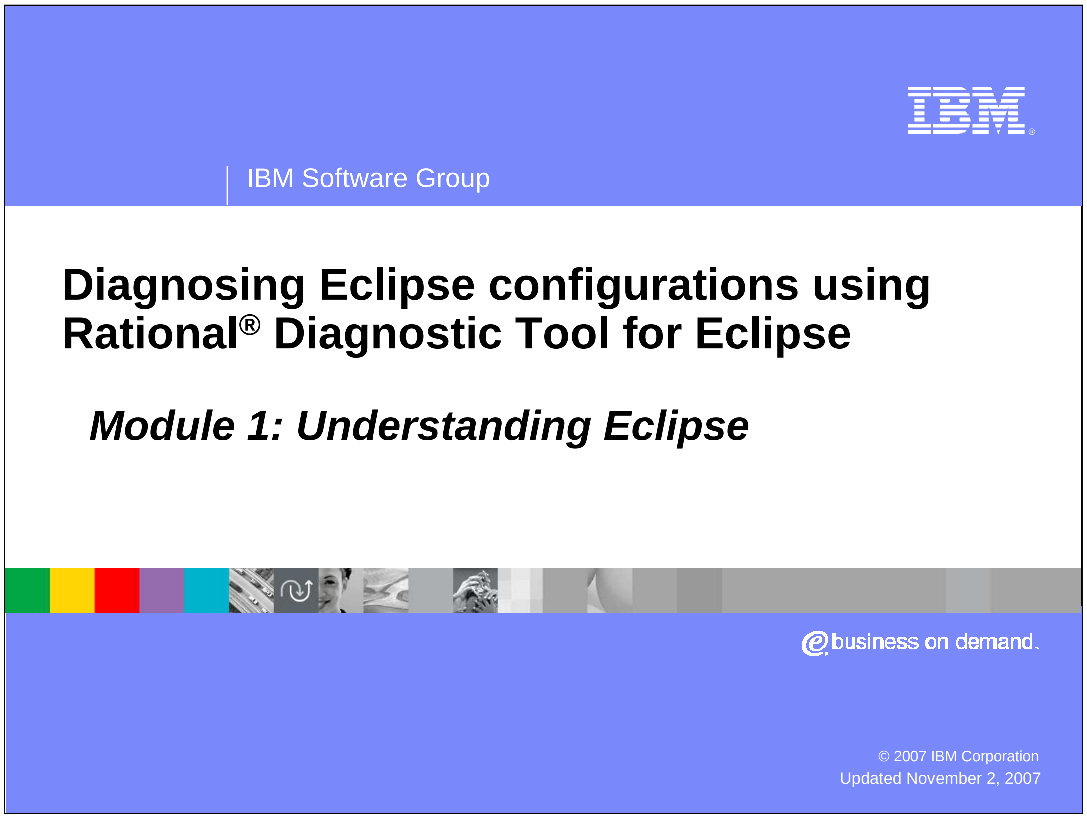


IBM Software Group

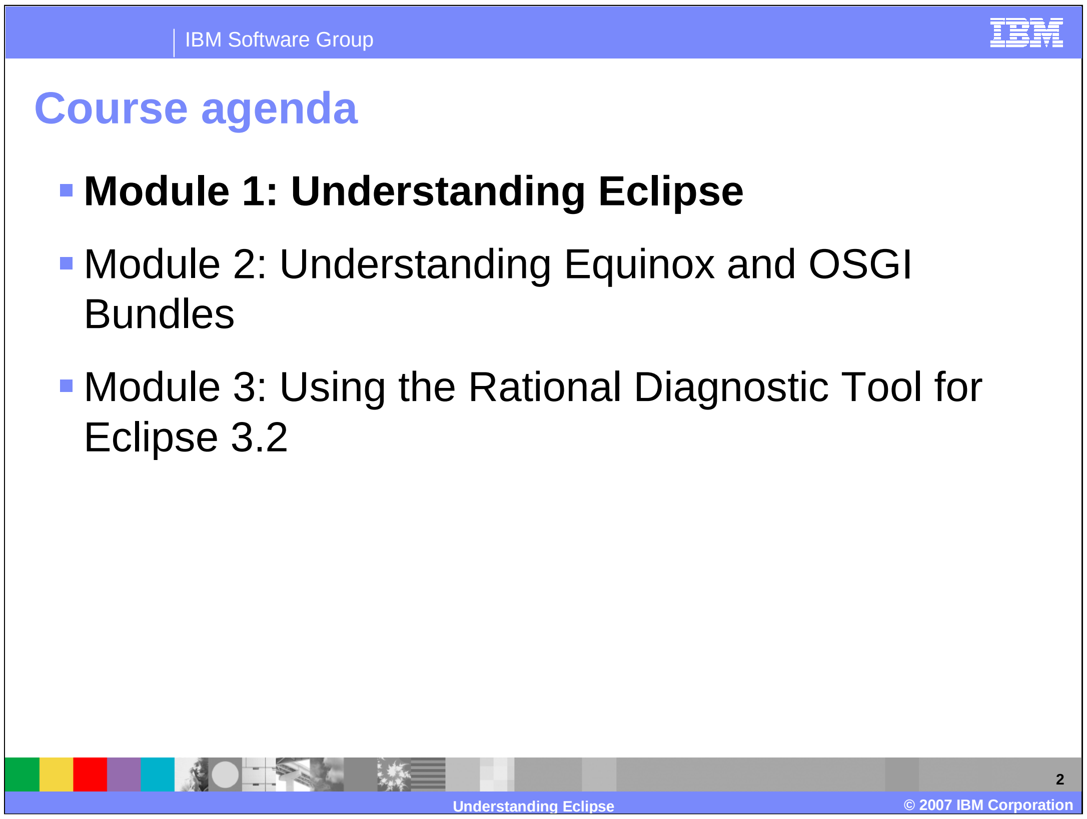
Diagnosing Eclipse configurations using Rational® Diagnostic Tool for Eclipse

Module 1: Understanding Eclipse

business on demand.

© 2007 IBM Corporation
Updated November 2, 2007

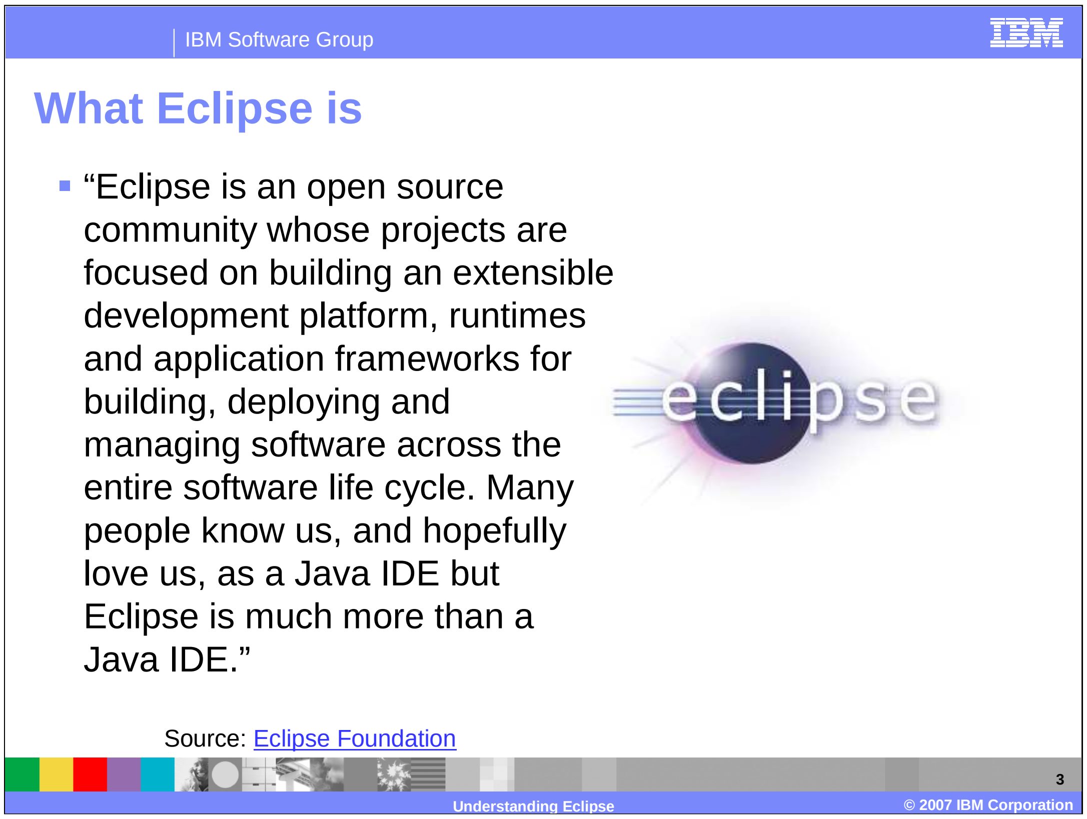
This module will cover Eclipse, which is the platform that the Rational Diagnostic Tool For Eclipse is built upon. The information contained in this module is based on Eclipse version 3.2.



