eclipse CON 2005

Advanced UI Development in the Eclipse Workbench

Part Two, of a two part tutorial series

© 2005 by IBM; made available under the EPL v1.0

About the Speakers

- Tod Creasey
  - Senior Software Developer, Platform UI Committer
  - working with the IBM OTI Ottawa labs (formerly known as Object Technology International) since 1993 on a variety of IDEs starting with Envy developer and progressing through the Visual Age family of Smalltalk and Java IDEs
  - joined the Eclipse team during the tech preview and has been a member of the Workbench team since
- Michael Van Meekeren
  - Platform UI Team Leader, Platform UI Committer
  - working with the IBM OTI Ottawa labs (formerly known as Object Technology International) since 1994, and has played an active role in the development of Envy developer, IBM Smalltalk, VisualAge for Java and WebSphere Studio Device Developer
  - also has J2ME and embedded programming experience
- We have a lot of experience on the UI team for Eclipse, however do not know everything, will try to give answers to as many questions as possible

 © 2005 by IBM; made available under the EPL v1.0

Getting Started:

- Survey
  - Eclipse Experience
  - Specific things to cover
- Laptop Setup
- Goals for this section
- What this tutorial is NOT
- JFace/Workbench/IDE plug-in structure
- Questions/Comments

 © 2005 by IBM; made available under the EPL v1.0

Setup

- Start your engines!!!
- If you have a laptop
  - Eclipse SDK installed?
  - Examples installed?
- No laptop?
  - Think of it as an opportunity to meet someone new
- CDs/Flash memory cards available
- NOTE: questions and comments are welcome at any point

 © 2005 by IBM; made available under the EPL v1.0

Goals

- Understand the IDE and Workbench plug-in structure
- Learn the advanced components that make up the Workbench
  - build on a simple “Image Navigator” application
    - taken from this mornings tutorial
    - take plug-ins based on JFace and the Workbench and take advantage of what advanced features are available
    - build an IDE enabled plug-in
    - work with things like views, wizards and menus
- Questions/Comments

 © 2005 by IBM; made available under the EPL v1.0

What this tutorial is NOT

- Not a tutorial about how to write plug-ins or use the PDE (Plug-in Development Environment)
- Not a basic Workbench tutorial
  - That was earlier on today
- Not a Rich Client Platform (RCP) tutorial
  - That's also happening now
- Not 4 hours of two guys talking
  - Hands on tutorial

