

Introduction to Eclipse Rich Client

Anshu Jain

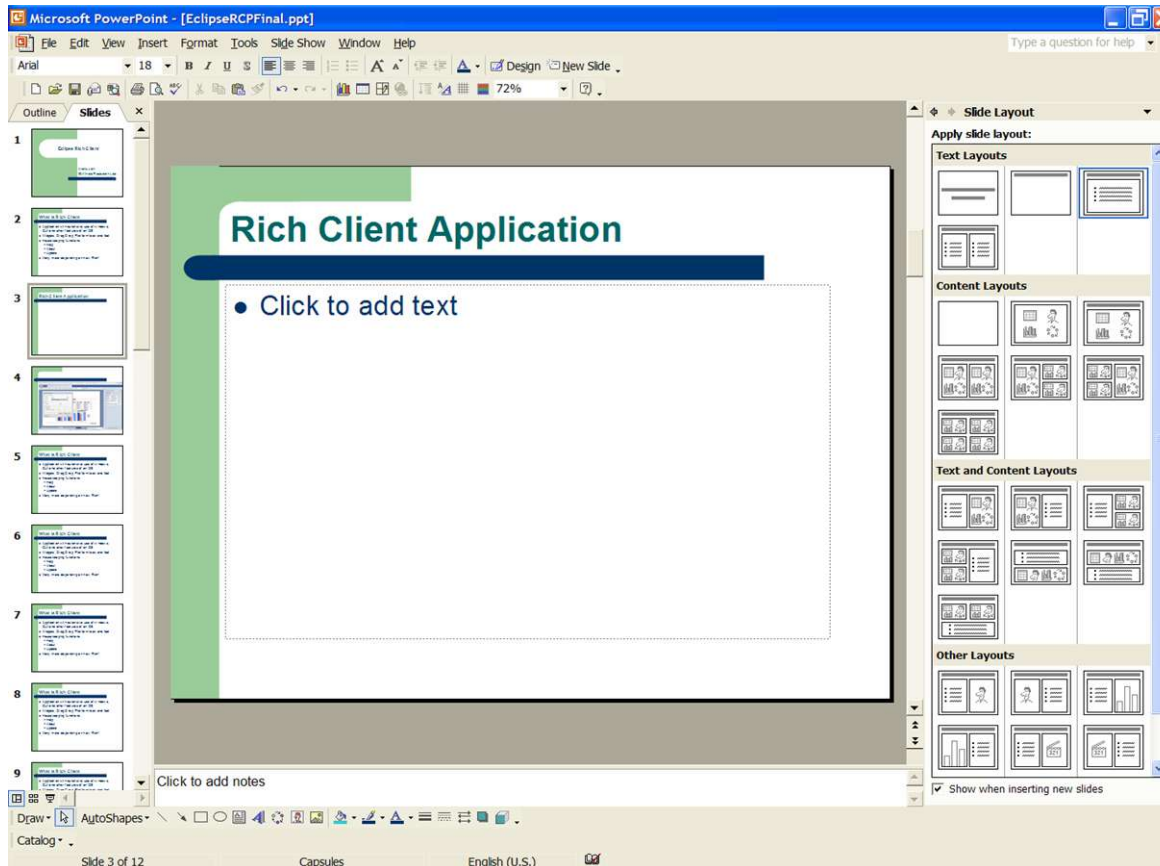
IBM India Research Lab



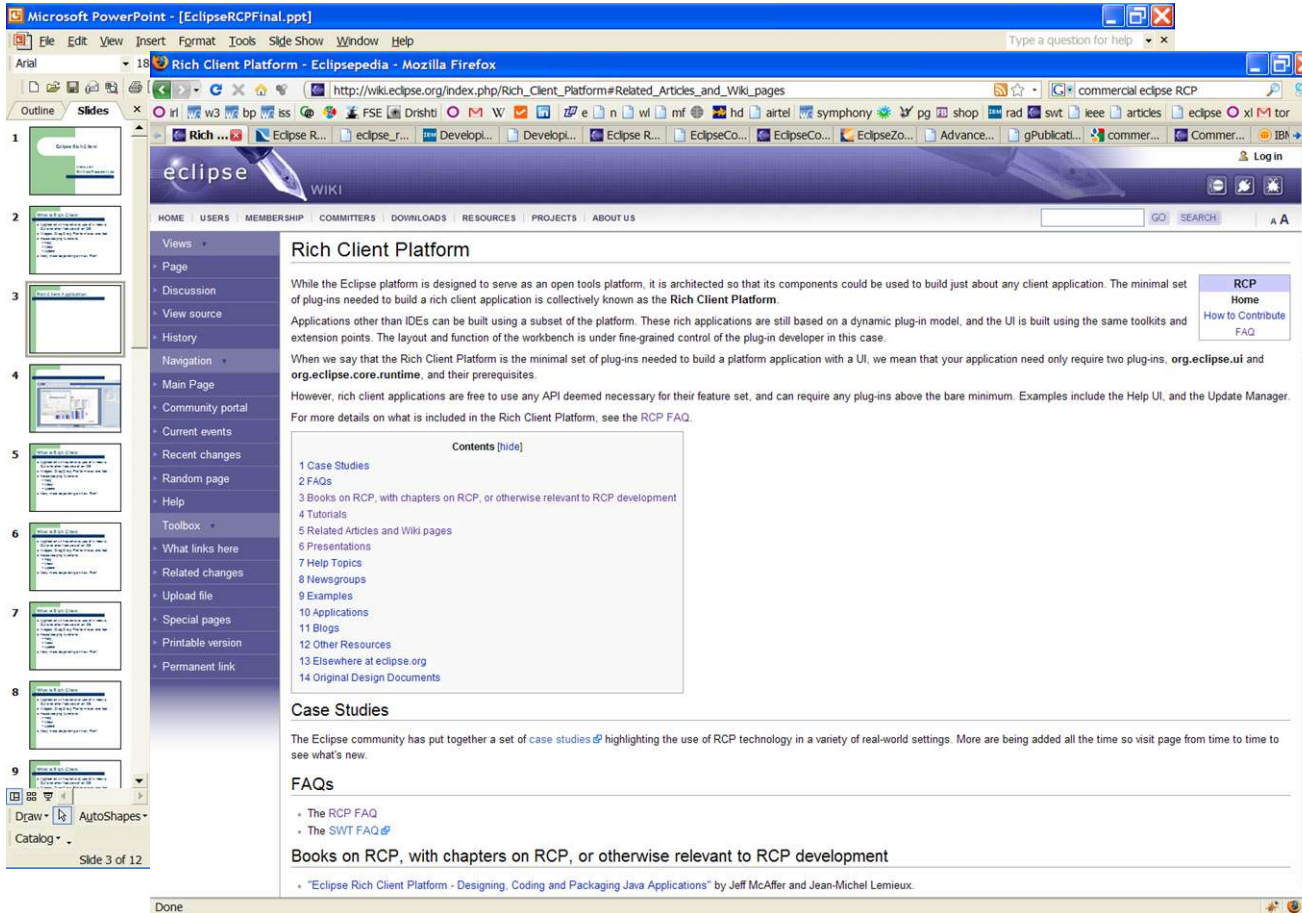
Agenda

- Introductions
- Motivation for Rich Client Platforms
- Architecture of Eclipse Rich Client Platform
- Dive into core components
- RCP specific issues
- This is not a step by step tutorial
 - <http://www.google.com/search?q=Eclipse+RCP+Tutorial>
- Lookout for footnotes on slides – references and links

Rich Client Applications



Rich Client Applications



Microsoft PowerPoint - [EclipseRCPFinal.ppt]

Rich Client Platform - Eclipsepedia - Mozilla Firefox

http://wiki.eclipse.org/index.php/Rich_Client_Platform#Related_Articles_and_Wiki_pages

Rich Client Platform

While the Eclipse platform is designed to serve as an open tools platform, it is architected so that its components could be used to build just about any client application. The minimal set of plug-ins needed to build a rich client application is collectively known as the **Rich Client Platform**.

Applications other than IDEs can be built using a subset of the platform. These rich applications are still based on a dynamic plug-in model, and the UI is built using the same toolkits and extension points. The layout and function of the workbench is under fine-grained control of the plug-in developer in this case.

When we say that the Rich Client Platform is the minimal set of plug-ins needed to build a platform application with a UI, we mean that your application need only require two plug-ins, **org.eclipse.ui** and **org.eclipse.core.runtime**, and their prerequisites.

However, rich client applications are free to use any API deemed necessary for their feature set, and can require any plug-ins above the bare minimum. Examples include the Help UI, and the Update Manager.

For more details on what is included in the Rich Client Platform, see the RCP FAQ.

Contents [hide]

- 1 Case Studies
- 2 FAQs
- 3 Books on RCP, with chapters on RCP, or otherwise relevant to RCP development
- 4 Tutorials
- 5 Related Articles and Wiki pages
- 6 Presentations
- 7 Help Topics
- 8 Newsgroups
- 9 Examples
- 10 Applications
- 11 Blogs
- 12 Other Resources
- 13 Elsewhere at eclipse.org
- 14 Original Design Documents

Case Studies

The Eclipse community has put together a set of [case studies](#) highlighting the use of RCP technology in a variety of real-world settings. More are being added all the time so visit page from time to time to see what's new.

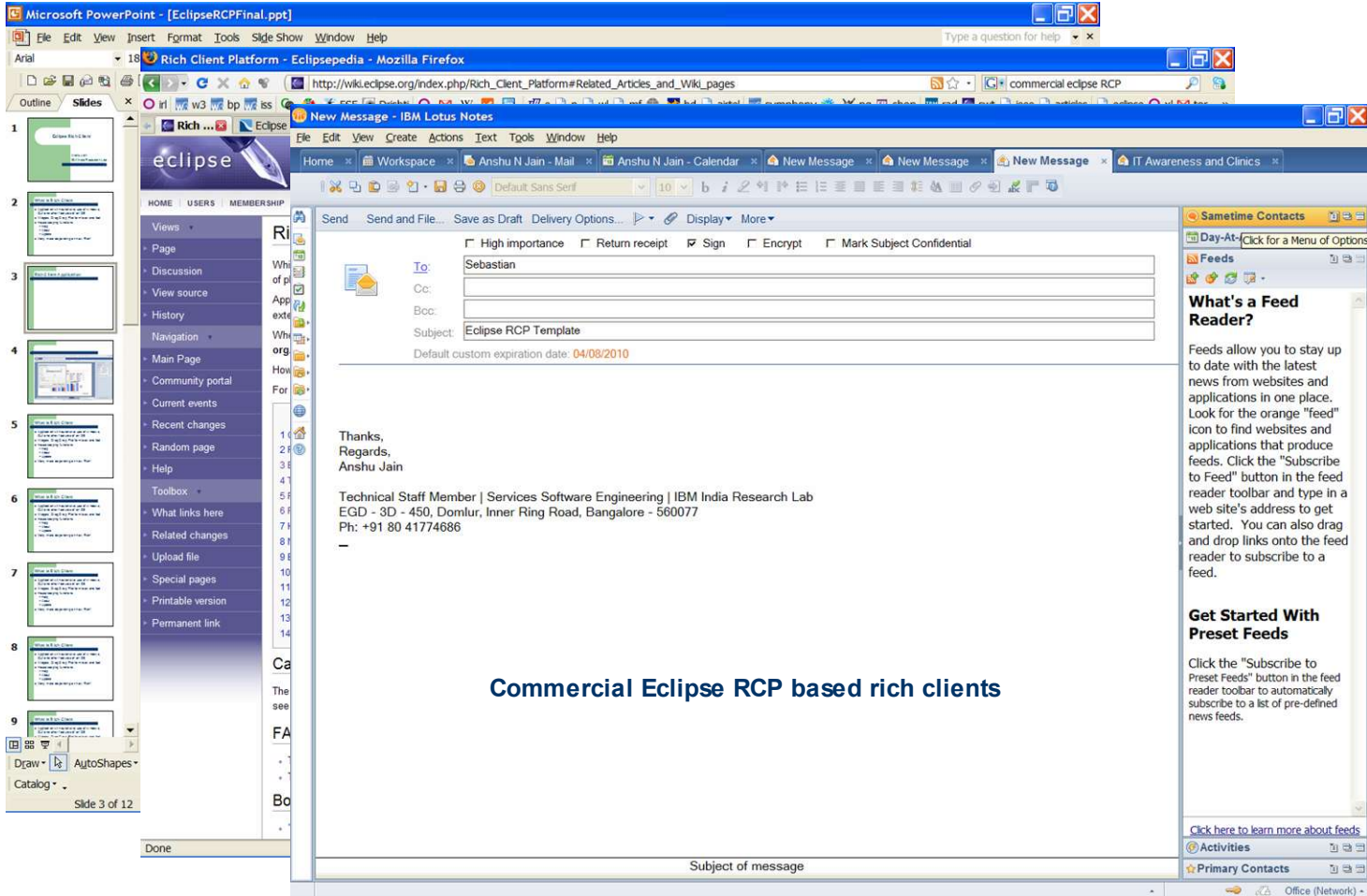
FAQs

- The RCP FAQ
- The SWT FAQ

Books on RCP, with chapters on RCP, or otherwise relevant to RCP development

- "Eclipse Rich Client Platform - Designing, Coding and Packaging Java Applications" by Jeff McAffer and Jean-Michel Lemieux.

