Eclipse Plugin Development

Martin Toshev

Dmitry Alexandrov

Agenda

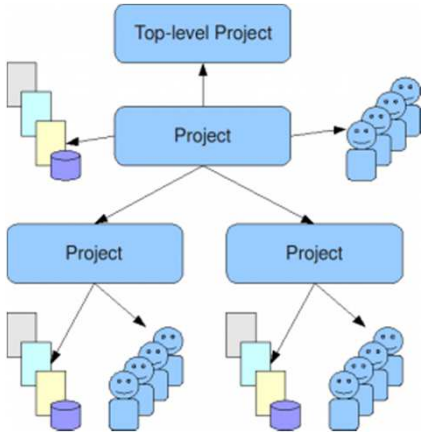
- Brief history and evolution of the platform
- OSGi
- Eclipse RCP – main components and architecture
- Deployment structure and plug-in installation flow
- Developing plug-ins
- Demos
- Q&A

History and evolution of the platform (1)

- 1998 – OTI (Object Technology International – a subsidiary of IBM purchased in 1996, now known as IBM Ottawa Lab) starts developing what is now known as Eclipse IDE
- 2001 – Eclipse IDE was outsourced in order to increase adoption and acceleration. The Eclipse consortium and eclipse.org were established (along with 8 other organizations)
- 2001 – Eclipse 1.0
- 2002 – Eclipse 2.0
- 2003 – Eclipse 2.1
- 2004 – Eclipse Software Foundation was created – on the 21st of June Eclipse 3.0 was shipped (codebase originated from VisualAge) with a runtime architecture following the OSGi Service Platform specification

History and evolution of the platform (2)

- 2004 (21st of June) – 3.0.[1]
- 2005 (28th of June) – 3.1
- 2006 (30th of June) – 3.2 (Calisto) – WTP donated from IBM
- 2007 (29th of June) – 3.3 (Europa)
- 2008 (25th of June) – 3.4 (Ganymede)
- 2009 (24th of June) – 3.5 (Galileo)
- 2010 (23rd of June) – 3.6 (Helios)
- 2011 (22nd of June) – 3.7 (Indigo)
- 2012 (27th of June) – 4.2 (Juno). Eclipse 4.2 is the result of the e4 incubation



History and evolution of the platform (3)

- 2013 (planned 26 of June) – 4.3 (Kepler)
- 2014 (planned 25 of June) – 4.4 (Luna)
- 2012 – started development of Eclipse Orion – open-source browser-based IDE for web development in the cloud

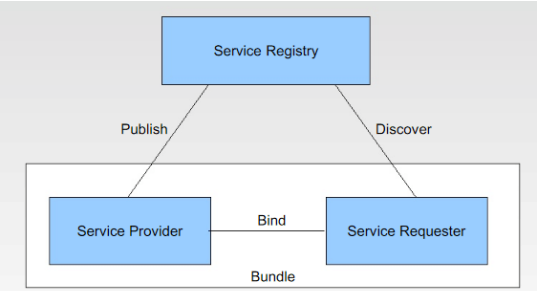
OSGi (1)

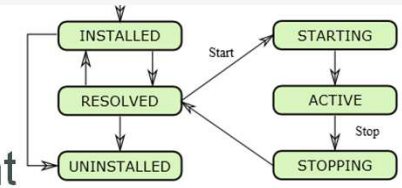
- *The **Open Services Gateway initiative framework** is a module system and service platform for the Java programming language that implements a complete and dynamic component model, something that as of 2011 does not exist in standalone Java/VM environments*
- *OSGi allows for applications or components to be installed, started, stopped, updated and uninstalled without requiring a reboot*
- *The OSGi specifications have moved beyond the original focus of service gateways, and are now used in applications ranging from mobile phones to the open source Eclipse IDE. Other application areas include automobiles, industrial automation, building automation, PDAs, grid computing, entertainment, fleet management and application servers*

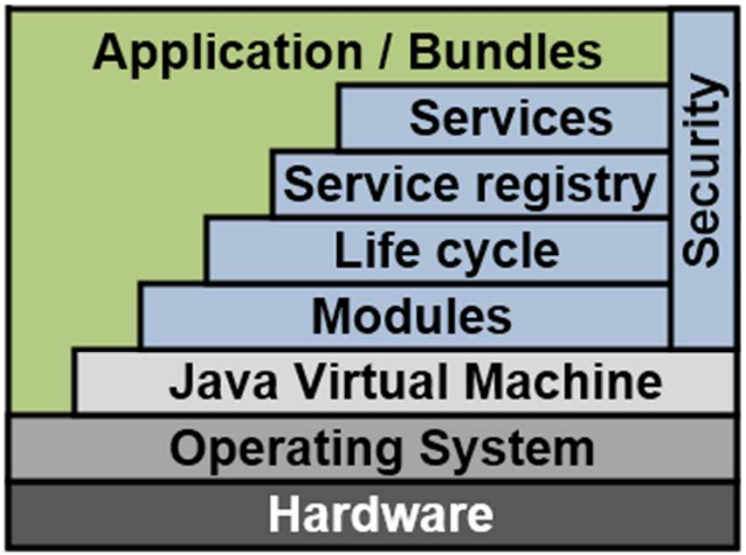
Wikipedia

OSGi (2)

- Any framework that provides an implementation based on OSGi provides an environment for the modularization of application into smaller bundles – tightly coupled, dynamically loadable collections of classes, jars and configuration files that explicitly declared their dependencies
- OSGi logical layers and units:
  - bundles
  - services
  - services registry
  - life-cycle
  - modules
  - security
  - execution environment







OSGi (3)

- Every bundle contains a manifest.mf file (the bundle descriptor) that specifies:

Bundle-Name: Defines a human-readable name for this bundle, Simply assigns a short name to the bundle

Bundle-SymbolicName: The only required header, this entry specifies a unique identifier for a bundle, based on the reverse domain name convention (used also by the java packages)

Bundle-Description: A description of the bundle's functionality

Bundle-ManifestVersion: This little known header indicates the OSGi specification to use for reading this bundle

Bundle-Version: Designates a version number to the bundle

Bundle-Activator: Indicates the class name to be invoked once a bundle is activated

Export-Package: Expresses what Java packages contained in a bundle will be made available to the outside world

Import-Package: Indicates what Java packages will be required from the outside world, in order to fulfill the dependencies needed in a bundle.

OSGi (4)

