Basic Tutorial: Activating the Hello World Bundle

Contents

[Overview 1](#_Toc336442204)

[Step 1: Create a HelloWorld bundle with the New Plug-in Project Wizard 1](#_Toc336442205)

[Step 2: Activate and run the HelloWorld Bundle 3](#_Toc336442206)

[Setting the Eager Activation Policy 4](#_Toc336442207)

[Setting the Bundle Class-Path 4](#_Toc336442208)

[Activating the Bundle 4](#_Toc336442209)

[Step 3: Managing the Bundle Life Cycle 5](#_Toc336442210)

[Changing the Source 5](#_Toc336442211)

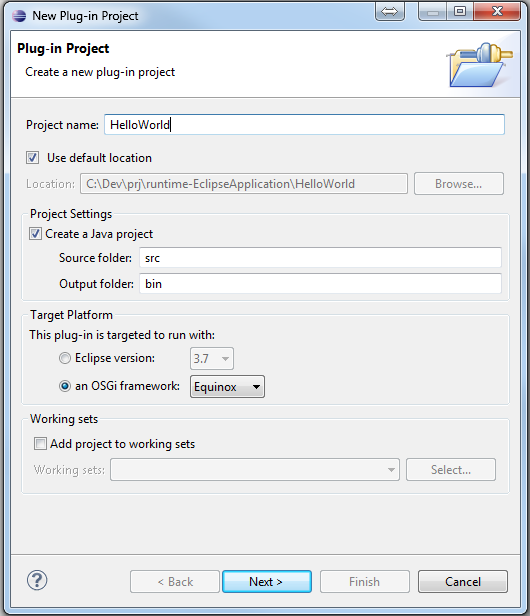
[Manage the Bundle Life Cycle. 6](#_Toc336442212)

## Overview

This tutorial guides you through the steps of managing the life cycle of a bundle, including activating and executing the bundle based on source code changes.

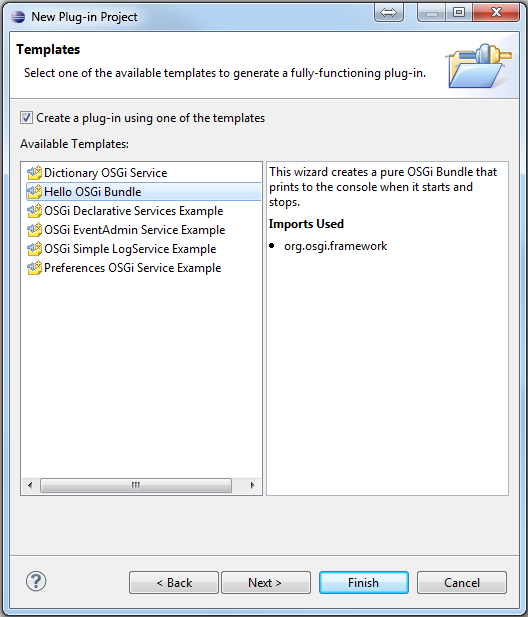
## Step 1: Create a HelloWorld bundle with the New Plug-in Project Wizard

Inside Eclipse select the menu item **File | New | Other | Plug-in Project...**. to open the **New Plug-in Project** wizard.



Enter HelloWorld in the Project Name field. Select the “**an OSGi framework”** radio button in the Target Platform section. Make sure that you select Equinox or Standard in the drop down box following the target radio button. Then click **Next** twice.

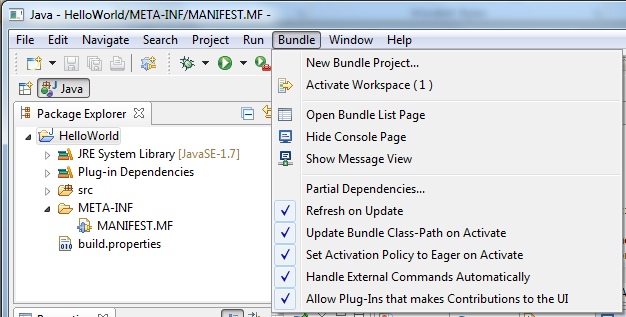
Select the **Hello OSGi Bundle** template and click **Finish**.