Rich Client Applications



Commercial Eclipse RCP based rich clients

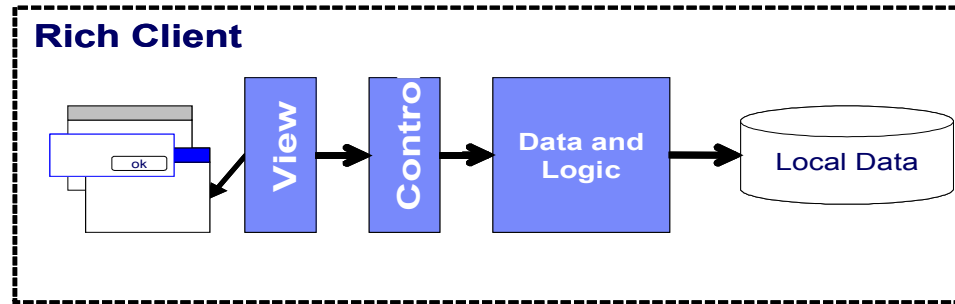
What is Rich Client

- Application with extensive use of windows, GUI and other features of an OS
- Widgets, Drag Drop, Platform look and feel
- Housekeeping functions
 - Help, About, Update
- Many more depending on how 'Rich'

What is a rich client - Wikipedia

- **A Rich Client Platform (RCP)** is software consisting of the following components:
 - A core (microkernel), lifecycle manager
 - A standard bundling framework
 - A portable widget toolkit
 - File buffers, text handling, text editors
 - A workbench (views, editors, perspectives, wizards)
 - Data binding
 - Update manager

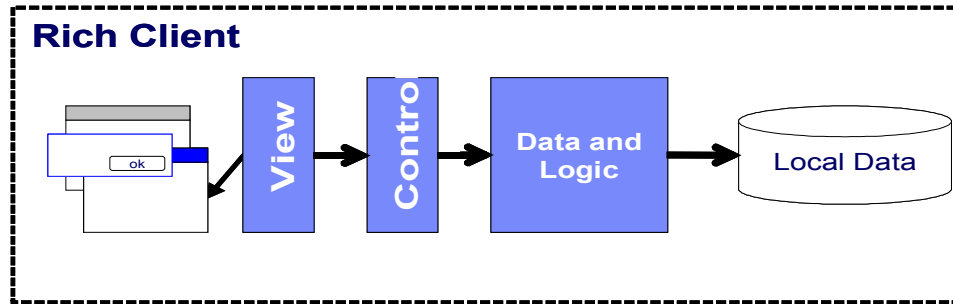
Possible Architectures



1 Tier

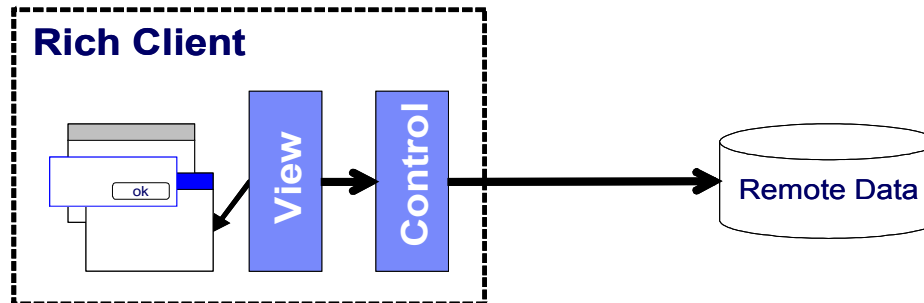
- office applications
- paint programs

Possible Architectures



1 Tier

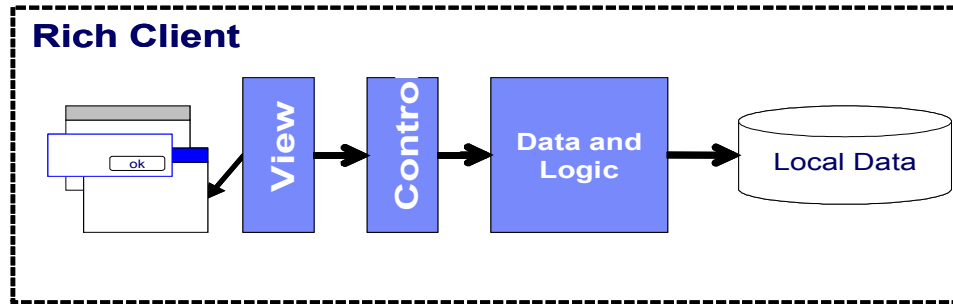
- office applications
- paint programs



2 Tier

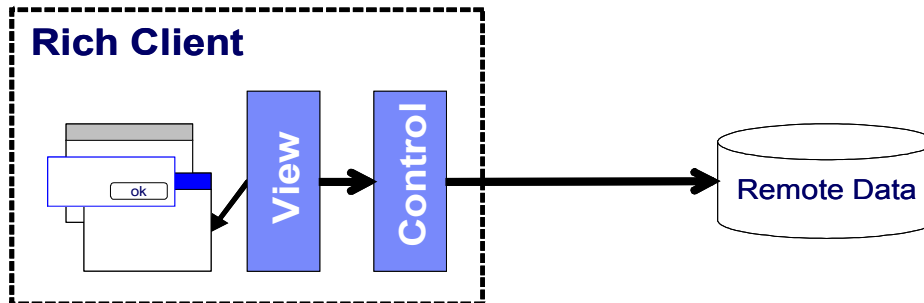
- mail clients
- p2p applications

Possible Architectures



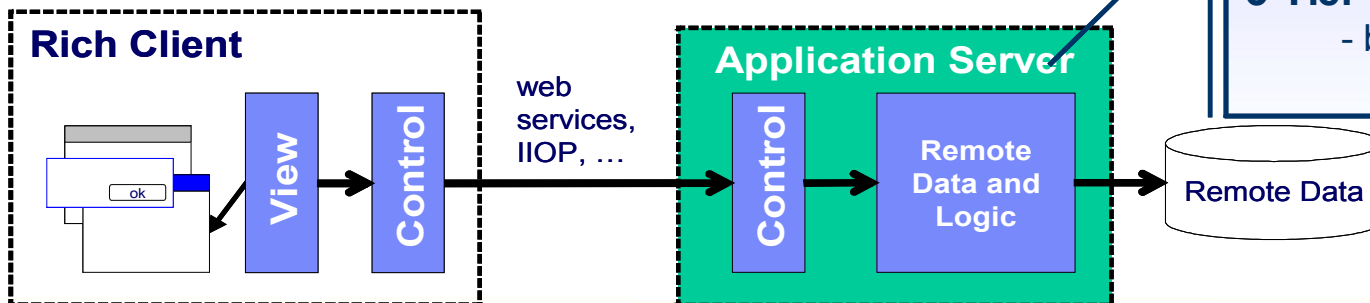
1 Tier

- office applications
- paint programs



2 Tier

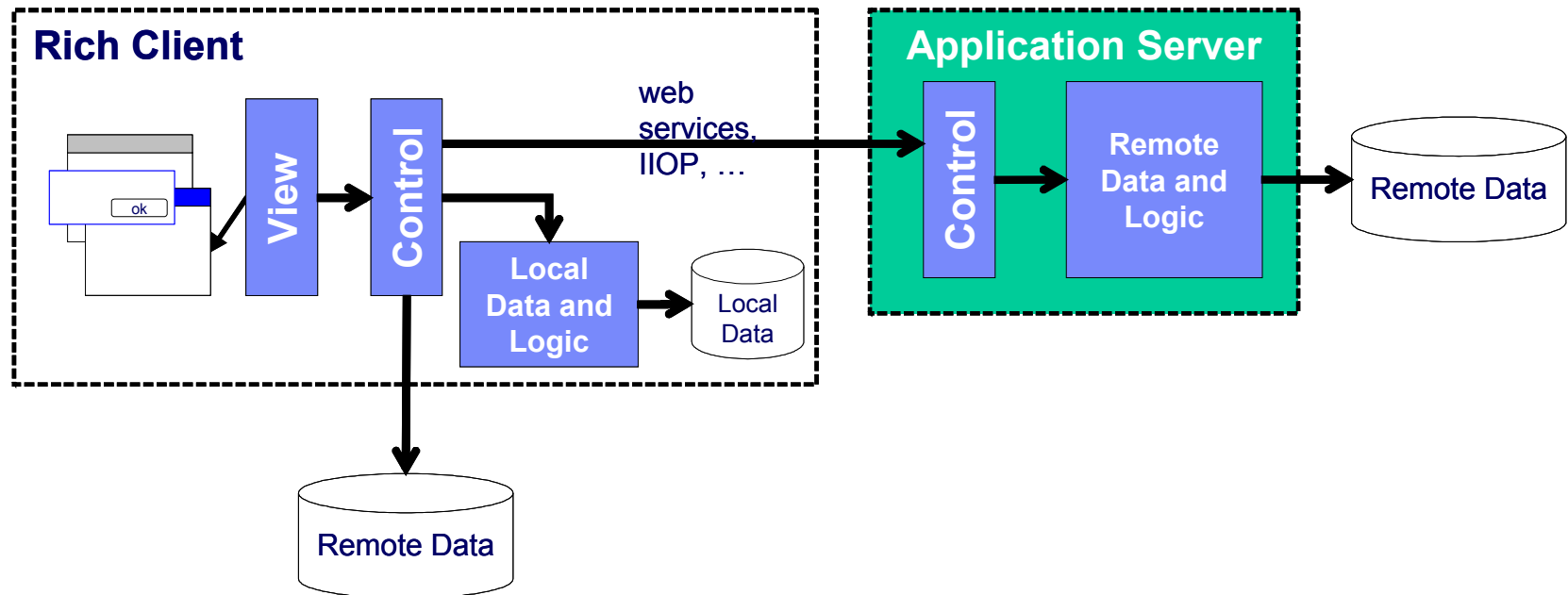
- mail clients
- p2p applications



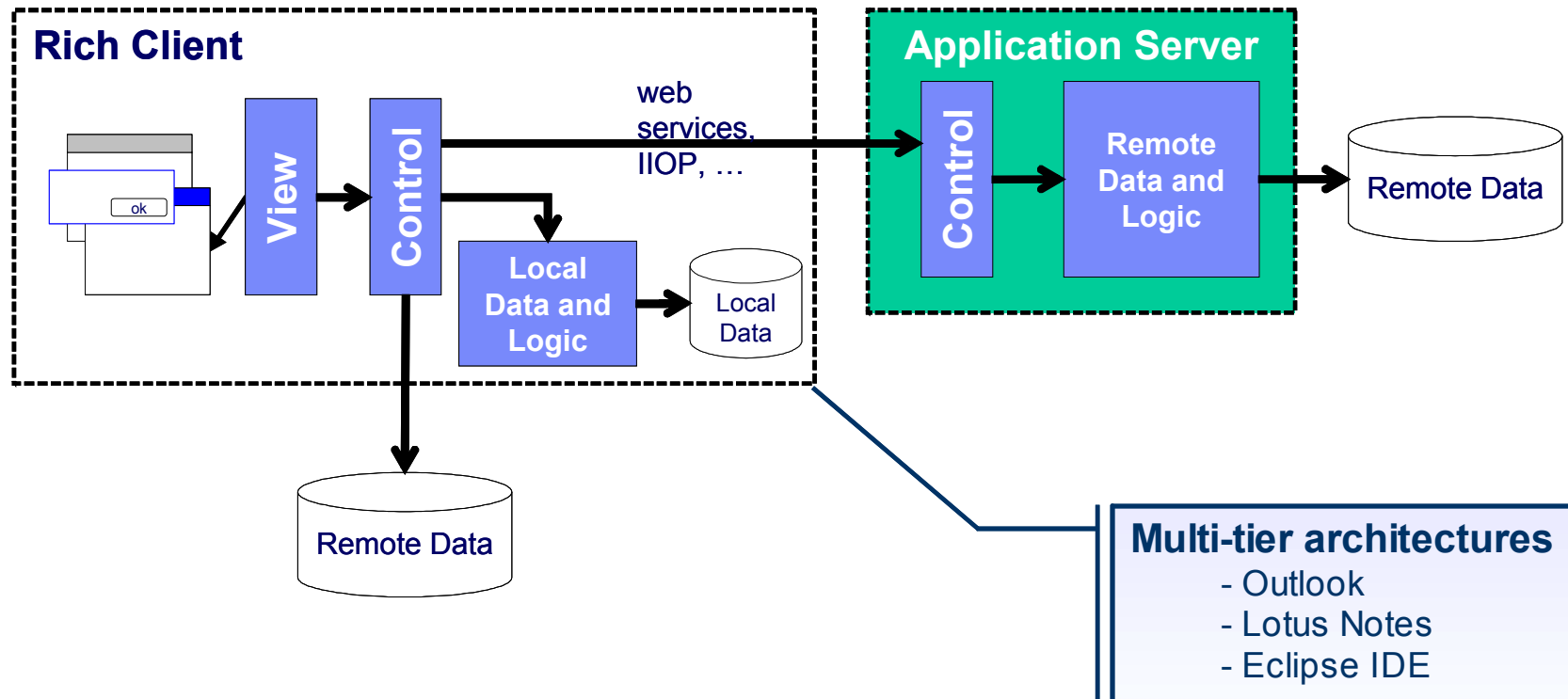
3 Tier

- browser

Combination of Architectures



Combination of Architectures

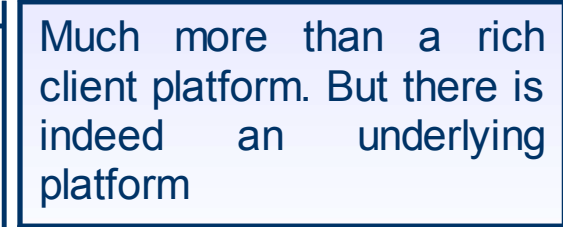


What is Eclipse IDE

- A rich client
- A tooling platform
- A framework
- Rich client platform ?

