

Developing and testing an MDB using RAD 7.5, WebSphere Application Server V7 and MQ V7 as JMS Provider

IBM Techdoc: 7016507

<http://www.ibm.com/support/docview.wss?rs=171&uid=swg27016507>

Date last updated: 30-Jun-2010

Angel Rivera - rivera@us.ibm.com
IBM WebSphere MQ Support

+++ Objective +++

To demonstrate all the steps to develop and test a Message Driven Bean (MDB) using Rational Application Developer (RAD) 7.5 and WebSphere Application Server V7, while using WebSphere MQ V7 as the Java™ Messaging Service (JMS) Provider.

The testing scenario shows how the Listener Port for WebSphere Application Server gets a message from an MQ Queue (Point to Point) and passes it to the MDB, which displays the contents of the text message. This MDB is a simple but functional application.

The following file (available with this techdoc) includes the MDB as it was developed, tested and exported as described in this document:
EAR file with MDB: SampleMDBEJB.ear

This document shows all the screen shots needed for beginners.

This document has the following chapters:

Chapter 1: RAD 7.5 development of the MDB

Chapter 2: RAD 7.5 testing of the MDB using WebSphere Application Server 7

+++ Related techdoc +++

It is necessary to configure MQ and WebSphere Application Server with the objects that are used in this scenario. Consult the following techdoc for details:

Using WebSphere MQ V7 as JMS Provider for WebSphere Application Server V7

<http://www.ibm.com/support/docview.wss?rs=171&uid=swg27016505>

Chapter 1: MQ V7 configuration: queue and topic

Chapter 2: WebSphere Application Server V7 configuration: JNDI objects

+++ Requisite software +++

The following software was used:

SUSE Linux Enterprise Server (SLES) 9:
Rational Application Developer 7.5
WebSphere Application Server 7.0.0.5
WebSphere MQ 7.0.0.2
Firefox (also known as Mozilla)

+++ Downloadable files +++

The following files are included as attachments to this techdoc

EAR file with MDB:
SampleMDBEJB.ear

Text file with code excerpt:
onMessage.txt

+++ Caveat +++

I encountered authorization problems with RAD and WebSphere Application Server/MQ when using a non-root user that belonged to the "mqm" group (MQ Administrator). In order to keep the scenario as simple as possible, I decided to perform the development and testing as user "root". In order for you to perform the steps described in this techdoc you will need to do the following:

- Add the user "root" to the UNIX group "mqm". All UNIX users in the mqm group are given full administrative authority in WebSphere MQ.

+++ Additional note on EJB 3.0 +++

Even though the instructions are EJB 2.1 and Listener Ports, I think that it is possible to modify the deployment descriptor to include Activation Specifications.

Regarding EJB 3.0 see the following :

<http://www-01.ibm.com/support/docview.wss?uid=swg21307203>

Information about using WebSphere MQ as the JMS Provider for WebSphere Application Server Version 6.1

"Using the WebSphere MQ messaging provider with the Feature Pack for EJB 3.0
The use of the WebSphere MQ messaging provider with EJB 3.0 message-driven beans (by way of the WebSphere Application Server V6.1 Feature Pack for EJB 3.0) is supported using Listener Ports. APAR PK86005 is required."

+++ About the MDB +++

The `onMessage()` method of this MDB has the following source code which displays the type of contents (payload) and an “eye catcher string” (+++ SAMPLE MDB) which can let you find quickly the output of the MDB in the `SystemOut.log` file.

The whole source for this method is available in the following text file associated with this techdoc:

`onMessage.txt`

```
public void onMessage(javax.jms.Message msg) {
    try {
        if (msg instanceof javax.jms.TextMessage) {
            System.out.println("+++ SAMPLE MDB: Text Message
=> " + ((javax.jms.TextMessage)msg).getText());
        }
    }
    ...
}
```

For a Text Message, the actual text of the message is displayed, such as “TESTING”. Thus, upon receiving a message the MDB will display the following in the `SystemOut.log`:

```
+++ SAMPLE MDB: Text Message => TESTING
```

+ Summary of objects and field values

EJB Project:	SampleMDBEJB
EAR Project Name:	SampleMDBEJBEAR
Message-driven bean: "Bean Name":	SampleMDB
Listener type:	Javax.jms.MessageListener
destinationType:	javax.jms.Queue
WebSphere Bindings:	Listener Port
Listener Port Name:	SampleMDBQueueLP
WebSphere profile name:	AppSrv01
WebSphere server name:	server1
Server's host name:	localhost
Server type:	WebSphere Application Server v7.0
Server runtime environment:	WebSphere Application Server v7.0

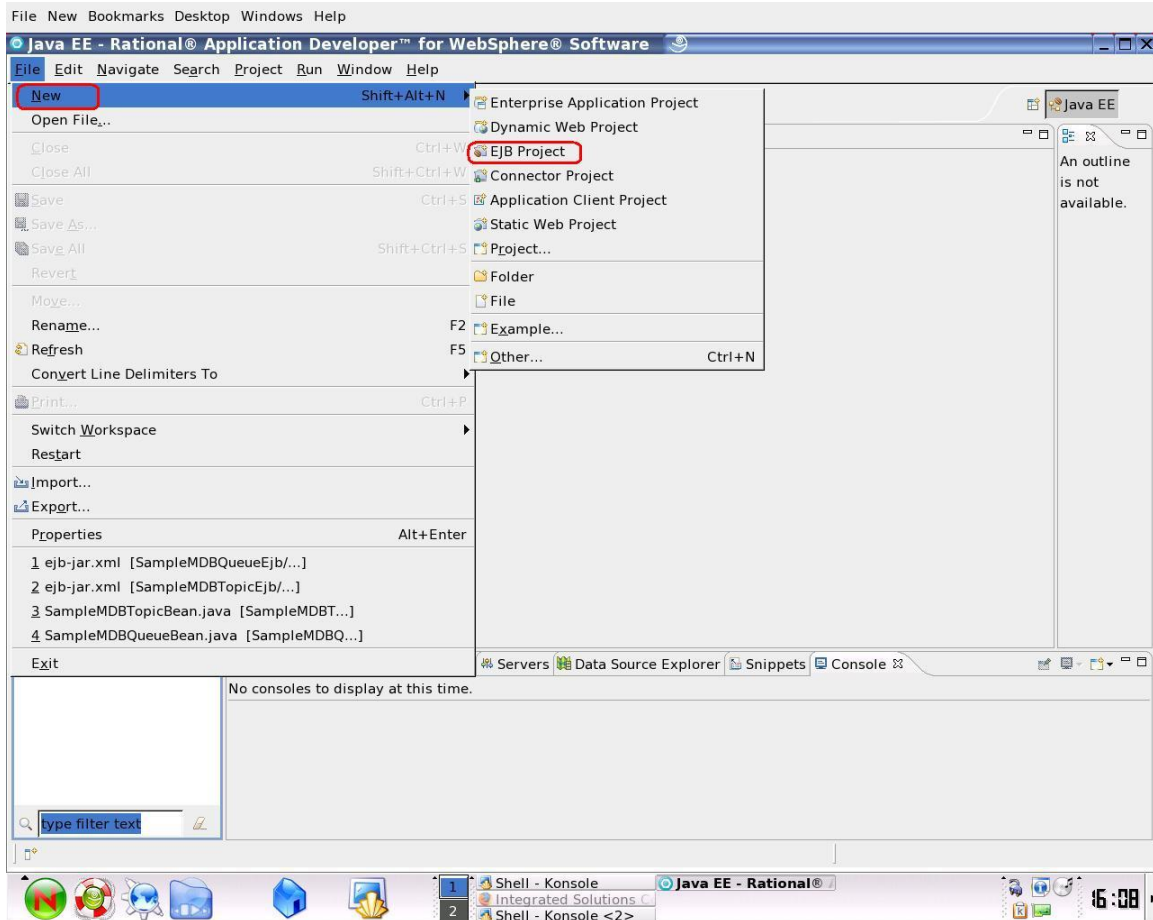
++++
+++ Chapter 1: RAD 7.5 development of the MDB
++++

The purpose for this chapter is to show the creation an Enterprise Java Bean (EJB) Project that as a Message Driven Bean (MDB) in RAD.

Start RAD. From a command prompt issue:
`/opt/IBM/SDP/eclipse`

You could add an icon into your desktop that issues the above command.

+ From RAD, create a New “EJB Project”. From the main menu bar, click:
File > New > EJB Project
Or
File > New > Other > EJB Project



+ Enter the name of the EJB Project. In this techdoc the following name is used:

EJB Project: SampleMDBEJB

- Field “EJB Module Version”: It is necessary to specify a version of 2.1 in order to use Listener Ports.

Even though EJB 3.0 MDBs can be deployed against the WebSphere MQ 7.0 resource adapter activation specifications, the EJB 3.0 does not support Listener Ports.