- Sample manifest.mf file:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Sample
Bundle-SymbolicName: com.sample
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: sample.Activator
Bundle-Vendor: test
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime
Bundle-
RequiredExecutionEnvironment:
JavaSE-1.7
Bundle-ActivationPolicy: lazy
```

OSGi (5)

- Advantages:

  ✓ Reduced complexity: changes can be made without affecting other modules

  ✓ Reuse: easy integration of third party components

  ✓ Easy deployment – the life-cycle management of components is well-defined

  ✓ Dynamic updates: Bundles can be installed, started, stopped, updated and uninstalled without bringing down the whole system

  ✓ Adaptive: The OSGi provides a dynamic service registry where bundles can register, get and listen to services. This dynamic service model allows bundle to find out what all services available in the system and can adapt those functionalities

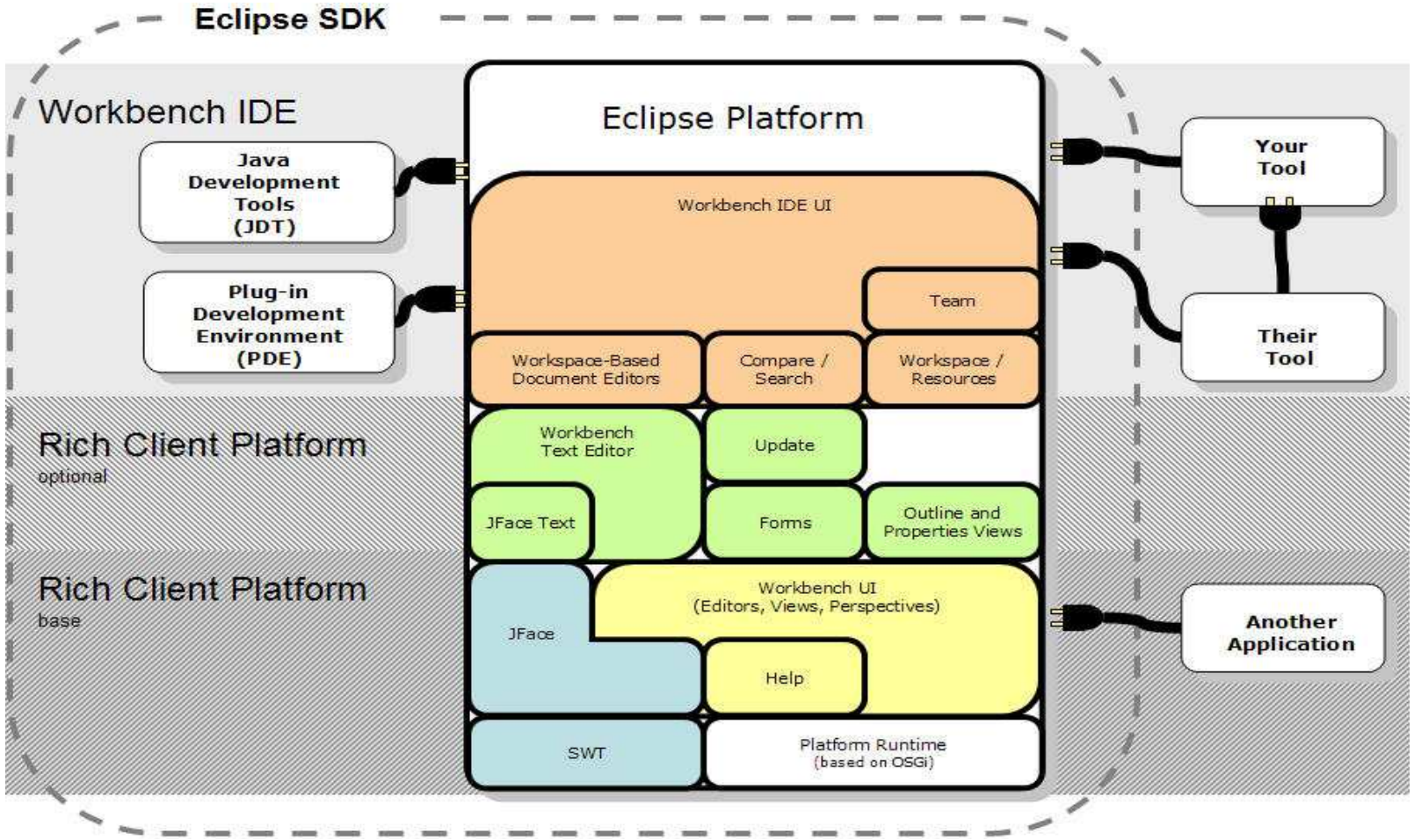
  ✓ Transparency: Certain parts of applications can be shutdown for bug fixing

Platform - main components

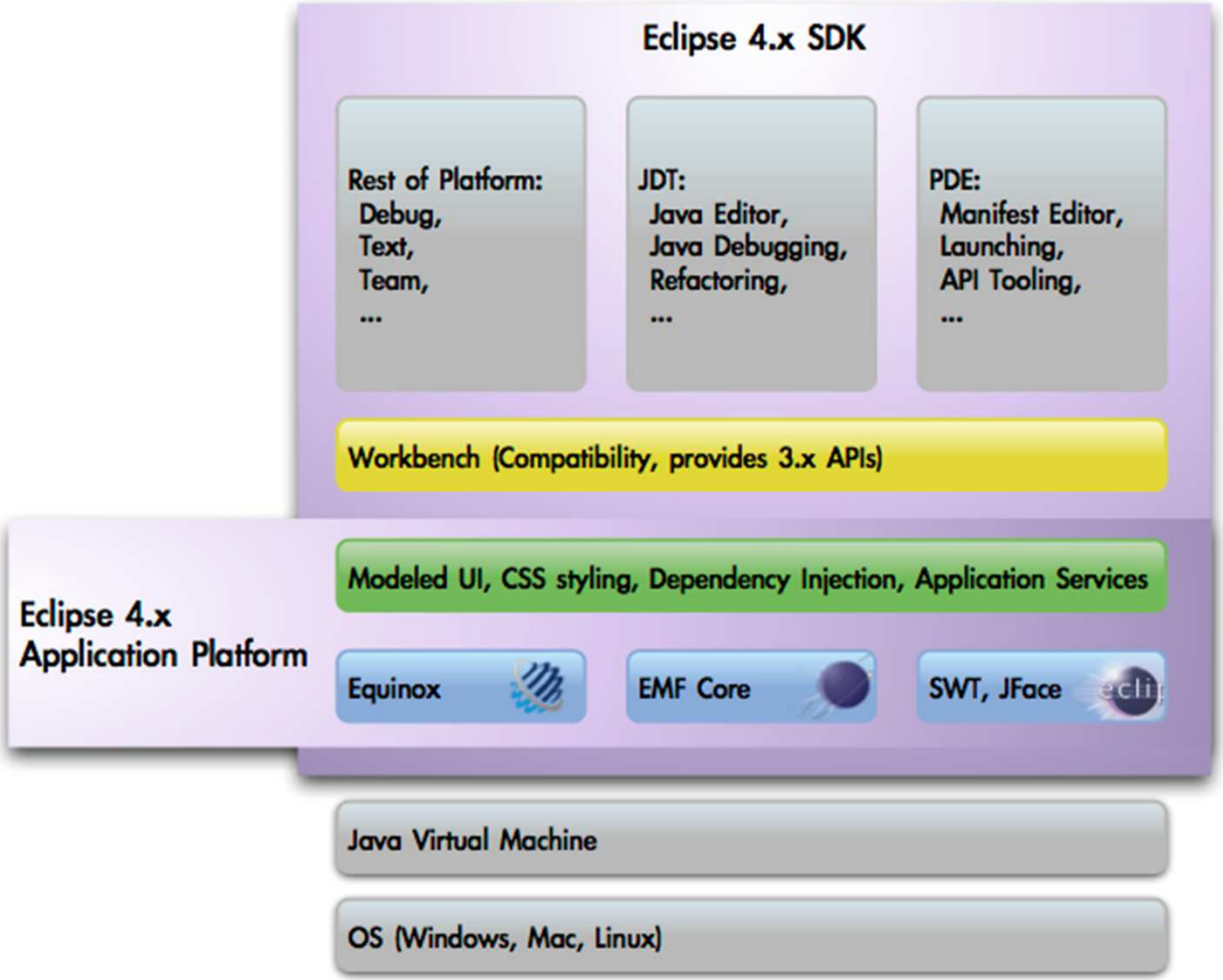
- The Eclipse RCP (rich client platform) for developing general purpose applications consists of:

  - ✓ Equinox – a standard bundling framework (OSGi implementation)
  - ✓ Core platform – boot Eclipse, run plug-ins
  - ✓ Standard Widget Toolkit (SWT) – a portable widget toolkit
  - ✓ JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors
  - ✓ Eclipse Workbench – views, editors, perspectives, wizards, actions, commands

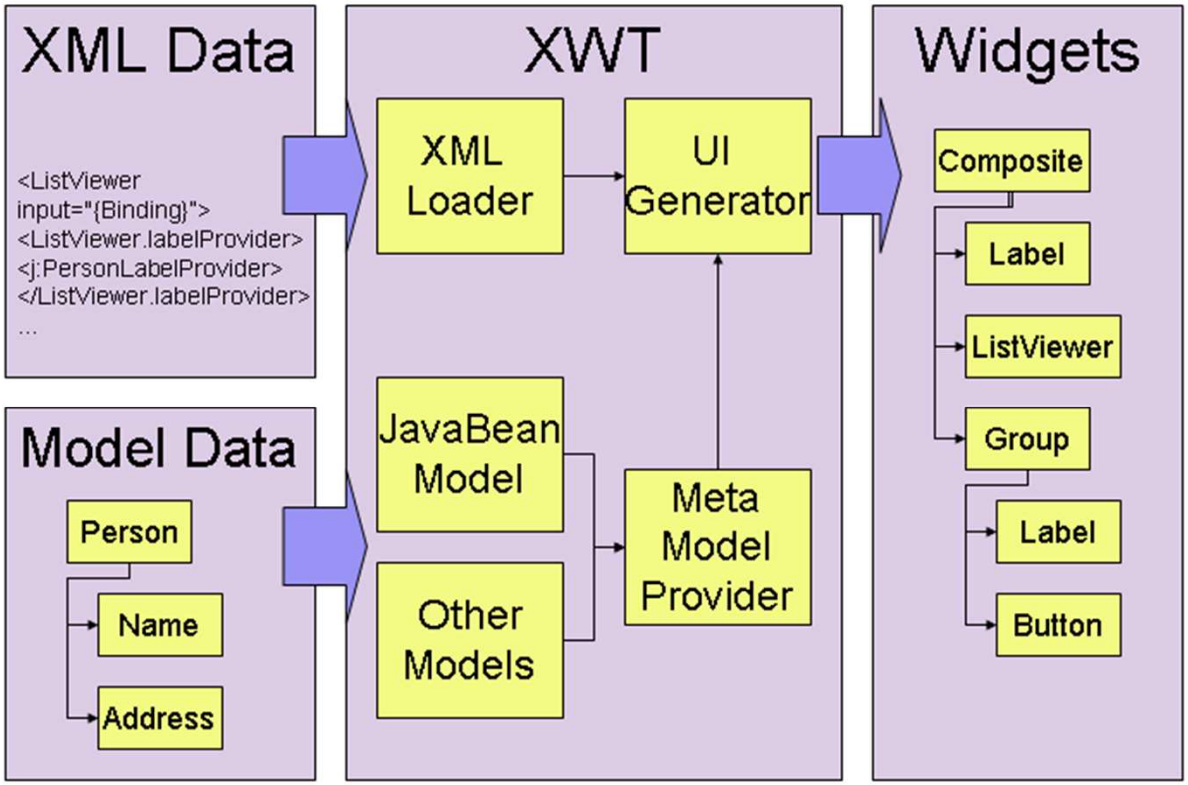
Platform - architecture



Platform – e4 adoption



Platform – e4 XWT

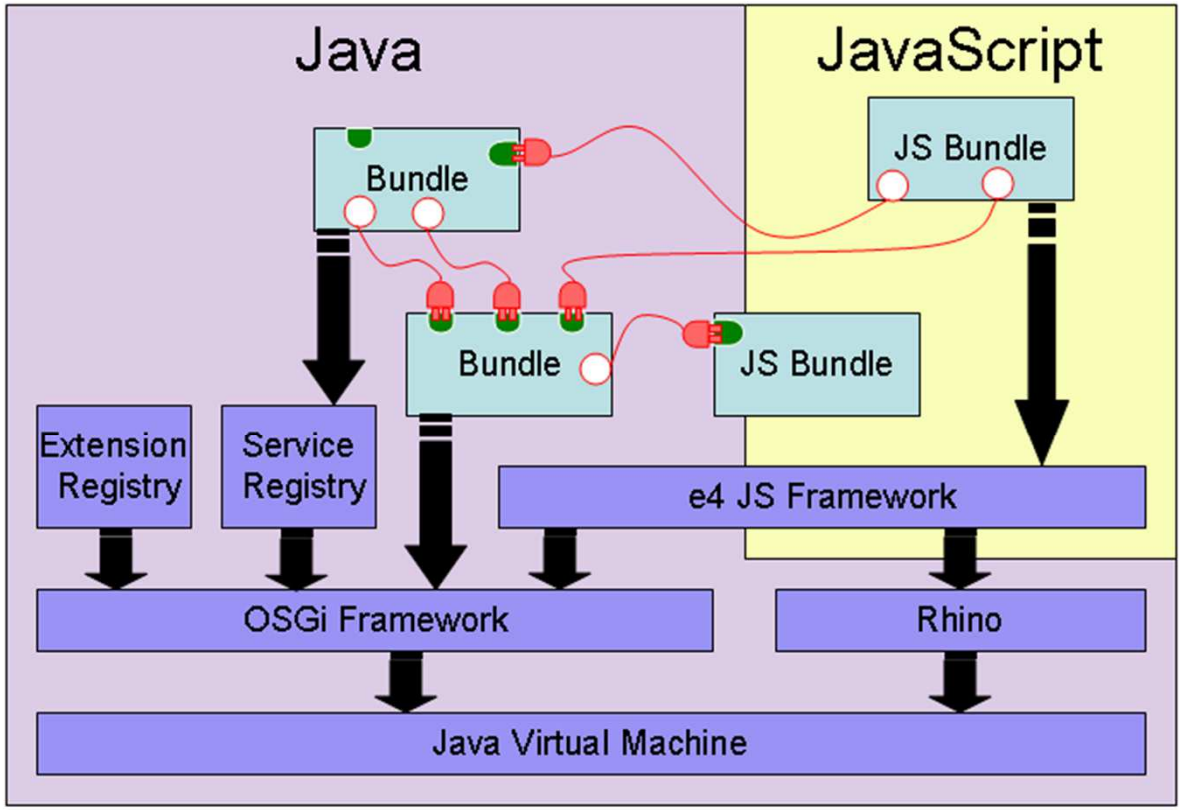


Platform – e4 XWT sample

