Introduction to Eclipse Rich Client Platform

William Bath
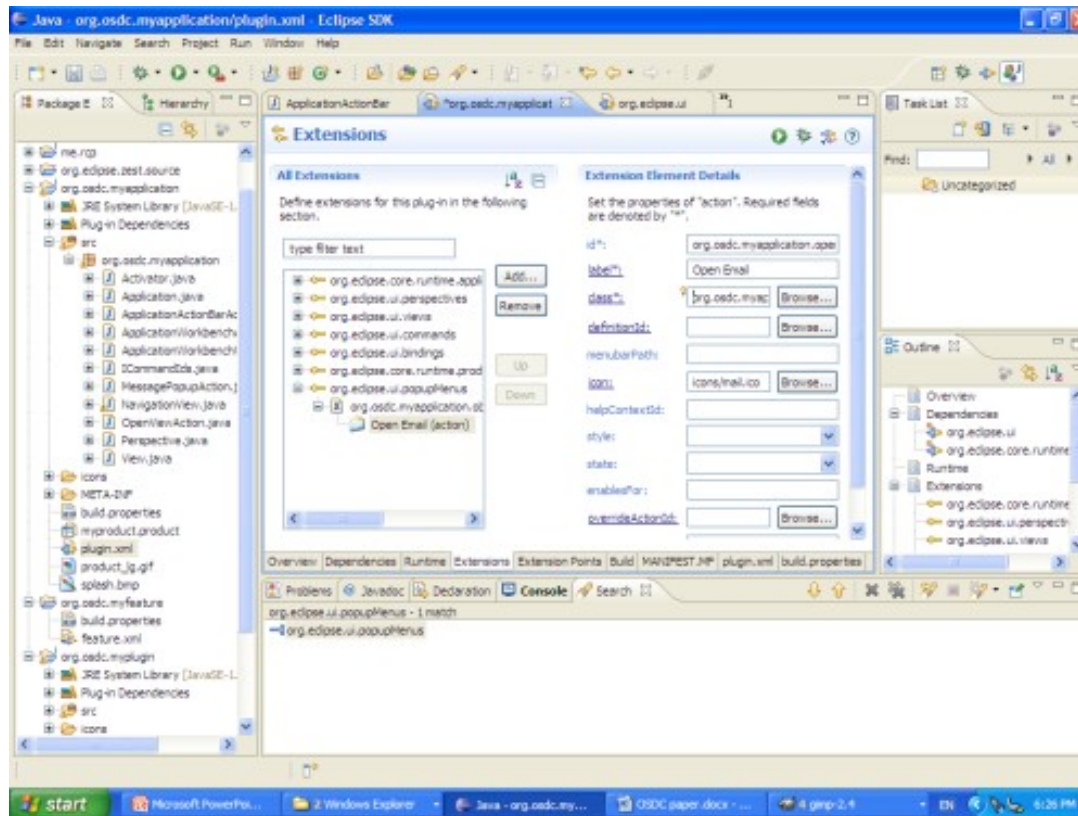
william.bath@gmail.com

What should you get out of this presentation?

- What is Eclipse RCP?

- Why should you use it?

- Core technologies

- Understand its structure
  - Plug-ins, Features and Products
  - IApplication and Co.
  - Extension Registry

