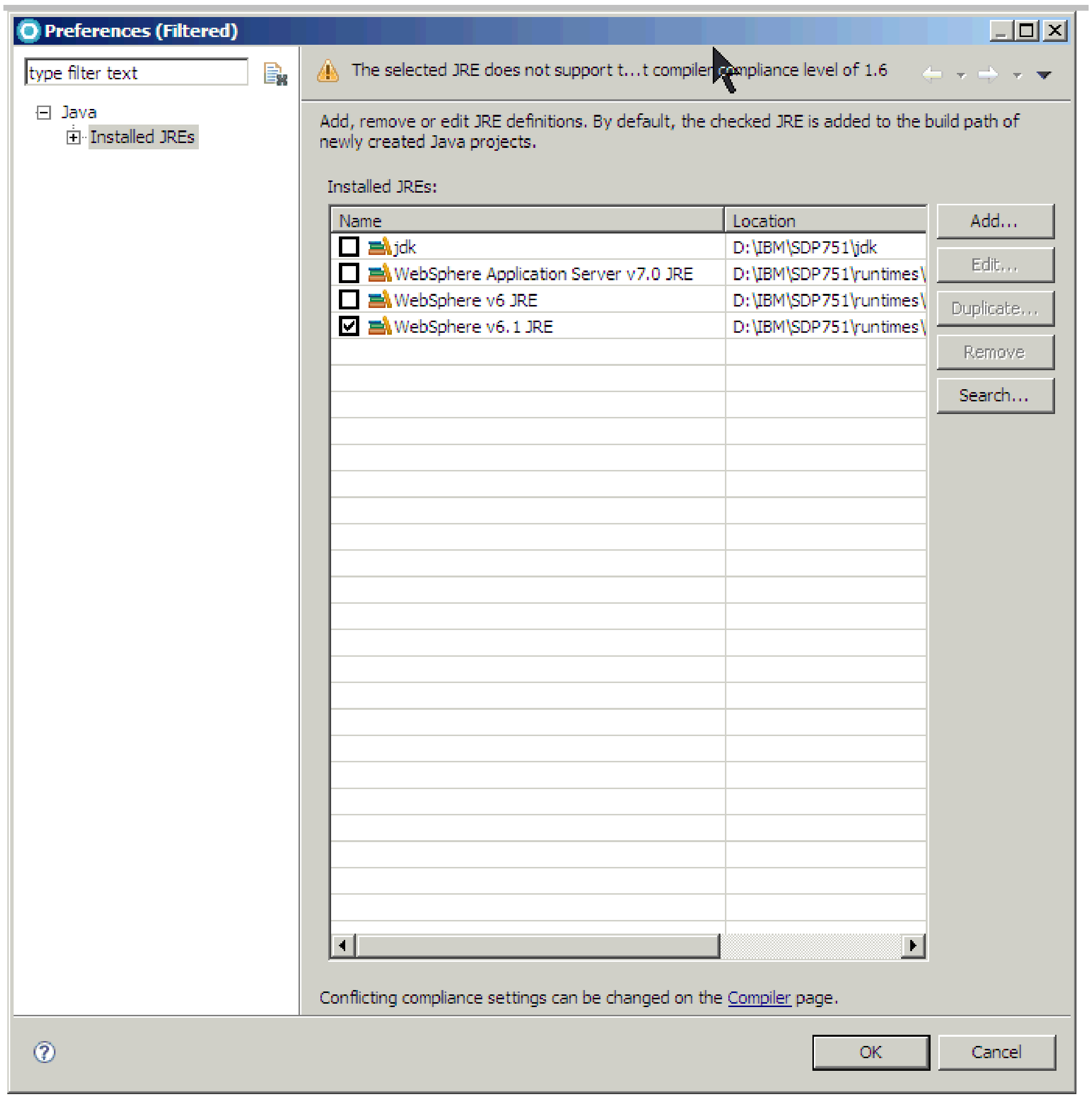
System library

( ) Workspace default JRE (jdk)

[✓ Alternate JRE:] jdk

( ) Execution Environment: CDC-1.0/Foundation-1.0 (jdk)

Select the JRE you desire.



Preferences (Filtered)

type filter text

Java
Installed JREs

The selected JRE does not support t...t compiler compliance level of 1.6

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

	Name	Location
☐	jdk	D:\IBM\SDP751\jdk
☐	WebSphere Application Server v7.0 JRE	D:\IBM\SDP751\runtimes\
☐	WebSphere v6 JRE	D:\IBM\SDP751\runtimes\
☑	WebSphere v6.1 JRE	D:\IBM\SDP751\runtimes\

Conflicting compliance settings can be changed on the Compiler page.

Summary

This article has shown you how to create J2C Java Beans and Data Binding beans from a Cobol program that support CICS Channel and save the session information into an Ant scripts.

You can run the ant script to regenerate the same artifacts without going through the wizard again.

You can also customize the ant script to generate similar artifacts.

Resources

Learn

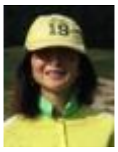
- The out-of-service date of your version of the WebSphere Application Server.
- Using Ant with WebSphere Studio Application Developer.
- Troubleshooting headless Ant builds with Rational Application Developer.
- Exploiting CICS Channels and Containers from Java clients,
- In the Architecture area on developerWorks, get the resources you need to advance your skills in the architecture arena.
- Browse the technology bookstore for books on these and other technical topics.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

About the Authors

Laszlo is currently working on the Adapter Tooling team in the IBM WebSphere area. He has a University and a Master's degree in Computer Science.



Ivy leads the Java Connector Tools team for Rational software. She earned a degree in mathematics with honors (major in computer science, minor in statistics) from the University of Waterloo. She is a certified Project Management Professional (PMP).