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 series of three modules are to help you understand Eclipse, the Equinox and OSGI Bundles, and diagnosing Eclipse with Rational Diagnostic Tool for Eclipse. Beyond defining what Eclipse is, this module explores Eclipse from three vantage points: 1) exploring the individual components of Eclipse, 2) exploring Eclipse visually, and 3) exploring Eclipse on the file-system.



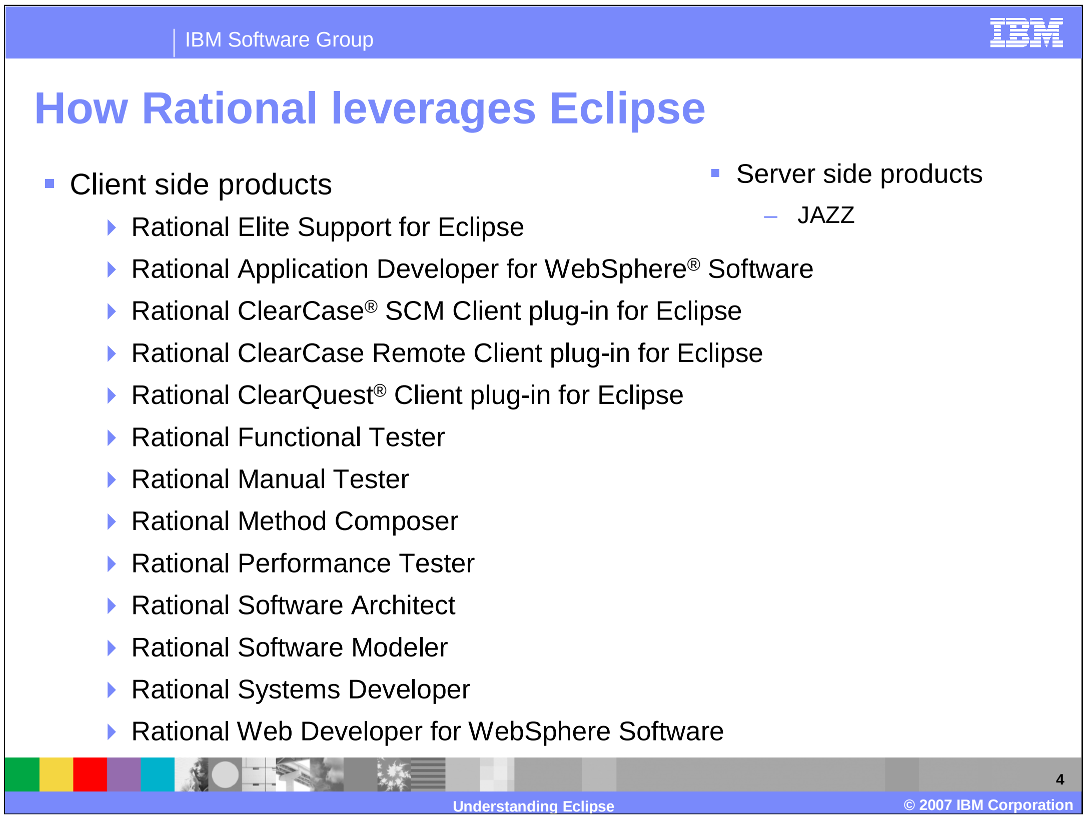
What Eclipse is

- “Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software life cycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE.”

Source: Eclipse Foundation

According to the Eclipse Foundation, “Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software life cycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE.”

The most important fact to understand is that Eclipse is much more than a Java IDE: It is a platform.


IBM Software Group

How Rational leverages Eclipse

- Client side products
  - Rational Elite Support for Eclipse
  - Rational Application Developer for WebSphere® Software
  - Rational ClearCase® SCM Client plug-in for Eclipse
  - Rational ClearCase Remote Client plug-in for Eclipse
  - Rational ClearQuest® Client plug-in for Eclipse
  - Rational Functional Tester
  - Rational Manual Tester
  - Rational Method Composer
  - Rational Performance Tester
  - Rational Software Architect
  - Rational Software Modeler
  - Rational Systems Developer
  - Rational Web Developer for WebSphere Software
- Server side products
  - JAZZ

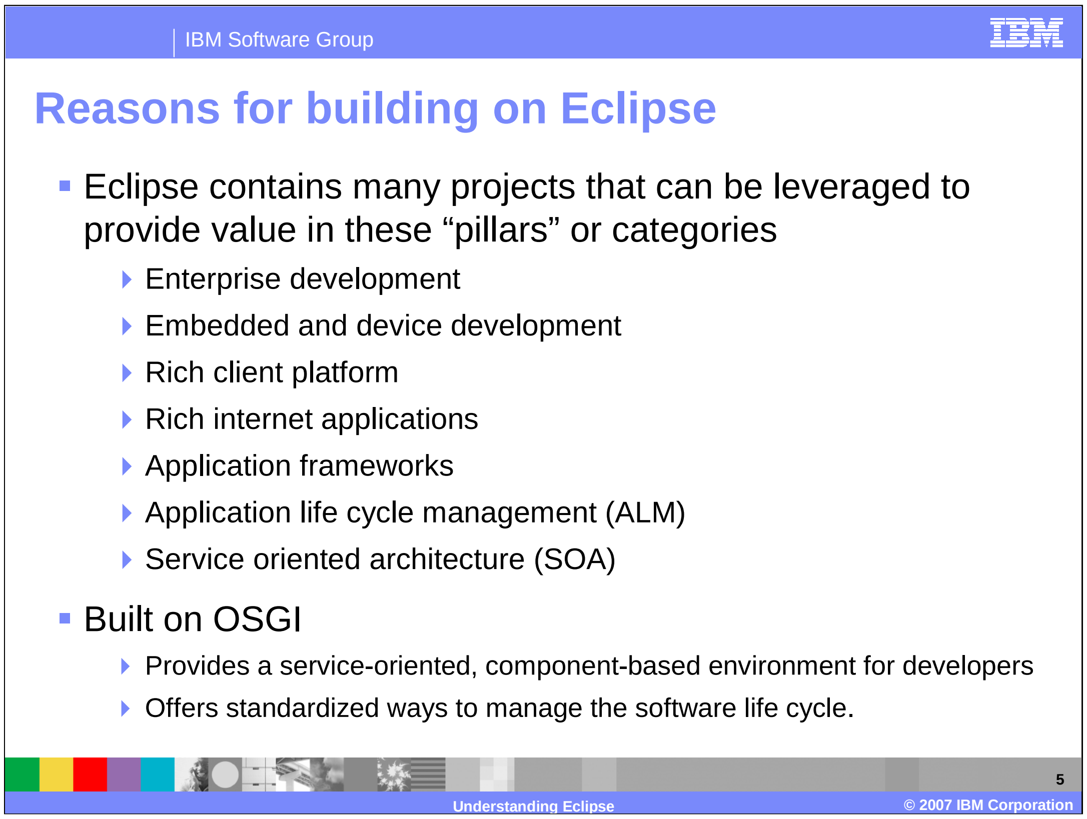
4

Understanding Eclipse

© 2007 IBM Corporation

Rational leverages Eclipse for several products. From the list shown here, you can see that Eclipse is a very flexible platform and is used in nearly the entire Rational offering portfolio on both the client and server sides; from Rational Application Developer to Rational Tester products to JAZZ.