- Field “Target Runtime”: Specify the proper version of WebSphere Application Server. In this case, it is a standalone WebSphere Application Server 7, thus, you can specify:

“WebSphere Application Server v7.0 stub”

- Field “EAR Membership”: check the box “Add project to an EAR” and accept the value for “EAR Project Name”:

SampleMDBEJBEAR

New EJB Project

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

Project name:

Contents

Use default

Directory:

Target Runtime

EJB Module version

Configuration

You can later add functions to your project by modifying the project facets (right-click a project and

EAR Membership

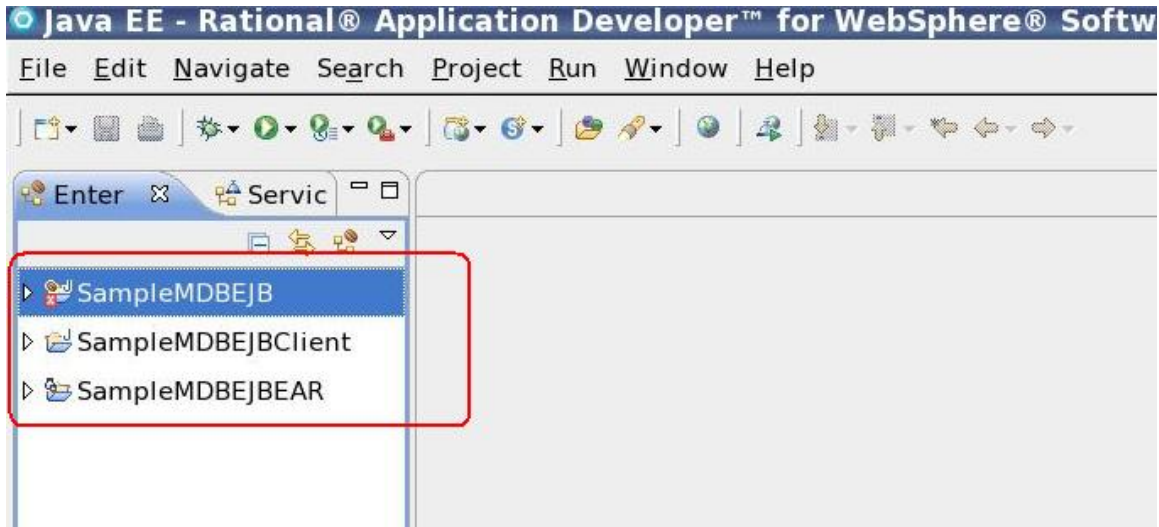
Add project to an EAR

EAR Project Name:

Click on Next and accept the subsequent defaults.
Then click on Finish.

Now you should see 3 folders in the left navigation panel “Workspace”:

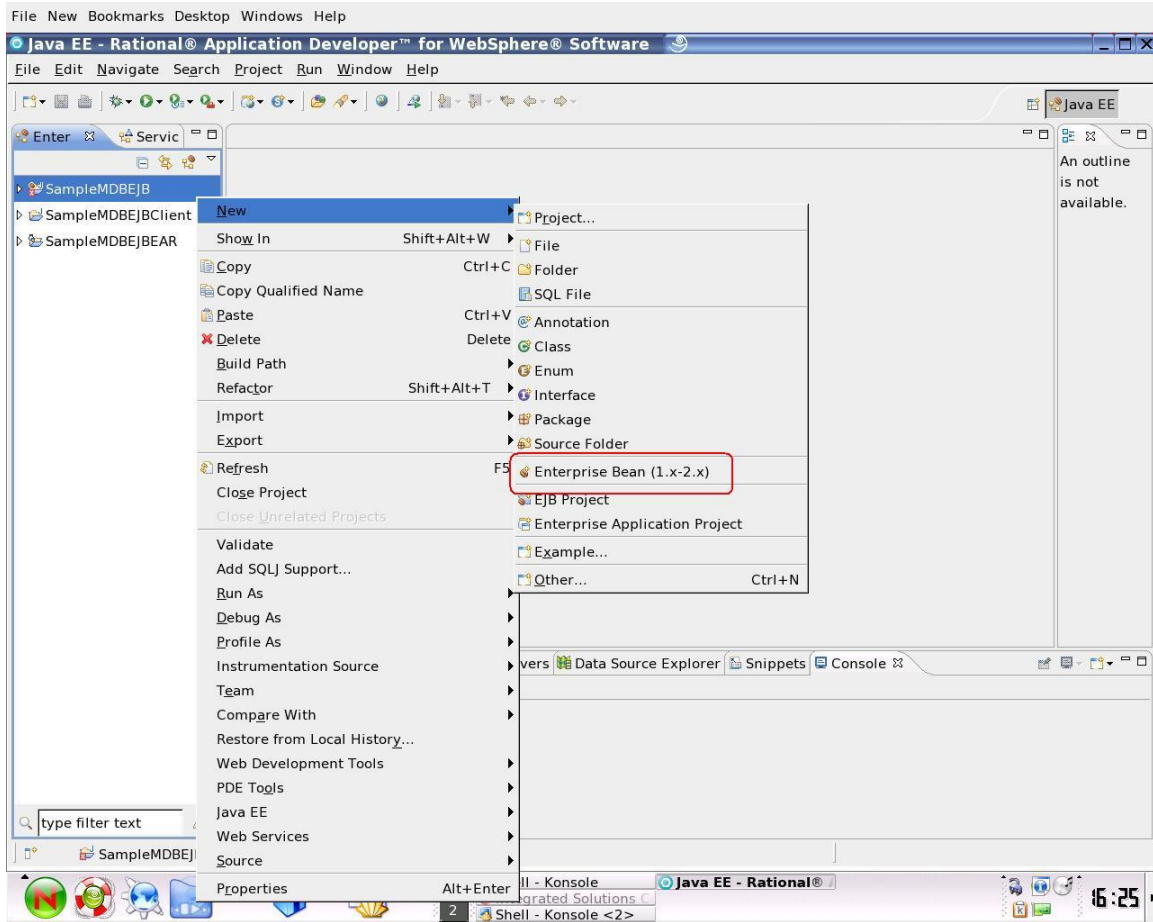
SampleMDBEJB
SampleMDBEJBClient
SampleMDBEJBEAR



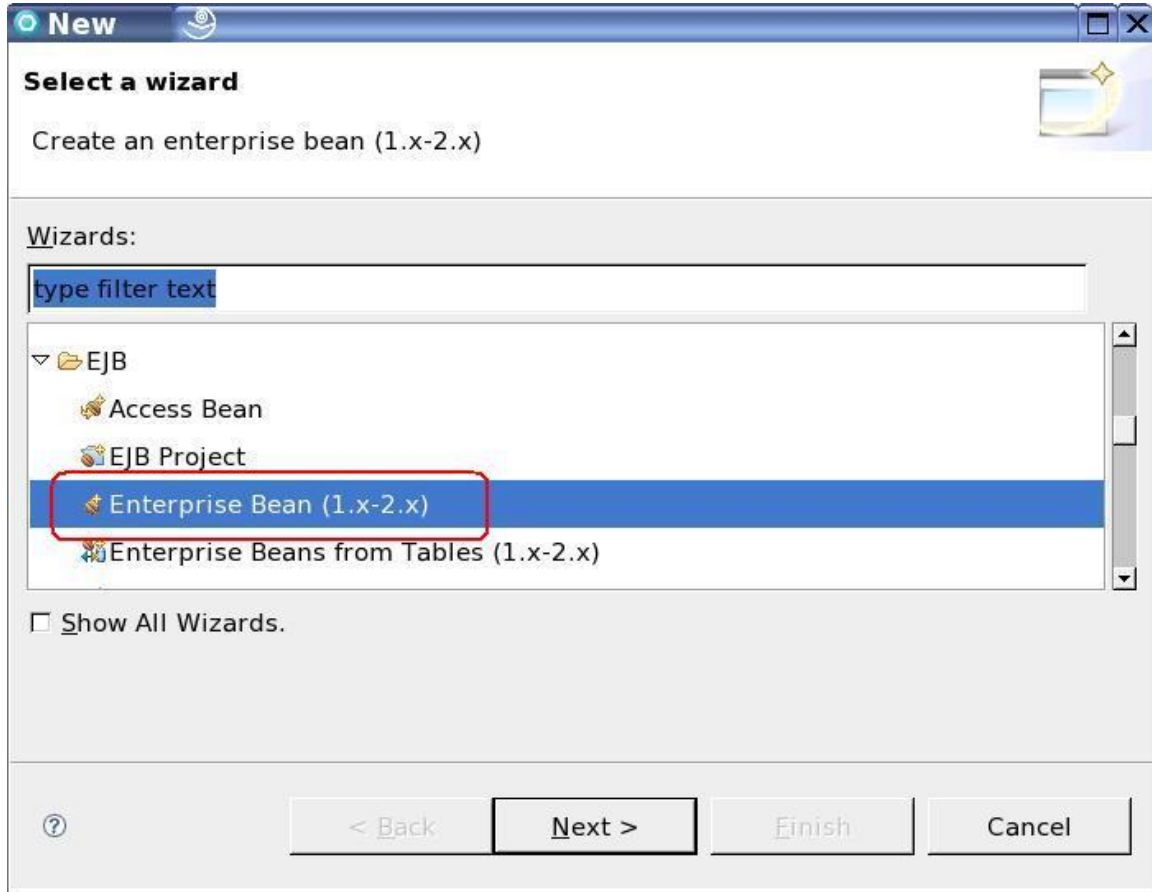
+ Create a New “Enterprise Bean (1.x - 2.x)” by right clicking on the EJB Project that you have created, in this case “SampleMDBEJB”.

