Working with Dependencies

Contents

[Overview 1](#_Toc336442151)

[Introduction 1](#_Toc336442152)

[Create the LogService bundle 1](#_Toc336442153)

[Add a Dependency between the two Bundles 2](#_Toc336442154)

[Activating and deactivating Bundles with Dependencies. 4](#_Toc336442155)

## Overview

A basic understanding of dependencies between bundles and how these dependencies are resolved is the goal of this tutorial. We start out with our HelloWorld bundle from the basic tutorial and add an OSGi service bundle that logs our two messages from the Start and Stop methods in the HelloWorld bundle.

## Introduction

For those of you unfamiliar with the terms used when working with bundles, a very short explanation of the most relevant terms for this tutorial are presented.

The life cycle of a bundle represents the possible states a bundle can take. Typically, when a bundle is activated it goes from an initial uninstalled state, to be installed, resolved and then started. When started, the bundle is in an active state. The most interesting state is probably the resolved state, and the conditions that must be satisfied for a bundle to enter the resolve state. If a bundle is resolved, it can become active and collaborate with other resolved bundles. Before a bundle can enter the resolved state, all dependencies the bundle has on other bundles must be satisfied.

When a bundle is installed or updated it is assigned a revision and that revision has a wiring with the bundles it collaborate with.

## Create the LogService bundle

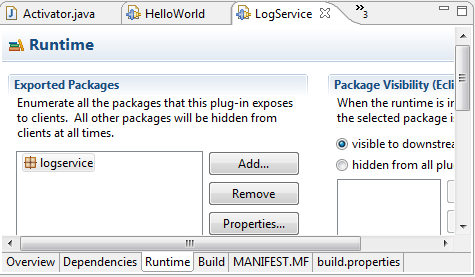
If you haven’t created the [HelloWorld bundle](Basic%20Tutorial%20Activating%20Bundles.htm), follow step 1 in the basic tutorial to do so. We follow the same procedure as in the basic tutorial - Step 1 - to create the LogService bundle. Select the **OSGi Simple LogService Example** template instead of the **Hello OSGi Bundle** template to create the bundle in the **New Plug-in Project** wizard. Assure that the [Update Bundle-ClassPath on Activate/Deactivate](../tasks/Setting%20Bundle%20Options.htm#AddbintoBundleClassPathonActivate) option in the Bundle main menu is checked.

You should now have two projects in your workspace; - The LogService; and - the HelloWorld project. Activate both bundles from the Eclipse main menu by selecting the **Bundle | Activate Workspace (2)** menu item where 2 specifies the number of deactivated bundles in the workspace. Both bundle projects should now be activated and in state ACTIVE.

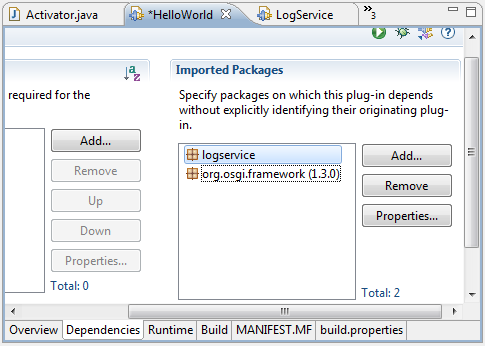
## Add a Dependency between the two Bundles

To provide and make the logging capability visible to the HelloWorld and other bundles, the log capability has to be exported. In the same manner, to consume a capability from another bundle, the exported capability has to be imported.

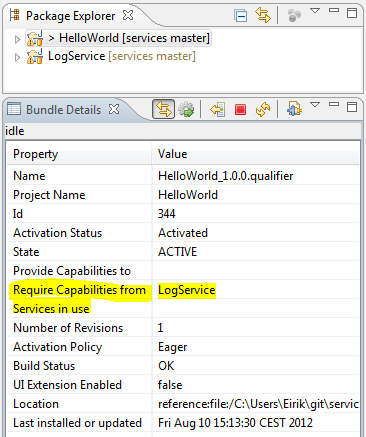
To do so, edit the LogService manifest.mf file by selecting the **Runtime** tab, then click the **Add ...** buttonin the **Exported Packages** region and add the logservice package to the exported packages and save the file.



To import the logservice package edit the manifest.mf in the HelloWorld project by selecting the **Dependencies** tab and click the **Add …** button in the **Imported Packages** region. Then select the logservice package and add it to the imported packages.