```
<Shell xmlns="http://www.eclipse.org/xwt/presentation"
    xmlns:x="http://www.eclipse.org/xwt">
    <Shell.layout>
       <FillLayout/>
    </Shell.layout>
    <Button text="Hello, world!">
    </Button>
</Shell>
```

```
Shell parent = new Shell();
parent.setLayout(new FillLayout());
Button button = new Button(parent, SWT.NONE);
button.setText("Hello, world!");
```

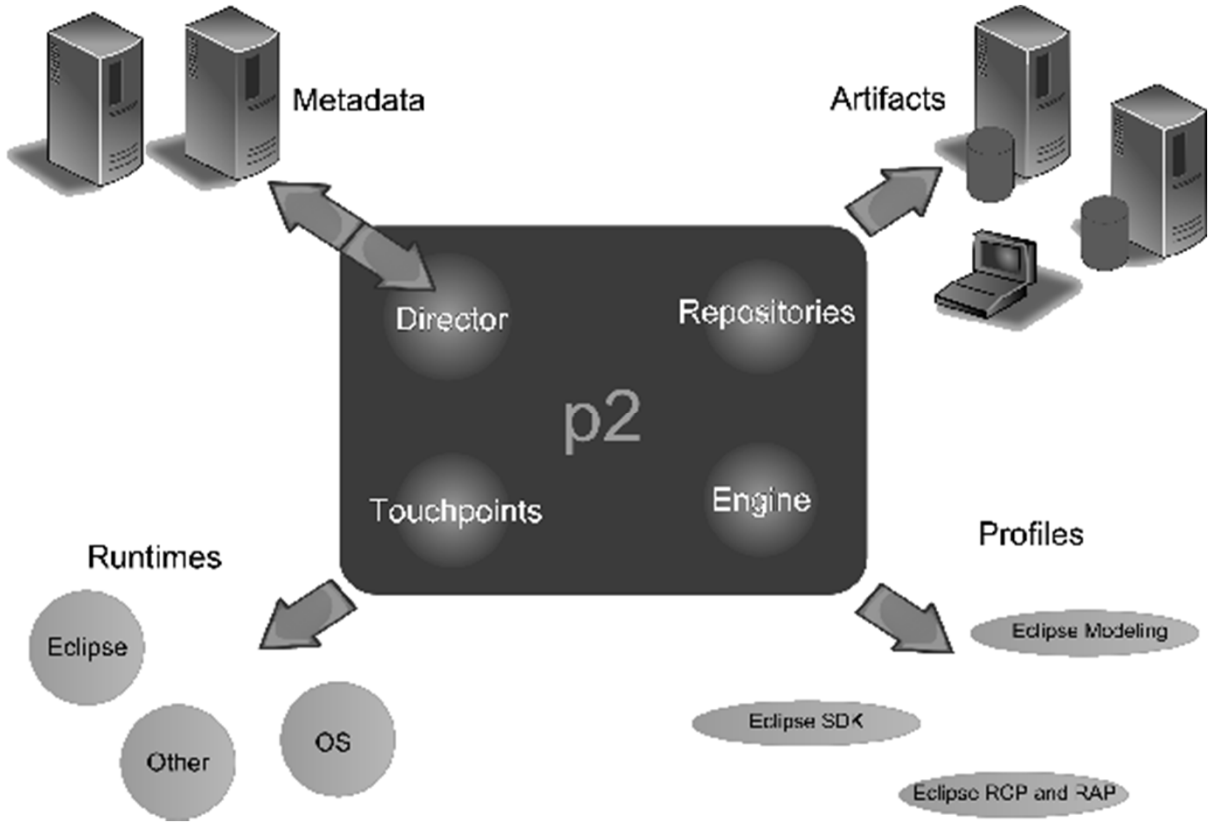
Platform – e4 javascript OSGi support



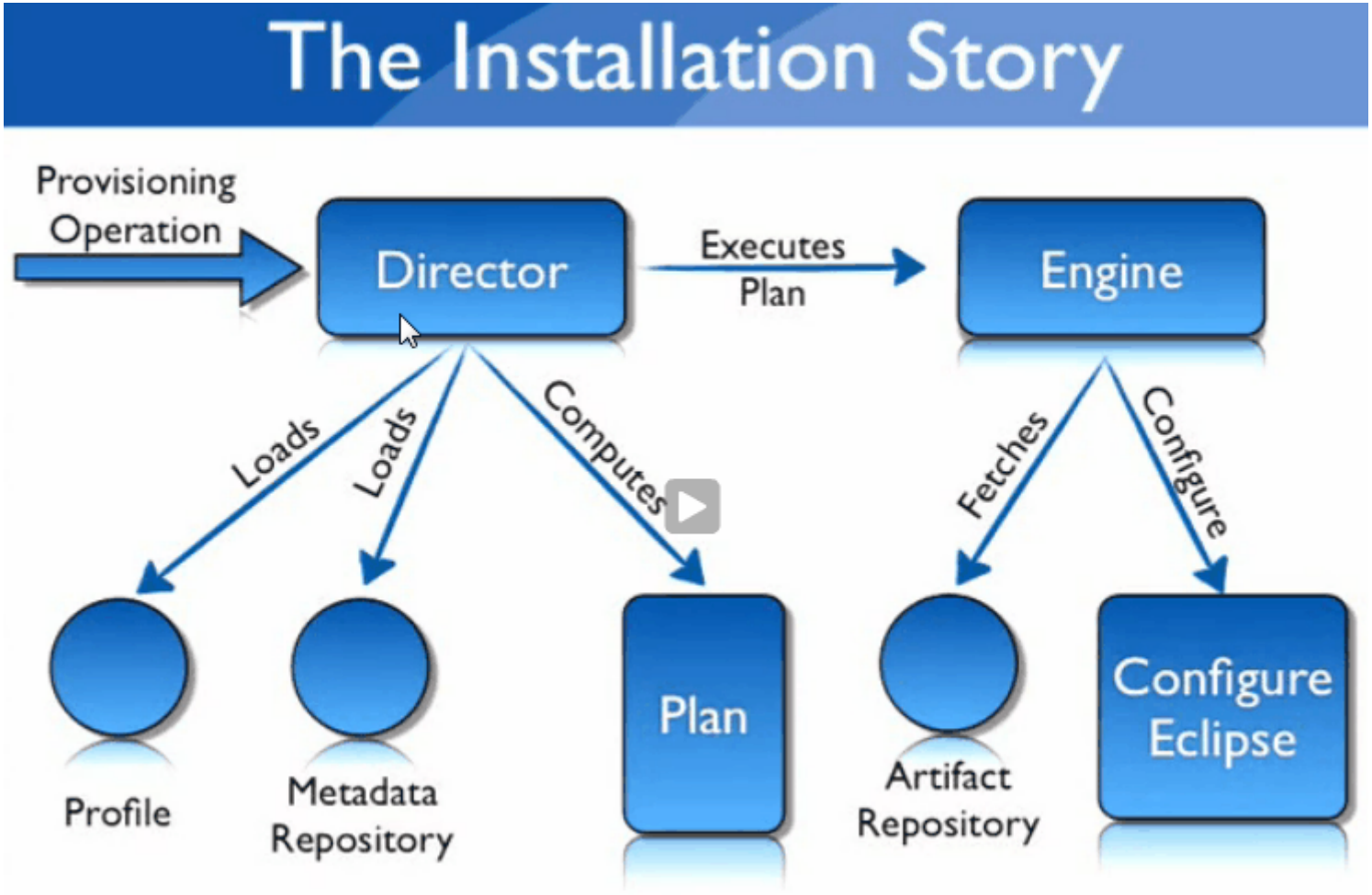
Deployment structure

- eclipse/ → main installation folder
  - configuration/ → Eclipse configuration
    - config.ini → main configuration file
    - org.eclipse.equinox.simpleconfigurator/
      - bundles.info → current set of plugins
  - dropins/ → folder scanned for plugins
  - features/ → features folder
  - p2/ → p2 configuration folder
  - plugins/ → plug-ins folder
  - eclipse.exe → Eclipse loader
  - eclipse.ini → Eclipse configuration file

Plug-in installation – p2 overview



Plug-in installation flow using p2



Plugin development – plug-ins

- OSGi bundles
- Use extension points
- Provide extension points

Plugin development – core components

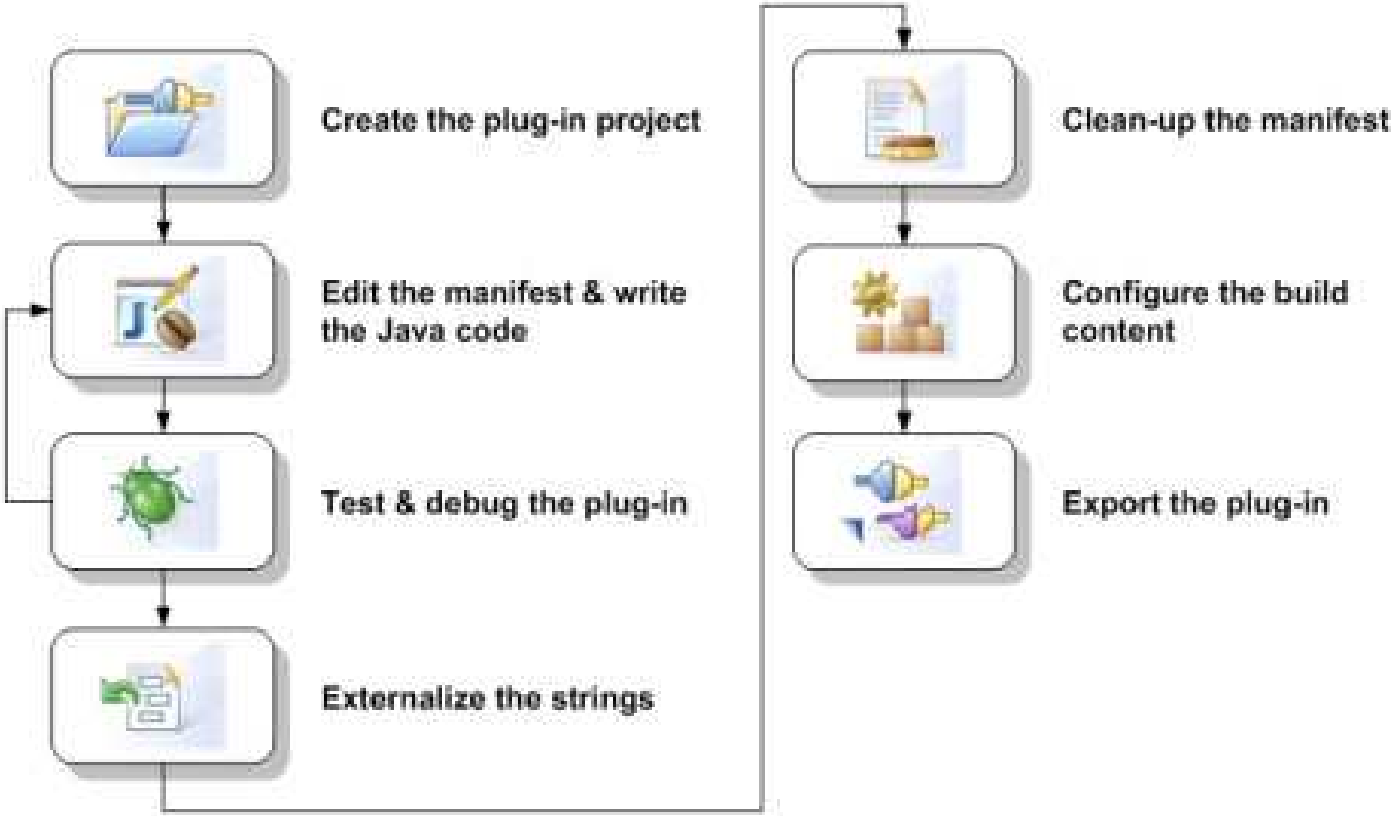
- The Eclipse Platform
- Equinox – An OSGi R4 specification implementation used by Eclipse (a common environment for executing plug-ins (bundles) and managing their life-cycle)
- Various Eclipse plug-ins – The services managed by Equinox
- The Eclipse IDE - ADT, CDT, JDT, PDT and many others

Plugin development – the workbench

Common characteristics of the workbench components:

- ✓ Created statically via the use of extensions to existing extension points (some can be created dynamically via the source code)
- ✓ Most of them have a well-defined lifecycle
- ✓ Most of them are represented by a particular class that must conform to certain rules

Plugin development – the process



Plugin development – plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
 <plugin>
        <extension
                point="org.eclipse.ui.views">
            <category
                id="sample.category"
                name="sample">
            </category>
            <view
                category="sample.category"
                class="sample"
                id="sample.view"
                name="sample"
                restorable="true">
            </view>
        </extension>
 </plugin>
```

Plugin development – views

(extension point: org.eclipse.ui.views)

- Typically used to navigate a hierarchy of information, open an editor or display properties for the active editor
- Can be grouped into categories
- Can be arranged by a perspective
- View classes must implement the IViewPart interface

Plugin development – actions & commands

- Used to supply functionality for toolbar buttons, context menu items and top-level menu items
- Commands separate presentation from implementation, while actions don't
- Can be enabled/disabled for a perspective or by a custom condition
- Action classes must implement the IActionDelegate interface, while command classes must implement the IHandler interface

Plugin development – editors (1)

(extension point: org.eclipse.ui.editors)

- Primary mechanism to modify resources – text editors, multipage editors and others
- Have an open-modify-save-close lifecycle
- Can be stacked
- Editors can be:
  - ✓ Text editors.
  - ✓ Form-based editors can layout controls in a fashion similar to a dialog or wizard.
  - ✓ Graphics intensive editors can be written using SWT level code.
  - ✓ List-oriented editors can use JFace list, tree, and table viewers to manipulate their data.

