

## IBM® WebSphere Application Server V7 Feature Pack for OSGi Applications and JPA 2.0 – Lab exercise

# Modify and re-deploy a single bundle from an application to update it without re-deploying the application.

What this exercise is about .....	1
Lab requirements .....	1
What you should be able to do .....	2
Introduction .....	2
Exercise instructions .....	2
Part 0: Import the solution from the Information Center hello-world application.....	3
Part 1: Create a new version of your service bundle .....	8
What you did in this exercise .....	13

## What this exercise is about

There is an exercise in the Information Center for the feature pack in which one creates a working hello world sample using eclipse or Rational® Application Developer. The purpose of that exercise is to teach the basic elements of OSGi and blueprint.

That exercise is a prerequisite to this one and you will need the solution to that exercise in order to perform this lab. In the event that you have completed that exercise in the past and did not save your result, you can import the solution supplied with this lab rather than repeat it.

The purpose of this lab is to go the next step in order to build on that understanding. In this lab you will take the sample completed from the Information Center exercise and make a simple change which will cause you to create a new version of one of the bundles and deploy it in order to propagate the change.

This lab is provided **AS-IS**, with no formal IBM support.

## Lab requirements

List of system and software required to complete the lab. Although other versions may function properly, the specified versions were used to create and test this sample application. The prerequisite activity for this lab is to complete the Information Center hello world exercise which has the same requirements so once you have completed that exercise you will have everything you need to complete this one.

- WebSphere Application Server Version 7.0
- WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0
- Java (J2SE) V1.6 – included with WebSphere Application Server V7
- Eclipse Version 3.6

- the IBM Rational Development Tools for OSGi Applications, Version 5
- Apache Derby database – included with WebSphere Application Server V7
- Solution to the exercise in the Feature Pack for OSGi Applications and Java Persistence 2.0 Information Center in section **Developing and deploying an OSGi application**. If you do not have the solution to that exercise you may import the one supplied with this lab.

## What you should be able to do

At the end of this lab you should be able to:

- Read and understand the imports and exports in the MANIFEST.MF files
- Read and understand the deployment and use of services in the blueprint.xml file
- Create and deploy a new version of a bundle and change the application configuration to use it rather than the version contained in the original EBA file when the application was deployed

---

## Introduction

OSGi overcomes a lot of the problems with developing, deploying and maintaining JEE applications. The OSGi Bundle is the unit of code deployment with each bundle having the capability to export and import packages while having other packages isolated from the environment at large. In this lab you will get a closer look at the building blocks for OSGi metadata in a very simple application. The metadata used in this lab is by no means exhaustive but that is really the point, to give you a simple place to start.

Useful links:

- Apache Aries Project:  
<http://incubator.apache.org/aries/>
- The OSGi Alliance:  
[www-01.ibm.com/software/webservers/appserv/was/featurepacks/](http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/)

---

## Exercise instructions

If you completed the prerequisite exercise in the Information Center you may go directly to Step 1. You are highly encouraged to go through that exercise and read the information provided regarding each artifact as you produce it. The background you get in that exercise combined with this one will better equip you to develop your own applications or to debug problems with OSGi applications.