 © 2005 by IBM; made available under the EPL v1.0

Conventions used in this presentation

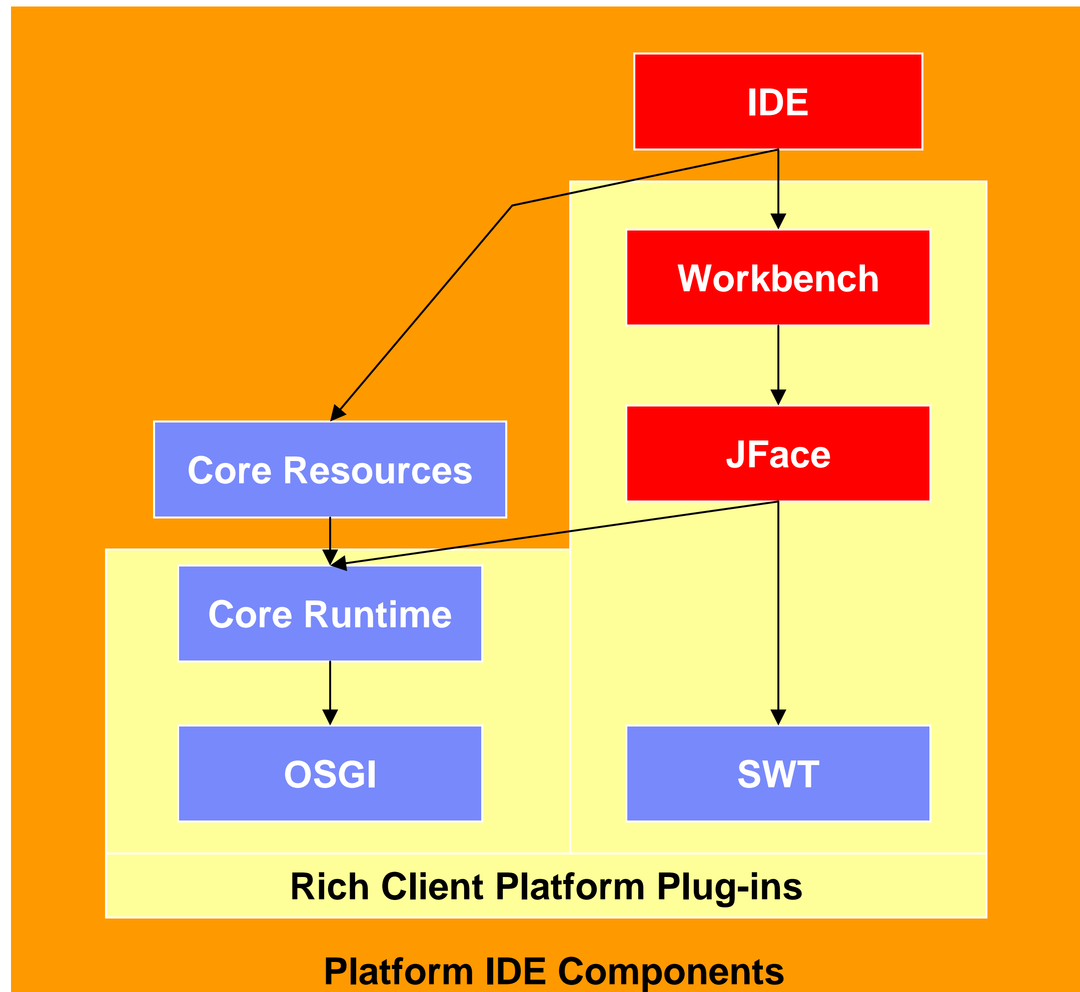
`Courier text refers to code or XML in the text`

```
Turquoise boxes are code or xml snippets
```

TIP: Yellow boxes are for tips

 © 2005 by IBM; made available under the EPL v1.0

Platform Plug-in Structure



 © 2005 by IBM; made available under the EPL v1.0

Plug-in Structure

- JFace is dependant on SWT and the Core runtime support (only uses some helper classes like IStatus)
- The workbench uses JFace and adds some functionality
- The IDE layer adds a dependency on Core Resources

 © 2005 by IBM; made available under the EPL v1.0

eclipse CON 2005

Questions and Comments?

 © 2005 by IBM; made available under the EPL v1.0

Workbench and IDE Advanced topics

- Workspace Root/Workbench Content Provider
- Step back and describe Workbench and IDE split
- Filters
- ActionSets/Commands/Keys
- Editors
- Wizards
- Perspectives
- Capabilities
- Startup and Shutdown
- Jobs
- Decorators
- Adaptable
- Common Expressions
- Help

 © 2005 by IBM; made available under the EPL v1.0

Workspace Root/Workbench Content Provider

- See …example.advanced.IDEImageNavigator
- The Workspace Root is the top directory of your workspace
- The WorkspaceContentProvider uses this as the root of the tree
- WorkspaceContentProvider is defined in org.eclipse.ui.ide
- This is not available to RCP applications by default
- Convert content provider to be WorkbenchContentProvider
- Preferences and resource listeners not required now

 © 2005 by IBM; made available under the EPL v1.0

Step back and describe Workbench,IDE and RCP

- Pre 3.0 `org.eclipse.ui.workbench` referred to the API in `org.eclipse.core.resources`
- RCP applications often use a different content story than IResource
  - we split workbench into workbench and IDE
- IResources are only used at the IDE level

 © 2005 by IBM; made available under the EPL v1.0

Questions and Comments?