Plugin development – editors (2)

- Editor classes must implement the IEditorPart interface or extend EditorPart
- Editor input is manipulated by means of the IEditorInput interface
- 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
  - ✓ visual presentation of text annotations in rulers or as squigglies in the text
  - ✓ automatic update of annotations as the user edits text
  - ✓ presentation of additional information such as line numbers
  - ✓ syntax highlighting
  - ✓ content assist
  - ✓ text outlining pages that show the hierarchical structure of the text
  - ✓ context sensitive behavior
  - ✓ hover support over rulers and text
  - ✓ key binding contexts
  - ✓ preference handling

Plugin development – editors (3)

- A custom text editor can be created by extending AbstractTextEditor or TextEditor
- For source code style editors, a SourceViewer is provided. It can be customized by extending SourceViewerConfiguration.
- Operations on text resources:
  - ✓ Partitioning & Syntax highlighting
    - ✓ Scanner
    - ✓ Rules
      - o IPredicateRule
      - o IRule
    - ✓ Detectors
  - ✓ Formatting
  - ✓ Completion

Plugin development – editors (5)

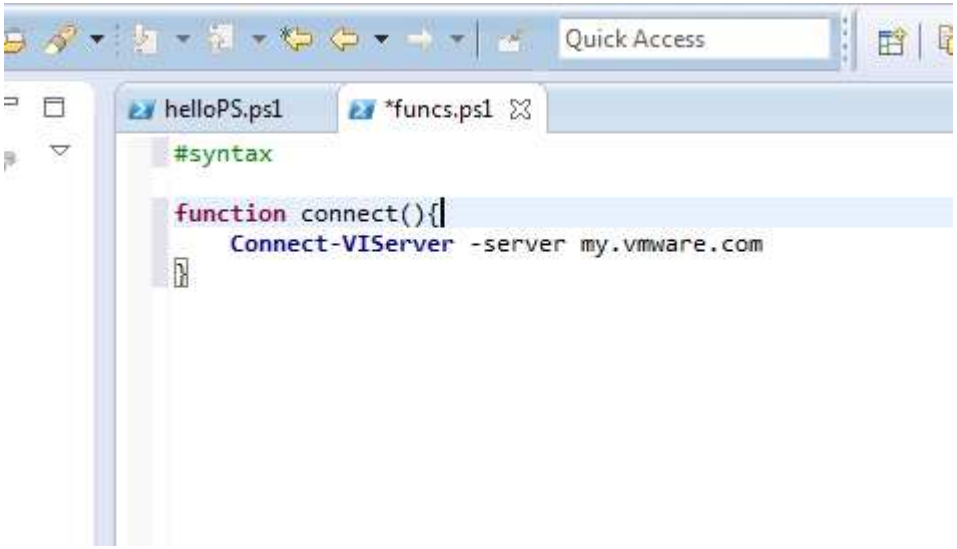
- Working with text resources – document structure:
  - Partitioning
    - IPredicateRule

*Class1.java

```
1  /**
2   * @author author
3   *
4   */
5  public class Class1 {
6
7      private static String STRING_VALUE;
8
9      private int intValue;
10
11     public static void main(String[] args) {
12         new Class1().helper();
13     }
14
15     private void helper() {
16         System.out.println("Printing string: " + STRING_VALUE);
17         System.out.println("Printing int: " + intValue);
18     }
19
20 }
21
```

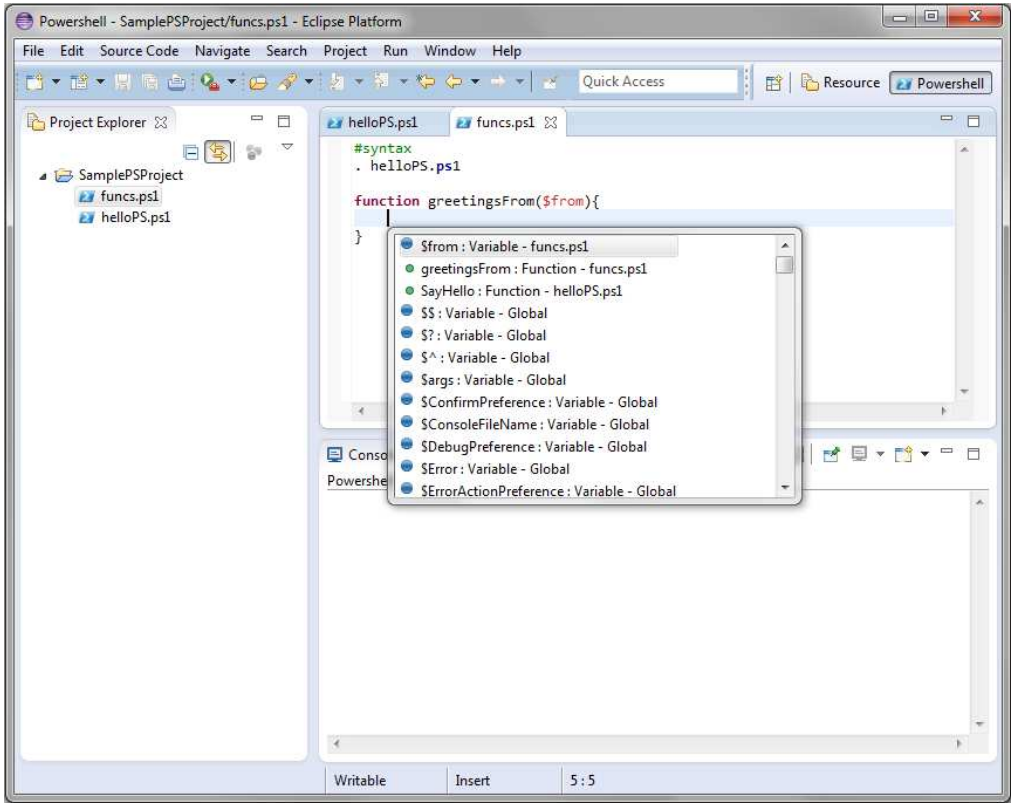
Plugin development – editors (6)

- Working with text resources - highlighting:
  - ✓ Damage, repair, and reconciling (IPresentationDamager, IPresentationRepairer, IPresentationReconciler)
  - ✓ Scanner
  - ✓ Detectors
  - ✓ Rules
    - o IRule



Plugin development – editors (4)

- Working with text resources – content assist:
  - ✓ IContentAssistant
  - ✓ IContentAssistProcessor



Plugin development – editors (7)