(pause 5 seconds)


IBM Software Group

Reasons for building on Eclipse

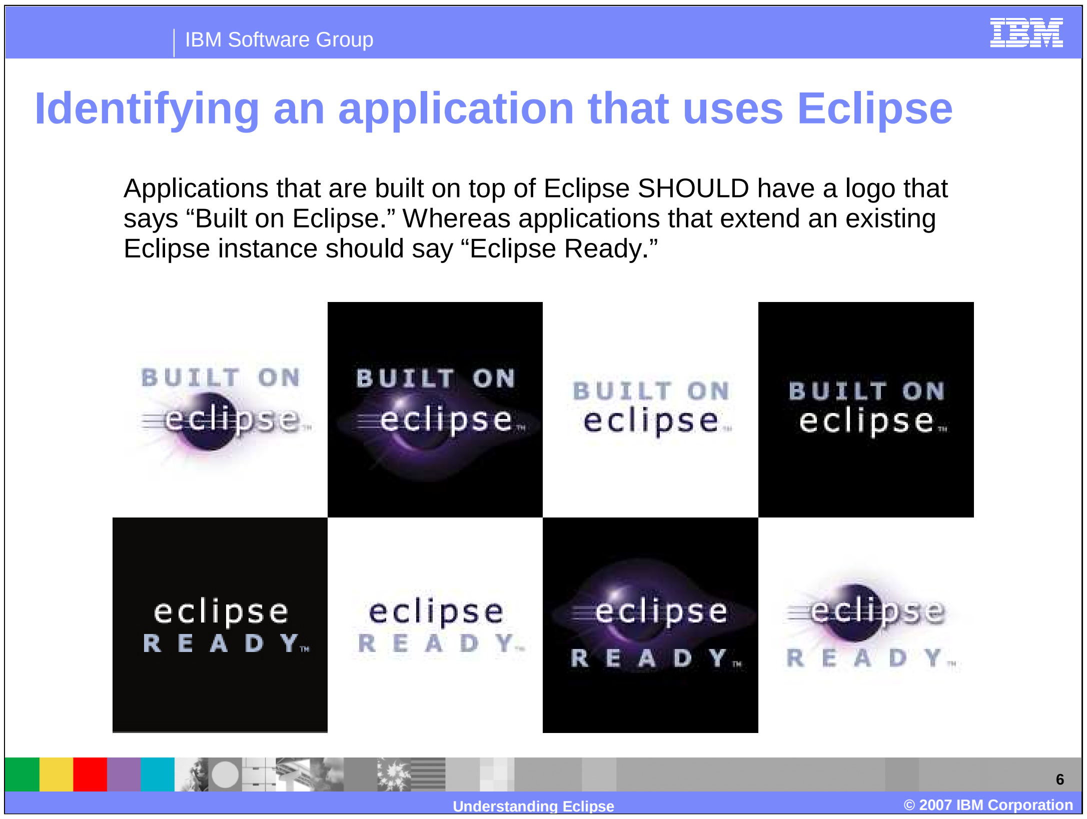
- Eclipse contains many projects that can be leveraged to provide value in these “pillars” or categories
  - Enterprise development
  - Embedded and device development
  - Rich client platform
  - Rich internet applications
  - Application frameworks
  - Application life cycle management (ALM)
  - Service oriented architecture (SOA)
- Built on OSGI
  - Provides a service-oriented, component-based environment for developers
  - Offers standardized ways to manage the software life cycle.

5

Understanding Eclipse © 2007 IBM Corporation

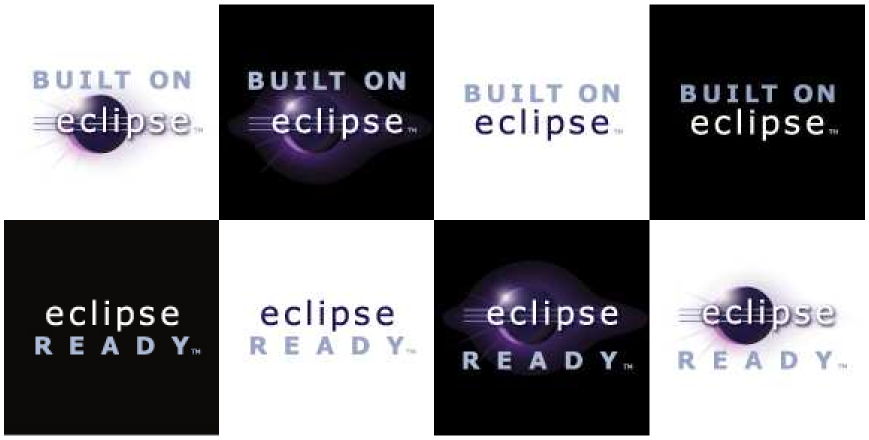
Building Rational products on Eclipse allows building on top of an existing, leading, open-source platform. While projects included in Eclipse differ based on the Eclipse version and packaging, there are many projects that can be leveraged to provide value in enterprise development, embedded and device development, rich client platform, rich internet applications, application frameworks, lifecycle management, and SOA – Service Oriented Architecture.

In addition, Eclipse is built on OSGI – a service-oriented, component-based environment for developers that offers standardized ways to manage the software life cycle. OSGI is covered later in this module.


IBM Software Group

Identifying an application that uses Eclipse

Applications that are built on top of Eclipse SHOULD have a logo that says “Built on Eclipse.” Whereas applications that extend an existing Eclipse instance should say “Eclipse Ready.”



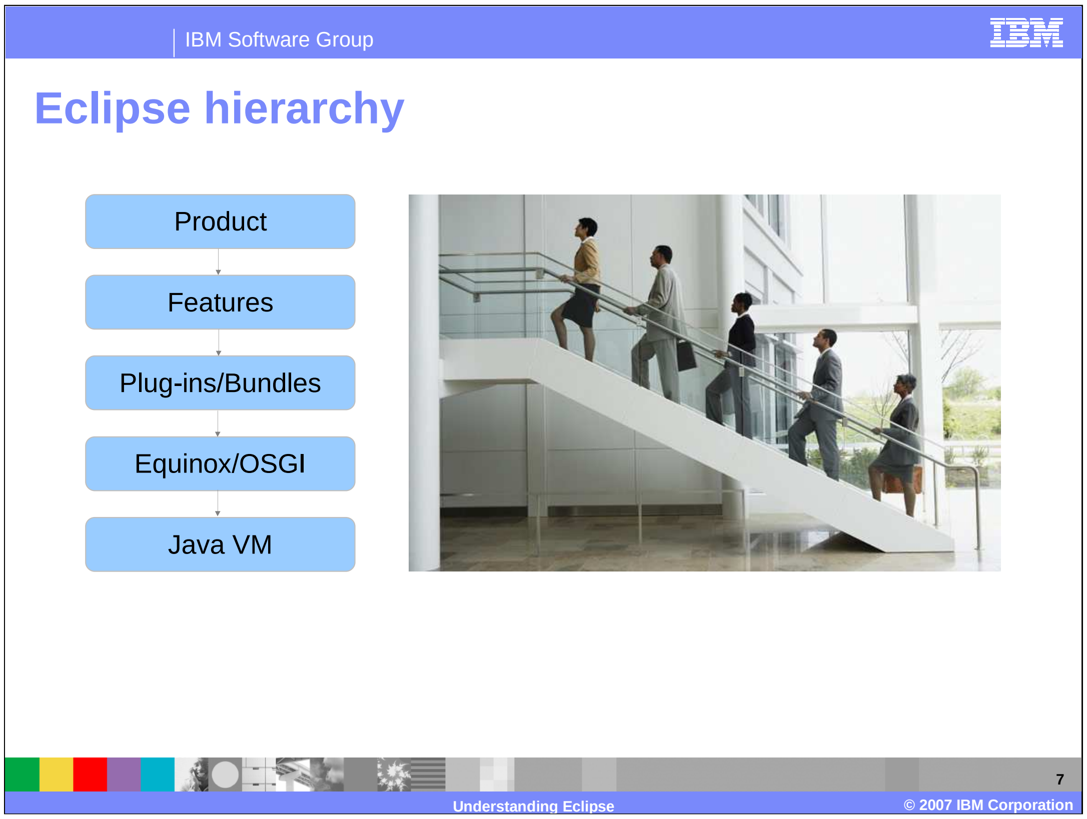
6

Understanding Eclipse

© 2007 IBM Corporation

Not all Eclipse applications look like an IDE but you should be able to spot them by looking for one of these logos.

Applications that are built on top of Eclipse SHOULD have a logo that says “Built on Eclipse.” Whereas applications that merely extend an existing Eclipse Instance should say “Eclipse Ready.”