© 2005 by IBM; made available under the EPL v1.0

ActionSets

- We sometimes want to add actions to the toolbar or menu bar
- We can use the action set extension point
- Create a menu bar entry and a group for it
- Toolbar paths are added if they don't match any existing ones
- Create an Action to open a filter extensions dialog
- Self host and use the menu and toolbars

TIP: Actions are a very useful way to reuse functionality in different places in the workbench

 © 2005 by IBM; made available under the EPL v1.0

Commands/Keys

- Define a command using the extension point (`org.eclipse.ui.commands`) for the action and provide a key binding
  - bind to our action that opened the filter settings dialog
  - provide a key binding for the command (Ctrl+Alt+Shift+I)
- Launch with demo

TIP: Eclipse applications must share key bindings between plug-ins so it is easy to run out of available key sequences. As the user can set these themselves using the Keys Preference Page it is not a requirement that you define key sequences for all of your commands.

 © 2005 by IBM; made available under the EPL v1.0

Commands plug-ins

- org.eclipse.ui.commands extension point provides:
  - Key configurations
    - e.g. EMACs
  - Contexts
    - contexts are a way to specify states in which a command is available
    - e.g when editing text
  - Commands
    - represents a request from the user
  - Key bindings
    - represents a binding between a command and a key sequence

 © 2005 by IBM; made available under the EPL v1.0

Command Definition

```
<extension point="org.eclipse.ui.commands">
    <category
       description="Work with images for the Advanced Tutorial"
       name="Images"
       id="org.eclipsecon.tutorials.advanced.imageCategory"/>
    <command
        description="Allows setting of filter options for images"
        categoryId=
           "org.eclipsecon.tutorials.advanced.imageCategory"
        name="Image Filter Settings"
        id="org.eclipsecon.tutorials.advanced.imagesFilter"/>
   <keyBinding
       contextId="org.eclipse.ui.contexts.dialogAndWindow"
       commandId="org.eclipsecon.tutorials.advanced.imagesFilter"
        keySequence="M1+M2+Alt+I"
        keyConfigurationId=
        "org.eclipse.ui.defaultAcceleratorConfiguration"/>
</extension>
```

TIP: When defining a key sequence it is better to use M1 (instead of Ctrl) and M2 (instead of Shift) for better portability across platforms

 © 2005 by IBM; made available under the EPL v1.0

Questions and Comments?

 © 2005 by IBM; made available under the EPL v1.0

Filters and Sorters

- Viewers can have filters and sorters
- Allows changes to a viewer without having to ask for contents again
- We want to sort the contents alphabetically
- Want also want to reduce the shown elements to just those that are images
- We used a viewer filter and a viewer sorter

TIP: The sorters and filters are a nice way for a reusable view to modify the contents of the view without having to ask for elements again. This is used a lot by TableViewers which sort by column.

 © 2005 by IBM; made available under the EPL v1.0

Advanced Preferences

- Now we have a command, set the filter preferences in a dialog
- Our view is unaware of the preference change, but we want to update the view contents
- We can add a preference listener to handle this
- First define a `PreferenceInitializer` to be run when the plug-ins preference store is first accessed
  - use the `org.eclipse.core.runtime.preferences` extension point to define the initializer
  - pre 3.0 we set these up on plug-in startup but this is inefficient if the preference is never referenced and adds to startup time
- Now add a preference listener to the view so the refresh will occur as soon as the preference changes

 © 2005 by IBM; made available under the EPL v1.0

Editor Actions

- We defined an editor in the foundations tutorial (`ImageEditor`)
- The editor actions extension point defines an action that is only added to a menu or toolbar when the editor has focus
- Useful for keeping the clutter down (action sets do not go away)
- Also has the active editor passed to it from the workbench
- We reuse the menu from our action set by using the same path name
  - `imagesMenu/imagesGroup`

 © 2005 by IBM; made available under the EPL v1.0

Questions and Comments?