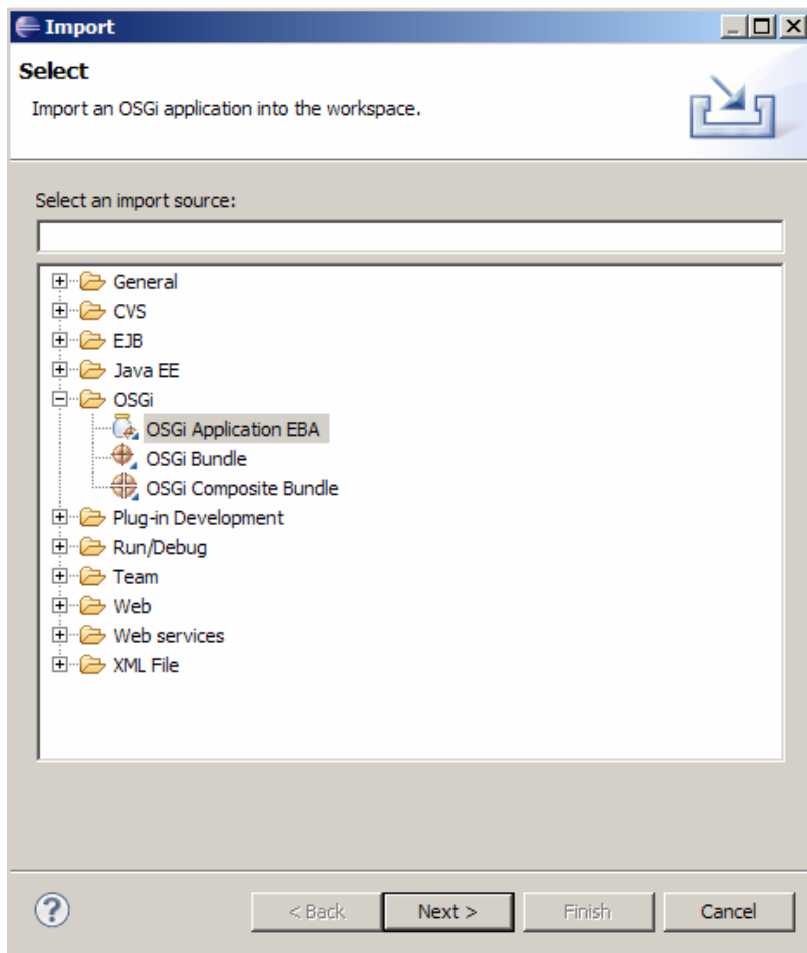
If you do not have the resultant project from the hello world exercise **Step 0** instructs you on how to import the solution to that exercise which is furnished by this lab. This lab was created on a Microsoft Windows system so instructions on importing and exporting have a drive specification. If you are on another operating system adjust your locations appropriately.

## Part 0: Import the solution from the Information Center hello-world application

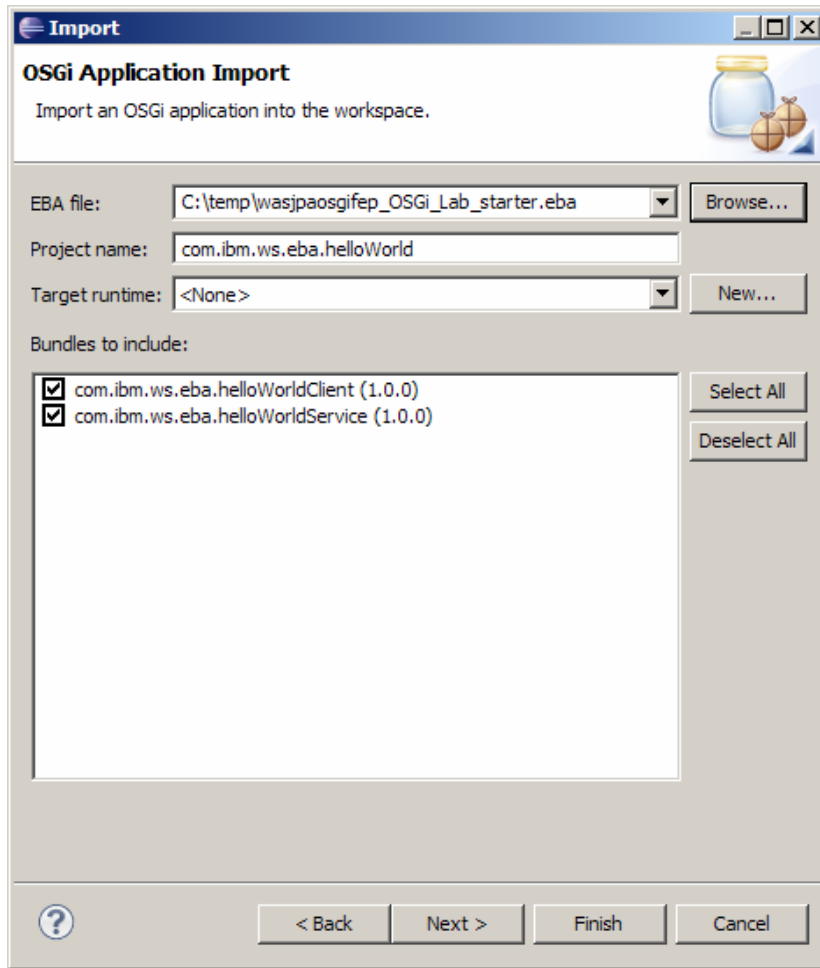
The solution to the hello world example provided in the feature pack's information center is supplied with this lab for your convenience should you not have saved your original work. The following steps will get you to where you were when you finished that example.

Once you have installed Eclipse 3.6 or later and have installed the IBM Rational Development Tools for OSGi Applications, Version 5 installed you will be able to import an EBA project. The file attached to this lab you need to download to your system and import is **wasjpaosgifep\_OSGi\_Lab\_starter.eba**. Assuming that you have downloaded it to your C:\temp directory follow the following instructions.