Eclipse hierarchy

$$\begin{tikzcd}[row sep=small] \text{Product} \arrow[d] \\ \text{Features} \arrow[d] \\ \text{Plug-ins/Bundles} \arrow[d] \\ \text{Equinox/OSGI} \arrow[d] \\ \text{Java VM} \end{tikzcd}$$

The next few slides will cover Eclipse basics, starting with the top-down hierarchy described on this slide.

Here the Product is the top of the Eclipse Hierarchy, but strictly speaking, a product is an extension to the extension point called org.eclipse.core.runtime.products. The purpose of a product is to define application-specific branding on top of a configuration of Eclipse plug-ins. Minimally, a product defines the ID of the application it is associated with and provides a name, description, and unique ID of its own. A product also stores a table of additional properties, where the UI stores information such as the application window icon and the information contained in the *Help > About...* dialog box.

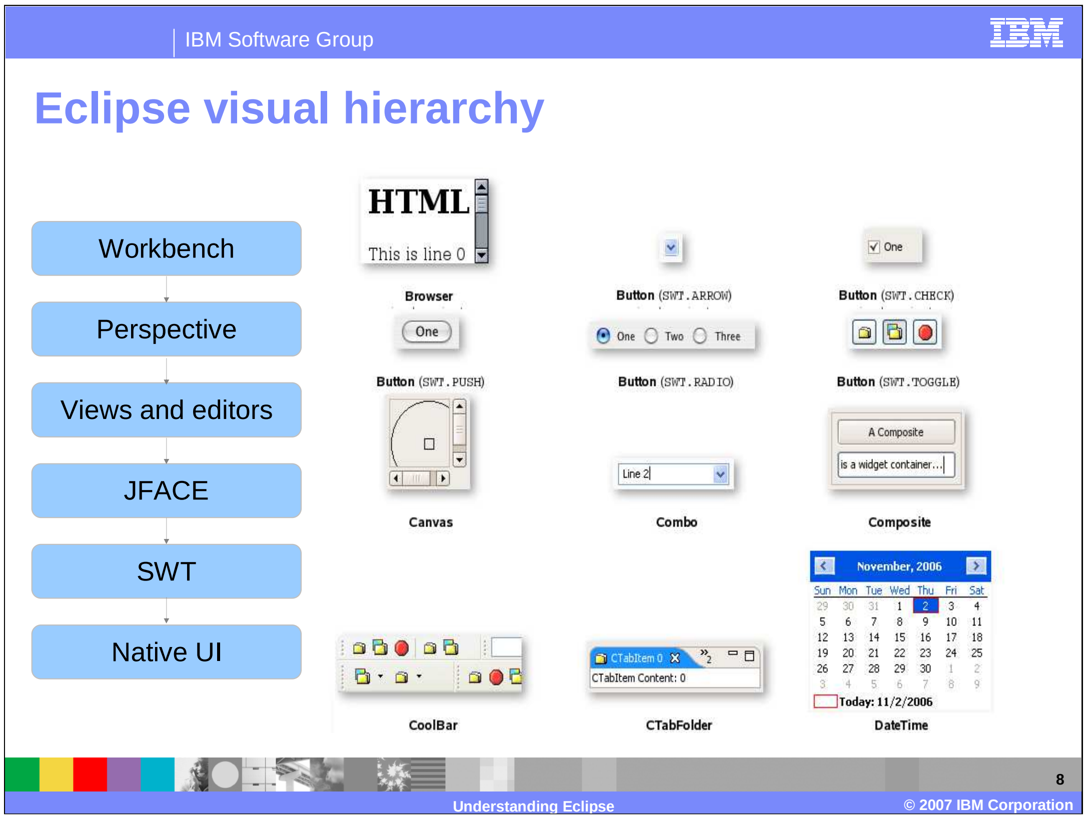
Skipping the Feature level for the moment, plug-ins make up nearly every part of Eclipse. A single Java source file like Main.java is *not* part of a plug-in. This java class is used only to find and invoke the plug-in responsible for starting up the Eclipse Platform. This class will typically in-turn be invoked by a native executable, such as eclipse.exe on Windows, although this is just to hide the incantations required to find and launch a Java virtual machine. In short, just about everything in Eclipse is a plug-in.

A plug-in minimally consists of a plug-in manifest file called plugin.xml. This manifest provides important details about the plug-in, such as its name, ID, and version number. The manifest may also tell the platform what Java code it supplies and what other plug-ins it requires, if any. Note that everything except the basic plug-in description is optional. A plug-in may provide code, or it may provide only documentation, resource bundles, or other data to be used by other plug-ins.

Even though it is not depicted on this slide, fragments [of plug-ins] are used, and are sometimes it is useful to make part of a plug-in optional; allowing it to be installed, uninstalled, or updated independently from the rest of the plug-in. For example, a plug-in may have a library that is specific to a particular operating system or windowing system or a language pack that adds translations for the plug-in’s messages. In these situations, you can create a fragment that is associated with a particular host plug-in. On disk, a fragment looks almost exactly the same as a plug-in, except for a few cosmetic differences such as fragment.xml instead of plugin.xml.

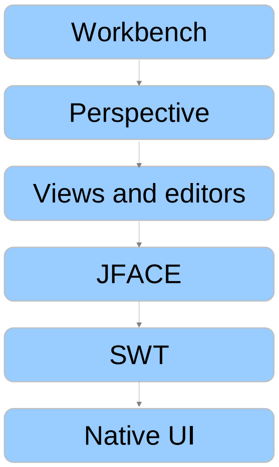
Features are essentially the way Eclipse organizes all of these plug-in bundles (sometimes known just as “Bundles”). Instead of making you go to the update UI and select hundreds of plug-ins, features include and require other features or plug-ins.

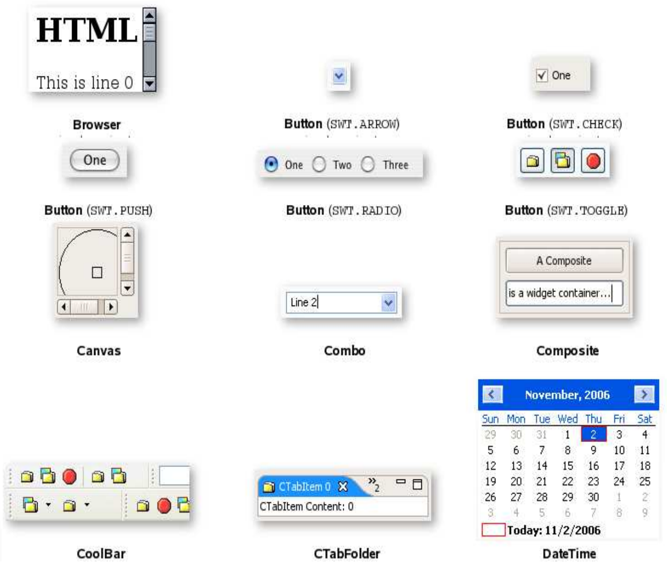
Equinox is the Eclipse implementation of the OSGI Framework. However, it is best to think of it as the Eclipse Kernel. It is the engine that runs Eclipse and it is designed to be a very dynamic engine that allows bundles to start, stop and be aware of other bundles. Of course this all runs on a Java VM.



