

Developing for the Rich Client Platform

Turning your application in to a “Really Cool Product”

Nick Edgar
Pascal Rapiçault

IBM Rational Software
Ottawa

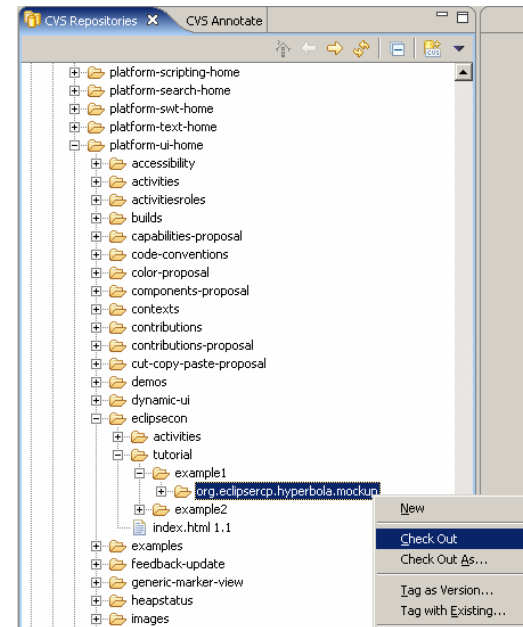
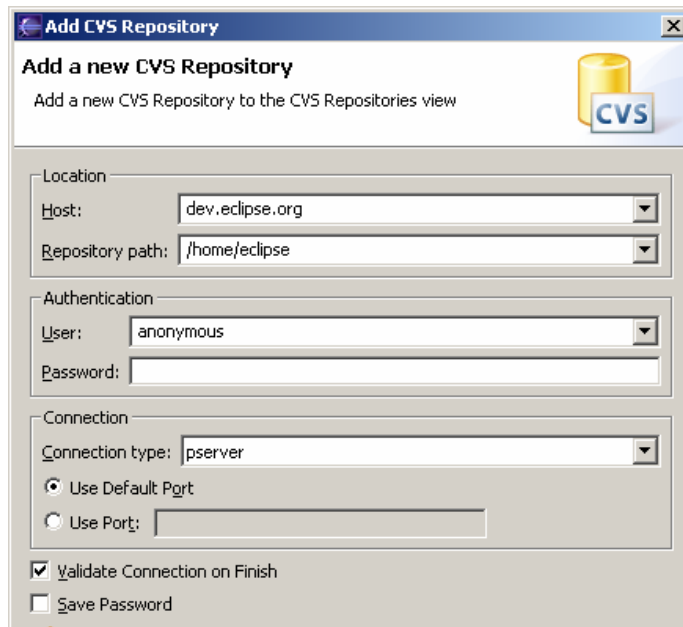
Eclipse Platform Committers
To contact us: news://news.eclipse.org/eclipse.platform.rcp

Your Infrastructure

- To get the most from this tutorial, you need to bring a laptop capable of running Eclipse and developing plug-ins.
- Before the tutorial, you need to have the following software loaded on your machine:
 - Eclipse SDK 3.1M5 <http://www.eclipse.org/downloads>
 - Eclipse RCP SDK 3.1M5
 - Eclipse RCP Delta Pack 3.1 M5
 - A Jabber client <http://www.jabber.org>
- Eclipse should be installed and running.
- The RCP SDK and RCP Delta pack should be kept as zip.
- Your jabber client should be installed and running with an account.

Loading the Hyperbola Example 1 code from CVS

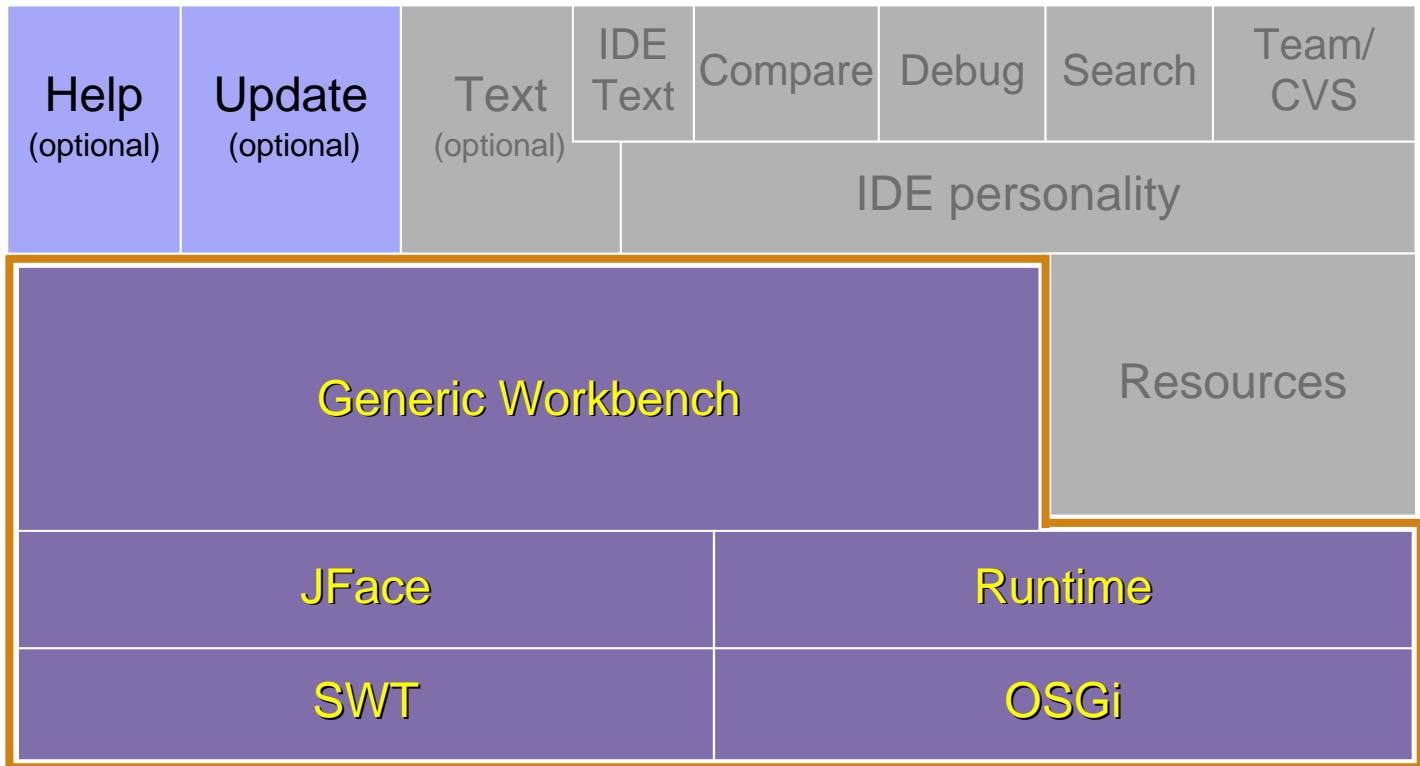
- Window > Open Perspective > Other... > CVS Repository Exploring
- In CVS Repositories view context menu: New > Repository Location
- Enter **host:** dev.eclipse.org **repo path:** /home/eclipse **user:** anonymous **password:** <none> **connection type:** pserver
- In CVS Repositories view, expand:
 HEAD/platform-ui-home/eclipsecon/tutorial/example1
- Select **org.eclipse.rcp.hyperbola.mockup** and choose Check Out



What is eclipse RCP ?

- “Really Cool Platform”?
- “Rich Client Platform”
A Platform for building Client applications with Rich functionality
- History
 - Hackers in the eclipse community started building non-IDE apps on 2.1
 - In 3.0 we made this real by factoring out the IDE aspects from the workbench

What is in RCP?



RCP by the numbers (as of 3.1 M5)

- Download size : ~ 5 Mb
- Unzipped size : ~ 6 Mb

- VM size of a Java hello world (not eclipse-based): ~ 8 Mb
- VM size of an headless eclipse-based hello world: ~ 9 Mb

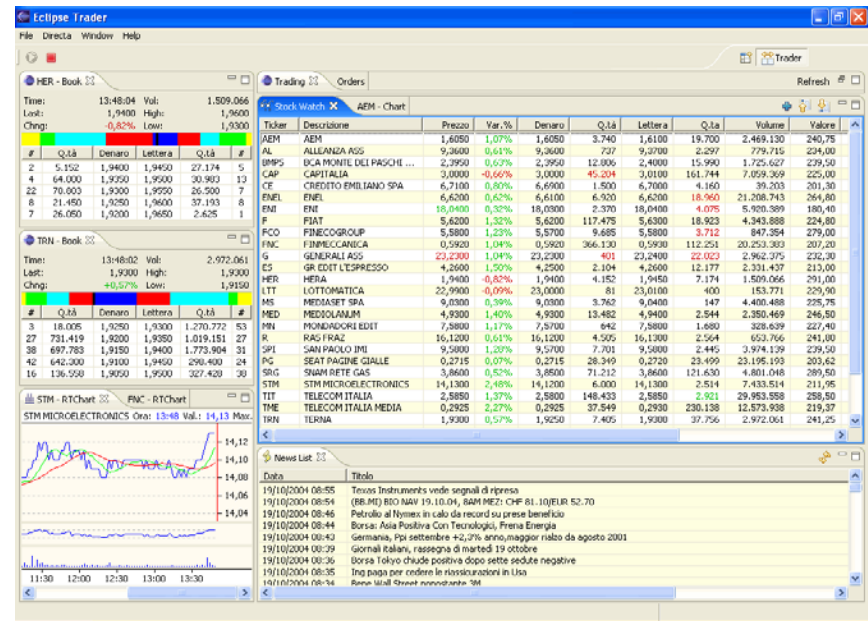
- Time before your application code runs: ~ 650ms

- Number of plug-ins in the RCP binary : 10

Where can you encounter Eclipse RCP?



IBM Lotus Workplace



Eclipse Trader

Reference

- EclipseCon'05: Eclipse RCP Everywhere (J.-M. Lemieux, J. McAffer)

Outline

- **Section I – Fundamental RCP concepts**
 - Eclipse application
 - RCP application
 - Workbench advisors
 - Simple branding
 - (Exercise)
- **Section II – The workbench specifics**
 - Window / page
 - Perspective
 - View / Editors
 - Actions, key bindings
 - Communication across parts
 - Access to the application model
 - (Exercise)
- **Section III – Extra RCP plug-ins, advanced topics**
 - Help
 - Update / provisioning
 - Advanced branding
 - Third party libraries
 - Modularity, integration and extensibility

Section I – Fundamental RCP concepts

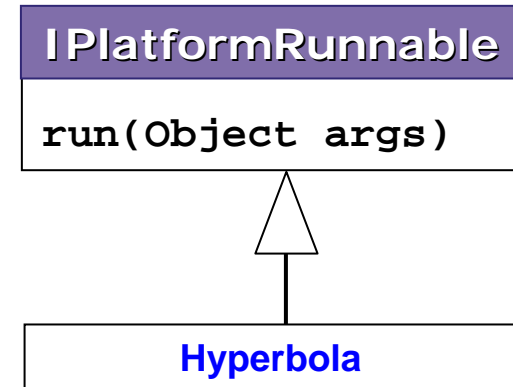
What is an eclipse application?

1. An extension to the `org.eclipse.core.runtime.applications` extension point.

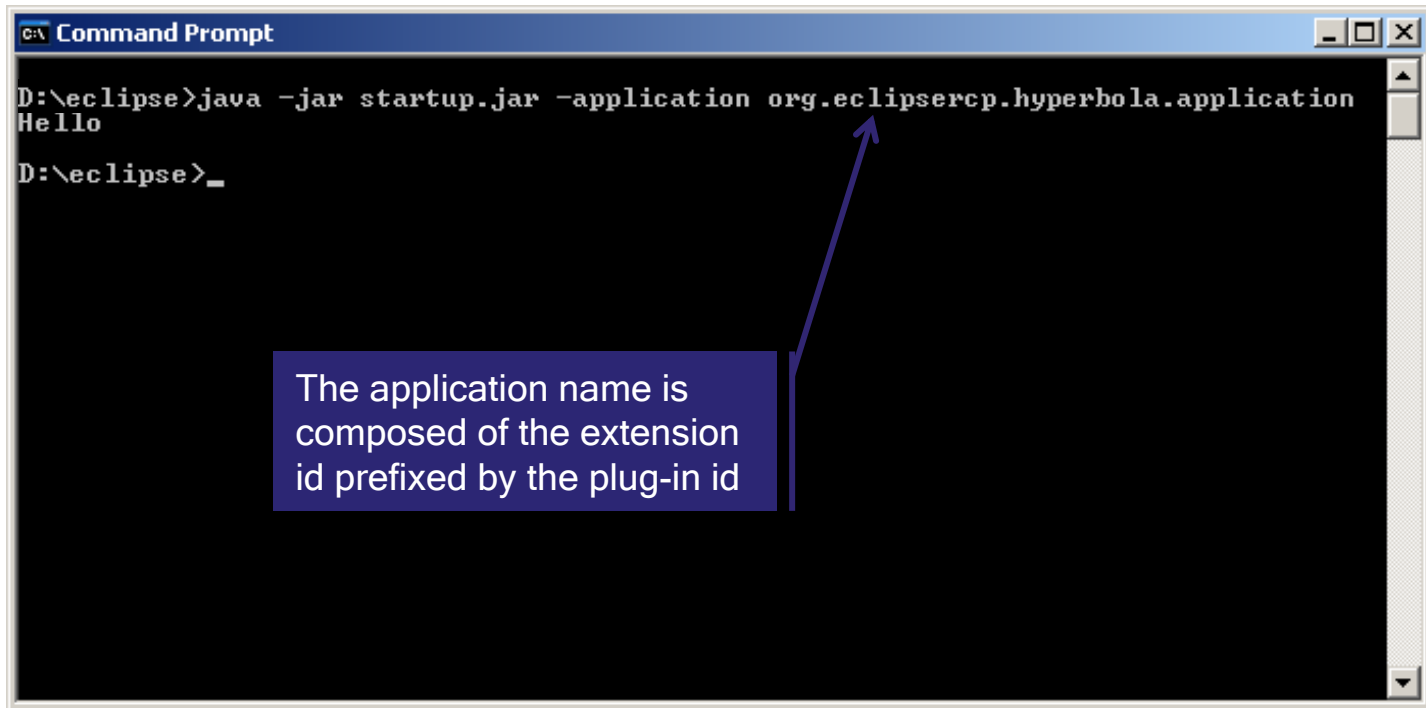
```
<extension id="application"
           point="org.eclipse.core.runtime.applications">
  <application>
    <run class="hyperbola.Hyperbola"/>
  </application>
</extension>
```