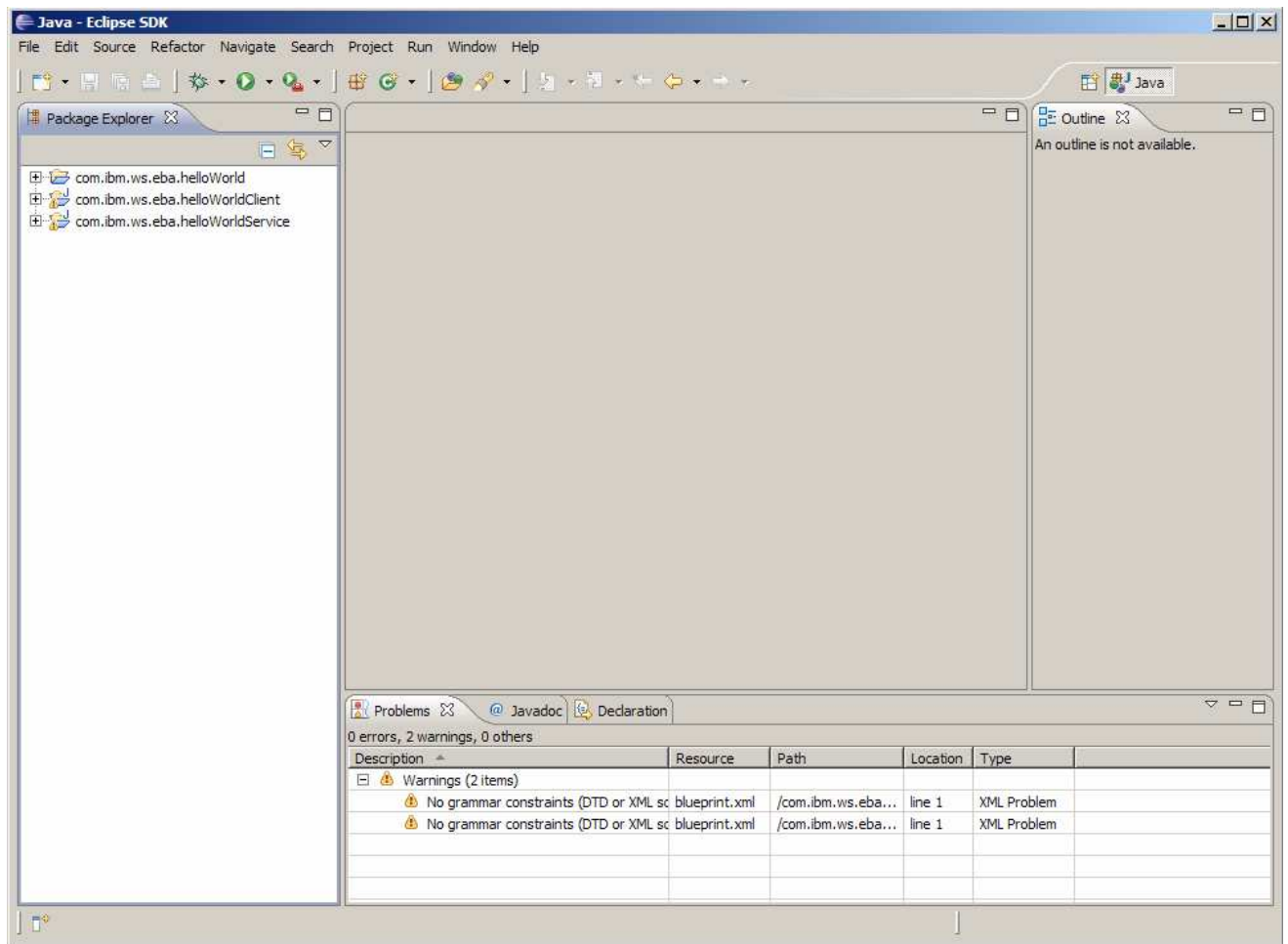
- \_\_\_ 1. Import the project into Eclipse
  - \_\_\_ a. Select File > Import > OSGi > OSGi Application EBA .



- \_\_\_ b. On the resulting screen either enter the path to the file or navigate to it using the Browse button. Accept all of the defaults on this screen. Click **Finish**.