There are 2 steps.

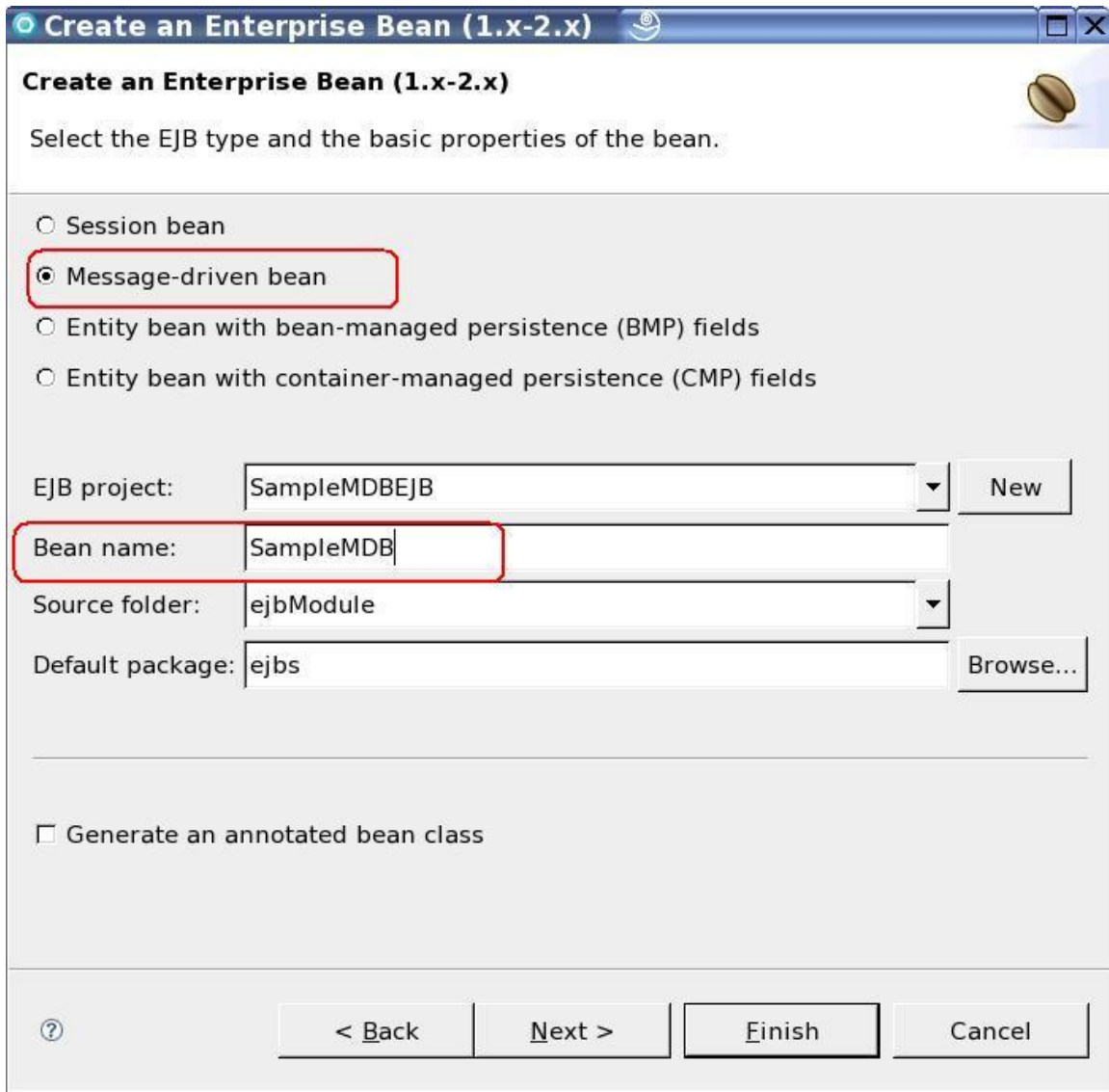
1: New > Other ... > Enterprise Bean (1.x - 2.x)



2: From the “New” window, select: Enterprise Bean (1.x - 2.x)



You will see the window “Create an Enterprise Bean (1.x-2.x)”.
Ensure that the radio button for “Message-driven bean” is selected.
Accept the value for: EJB Project => SampleMDBEJB
Enter the value for “Bean Name” => SampleMDB
Accept the other defaults.



Create an Enterprise Bean (1.x-2.x)

Select the EJB type and the basic properties of the bean.

Session bean

Message-driven bean

Entity bean with bean-managed persistence (BMP) fields

Entity bean with container-managed persistence (CMP) fields

EJB project: SampleMDBEJB New

Bean name: SampleMDB

Source folder: ejbModule Browse...

Default package: ejbs Browse...

Generate an annotated bean class

< Back Next > Finish Cancel

Click on Next.

Ensure that the radio button for “JMS Type” is selected (the default).
Accept the default for “Listener type” => Javax.jms.MessageListener

The screenshot shows a dialog box titled "Create an Enterprise Bean (1.x-2.x)" with a sub-tab "Message-Driven Bean Type". The dialog contains the instruction "Choose the type of messaging service to use for the message-driven bean." There are two radio button options: "JMS type" (selected and highlighted with a red box) and "Other type". Under "JMS type", the "Listener type" field is populated with "javax.jms.MessageListener". Under "Other type", the "Listener type" field is empty and has a "Browse..." button next to it. At the bottom of the dialog, there are four buttons: "< Back", "Next >" (highlighted with a black box), "Finish", and "Cancel".

Click on Next.

Ensure that the “Transaction Type” is “Container”
Add an Activation Configuration as follows.

Create an Enterprise Bean (1.x-2.x)

Create a JMS Message-Driven Bean

Select the transaction type, destination and the bean class necessary for this m

Transaction type: Container

Activation Configuration:

Name	Value
------	-------

Add...
Remove

Bean supertype:

Bean class: ejbs.SampleMDBBean Package... Class...

Message destination link: Browse...

< Back Next > Finish Cancel

Click the button “Add...”.

Go to the first column, labeled “Name”. Click on the first row then click on the down arrow on the right side of the frame. You will see 3 values. Select “destinationType”.

Go to the second column, labeled “Value”. Select “javax.jms.Queue”. Notice that neither the “Next” nor the “Finish” buttons are active. You will need to click on the “Add...” button again or inside the value frame for “Bean supertype”, in order for this GUI to acknowledge that you have finished with the addition of the Activation Specification (it seems like a usability bug for me). You do not need to enter anything else. Notice that the “Next” and “Finish” buttons are now active.

Create an Enterprise Bean (1.x-2.x)

Create a JMS Message-Driven Bean

Select the transaction type, destination and the bean class necessary for this m

Transaction type: Container

Activation Configuration:

Name	Value
destinationType	javax.jms.Queue

Buttons: Add... Remove

Bean supertype: << Click here to enable the "Finish" button

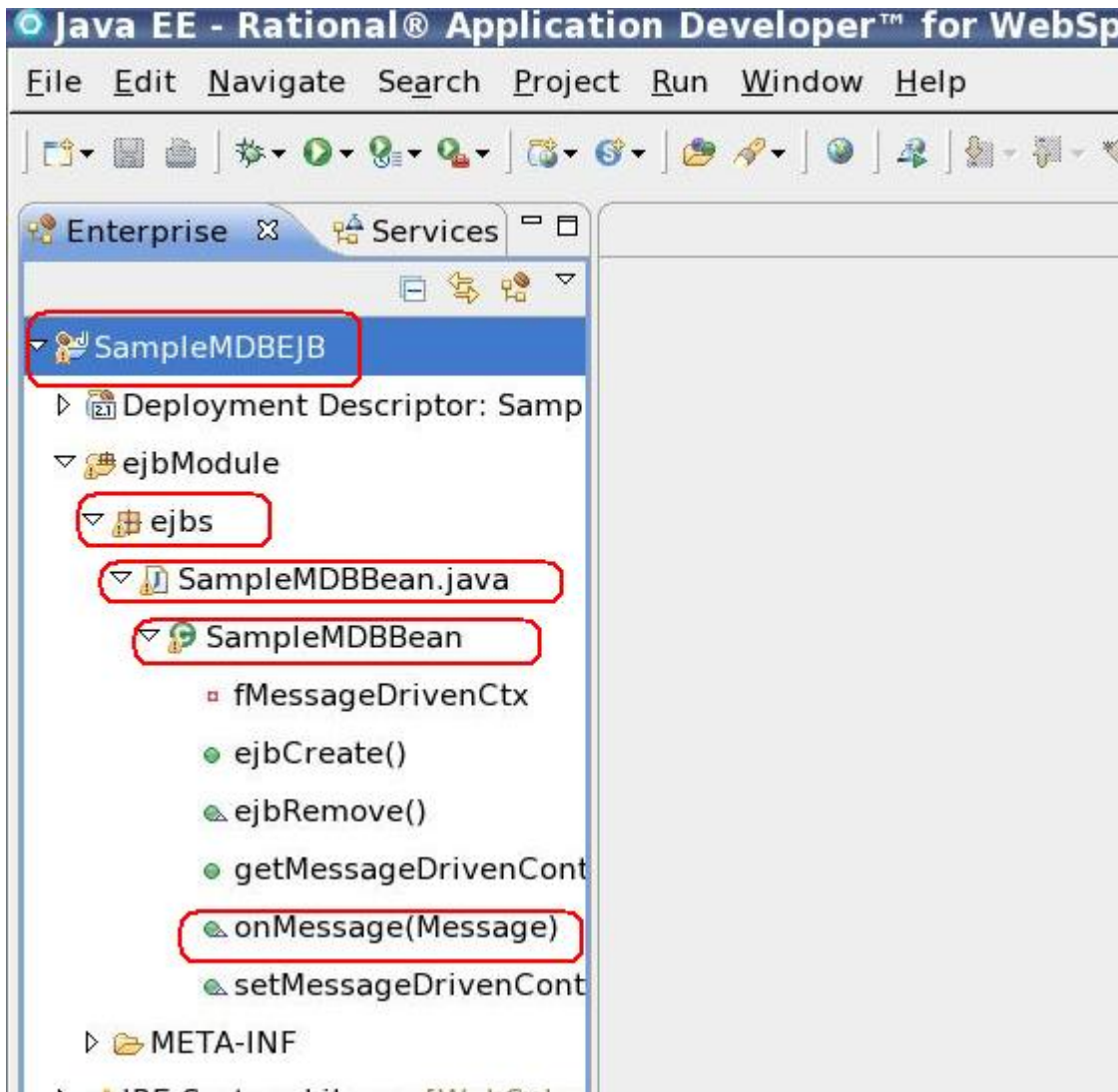
Bean class: ejbs.SampleMDBBean Package... Class...

