Tutorial III. Eclipse

Outline

- Basics
- Eclipse Plug-in feature, MVC
- How to build Plug-ins
- Exploring Eclipse source code for Editor
- Using CVS inside Eclipse
- Eclipse JDK Tips

Basics

- Eclipse projects:
 - Eclipse platform
 - Plugin architecture
 - Platform, JDT, PDT
 - A number of integrated plugins: JUNIT, CVS, etc.
 - Eclipse tools project
 - CDT, VE, AspectJ, Hipikat
- The official website of Eclipse: http://www.eclipse.org
- Eclipse forums
- Articles
- Eclipse plugins repository
- Eclipse bugzilla
- Using Eclipse in CDF: >setenv LD_LIBRARY_PATH /local/lib/eclipse (".cshrc") >eclipse

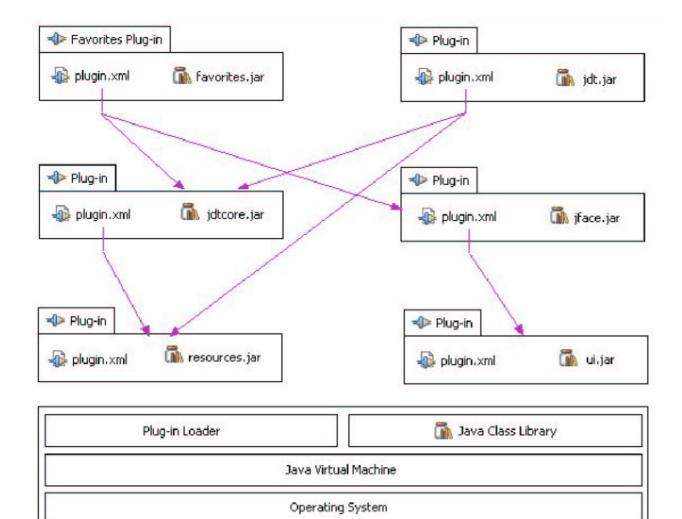
Eclipse Plug-in Feature

• Eclipse =

a core runtime engine + a set of plug-ins

- Plug-in: the smallest extensible unit to contribute additional functions to the system.
- Extension point: boundaries between plug-ins

Eclipse Plug-in Structure



Plug-in Manifest file (plugin.xml)

≩ plugin.xml		🚮 favorites.jar	
<pre>(?xml version="1.0" encoding="UTF-8"?> cplugin id="com.qualityeclipse.favorites"></pre>		com.qualityee G Favorites	lipse.fa¤orites.view View
<runtime></runtime>			
library name="favorites.jar"/>			
<requires></requires>			
<pre><import plugin="org.eclipse.core.resources"></import></pre>			
<import plugin="org.eclipse.ui.views"></import>			
<extension point="org.eclipse.ii.views"></extension>			
<category< td=""><td></td><td></td><td></td></category<>			
name="Quality Eclipse"			
id="com.qualityeclipse.favorites">			
<view< td=""><td></td><td></td><td></td></view<>			
name="Favorites"			
icon="icons/sample.gif"			
<pre>category="com.qualityeclipse.favorites" class="com.qualityeclipse.favorites.views.Fa</pre>			
id="com.qualityeclipse.favorites.views.Favor			
<pre></pre>	ICEDVIEW 72		
(/EXCENSION)		-	
org.eclipse.ui	org.eclipse.core	. resources	1
pilugin .xml			
<extension-point id="views"></extension-point>			

Plug-in Lifecycle

- Plug-in registry
- Lazy loading
- Unfortunately, never unloaded
- Equinox project (www.eclipse.org/equinox)

How to build Plug-Ins

- Plug-ins contribute functionality to the platform by contributing to pre-defined extension points.
- The platform has a well-defined set of extension points - places where you can hook into the platform and contribute system behavior.

How to build Plug-Ins (cond')

- 1. Decide how your plug-in will be integrated with the platform.
- 2. Identify the extension points that you need to contribute in order to integrate your plug-in.
- 3. Implement these extensions according to the specification for the extension points.
- Provide a manifest file (plugin.xml) that describes the extensions you are providing and the packaging of your code.

