



| IBM Software Group | Rational software

Diagnosing Eclipse configurations using Rational® Diagnostic Tool For Eclipse

Module 2: Introduction to OSGi technology

Rational software



@business on demand.

© 2007 IBM Corporation

Updated November 2, 2007

This module will cover an introduction to OSGi technology and framework.

Course Agenda

- Module 1: Understanding Eclipse
- **Module 2: Understanding Equinox and OSGI Bundles**
- Module 3: Using the Rational Diagnostic Tool for Eclipse 3.2



The objectives of this module are to help you understand OSGi, the Equinox and OSGI Bundles, and exposure to some error conditions.

Introduction to OSGi technology

- Dynamic module system for the Java™ platform
 - ▶ Includes visibility rules, dependency management and version handling of bundles, and the OSGi modules
- It is dynamic
 - ▶ Installing, starting, stopping, updating, uninstalling bundles, all dynamically at runtime
- It is service-oriented
 - ▶ Services can be registered and consumed inside a Virtual Machine, again all dynamically at runtime
 - ▶ A specification of the OSGi Alliance, a non-profit organization.
 - ▶ For more information: <http://www.osgi.org>



OSGi technology is the dynamic module system for Java. The OSGi service platform provides functionality to Java that makes Java the premier environment for software integration, and thus, for development. The OSGi technology provides the standardized primitives that allow applications to be constructed from small, reusable and collaborative components. These components can be assembled into an application and deployed.

The OSGi service platform provides the functions to change the composition dynamically on the device of a variety of networks, without requiring restarts.

It is service-oriented, which means services can be registered and consumed inside a virtual machine. For more information on the OSGi technology or the OSGi alliance, visit www.osgi.org.

Equinox and the OSGi

- Equinox is an implementation of
 - ▶ OSGi Alliance R4 core specifications
 - ▶ a set of bundles that implement various optional OSGi services
 - ▶ other infrastructure for running OSGi-based systems
- Equinox framework
 - ▶ produces launchers
 - ▶ bootstrap infrastructure and application models that facilitate the use of Equinox OSGi in end-user product scenarios.



The OSGi Alliance (formerly known as the Open Services Gateway initiative - now an obsolete name) is an open standards organization founded in March 1999. The Alliance and its members have specified a Java-based service platform that can be remotely managed. The core part of the specifications is a framework that defines an application life cycle model and a service registry.

Based on this framework, several OSGi layers, APIs, and services have been defined:

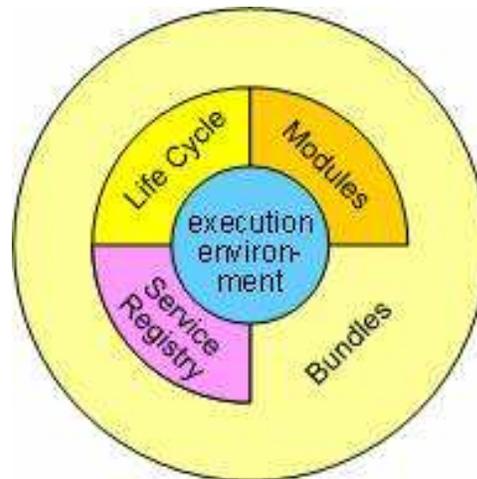
Equinox is a implementation of the OSGi R4 Core specification, beyond that are selected implementations of bundles from other standard organizations such as MEG and VEG.

The OSGi Mobile Expert Group (MEG) is chartered to define the requirements and specifications to tailor and extend the OSGi Service Platform for mobile devices that are data-capable, and capable of connecting to wireless networks. Examples of such devices include, but are not limited to, digital mobile phones, smart phones, Personal Digital Assistants (PDAs), and so on.

Vehicle Expert Group (VEG) is chartered to tailor and extend the OSGi specification in order to meet vehicle-specific requirements. To achieve this, the VEG will define a list of topics that cover vehicle-specific issues.

OSGi framework

- The core component of the OSGi specifications is the OSGi framework.
- The framework provides a standardized environment to applications (called bundles).
- The framework is divided in a number of layers:
 - ▶ L0: Execution environment
 - ▶ L1: Modules
 - ▶ L2: Life cycle management
 - ▶ L3: Service registry



The core component of the OSGi specifications is the OSGi framework. The framework provides a standardized environment to applications (called bundles). The framework is divided in a number of layers

The L0 Execution environment is the specification of the Java environment. Java 2 Configurations and Profiles, such as J2SE, CDC, CLDC, and MIDP are all valid execution environments.