Message destination link: Browse...

Buttons: < Back Next > Finish Cancel

Click on Finish.

+ After creating a new message driven bean, go to the Enterprise Workspace (Project Explorer) and expand:
SampleMDBEJB > ejbModule > ejbs > SampleMDBBean.java > SampleMDBBean



The method that handles the messages is called: `onMessage(Message)`

Now let's edit the `onMessage` method of the "SampleMDBBean". From the Project Explorer (again, the left navigation panel), select the method "`onMessage(Message)`" and double click. The default action is to open the Java editor.

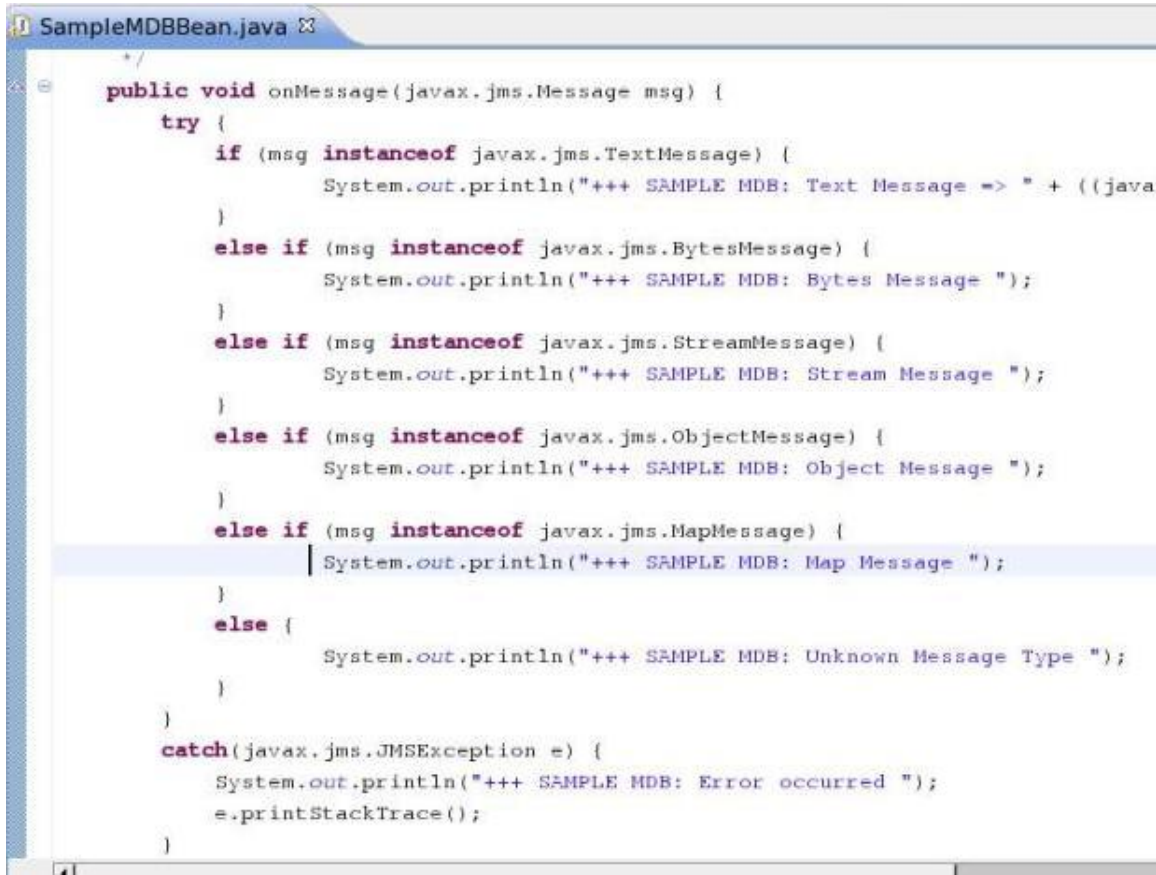
You will see that the source code for the "SampleMDBBean" is shown in the editor, and that the cursor is already positioned in the "`onMessage`" method, as shown below:



- Insert the following source code to display the type of contents (payload) and an “eye catcher string” (+++ SAMPLE MDB) which can let you find quickly the output of the MDB in the SystemOut.log file. In addition, for a Text Message, the actual text is displayed.

```
public void onMessage(javax.jms.Message msg) {
    try {
        if (msg instanceof javax.jms.TextMessage) {
            System.out.println("+++ SAMPLE MDB: Text Message
=> " + ((javax.jms.TextMessage)msg).getText());
        }
        else if (msg instanceof javax.jms.BytesMessage) {
            System.out.println("+++ SAMPLE MDB: Bytes
Message ");
        }
        else if (msg instanceof javax.jms.StreamMessage) {
            System.out.println("+++ SAMPLE MDB: Stream
Message ");
        }
        else if (msg instanceof javax.jms.ObjectMessage) {
            System.out.println("+++ SAMPLE MDB: Object
Message ");
        }
        else if (msg instanceof javax.jms.MapMessage) {
            System.out.println("+++ SAMPLE MDB: Map Message
");
        }
        else {
            System.out.println("+++ SAMPLE MDB: Unknown
Message Type ");
        }
    }
    catch(javax.jms.JMSEException e) {
        System.out.println("+++ SAMPLE MDB: Error occurred ");
        e.printStackTrace();
    }
}
```

- The onMessage method should look like this:

A screenshot of an IDE window titled 'SampleMDBBean.java'. The code shows a public void onMessage method that takes a javax.jms.Message object. It uses a try-catch block to handle different message types. The try block contains several if-else-if statements: if (msg instanceof javax.jms.TextMessage), if (msg instanceof javax.jms.BytesMessage), if (msg instanceof javax.jms.StreamMessage), if (msg instanceof javax.jms.ObjectMessage), and if (msg instanceof javax.jms.MapMessage). Each if statement prints a message to the console. The else block prints 'Unknown Message Type'. The catch block catches javax.jms.JMSEException and prints an error message along with the stack trace.

```
public void onMessage(javax.jms.Message msg) {
    try {
        if (msg instanceof javax.jms.TextMessage) {
            System.out.println("+++ SAMPLE MDB: Text Message => " + ((java.
        )
        else if (msg instanceof javax.jms.BytesMessage) {
            System.out.println("+++ SAMPLE MDB: Bytes Message ");
        }
        else if (msg instanceof javax.jms.StreamMessage) {
            System.out.println("+++ SAMPLE MDB: Stream Message ");
        }
        else if (msg instanceof javax.jms.ObjectMessage) {
            System.out.println("+++ SAMPLE MDB: Object Message ");
        }
        else if (msg instanceof javax.jms.MapMessage) {
            System.out.println("+++ SAMPLE MDB: Map Message ");
        }
        else {
            System.out.println("+++ SAMPLE MDB: Unknown Message Type ");
        }
    }
    catch(javax.jms.JMSEException e) {
        System.out.println("+++ SAMPLE MDB: Error occurred ");
        e.printStackTrace();
    }
}
```

