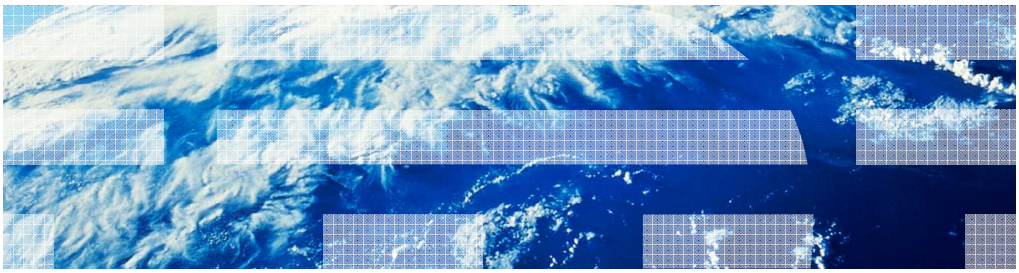
IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0

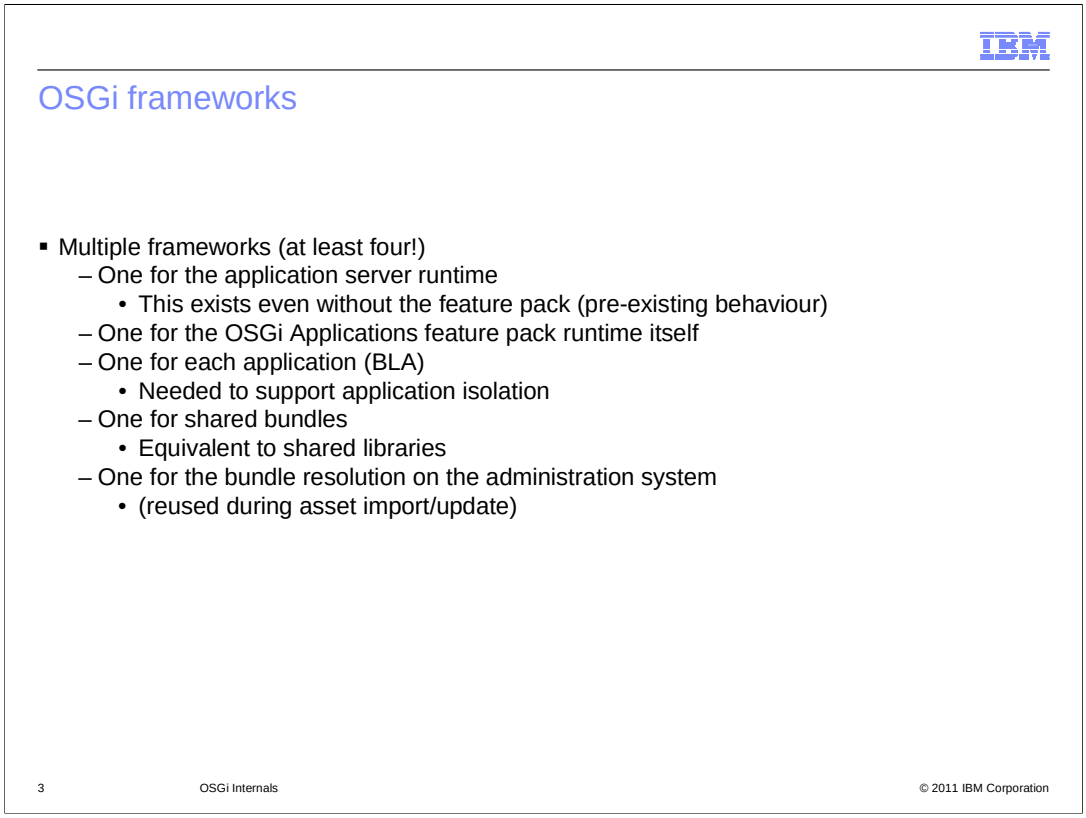


WebSphere software

This module covers OSGi Internals for the IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0.

Table of contents

This educational module for the OSGi applications feature pack will take a deeper dive into some of the internals of the feature pack, including how the feature pack interacts with the WebSphere application server and some troubleshooting hints and tips.



An OSGi framework is actually a component that holds the OSGi bundles, manages the life cycle and gets them started as appropriate.

Since version 6.1 of the WebSphere application server, the application server itself uses an OSGi bundle framework to manage the code of the application server itself. This support was used as the basis for addition of runtime provisioning of the application server in version 7 (for example, the EJB container isn't started until it is actually needed by an application that uses EJBs).

This means that there are several OSGi frameworks running once you have the feature pack installed and have some OSGi application installed and running.