Now, by saving the bundles (File | Save All), the HelloWorld bundle becomes wired to the LogService bundle. You can verify this by selecting the **Show Bundle List Page** item from the Bundle main menu and select the HelloWorld bundle in the Bundle List Page and then select the Bundle Details Button. This reveals the **Bundle Details Page** for the HelloWorld Bundle.



At this stage we see that HelloWorld consumes or requires capabilities from LogService but is not registered with LogService service yet. By checking the link with button C:\Dev\prj\e4.2\ui\no.javatime.inplace.ui\icons\synced.gif in the Bundle Details Page and selecting the LogService project in the Package Explorer, we also see in the Details Page that the LogService bundle provides capabilities to HelloWorld and that the LogService service is registered.

Now it is time to actually utilize the log service from HelloWorld. To use a service we have to add some boiler plate code to access the service from the HelloWorld bundle.

First import **org.osgi.util.tracker** into the HelloWorld project in the same way as you imported the logservice package earlier in this tutorial.

Replace all of the source code in the helloworld.Activator.java file in the HelloWorld project with the content from the following listing. You should be able to copy and paste the code in the listing. Then save the file.

package helloworld;

import logservice.SimpleLogService;

import org.osgi.framework.BundleActivator;

import org.osgi.framework.BundleContext;

import org.osgi.util.tracker.ServiceTracker;

**public** **class** Activator **implements** BundleActivator {

**private** ServiceTracker<SimpleLogService, SimpleLogService> simpleLogServiceTracker;

**public** **void** start(BundleContext context) **throws** Exception {

simpleLogServiceTracker = **new** ServiceTracker<SimpleLogService, SimpleLogService>

(context, SimpleLogService.**class**, **null**);

simpleLogServiceTracker.open();

log("Hello World!! You have changed");

}

**public** **void** stop(BundleContext context) **throws** Exception {

log("Goodbye World!! I'll be back");

simpleLogServiceTracker.close();

simpleLogServiceTracker = **null**;

}

**private** **void** log(String msg) {

SimpleLogService logService = getLogService();

**if** (logService != **null**) {

logService.log(msg);

} **else** {

System.*out*.println("Alternative log: " + msg);

}

}

**private** SimpleLogService getLogService() {

**return** simpleLogServiceTracker.getService();

}

}

If built successfully, the HelloWorld bundle is updated and re-wired with the LogService. If you now take a look at the Bundle Details Page of the HelloWorld project again, the Services in use field reveals that the LogService service is in use by the HelloWorld bundle. The output in the Bundle Console Page is the same as before, except that now it is routed through the LogService service. If closed, open the Bundle Console Page from the Bundle main menu.

We have now registered the LogService bundle as a service using the OSGi service platform and implemented a consumer of that service.

## Activating and deactivating Bundles with Dependencies.

By now we know that HelloWorld requires log capabilities from LogService and LogService provide log capabilities to HelloWorld and to bundles in general.

If we deactivate LogService, both the HelloWorld and the LogService bundles will be deactivated and enter state UNINSTALLED.

A bundle is resolved (state RESOLVED) before it is started (state ACTIVE). During the time HelloWorld is resolved it will be in a stable state with respect to its code dependencies. LogService can be unresolved (entering state INSTALLED) and uninstalled during its life cycle. However, as long as HelloWorld is resolved it will continue to load classes from the LogService bundle revision it was wired to when it was initially resolved and started, even if LogService is unresolved or uninstalled.

One alternative is to let HelloWorld stay resolved and be in an active state, after LogService has been uninstalled, and function as normal. Another alternative is to implicit deactivate all bundles that, directly and indirectly, require capabilities from the bundle being deactivated. In a development environment, where bundles are continuously changed the default is to deactivate requiring bundles when a provider is deactivated.

In our case HelloWorld is deactivated along with LogService. The dependencies are directed, so if you deactivate HelloWorld, LogService will stay activated.

The same applies when activating bundles, but in the opposite direction. If you activate HelloWorld, LogService will be activated as well. If you activate LogService, HelloWorld stays deactivated. It is possible to influence in to which extend dependent bundles are being activated, deactivated, started or stopped by the [Partial Dependencies](../tasks/Setting%20Bundle%20Options.htm#Dependencies) option.

Also note that dependencies are transitive. If bundle A is dependent on bundle B and B is dependent on bundle C, than bundle A is dependent on bundle C. This may result in a cascade of activations or deactivations when a bundle is activated or deactivated.