- Understand the makeup of the UI
  - Perspectives, Views, Editors and the Workbench
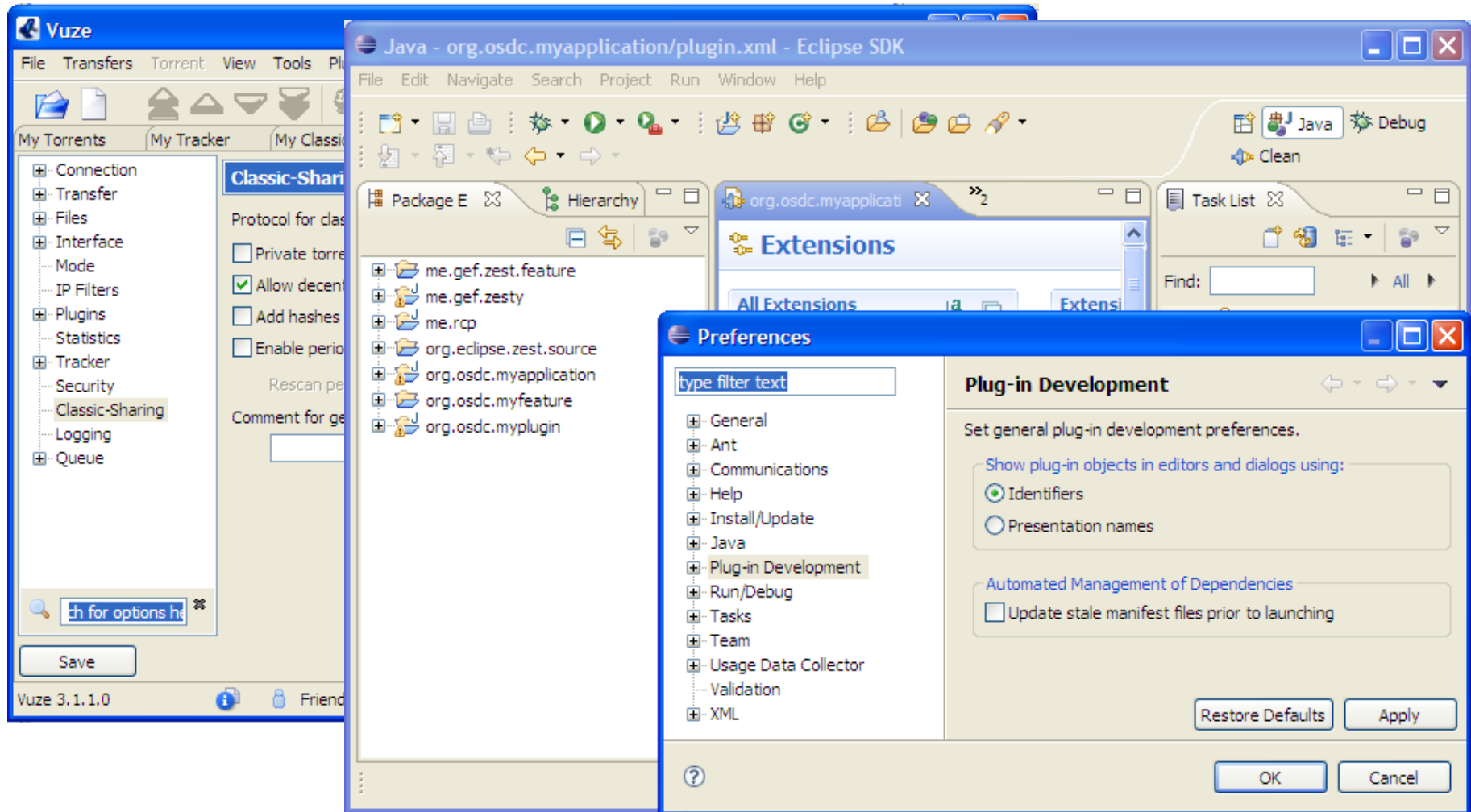
- What you need to get started

What is Eclipse RCP?

- **This is Eclipse**, world-famous Java IDE and generally cool tool. **RCP: SPOT THE DIFFERENCE?**

What is Eclipse RCP?

- Can be though of as: stripping out the IDE specific components of Eclipse and reusing the generic components with your own custom plug-ins

- Is in fact: A generic extensible scalable GUI(?) client that facilitates extremely rapid and sustainable development.

- Translation: A set of generic components that can be customised to create your application.
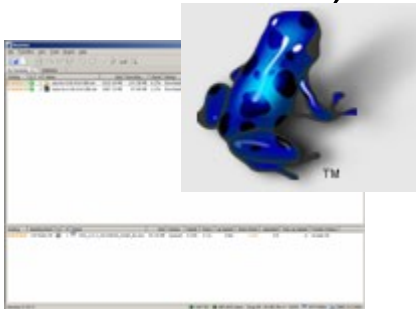
What is Eclipse RCP? - Examples

Why should you use it? – Sell it to the managers

- Well supported by industry with over 170 members
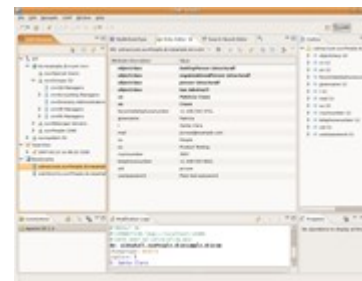