The L1 Modules layer defines the class loading policies

The L2 Life Cycle layer adds bundles that can be dynamically installed, started, stopped, updated and uninstalled. Bundles rely on the module layer for class loading but add an API to manage the modules in run time.

The L3 layer adds a Service Registry. The service registry provides a cooperation model for bundles that takes the dynamics into account.

And finally, a ubiquitous security system is deeply intertwined with all the layers.

(source: www.osgi.org).

The life cycle of a bundle



The OSGi framework manages the life cycle of bundles. As you install and run a bundle, it goes through various states including: installed, resolved, starting, active, stopping, and uninstalled.

INSTALLED - the bundle has been installed, but all of the bundle's dependencies have not been met. The bundle requires packages that have not been exported by any currently installed bundle.

RESOLVED - the bundle is installed, and its dependencies have been met, but it is not running. If a bundle is started and all of the bundle's dependencies are met, the bundle skips this state.

STARTING - a temporary state that the bundle goes through while the bundle is starting.

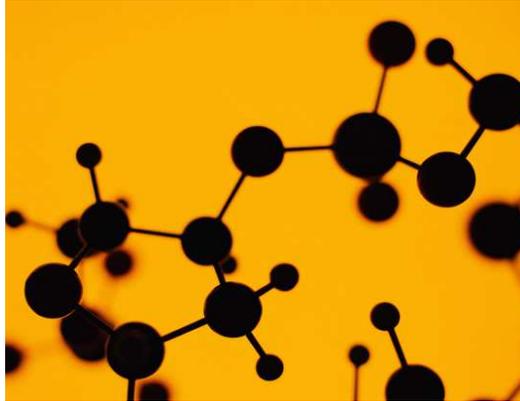
ACTIVE - the bundle is running.

STOPPING - a temporary state that the bundle goes through while the bundle is stopping.

UNINSTALLED - the bundle no longer exists in the framework.

The bundle manifest

- Bundle-Name
- Bundle-Description
- Bundle-Copyright
- Bundle-Vendor
- Bundle-Version
- Bundle-DocUrl
- Bundle-ContactAddress
- Bundle-Fragment



Each bundle must contain a manifest file. The bundle's manifest file contains data that the framework needs to correctly install and activate the bundle. Legacy Eclipse bundles can provide some manifest information in their plugin.xml files, but META-INF/MANIFEST.MF files are the recommended files for Manifest information.

If a bundle contains only a plugin.xml, the Eclipse platform will generate a MANIFEST.MF-equivalent when the platform starts. When you specify data in a manifest file, you must use the headers that were defined by the OSGi specification. You can use user-defined headers, however, the framework ignores any headers that it does not understand. Refer to the OSGi Service Platform Release 4 specification for more information about the OSGi Manifest file format and syntax.

The MANIFEST.MF file is located in the META-INF directory of your bundle project. The plugin.xml file, if present, should be under the root directory.

The following headers are defined in the OSGi Service Release 4 specification and by the Eclipse 3.2 extensions to the OSGi framework:

* **Import-Package** - Use this header to specify the names of any package that you want your bundle to import from the runtime. If you do not specify the package your bundle needs in this header, you may get a NoClassDefFound exception when the bundle loads. Note: You must also specify the package you want to import (using Import-Package) in the Export-Package header of the bundle that contains the package.

* **Export-Package** - Use this header to specify the name of any package that you want your bundle to export to the runtime. If you do not specify the packages needed by other bundles in this header, the dependent bundles may not resolve.

* **Require-Bundle** - Use this header to specify the specific bundles that provides packages you use in your bundle. If you do not specify the bundle which provides the packages you need, you may get a NoClassDefFound exception when the bundle loads.

* **Bundle-Activator** - Use this header to specify the fully-qualified name of the BundleActivator class. A bundle designates a special class to act as a BundleActivator. The Framework must instantiate this class and invoke the start and stop methods to start

Bundles causing malfunctions in eclipse, why?

- Dependencies
- Conflicts
- Permissions
- Incorrect environment



So why are there malfunctions in Eclipse? Think of Equinox as an Air Traffic Controller. Equinox does not care how many aircraft are in the air as they have different identification. Essentially Equinox tries to minimize problems by not allowing any bundle to load unless: 1) they have all of their dependencies, 2) they do not conflict with another Bundle, and 3) they are going to be loaded in an environment that they are expecting.