What is Eclipse IDE

- A rich client
- A tooling platform
- A framework
- Rich client platform ?

A rectangular callout box with a blue border and a light blue background. A line connects the top-left corner of the box to the text 'Rich client platform ?' in the list above.

Much more than a rich client platform. But there is indeed an underlying platform

Why Use *Eclipse* RCP?

- Common and consistent application services
 - Native look and feel
 - Window management
 - Standardized component model (Equinox)
 - Pervasive extensibility – Extension registry
 - Update Manager
 - Help system
- State of the art development tools
- Middleware for building rich client applications!
- Other Rcps
 - Netbeans, Spring Framework for Java
 - Other Rcps: QT Toolkit for c++ (and also some form for java)

Why use RCP – What are my options

- Use Plain old Java, Swing, VB, MFC etc.
- Assembly language...!!

Why use RCP – What are my options

- Use Plain old Java, Swing, VB, MFC etc.
- Assembly language..!!



Why use RCP – What are my options

- Use Plain old Java, Swing, VB, MFC etc.
- Assembly language..!!

MACPaint 😊
The Premier Rich Client



Why use RCP – What are my options

- Use Plain old Java, Swing, VB, MFC etc.
- Assembly language..!!



MACPaint ☺
The Premier Rich Client

5,804 lines - Pascal
2,738 lines - 68000

Why use RCP – What are my options

- Use Plain old Java, Swing, VB, MFC etc.
- Assembly language..!!

MACPaint ☺
The Premier Rich Client

5,804 lines - Pascal
2,738 lines - 68000

help, update, localization,
resizing of views. Plug-ins
– **none of above**



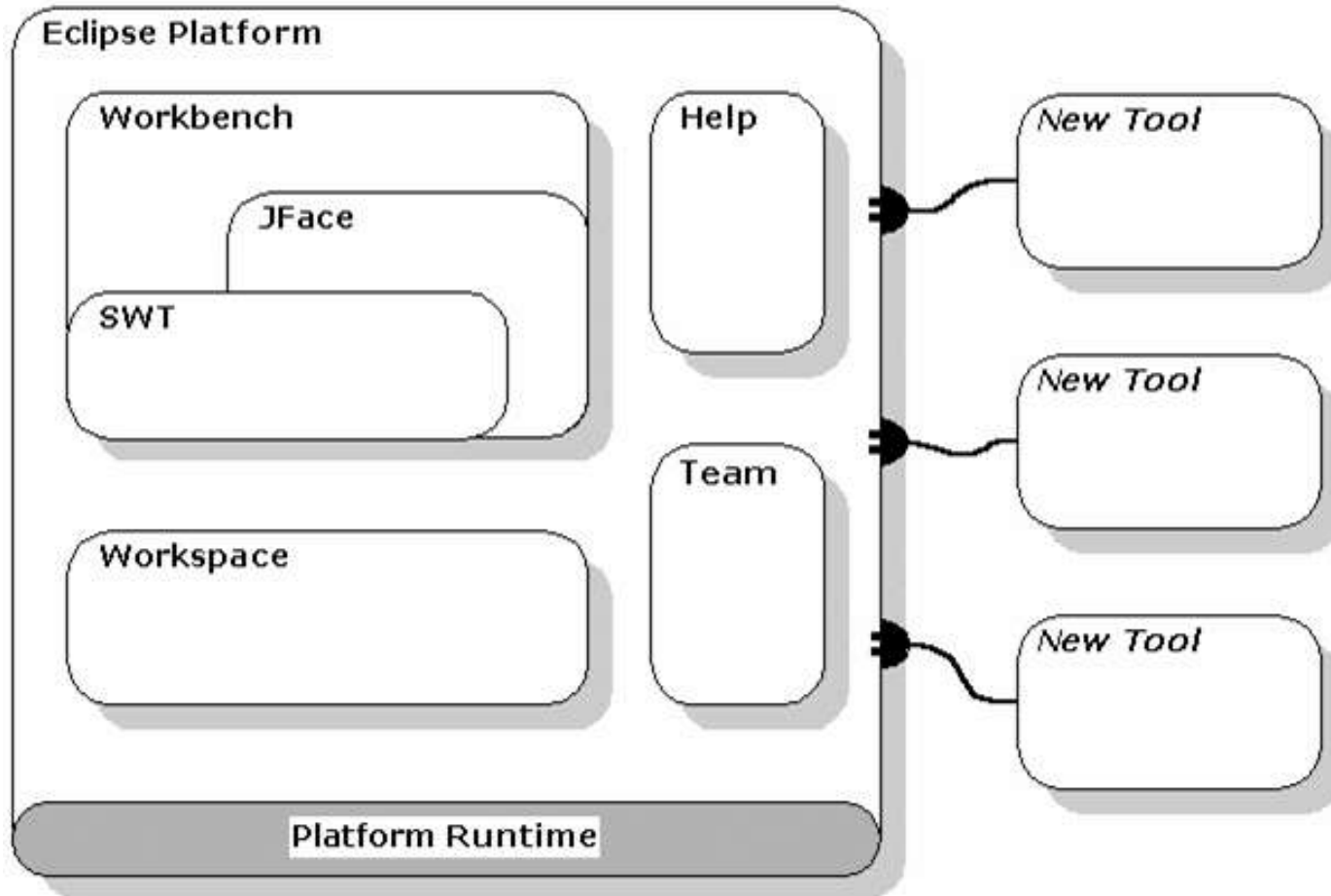
Why use RCP – What are my options

- RCP offers higher levels of abstractions
 - Registry
 - Perspective
 - Wizards
 - Dialogs
- Because you don't want to build the Rocket,
- You want to focus on the payload..!!

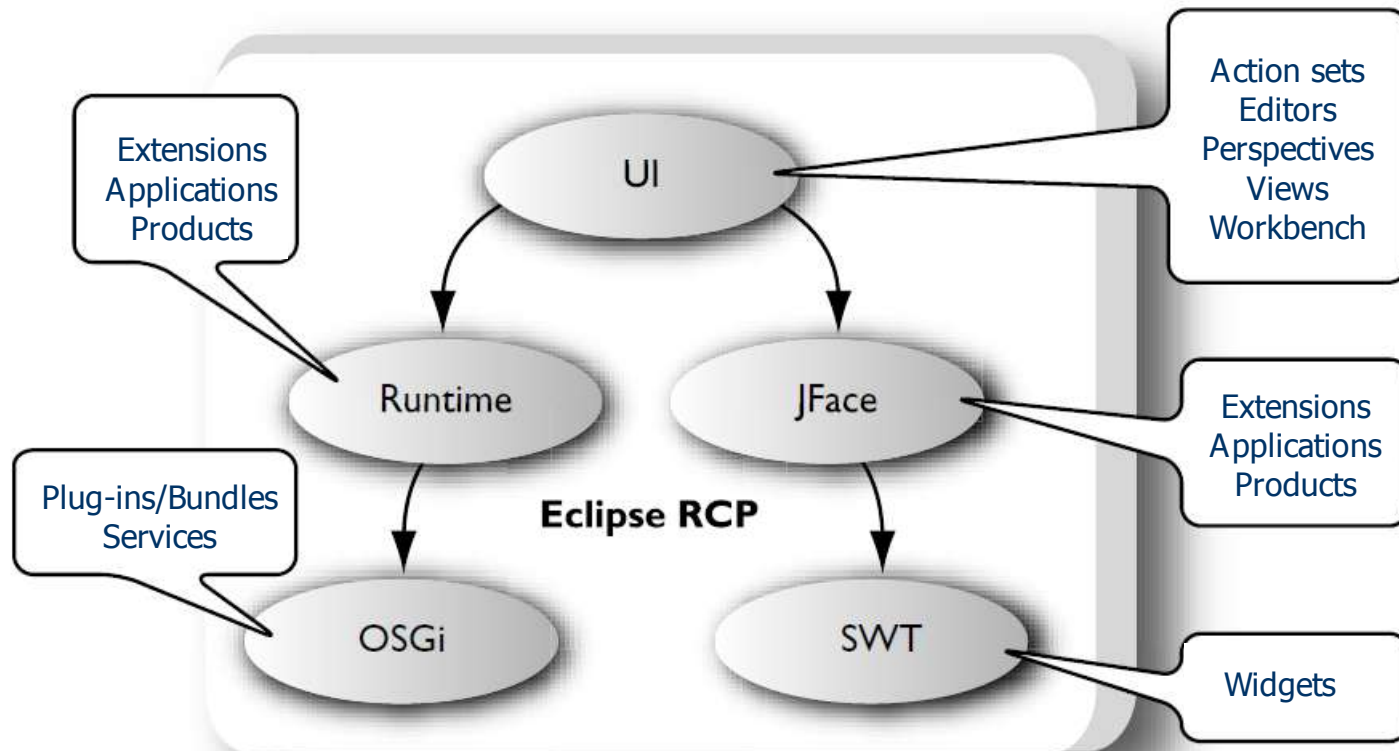
What is Eclipse RCP

- The minimal set of plug-ins needed to build a rich client application is collectively known as the **Rich Client Platform**
 - org.eclipse.ui
 - org.eclipse.core.runtime
(prerequisites of above)
- Eclipse platform whitepaper

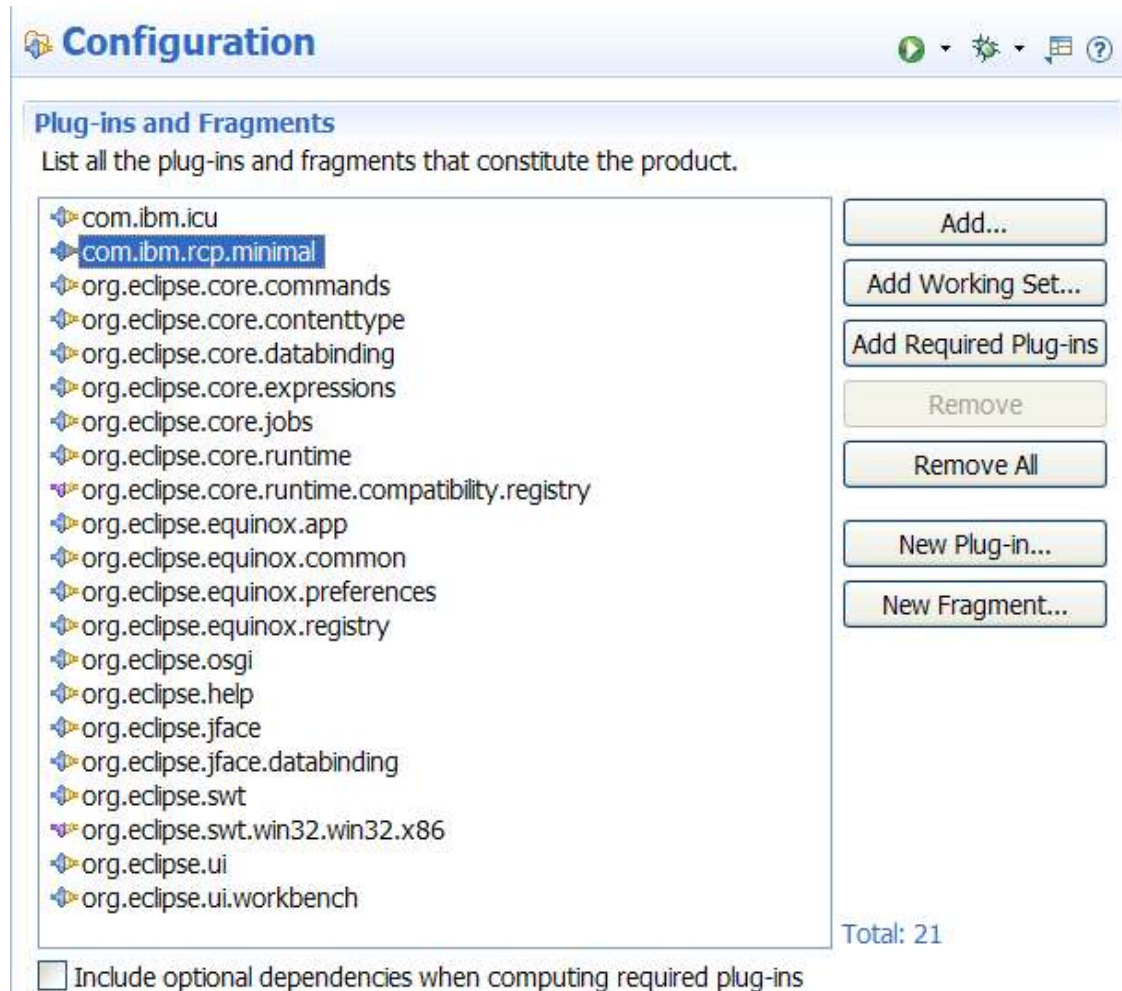
Eclipse Platform Architecture



RCP Architecture



Core Platform Plug-ins



Configuration

Plug-ins and Fragments
List all the plug-ins and fragments that constitute the product.