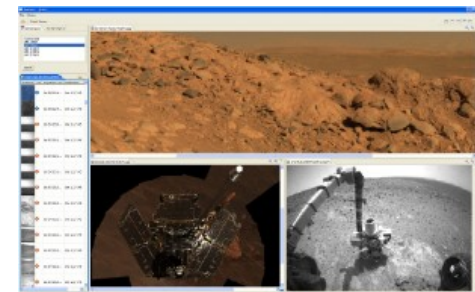- Well known RCP applications (but did you know they were RCP?)

Azureus/Vuze BitTorrent Client

IBM Lotus Expeditor/Symphony

Apache Directory Studio

NASA C/C for Mars Rovers, Phoenix etc

Why should you use it? – The good reasons

- **Plug-in based**
  - Modular design, easy breakup of work, encapsulated, reusable

- **Manages GUI layout**
  - Persistent layouts, and sizing, user configurable layouts (perspectives)

- **Extensible Interface**
  - The main menu, tool bar, status bar can be incrementally added to from plug-ins

- **Code reuse**
  - Commands, menus, command handlers, core expressions, re-targetable actions etc.

- **Update support**

Why should you use it? – The good reasons

- **Branding**

  - Splash screens, intro screen, icon sets, cheat sheets, No need to have anything "Eclipse" about it

- **Extensible help**

- **Manages the mundane**

  - Preference store, update manager, tips and tricks, navigation

- **Access to a world of plug-ins**

  - Leverage other plug-ins, your application is improved as Eclipse improves (force multiplier!)

- **Much more, just ask, look and find; chances are if it is not application specific it is in Eclipse!**
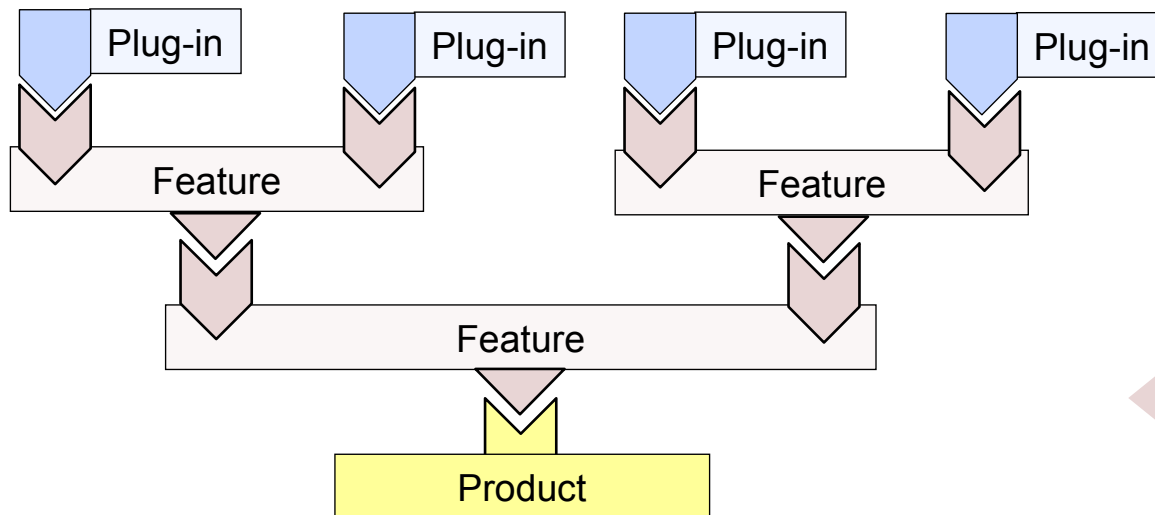
Core Technologies

- # OSGI

  - Provides the plug-in based environment for Eclipse

  - Is a standard of which Equinox is the implementation Eclipse uses

  - Permits lazy-loading of plug-ins

  - Permits multiple versions of the same plug-ins/packages to coexist

  - Every Eclipse Plug-in is an OSGI plug-in

  - Every OSGI plug-in can be treated as a standard JAR outside OSGI

Core Technologies

SWT/JFace

- SWT is a Platform independent Widget toolkit that wraps native Widgets offering native feel and performance.