IBM Software Group

Eclipse visual hierarchy





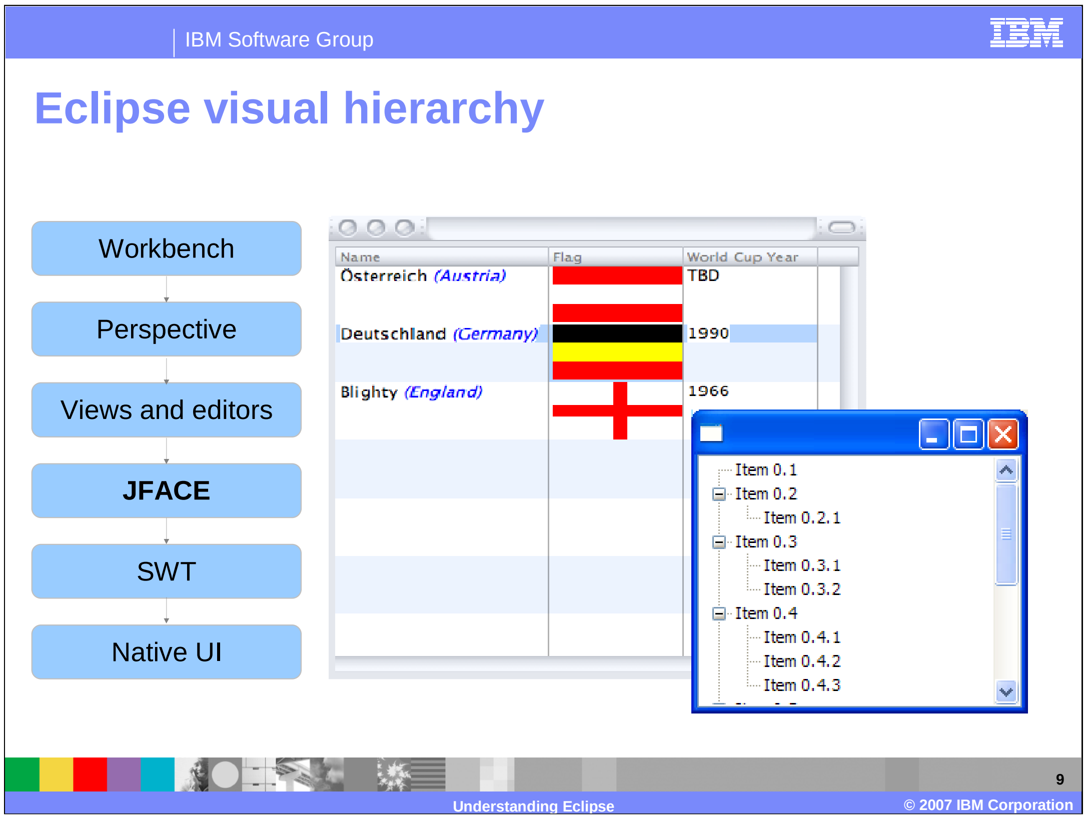
8

Understanding Eclipse

© 2007 IBM Corporation

Visually, the Eclipse IDE has an equally complex hierarchy. Starting from the bottom, you see Native UI and SWT.

Native UI is exactly what it sounds like – the native user interface. Next up the list from the bottom is the SWT. The ***SWT*** is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented. So, a Button in SWT directly relates to a Button in an OS specific widget toolkit such as GTK under Linux, and Carbon under Apple OS X.


IBM Software Group

Eclipse visual hierarchy

9

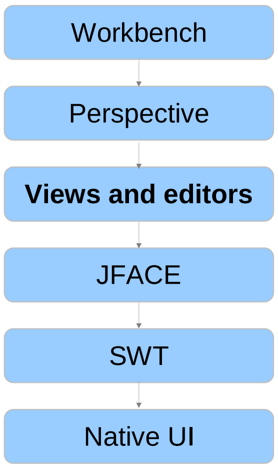
Understanding Eclipse

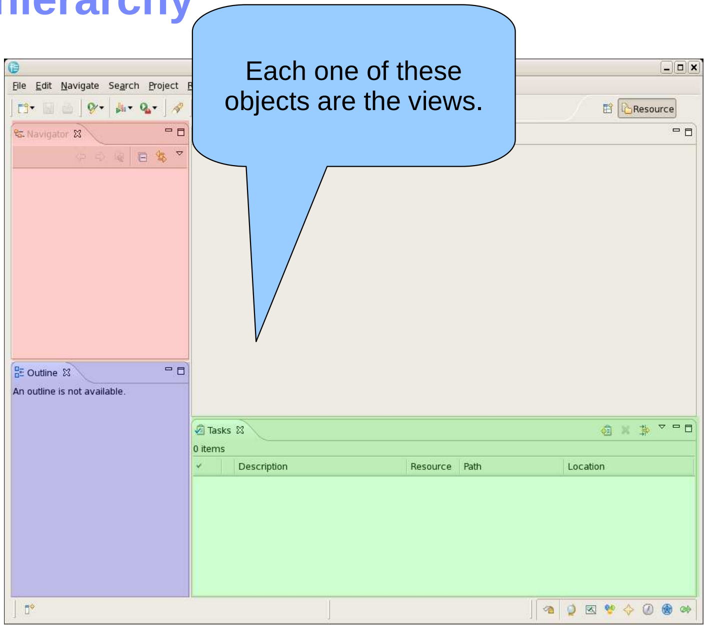
© 2007 IBM Corporation

Next up the list from the bottom is ***JFace***. JFace is a UI toolkit with classes for handling many common UI programming tasks. JFace is window-system-independent in both its API and implementation, and is designed to work with SWT without hiding it.

JFace includes the usual UI toolkit components of image and font registries, text, dialog, preference and wizard frameworks, and progress reporting for long running operations. Two of its more interesting features are actions and viewers. The action mechanism allows user commands to be defined independently from their exact whereabouts in the UI. The viewers are model-based adapters for certain SWT widgets, simplifying the presentation of application data structured as lists, tables or trees.

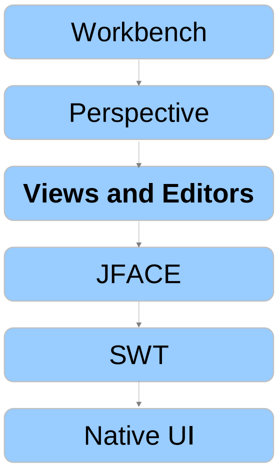
Eclipse visual hierarchy

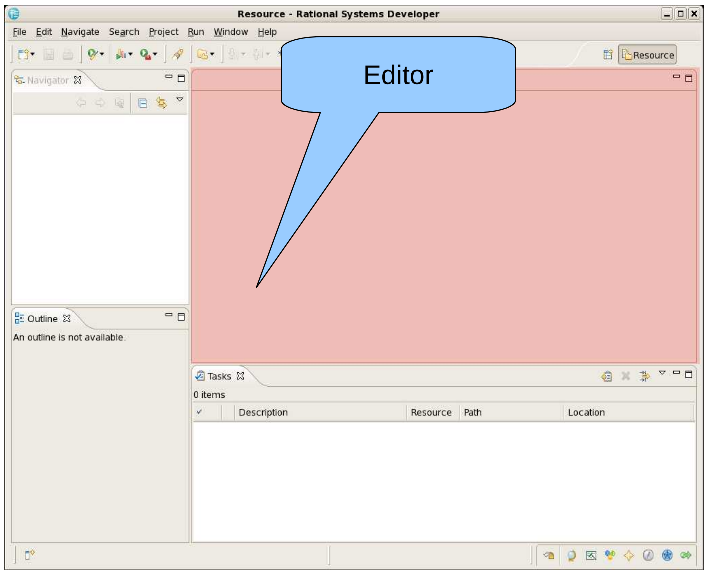



Each one of these objects are the views.
File Edit Navigate Search Project
Resource
Navigator
Outline
An outline is not available.
Tasks
0 items
Description
Resource
Path
Location

Next up, Views. ***Views*** are one of the two kinds of parts that make up a workbench window. At their most basic, views are a subclass of the SWT Composite class, containing arbitrary controls below a title bar. The title bar contains the view name, an area for toolbar buttons, and one or two drop-down menus. The drop-down menu on the upper left is the standard shell menu with actions for moving, resizing, and closing the view. The menu on the upper right and the button area are the view's action bar and may contain arbitrary actions defined by the implementer of that view.

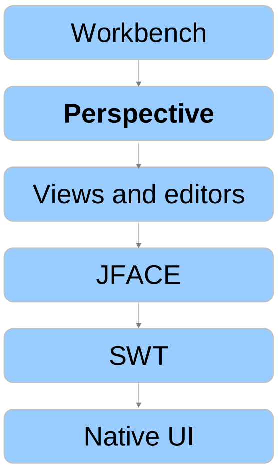
Eclipse visual hierarchy

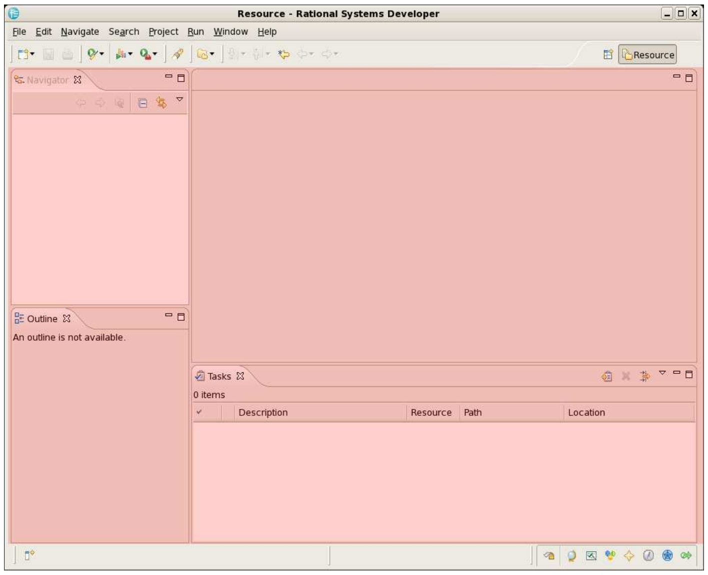



Resource - Rational Systems Developer
File Edit Navigate Search Project Run Window Help
Resource
Navigator
Editor
Outline
An outline is not available.
Tasks
0 items
Description
Resource
Path
Location

Editors are very similar to views but different. Editors can appear in only one region of the page, whereas views can be moved to any part of the page and minimized as fast views. Editors can be in a dirty state, meaning that their contents are unsaved and will be lost if the editor is closed without saving. Views have a local toolbar, whereas editors contribute buttons to the global toolbar. Editors can be associated with a file name or an extension, and this association can be changed by users.