- Now save the changes, by pressing: Ctrl + S

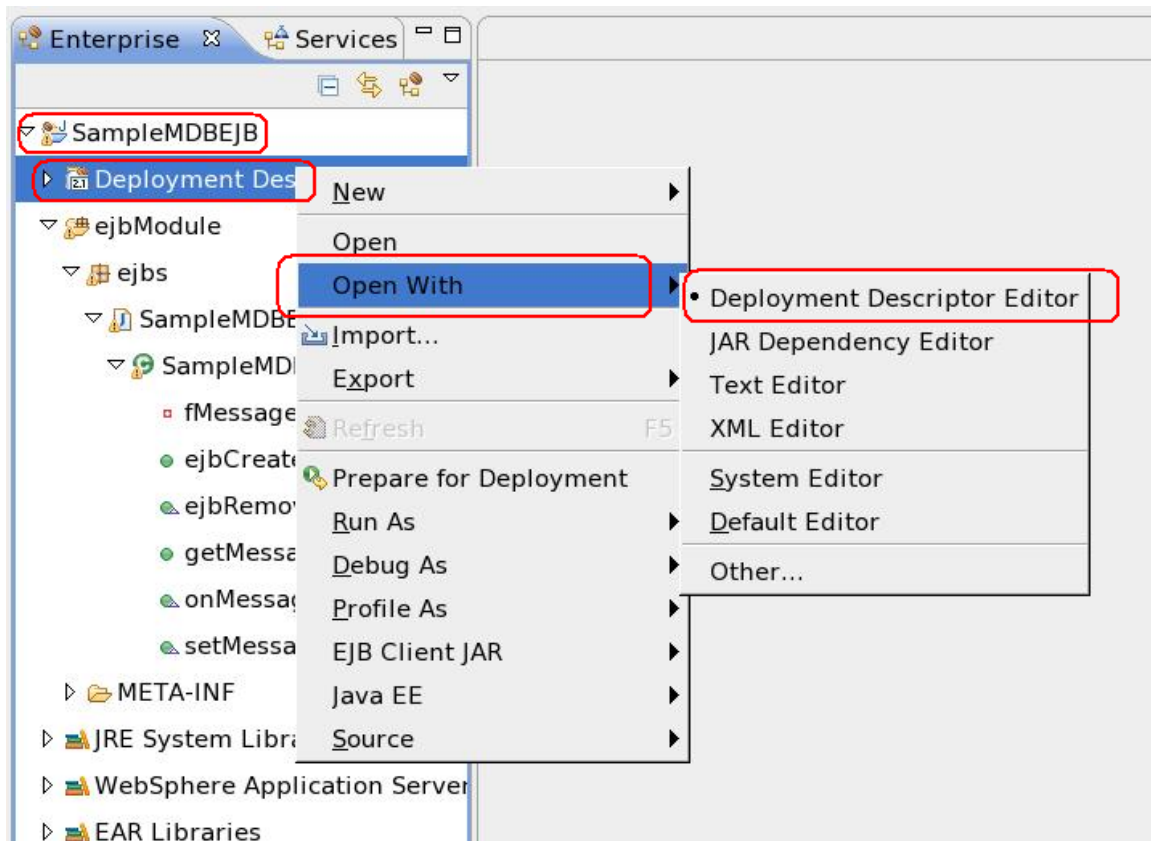
Note:

In order to facilitate the “silent” deployment of the EJB from RAD to WebSphere Application Server it is necessary to edit the Deployment Descriptor and indicate an existing Listener Port from the WebSphere Application Server: SampleMDBQueueLP

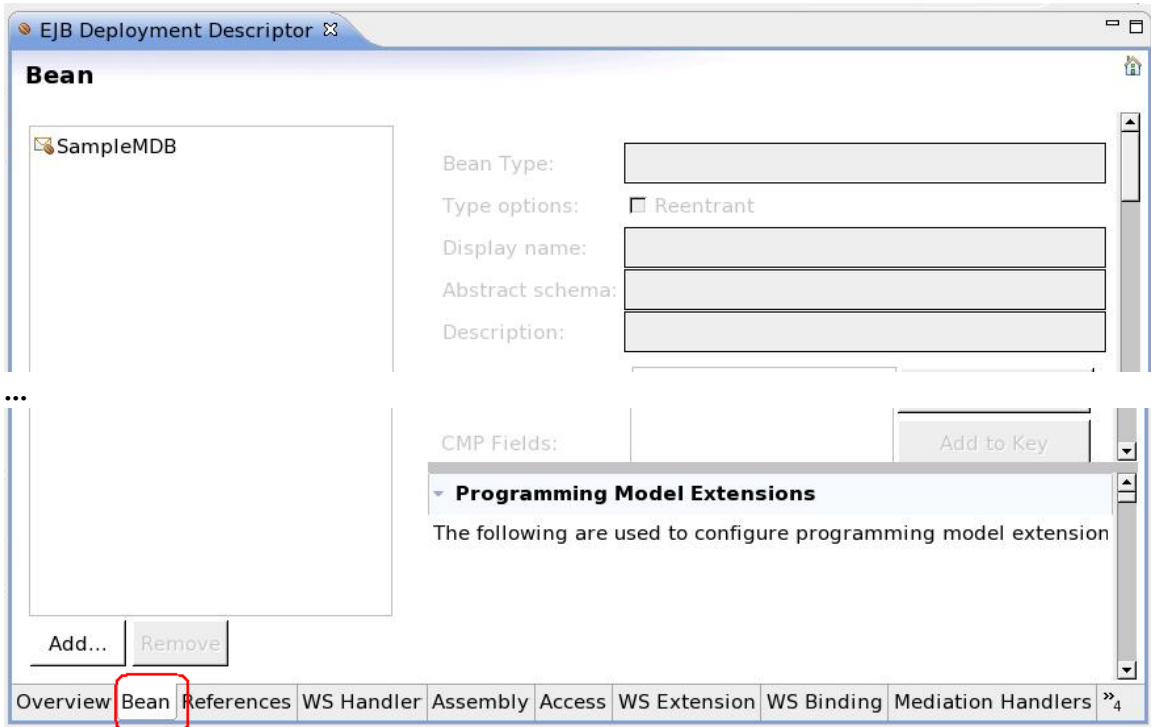
Otherwise, when using RAD to deploy the MDB, it needs to know which WebSphere Application Server Listener Port to use and if there are none, then WebSphere Application Server will not fully deploy the project, giving a warning.

- Edit the Deployment Descriptor.

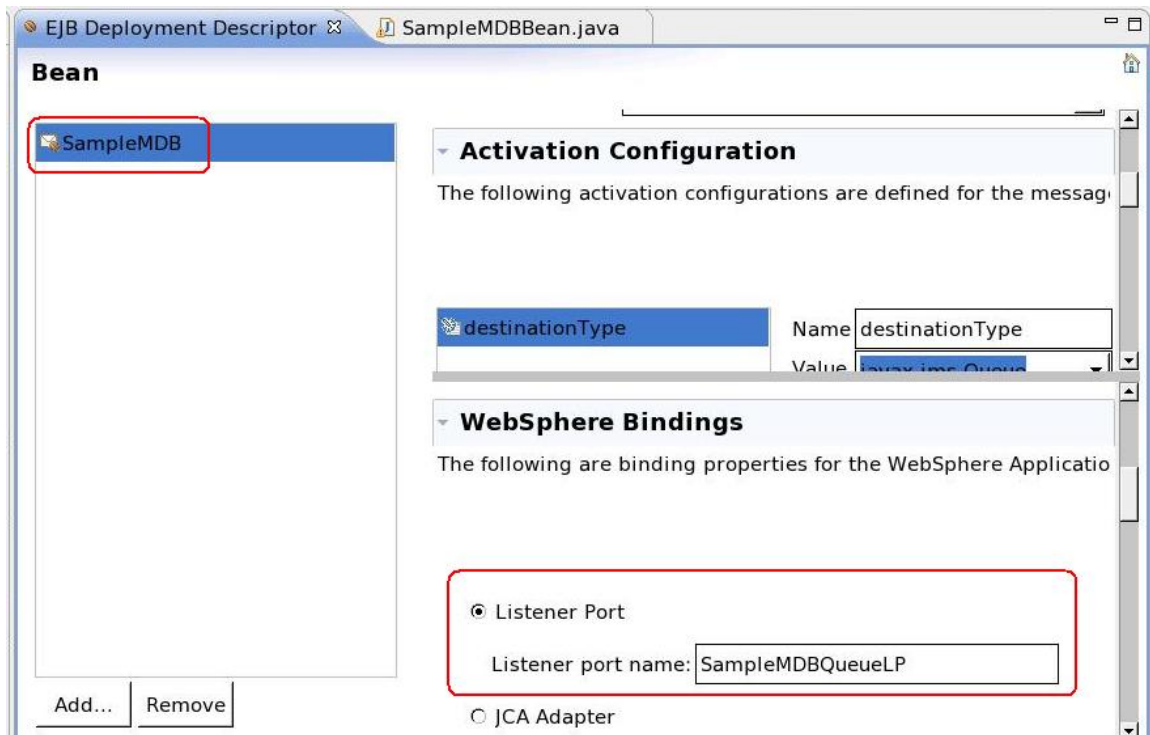
Expand the project “SampleMDBEJB”, then right click on the “Deployment Descriptor”, then select “Open with...” > “Deployment Descriptor Editor”.



- You will see the EJB Deployment Descriptor editor. Click on the tab "Bean".



From the Bean tab select the message driven bean SampleMDB from the left panel. Then on the bottom right panel, scroll down to the section: “WebSphere Bindings” and give the Listener Port Name: SampleMDBQueueLP



Notice that the title for the major tab “*EJB Deployment Descriptor” has a leading asterisk. This indicates that there are changes that have not been saved yet.