- JFace provides Model-View-Controller wrapping for SWT widgets such as TreeViewer, TableViewer etc.

- Swing is arguably the main competing framework with SWT. Many religious arguments with no clear winner in performance.

- Widgets must be disposed – in practice not a big deal

Under the hood…

Plug-ins

- ## Holds code

- ## Resources

- ## Three magical files:

  - build.properties

  - MANIFEST.MF

  - plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
   <extension
      point="org.eclipse.ui.views">
      <category
         name="Sample Category"
         id="org.osdc.myplugin">
      </category>
      <view
         name="Sample View"
         icon="icons/sample.gif"
         category="org.osdc.myplugin"
         class="org.osdc.myplugin.views.SampleView"
         id="org.osdc.myplugin.views.SampleView">
      </view>
   </extension>
</plugin>
```

```
source.. = src/
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Myplugin Plug-in
Bundle-SymbolicName: org.osdc.myplugin;
singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator:
org.osdc.myplugin.Activator
Require-Bundle: org.eclipse.ui,
org.eclipse.core.runtime
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment:
JavaSE-1.6
```

Features

- ## Groups Plug-ins

- ## Groups Features

- ## Specifies Versions

- ## Two magical files:
  - build.properties
  - feature

```xml
<?xml version="1.0" encoding="UTF-8"?>
<feature
    id="org.osdc.myfeature"
    label="Myfeature Feature"

    version="1.0.0">

<description url="http://www.example.com/description">
  [Enter Feature Description here.]
</description>

<copyright url="http://www.example.com/copyright">
  [Enter Copyright Description here.]
</copyright>

<license url="http://www.example.com/license">
  [Enter License Description here.]
</license>

  <requires>
    <import plugin="org.eclipse.ui"/>
    <import plugin="org.eclipse.core.runtime"/>
  </requires>

  <plugin
      id="org.osdc.myplugin"
    download-size="0"
    install-size="0"
      version="0.0.0"
    unpack="false"/>

  <plugin
      id="org.osdc.myapplication"
    download-size="0"
    install-size="0"
    version="0.0.0"
    unpack="false"/>

</feature>
```

org.osdc.myfeature
build.properties
feature.xml

bin.includes = feature.xml
src.includes = feature.xml

Product

- # One XML file:

  - myapplication.product

```
<?xml version="1.0" encoding="UTF-8"?>
<?pde version="3.4"?>

<product name="My RCP Product"
id="org.osdc.myapplication.product"
application="org.osdc.myapplication.application"
version="1.0.0" useFeatures="true">

  <aboutInfo>
    <image path="product_lg.gif"/>
    <text>
      RCP Mail template created by PDE
    </text>
  </aboutInfo>

  <configIni use="default">
  </configIni>

  <launcherArgs>
    <programArgs>-console -consolelog -debug</programArgs>
    <programArgsWin>-console</programArgsWin>
    <vmArgs>-Xmx512m</vmArgs>
    <vmArgsMac>-XstartOnFirstThread
-Dorg.eclipse.swt.internal.carbon.smallFonts</vmArgsMac>
  </launcherArgs>