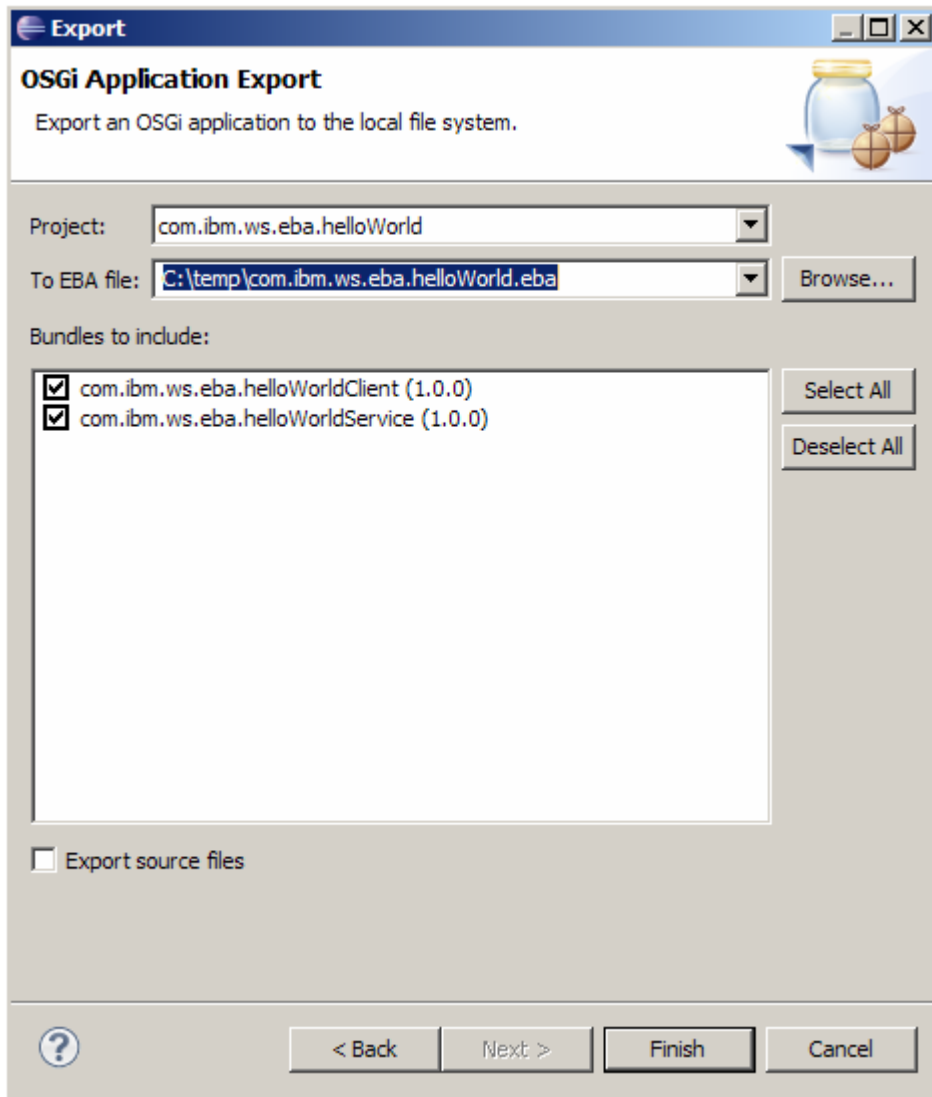


As a result you will have the three projects created during the Information Center exercise.



Note the two warnings about "No grammar constraints". You may ignore or delete them.

- \_\_\_ 2. Export the EBA file
  - \_\_\_ a. Select the `com.ibm.ws.eba.helloWorld` project.
  - \_\_\_ b. Right click and select **Export**
  - \_\_\_ c. On the Select panel, find and select OSGi → OSGi Application EBA, then click next.
  - \_\_\_ d. On the OSGi Application Export panel, either enter by hand or use the **Browse** button to select a location to write the exported EBA file. For this example the directory being used is `C:\temp`. If you use a different directory remember it and substitute it for `C:\temp` in later steps.



\_\_\_ e. Click **Finish**

\_\_\_ 3. Import the EBA file as a Business-level application asset. (OSGi applications are handled as Business-level Applications in WebSphere Application Server)

\_\_\_ a. Open an administrative console session

\_\_\_ b. From the left navigation panel select **Applications** → **Application Types** → **Assets**

\_\_\_ c. On the **Assets** panel, locate the Asset collection and click the **Import** button.

\_\_\_ d. On the **Upload asset** panel, in the **Local file system** input field, either type in the path or use the Browse button to navigate to the *com.ibm.ws.eba.helloWorld.eba* file you exported in the previous step. Click **Next**.