 © 2005 by IBM; made available under the EPL v1.0

Wizards

- The workbench and IDE have three wizard types defined in extension points
  - New
  - Import
  - Export
- When one of the wizard actions are invoked we open a WizardDialog which provides a selection page (generated internally from the extension points) to choose one of the wizards defined in the extension points

 © 2005 by IBM; made available under the EPL v1.0

New Wizards

- New wizards have a series of re-usable first pages
  - `BasicNewResourceWizard` (abstract)
  - `BasicNewFileResourceWizard`
  - `BasicNewFolderResourceWizard`
  - `BasicNewProjectResourceWizard`
- Use these in your wizards so that the resource is already created for you

 © 2005 by IBM; made available under the EPL v1.0

Import/Export Wizards

- Write a wizard to import image files …ImageImportWizard
- Several classes are provided to assist In building an import or export wizard
  - WizardResourceImportPage / WizardResourceExportPage
    - Provides basic layout for source and destination
    - Provides an options group
  - FileSystemStructureProvider
    - provides structure for the ImportOperation based on java.io.File
  - ImportOperation/ExportOperation
    - Basic operations for performing imports and exports

 © 2005 by IBM; made available under the EPL v1.0

Perspective Extensions

- View and editors are layed out in perspectives
  - e.g. Resource or Java perspectives
- We can add to an existing perspective using the perspectiveExtension extension point
- Create a perspectiveExtension to add the `ImageNavigator` to the resource perspective
- Restart then reset the resource perspective
- See the `ImageNavigator` there

 © 2005 by IBM; made available under the EPL v1.0

Perspectives

- Can also define our own perspectives
- Use a `PageLayout` positions parts and setup other standard menus etc…
- Create an “Image Navigation” perspective
  - Define our own `IPerspectiveFactory`
  - See `org.eclipsecon.tutorials.advanced.perspectives.ImagePerspectiveFactory`
  - Add our view in the Page Layout
  - Add the TasksView as a fast view
  - Add the Navigator View to the short list of views

 © 2005 by IBM; made available under the EPL v1.0

eclipse CON 2005

Capabilities

- A way to filter the user interface for large applications so as not to overwhelm the user with too much functionality
- The IDE has a series of trigger points that will enable a capability
  - i.e. project creation or loading
- Uses regular expressions to map to the ids of extension to filter
- Applied to:
  - views perspectives, preference pages, etc.
- See Kim Hornes talk on Wednesday 10:30 AM

TIP: Enabling capabilities should happen seamlessly for the user. We provide a preference page but this is really only useful for developers to test their definitions.

 © 2005 by IBM; made available under the EPL v1.0

Our Capability Definition

```
<extension
        point="org.eclipse.ui.activities">
     <activity
           description="The activity for the editing of images"
           name="Edit Images"
           id="org.eclipsecon.tutorials.advanced.imageActivity"/>
     <activityPatternBinding
           activityId="org.eclipsecon.tutorials.advanced.imageActivity"
           pattern="org\.eclipsecon\.tutorials\..*/.*"/>
     <defaultEnablement id="org.eclipsecon.tutorials.advanced.imageActivity"/>
     <category
           description="The activities for this example"
           name="Image Activities"
           id="org.eclipsecon.tutorials.advanced.category"/>
     <categoryActivityBinding
           activityId="org.eclipsecon.tutorials.advanced.imageActivity"
           categoryId="org.eclipsecon.tutorials.advanced.category"/>
  </extension>
```

 © 2005 by IBM; made available under the EPL v1.0

Questions and Comments?

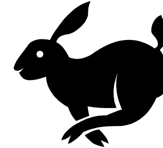
Startup and Shutdown

- Lazy create image caches and dispose on shutdown
- Want to minimize the amount of work done here as the workbench window is not visible or usable
- The IDE tries to avoid loading other plug-ins by allowing all required information to be defined in the xml of an extension point

TIP: Plug-in loading should be avoided in general but it is particularly noticeable when it slows down startup. You can avoid this by a lazy initialization approach whenever possible.