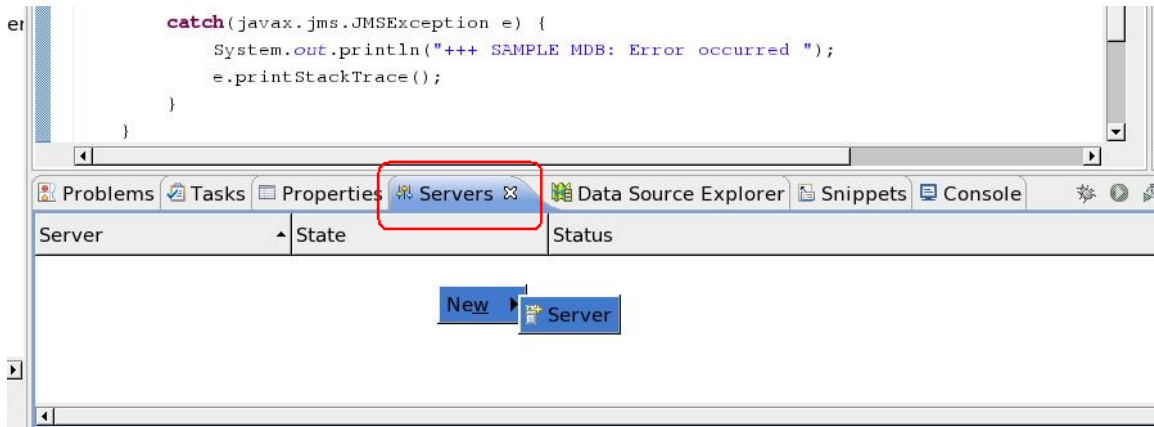
Save the changes. Either:
- From main menu: File > Save
- Or use the keys: Ctrl + S

Notice that the leading asterisk is removed, and the title changes back to: “EJB Deployment Descriptor”

++++
++ Chapter 2: RAD 7.5 testing of the MDB using WebSphere Application Server 7
++++

In RAD 7.5 and with full WebSphere Application Server 7 installed in the same machine, it is necessary to register the WebSphere Application Server into RAD.

In the section for the Console (bottom pane), select the Servers tab, move the mouse to lower portion, right click, then New > Server.



In the “New Server” wizard, enter:

Server’s host name: localhost
Select the server type: WebSphere Application Server v7.0
Server runtime environment: WebSphere Application Server v7.0

Click Next.

New Server

Define a New Server

Choose the type of server to create

Server's host name: localhost

[Download additional server adapters](#)

Select the server type:

type filter text

WebSphere Application Server v7.0

Runs J2EE projects on the WebSphere Application Server v7.0.

Server name: WebSphere Application Server v7.0 at localhost

Server runtime environment: WebSphere Application Server v7.0 [Add...](#)

[Configure runtime environments...](#)

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

In the “WebSphere Server Settings” page, ensure that the proper values for your installation are used. The following are the defaults for WebSphere Application Server 7:

WebSphere profile name: AppSrv01

WebSphere server name: server1

New Server

WebSphere Server Settings

Input settings for connecting to an existing WebSphere Application Server.

WebSphere profile name: [Configure profiles...](#)

Server connection types and administrative ports

Automatically determine connection settings

Manually provide connection settings

Connection Type	Port	Default port	Description
<input checked="" type="checkbox"/> IPC	9633	9633	Recommended for local server
<input checked="" type="checkbox"/> RMI	2809	2809	Designed to improve communi
<input checked="" type="checkbox"/> SOAP	8880	8880	Designed to be more firewall c

Run server with resources within the workspace

Security is enabled on this server

Current active authentication settings:

User ID:

Password:

WebSphere server name:

[Test Connection](#)

Click on Next

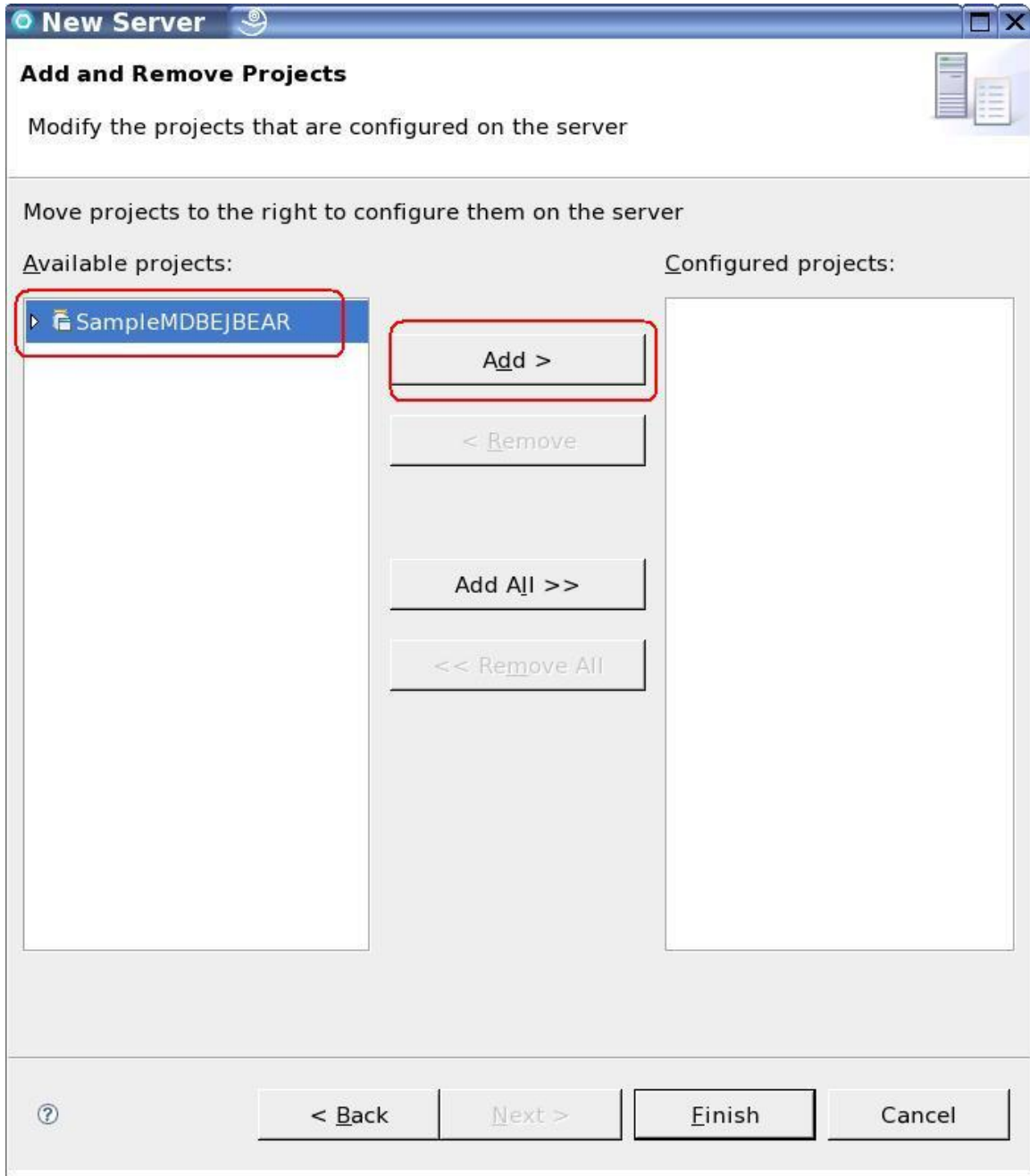
- Click on "Test Connection".