- com.ibm.icu
- com.ibm.rcp.minimal**
- org.eclipse.core.commands
- org.eclipse.core.contenttype
- org.eclipse.core.databinding
- org.eclipse.core.expressions
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility.registry
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.osgi
- org.eclipse.help
- org.eclipse.jface
- org.eclipse.jface.databinding
- org.eclipse.swt
- org.eclipse.swt.win32.win32.x86
- org.eclipse.ui
- org.eclipse.ui.workbench

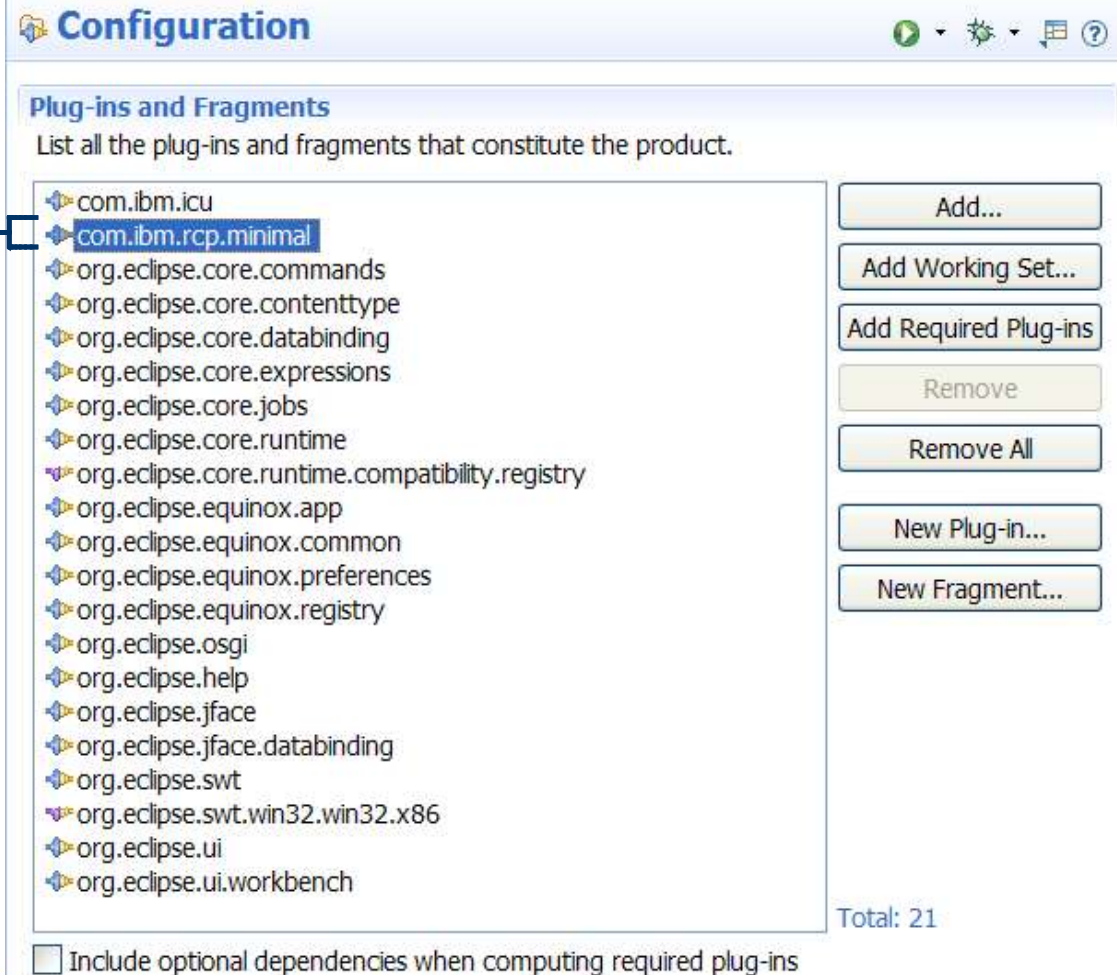
Total: 21

Include optional dependencies when computing required plug-ins

Buttons: Add..., Add Working Set..., Add Required Plug-ins, Remove, Remove All, New Plug-in..., New Fragment...

Core Platform Plug-ins

My Test Plug-in



Configuration

Plug-ins and Fragments
List all the plug-ins and fragments that constitute the product.

- com.ibm.icu
- com.ibm.rcp.minimal**
- org.eclipse.core.commands
- org.eclipse.core.contenttype
- org.eclipse.core.databinding
- org.eclipse.core.expressions
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility.registry
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.osgi
- org.eclipse.help
- org.eclipse.jface
- org.eclipse.jface.databinding
- org.eclipse.swt
- org.eclipse.swt.win32.win32.x86
- org.eclipse.ui
- org.eclipse.ui.workbench

Total: 21

Include optional dependencies when computing required plug-ins

Buttons: Add..., Add Working Set..., Add Required Plug-ins, Remove, Remove All, New Plug-in..., New Fragment...

Core Platform Plug-ins

My Test Plug-in

OSGI Platform

- Component model
- Eclipse Adaptors
- Registry Framework

Configuration

Plug-ins and Fragments
List all the plug-ins and fragments that constitute the product.

- com.ibm.icu
- com.ibm.rcp.minimal**
- org.eclipse.core.commands
- org.eclipse.core.contenttype
- org.eclipse.core.databinding
- org.eclipse.core.expressions
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility.registry
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.osgi
- org.eclipse.help
- org.eclipse.jface
- org.eclipse.jface.databinding
- org.eclipse.swt
- org.eclipse.swt.win32.win32.x86
- org.eclipse.ui
- org.eclipse.ui.workbench

Total: 21

Include optional dependencies when computing required plug-ins

Add...

Add Working Set...

Add Required Plug-ins

Remove

Remove All

New Plug-in...

New Fragment...

Core Platform Plug-ins

My Test Plug-in

Eclipse Runtime

- Plug-in Model
- Preferences
- Scheduling
- Core services

OSGI Platform

- Component model
- Eclipse Adaptors
- Registry Framework

Configuration

Plug-ins and Fragments

List all the plug-ins and fragments that constitute the product.

com.ibm.icu
com.ibm.rcp.minimal
org.eclipse.core.commands
org.eclipse.core.contenttype
org.eclipse.core.databinding
org.eclipse.core.expressions
org.eclipse.core.jobs
org.eclipse.core.runtime
org.eclipse.core.runtime.compatibility.registry
org.eclipse.equinox.app
org.eclipse.equinox.common
org.eclipse.equinox.preferences
org.eclipse.equinox.registry
org.eclipse.osgi
org.eclipse.help
org.eclipse.jface
org.eclipse.jface.databinding
org.eclipse.swt
org.eclipse.swt.win32.win32.x86
org.eclipse.ui
org.eclipse.ui.workbench

Total: 21

Include optional dependencies when computing required plug-ins

Add...

Add Working Set...

Add Required Plug-ins

Remove

Remove All

New Plug-in...

New Fragment...

Core Platform Plug-ins

My Test Plug-in

Eclipse Runtime

- Plug-in Model
- Preferences
- Scheduling
- Core services

OSGI Platform

- Component model
- Eclipse Adaptors
- Registry Framework

UI Framework

- Native Widget API
- MVC Controllers
- Workbench UI artifacts

Configuration

Plug-ins and Fragments

List all the plug-ins and fragments that constitute the product.

- ▶ com.ibm.icu
- ▶ **com.ibm.rcp.minimal**
- ▶ org.eclipse.core.commands
- ▶ org.eclipse.core.contenttype
- ▶ org.eclipse.core.databinding
- ▶ org.eclipse.core.expressions
- ▶ org.eclipse.core.jobs
- ▶ org.eclipse.core.runtime
- ▶ org.eclipse.core.runtime.compatibility.registry
- ▶ org.eclipse.equinox.app
- ▶ org.eclipse.equinox.common
- ▶ org.eclipse.equinox.preferences
- ▶ org.eclipse.equinox.registry
- ▶ org.eclipse.osgi
- ▶ org.eclipse.help
- ▶ org.eclipse.jface
- ▶ org.eclipse.jface.databinding
- ▶ org.eclipse.swt
- ▶ org.eclipse.swt.win32.win32.x86
- ▶ org.eclipse.ui
- ▶ org.eclipse.ui.workbench

Total: 21

Include optional dependencies when computing required plug-ins

Add...

Add Working Set...

Add Required Plug-ins

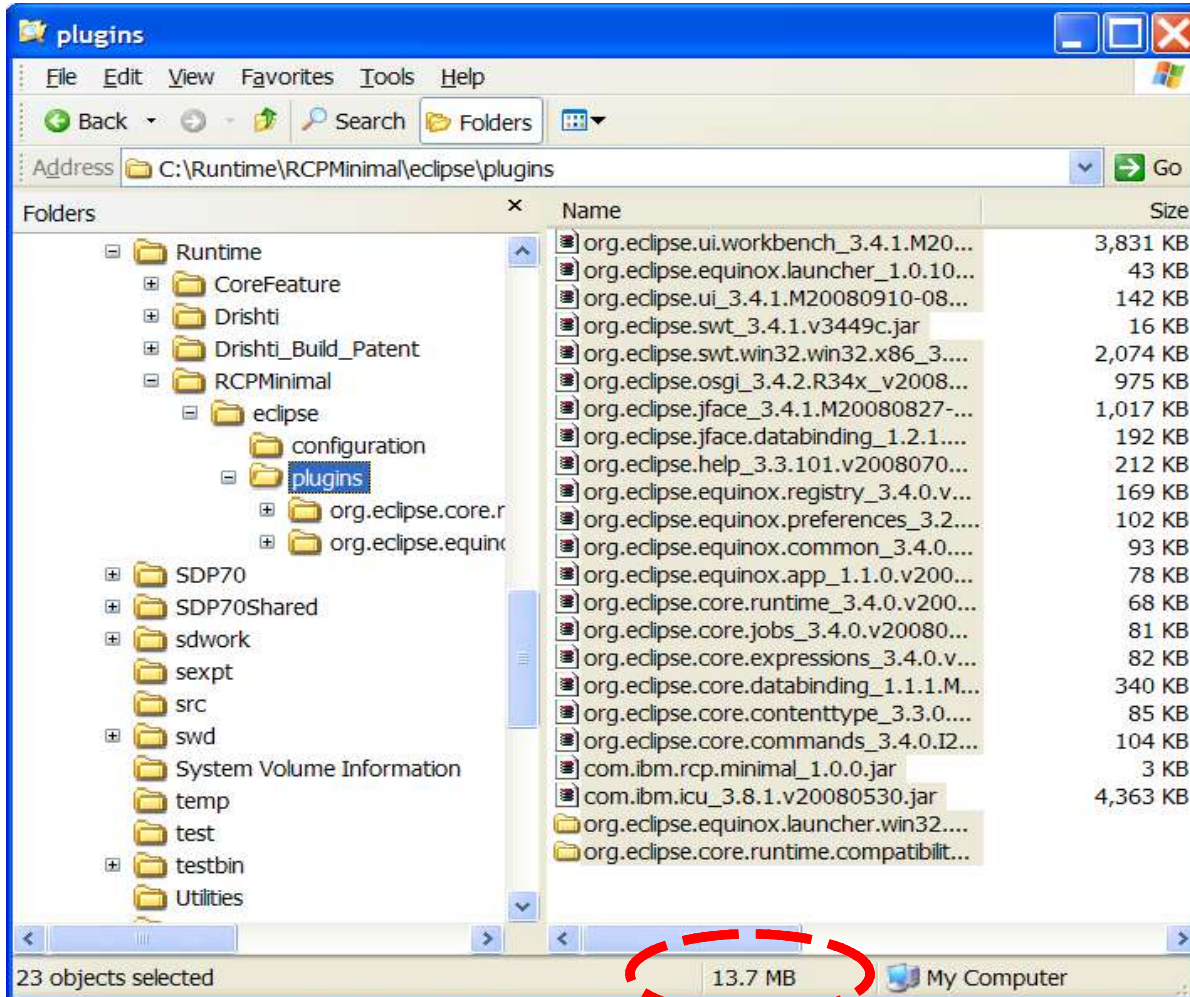
Remove

Remove All

New Plug-in...