\_\_\_ e. On the Select options for importing an asset panel you do not need to change anything but notice the text “EBA, version1.0” in the middle of the screen. That means the application server is recognizing your EBA file as an OSGi application. Click **Next**.

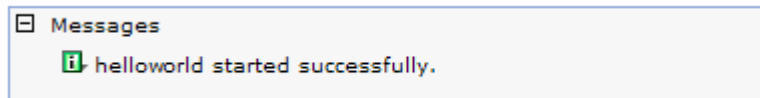
- \_\_\_ f. Click **Finish**.
- \_\_\_ g. Click **Save**

\_\_\_ 4. Install your application in WebSphere Application Server

- \_\_\_ a. From the left navigation panel select **Applications → New Application**
- \_\_\_ b. From the New Application panel select **New Business Level Application**
- \_\_\_ c. Enter "helloworld" in the **Name** field and click the **Apply** button.
- \_\_\_ d. On the resulting panel locate the Deployed assets collection. It may be empty. Click  
Add → Add Asset
- \_\_\_ e. On the resulting panel select *com.ibm.ws.eba.helloWorld.eba* with the radio button and click **Continue**.
- \_\_\_ f. On the resulting two panels click **Next** and then click **Finish** on the third.
- \_\_\_ g. Click **Save**.

\_\_\_ 5. Start the application

- \_\_\_ a. From the left navigation panel select **Applications → Application Types → Business-level applications**.
- \_\_\_ b. Locate your newly installed application, select the check box beside it and click **Start**. You should see:



\_\_\_ 6. View the output.

- \_\_\_ a. Open a command line window and navigate to your logs. This example is on Microsoft Windows.
- \_\_\_ b. Navigate to `<WAS_HOME>\profiles\<yourprofilename>\logs\<yoursevername>`
- \_\_\_ c. Depending upon your operating system **type** or **cat** your **SystemOut.log** file. Your output should have something like this near the end.

```
[5/25/10 14:21:06:687 EDT] 00000014 SystemOut      0 Client: Start...
[5/25/10 14:21:06:687 EDT] 00000014 SystemOut      0 OSGi Service: Hello World!
[5/25/10 14:21:06:687 EDT] 00000014 SystemOut      0 Client: End...
```

Now you are where the Information Center article ended and ready to start the new lab.

---

## Part 1: Create a new version of your service bundle

Now you have a working OSGi application and it is time to make a change. In this exercise you will make a change to your service implementation. You will then create a new version of the service bundle and deploy it into your WebSphere Application Server instance's local repository. Then you will alter your application to use the new version and restart it to see the result.

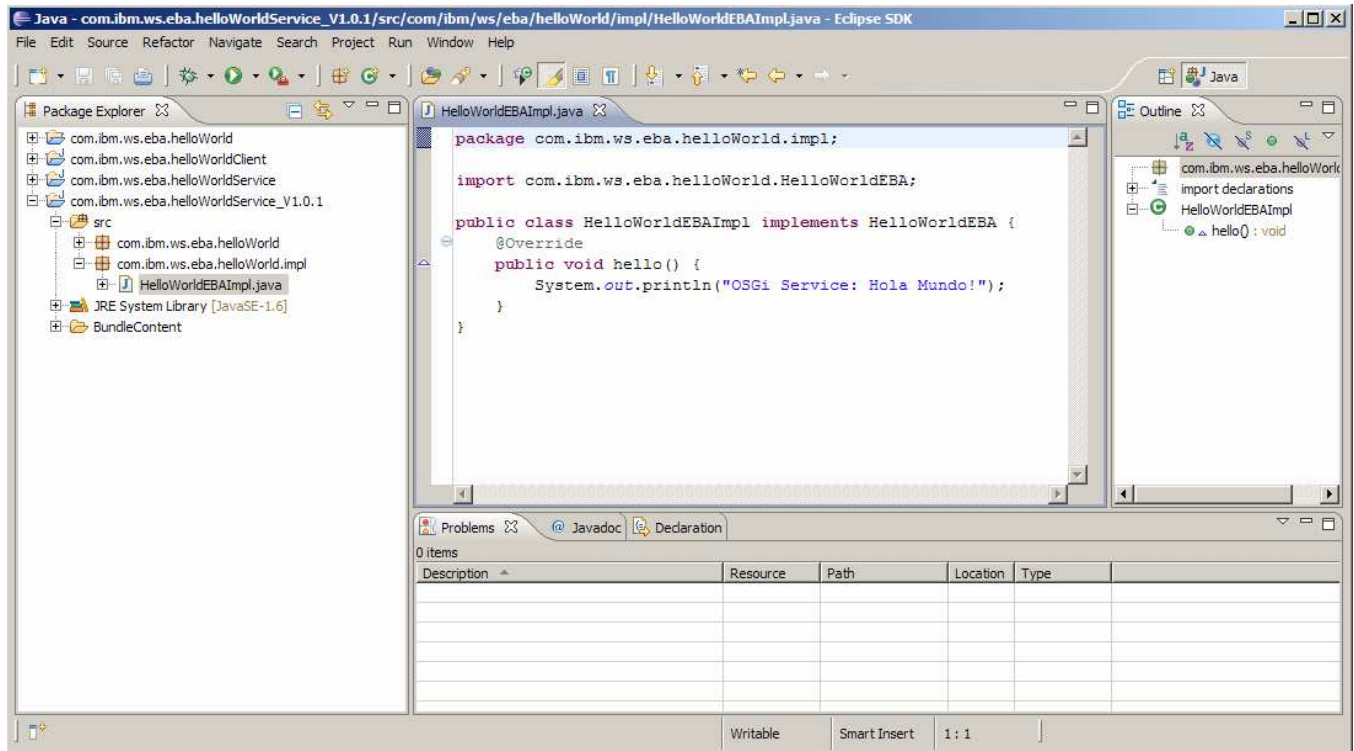
- \_\_\_ 1. Create a copy of your service project with a new version
  - \_\_\_ a. In Eclipse, select the *com.ibm.ws.eba.helloWorldService* project
  - \_\_\_ b. Right click and select **Copy**.
  - \_\_\_ c. Put your pointer in the white space of the Package Explorer, right click and select **Paste**.
  - \_\_\_ d. In the **Copy Project** window change the name of the project from:  