  <windowImages i16="icons/sample2.gif"/>

  <launcher>
    <solaris/>
    <win useIco="false">
      <bmp/>
```
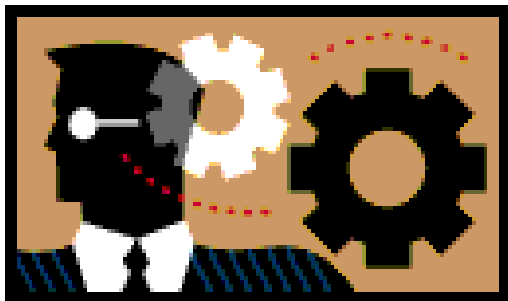
The GUI centre of Eclipse

Indra: "I want a custom email/chat client program to help me work more efficiently"

Bruce: "I will write the XMPP/IRC Chat component"
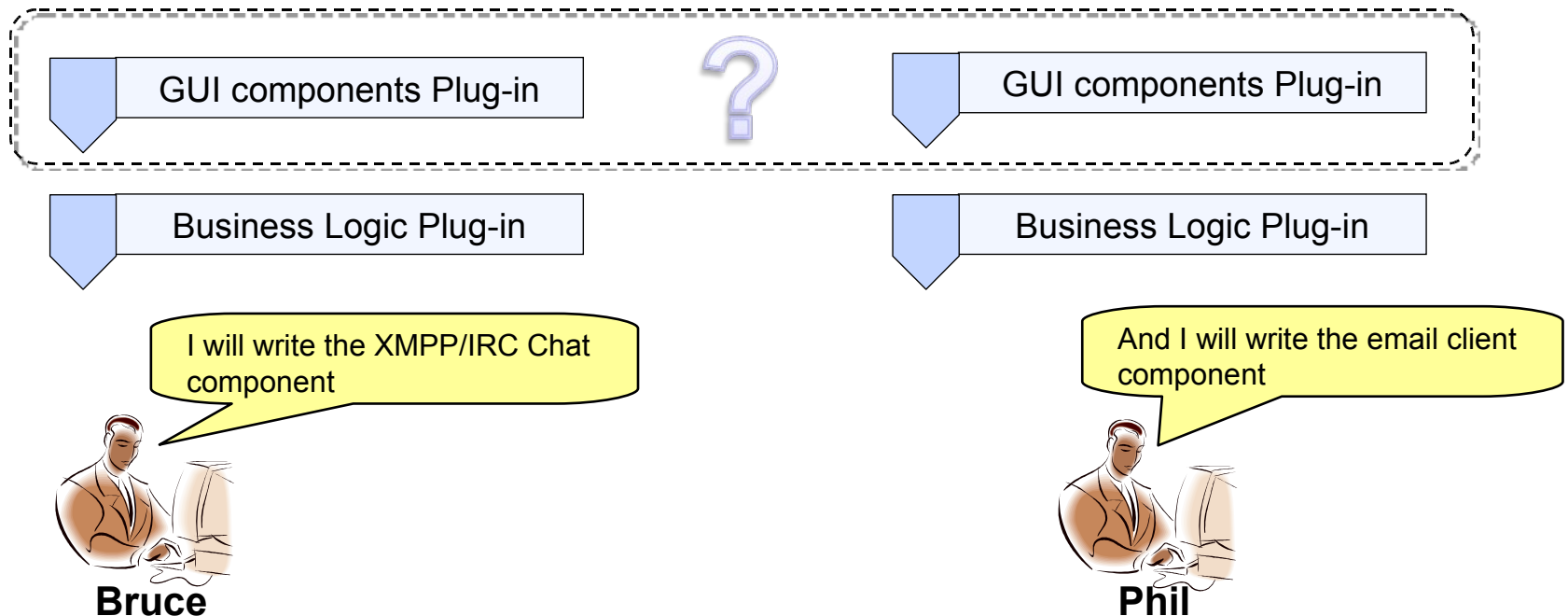
Phil: "And I will write the email client component"
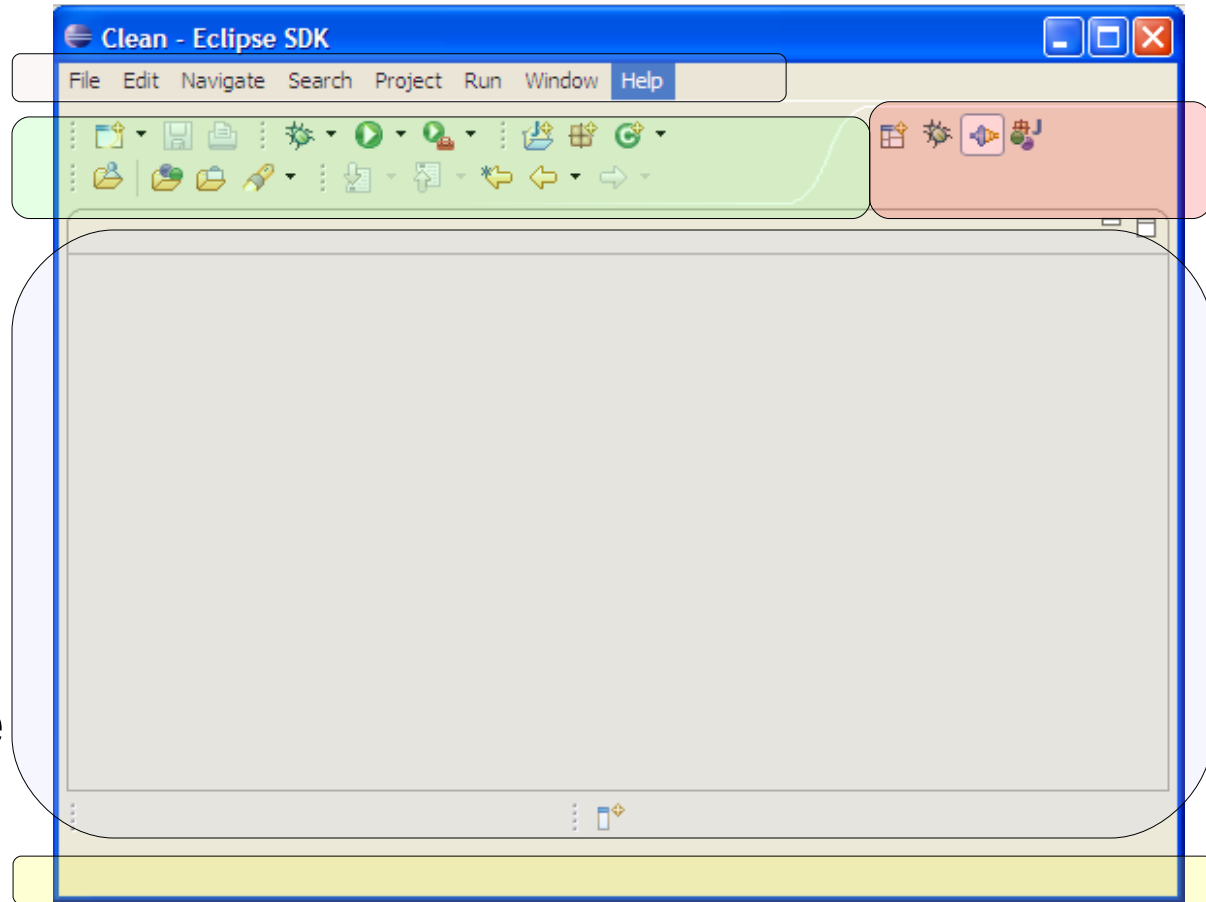
The GUI centre of Eclipse

How do we get disparate GUI components to seamlessly integrate forming one application?

With the WORKBENCH!

GUI components Plug-in	?	GUI components Plug-in
Business Logic Plug-in		Business Logic Plug-in

I will write the XMPP/IRC Chat component

Bruce

And I will write the email client component

Phil

The Workbench