- Working with text resources – document readability:
  - ✓ IFormattingStrategy

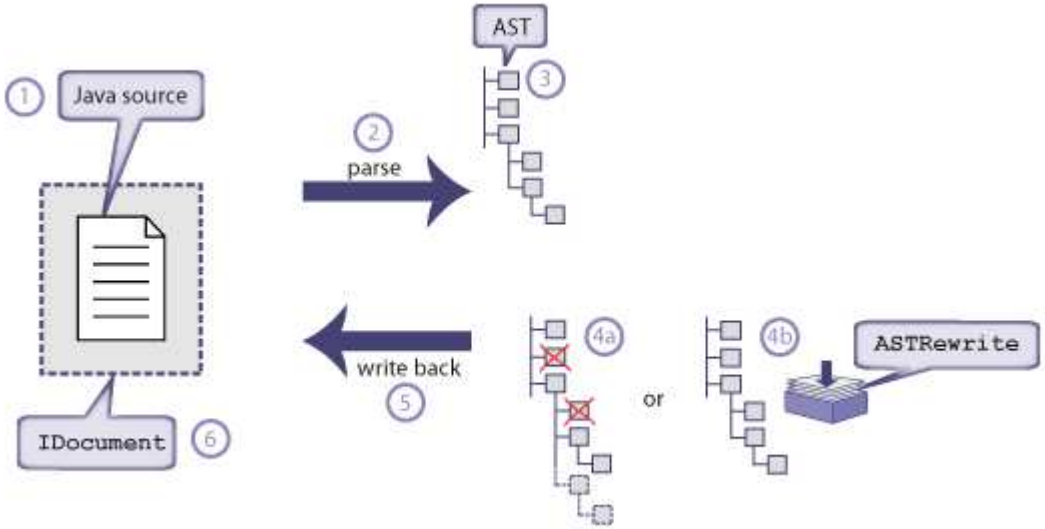
funcs.ps1 | *helloPS.ps1

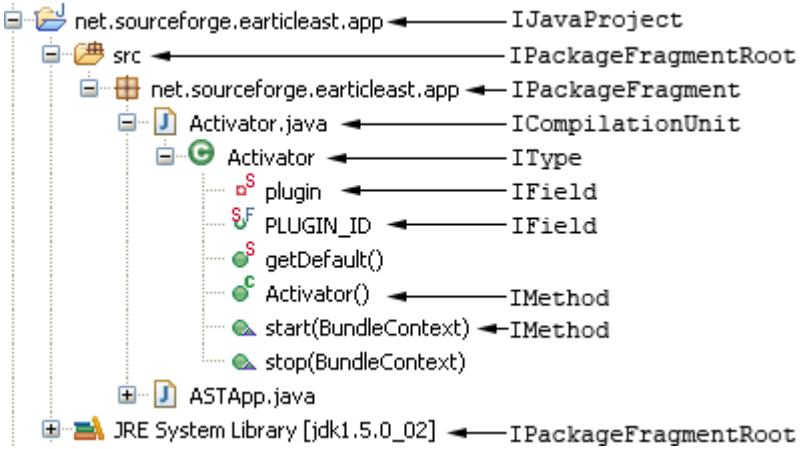
```
#Hello PS Example

function SayHello(){ function ForMe(){function Here(){
Write-Host "Hello PS!"
        Get-Acl -opt1
    -opt2
          -opt3
        }SayHello}
    SayHello}
```

funcs.ps1 | *helloPS.ps1

```
#Hello PS Example

function SayHello(){
    function ForMe(){
        function Here(){
            Write-Host "Hello PS!"
            Get-Acl -opt1
                    -opt2
                    -opt3
        }
        SayHello
    }
    SayHello
}
```

Plugin development – the AST



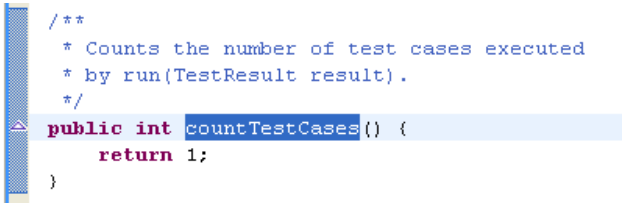


Plugin development – editors (8)

- Working with text resources – Annotations and rulers:

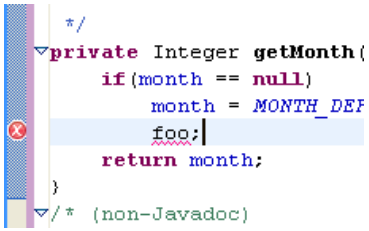
```
<extension
        point="org.eclipse.core.filebuffers.annotationModelCreation">
        <factory extensions="*"
        class="org.eclipse.ui.texteditor.ResourceMarkerAnnotationModelFactory">
        </factory>
</extension>
```

✓ Vertical ruler



✓ Overview ruler

✓ Text annotations



Plugin development – editors (9)

- Working with text resources – Hover information:
  - ✓ Text hover

```
* The language reserved words. */
ivate static String[] languageReservedWords = { "if", "else", "elseif",
        "switch", "while
        "break", "contin
        "trap", "throw",

* The dash operators.
ivate static String[]
        "-lt", "-le", "-
```

String[] org.powershell.editors.PowershellScanner.languageReservedWords

The language reserved words.

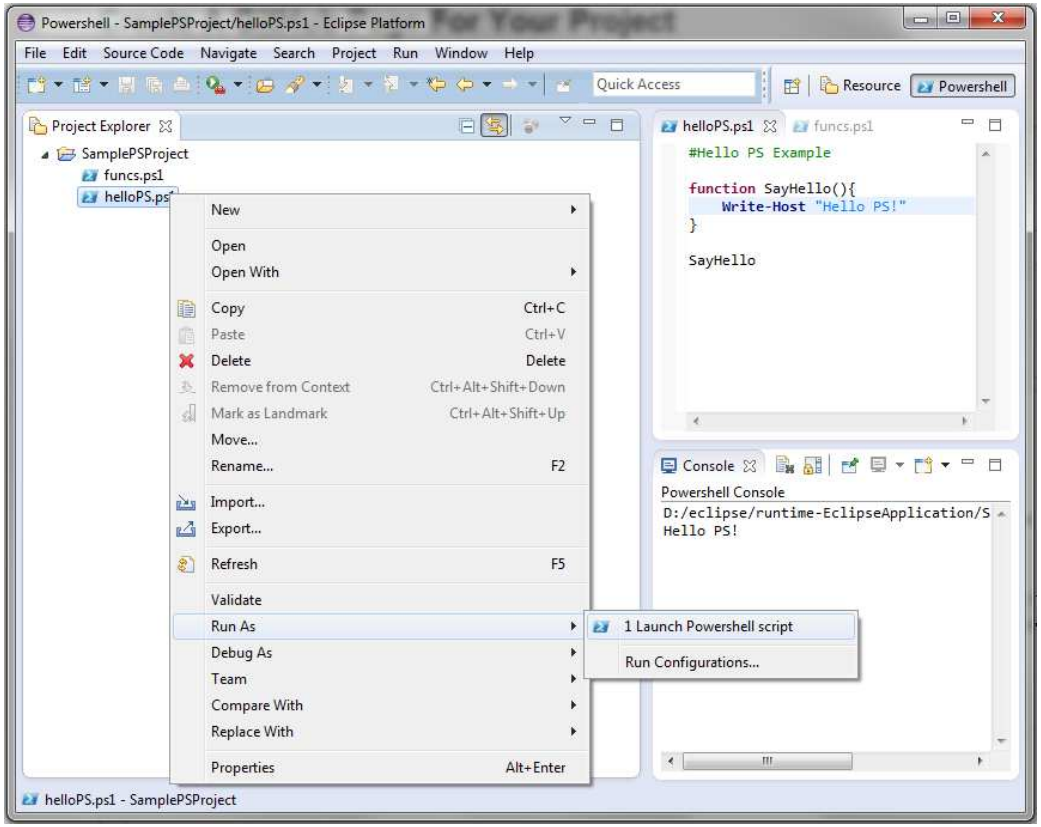
Press 'F2' for focus

  - ✓ Ruler hover

```
25      */
26      @Override
27  overrides org.eclipse.ui.editors.text.TextEditor.initializeEditor
28          super.initializeEditor();
29          setSourceViewerConfiguration(new Powershel
```

Plugin development – launching & console view

- IDE sugar
  - Launch functionality
  - Delegate
  - Shortcut
  - LaunchConfiguration
- Console
  - Work with MessageConsole



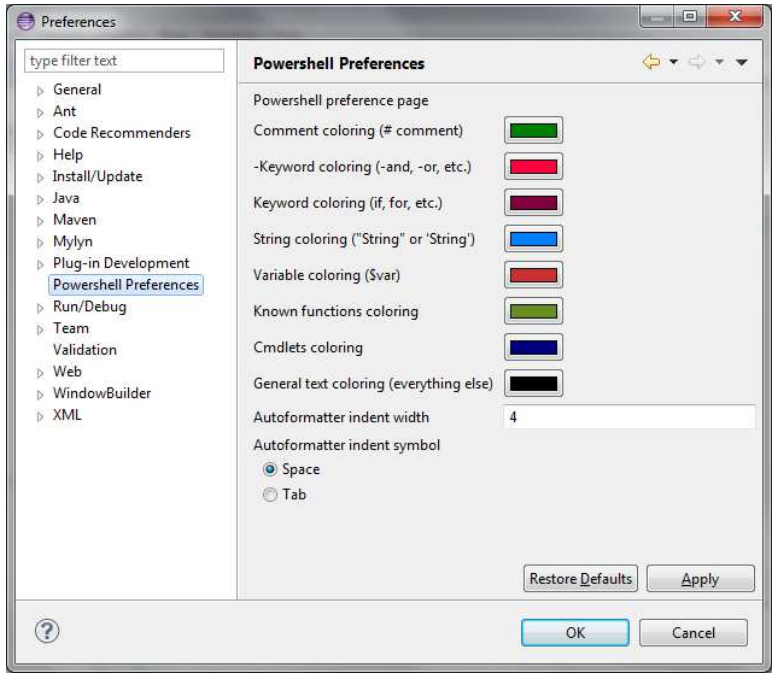
Plugin development – perspectives

(extension point: org.eclipse.ui.perspectives)