2. A class implementing `IPlatformRunnable`. It is the main.

```
public class Hyperbola implements IPlatformRunnable {
  public Object run(Object args) throws Exception {
    System.out.println("Hello");
  }
}
```



Headless application



```
c:\ Command Prompt
D:\eclipse>java -jar startup.jar -application org.eclipsercp.hyperbola.application
Hello
D:\eclipse>_
```

The application name is composed of the extension id prefixed by the plug-in id

My first RCP application

⇒ **An RCP application is an eclipse application where the workbench runs the main event loop.**

Overall life cycle of the application

```
public class Hyperbola implements IPlatformRunnable {
    public Object run(Object args) throws Exception {
        Display d = null;
        try {
            d = PlatformUI.createDisplay();
            WorkbenchAdvisor wa = new MinimalAdvisor();
            PlatformUI.createAndRunWorkbench(d, wa);
            return new Integer(0);
        } finally {
            if (d != null)
                d.dispose();
        }
    }
}
```

Create the display

create a workbench advisor

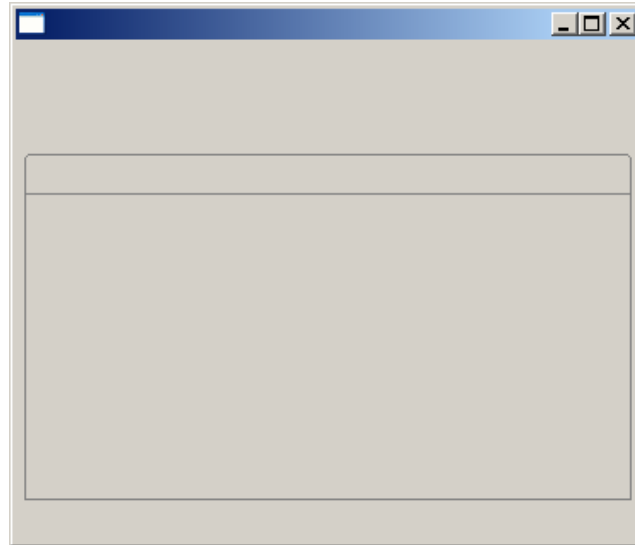
Run the workbench

Application return code

Dispose the display

Configuring the Workbench and its Windows

- WorkbenchAdvisor, WorkbenchWindowAdvisor, ActionBarAdvisor
 - *strategy* objects to configure the workbench and its windows
 - provide hook methods called at strategic points during the workbench life cycle



The WorkbenchAdvisor and its configurer

⇒ Advises on how the workbench should be configured

```

public class HyperbolaWorkbenchAdvisor extends WorkbenchAdvisor {

    public void initialize(IWorkbenchConfigurer configurer) {
        super.initialize(configurer);

        configurer.setSaveAndRestore(true);
    }

    public String getInitialWindowPerspectiveId() {
        return HyperbolaPerspective.ID;
    }

    public WorkbenchWindowAdvisor createWorkbenchWindowAdvisor(
        IWorkbenchWindowConfigurer configurer) {
        return new HyperbolaWorkbenchWindowAdvisor(configurer);
    }
}

```

Enable the workbench state save mechanism

Defines the initial perspective

Factory method for the window advisor

The WorkbenchWindowAdvisor and its configurer

⇒ Advises on how the window should be configured

The screenshot shows a window titled "Hyperbola" with a menu bar (Hyperbola, Contacts, View, Tools, Help), a toolbar with icons and a search box, a central area with a "Click to Sign On" button, and a status bar at the bottom with a refresh icon and an "Offline" indicator. Arrows point from callout boxes to these specific UI elements.

Title
I WorkbenchWindowConfigurer
. setTitle()

Menu bar
I WorkbenchWindowConfigurer
. setShowMenuBar()

Cool bar
I WorkbenchWindowConfigurer
. setShowCoolBar()

Status bar
I WorkbenchWindowConfigurer
. setShowStatusLine()

Note Not shown here:
I WorkbenchWindowConfigurer
setShowPerspectiveBar()
setShowFastViewBars()
setShowProgressIndicator()
Shell size and style

The WorkbenchWindowAdvisor and its configurer (sample)

```

public HyperbolaWorkbenchWindowAdvisor extends WorkbenchWindowAdvisor {

    HyperbolaWorkbenchWindowAdvisor(IWorkbenchWindowConfigurer configurer) {
        super(configurer);
    }

    public void preWindowOpen() {
        IWorkbenchWindowConfigurer configurer = getWindowConfigurer();
        configurer.setTitle("Hyperbola");
        configurer.setInitialSize(new Point(275, 475));
        configurer.setShowProgressIndicator(true);
        configurer.setShowPerspectiveBar(false);
    }

    public ActionBarAdvisor createActionBarAdvisor(
        IActionBarConfigurer configurer) {
        return new HyperbolaActionBarAdvisor(configurer);
    }
}

```

Layout and appearance settings

Factory method for the action bar advisor

The ActionBarAdvisor and its configurer

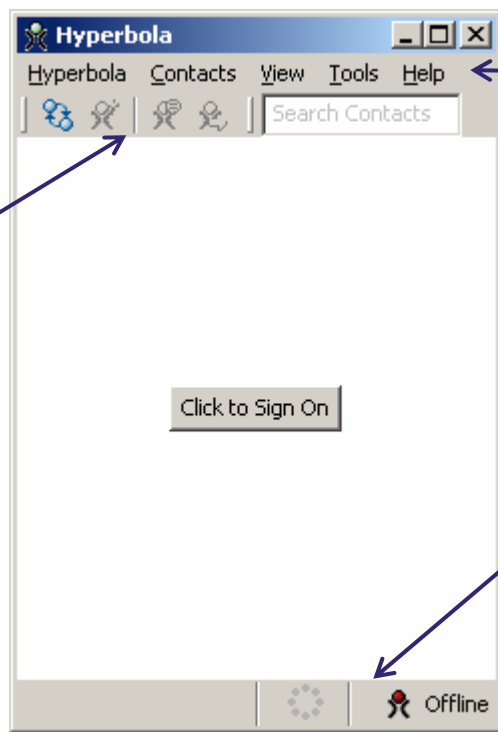
⇒ Create actions and use them in the various bars

Creation of actions

`makeActions()`

Cool bar

`fillCoolBar()`
`ICoolBarManager`



Menu bar

`fillMenuBar()`
`IMenuBar`
`IMenuManager`

Status bar

`fillStatusLine()`
`IStatusLine`
`IStatusLineManager`

The ActionBarAdvisor and its configurer (sample)

```

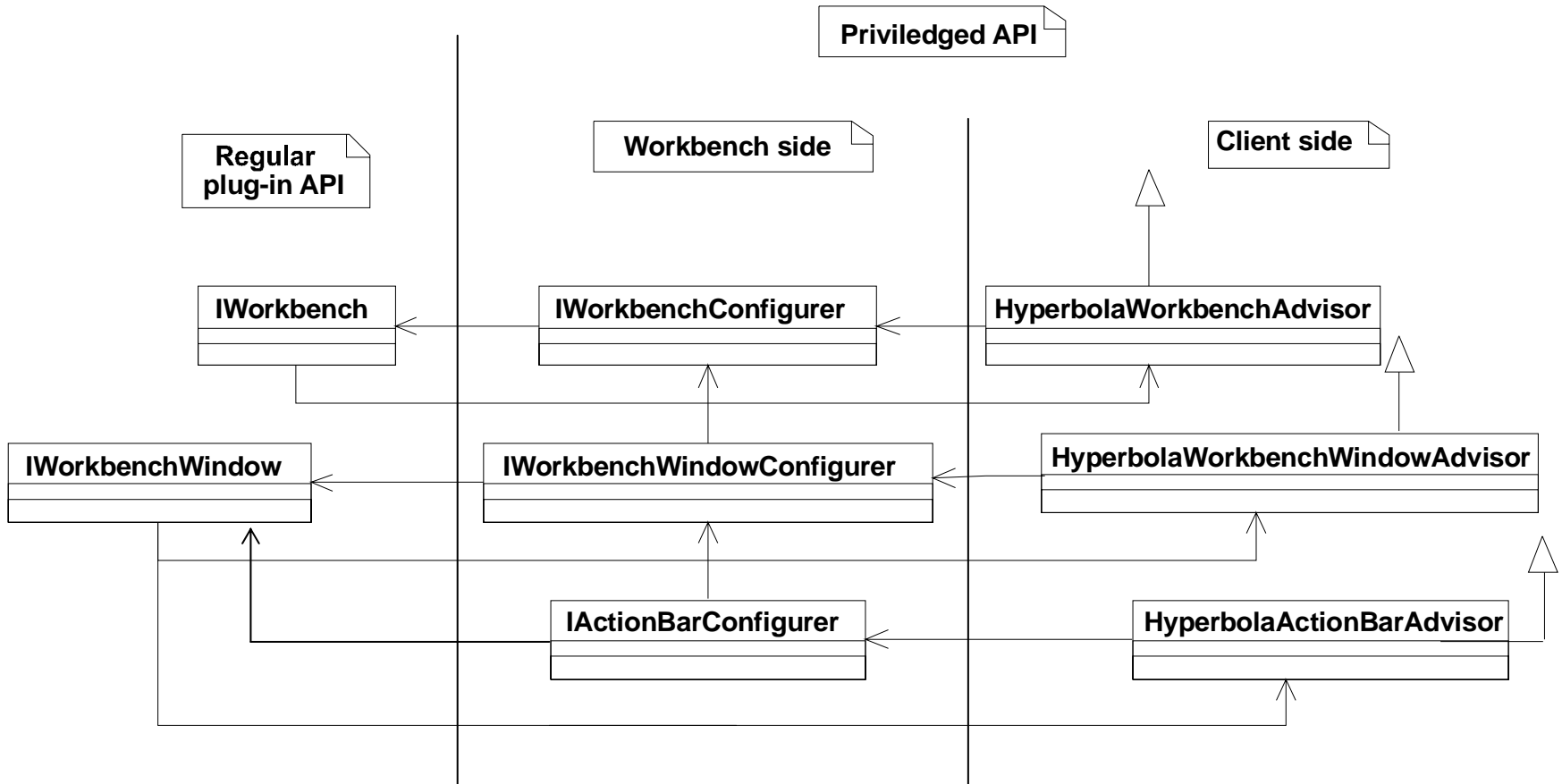
public class HyperbolaActionBarAdvisor extends ActionBarAdvisor {
    private IWorkbenchAction quitAction;

    public HyperbolaActionBarAdvisor(IActionBarConfigurer configurer) {
        super(configurer);
    }

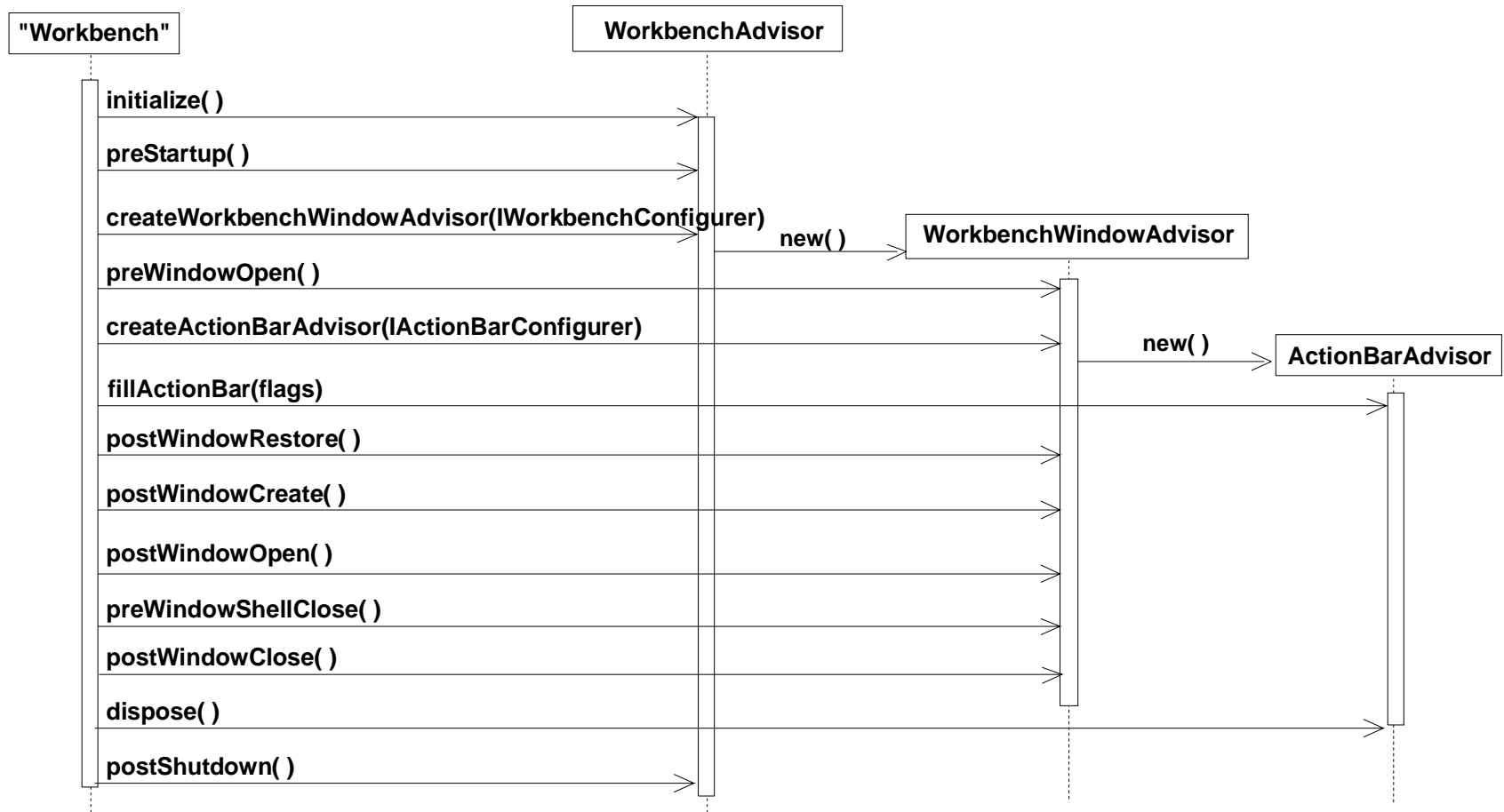
    protected void makeActions(IWorkbenchWindow window) {
        quitAction = ActionFactory.QUIT.create(window);
        register(quitAction);
    }

    protected void fillMenuBar(IMenuManager menuBar) {
        IMenuManager fileMenu;
        fileMenu = new MenuManager("&Hyperbola", IWorkbenchActionConstants.M_FILE);
        menuBar.add(fileMenu);
        fileMenu.add(quitAction);
    }
    ...
  