New Fragment...

Trivia on memory footprint



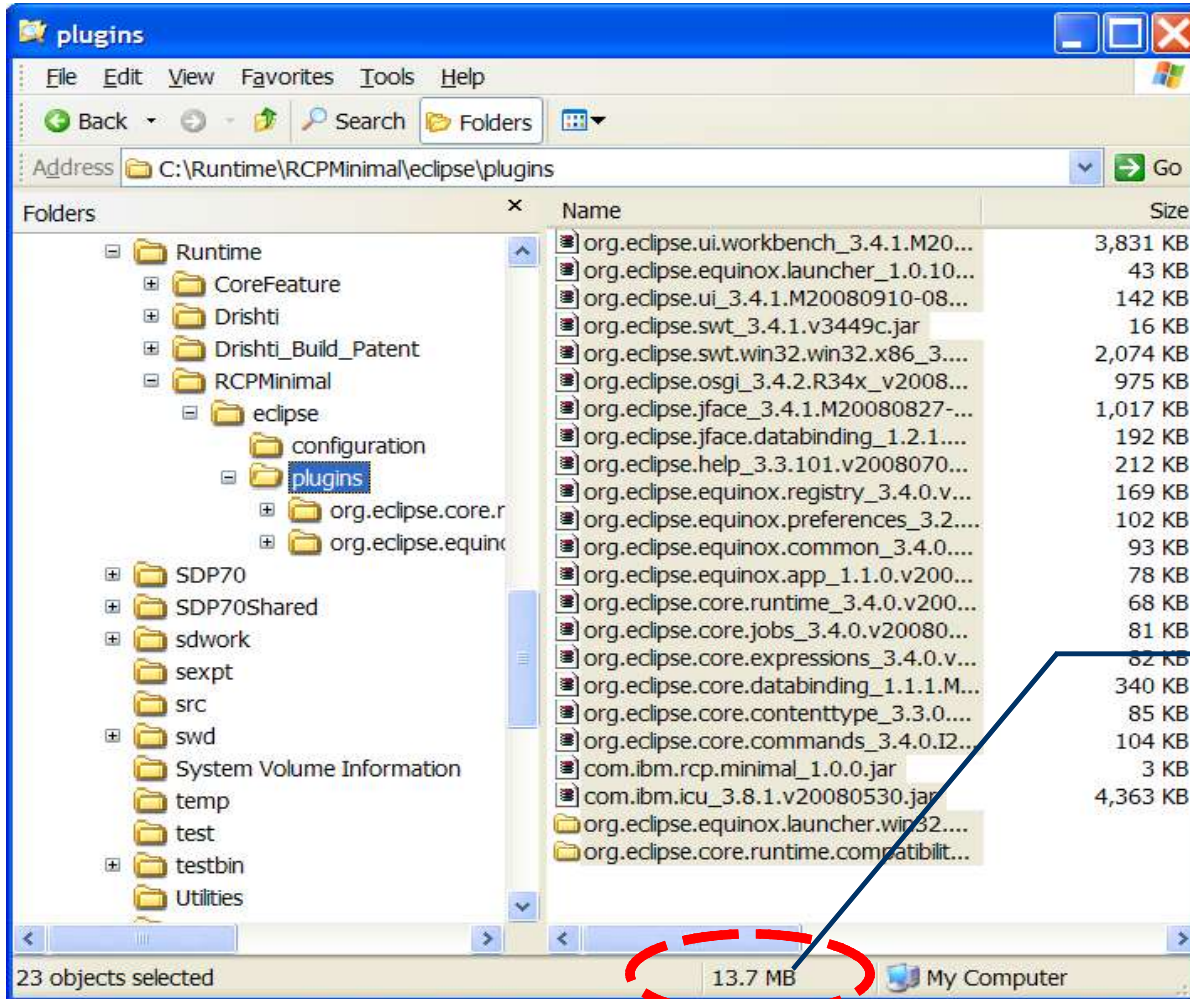
Name	Size
org.eclipse.ui.workbench_3.4.1.M20...	3,831 KB
org.eclipse.equinox.launcher_1.0.10...	43 KB
org.eclipse.ui_3.4.1.M20080910-08...	142 KB
org.eclipse.swt_3.4.1.v3449c.jar	16 KB
org.eclipse.swt.win32.win32.x86_3...	2,074 KB
org.eclipse.osgi_3.4.2.R34x_v2008...	975 KB
org.eclipse.jface_3.4.1.M20080827-...	1,017 KB
org.eclipse.jface.databinding_1.2.1...	192 KB
org.eclipse.help_3.3.101.v2008070...	212 KB
org.eclipse.equinox.registry_3.4.0.v...	169 KB
org.eclipse.equinox.preferences_3.2...	102 KB
org.eclipse.equinox.common_3.4.0...	93 KB
org.eclipse.equinox.app_1.1.0.v200...	78 KB
org.eclipse.core.runtime_3.4.0.v200...	68 KB
org.eclipse.core.jobs_3.4.0.v20080...	81 KB
org.eclipse.core.expressions_3.4.0.v...	82 KB
org.eclipse.core.databinding_1.1.1.M...	340 KB
org.eclipse.core.contenttype_3.3.0...	85 KB
org.eclipse.core.commands_3.4.0.I2...	104 KB
com.ibm.rcp.minimal_1.0.0.jar	3 KB
com.ibm.icu_3.8.1.v20080530.jar	4,363 KB
org.eclipse.equinox.launcher.win32...	
org.eclipse.core.runtime.compatibil...	

23 objects selected

13.7 MB

My Computer

Trivia on memory footprint



The screenshot shows a Windows Explorer window titled 'plugins' with the address bar set to 'C:\Runtime\RCPMinimal\eclipse\plugins'. The left pane shows a tree view of folders, with 'plugins' selected under the 'eclipse' folder. The right pane displays a list of files with their names and sizes. A red dashed circle highlights the '13.7 MB' value at the bottom of the window, indicating the total size of the selected files.

Name	Size
org.eclipse.ui.workbench_3.4.1.M20...	3,831 KB
org.eclipse.equinox.launcher_1.0.10...	43 KB
org.eclipse.ui_3.4.1.M20080910-08...	142 KB
org.eclipse.swt_3.4.1.v3449c.jar	16 KB
org.eclipse.swt.win32.win32.x86_3...	2,074 KB
org.eclipse.osgi_3.4.2.R34x_v2008...	975 KB
org.eclipse.jface_3.4.1.M20080827-...	1,017 KB
org.eclipse.jface.databinding_1.2.1...	192 KB
org.eclipse.help_3.3.101.v2008070...	212 KB
org.eclipse.equinox.registry_3.4.0.v...	169 KB
org.eclipse.equinox.preferences_3.2...	102 KB
org.eclipse.equinox.common_3.4.0...	93 KB
org.eclipse.equinox.app_1.1.0.v200...	78 KB
org.eclipse.core.runtime_3.4.0.v200...	68 KB
org.eclipse.core.jobs_3.4.0.v20080...	81 KB
org.eclipse.core.expressions_3.4.0.v...	82 KB
org.eclipse.core.databinding_1.1.1.M...	340 KB
org.eclipse.core.contenttype_3.3.0...	85 KB
org.eclipse.core.commands_3.4.0.I2...	104 KB
com.ibm.rcp.minimal_1.0.0.jar	3 KB
com.ibm.icu_3.8.1.v20080530.jar	4,363 KB
org.eclipse.equinox.launcher.win32...	
org.eclipse.core.runtime.compatibilit...	

23 objects selected

13.7 MB

*If this is too big,
explore eRCP or
embedded RCP.*

Before beginning RCP

- ***Must have***
- Core Java Knowledge 😊
- Understanding of eclipse plug-in architecture
- Basics of plug-in development
- A decent pc and a will to explore 😊
- ***Nice to have***
- Fluency with SWT/JFace
- Previous experience with eclipse runtime
- A good command on design patterns – MVC, Interfaces, Abstractions

Tools

- JRE preferably 1.5+
- Eclipse IDE – Preferably latest
- A browser and a good internet connection 😊
(so u can have tutorials/articles at your disposal)

What would you need to do - Intuitively

Assuming you have figured out your application logic

1. Create components
 - plug-ins ...of course.!!
2. Provide launching point
 - so eclipse platform can invoke your code on launch
3. Configure main application window on launch
 - window layout, size, look and feel
- Provide branding information
- Create deployable for distribution
- Wait for user action to execute your RCP and trigger your application code 😊

Task 1: Component Architecture

- Based on OSGi modules for the Java platform
 - Dynamic and flexible
 - Loose coupling of Java modules
- OSGi Bundles (Eclipse Plug-ins)
- Eclipse Features – Important for Development and deployment lifecycle
 - Enable multiple branding
 - Enable product lines
 - Ease of deployments and updates

Task 1: Tradeoffs while planning for plug-ins

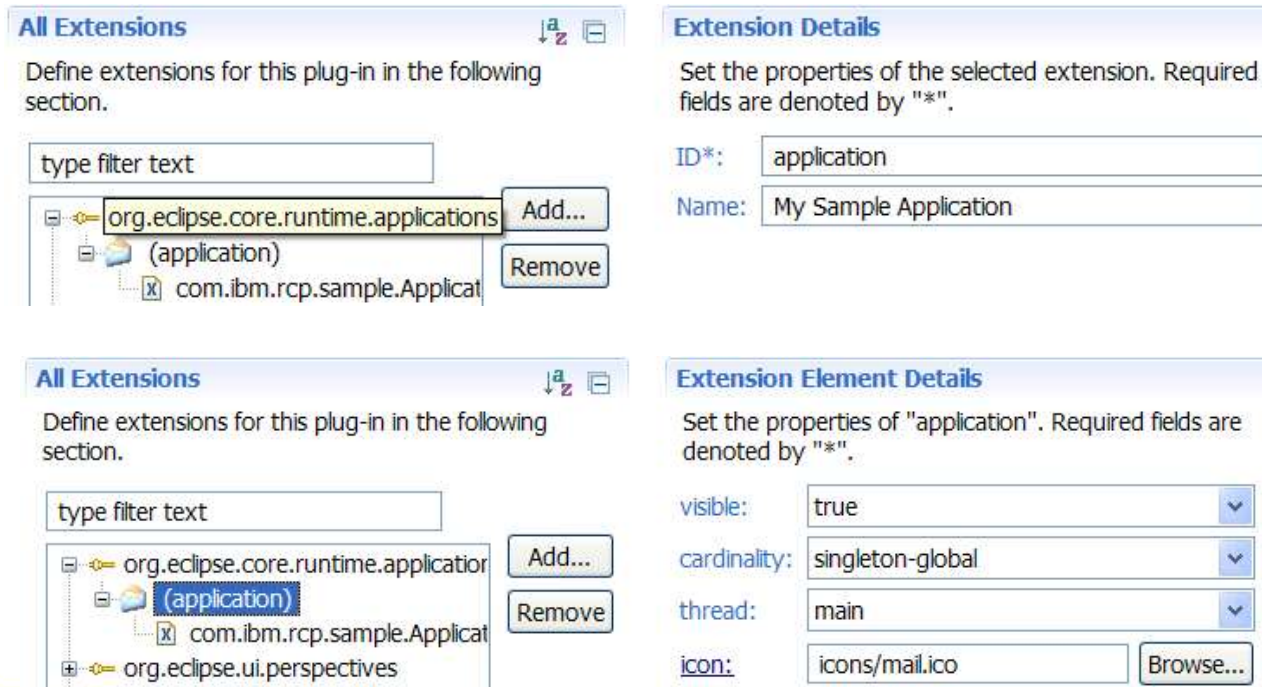
- Bundle Size
- Grouping of logical functionality
- Monolithic vs. Smaller granular bundles
- What goes into what features

Task 2: Launching point – converse with rcp

- Announce yourself to rcp
- Extension point
 - `org.eclipse.core.runtime.applications`

Task 2: Launching point – converse with rcp

- Announce yourself to rcp
- Extension point
 - `org.eclipse.core.runtime.applications`



The image shows two screenshots of the Eclipse IDE's 'All Extensions' and 'Extension Details' panels.