Copy of com.ibm.ws.eba.helloWorldService

to:

com.ibm.ws.eba.helloWorldService\_V1.0.1
  - \_\_\_ e. Click **OK**.
  - \_\_\_ f. You may get another "No grammar constraints" warning, it can be ignored.
- \_\_\_ 2. Make a change that will be noticeable when the application runs.
  - \_\_\_ a. Open the *com.ibm.ws.eba.helloWorldService\_V1.0.1* project by clicking the plus sign beside it
  - \_\_\_ b. Open src directory by clicking the plus sign beside it
  - \_\_\_ c. Open the *com.ibm.ws.eba.helloWorld.impl* package by clicking the plus beside it
  - \_\_\_ d. Double click **HelloWorldEBAImpl.java** to open the editor
  - \_\_\_ e. In the editor change "Hello World" to "Hola Mundo".





\_\_\_ f. Save and close the editor.

\_\_\_ 3. Change the version on the bundle and the package.

\_\_\_ a. Still in the com.ibm.ws.eba.helloWorldService\_V1.0.1 package navigate to:

BundleContent → META-INF

and edit the MANIFEST.MF file.

\_\_\_ b. The editor should open in the Overview tab. The tabs are across the bottom of the window. Click the MANIFEST.MF tab and have a look at the text of the file. Particularly note the Bundle-Version and the version on the Export-Package statement. You could change them here but do not in order to get experience with the editor.

\_\_\_ c. Change the package version

- 1) Click the **Runtime** tab
- 2) In the **Exported Packages** pane, select *com.ibm.ws.eba.helloWorld(1.0.0)*
- 3) Click **Properties**
- 4) Change the “**Version exported**” to 1.0.1 and click **OK**.

\_\_\_ d. Change the bundle version

- 1) Click the **Overview** tab

2) In the General Information pane change the Version to 1.0.1

\_\_\_ e. Click the MANIFEST.MF tab and note your changes, save and close the editor