```

Structure of the advisors and configurers



Workbench and advisors life cycle (partial)



The shell is available as of `postWindowCreate()`

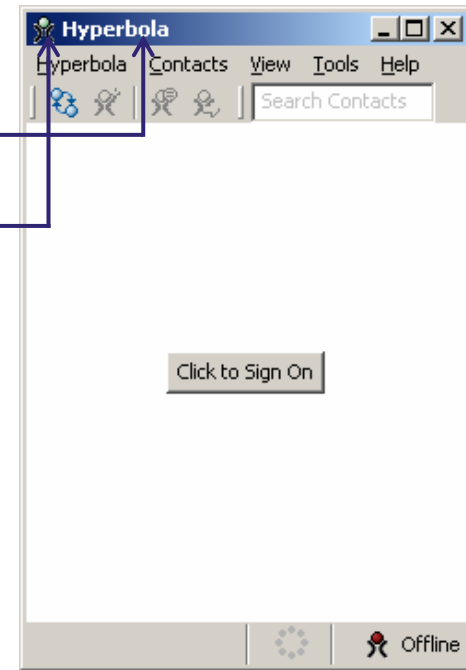
Branding – the basics

⇒ Branding consists in giving your product the colors of your company

Extension-point: org.eclipse.core.runtime.products

```
<extension id="hyperbolaProduct" point="org... products">
  <product name="Hyperbola"
    application="org.eclipse.rcp.hyperbola.application">
    <property name="windowImage" value="online.gif" />
    ...
  </extension>
```

Default application to execute



Branding – the basics (cont'd)

About dialog

```
<property name="aboutImage"
  value="image.gif" />
```

```
<property name="aboutText"
  value="Hyperbola Chat client" />
```



Default preferences (in a java property file)

```
<property name="preferenceCustomization"
  value="plugin_customization.ini" />
```

In `plugin_customization.ini` file:

```
org.eclipse.ui /SHOW_TRADITIONAL_STYLE_TABS=false
org.eclipse.ui /SHOW_TEXT_ON_PERSPECTIVE_BAR=false
```

References

- Product editor in Eclipse 3.1
- Eclipse.org article: Branding your application (A. Eisdness and P. Rapicault)

The config.ini

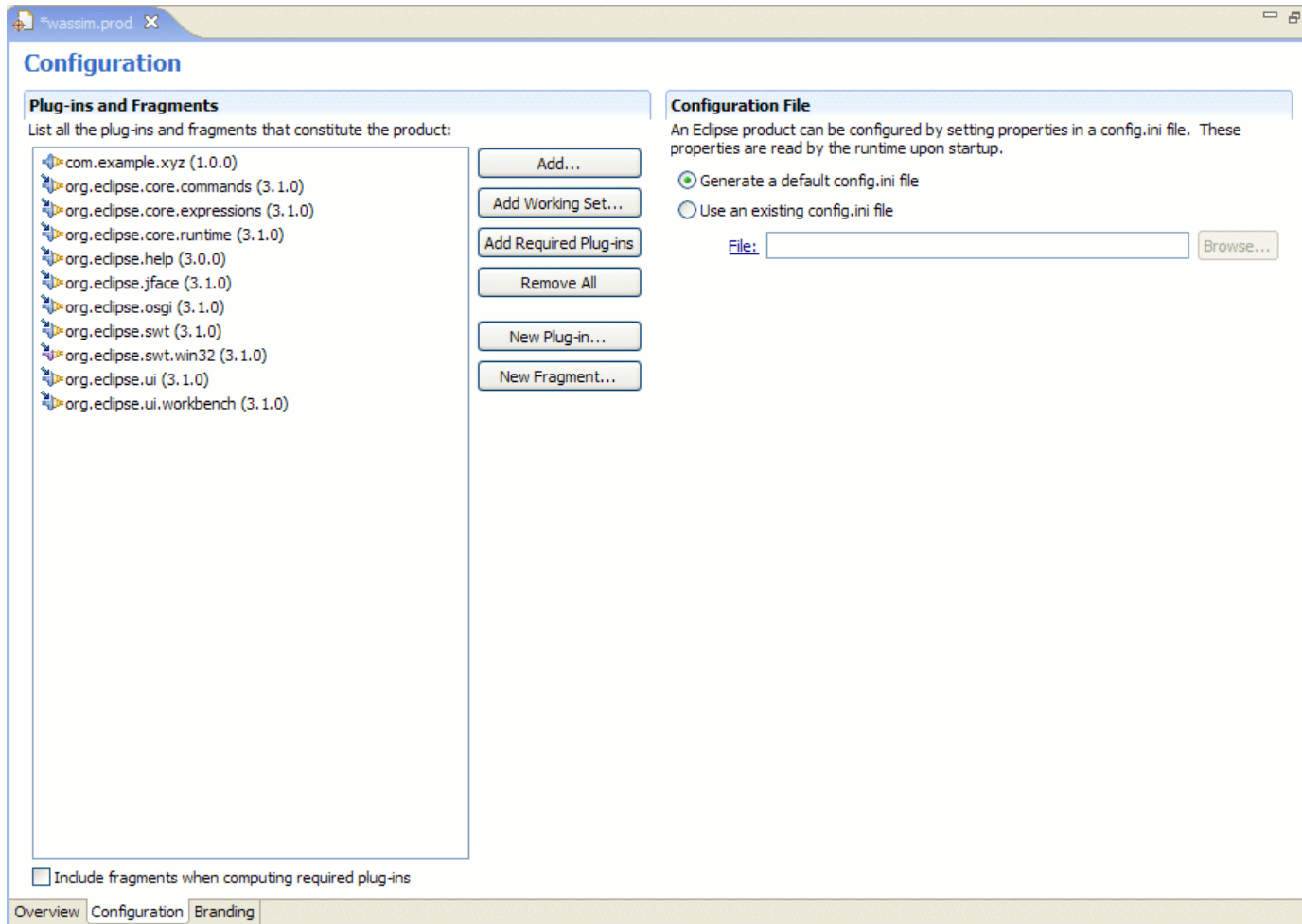
- Controls many aspects of the platform (locations, start level, ...)
- Used in the branding to declare the splash screen and the product to run
- Java properties file read on startup and merged with the system properties

```
# The product to run
eclipse.product = org.eclipse.rcp.hyperbola.product

# The splash screen to display
org.osgi.splashPath = platform:/base/plugins/org.eclipse.rcp.hyperbola

# The list of bundles to run
org.osgi.bundles = org.eclipse.core.runtime@2: start, org.eclipse.ui,
                  org.eclipse.swt, org.eclipse.swt.win32, ...
```

Producing an RCP application – the product editor



Layout of an RCP application in the file system

hyperbola/

hyperbola.exe
hyperbola.ini

Branded eclipse.exe and its optional partner file allowing to set VM and VM arguments

configuration/
config.ini

Controls the product to run, the bundles to run, the splash screen, and other aspect of the platform

plugins/
<plug-ins from the rcp base>
<hyperbola plug-ins>

Your plug-ins and their prerequisites

jre/

Optional JRE

Alternate layout of an RCP application in the file system

hyperbol a/

hyperbol a. exe
hyperbol a. i ni

confi gurati on/
confi g. i ni

hyperbol a-content/
ecl i pse/
pl ugi ns/

<hyperbol a pl ug-i ns>

Your plug-ins

ecl i pse/
l i nks/
hyperbol a. l i nk

Link file whose content refers to the root of the folders containing an eclipse folder

pl ugi ns/
<pl ug-i ns from the RCP base>

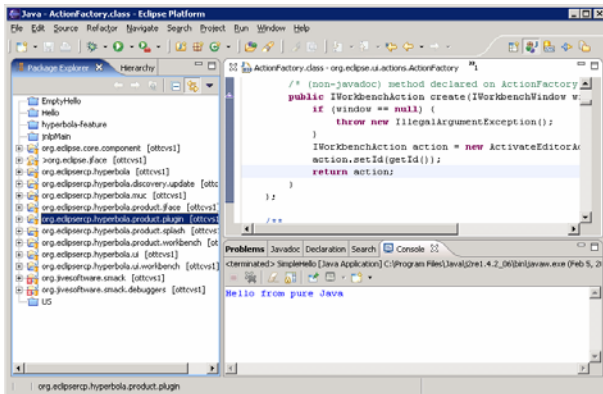
RCP plug-ins

j re/

This layout allows for multiple plug-in folders

Self-hosting concepts – exercise warm-up

Workspace



Target (Preferences > PDE > Target platform)

```
d: /RCPSDK/
eclipse/
plugins/
<plugins from the rcp base>
<plugins from the rcp delta pack>
```

+

=

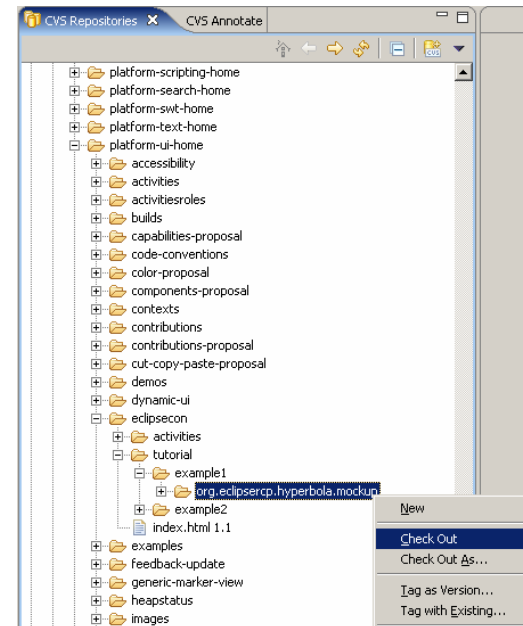
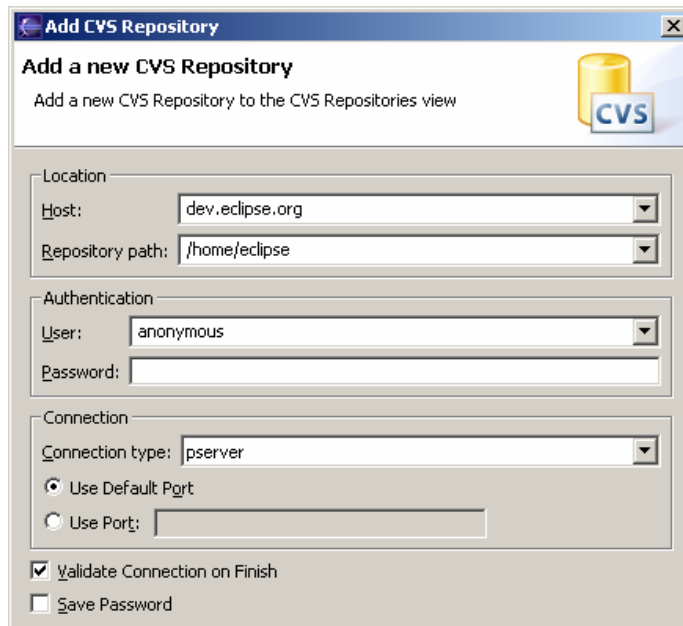
Your eclipse application (runtime-workbench)