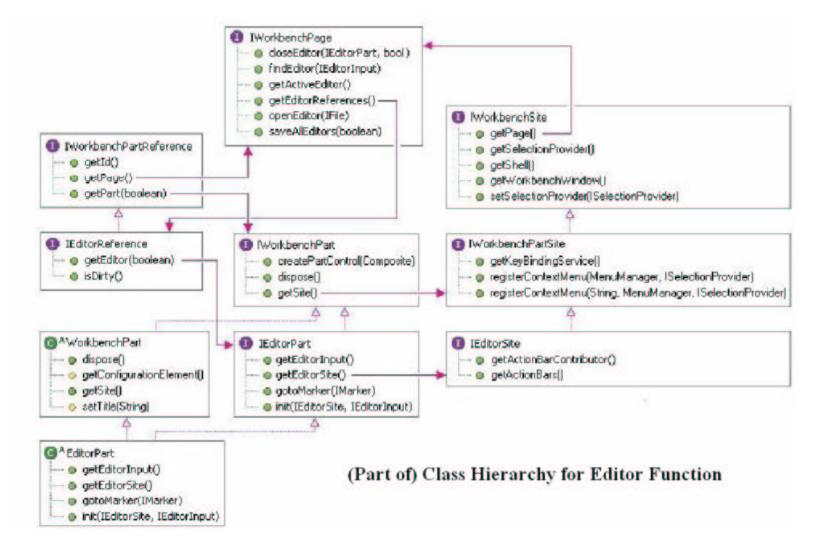
Customized Editor

- An editor is a workbench part that allows a user to edit an object (often a file). An editor is always associated with an input object (IEditorInput).
- The interface for editors is defined in <u>IEditorPart</u>, but plug-ins can choose to extend the <u>EditorPart</u> class rather than implement an <u>IEditorPart</u> from scratch.

Using plug-in to enhance existing editors

- The workbench defines extension points that allow plug-ins to contribute behaviors to existing editors or to provide implementations for new editors.
- The workbench extension point <u>org.eclipse.ui.editors</u> is used by plug-ins to add editors to the workbench.
- Plug-ins that contribute an editor must register the editor extension in their **plugin.xml** file, along with configuration information for the editor.
- Editors can also define a contributorClass, which is a class that adds actions to workbench menus and tool bars when the editor is active

Exploring Eclipse Source Code



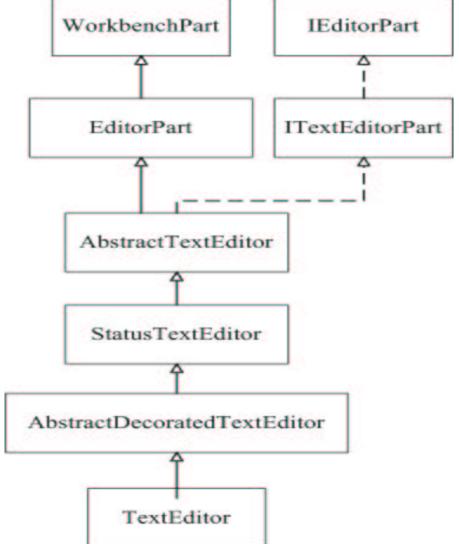
Class hierarchy for Text Editor

ITextEditor is defined as a text specific extension of **IEditorPart**.

The implementation of **ITextEditor** in the platform is structured in layers:

AbstractTextEditor defines the framework for extending the editor to support source code style editing of text. This framework is defined in org.eclipse.ui.texteditor.

The concrete implementation class **TextEditor** defines the behavior for the standard platform text editor. It is defined in the package **org.eclipse.ui.editors.text**.



A good example

- The text editor framework provides a model-independent editor that supports the following features:
 - presentation and user modification of text
 - standard text editing operations such as cut/copy/paste, find/replace
 - support for context and pulldown menus
 - syntax highlighting
 - content assist

.

- key binding contexts
- Exploring how these features can be implemented in an editor by studying the **org.eclipse.ui.examples.javaeditor** example.

CVS (Concurrent Versions System)

- Help support and enhance the process of managing source code in two major ways:
 - by **controlling access** to the source code, using a locking system to serialize access
- by **keeping a history** of the changes made to every file.