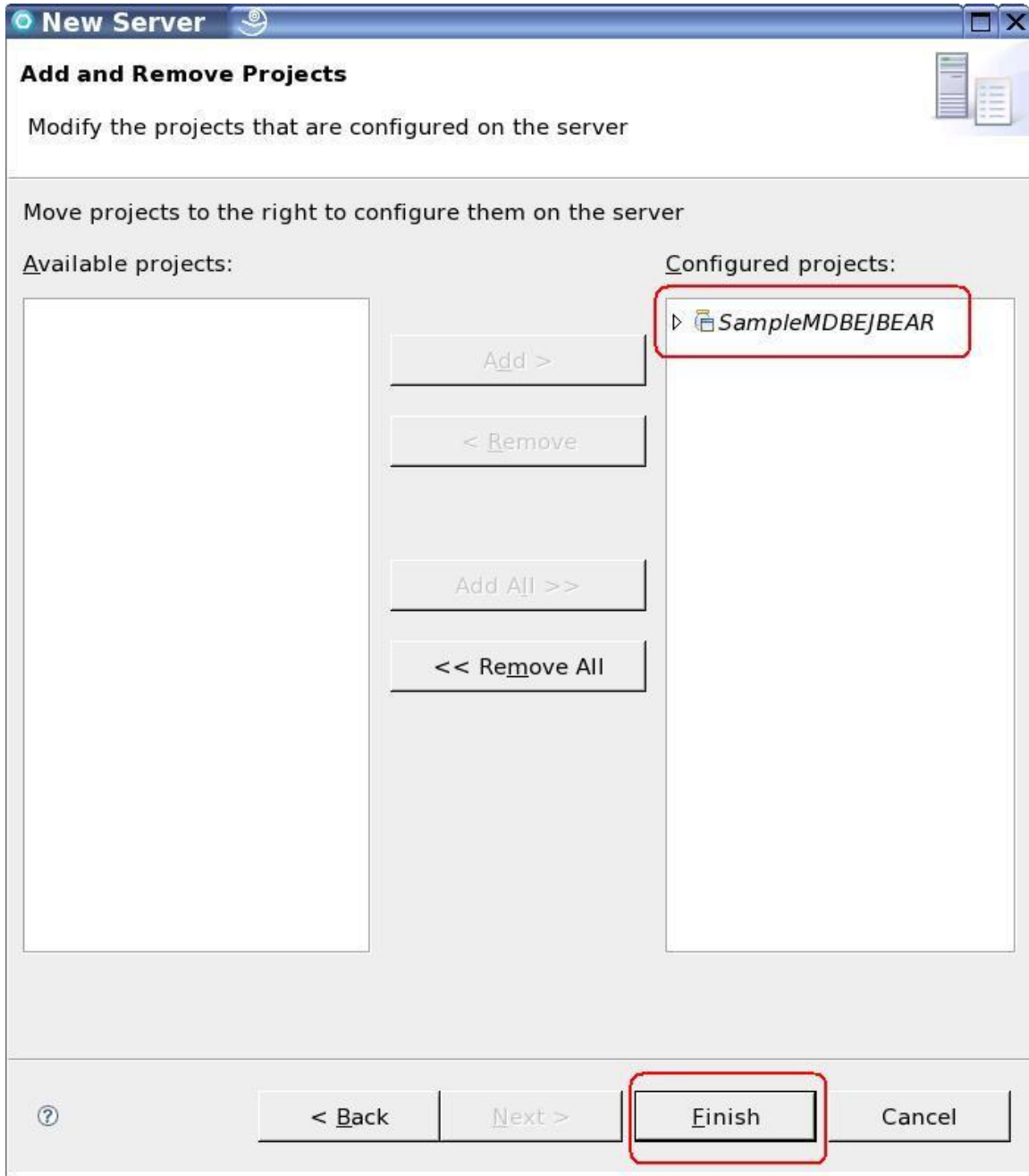
Ensure that it runs fine. If so, click Finish.

If there are any problems, STOP! And fix them now before proceeding.

Now click on Next. You will see the page “Add and Remove Projects”. Select “SampleMDBEJBEAR” from the left side (Available projects)



Click Add. You will see that this EAR will be moved to the right side (Configured projects)



Click on Finish

This action will deploy the EAR file into the WebSphere Application Server.

Notice that the Console tab (at the bottom of RAD) will show the progress of the publishing of the EJB.

Note:

You MUST create the SampleMDBQueueLP Listener Port in WebSphere Application Server and specify it in the project descriptor. This Listener Port must be up and running.

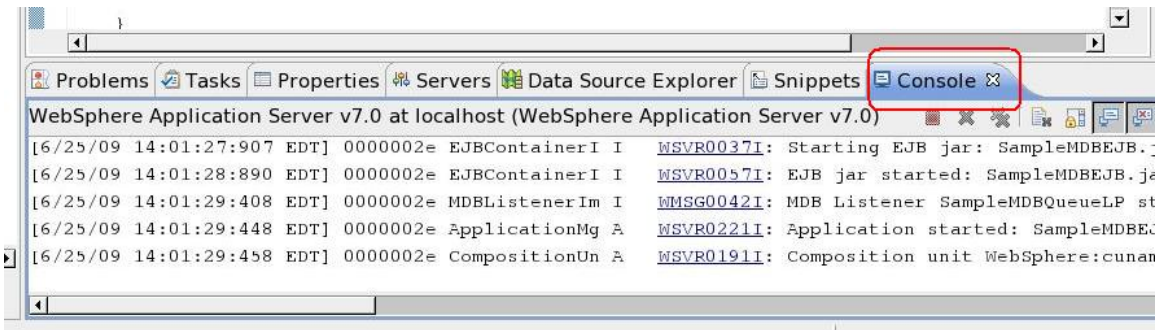
Otherwise, when using RAD to deploy the MDB, it needs to know which LP to use and if there are none, then WebSphere Application Server will not fully deploy the project, giving an error:

Application Failed to Start. SampleMDBEJBEAR
Java.lang.ClassNotFoundException: com.ibm.ejs.jms.listener.MDBException

Now let's test the MDB.

We will need:

- A UNIX command prompt window for entering an MQ command that places one message into the queue that is monitored by the Listener Port of the WebSphere Application Server and which passes the message to the MDB.
- From RAD, click on the "Console" tab to watch the most recent entries appended to the SystemOut.log:



A: From the UNIX Window:

Login as user "mqm" (or another user who has access to MQ).

Enter the command to put a message into the queue Q_MDB from the queue manager QM_MDB.

```
$ amqsput Q_MDB QM_MDB
```

Enter a text that you could easily identify from the SystemOut.log, such as:

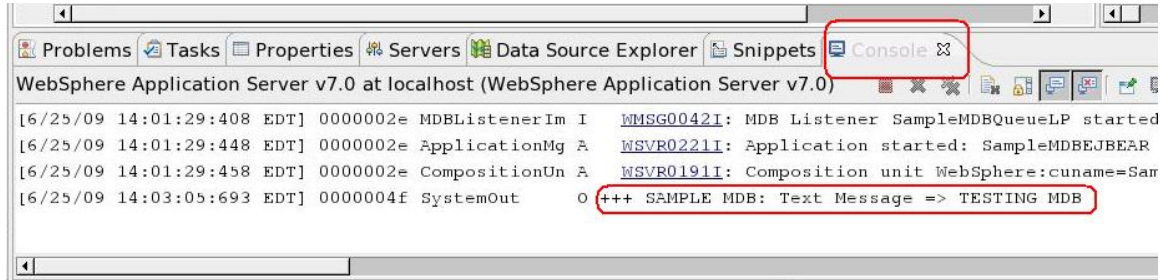
```
TESTING MDB
```

Example snapshot:

```
mqm@veracruz: /home/mqm
(448) amqsput Q_MDB QM_MDB
Sample AMQSPUT0 start
target queue is Q_MDB
TESTING MDB
```

Press Enter to end amqsput.

B: From the Console in RAD, see the output:



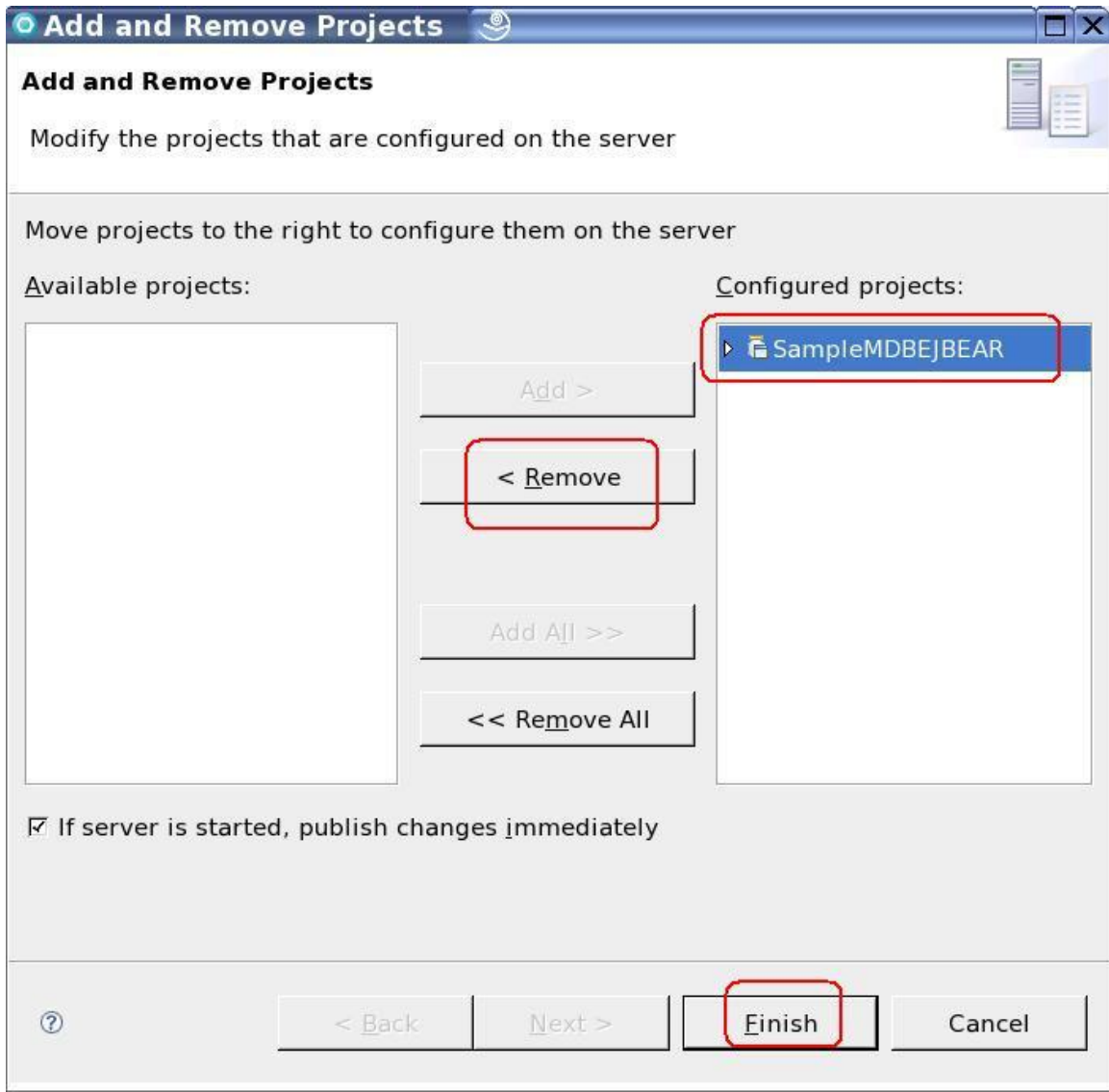
Notice the text:

```
+++ SAMPLE MDB: Text Message => TESTING MDB
```

Now that we were able to verify that the MDB works fine (reads a message from the MQ queue and process the contents of the message), we can proceed to stop the testing of the MDB from RAD, export the EJB project and deploy the EJB directly to WebSphere Application Server, without further need to use RAD.