Bundle errors related to dependencies

- **MISSING_IMPORT_PACKAGE**
 - ▶ Indicates that an Import-Package could not be found
- **MISSING_REQUIRE_BUNDLE**
 - ▶ Indicates that a Require-Bundle could not be found
- **MISSING_FRAGMENT_HOST**
 - ▶ Indicates that a Fragment-Host could not be found



When a bundle does not load due to a dependency error, you might see one of these three errors:

MISSING_IMPORT_PACKAGE, which means that an Import-Package could not be found, MISSING_REQUIRE_BUNDLE indicating that a Require-Bundle could not be found, or MISSING_FRAGMENT_HOST indicating that a Fragment-Host could not be found.

Bundle errors related to conflicts

- **SINGLETON_SELECTION**
 - ▶ Indicates that the bundle could not be resolved because another singleton bundle was selected.
- **FRAGMENT_CONFLICT**
 - ▶ Indicates that the bundle fragment could not be resolved because a constraint conflict with a host.
- **IMPORT_PACKAGE_USES_CONFLICT**
 - ▶ Indicates that an Import-Package could not be resolved because of a “uses” directive conflict.
- **REQUIRE_BUNDLE_USES_CONFLICT**
 - ▶ Indicates that a Require-Bundle could not be resolved because of a “uses” directive conflict.



Referring back to the air traffic controller analogy, Equinox wants the bundles to not fight over air space and be easily identified rather than colliding. When bundle errors occur related to conflicts, you might see one or more of the errors shown here.

Bundle errors related to permissions

- **IMPORT_PACKAGE_PERMISSION**
 - ▶ Indicates that an Import-Package could not be resolved because the importing bundle does not have the correct permissions to import the package.
- **EXPORT_PACKAGE_PERMISSION**
 - ▶ Indicates that an Import-Package could not be resolved because no exporting bundle has the correct permissions to export the package.
- **REQUIRE_BUNDLE_PERMISSION**
 - ▶ Indicates that a Require-Bundle could not be resolved because the requiring bundle does not have the correct permissions to require the bundle.
- **PROVIDE_BUNDLE_PERMISSION**
 - ▶ Indicates that a Require-Bundle could not be resolved because no bundle with the required symbolic name has the correct permissions to provide the required symbolic name.
- **HOST_BUNDLE_PERMISSION**
 - ▶ Indicates that a Fragment-Host could not be resolved because no bundle with the required symbolic name has the correct permissions to host a fragment.
- **FRAGMENT_BUNDLE_PERMISSION**
 - ▶ Indicates that a Fragment-Host could not be resolved because the fragment bundle does not have the correct permissions to be a fragment.



Equinox may also determine that a package does not have the right authorization to do what it wants to do. When Bundle errors occur due to permissions, you may see one or more of the errors listed here.

Bundle errors due to incorrect environment

- PLATFORM_FILTER
 - ▶ Indicates that a bundle could not be resolved because a platform filter did not match the runtime environment.
- MISSING_EXECUTION_ENVIRONMENT
 - ▶ Indicates that a bundle could not be resolved because the required execution environment did not match the runtime environment.
- MISSING_GENERIC_CAPABILITY
 - ▶ Indicates that a bundle could not be resolved because the required generic capability could not be resolved.



Eclipse runs on multiple JREs and operating systems and the Eclipse bundles may not have been designed for all of them. When bundle errors occur due to an incorrect environment, you may see PLATFORM_FILTER, MISSING_EXECUTION_ENVIRONMENT, or MISSING_GENERIC_CAPABILITY errors, or combinations of these errors.

Summary

- Introduction to OSGi technology
- Equinox and the OSGI
- OSGi framework
- Life cycle of a bundle
- The bundle manifest
- Errors associated with bundles



This module provided an introduction to OSGi technology including the OSGi framework, discussion about Equinox, Bundles, and error conditions. To learn more about OSGi technology, visit www.osgi.org.

Now that you have a foundation for both Eclipse and OSGi technology, the last module in this course will cover: ***Using Rational Diagnostic Tool for Eclipse to Diagnose Eclipse 3.2.***

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 e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_RDTE_Module2.ppt



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

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM Rational

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