Eclipse visual hierarchy




Resource - Rational Systems Developer
File Edit Navigate Search Project Run Window Help
Resource
Navigator
Outline
Tasks

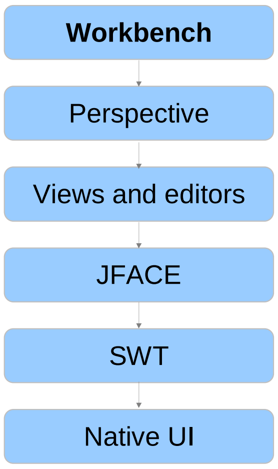
An outline is not available.

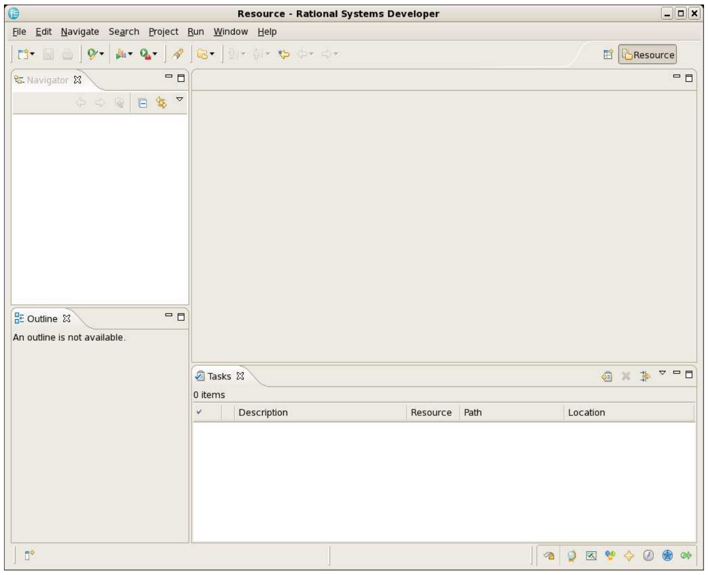
0 items

Description Resource Path Location

The perspective is a collection of views arranged in a specific manner; it is another level of organization. Note that just because a view is in a perspective, it does not mean that the view cannot be shared among many perspectives.

Eclipse visual hierarchy



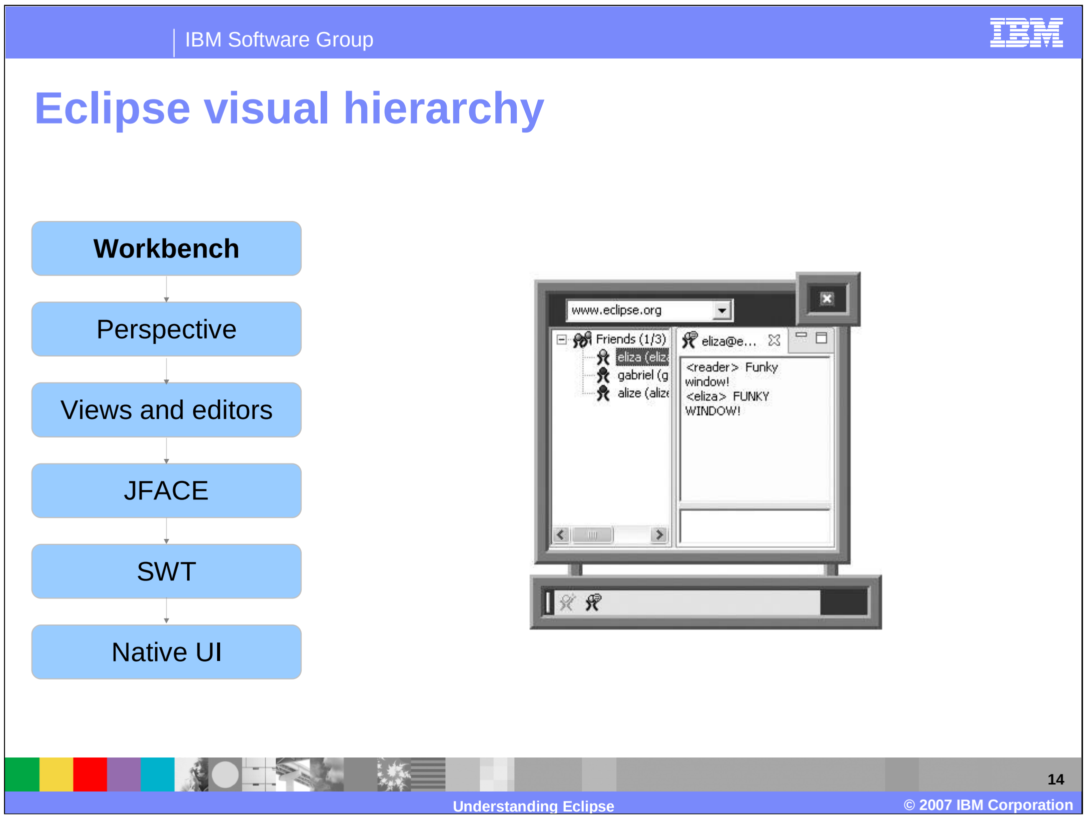

Resource - Rational Systems Developer
File Edit Navigate Search Project Run Window Help
Resource
Navigator
Outline
Tasks
Description
Resource
Path
Location

An outline is not available.

0 items

© 2007 IBM Corporation

At the top of the hierarchy, the Workbench provides the user interface structure for Eclipse. The purpose of the Workbench is to facilitate the seamless integration of tools. These tools contribute to extension points defined by the Workbench. The Workbench is responsible for the presentation and coordination of the user interface. The tools metaphor is not specific to development tools, but can apply equally well to arbitrary applications. Note that the Workbench is sometimes called the Generic Workbench or Eclipse Shell, to distinguish it from its instantiation in the Eclipse IDE, where it is called the IDE Workbench.



IBM Software Group

Eclipse visual hierarchy

Workbench

Perspective

Views and editors

JFACE

SWT

Native UI

www.eclipse.org
Friends (1/3)
eliza (eliz
gabriel (g
alize (alize
eliza@e...