Proceed to remove the SampleMDBEJBEBEAR project by accessing the “Add and Remove Projects” from the WebSphere Application Server in RAD, as mentioned before.

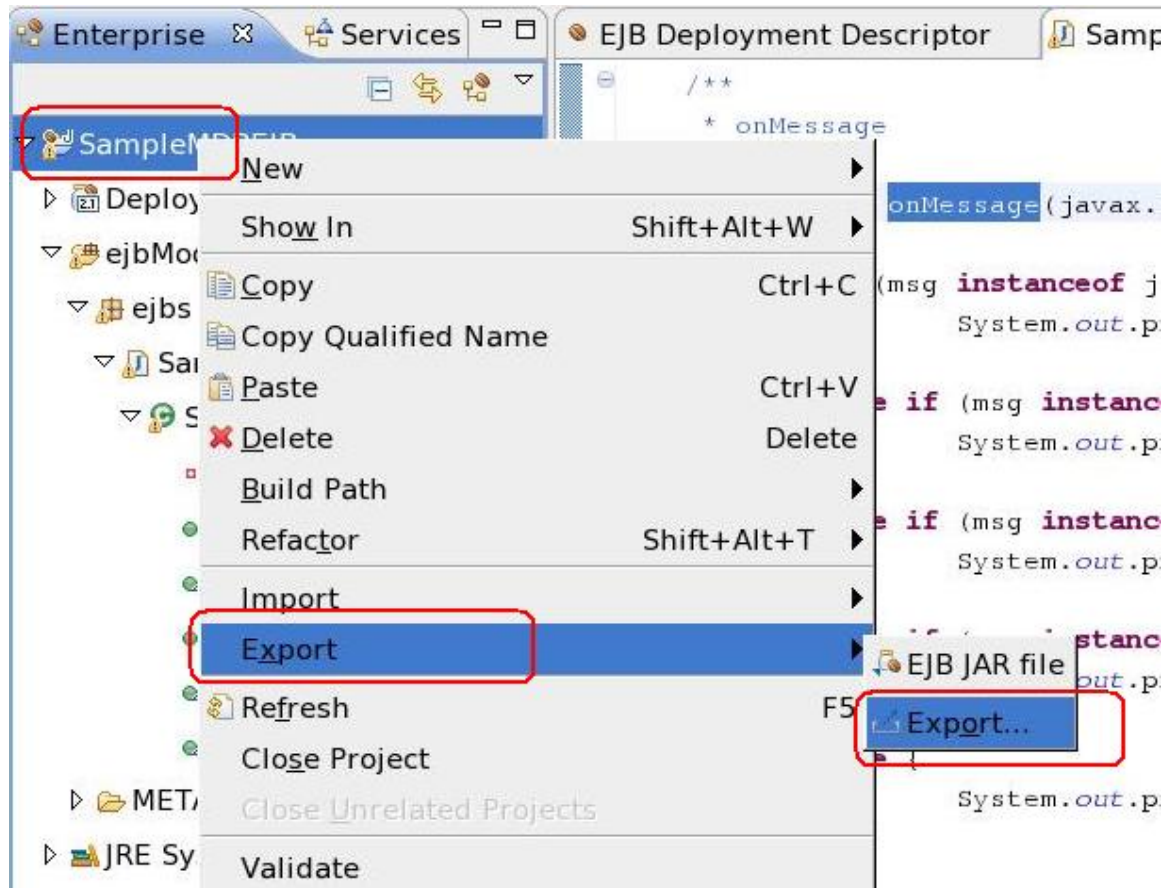
Then click on “< Remove” and then on “Finish”



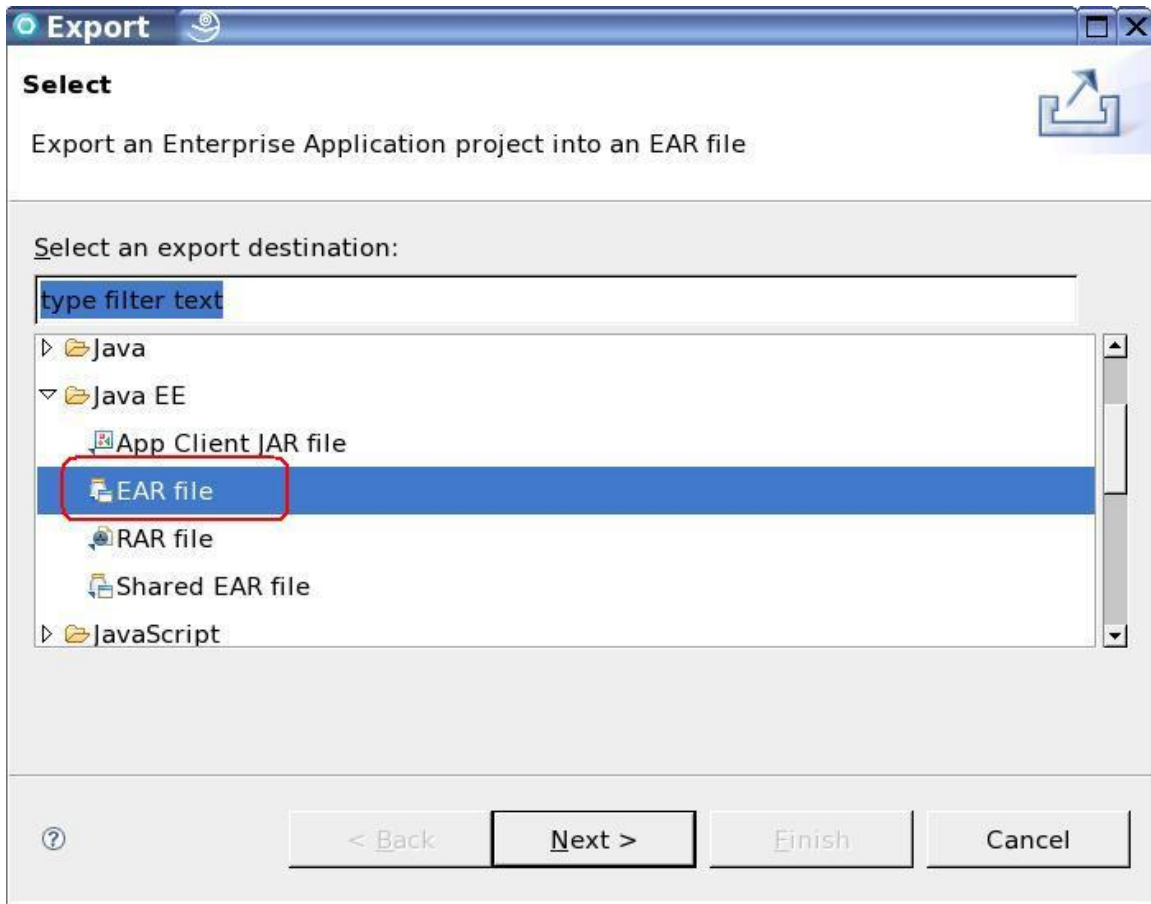
You can see in the Console tab (which is monitoring the SystemOut.log file for server1) the details on the stopping and removal of this project.

Let's export the EJB in an EAR file for direct deployment (outside RAD) into WebSphere Application Server.

From the "Enterprise" navigation panel, select the top level "SampleMDBEJB" then Export > Export ...



You will see the following dialog.
Select “EAR file”



Click on Next.

In the Destination field, specify the location where you want the exported EAR file to be stored.

Be sure to uncheck the option for “Target Runtime”. By not specifying a particular version of the WebSphere Application Server, this MDB can be used with Version 6 or Version 7. If you specify a version, then, you may be limiting your options.

Export

EAR Export

Export Enterprise Application project to the local file system.

EAR project: SampleMDBEJBEAR

Destination: /downloads/mdb/SampleMDBEJB.ear **Browse...**

Target Runtime

Optimize for a specific server runtime

was.base.v7

Export source files

Overwrite existing file

? < Back Next > Finish Cancel

Click Finish.

The EAR file will be ready to be deployed into WebSphere Application Server.

For an example of such deployment, see the mentioned techdoc:

Using WebSphere MQ V7 as JMS Provider for WebSphere Application Server V7

Chapter 3: WebSphere Application Server V7 deployment and testing of MDB

This is the end of the techdoc.

+++ end +++