Note

- This can be further controlled using the plug-ins tab in the launch configuration

Loading the Hyperbola Example 1 code from CVS

- Window > Open Perspective > Other... > CVS Repository Exploring
- In CVS Repositories view context menu: New > Repository Location
- Enter **host:** dev.eclipse.org **repo path:** /home/eclipse **user:** anonymous **password:** <none> **connection type:** pserver
- In CVS Repositories view, expand:
 HEAD/platform-ui-home/eclipsecon/tutorial/example1
- Select **org.eclipse.rcp.hyperbola.mockup** and choose Check Out



Exercise

- Create the application class
- Show the toolbar and the status bar
- Create a product and brand it
- Export your product
 - For your configuration
 - For another configuration

Grouping plug-ins using features

- ⇒ Features provide a mechanism for grouping plug-ins
- ⇒ Features capture platform-specific information

feature.xml

```

<feature id="org.eclipse.rcp"
        label="Eclipse RCP feature"
        version="3.1.0">

    <plugin id="org.eclipse.swt" version="3.1.0"/>

    <plugin id="org.eclipse.swt.win32" version="3.1.0"
            os="win32" ws="win32" arch="x86"/>

    <plugin id="org.eclipse.swt.gtk" version="3.1.0"
            os="linux" ws="gtk" arch="x86"/>

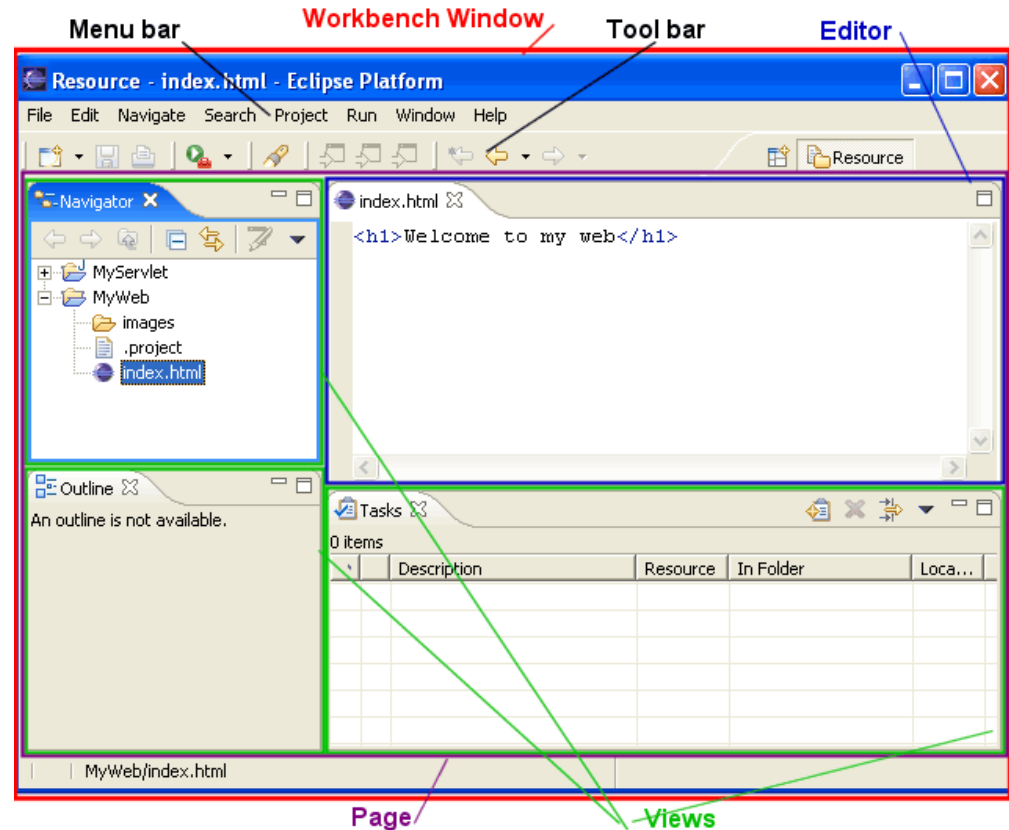
    [...]
</feature>
  
```

Note

- Originally features have been designed to be a delivery mechanism for the eclipse update system, however, here we are adapting their convenient abstraction. Those features do not appear in the end product

Section II – The workbench specifics

Workbench structure



Workbench (1)

Window (0 – N)

Page (0 – 1)

Perspective (0 – N), 1 active

View (0 – N)

Editor (0 – N)

Window / Page

⇒ A window contains a page, which contains an arrangement of views and editors whose layout is defined by a perspective.

Appearance

```
IWorkbenchWindowConfigurer
```

Controlling (opening/closing)

```
IWorkbench.openWorkbenchWindow()  
IWorkbenchWindow.close()
```

Introspection / tracking

```
org.eclipse.ui.IWorkbench.getActiveWorkbenchWindow()  
org.eclipse.ui.IWorkbench.getWorkbenchWindows()  
org.eclipse.ui.IWindowListener / workbench.addWindowListener()  
org.eclipse.ui.IPageListener (org.eclipse.ui.IPageService)  
org.eclipse.ui.IWorkbenchWindow.getActivePage()
```

Perspective

⇒ Controls the layout of views and editors

- Defined programmatically in the perspective factory (IPerspectiveFactory)

```

public void createInitialLayout(IPageLayout layout) {

    String editorArea = layout.getEditorArea();
    layout.setEditorAreaVisible(false);
    IFolderLayout left = layout.createFolder("left", IPageLayout.LEFT,
                                             0.25f, editorArea);
    left.addView(MyView.VIEW_ID);
}
  
```

- Perspective extensions (org.eclipse.ui.perspectiveExtensions)
 - Allow plug-ins to extend perspectives

Perspective – control and lifecycle

Controlling

```
IWorkbench.showPerspective()  
IWorkbench.openWorkbenchWindow()  
IWorkbenchPage.setPerspective()  
IWorkbenchWindow.close()
```

Introspection / tracking

```
IWorkbenchPage.getPerspective()  
IWorkbenchPage.getOpenPerspectives()  
org.eclipse.ui.IPerspectiveListener3
```

Saving

```
IWorkbenchPage.savePerspective()
```

Reference

- Eclipse.org article: [Using Perspectives in the Eclipse UI](#) (D. Springgay)

Views

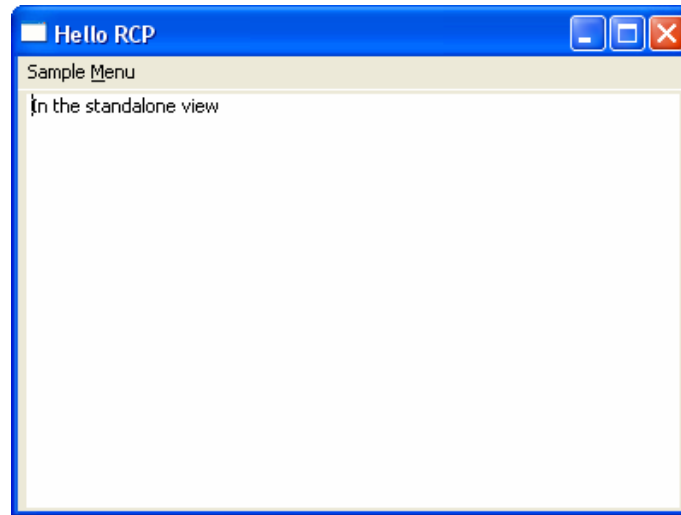
⇒ A part of the workbench page outside of the editor area

Typical roles for views:

- Allow overall navigation of the application's model (e.g. the Package Explorer)
- Support navigation in the editor (e.g. Outline)
- Provide additional context-sensitive info (e.g. Properties)

Views – standalone

⇒ A view that can't be docked with others



Declared in the perspective factory

```

layout.addStandaloneView(HelloRCPView_ID,
    false,
    IPageLayout.TOP,
    .95f,
    layout.getEditorArea());
    
```

Id of the view

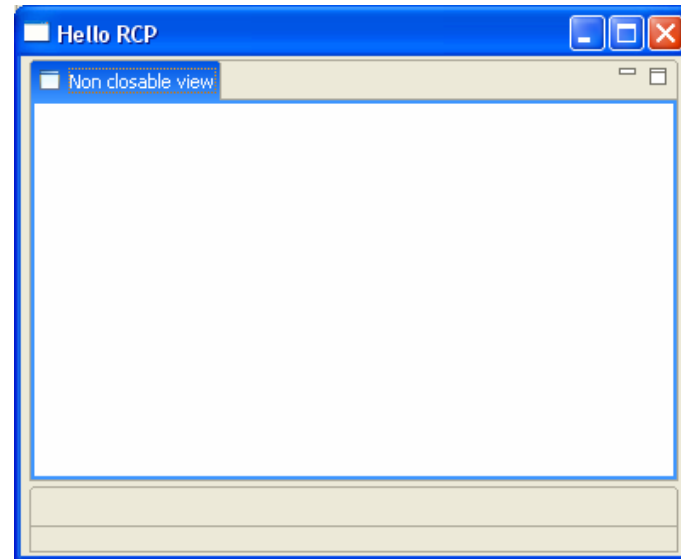
Flag to show the title bar

Views – non-moveable

Declared in the perspective factory

```
public void createInitialLayout(IPageLayout layout) {  
    layout.addView(View1.ID_VIEW1, IPageLayout.TOP,  
                  IPageLayout.RATIO_MAX,  
                  IPageLayout.ID_EDITOR_AREA);  
  
    IViewLayout viewLayout = layout.getViewLayout(View1.ID_VIEW1);  
  
    viewLayout.setMoveable(false);  
}
```

Views – non-closeable

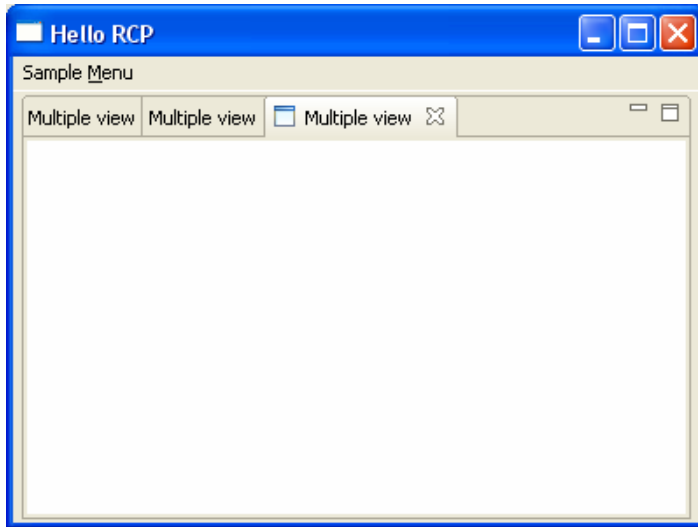


Declared in the perspective factory

```
public void createInitialLayout(I PageLayout layout) {
    layout.addView(View1.ID_VIEW1, I PageLayout.TOP,
        I PageLayout.RATIO_MAX,
        I PageLayout.ID_EDITOR_AREA);

    I ViewLayout viewLayout = layout.getViewLayout(View1.ID_VIEW1);
    viewLayout.setCloseable(false);
}
```

Views – multiple instance



In the plugin.xml

```
<extension point="org.eclipse.ui.views">
  <view allowMultiple="true"
        class="demoViews.View1"
        name="Non closable view"
        id="DemoViews.view1"/>
</extension>
```

When opening the view

```
window.getActivePage().showView(View1.ID_VIEW1,
var,
```