 © 2005 by IBM; made available under the EPL v1.0

Jobs



- SWT and JFace must be accessed in the main Thread
- Without creating Jobs or Threads Eclipse is single Threaded (referred to as the UI or main Thread)
- We want to process long operations that do not require SWT in a background Thread using a Job
- We are moving image loading to a background job
- Progress reporting is done in the progress area and the progress view

TIP: Any work done in a Job does not block the main Thread which makes the UI more responsive as it does not stop the user from working

 © 2005 by IBM; made available under the EPL v1.0

Questions and Comments?

© 2005 by IBM; made available under the EPL v1.0

Decorators

- Annotations to the text or image of entries in a Viewer to provide extra information
- The workbench has an `IDecoratorManager` which applies the decorations defined in the decorators extension point
- Using a `DecoratingLabelProvider` allows a view to hook into the `IDecoratorManager`
- Uses jobs to calculate decorations so it does not block
- Can be done declaratively – no plug-in activation ever!
- Optionally can define a class if it is too complex to define declaratively
- We will add a declarative decoration on the top left for all resources

 © 2005 by IBM; made available under the EPL v1.0

IAdaptable

- What is it?
  - IAdaptable is a mechanism for accessing an Object of one type via another Object of a different type without using interfaces

```
if (object instanceof IAdaptable) {//See if it supports IAdaptable

    //See if it adapts to IFile
    Object resource = ((IAdaptable) object).getAdapter(IFile.class);

    if (resource == null)//Returns null if not
            return;

    //Now we can safely cast to IFile and continue
    doFileOperation((IFile) resource);
}
```

 © 2005 by IBM; made available under the EPL v1.0

IAdaptable

- Why do you care?
  - this is a good way to give access to an object without defining additional interfaces
    - JDT uses this for CompilationUnits (IFile), Packages (IFolder) and Java Projects (IProject)
- Do we need it here?
  - no - we are using resources directly
- Decorators can be adaptable
  - if you want to annotate both the resource and the object that wraps it this is useful
  - CVS decorators are adaptable to IResource so JDT objects are decorated by the CVS decorator as they adapt to IResource

 © 2005 by IBM; made available under the EPL v1.0

Common Expressions

- Common Expressions are a way of defining enablement rules in XML for a variety of extensions
- Prevents plug-in loading as they can be done declaratively
- Used by:
  - decorators
  - editorActions
  - popup menus
  - viewActions
- Simple example using the decorators
  - want to only decorate files, not folders or projects
  - just using the objectClass tag and some logic

 © 2005 by IBM; made available under the EPL v1.0

Our Common Expression

Can you simplify this expression further?

```
<enablement>
   <and>
     <objectClass name="org.eclipse.core.resources.IResource"/>
     <not>
        <or>
          <objectClass name="org.eclipse.core.resources.IProject"/>
          <objectClass name="org.eclipse.core.resources.IFolder"/>
        </or>
     </not>
   </and>
</enablement>
```

 © 2005 by IBM; made available under the EPL v1.0

Help

- A link between widgets and the help system
- Create your help html document
- Create an xml file that maps topics to your html document

```
<context id="tutorial_context">
  <description>Go to the information about the Image Navigator
  </description>
  <topic label="Tutorial Help" href="doc/imageNavigator.htm" />
</context>
```

 © 2005 by IBM; made available under the EPL v1.0

Help

- Define a helpContext extension point that refers to the xml file

```
<contexts file="contexts_Tutorial.xml"
      plugin="org.eclipsecon.tutorials.advanced"/>
```

- Use WorkbenchHelp to refer to one of the ids in your xml file

```
getWorkbench().getHelpSystem().setHelp(getControl(),
          AdvancedTutorialPlugin.getDefault().getBundle().
          getSymbolicName()   + "." + "tutorial_context");
```

- Test it out by hitting F1 in the IDEImageNavigator

 © 2005 by IBM; made available under the EPL v1.0

The End

- Questions?

© 2005 by IBM; made available under the EPL v1.0