Top Screenshot:

- All Extensions:** Shows a tree view with the extension point `org.eclipse.core.runtime.applications` selected. Below it, the extension `com.ibm.rcp.sample.Applicat` is listed. Buttons for 'Add...' and 'Remove' are visible.
- Extension Details:** Shows the configuration for the selected extension. The ID is `application` and the Name is `My Sample Application`.

Bottom Screenshot:

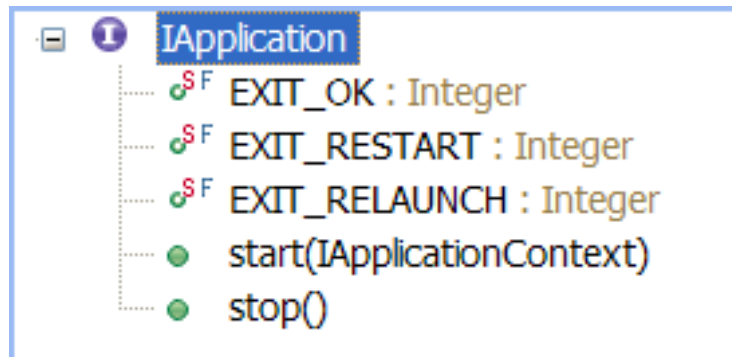
- All Extensions:** Shows the same tree view, but now the `(application)` extension is selected. The `com.ibm.rcp.sample.Applicat` extension is still visible below it.
- Extension Element Details:** Shows the configuration for the selected `(application)` extension. The properties are:
 - `visible:` `true`
 - `cardinality:` `singleton-global`
 - `thread:` `main`
 - `icon:` `icons/mail.ico` (with a 'Browse...' button)

Task 2: Launching point – converse with rcp

- Talk rcp's language
- Interface
 - IApplication
- Catch the hooks it provides for lifecycle management

Task 2: Launching point – converse with rcp

- Talk rcp's language
- Interface
 - IApplication
- Catch the hooks it provides for lifecycle management



Task 3: Configure application window

- Talk rcp's language

Task 3: Configure application window

- Talk rcp's language

```
public Object start(IApplicationContext context) {
    Display display = PlatformUI.createDisplay();
    try {
        int returnCode = PlatformUI.createAndRunWorkbench(display, new
        ApplicationWorkbenchAdvisor());
        if (returnCode == PlatformUI.RETURN_RESTART) {
            return IApplication.EXIT_RESTART;
        }
        return IApplication.EXIT_OK;
    } finally {
        display.dispose();
    }
}
```

Task 3: Configure application window

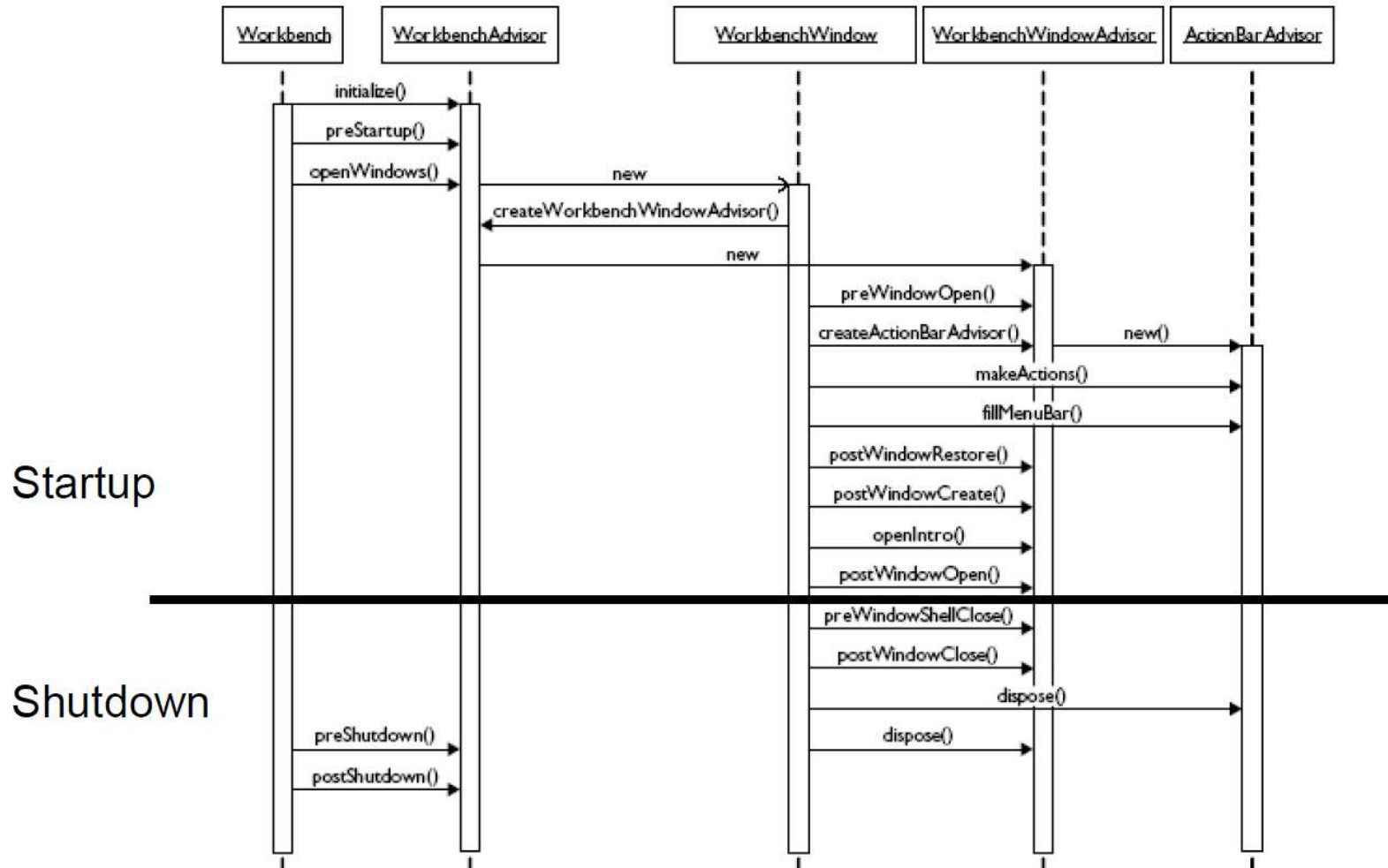
- Talk rcp's language

```
public Object start(IApplicationContext context) {
    Display display = PlatformUI.createDisplay();
    try {
        int returnCode = PlatformUI.createAndRunWorkbench(display, new
        ApplicationWorkbenchAdvisor());
        if (returnCode == PlatformUI.RETURN_RESTART) {
            return IApplication.EXIT_RESTART;
        }
        return IApplication.EXIT_OK;
    } finally {
        display.dispose();
    }
}
```

Extends:

- *WorkbenchAdvisor*
- *Starting point to set up main workbench window info*
- *perspective info*
- *actions*
- *menubar*

Task 3: Configure application window



Task 4: Product and Branding

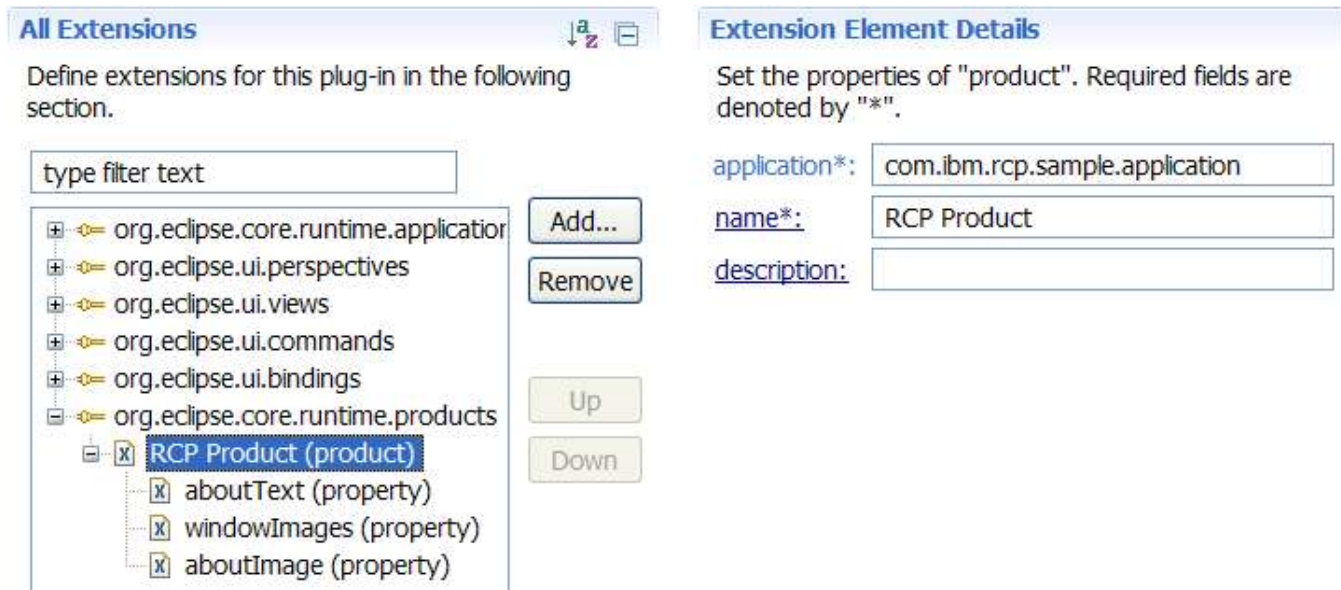
- What do you need to tell eclipse
 - My product name
 - My product icons
 - My splash images
 - My about info
 - Launcher name
 - Title bar label
 -on and on...
- Almost as simple as “fill in the blanks”

Task 4: Product and Branding

- Product Configuration
 - .product files
- Extension point
 - org.eclipse.runtime.products

Task 4: Product and Branding

- Product Configuration
 - .product files
- Extension point
 - org.eclipse.runtime.products



The screenshot displays the Eclipse IDE's configuration interface for an extension point. The 'All Extensions' panel on the left shows a tree view of available extension points. The 'org.eclipse.core.runtime.products' extension point is expanded, and 'RCP Product (product)' is selected. The 'Extension Element Details' panel on the right shows the configuration for the selected extension point, with required fields marked with an asterisk.

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.core.runtime.applicator
- org.eclipse.ui.perspectives
- org.eclipse.ui.views
- org.eclipse.ui.commands
- org.eclipse.ui.bindings
- org.eclipse.core.runtime.products
 - RCP Product (product)**
 - aboutText (property)
 - windowImages (property)
 - aboutImage (property)

Extension Element Details

Set the properties of "product". Required fields are denoted by "*".




application*: com.ibm.rcp.sample.application

name*: RCP Product

description:

Task 4: Product and Branding

Overview

Product Definition

This section describes general information about the product.

Specify the name that appears in the title bar of the application.

Name:

Specify the product identifier.

ID: New...

Specify the product version.





Version:

Specify the application to run when launching this product.