The secondary id of the view

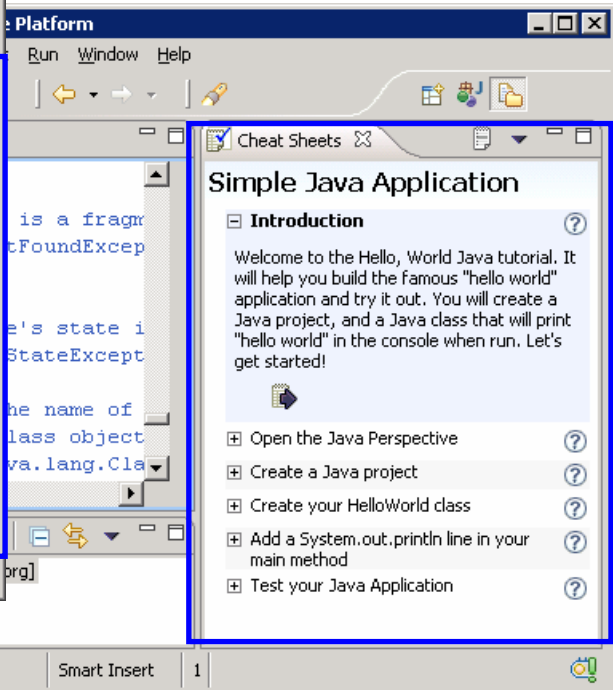
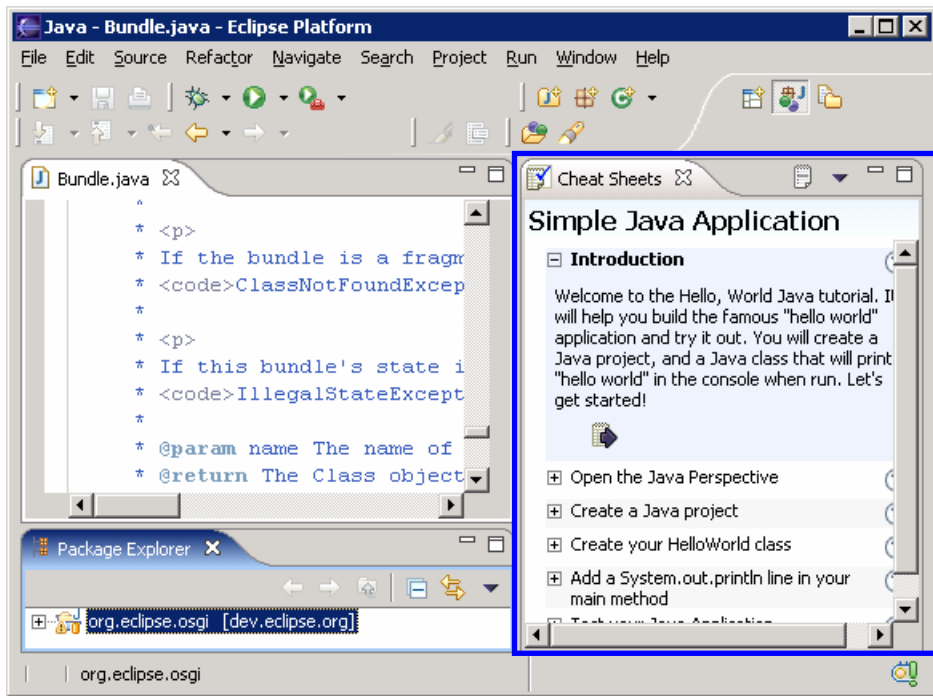
```
IWorkbenchPage.VIEW_VISIBLE);
```

Views – sticky

⇒ A view that stays open across perspective switches

In the plugin.xml

```
<stickyView id="id.myview"
  location="RIGHT"
  closeable="true"
  moveable="false" />
```



Views – positioning (single instance case)

⇒ Controlling the position of the view when it opens

Declared in the perspective factory

```
I FolderLayout folder = layout.createFolder("demoViews", I PageLayout. TOP,  
0.5f, layout.getEditorArea());  
folder.addPlaceholder("DemoViews.view1");
```

ID of the views that will
be opened in this folder



Views – positioning (multiple instance case)

⇒ Controlling the position of the view when it opens

Declared in the perspective factory

```
I FolderLayout folder = layout.createFolder("demoViews", I PageLayout. TOP,
                                             0.5f, layout.getEditorArea());
folder.addPlaceholder("DemoViews.view1: *");
```

ID of the views that will
be opened in this folder

Pattern matching the
secondary ID

Views – title bar

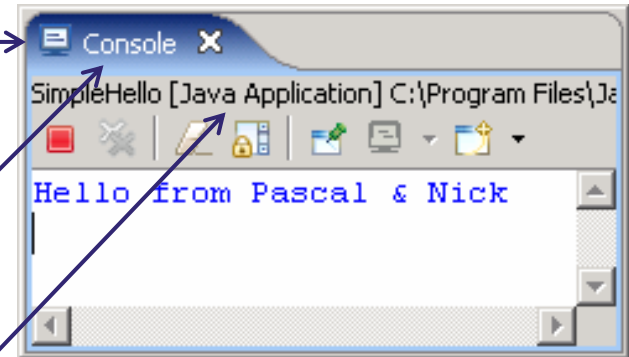
- Content of the title bar – (org.eclipse.ui.part.WorkbenchPart – ViewPart)

```
setPartName()
setTitleToolTip()
setTitleImage()
setContentDescription()
```

Title image

Part name

Content description



- Indicate things are happening in the view – only through jobs

```
getSite().getAdapter(IWorkbenchSiteProgressService.class)
```

Notes

- To hide the title bar, use a standalone view.
- To have a title bar that goes across the whole view, use a standalone view.

Views – control and lifecycle

Controlling

```
IWorkbenchPage. showView()  
IWorkbenchPage. hideView()  
IWorkbenchPage. bringToTop()
```

Tracking

```
org.eclipse.ui.IPartListener2  
org.eclipse.ui.IPartService
```

Introspection

```
IWorkbench. getViewRegistry()  
IWorkbenchPage. getViewReferences()  
IWorkbenchPage. findViewReference()  
IWorkbenchPage. findView()  
IPartService. getActivePart()  
IPartService. getActivePartReference()  
IWorkbenchPartSite. getId()  
IViewSite. getSecondaryId()
```

Saving

- To save the widgets state
 ViewPart. saveState()
 ViewPart. restoreState()
- To save the underlying model
 ISaveablePart

Reference

- Eclipse.org article: Creating an Eclipse View (D. Springgay)

Editors

Files not required !!!

References

- API: `org.eclipse.ui.part.EditorPart` and `org.eclipse.ui.IEditorInput`
- Article: http://www.jroller.com/page/Zhou/20040215#eclipse_editors_not_tied_to

Editors – title bar

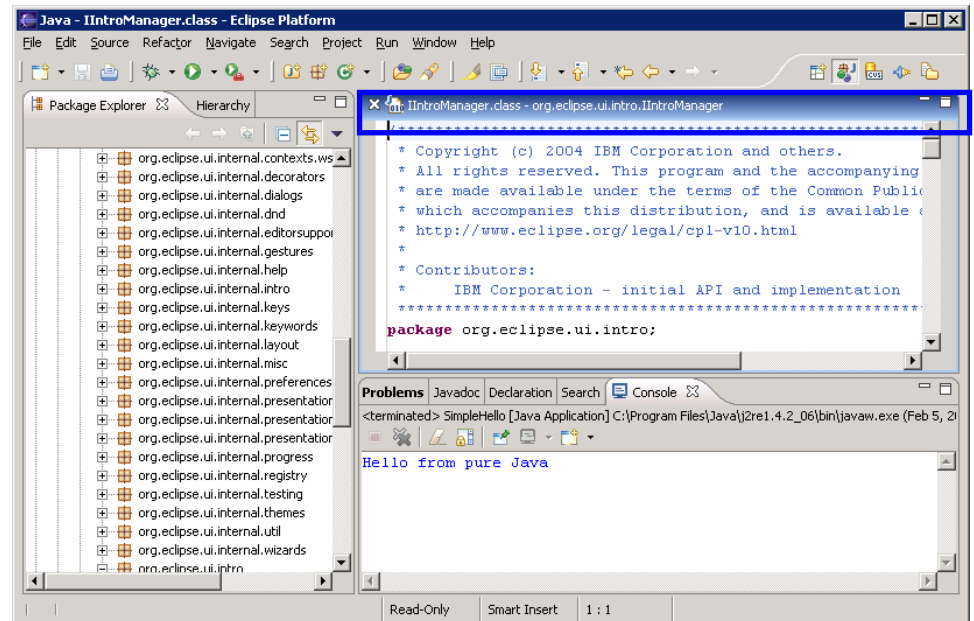
- Content of the title bar – (org.eclipse.ui.part.WorkbenchPart - EditorPart)

```

setPartName()
setTitleToolTip()
setTitleImage()
setContentDescription()
    
```

- Title bar spanning the whole editor area

See bug 84706



Editors – control & lifecycle

Controlling

```
IWorkbenchPage.closeAllEditors()  
IWorkbenchPage.closeEditor()  
IWorkbenchPage.closeEditors()  
IWorkbenchPage.bringToTop()  
IWorkbenchPage.openEditor()
```

Introspection

```
IWorkbenchPage.getActiveEditor()  
IWorkbenchPage.getDirtyEditors()  
IWorkbenchPage.getEditorReferences()  
IWorkbenchPage.findEditor()  
IWorkbenchPage.getDirtyEditors()  
IEditorReference.isDirty()  
IWorkbenchPartSite.getId()
```

Tracking

```
org.eclipse.ui.IPartListener2
```

Editors – misc.

- Saving / restoring editors state
 - Editor input should be persistable (implement `IPersistableElement`)
 - Restoration of the content should be lazy and the editor input should be lightweight
- Hiding/showing the editor area after the fact
 - `IWorkbenchPage.setEditorAreaVisible(boolean)`
- Misc.
 - `EditorInput` must implement `equals()` and `hashCode()`

References

- Plug-ins: `org.eclipse.ui.workbench.texteditors` and its prerequisites
- Example: http://www.eclipse.org/rcp/main.html#text_editor_example
- Other plugins: `org.eclipse.ui.forms`, `org.eclipse.gef`

Editors or Views

	Editors	Views
Position	Primary area	Around primary area
Open/save life cycle	Yes	No*
Input object	Yes	No*
Contribute to main menu	Yes	No*
Rearrangeable	No	Yes

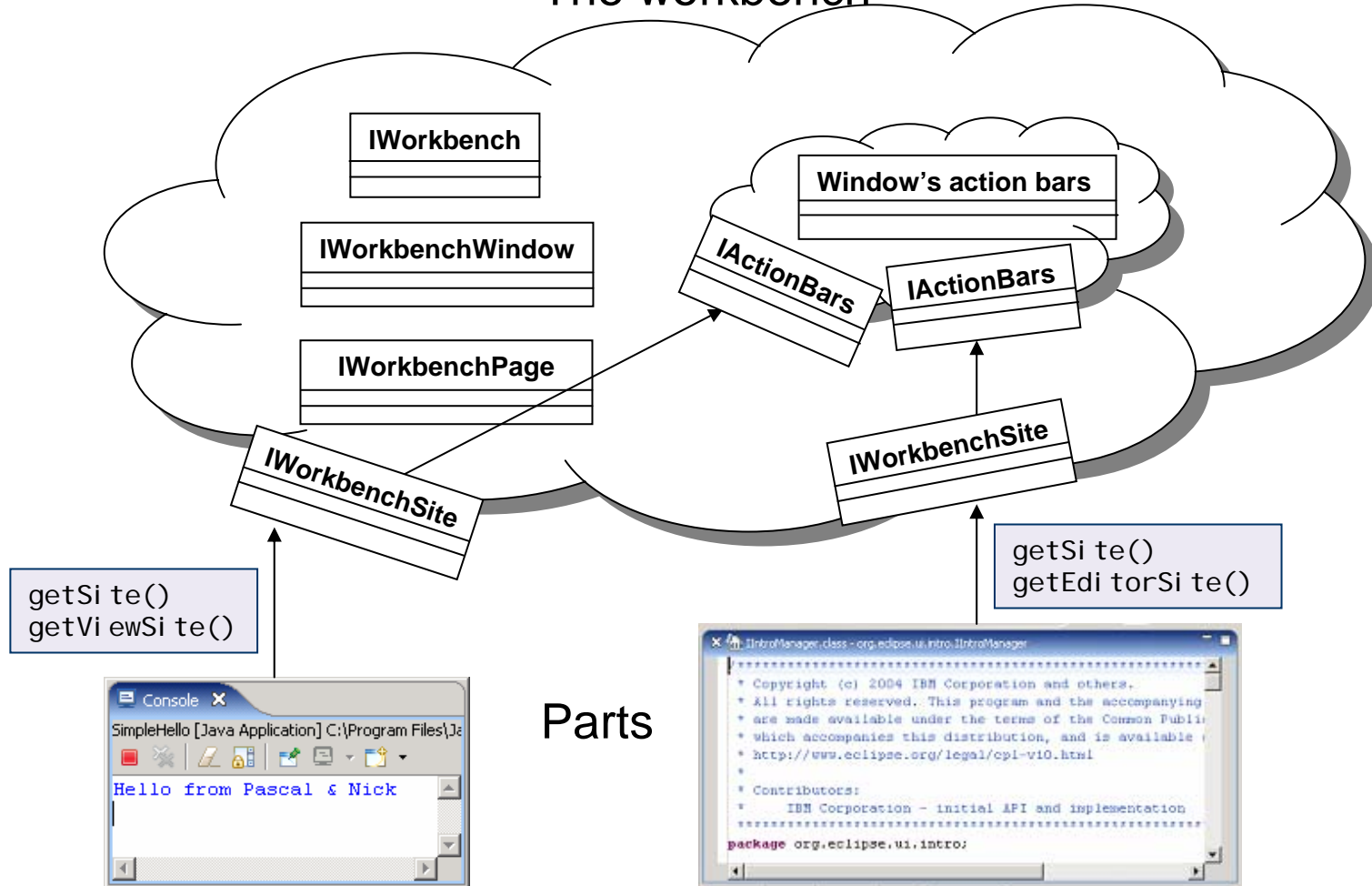
*: It is an out-of-the-box no, however see the section on views for solutions

Reference

- <http://www.eclipsefaq.org/chris/faq/html/faq206.html>

Workbench, parts and sites

The workbench



Actions

⇒ An action abstracts the differences between a menu item and a toolbar item

Makes it easier to reorganize the UI

- Has behavior: `run()`
- Describes its own enablement (enabled property)
- Describes its own presentation attributes (text, image, tooltip)

Actions (sample)

```

class AddContactAction extends Action {
    public AddContactAction(Session s) {
        setId("hyperbola.action.addContact");
        setText("Add &Contact...");
        setEnabled(s.isConnected());
        //After that you want to track the connection state
    }
    public void run() {
        //Prompt for contact details
    }
}

```

Action id

Presentation attributes

Enablement logic

Retargetable actions

⇒ It allows the current part to provide the behavior (handler) for an action added elsewhere (e.g. cut, copy, paste).

```
public static final ActionFactory ADD_CONTACT =
    new ActionFactory("hyperbola.action.addContact"){

    public IWorkbenchAction create(IWorkbenchWindow window) {
        RetargetAction action = new RetargetAction(getId(), "Add &Contact...");

        action.setToolTipText("Add Contact");
        action.setImageDescriptor(...);

        window.getPartService().addPartListener(action);

        action.setActionDefinitionId("hyperbola.command.addContact")

        return action;
    }
};
```

Presentation attributes

Listener needed for enablement logic

Note

- Text and tool tip can be overridden by the handler if the action implements Label RetargetAction

Retargetable actions – hooking a handler

```
class RosterView extends ViewPart {
    IAction addContactAction;

    void hookActions() {
        addContactAction = new AddContactAction(getSession());
        getViewSite().getActionBars().setGlobalActionHandler(
            "hyperbola.action.addContact", addContactAction);
    }
}
```

Note

- Handlers can only be hooked from parts.