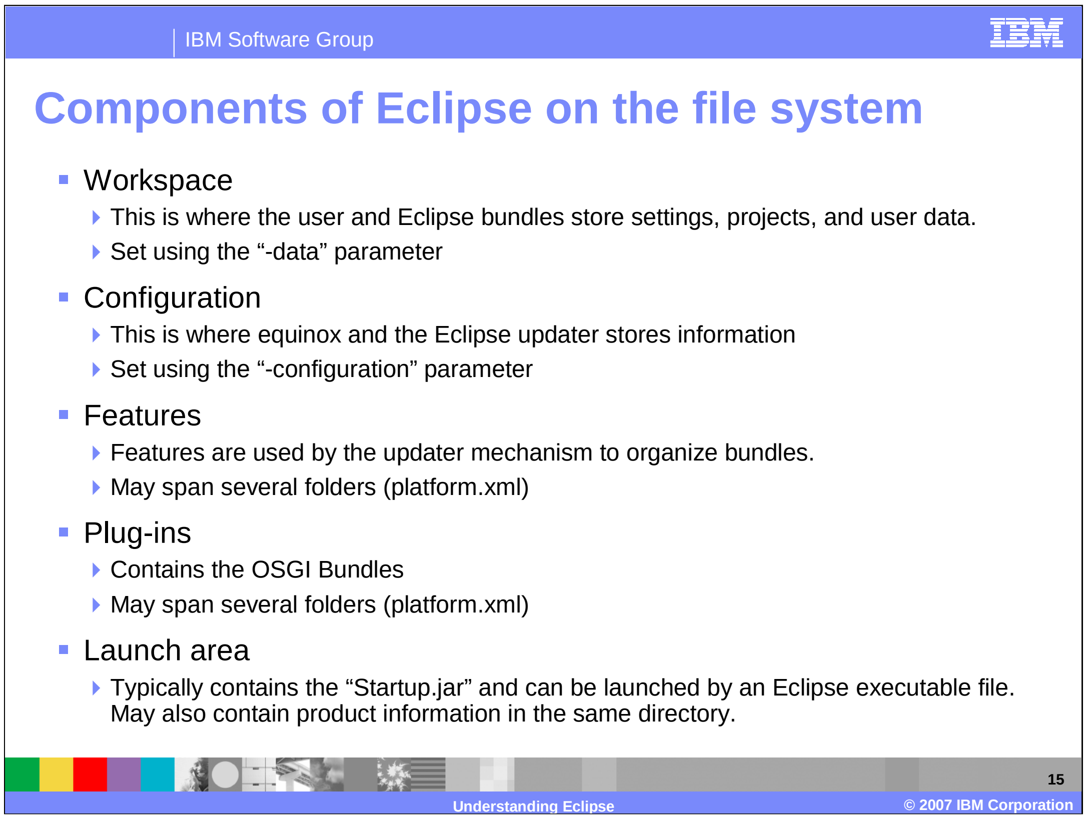
<reader> Funky window!
<eliza> FUNKY WINDOW!

14

Understanding Eclipse

© 2007 IBM Corporation

Note that the workbench can take different forms. This example is from Addison Wesley's Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications.


IBM Software Group

Components of Eclipse on the file system

- Workspace
  - This is where the user and Eclipse bundles store settings, projects, and user data.
  - Set using the "-data" parameter
- Configuration
  - This is where equinox and the Eclipse updater stores information
  - Set using the "-configuration" parameter
- Features
  - Features are used by the updater mechanism to organize bundles.
  - May span several folders (platform.xml)
- Plug-ins
  - Contains the OSGI Bundles
  - May span several folders (platform.xml)
- Launch area
  - Typically contains the "Startup.jar" and can be launched by an Eclipse executable file. May also contain product information in the same directory.

15

Understanding Eclipse © 2007 IBM Corporation

There are five main components of Eclipse on the File System: The workspace, configuration, features, plug-ins, and the launch area.

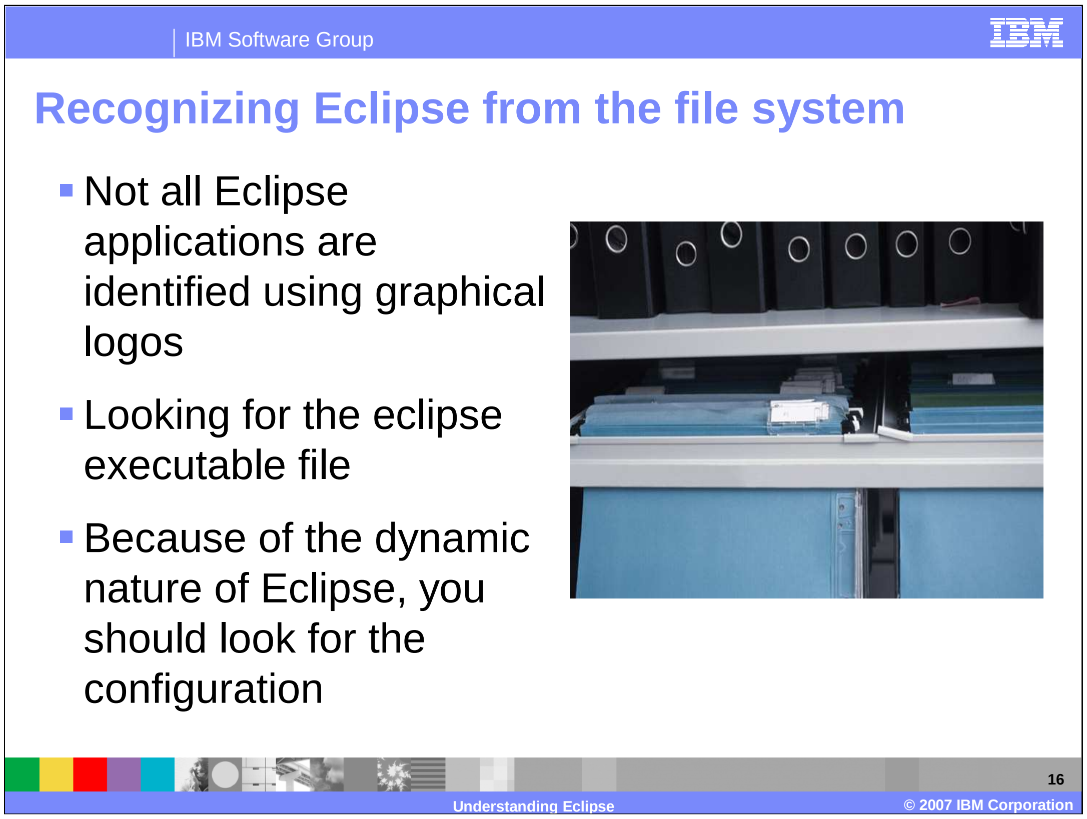
The ***workspace*** is where the user and Eclipse bundles store settings, projects, and user data.

The ***configuration*** is the set of plug-ins available in a particular instance of the Eclipse Platform. A given installation of Eclipse may contain hundreds or even thousands of plug-ins. More than one Eclipse-based application can share this same install location, but they don't always want to use all the same plug-ins. When Eclipse is started, a configurator determines what subset of the installed pool of plug-ins will be used for that particular instance of the platform. By default, all installed plug-ins will be in the configuration, but a configuration can be customized to contain different groups of plug-ins. You can go to *Help > Software Updates > Manage Configuration* to see and modify what plug-ins are in your configuration.

Because more than one product can be installed at a given time, the main product is singled out in a special marker file called *.eclipseproduct* in the Eclipse install directory. This file denotes the name, ID, and version number of the main product that will be used. The product in turn is a plug-in in the plugins directory, which includes product branding elements, such as the splash screen and workbench window icons.

Features and ***plug-ins*** contains jars/folders with the features/plug-in files. Note that the Plug-in folder does not contain a hyphen '-' in the directory name.

Finally, the ***launch area*** typically contains an executable file. This small launcher essentially finds and loads the JVM that is on your PATH. In Eclipse 3.2 and earlier can contain a startup.jar. For Eclipse 3.3, the Equinox launcher will be in the plug-in directory. You may also see a .eclipseproduct file with product information.