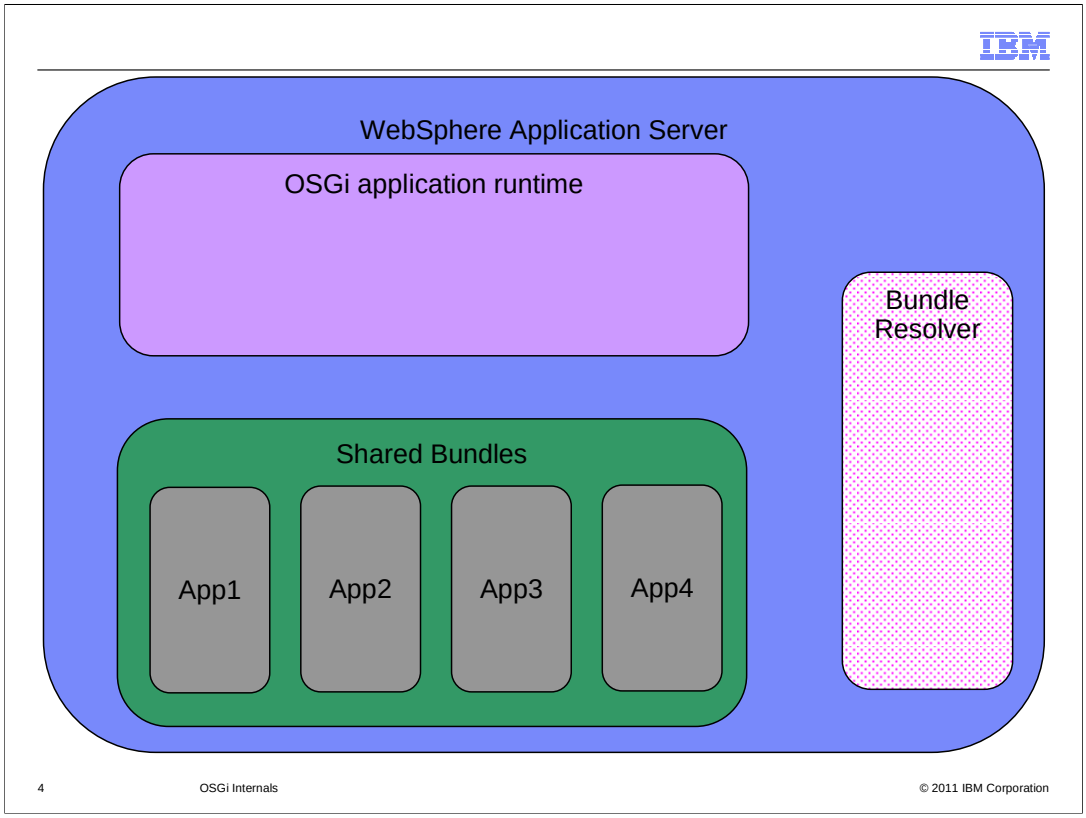
The first framework is that of the application server itself. Some of the code in the OSGi Applications Feature Pack runs in this framework (basically this is the administration code, plus that needed to start the other frameworks)

The next framework is for the feature pack itself. This one is needed as the OSGi framework used by the application server is comes eclipse equinox 3.2.1, yet in order to support the OSGi applications, the feature pack needed to use version 3.5.1. The feature pack therefore starts up a second OSGi framework within which the feature pack runs.

Next each application runs in its own instance of the OSGi framework. This is to ensure that the applications are correctly isolated between each other as they run.

Of course in many cases the applications do need to share classes. This can be achieved when those bundles are placed in another framework that allows multiple applications to use the same bundle. One of the tasks for the OSGi applications runtime is to ensure that the bundles in the shared framework are available to the individual applications (running in their own framework).

Finally, there is yet one framework that is used during bundle resolution. This only exists



This is another view of the various OSGi frameworks running in an application server that has the OSGi applications feature pack installed. It shows what functions are available to code running in that framework.

So code running in the App1 framework, for example, is able to use functions from its own framework, from the shared bundle framework and from the WebSphere Application Server framework.

It cannot use function from the OSGi application runtime or from the bundle resolver or from the other applications, App2, App3 or App4.

JNDI Changes

- OSGi applications need to be able to use JNDI
  - However when looking up a object, the result must be something that comes from a classloader visible to the application
    - Rather than provide a new implementation OSGi runtime wraps the existing implementation
      - And all the results!

One interesting area of complication for OSGi applications is what happens when that application uses JNDI to locate information and functions that it wants to use. In order to support this functionality, the OSGi Application feature pack runtime has to ensure that the objects (and, in particular, the classes of those objects) are usable by the application.