Declarative actions

⇒ They add onto menus and toolbars previously defined by the application, editors and views

- Extension points: `org.eclipse.ui.actionSets`, `...editorActions`, `...viewActions`, `...popMenus`
- Give flexibility for progressive exposure to functionality.
- Can be associated with perspectives and parts, filtered by activities, etc.
- Code only run when action selected. Enablement rules can be defined declaratively. Allows lazy plug-in activation.
- Restriction: they are not designed to build on each other.

Note

- The new commands support in 3.1 relaxes this restriction

Declarative actions (sample)

```

<extension point="org.eclipse.ui.actionSets">
  <actionSet
    label="Contacts"
    id="org.eclipse.rcp.hyperbola.contactsActionSet">

    <menu
      id="hyperbola.menu.contacts"
      label="& Contacts"
      path="additions"/>

    <action
      id="hyperbola.action.addContact"
      definitionId="hyperbola.command.addContact"
      label="Add & Contact"
      icon="icons/add_contact.gif"
      retarget="true"
      menubarPath="hyperbola.menu.contacts/additions"
      toolbarPath="mainToolBar/contacts"/>

  </actionSet>
</extension>

```

Action id

Command id

Presentation attributes

Location in the UI

Actions, programmatically or declaratively?

- **Define a minimal skeleton programmatically in the advisor.**
 - Include only the menus and actions that will always be visible.
- Define action sets for other contributions.
- For views and editors, define their actions programmatically, but allow for extension.
- When views need to add to main menus and toolbars, use `org.eclipse.ui.actionSets` and `org.eclipse.ui.actionSetPartAssociations`.

Note

- Action sets can enable progressive exposure and enhance UI stability due to the flexibility of perspective- and part-associations.

Reference

- Eclipse.org article: [Contributing Actions to the Eclipse Workbench](#) (S. Arsenault)

Key bindings

⇒ Key bindings associate a key sequence with a command.

- To associate an action with a command, use

```
IAction.setActionDefinitionId(String commandId)
```

- Actions added in code must be registered with the key binding service.

```
addContactAction = JabberActionFactory.ADD_CONTACT.create(window);  
register(addContactAction);
```

Helper method on the
ActionBarAdvisor

- Registration is done automatically for actions added declaratively

Key bindings (sample)

```
<extension point="org.eclipse.ui.commands">
```

```
  <category
    name="Hyperbola"
    description="Hyperbola commands"
    id="hyperbola.commands"/>
```

```
  <command
    name="Add a contact"
    description="Add a contact to your roster"
    categoryId="hyperbola.commands"
    id="hyperbola.command.addContact"/>
```

← Command id

```
  <keyConfiguration
    name="Hyperbola Key Configuration"
    description="Hyperbola Key Configuration"
    id="hyperbola.commands.keyConfiguration"/>
```

```
  <keyBinding
    commandId="hyperbola.command.addContact"
    keySequence="CTRL+N"
    keyConfigurationId="hyperbola.commands.keyConfiguration"/>
```

Communication across parts

Direct communication

- View to view
 - Use the APIs to open and hide view
- View to editor
 - Use the APIs to open editor with the appropriate input

Tracking the selection

- To publish:
 - `IWorkbenchSite.setSelectionProvider(ISelectionProvider)`
 - `IWorkbenchPart.getSelection()`
- To subscribe:
 - `ISelectionService` and `ISelectionListener`
 - `IWorkbenchPage` extends `ISelectionService`
 - `IWorkbenchWindow.getSelectionService()`

Tracking part lifecycle and activation

- To subscribe:
 - `IPartService` and `IPartListener[2]`
 - `IWorkbenchPage` extends `IPartService`
 - `IWorkbenchWindow.getPartService()`

Tracking window lifecycle and activation

- To subscribe:
 - `IWindowListener`
 - `IWorkbench.addWindowListener(IWindowListener)`

Accessing the application Model

From Views

- When the view is opened programmatically

```
HelIoView view = (HelIoView) page.showView(...);
view.setInput(...)
```

- When the view is opened by the user
 - Access the model in createControlPart()
 - (i.e.: get the current selection)
 - For views with multiple instances: keep a map of secondary id to input

From an Action

- Track the current selection
- Take an instance from the model in the constructor

From Editors

- Give input when opening the editor
 - openEditor

```
page.openEditor(myInput, "hyperbolic.editor")
```

Static Fields

- For example the resources plug-in: ResourcesPlugin.getWorkspace()

Saving / restoring the application state

You must save your model yourself!

- Workbench
 - Need to explicitly enable workbench state persistence (see `IWorkbenchConfigurer`)
 - Preferences should be saved explicitly on change
 - Editors and views, see the appropriate sections

Be Lazy

Locations – where can I store my data?

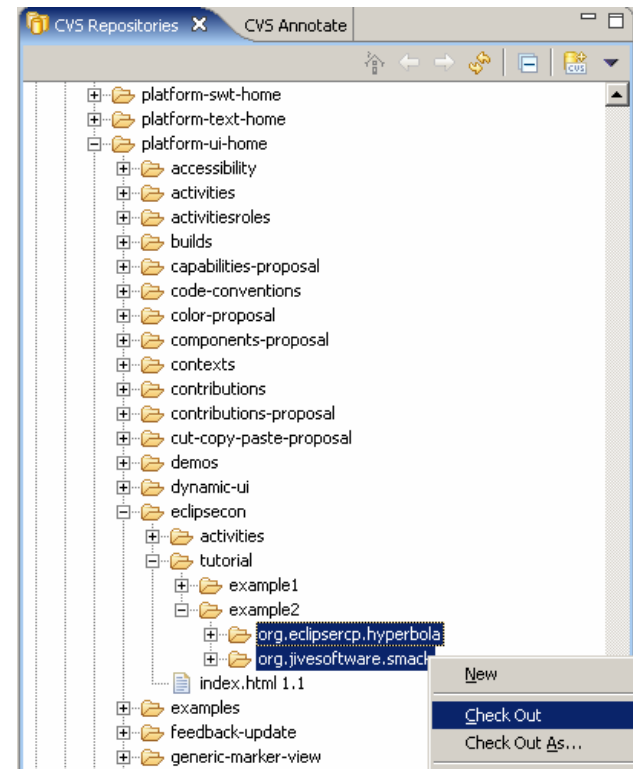
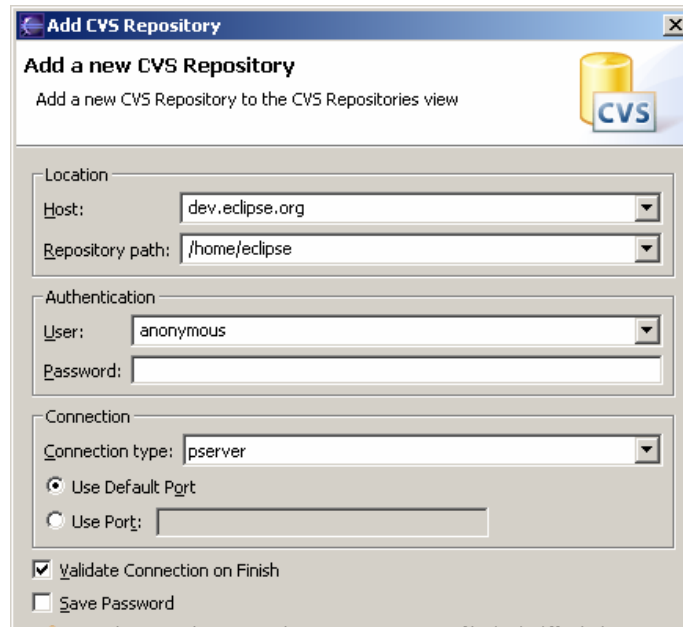
- Instance** application data (PI at form. getInstanceLocation())
- Writeable, controlled by the application.
 - osgi . instance . area
- Configuration** configuration being run (PI at form. getConfigurati onLocati on())
- Writeable. May have a parent location.
 - osgi . confi gurati on . area
 - osgi . sharedConfi gurati on . area
- User** data specific to a user (PI at form. getUserLocati on())
- osgi . user . area
- Special values**
- @none, @noDefaul t, @user . home, @user . di r

Notes

- The eclipse install folder is accessible from PI at form. getInstal l Locati on() and is read only. The associated system property is osgi . i nstal l . area.
- Information read-only and lock from the Location objects are advisory only.
- Default values can be specified in the config.ini

Loading the Hyperbola Example 2 code from CVS

- Back in the CVS perspective,
- In CVS Repositories view, expand:
 HEAD/platform-ui-home/eclipsecon/tutorial/example2
- Check Out **org.eclipsercp.hyperbola** and **org.jivesoftware.smack**



Exercise

- Make the roster view non-closable, without title
- Create a login prompt (dialog is provided)
- Create a “Delete Contact” action
 - Add it to the tool bar and to the menu bar
 - The action should only be active when a contact is selected
- Modify the roster view to show details about the selected contact in the status line
- Track in the roster view the active chat editor

Section III – Extra RCP plug-ins, advanced topics and other cool stuff

1. Helping your user
2. Provisioning and deploying
3. Advanced / extreme branding
4. Using third party libraries
5. Modularity, integration and extensibility

1. Helping your users

Eclipse Help infrastructure – 101

- HTML and XML based system
- Dynamic content generation
- Contextual help
- Search engine

Help UI
User interface and dialogs

Help Core
API to access the documents

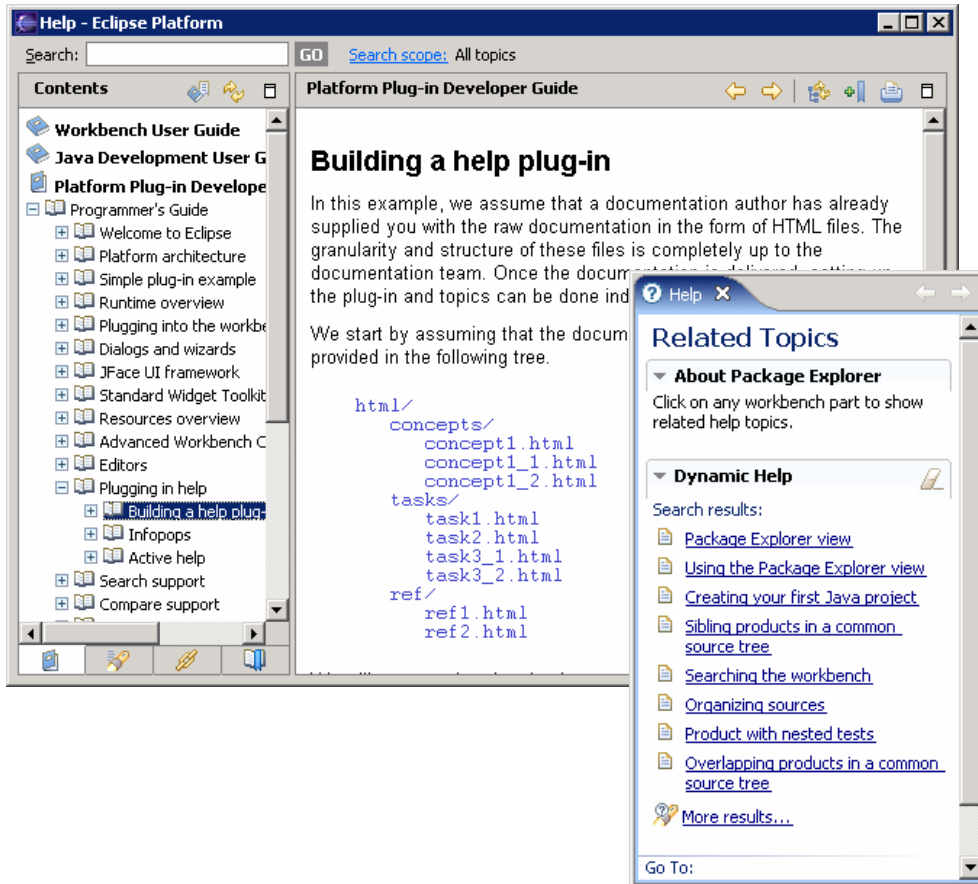
plugin.xml

```
<extension point="org.eclipse.help.toc">
  <toc primary="true"
        file="doc/guide.xml" />
  <toc file="doc/tipsAndTricks.xml" />
</extension>
```

guide.xml

```
<toc label="Hyperbola User Guide">
  <topic label="Getting Started">
    <anchor id="gettingstarted"/>
  </topic>
  <topic label="Commands"
        href="doc/cmds.html" />
</toc>
```