Application:

The [product configuration](#) is based on: plug-ins features

Testing

1. [Synchronize](#) this configuration with the product's defining plug-in.
2. Test the product by launching a runtime instance of it:
 -  [Launch a RAP Application](#)
 -  [Launch an Eclipse application](#)
 -  [Launch a RAP Application in Debug mode](#)
 -  [Launch an Eclipse application in Debug mode](#)

Exporting

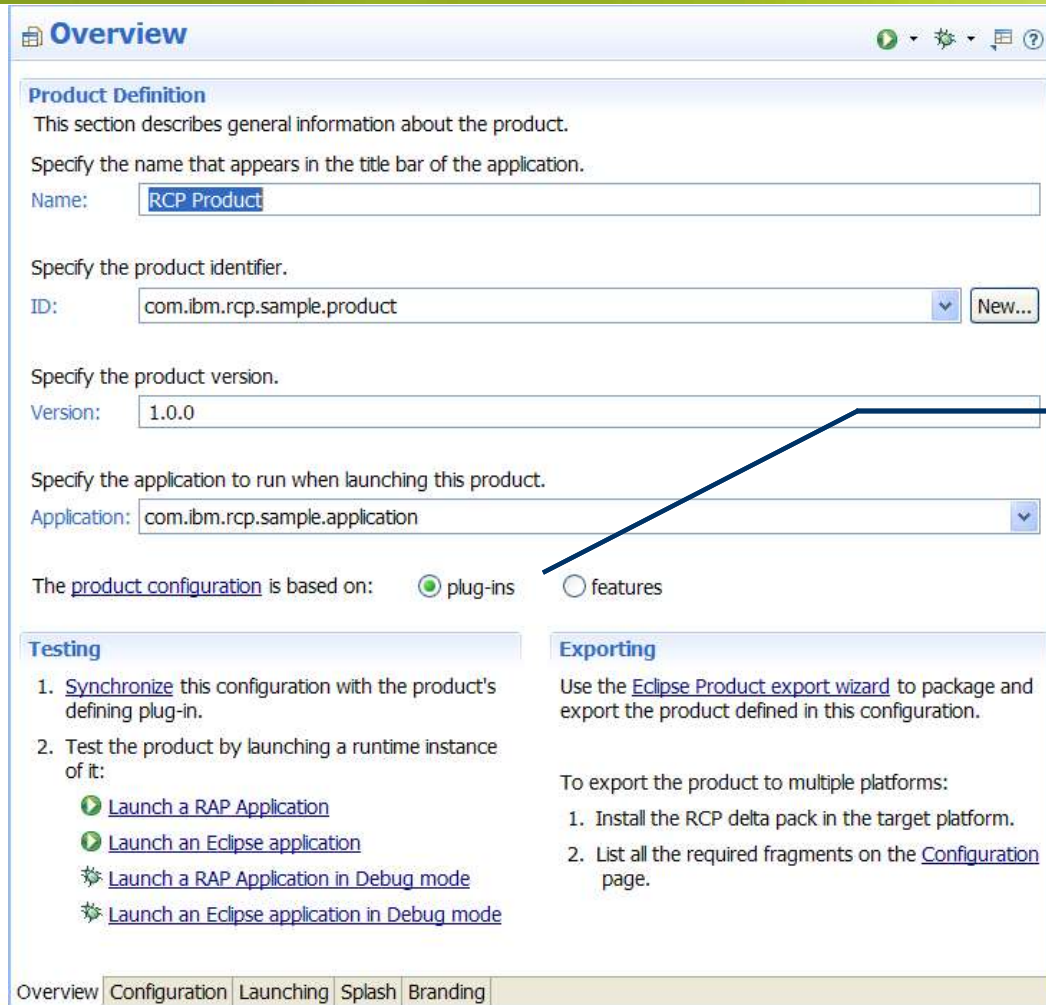
Use the [Eclipse Product export wizard](#) to package and export the product defined in this configuration.

To export the product to multiple platforms:

1. Install the RCP delta pack in the target platform.
2. List all the required fragments on the [Configuration](#) page.

Overview
Configuration
Launching
Splash
Branding

Task 4: Product and Branding



Overview

Product Definition
This section describes general information about the product.
Specify the name that appears in the title bar of the application.
Name:

Specify the product identifier.
ID:

Specify the product version.
Version:

Specify the application to run when launching this product.
Application:

The product configuration is based on: plug-ins features

Testing

1. [Synchronize](#) this configuration with the product's defining plug-in.
2. Test the product by launching a runtime instance of it:
 - [Launch a RAP Application](#)
 - [Launch an Eclipse application](#)
 - [Launch a RAP Application in Debug mode](#)
 - [Launch an Eclipse application in Debug mode](#)

Exporting
Use the [Eclipse Product export wizard](#) to package and export the product defined in this configuration.

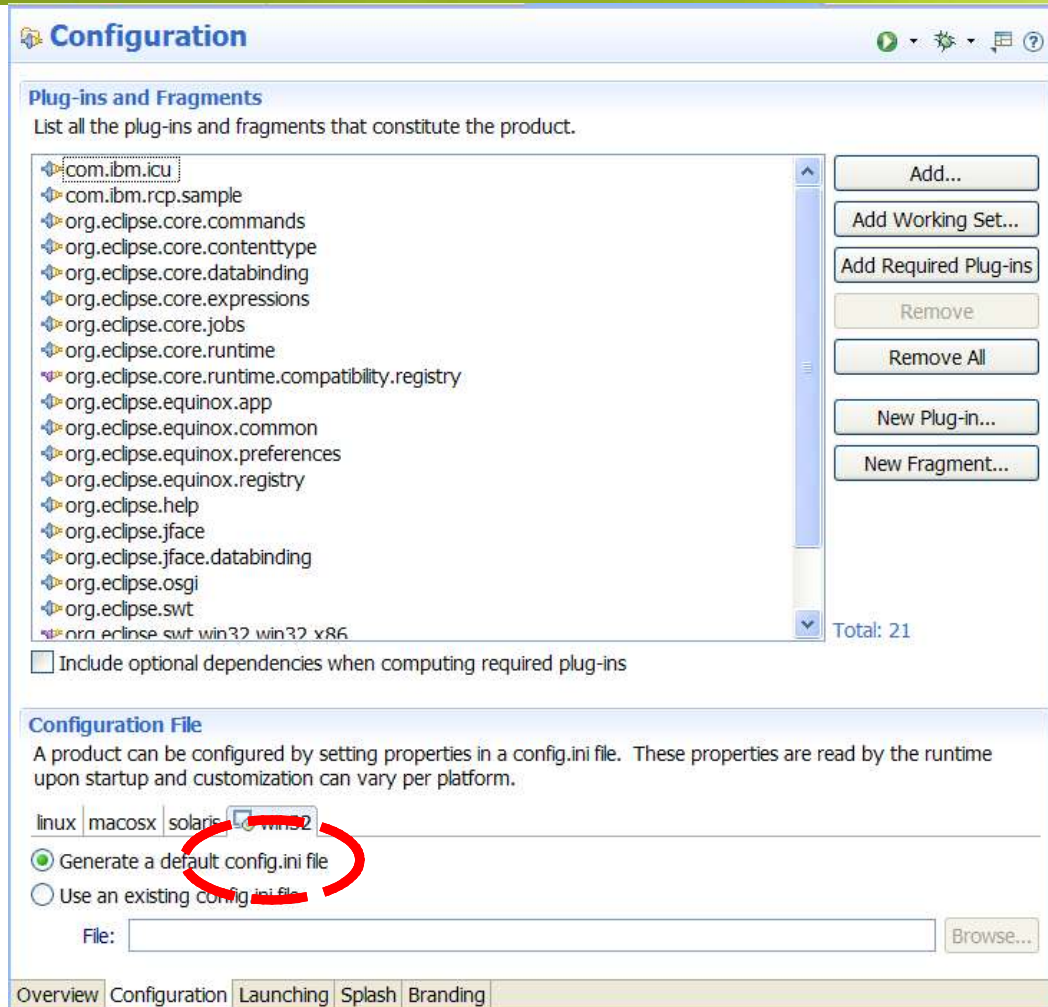
To export the product to multiple platforms:

1. Install the RCP delta pack in the target platform.
2. List all the required fragments on the [Configuration](#) page.

Overview | Configuration | Launching | Splash | Branding

Features mode is recommended for easier management & update capabilities

Task 4: Product and Branding



Configuration

Plug-ins and Fragments
List all the plug-ins and fragments that constitute the product.

- com.ibm.icu
- com.ibm.rcp.sample
- org.eclipse.core.commands
- org.eclipse.core.contenttype
- org.eclipse.core.databinding
- org.eclipse.core.expressions
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility.registry
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.help
- org.eclipse.jface
- org.eclipse.jface.databinding
- org.eclipse.osgi
- org.eclipse.swt
- org.eclipse.swt.win32.win32.x86

Total: 21

Include optional dependencies when computing required plug-ins

Configuration File
A product can be configured by setting properties in a config.ini file. These properties are read by the runtime upon startup and customization can vary per platform.

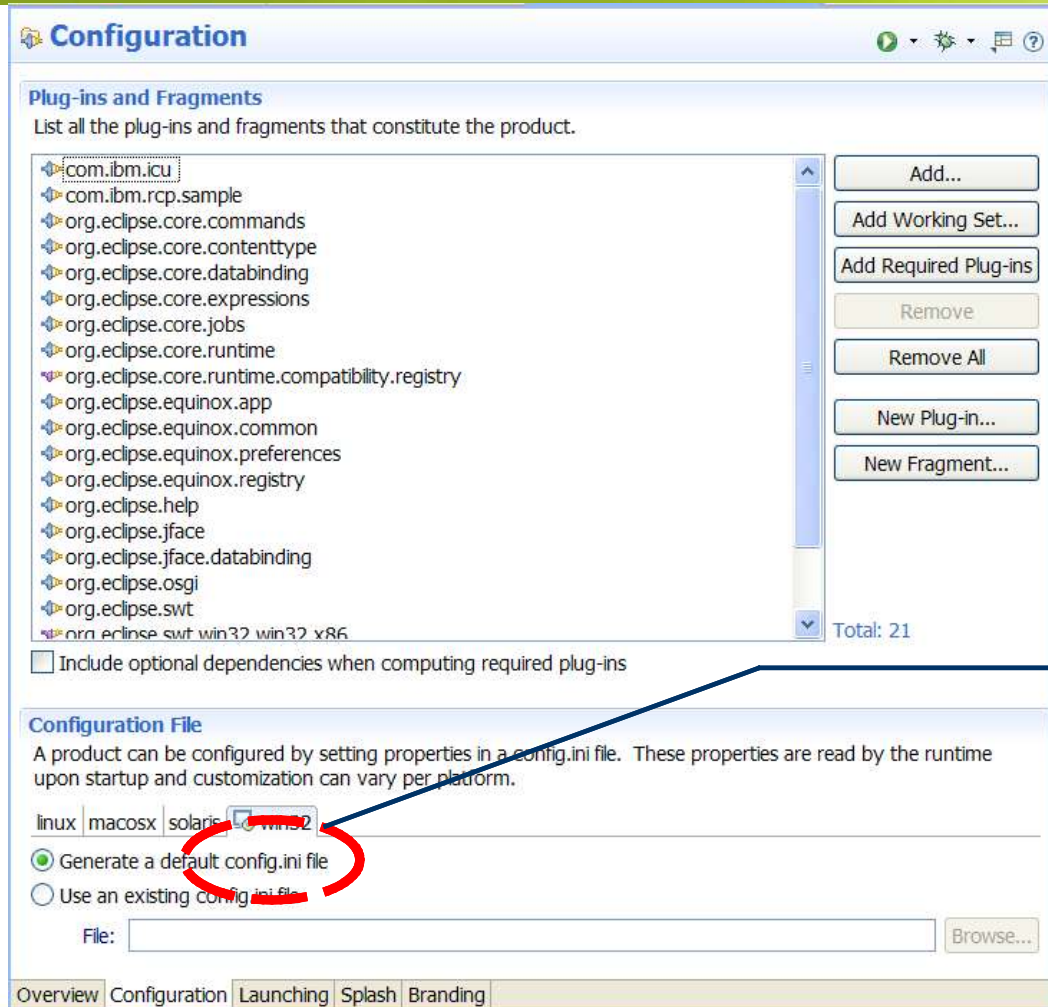
linux | macosx | solaris | **win32**

Generate a default config.ini file
 Use an existing config.ini file

File: Browse...

Overview | Configuration | Launching | Splash | Branding

Task 4: Product and Branding



Configuration

Plug-ins and Fragments
List all the plug-ins and fragments that constitute the product.

- com.ibm.icu
- com.ibm.rcp.sample
- org.eclipse.core.commands
- org.eclipse.core.contenttype
- org.eclipse.core.databinding
- org.eclipse.core.expressions
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility.registry
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.help
- org.eclipse.jface
- org.eclipse.jface.databinding
- org.eclipse.osgi
- org.eclipse.swt
- org.eclipse.swt.win32.win32.x86

Total: 21

Include optional dependencies when computing required plug-ins

Configuration File
A product can be configured by setting properties in a config.ini file. These properties are read by the runtime upon startup and customization can vary per platform.