- Used for organizing/arranging views, editors and commands in the workbench
- Already existing perspectives can be enhanced as well
- Perspectives classes must implement the IPerspectiveFactory interface

Plugin development – other extensions

- Preferences (org.eclipse.ui.preferencePages)
- Properties (org.eclipse.ui.propertyPages)
- Help pages (org.eclipse.ui.help)

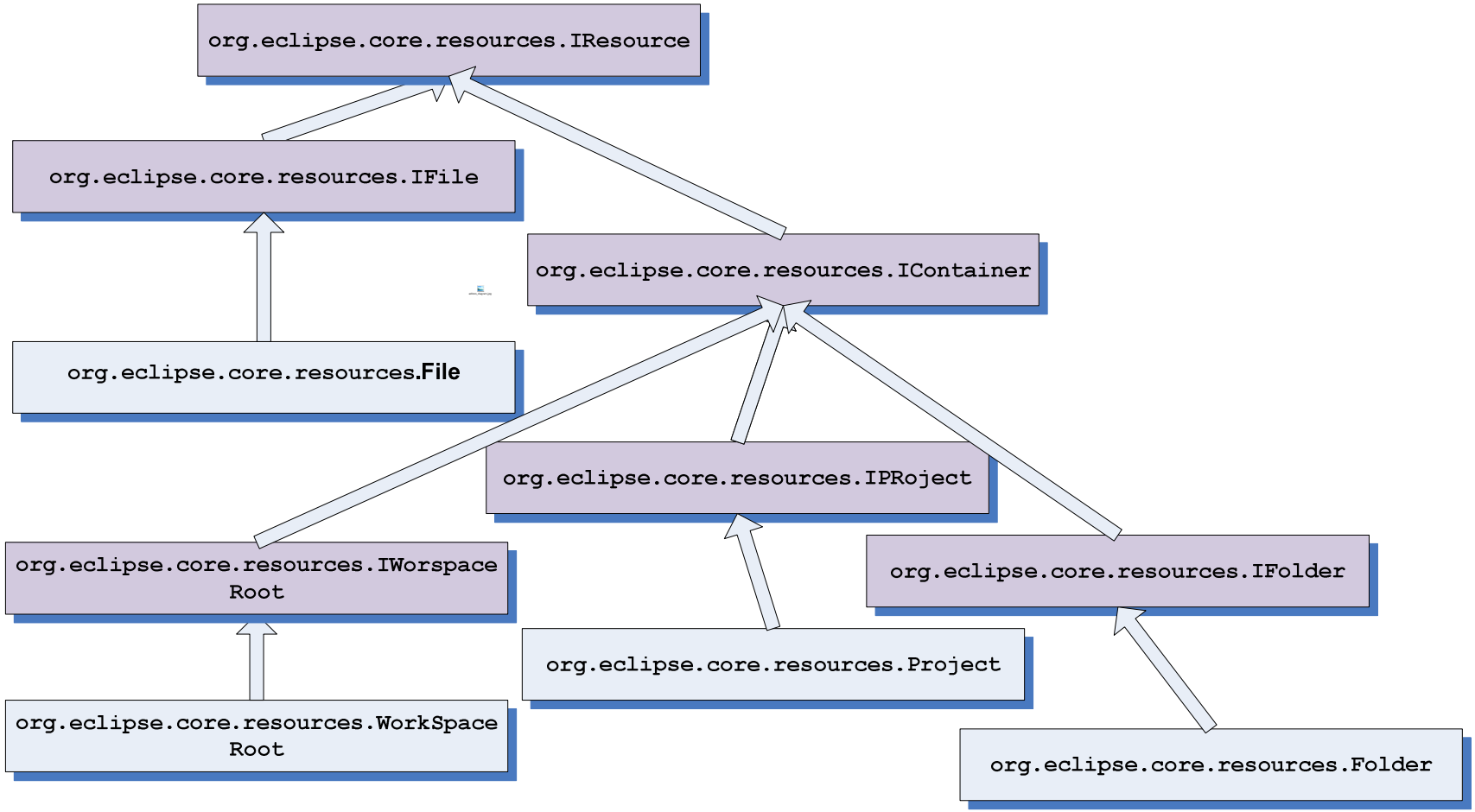


Plugin development – EFS

In the EFS (Eclipse file system) all resources implement the IResource interface:

- ✓ files (IFile resources)
- ✓ folders (IFolder resources)
- ✓ projects (IProject resources)
- ✓ the workspace root (IWorkspaceRoot resource)

Plugin development – EFS



Plugin development – resource change tracking

- IResourceChangeListener

```
ResourcesPlugin.getWorkspace().addResourceChangeListener(
    listener, IResourceChangeEvent.POST_CHANGE);
```

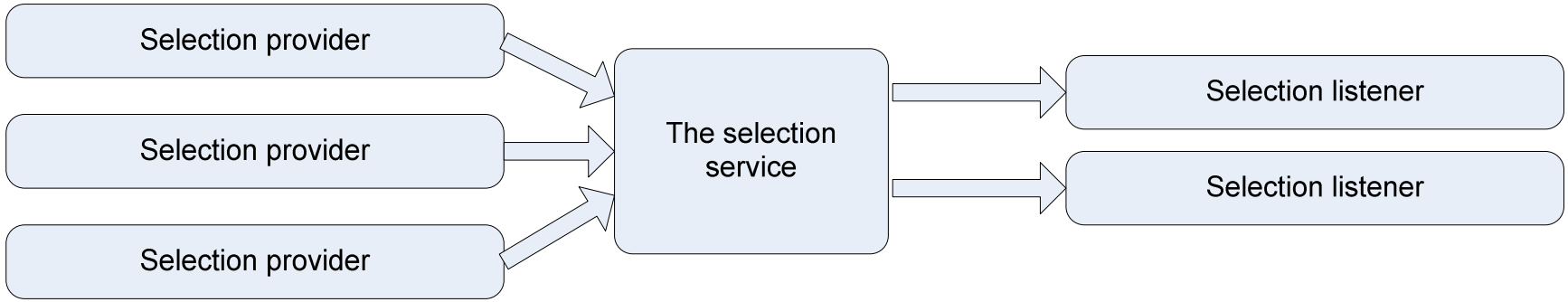
- IResourceDeltaVisitor
  - ✓ The resource delta is structured as a tree rooted at the workspace root
  - ✓ Resources that have been created, deleted, or changed. If you have deleted (or added) a folder, the resource delta will include the folder and all files contained in the folder
  - ✓ Resources that have been moved or renamed using the IResource.move() API
  - ✓ Markers that have been added, removed, or changed. Marker modification is considered to be a workspace modification operation
  - ✓ Files that have been modified. Changed files are identified in the resource delta, but you do not have access to the previous content of the file in the resource delta

Plugin development – services

- The workbench defines a number of services that can be retrieved from the org.eclipse.ui.services.IServiceLocator:

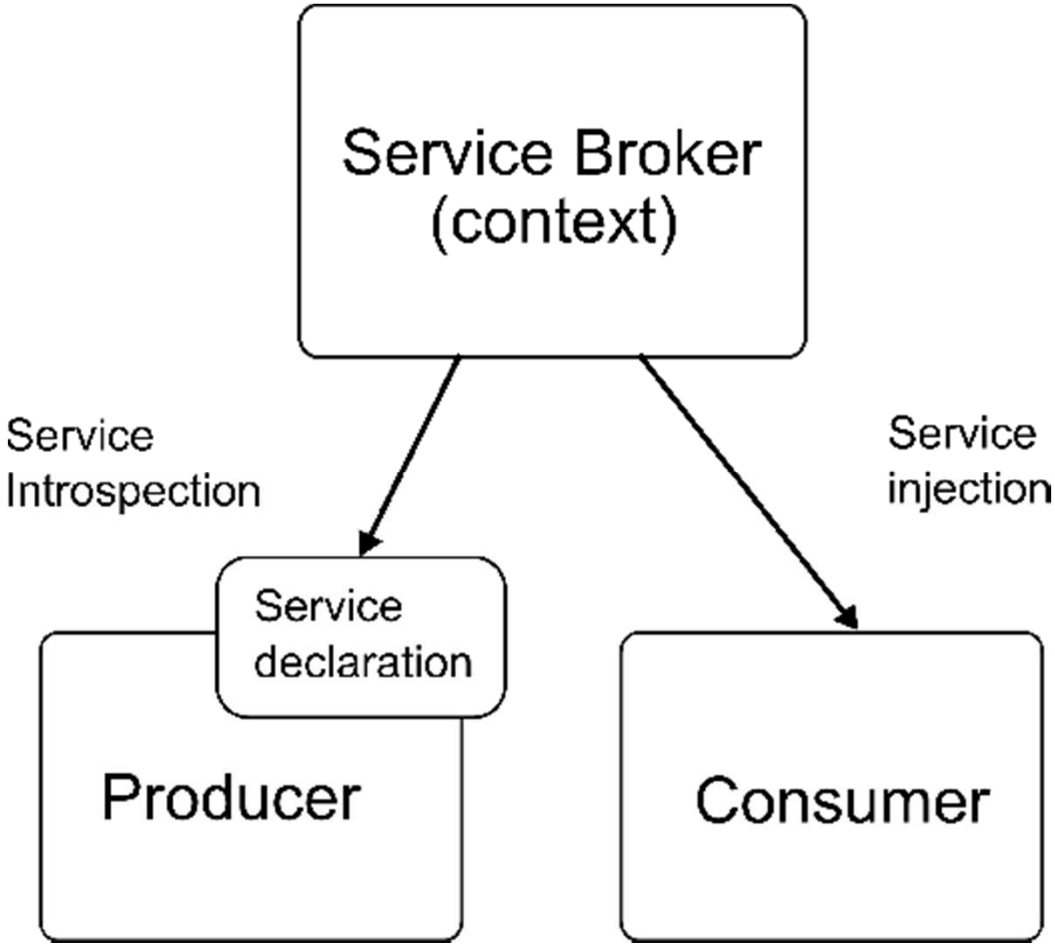
Service	Description	Availability
IBindingService	Provides services related to the binding architecture (e.g., keyboard shortcuts) within the workbench.	Globally
ICommandService	Provides services related to the command architecture within the workbench.	Globally
ICommandImageService	Provides a look-up facility for images associated with commands.	Globally
IContextService	Provides services related to contexts in the Eclipse workbench, like context activation and definitions.	Globally
IContributionService	Standard mechanisms that clients may use to order, display, and generally work with contributions to the Workbench.	Globally
IEvaluationService	Evaluate a core expression against the workbench application context and report updates using a Boolean property.	Globally
IFocusService	Tracks focusGained and focusLost events.	Globally
IHandlerService	Provides services related to activating and deactivating handlers within the workbench.	Globally
IMenuService	Provides services related to the menu architecture within the workbench.	Globally
IPageService	A page service tracks the page and perspective lifecycle events within a workbench window.	Workbench Window
IPartService	A part service tracks the creation and activation of parts within a workbench window.	Workbench Window
IProgressService	The progress service is the primary interface to the workbench progress support.	Globally
IWorkbenchSiteProgressService	The part progress service is an IProgressService that adds API for jobs that change the state in a IWorkbenchPartSite while they are being run.	Part Site
ISaveablesLifecycleListener	Listener for events fired by implementers of ISaveablesSource.	Globally
ISelectionService	A selection service tracks the selection within an a workbench window.	Workbench Window

Plugin development – the selection service




```
getSite().getWorkbenchWindow()
        .getSelectionService().addSelectionListener(<listener>);
        // <listener> must implement ISelectionListener

getSite().setSelectionProvider(<provider>);
        // <provider> must implement ISelectionProvider)
```

Plugin development – e4 service overview



Plugin development – packaging

- Plug-in – an OSGi bundle
- Feature – grouping of plug-ins and fragments
- Fragment – extend the functionality of another plug-in
- Update site – grouping of features that can be installed from a site

Plugin development – IDE inspection

- The plug-in selection spy (Alt + Shift + F1) / menu spy (Alt + Shift + F2)
- Import plug-ins / fragment from the Import Wizard
- Window -> Preferences -> Plug-in Development -> Include all plug-ins from target in Java search
- Eclipse Git repo
- Plug-in Registry/Plug-ins/Plug-in Dependencies

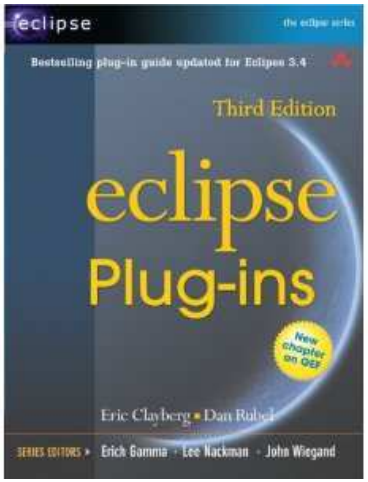
Further topics

- Implementing help – providing useful information for a plug-in
- Internationalization – expanding the scope of our plug-in
- Building a plug-in – creating an Ant/Maven - based build script for a plug-in
- Publishing a plug-in – creating an update site for a plug-in
- SWT and JFace
- GEF and GMF
- Comparison with another platforms in terms of plug-in development
- Overview of existing plug-ins and features
- Providing extension points for a plug-in

Recommended readings

- Eclipse Plug-ins (3rd edition) – Eric Clayberg, Dan Rubel

- Eclipse corner articles (http://www.eclipse.org/articles/)



Q & A

Additional references (1)

- Brief history of Eclipse (up to 2006)
  http://www.venukb.com/2006/07/21/history-of-eclipse/
  last visited: 01.03.2013

- Wikipedia's entry on Eclipse
  http://en.wikipedia.org/wiki/Eclipse_(software)
  last visited: 01.03.2013

- eclipse.org: Eclipse e4
  http://eclipse.org/e4/
  last visited: 01.03.2013

- EclipseCon 2009: e4 Project in Review
  http://live.eclipse.org/node/737
  last visited: 01.03.2013

Additional references (2)

- eclipse.org: Eclipse development process
  http://www.eclipse.org/eclipse/development/
  last visited: 01.03.2013

- Eclipse wiki: Development Resources
  http://wiki.eclipse.org/Development_Resources
  last visited: 01.03.2013

- Eclipse wiki:
  http://wiki.eclipse.org/Development_Resources/Process_Guidelines/What_is_Incubation
  last visited: 01.03.2013

Additional references (3)

- Architecture of the Eclipse platform
  https://wiki.engr.illinois.edu/download/attachments/183861826/cs427-12.pdf?version=1&modificationDate=1317337982000
  last visited: 01.03.2013

- vogella.de: Extending Eclipse –Plug-in Development Tutorial
  http://www.vogella.de/articles/EclipsePlugIn/article.html
  last visited: 01.03.2013

- OSGi tutorial
  http://www.knopflerfish.org/tutorials/osgi_tutorial.pdf
  last visited: 01.03.2013

- vogella.de: OSGi with Eclipse Equinox
  http://www.vogella.de/articles/OSGi/article.html
  last visited: 01.03.2013

Additional references (4)

- Wikipedia entry on OSGi
  http://en.wikipedia.org/wiki/OSGi
  last visited: 01.03.2013

- Wikipedia entry of Equinox
  http://en.wikipedia.org/wiki/Equinox_(OSGi)
  last visited: 01.03.2013

- Understanding the Eclipse p2 provisioning system
  http://eclipse.dzone.com/articles/understanding-eclipse-p2-provi
  last visited 01.03.2013

- Introduction to p2 (video)
  http://www.fosslc.org/drupal/content/gentle-introduction-p2
  last visited: 01.03.2013

Additional references (5)

- eclipse.org: project plan for Eclipse project, version Juno
  http://www.eclipse.org/projects/project-plan.php?planurl=http://www.eclipse.org/eclipse/development/plans/eclipse_project_plan_4_2.xml
  last visited: 01.03.2013

- The architecture of open source applications (chapter 6: Eclipse)
  http://www.aosabook.org/en/eclipse.html
  last visited: 01.03.2013

- Eclipse Plug-in developers guide – Editor
  http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Feditors.htm&cp=2_0_13
  last visited: 01.03.2013