Eclipse Help infrastructure – UI



Required plug-ins

org.apache.ant
 org.apache.lucene
 org.eclipse.help.appserver
 org.eclipse.help.base
 org.eclipse.help.ui
 org.eclipse.help.webapp
 org.eclipse.tomcat

Eclipse Help infrastructure – Core API

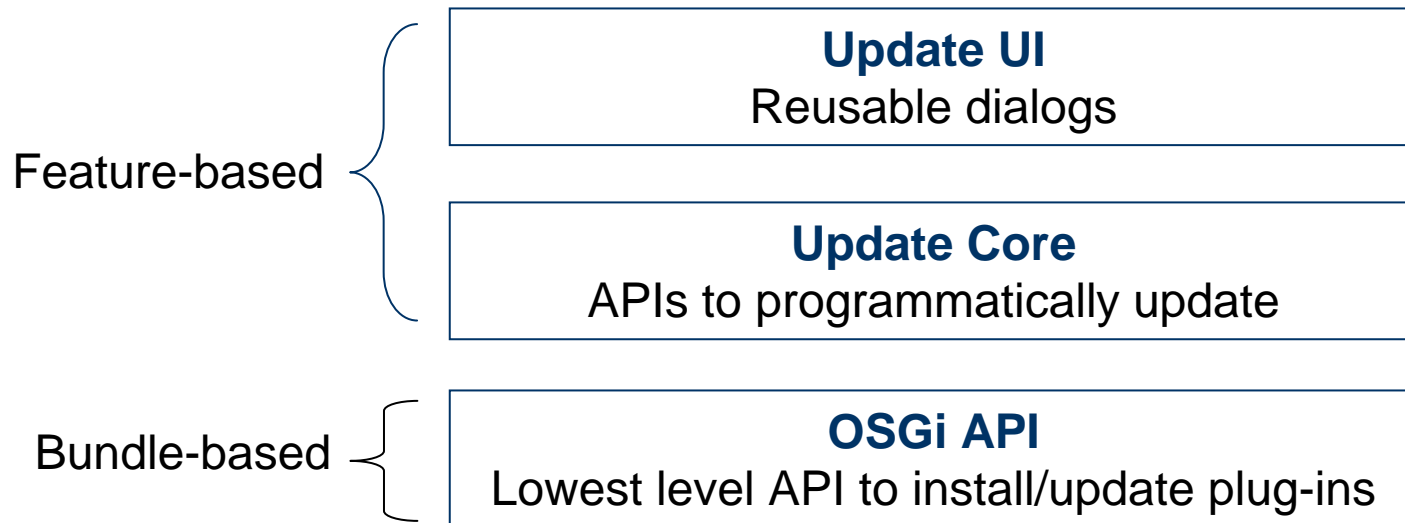
- Lightweight API
- Define a hierarchical table of contents, giving access to the help topics and their content
- Define content for context-sensitive help
- Plug-in: `org.eclipse.help`
 - See the class `org.eclipse.help.HelpSystem`

Reference

- Help: Plugging in Help

2. Provisioning and deploying

Updating your application



Update site and features – 101

⇒ An update site is an URL where you can find a site.xml

- No special server is required!
- Update site delivers “features”
- A Feature references plug-ins it manages

feature.xml

```
<feature id="hyperbol afeature"
        label="hyperbol a-feature"
        version="1.0.0">
[... ]
  <plugin id="org.ecl ipsercp.hyperbol a"
          download-size="0"
          install-size="0"
          version="1.0.0" />
[... ]
</feature>
```

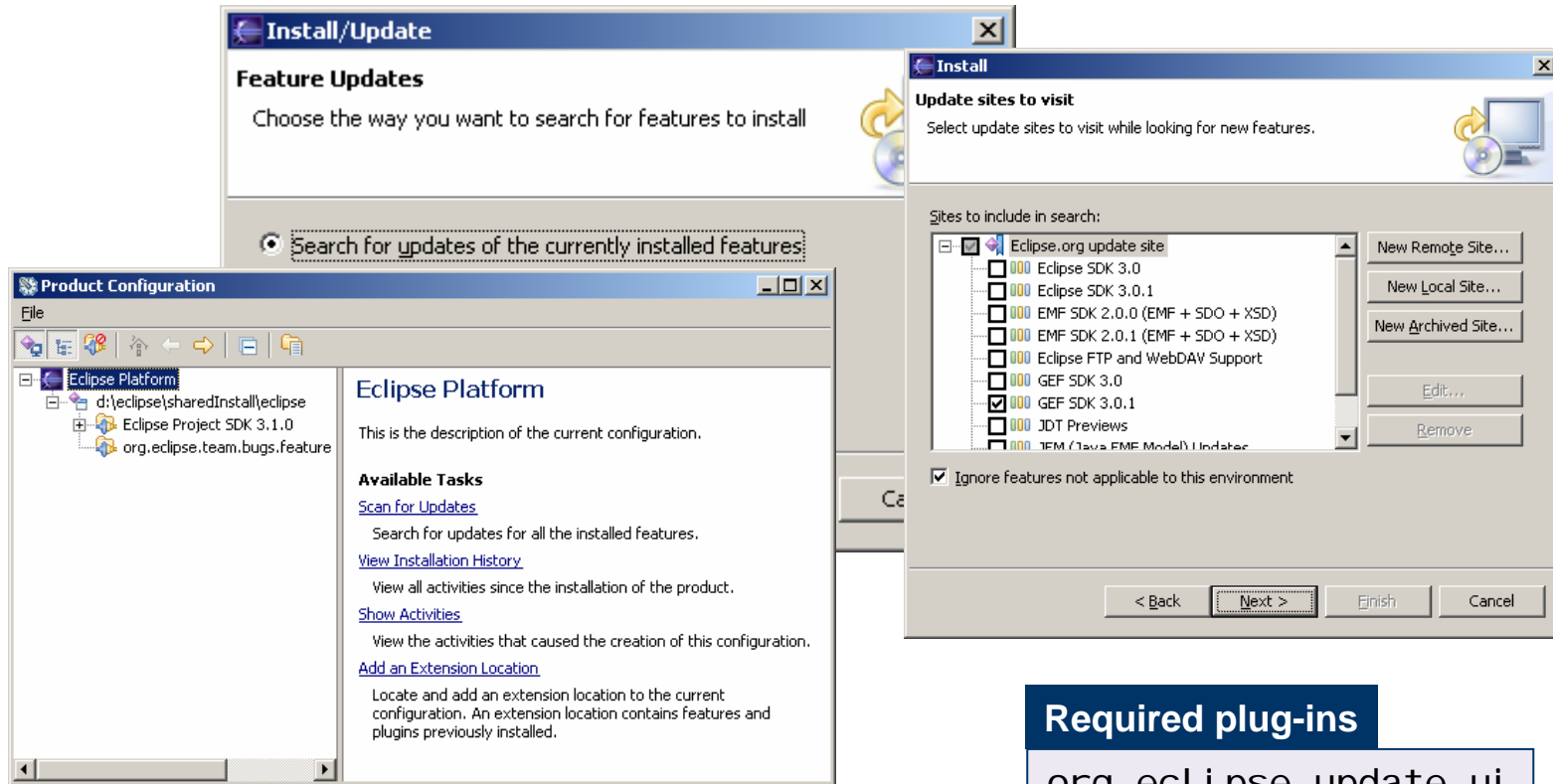
typical layout of an update site

```
si te.xml
features/
    hyperbol a-feature.jar
pl ugi ns/
    org.ecl ipsercp.hyperbol a.jar
    org.ecl ipsercp.hyperbol a.ui.jar
```

Reference

- EclipseCon'05: Creating, Packaging, [...] Features in Eclipse 3.0 (S. Minocha, P. McCarthy)

Updating your application – Update UI



Required plug-ins
 org.eclipse.update.ui
 and its prerequisites

Reference

- Eclipse.org article: How to Keep Up To Date (D. Glozic and D. Birsan)

Updating your application – the update core API

- **Plugin:** org.eclipse.update.core
Package: org.eclipse.update.operation

Installing a feature

```
InstallCommand cmd = new InstallCommand("myFeature", "1.0.0",  
                                         http://myCompany/product, null, "false");  
cmd.run(new NullProgressMonitor());  
cmd.applyChangesNow();
```

Reference

- Hyperbola code: org.eclipse.rcp.hyperbola.discovery.update.Directory.installProvider()

Updating your application – the OSGi API

org.osgi.framework.Bundle / BundleContext API

```
//to install
Bundle[] toRefresh = new Bundle[1];
toRefresh[0] = context.installBundle("reference:file:d:/aBundle");
refreshPackages(toRefresh);

//to uninstall
toRefresh[0].uninstall();
```

To predict the state of the system (org.eclipse.osgi.service.resolver)
See PlatformAdmin service, more precisely the State class

Note

- When using this API, it is recommended to download the files manually and use a reference: URL, otherwise OSGi copies the jars into the configuration area.

Reference

- Eclipse code: org.eclipse.update.configurator.ConfigurationActivator

Java WebStart

- Eclipse 3.1 can be Java WebStarted
 - It requires full permission on the client
- Preparing for Java WebStart
 - Jar up your plug-ins
 - Sign your plug-ins
- The main is `org.eclipse.core.launcher.WebStartMain`
- Values from the `config.ini` goes in the JNLP file except:
 - `osgi.splashPath` use the JNLP mechanism
 - `osgi.bundles` not necessary. Implicit by the jars listed in the JNLP file

Reference

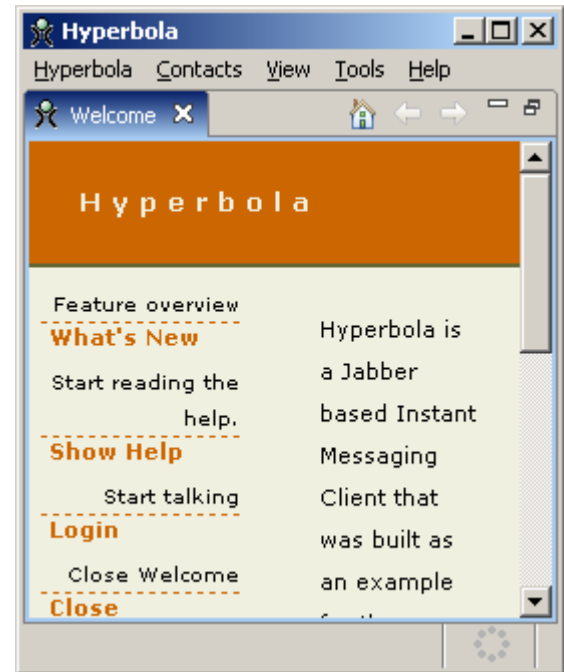
- Bug number: 80149
- EclipseCon'05: Packaging, Deploying and Running Rich Client (E. Burnette)

3. Advanced / extreme branding

Intro support

⇒ It provides the “welcome page” for a product.

- HTML / CSS or SWT / Forms based
- It supports two modes:
 - Full mode
 - Standby mode
- Can run actions to drive the UI



Reference

- Eclipse.org article (to be published): Introducing Intro: How to create Welcome pages [...] (M. Faraj)

Themes

- ⇒ General mechanism provided by the workbench to help plug-ins provide uniform look and feel to their components
- A theme has an ID
 - Declare categorized colors, fonts, and other data items (identified with IDs)
 - Assign specific values to these within a named theme
 - To access the values use JFace color and font registries and track the change using change listeners

 - Workbench uses this for its own colors
 - Other parts (views and editors) can do so as well

To set a theme

```
IWorkbench. getThemeManager(). setCurrentTheme\(\)
```

bug 84738

The presentation API

- ⇒ The presentation controls the appearance of editors, views and other components in the workbench.
- Relation themes / presentations
 - A presentation should define its colors and fonts in a theme category
 - A theme category can be bound to a presentation

Reference

- EclipseCon'05: Presentations API - the look and feel of Eclipse (J.M. Lemieux, S. Xenos)
- Example: http://www.eclipse.org/rcp/main.html#text_editor_example
- Example: R2.1 presentation (plugin org.eclipse.ui.presentations.r21)