linux | macosx | solaris | **win32**

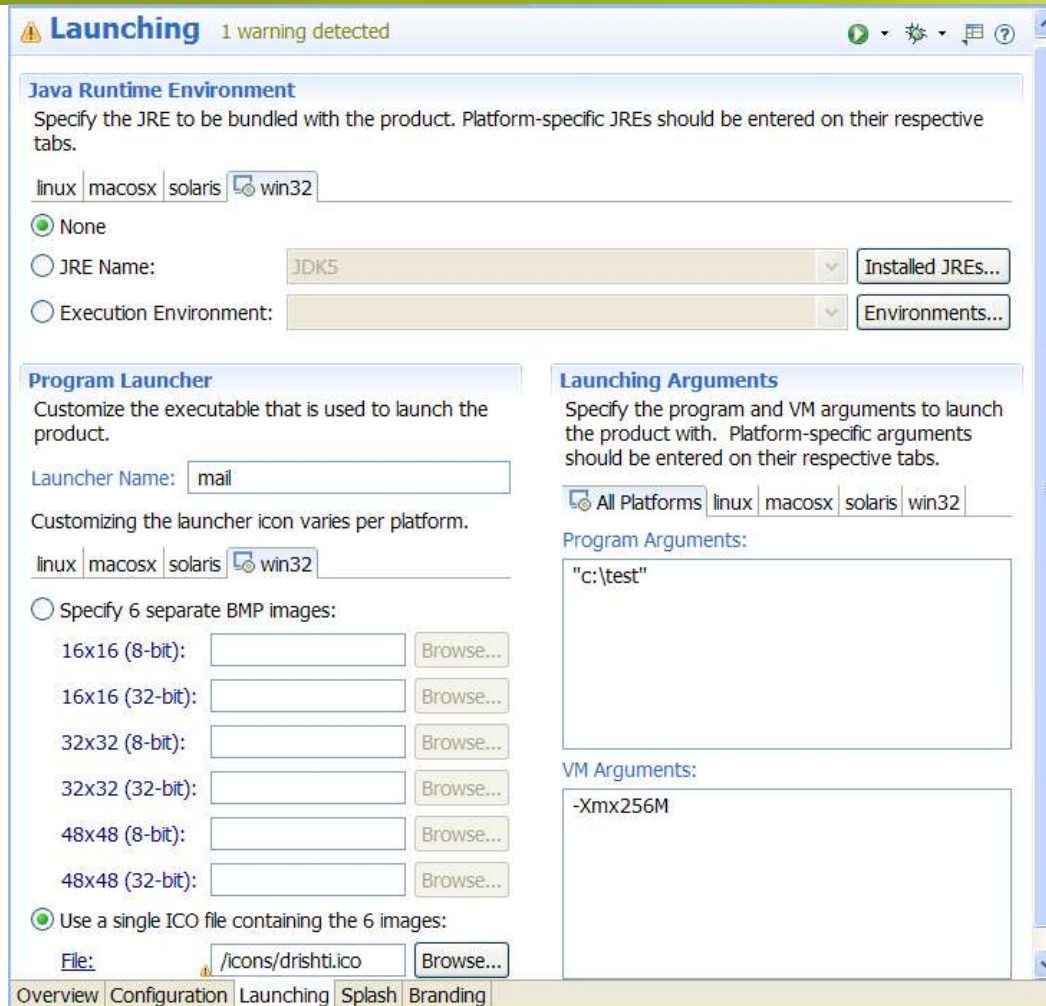
Generate a default config.ini file
 Use an existing config.ini file

File:

Overview | Configuration | Launching | Splash | Branding

.ini file
- initialization params, like product location, vm arguments, etc.

Task 4: Product and Branding



Launching 1 warning detected

Java Runtime Environment
Specify the JRE to be bundled with the product. Platform-specific JREs should be entered on their respective tabs.

linux | macosx | solaris | **win32**

None

JRE Name:

Execution Environment:

Program Launcher
Customize the executable that is used to launch the product.

Launcher Name:

Customizing the launcher icon varies per platform.

linux | macosx | solaris | **win32**

Specify 6 separate BMP images:

16x16 (8-bit):

16x16 (32-bit):

32x32 (8-bit):

32x32 (32-bit):

48x48 (8-bit):

48x48 (32-bit):

Use a single ICO file containing the 6 images:

File:

Launching Arguments
Specify the program and VM arguments to launch the product with. Platform-specific arguments should be entered on their respective tabs.

All Platforms | linux | macosx | solaris | win32

Program Arguments:

VM Arguments:

Overview | Configuration | **Launching** | Splash | Branding

Task 4: Product and Branding

Launching 1 warning detected

Java Runtime Environment
Specify the JRE to be bundled with the product. Platform-specific JREs should be entered on their respective tabs.

linux | macosx | solaris | win32

None

JRE Name: JDK5

Execution Environment:

Program Launcher
Customize the executable that is used to launch the product.

Launcher Name: mail

Customizing the launcher icon varies per platform.

linux | macosx | solaris | win32

Specify 6 separate BMP images:

16x16 (8-bit):

16x16 (32-bit):

32x32 (8-bit):

32x32 (32-bit):

48x48 (8-bit):

48x48 (32-bit):

Use a single ICO file containing the 6 images:

File: /icons/drishti.ico

Launching Arguments
Specify the program and VM arguments to launch the product with. Platform-specific arguments should be entered on their respective tabs.

All Platforms | linux | macosx | solaris | win32

Program Arguments:
"c:\test"

VM Arguments:
-Xmx256M

Overview | Configuration | **Launching** | Splash | Branding

*Name of executable –
mail.exe*

Task 4: Product and Branding

Launching 1 warning detected

Java Runtime Environment
Specify the JRE to be bundled with the product. Platform-specific JREs should be entered on their respective tabs.

linux | macosx | solaris | **win32**

None
 JRE Name:
 Execution Environment:

Program Launcher
Customize the executable that is used to launch the product.

Launcher Name:

Customizing the launcher icon varies per platform.

linux | macosx | solaris | **win32**

Specify 6 separate BMP images:
16x16 (8-bit):
16x16 (32-bit):
32x32 (8-bit):
32x32 (32-bit):
48x48 (8-bit):
48x48 (32-bit):
 Use a single ICO file containing the 6 images:
File:

Launching Arguments
Specify the program and VM arguments to launch the product with. Platform-specific arguments should be entered on their respective tabs.

All Platforms | linux | macosx | solaris | win32

Program Arguments:

VM Arguments:

Overview | Configuration | **Launching** | Splash | Branding

*Name of executable –
mail.exe*

Launch arguments

Task 4: Product and Branding

Launching 1 warning detected

Java Runtime Environment
Specify the JRE to be bundled with the product. Platform-specific JREs should be entered on their respective tabs.

linux | macosx | solaris | **win32**

None
 JRE Name: Installed JREs...
 Execution Environment: Environments...

Program Launcher
Customize the executable that is used to launch the product.

Launcher Name:

Customizing the launcher icon varies per platform.

linux | macosx | solaris | **win32**

Specify 6 separate BMP images:
 16x16 (8-bit): Browse...
 16x16 (32-bit): Browse...
 32x32 (8-bit): Browse...
 32x32 (32-bit): Browse...
 48x48 (8-bit): Browse...
 48x48 (32-bit): Browse...
 Use a single ICO file containing the 6 images:
 File: Browse...

Launching Arguments
Specify the program and VM arguments to launch the product with. Platform-specific arguments should be entered on their respective tabs.

All Platforms | linux | macosx | solaris | win32

Program Arguments:

VM Arguments:

Overview | Configuration | **Launching** | Splash | Branding

Name of executable – mail.exe

Optional JRE/ Platform Specification

Launch arguments

Task 4: Product and Branding

Splash

Location
The splash screen appears when the product launches. If its location is not specified, the 'splash.bmp' file is assumed to be in the product's defining plug-in.
Specify the plug-in in which the splash screen is located.
Plug-in:

Customization
Create a custom splash screen using one of the provided templates or by adding a progress bar and message.
Template:

Specify the geometry and color of the progress bar and message.

Add a progress bar
x-offset: y-offset: width: height:

Add a progress message
x-offset: y-offset: width: height: Text Color:

Overview Configuration Launching Splash Branding

Task 4: Product and Branding

Branding 4 warnings detected

Window Images
Specify the images that will be associated with the application window. These images are typically located in the product's defining plug-in.

16x16 Image:

32x32 Image:

48x48 Image:

64x64 Image:

128x128 Image:

About Dialog
Customize the text and image of the About dialog. The image is typically located in the product's defining plug-in and its size must not exceed 500x330 pixels. The text is not shown if the image size exceeds 250x330 pixels.

Image:

Text:

Welcome Page
The welcome page appears the first time the product is launched. It is intended to introduce the features of the product to new users.
Specify this product's welcome page.

Intro ID:

Overview | Configuration | Launching | Splash | Branding

Task 5: Packaging

- Transformation
 - Development form \Rightarrow Deployment form
- For example
- Workspace projects \Rightarrow Built JARs
- Possible outputs
 - Stand-alone, runnable application in directory or archive
 - Update site features and plug-ins
 - JNLP/WebStart application
 - Installers (e.g., InstallShield) not directly supported
 - Eclipse IDE provides wizard for all the above

Task 5: Packaging with Security

- Sign jars to avoid unauthorized updates
- Signing jars imperative to run webstart with all security permissions on JVM
- IDE has out of the box support for signing jars
- Nice tool for keystore and certificate mgmt

Update Management

- Add plug-ins to rcp:
 - org.eclipse.update.core
 - org.eclipse.update.ui
- Setup features properly
- All base plug-ins which are not written by you go in one feature e.g.
 - <yourdomain>.eclipse.base.feature
- Your main application plug-ins go in one (or more) e.g.
 - yourdomain.eclipse.yourapplication.feature
- Setup an update site using eclipse's update site project.
- Build features into update site

Update Management

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(new Shell(
    display, SWT.NONE));
    dialog.run(false, false, new IRunnableWithProgress() {
        public void run(IProgressMonitor monitor) {
            UpdateCommand command = new UpdateCommand(
                "<yourdomain>.eclipse.<yourfeature>,name", "false");
            command.run(monitor);
        }
    });
```

Update Management

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(new Shell(
    display, SWT.NONE));
    dialog.run(false, false, new IRunnableWithProgress() {
        public void run(IProgressMonitor monitor) {
            UpdateCommand command = new UpdateCommand(
                "<yourdomain>.eclipse.<yourfeature>,name", "false");
            command.run(monitor);
        }
    });
```

May not work in debug mode.

Update Management

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(new Shell(
    display, SWT.NONE));
    dialog.run(false, false, new IRunnableWithProgress() {
        public void run(IProgressMonitor monitor) {
            UpdateCommand command = new UpdateCommand(
                "<yourdomain>.eclipse.<yourfeature>,name", "false");
            command.run(monitor);
        }
    });
```

May not work in debug mode.

New Update architecture based on Equinox P2 Provisioning System is available

Update Management

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(new Shell(
    display, SWT.NONE));
    dialog.run(false, false, new IRunnableWithProgress() {
        public void run(IProgressMonitor monitor) {
            UpdateCommand command = new UpdateCommand(
                "<yourdomain>.eclipse.<yourfeature>,name", "false");
            command.run(monitor);
        }
    });
```

May not work in debug mode.

New Update architecture based on Equinox P2 Provisioning System is available

Not so popular in RCPs yet...!!

Update Management

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(new Shell(
    display, SWT.NONE));
    dialog.run(false, false, new IRunnableWithProgress() {
        public void run(IProgressMonitor monitor) {
            UpdateCommand command = new UpdateCommand(
                "<yourdomain>.eclipse.<yourfeature>,name", "false");
            command.run(monitor);
        }
    });
```

May not work in debug mode.

New Update architecture based on Equinox P2 Provisioning System is available

IMHO: Inno Setup, Installshield are also possible options

Not so popular in RCPs yet...!!

Update Management – Some Tradeoffs

- Where to call update from
 - From custom actions
 - From update UI – Not necessary for small RCPs
 - From startup code
 - From scheduled code
- Whether to use update at all??
 - Update plug-ins add to memory footprint

Help

- Very straightforward, like in eclipse IDE
- Article on using help with RCP
- Needs
 - org.apache.lucene
 - org.eclipse.help.appserver
 - org.eclipse.help.base
 - org.eclipse.help.ui
 - org.eclipse.help.webapp
 - org.eclipse.tomcat
 - org.eclipse.ui.forms

Closing thoughts

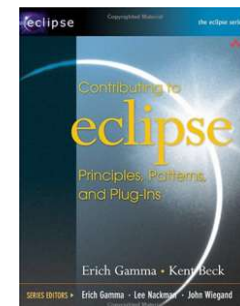
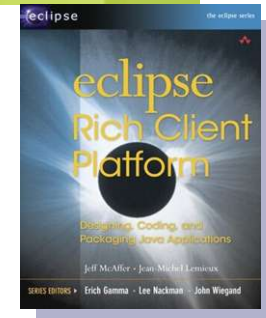
- *“In **theory**, there is no difference between **theory** and **practice**. But, in **practice**, there is”*
 - Jan La Van De Snepscheut
- Master your debugging skills – sit on eclipse’s shoulders...!!
 - Eclipse Debugging Tutorial
- **Please...please..try to understand eclipse platform architecture in detail**
 - Extension point architecture
 - Registries
 - Bundle loading, class loading

Some must reads for RCP Developers

- Official Eclipse RCP FAQ's
- Eclipse RCP How-to
- Eclipse Con 2005 Tutorial
- Eclipse platform whitepaper
- Eclipse plug-in architecture article
- For advanced users >
- Advanced RCP Presentation

Recommended Reading

- Contributing to Eclipse: Principles, Patterns, and Plug-ins
 - By Erich Gamma, Kent Beck
 - Addison-Wesley Professional
 - ISBN: 0321205758
- Eclipse Rich Client Platform
 - By Jeff McAffer and Jean-Michel Lemieux
 - Addison-Wesley Professional
 - ISBN: 0321334612
- SWT : The Standard Widget Toolkit, Volume 1
 - By Steve Northover, Mike Wilson
 - Addison-Wesley Professional
 - ISBN: 0321256638



Take off ...with eclipse RCP...!!!!

Thank you

Anshu Jain

**Technical Staff Member,
IBM India Research Lab,
Bangalore**

anshu.jain@in.ibm.com | anshu.n.jain@gmail.com