IBM Software Group

Recognizing Eclipse from the file system

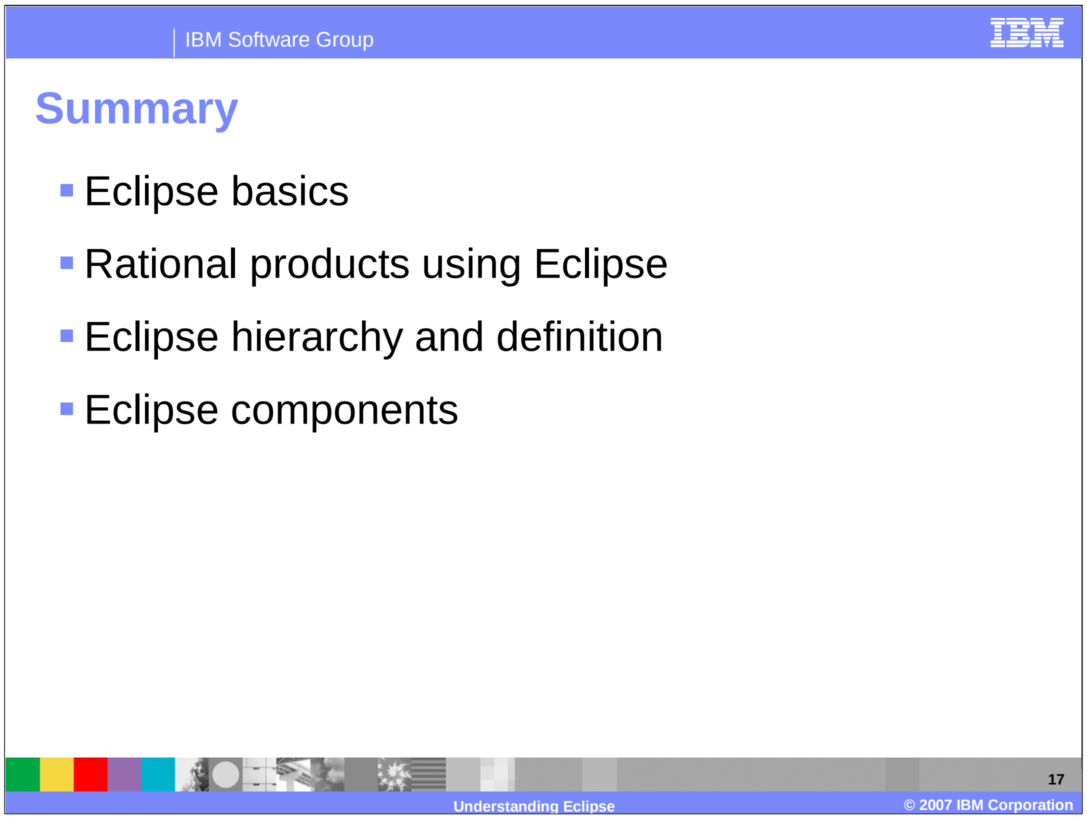
- Not all Eclipse applications are identified using graphical logos
- Looking for the eclipse executable file
- Because of the dynamic nature of Eclipse, you should look for the configuration

16

Understanding Eclipse

© 2007 IBM Corporation

While you can look for the graphical logo that denotes an Eclipse application in a previous slide, not all Eclipse applications are identified using graphical logos. Some applications that are command line or headless do not indicate that they are based on Eclipse. Looking for the Eclipse executable file is also not reliable, because it may have been renamed and not contain the word Eclipse at all. Because of the dynamic nature of Eclipse, in the worst case scenario, look for the configuration.

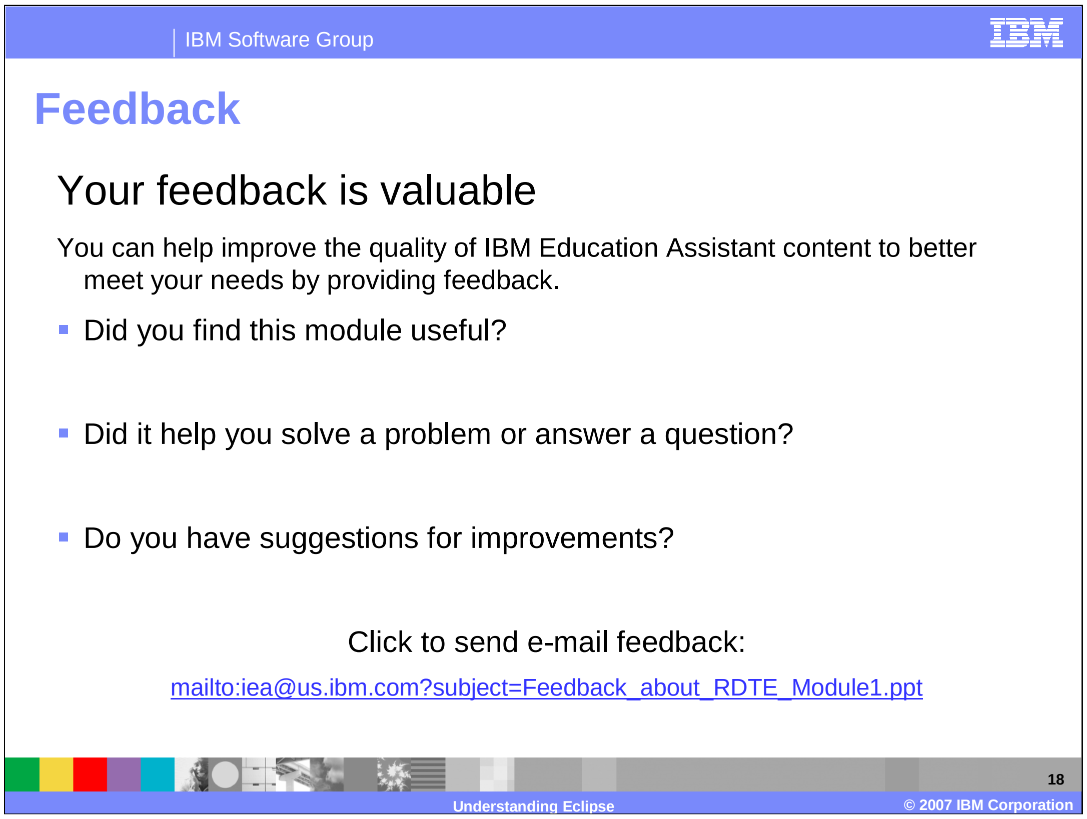


Summary

- Eclipse basics
- Rational products using Eclipse
- Eclipse hierarchy and definition
- Eclipse components

© 2007 IBM Corporation

In summary, this module covered a number of Eclipse basics including: definition, logos, hierarchy and its sub-parts, graphical hierarchy, and Eclipse components. The next module will cover the introduction the OSGI technology.



Feedback

Your feedback is valuable

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

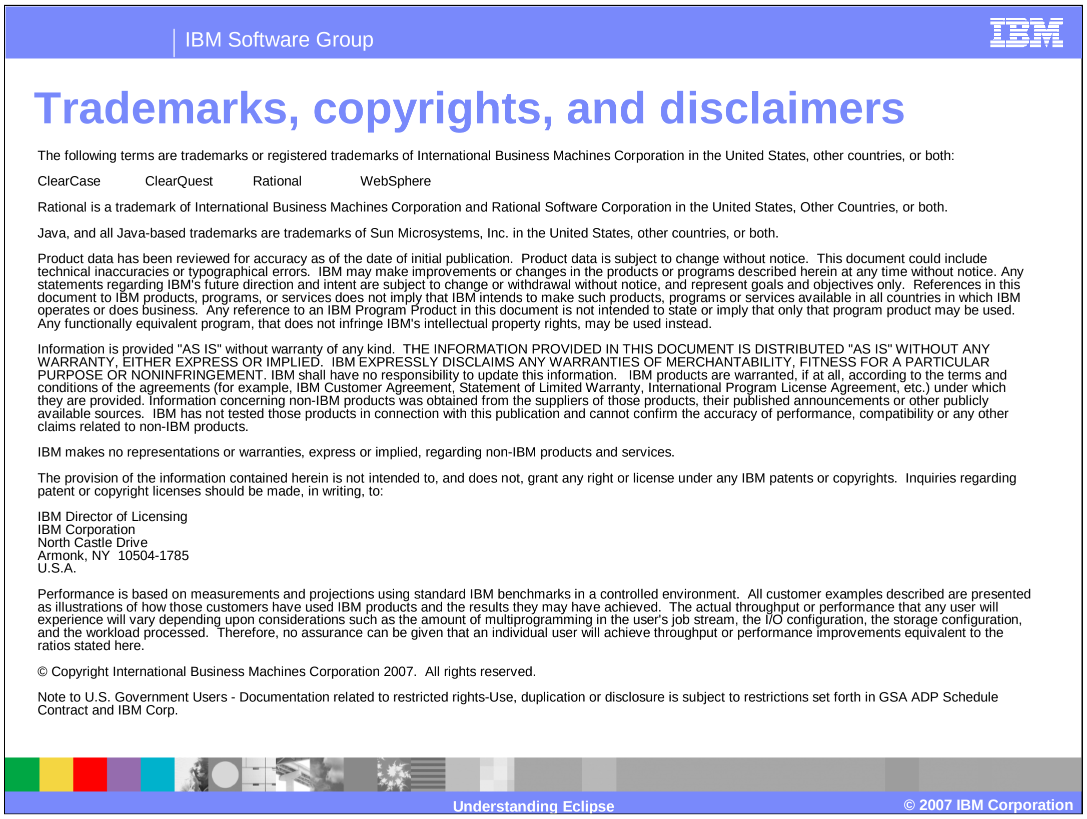
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_RDTE_Module1.ppt

© 2007 IBM Corporation

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:

ClearCase ClearQuest Rational WebSphere

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

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.