\_\_\_ 4. Export the new bundle

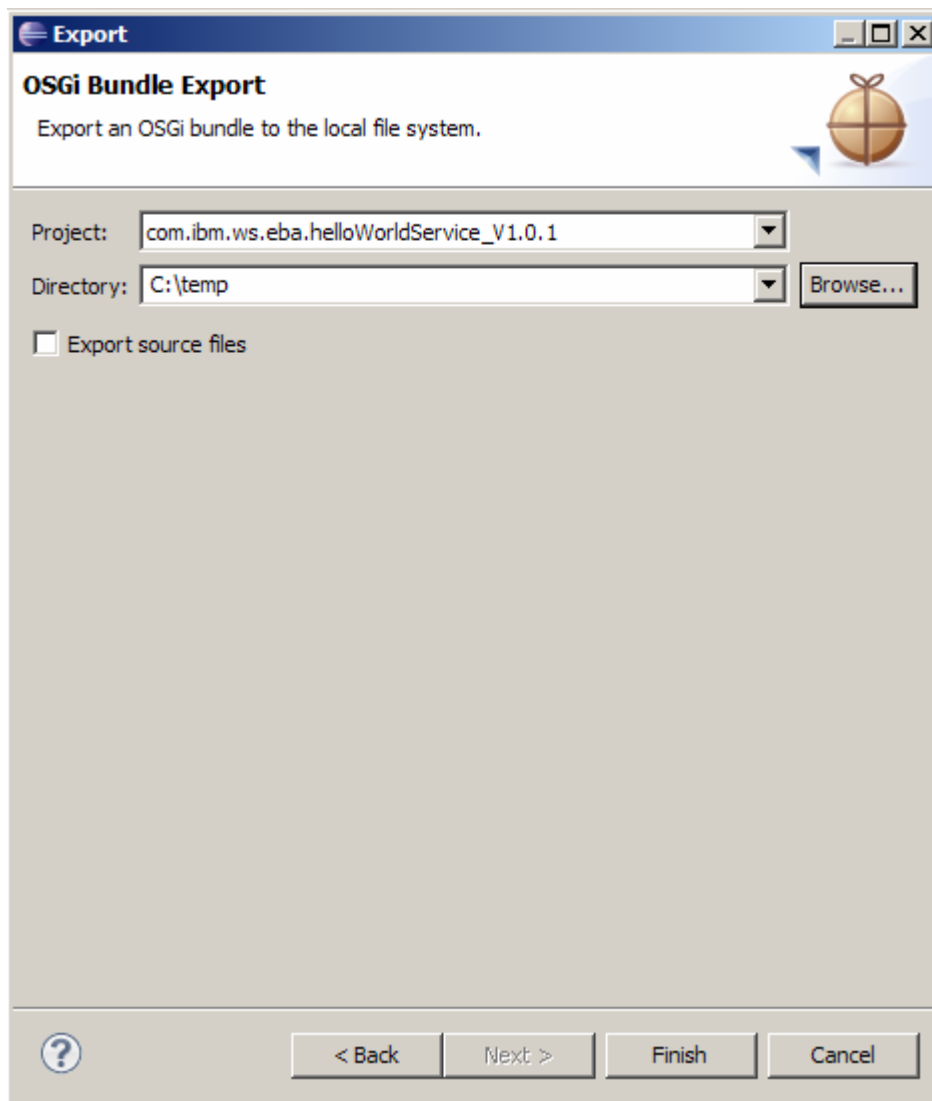
\_\_\_ a. Click the minus sign to collapse the com.ibm.ws.eba.helloWorldService\_V1.0.1 project

\_\_\_ b. Click the project to select it

\_\_\_ c. Right click and select **Export**.

\_\_\_ d. Select OSGi → OSGi Bundle and click Next

\_\_\_ e. On the OSGi Bundle Export panel, verify the name of your project in the Project: field and either type the directory name where you want to put the bundle or use the Browse button to navigate to it. Your completed panel may look like this:



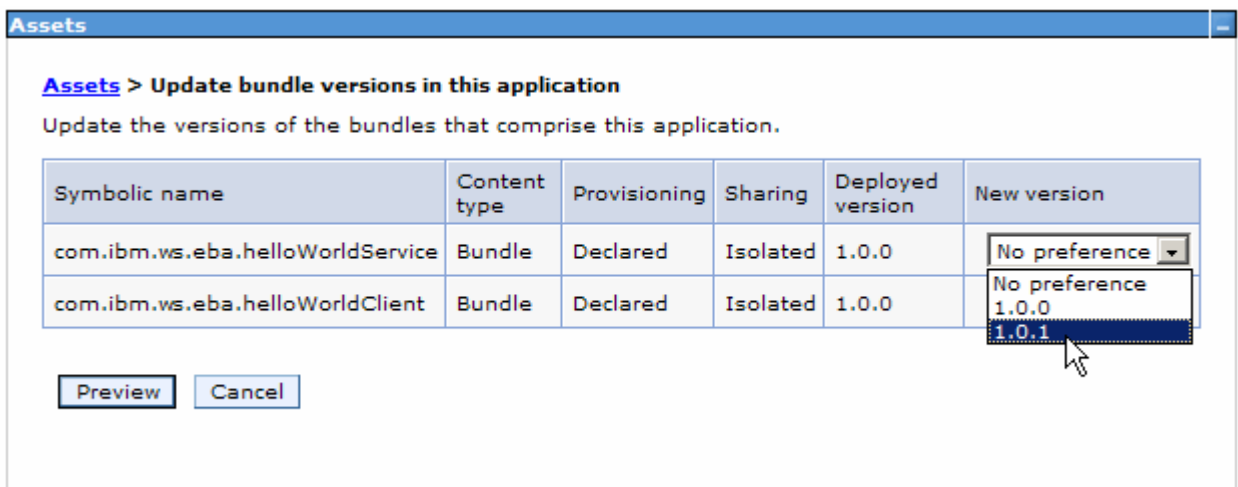
- \_\_\_ f. Look at the file created by the export. Note that the name did not come from the name of the renamed project but from the bundle name and the version of the bundle.
- \_\_\_ 5. Import your new bundle into your WebSphere Application Server instance's internal repository
  - \_\_\_ a. Open an administrative console session
  - \_\_\_ b. From the left navigation panel select:

**Environment → OSGi bundle repositories → Internal bundle repository**

- \_\_\_ c. On the Internal bundle repository panel click New.
- \_\_\_ d. On the **Upload bundle** box either type the full path of your new bundle or use **Browse** to navigate to it and select it. Click **OK** to upload the bundle.
- \_\_\_ e. When returned to the Internal bundle repository window note that your bundle is now in the collection and click **Save**.
- \_\_\_ 6. Update your application to use the new bundle
  - \_\_\_ a. From the left navigation panel select:

**Applications → Assets**

- \_\_\_ b. Click the com.ibm.ws.eba.helloWorld.eba asset
- \_\_\_ c. In the new panel you see the properties of the asset. Near the bottom of this view there is a link named **Update bundle versions in this application**. Click it.
- \_\_\_ d. In the new panel you will see a collection containing the two bundles of your application. In the New version column each of them has a pull-down which allows one to change the version of the bundle the application will use. Note that there is only one version of the client bundle but because you have created a version 1.0.1 bundle there are two versions of the service bundle. Use the pull-down to change the version of the service to 1.0.1.



- \_\_\_ e. Click **Preview**.
- \_\_\_ f. On the **Preview** screen, click **Commit**.
- \_\_\_ g. You will be taken back to the asset panel, click **Save**.

\_\_\_ 7. Restart your application and check the results






















- \_\_\_ a. From the left navigation panel select **Applications** → **Application Types** → **Business-level applications**.
- \_\_\_ b. Locate the helloworld application, select the check box beside it and click **Stop..**
- \_\_\_ c. Once it shows stopped, select the check box again and click **Start**.
- \_\_\_ d. Look at your system out log the same way you did before and note the difference in the output.
- \_\_\_ e. You should see something like:

```
[5/25/10 16:43:35:765 EDT] 00000087 SystemOut      0 Client: Start...
[5/25/10 16:43:35:765 EDT] 00000087 SystemOut      0 OSGi Service: Hola Mundo!
[5/25/10 16:43:35:765 EDT] 00000087 SystemOut      0 Client: End...
```

## What you did in this exercise

In this exercise the key thing you did was update an application by deploying a newer version of one of its bundles and then changing it to use a newer version. The updates you made to the metadata gave both the bundle and the package being exported a new identity so the bundle could be deployed and the package could be selected for use by the application.

If you completed the prerequisite exercise in the Information Center where all of the artifacts were described and relationships between the code and the metadata were explained this should pull it together for you with regard to the OSGi metadata. If you did not do that exercise before this one it is a good idea to go back and do it for the explanations which accompany the instructions.

- [-]  **Feature Pack for OSGi Applications and Java Persistence API 2.0 (All Operating Systems)**
  - [+]  Installing and uninstalling the feature pack on distributed operating systems
  - [+]  Installing and configuring the feature pack on z/OS systems
  - [-]  Working with OSGi Applications
    - [+]  About OSGi Applications
    - [-]  **Developing and deploying an OSGi application**
      -  Creating a service bundle
      -  Creating a client bundle
      -  Creating an OSGi application
    - [+]  Deploying an OSGi application as a business-level application
      -  Debugging bundles at run time using the command-line console
      -  Sample OSGi applications
  - [+]  Administering OSGi Applications
  - [+]  Converting existing applications to OSGi applications
  - [+]  Troubleshooting OSGi Applications
    -  Securing OSGi Applications
  - [+]  Other OSGi Applications-related information
  - [+]  Working with the Java Persistence API (JPA) 2.0
  - [+]  Reference
    -  Release Notes
    -  Site Map

This page is left intentionally blank.