Using CVS inside Eclipse

CVS repository parameters for CDF: CVS Server: werewolf or seawolf Repository Path: /u/yijun/cvsroot/c408h001 Connection Type: extssh, NOT pserver

Step 1: Creating a repository location

1. Using the Window > Open Perspective > Other command to open CVS Repository Exploring Perspective.

2. Right-click within the CVS
Repositories view and select the
New > Repository Location
command from the context menu.

New New Refresh Branches Copy to Clipboard	CVS Annotate 🕅 CVS Repo	ositories 🗙 📃 🗖
Refresh Branches Date Tag Copy to Clipboard Image: Copy to Clipboard Image: Copy to Clipboard Image: Copy to Clipboard Image: Copy to Clipboard Image: Copy to Clipboard		è 🗘 🕹 🖗 🖪 🔡 💌
Copy to Clipboard	New 🔸	🔡 Repository Location
Refresh View	Refresh Branches	🔁 Date Tag
	Copy to Clipboard	
	🔗 Refresh View	
Discard location	Discard location	
Properties	Properties	

Step 1: Creating a repository location (cond')

- Specify the address of CVS host;
- 2. Specify the **location** of your repository;
- 3. Enter your login information;
- 4. Select Connection Type;
- Check 'Save Password' (Optional);
- 6. Click 'Finish' button.

ALL OTO D		
Add CVS Repo		
dd a new CVS Rep o Add a new CVS Repo	ository to the CVS Repositories view	s
Location		
Host:	werewolf	•
Repository path:	/u/yijun/cvsroot/c408h001	•
Authentication		
User: yuye		•
Password: ***		
Connection		
Connection type:	extssh	•
. Use Default P	ort	
C Use Port:		
🗸 <u>V</u> alidate Connec	tion on Finish	
Zave Password		
A Saved passwor not impossible	ds are stored on your computer in a file that's difficult, but e, for an intruder to read	Ľ
	Einish Cancel	

Step 2: Share a project

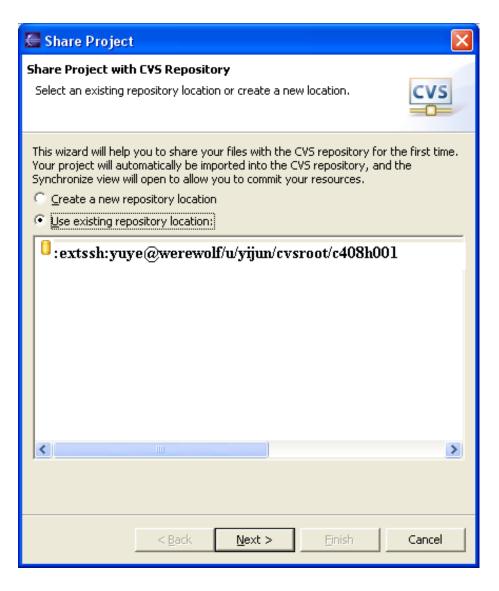
Pacl

- In the Navigator view select the project
 SampleProject.
- 2. From the project's context menu choose Team > Share Project.

Proj	New		•	
	Go Into			
	Open in New Window			
	Сору	Ctrl+C		
	Taste.	Ctrl+V		
	X Delete	Delete		
	Refactor	Alt+Shift+T	•	
	mport			
	Z Export			
	🛷 Refresh	F5		
	Cloge Project			
	Bun		•	
	Debug		<u>•</u> II	
	T <u>e</u> am		•	Apply Patch
	Compare With			Share Project.

Step 2: Share a project (cond')

 In the sharing wizard page, select the location that was previously created.



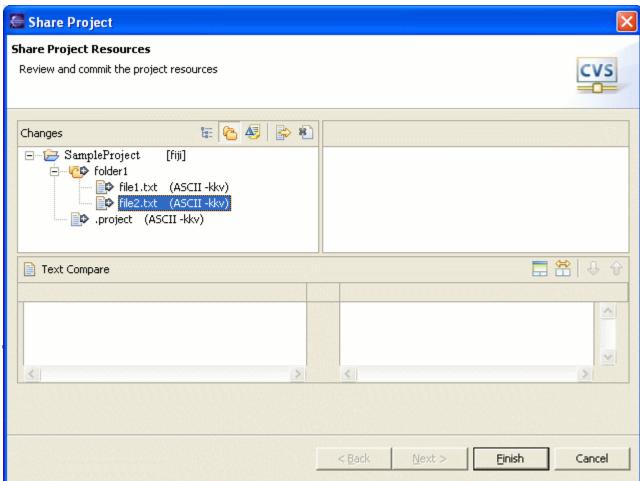
Step 2: Share a project (cond')

 Specify the module name to create on the server. Simply use the default value and use the name of the project you are sharing. Click Next.