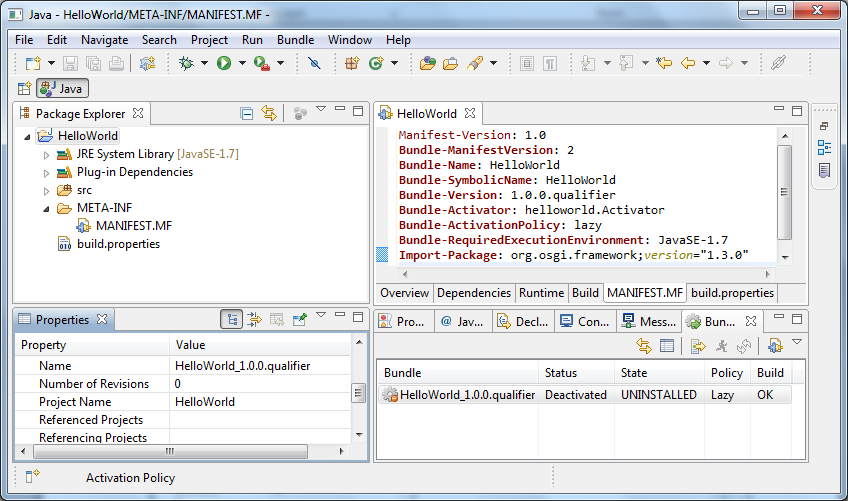
This tutorial use the Java perspective, but you may use any perspective. If not **Build Automatically** is on, you have to manually build the project before you can activate the HelloWorld bundle. For this tutorial check the **Build Automatically** menu item in the Project main menu.

## Step 2: Activate and run the HelloWorld Bundle

If not visible, select **Bundle | Open Bundle List Page** from the Bundle main menu.



In the **Bundle List Page** we see that the HelloWorld bundle is deactivated and uninstalled. The project must also be free from build errors to get activated.



### Setting the Activation Policy

By default the [Set Activation Policy to Eager on Activate](../tasks/Setting%20Bundle%20Options.htm#SetActivationPolicyToEagerOnActivate) option in the bundle main menu is enabled. This means that if the activation policy is lazy for the bundle it is set to an eager activation policy when the bundle is activated. The policy is set to lazy for the Hello OSGi Bundle template when generated by the “New PIug-in Project” wizard. In general this tells us that when a bundle has a lazy activation policy, the bundle will not be loaded into memory when started. The class loading first happens when someone need some capability or functionality from the bundle. Someone may for instance be another bundle, an external tool or a user activating a requiring bundle through the UI. In our case we want the activation policy to be Eager so the bundle runs when we activate it. To achieve this assure that the **Set Activation Policy to Eager on Activate** option is checked in the Bundle main menu. An alternative is to remove the **Bundle-ActivationPolicy: lazy** header from the manifest file or from the overview tab in the manifest editor.

If you activate the bundle with a lazy activation policy, you can always change this after the bundle is activated by enabling the [Eager Activation Policy](../tasks/Setting%20Bundle%20Options.htm#EagerActivationPolicy) menu entry from the pop-up context menu in the bundle view