If the runtime did nothing to interpose itself in the interaction with JNDI, an OSGi application can perform a lookup of an object in JNDI and then discover that the class of that object is not a class that the OSGi application itself is permitted to use (that is. instanceof checks made by the application will always fail as the class and interfaces of that object are not those of the OSGi framework in which the application is running.

To overcome this problem, the OSGi runtime uses Java dynamic proxies to wrap the actual objects retrieved from JNDI allowing the object given to the OSGi application to implement the interfaces that are expected by that application.

So the actual object the application gets back is not the actual object, rather a proxy object that points to the object in JNDI so if you do an initial context and do a lookup and get another context from JNDI, that context is wrapped as well and that keeps going until you get to the final object for which you are searching. Once you get a proxy object for that final object the results are passed through as normal.

Interaction with the web container

- OSGi web applications
  - Web container handles the runtime
  - Web container sees a standard web application
    - OSGi Applications framework creates a virtual web applications
    - Looks like a standard web application to the web container

The web content of any OSGi application is handled in the typicalmanner by the application server, namely that the content is managed by the web container. The web container has not been modified and, from its perspective, the web content is a standard web application.

To achieve this, the OSGi application framework creates a virtual web application that looks like a standard web application to the rest of the application server, but whose functionality is provided by the OSGi application.

Troubleshooting

- Possible GOTCHA's
- MUSTGATHERs
  - Including trace

Let's now move onto some hints and tips about how to troubleshoot OSGi applications and the OSGi framework, together with the "must gather" information that is required by IBM Service to investigate PMRs related to the OSGi Applications feature.

Possible GOTCHA's

- If two bundles have the same symbolic name and version, they are treated as the same.
- If many bundles provide an export which can satisfy another bundle's import, the bundle resolver can pick any of them to resolve the import

Both the OSGi specification and the OSGi applications feature assume that the symbolic name and bundle version of a bundle identifies the bundle uniquely. The symbolic name and bundle version are defined in the manifest file (the file META-INF/MANIFEST.MF within the bundle). The symbolic name is required to be defined by the manifest file, while the bundle version defaults to 0.0.0.

Once a bundle has been discovered by OSGi applications feature pack runtime (that is., it has been downloaded from a bundle repository), the bundle is saved by the runtime in the bundle cache. If, subsequently, another bundle requires packages or services that can be satisfied by that bundle, that saved bundle is reused by the bundle resolution process. With this release even deleting a bundle from the repository does not delete it from the bundle cache so deleting a bundle then installing a different bundle using the same name and version will not result in a change to the application.

As a consequence of this behavior, if you need update such a bundle you will need to update the bundle version and will need to update the applications that need the bundle to require the new bundle version.

A second consequence is that if a specific combination of symbolic name and bundle version can be satisfied by bundles from multiple bundle repositories you cannot determine from where the OSGi application will download the bundle when it needs to be used to resolve an OSGi application that says it needs that combination of symbolic name and bundle version.

Basic troubleshooting

- Use the osgiApplicationConsole to investigate results of the bundle resolution and what is actually in each OSGi framework

If you need to discover how your OSGi application has been wired, or what bundles are loaded into the shared bundle framework, you can use the osgiApplicationConsole command line tool to explore the contents of the frameworks.

This tool cannot examine the contents of either the OSGi framework used by the application server directly or the OSGi framework used by the OSGi Applications runtime.

The “must gather” information required for the OSGi Applications feature is that information that is required by IBM Service to investigate problems reported with the feature.

In the ideal case, the must gather information is collected by using the IBM Support Assistant. This program will collect the ffdc (“first failure data capture”) reports, the configuration of the application server. In addition, if the problem can be easily reproduced, the support assistant can enable the relevant trace group while reproducing the problem.

If the support assistant cannot be used and trace needs to enabled manually, the standard trace specification to use is “Aries*=all” (the asterisk in the specification enables trace for all trace groups that start Aries. As the trace groups used by all of the new code for the OSGi applications feature pack start with Aries this will capture everything introduced by the OSGi Applications feature pack.

Of course tracing everything does generate a *lot* of trace, so need to enable just a subset of the available trace. For example, the “Aries.eba.bundledownload=all” trace specification will enable trace for the bundle download process (this is subcomponent in charge of downloading bundles from bundle repositories.

If you know the problem is in a specific area you may want to narrow it down, otherwise you will have a lot of information. The IBM Support Assistant has been updated to allow for this level of tracing.

Summary

- OSGi frameworks
- Web container interaction
- Troubleshooting

This module took a deeper dive into some of the internals of the feature pack, including how the feature pack interacts with the WebSphere Application Aerver and some troubleshooting hints and tips.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_wasosgijpafep_OSGi_internals.ppt

This module is also available in PDF format at: ../wasosgijpafep_OSGi_internals.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information