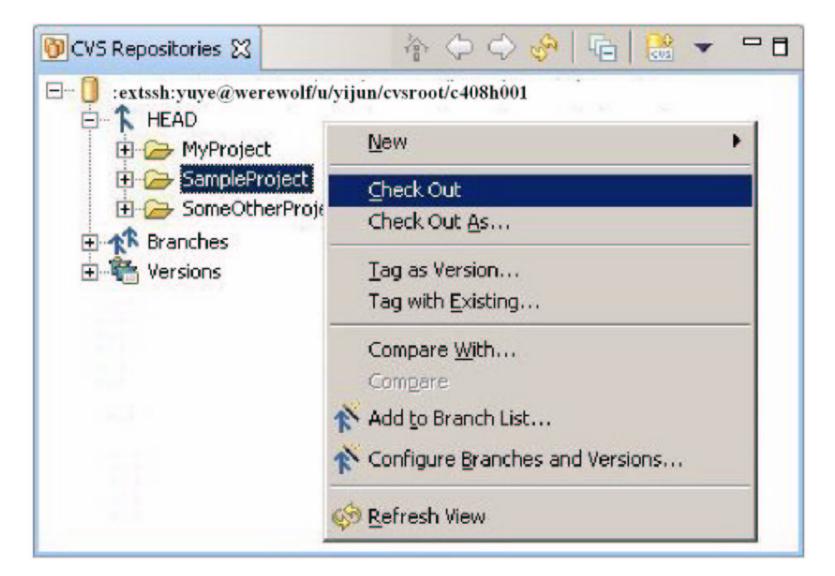
🚰 Share Project				X
Enter Module Name				
Select the name of the m	odule in the CV	5 repository.		CVS
• Use project name as i	module name			
C Use specified module	name:			
C Use an exisiting modu	le (this will allow	you to browse I	the modules in th	ne repository)
	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

Step 2: Share a project (cond')

- This page will allow you to see the files that are going to be shared with your team. The arrows with the plus sign show that the files are new outgoing additions.
- Click 'Finish' button



Check out a project from CVS



Step 3: Synchronize with the repository

Right-click on the resource (or the project containing the resource) and select the **Team > Synchronize with Repository** command.

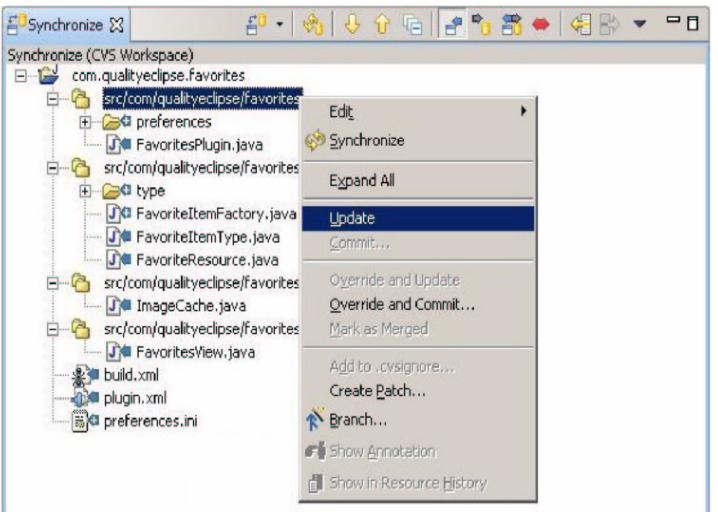
Synchronize with Repository.... Commit... Update Create Patch... Apply Patch... Tag as Version... Branch... Merge.... Change ASCII/Binary Property... Show Editors Restore from Repository... Share Project Disconnect...

Step 4: update/commit the changes

The Incoming Mode causes the view to only show incoming changes.

The **Outgoing Mode** causes the view to only show outgoing changes

The Incoming/ Outgoing Mode will show changes in both sides.

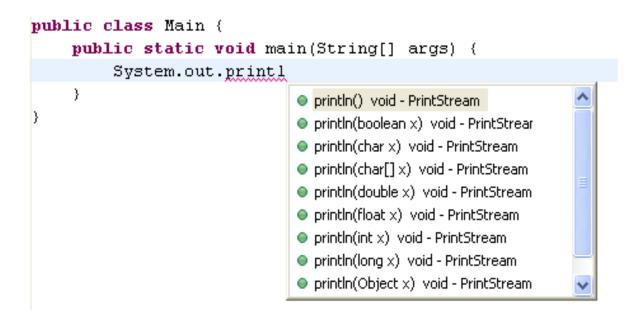


.cvsignore

Eclipse JDK Tips - Content Assist

 Content assist provides you with a list of suggested completions for partially entered strings.