### Setting the Bundle-ClassPath

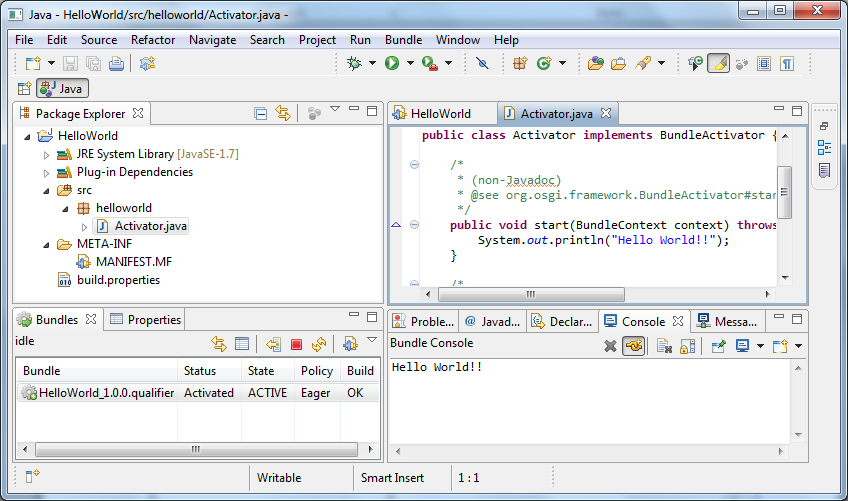
Also assure that the [Update Bundle Class-Path on Activate/Deactivate](../tasks/Setting%20Bundle%20Options.htm#AddbintoBundleClassPathonActivate) is checked in the Bundle main menu. This is the default and instructs OSGi to resolve and load the classes from the default output folder directory. This is the same as set as the [output folder](../tasks/Setting%20Bundle%20Options.htm#OutputFolder) in the **New Plug-in Project** wizard.

See [Setting Bundle Options](../tasks/Setting%20Bundle%20Options.htm) for an explanation of all available options.

### Activating the Bundle

To activate and run the bundle select **Bundle | Activate Workspace (1)** from the Eclipse main menu.

When the bundle is activated in-place it is first installed, then resolved and lastly started. When successfully started the bundle enters the ACTIVE state. This can be seen in the Bundle List Page where the State field has changed from UNINSTALLED to ACTIVE. Starting a bundle loads the bundle in to memory and executes the Start method in the Activator class of the bundle.



The HelloWorld bundle prints “Hello World!” in the Start method and “Goodbye World!” in the Stop method as can be seen in the Start and Stop methods in the helloworld.Activator class. The output is displayed in the Bundle Console. If the Bundle Console is hidden, you can bring it up by selecting **Show Console Page** from the Bundle main menu.

## Step 3: Managing the Bundle Life Cycle

After a bundle is activated, it participates in a round trip cycle of code change, build and execution, as illustrated in this last step of this tutorial.

### Changing the Source

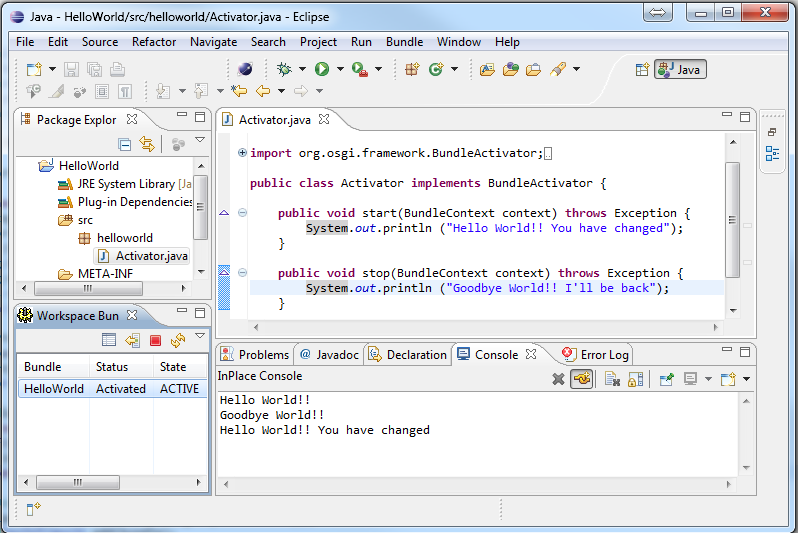
Open helloworld.Activator.java and change the print statement in the start method to:

System.out.println ("Hello World!! You have changed");

And the print statement in the Stop method to:

System.out.println ("Goodbye World!! I'll be back");

Then save the file. If **Build Automatically** is checked and the [Update on Build](../tasks/Setting%20Bundle%20Options.htm#UpdateOnBuild) option in the main bundle menu is enabled you should see the following in the Bundle Console:



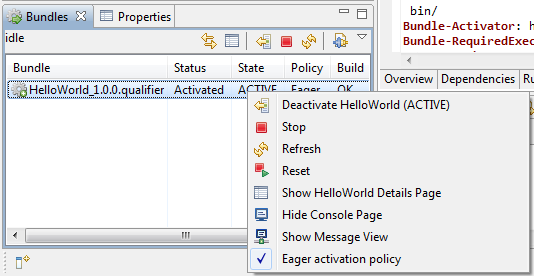
After the bundle has been built, it is updated. If the bundle is in state ACTIVE when updated it is first stopped, unresolved and uninstalled. The changed bundle is then read from the bundle location, installed, updated and lastly started again.

This is illustrated in the Bundle Console, where ”Good Bye World!!” is the result of invoking the Stop method, which prints a message to standard out, when the bundle is stopped. Note that “I’ll be back” is not part of the printout when the bundle is stopped the first time. This is because the changes in the Activator class have not been read and the changed bundle is not updated at that time. The last message in the console view is the result of the OSGi start operation invoking the Start method printing the new message from the updated bundle.

### Manage the Bundle Life Cycle.

So far we have been through the most common lifecycle workflow; - Activating a bundle; -Performing code changes and saving the file; which - Trigger an automatic build; which results in an - Update and a restart of the bundle with the newly built source. The source code changes part is done by you, and the rest of the bundle operations are done behind the scenes, triggered by an automatic build.

The life cycle of a bundle is also controlled through menu and tool bar commands. There is one main menu and a context menu when selecting a project in Package Explorer or a bundle in the Bundle List Page. As a short cut you may use the tool bar in the Bundle List Page. The menus are dynamic in the sense that they are changing dependent on the state of the different workspace bundles you are working with. In our case we have one active bundle. This allows for the following life cycle commands:



* **Deactivate**

When deactivating the HelloWorld bundle it is stopped, un-resolved and uninstalled. In the uninstalled state, the bundle is not active in the workspace, and the InPlace Activator will ignore it, until activated again. See [Activate and Deactivate Bundles](../tasks/Activate%20and%20Deactivate%20Bundles.htm) for a detailed explanation of the Activate and Deactivate commands.

* **Stop**

If you select the Stop menu item, the stop method is invoked, and the “Goodbye …” message is printed to the console. After stopping the bundle, it can be started from the same menu entry. See [Start and Stop Bundles](../tasks/Start%20and%20Stop%20Bundles.htm) for a detailed explanation of the Start and Stop commands.

* **Refresh**

Refresh is an operation that renews the bundle by moving it to the installed state and then resolves it before starting it again, provided that it was in the active state before it was refreshed. You will see the start and stop messages in the console when you try it. Resolve is the process of renewing all dependencies between bundles. This is discussed in more depth in [Update and Refresh Bundles](../tasks/Update%20and%20Refresh.htm). There is also a tutorial illustrating this when working with [multiple bundles](Tutorial%20Working%20with%20Multiple%20Bundles.htm) that are dependent on each other.

* **Reset**

Resetting the workspace or a bundle is a way to get a clean workspace state where all revisions are released and new revisions created. All workspace bundles will only have one revision after a reset. See [Resetting Bundles](../tasks/Resetting%20Bundles.htm) for a detailed explanation of the Reset command.

This completes the basic tutorial. Thanks for your time.