Non rectangular window

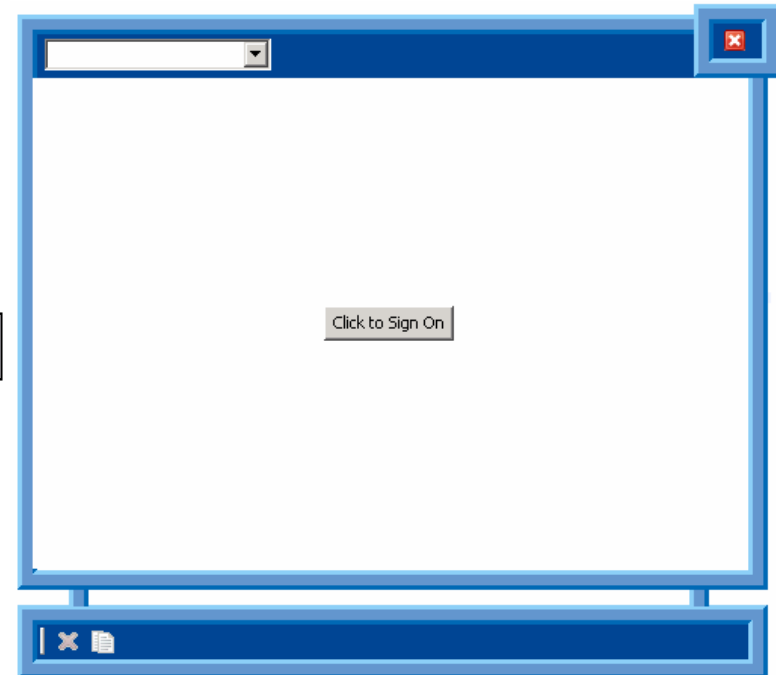
Handled by the workbench window advisor

Shell creation and customization

```
WorkbenchWindowAdvisor.postWindowCreate()
```

Filling the window with toolbar, menu bar, etc...

```
WorkbenchWindowAdvisor.createWindowContents()
```



4. Using third party libraries

Packaging third party libraries

- Recommended way:
 - **Create one or many bundles !!!**
Examples:
 - junit.jar → Junit plugin.
 - Hibernate 2.1.7 → log4j, junit, xalan, xerces, jta, ... plugins
- Less recommended ways
 - Embed the library in your own plug-in
 - Add the library on the boot classpath
 - Set the parent loader of the framework – and place the library accordingly
`osgi . parentClassLoader = {boot | ext | app | fwk }`

Reference

- (Potential) PDE support in 3.1 for the recommended way
- <http://www.eclipsefaq.org/chris/faq/html/faq105.html>

Frequent problems with third party libraries

- Context classloaders

- In the library:

```
Thread.currentThread().getContextClassLoader().getResource()
```

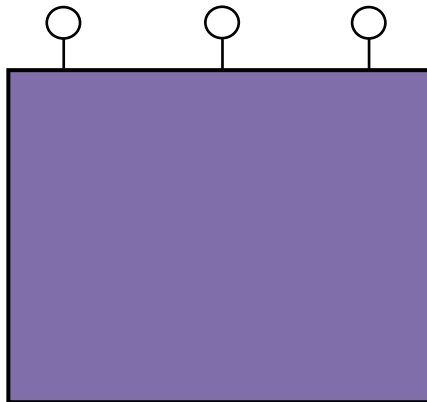
- The solution / workaround

```
Thread current = Thread.currentThread();
ClassLoader oldLoader = current.getContextClassLoader();
try {
    current.setContextClassLoader(getClass().getClassLoader());
    //call library code here
} finally {
    current.setContextClassLoader(oldLoader);
}
```

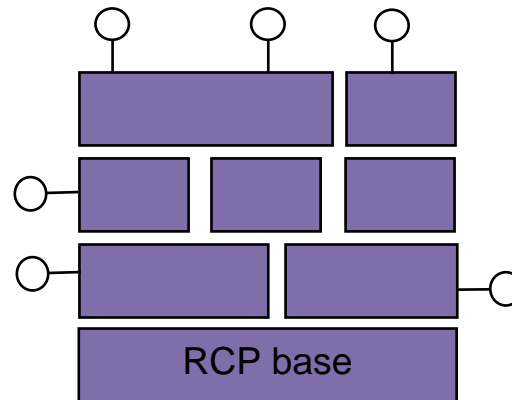
- Property files

5. Modularity, integration and extensibility

The platform mindset



Your app without RCP



Your app with RCP

- Inherent extensibility of the RCP base
- Componentize your application
- Identify APIs
- Provide extensibility (extension / extension points)

Think modularity! Think Eclipse!

RCP Application and/or plug-ins to the IDE (or other RCP apps)?

- The IDE is an RCP application whose target audience is developers.
 - The main difference is that you do not have privileged access to the workbench through advisors
- An IDE and an RCP app are both made of plug-ins ☺
- The main recommendations are:
 - Do not put “business” logic into the advisor
 - Do not depend too much on the look and feel
 - Isolate the concerns of your application
 - You may have to write an integration plug-in to hook in some things in the IDE and adapt the workflows

Reference

- EclipseCon'05 : Eclipse RCP Everywhere (J.-M. Lemieux, J. McAffer)

Turning your RCP application into a domain-specific platform

- Define API to access your model
- Expose extension-points
- Other UI API?
 - Menu bar and tool bar paths
 - View ids
 - Action ids
- Build a community

References

- Eclipse.org article: How to Use the Eclipse API? (J. des Rivieres)
- EclipseCon'05 session: API First (J. des Rivieres)
- EclipseCon'04 session: Eclipse APIs: Lines in the Sand (J. des Rivieres)

The end...

F.A.Q.

Productization issue – Accessibility and NL

- Translation:
 - Plugin.xml – plugin.properties
 - Code –regular java techniques
 - NL fragments – to provide after the fact translation
 - org.eclipse.osgi.util.NLS mechanism

- Accessibility:
 - The workbench is already accessible

References

- Eclipse help: Locale specific files
- Eclipse.org article: Designing Accessible Plug-ins in Eclipse (T. Creasey)

Eclipse and Java serialization

- Because of Eclipse multiple classloaders, you may have trouble to restore objects from a plug-in that is not in your prerequisites
- To work around that, annotate the output stream:

```
class EclipseOutputStream extends ObjectOutputStream {
    PackageAdmin pkgAdmin; //initialize it in the constructor

    protected void annotateClass(Class clazz) throws IOException {
        super.annotateClass(clazz);
        Bundle declaringBundle = pkgAdmin.getBundle(clazz);
        String bundleName = null;
        if (declaringBundle != null) {
            bundle = declaringBundle.getSymbolicName();
        } else {
            bundle = "bootClasspath";
        }
        this.writeUTF(bundleName);
    }
}
```

Write in the stream the bundle that loaded the class. It will be read by `resolveClass()` of `EclipseInputStream`.

What does it take to make my plug-ins dynamic?

- Good practices
 - Keep your objects for yourself and do not offer API for other plug-ins to store their objects in your plug-in
 - Track extension and extension-points
 - Clean-up after yourself (unregister listeners, free OS resources)

- Runtime facilities:
 - `org.eclipse.core.runtime.dynamic.helpers`
 - `org.eclipse.core.runtime.IRegistryChangeListener`
 - `org.eclipse.core.runtime.InvalidRegistryObjectException`
 - Dynamic capabilities markup in the `manifest.mf`

- OSGi facilities
 - Services, declarative services
 - `org.osgi.framework.BundleEvent` / `org.osgi.framework.FrameworkEvent`

References

- Platform Core webpage: Markup to express bundle dynamic capabilities.

How do I do a system tray icon / item?

<http://dev.eclipse.org/viewcvs/index.cgi/%7Echeckout%7E/platform-swt-home/dev.html>

Can I reuse the resources plug-in?

- Yes – however it only provides you with the concepts of workspace, projects, markers, and it is bound to the file system.
- The resources navigator is **not** in the resources plug-in.
- Some other interesting views are in the IDE.

How do I access a file in my plug-in?

- In the given bundle

```
Bundl e. getEntry(" i mages/run. gi f" )  
Bundl e. getEntri es(" i mages" )
```

- In the given bundle and its fragments, supports translation

```
Pl atform. fi nd(bundl e, new Path(" i mages/hel l o. txt" ))  
Pl atform. fi nd(bundl e, new Path("$nl $/fi l e. txt" ))
```

What is the licensing model?

References

- EclipseCon'05: Getting your Plug-in Legal: a Primer for Eclipse Developers (I. Heffan)
- <news://news.eclipse.org/eclipse.foundation>

How do I filter contributions?

- Activities – An **activity** is a logical grouping of function that is centered around a certain kind of task. The contributions made available to the users are filtered using the enabled activities.
- It can also be adapted to hide unwanted contributions from other plug-ins.

References

- EclipseCon'05: Addressing UI Scalability in Eclipse (K. Horne)
- EclipseCon'05: Techniques for Seamless Integration: A Talk from the Trenches (J. Jones)

What is the shape of a plug-in – the alternatives?

1. Plug-in folder containing the code jar

```
pl ugi ns/
  org. ecl i pse. ui _3. 1. 0/
    ui . jar
    pl ugi n. xml
```

2. One jar containing the code jar

```
pl ugi ns/
  org. ecl i pse. ui _3. 1. 0. jar
    (the jar contains)
    ui . jar
    pl ugi n. xml
```

3. Plug-in folder containing the class files “dot” is the classpath

```
pl ugi ns/
  org. ecl i pse. ui _3. 1. 0/
    org/ecl i pse/ui /UI . cl ass
    org/ecl i pse/ui /Workbench. cl ass
    pl ugi n. xml
```

4. One jar containing the classes “dot is the classpath”

```
pl ugi ns/
  org. ecl i pse. ui _3. 1. 0. jar
    (the jar contains)
    org/ecl i pse/ui /UI . cl ass
    org/ecl i pse/ui /Workbench. cl ass
    pl ugi n. xml
```

Note

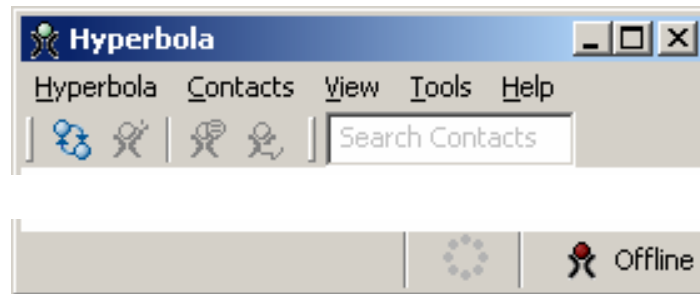
- 1 and 4 are recommended. 2 can't be tooled since java compilers do not support nested jars and 3 will result in too many files.

How do I reuse workbench preference pages?

See bug 73587

Widgets in action bars

- To populate your cool bars and status bars with text fields, drop down, etc.
 1. Create a contribution item (by extending `org.eclipse.jface.action.ContributionItem` or `ControlContributionItem`)
 2. Add this contribution item to the manager (for example `MenuManager`)



Reference

- Hyperbola code: `SearchContributionItem` or `StatusActionGroup`

MANIFEST.MF or plugin.xml

If you are targeting 2.1 the plugin.xml must be used (No RCP in 2.1)

If you are targeting 3.0: the plugin.xml is recommended since there is no good support to edit 3.0 manifest

If you are targeting with one plug-in 3.0 and 3.1, the plugin.xml is recommended

If you are targeting 3.1, the manifest.mf is recommended since there will be appropriate tooling for it

How do I acquire an OSGi service?

- **ServiceTracker**

```
ServiceTracker tracker;  
tracker = new ServiceTracker(bundleContext, "service.interface", null);  
tracker.open()  
tracker.getService()
```

- **BundleContext.getServiceReference**

```
ServiceReference packageAdminRef;  
packageAdminRef = ctx.getServiceReference(PackageAdmin.class.getName());  
if (packageAdminRef != null)  
    ctx.getService(packageAdminRef);
```

- **Declarative services API**

See bug 43890

Authentication – when the application starts

```

public class Application implements IPlatformRunnable {
    public Object run(Object args) throws Exception {
        try {
            //Do authentication here ← login points
            Display d = PlatformUI.createDisplay();
            //Or here if you need to have a display ← login points

            WorkbenchAdvisor wa = new MinimalAdvisor();
            PlatformUI.createAndRunWorkbench(d, wa); ← run the workbench
            return new Integer(0);
        } finally {
            if (d != null)
                d.dispose();
        }
    }
}

```

Authentication – before the application starts

- The authentication bundle
 - Do the full authentication in the start method
 - This is usually bad practice, but here we need to be synchronous
 - On authentication failure, change the application to run to be an empty one.
 - On authentication success, set the start level to 6

- Other settings
 - Add the authentication bundle on the `osgi . bundl es` list and set its start level


```
osgi . bundl es = org. ecl i pse. core. runti me@1: start,
                authenti cati on@2: start
```
 - Set the start level of the framework to the value to which the authentication bundle has been set

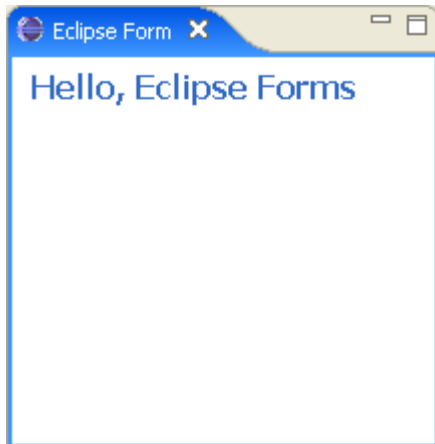

```
osgi . startl evel = 2
```
 - Ensure that the default start level of other bundles is greater

Note

- This technique can also be used to do user based provisioning and/or to change the application to run

Using flat look forms support in my editors and views?

- Use the plug-in: org. eclipse. ui . forms



```
public void createPartControl (Composite parent) {
    toolkit = new FormToolkit(parent.getDisplay());
    form = toolkit.createScrolledForm(parent);
    form.setText("Hello, Eclipse Forms");
}
```

References

- Eclipse.org article (to be published): Eclipse Forms: Rich UI for the Rich Client (D. Glozic)
- Guide: <http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/pde-ui-home/working/EclipseForms/EclipseForms.html>