Ctrl+Space or Edit > Content Assist

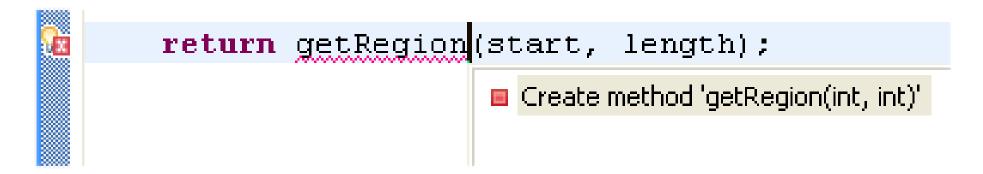


Eclipse JDK Tips - Parameter Hints

- With the cursor in a method argument, you can see a list of parameter hints.
- Ctrl+Shift+Space or Edit > Parameter Hints.
 - if (moveCursor) {
 int selectionOffset, int selectionLength
 setSelectedRange(start, 0);
 revealRange(start, length);
 }

Eclipse JDK Tips - Quick Fix

- Start with the method invocation and use
 Quick Fix to create the method.
- Ctrl+1



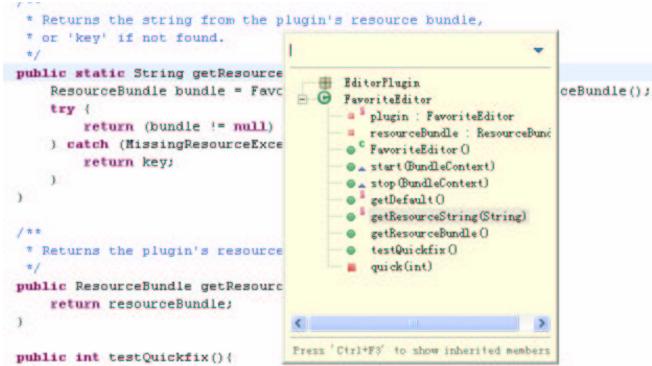
Eclipse JDK Tips – Code Navigation

- There are two ways that you can open an element from its reference in the Java editor.
- 1.Select the reference in the code and press F3 (Navigate > Open Declaration)
- 2.Hold **Ctrl** and move the mouse pointer over the reference.

MessageDialog.openError(fShell, title, message);
return;

Eclipse JDK Tips – In-place outlines

 Press Ctrl+F3 in the Java editor to pop up an in-place outline of the element at the current cursor position.



Eclipse JDK Tips – In-place hierarchy

 Place the cursor inside the method call and press Ctrl+T. The view shows all types that implement the method with a full icon.

ypes imple	menting or defining 'Control.setRedraw(boolean)'
	ject - java.lang A uudaata aya adaasa aya yidaata
	Widget - org.eclipse.swt.widgets
-	Control - org.eclipse.swt.widgets
	Scrollable - org.eclipse.swt.widgets
	🚊 ··· 😋 Composite - org.eclipse.swt.widgets
	Table - org.eclipse.swt.widgets
	G Tree - org.eclipse.swt.widgets
	Text - org.eclipse.swt.widgets

Eclipse JDK Tips - Refactoring

 Select the element to be manipulated in the Java editor or in a Java view and press
 Alt+Shift+T for the quick refactor menu.

if	(text.length() > 0) (
	Re <u>n</u> ame	Alt+Shift+R		
}	<u>M</u> ove	Alt+Shift+V		
	Change Method Signature	Alt+Shift+C		
	Pull <u>U</u> p			
	Extract Interface			
	<u>G</u> eneralize Type			
	Use Supertype <u>W</u> here Possible			
	Inline	Alt+Shift+I		
	E <u>x</u> tract Method	Alt+Shift+M		
	Extract Local <u>V</u> ariable	Alt+Shift+L		
	Extr <u>a</u> ct Constant			
	Introduce <u>P</u> arameter			
	Convert Local Variable to Field	Alt+Shift+F		

Eclipse JDK Tips

 More Tips and Tricks can be found in *Eclipse Help > Tips and Tricks...*

Checkout the right version of the Editor part

• cvs repository:

:pserver:anonymous@dev.eclipse.org:/home/eclipse

- Versions
- org.eclipse.ui.editors
- org.eclipse.ui.editors R3_0
- Checkout

Reference

- Eclipse 3.0 Help
- << Building Commercial Quality Eclipse Plug-ins >>
- << Eclipse In Action >>