Eclipse 4.x: Tips on API best practices for a 3.x plugin running on both platforms
About the Speakers

- Paul Webster
  - Platform UI Team Lead
  - Eclipse 4 and the 4.x Workbench
  - Platform CBI
Overview

- Everybody loves API

- Usage Patterns
  - Command Patterns
  - Part Management
  - Interaction with the Framework
Everybody loves API

/**
 * Asks this part to take focus within the workbench. Parts must
 * assign focus to one of the controls contained in the part's
 * parent composite.
 * <p>
 * Clients should not call this method (the workbench calls this method at
 * appropriate times). To have the workbench activate a part, use
 * <code>IWorkbenchPage.activate(IWorkbenchPart) instead</code>.
 * </p>
 */

def setFocus();
A command is an abstraction of some semantic behaviour.
A command is not an implementation of that behaviour.
A command is not the visual presentation of that behaviour.
Commands – activating a handler

- Provide handler declaratively using org.eclipse.ui.handlers
- Provide handlers programmatically using `getSite().getService(IHandlerService.class)`
- Do not use `Command.setHandler(IHandler)`
Commands – executing a handler

- Always use the IHandlerService
- Methods for executing a handler and executing specific context
- Do not use Command.executeWithChecks(ExecutionEvent)
Handlers – caching

- Handlers should not carry state
  - Get it from the event with HandlerUtil
  - Get it from the Command
  - Get it from the “framework”

```java
public Object execute(ExecutionEvent event) throws ExecutionException {
    IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
    ICountService service = (ICountService) window.getService(ICountService.class);
    System.out.println("New value: "+service.incrementCount());
    return null;
}
```
Parts – accessing services

- Access a service locally if available
  - It can scope its functionality
  - It can cleanup

```java
private void activateContext() {
    IContextService contextService = (IContextService) getSite()
        .getService(IContextService.class);
    contextService.activateContext(VIEW_CONTEXT_ID);
}
```
Parts – Parent Composite assumptions

- Never assume anything about the Composite that is provided to you

```java
@Override
public void createPartControl(Composite parent) {
    Composite composite = new Composite(parent, SWT.NONE);
    composite.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, true));
    /* create the rest of my UI */
}
```

- `createPartControl(Composite)`'s contract does not say that the parent Composite's layout is a GridLayout (bug 342806)
Parts - Parent Composite assumptions

```java
@Override
public void createPartControl(Composite parent) {
    Composite composite = new Composite(parent, SWT.NONE);
    composite.setLayoutData(new GridLayout());
    initializationBtn = new Button(composite, SWT.PUSH);
    initializationBtn.setText("Initialize...");
    /* create the rest of my UI */
}

public void initializationCompleted() {
    GridData gridData = (GridData) initializationBtn.getLayoutData();
    gridData.exclude = false;
    initializationBtn.setVisible(false);
}
```

- Some layouts will define a layout data for a child control for free
- Layout calls may not happen when you think it will (bug 342806)
- Set layout data explicitly
Parts - Caching of values

```java
private Shell partShell;

@Override
public void createPartControl(Composite parent) {
    partShell = parent.getShell();
}

public void warn(String title, String message) {
    MessageDialog.openInformation(partShell, title, message);
}
```

- Looks pretty innocuous
- Why would a Part's Shell change?
Parts - Caching of values

- Parts were originally always created under the workbench window first even if detached
- No longer the case in 4.x for detached windows, the model is respected at rendering time
- Caching the shell spells death if the part is reattached to the main workbench window (bug 334512)
Parts - setFocus()

- A method that needs to be implemented by IWorkbenchPart implementations
- Implementing this incorrectly can cause strange problems that are difficult to debug
- Usually a runtime thing, getting the method wrong shouldn't bring down Eclipse
Parts - setFocus()

```java
@Override
public void setFocus() {
    control.getShell().setFocus();
    control.setFocus();
}
```

- Infinite loops (bug 330386)
Parts - setFocus()

```java
@ Override
public void setFocus() {
    final Thread setFocusThread = new Thread() {
        @Override
        public void run() {
            final IWorkbench wb = PlatformUI.getWorkbench();
            final IWorkbenchWindow window = wb.getActiveWorkbenchWindow();
            final IWorkbenchPage page = window.getActivePage();
            if (page != null) {
                page.activate(SampleView.this);
            }
        }
    };
    // We need to switch to the thread that is allowed to change the UI
    Display.getDefault().syncExec(setFocusThread);
}
```

- Infinite loops, uh...again (bug 344261)
Parts – view toolbars

- IActionBars can be used to directly change the view toolbar.
- Especially handy for quick additions or a variable number of items
- Always use IActionBars.updateActionBars()

```java
if (extrasSupported) {
    IActionBars actionBars = getViewSite().getActionBars();
    IToolBarManager toolBarManager = actionBars.getToolBarManager();
    toolBarManager.add(getUserAnalyzeCommand());
    actionBars.updateActionBars();
}
```
Framework – downcasting objects

- Avoid downcasting objects
- Internals are avoided for obvious reasons
- This happens with the workbench window
- This happens with the view toolbar

```java
isFVBCONFIGURED = ((WorkbenchWindow) workbench.getActiveWorkbenchWindow()).getShowFastViewBars();
```
Framework - Preferences

- Underlying preference technology is the same
- Using it at the Workbench level or the Equinox level

WorkbenchPlugin.getDefault().getPreferenceStore().
  .getBoolean(IPreferenceConstants.REUSE_EDITORS_BOOLEAN);
InstanceScope.INSTANCE.getNode("org.eclipse.ui.workbench").getBoolean(
  IPreferenceConstants.REUSE_EDITORS_BOOLEAN, false);
Framework – using services

- Services allow interaction with the framework
- The workbench is hierarchical
- Avoid workbench.getProgressService()
- Use getSite().getService(IProgressService.class)
  - IWorkbenchSiteProgressService
  - IPartService
  - IPageService
  - ISelectionService
Eclipse 4 – why these patterns work

- It's all about keeping your part isolated
  - Try not to 'know' too much about your container

- Don't make assumptions about the overall UI Structure

- This frees you up to take advantage of 4.x's coming ability to use a part in new locations (i.e. Dialogs, Wizards, Pref pages...)
More Information

- You can get general information from the wiki:

- You can contact us and ask questions:
  - On IRC: irc://freenode.net/#eclipse
  - On the newsgroup: http://www.eclipse.org/forums/
    Eclipse Projects > Eclipse Platform
    Eclipse Projects > Rich Client Platform
Give Feedback on the Sessions

1. Sign In: www.eclipsecon.org

2. Select Session Evaluate

3. Vote +1 0 -1
Legal Notices

- Copyright © IBM Corp., 2007-2013. All rights reserved. This presentation and the source code in it are made available under the EPL, v1.0.

- Written with Eric Moffat and Remy Suen

- Eclipse and the Eclipse logo are trademarks of Eclipse Foundation, Inc.

- IBM and the IBM logo are trademarks or registered trademarks of IBM Corporation, in the United States, other countries or both.

- Other company, product, or service names may be trademarks or service marks of others.

- THE INFORMATION DISCUSSED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, AND IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, SUCH INFORMATION. ANY INFORMATION CONCERNING IBM'S PRODUCT PLANS OR STRATEGY IS SUBJECT TO CHANGE BY IBM WITHOUT NOTICE