- Main Menu bar

- Main Cool/Toolbar

- Perspective Bar

- Editor Area

- Statusbar

- Fastview bar, progress bar, etc.

- ## Everything here is designed to be contributed to!
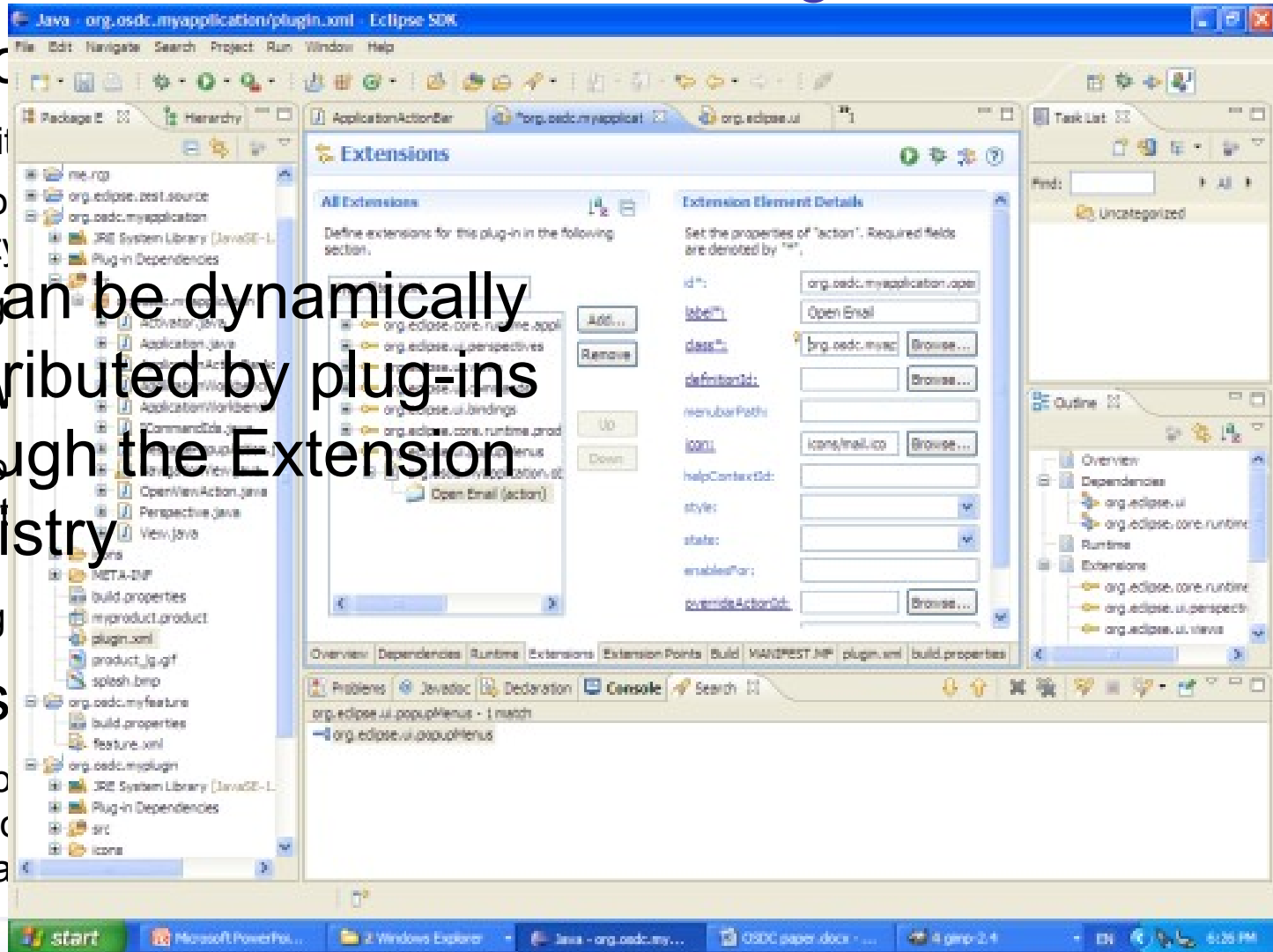
GUI Goodness… Understanding the UI breakdown

- Edit
  - Edit
  - Pro
    dirty
- All can be dynamically
  contributed by plug-ins
  through the Extension
  Registry
    edit
    ii
  - E.g
- Pers
  - Gro
    a co
    a ta

The Extension Registry – What is it?

- A repository of XML globally accessible and modifiable by plug-ins (typically through the Plugin.xml but also through code)

- Broken up into Extension Points and Extensions

- Extension Points are points which Plug-ins expose to permit other plug-ins to extend their functionality

- Extensions are what Plug-ins contribute to Extension Points

The Extension Registry – An Example

- Plug-in org.eclipse.ui exposes extension "org.eclipse.ui.popupMenus" in its Plugin.xml

- Plug-in org.osdc.myapplication uses it in its Plugin.xml

```
<extension-point id="popupMenus"
name="%ExtPoint.popupMenus"
schema="schema/popupMenus.exsd"/>
```

org.eclipse.ui (plugin.xml)

```
<extension
      point="org.eclipse.ui.popupMenus">
   <objectContribution
      adaptable="false"
      id="org.osdc.myapplication.objectContribution1"
      objectClass="org.osdc.emailplugin.Email">
      <action
         class="org.osdc.myapplication.Action1"
         icon="icons/mail.ico"
         id="org.osdc.myapplication.openemailaction"
         label="Open Email"
         tooltip="Opens an email">
      </action>
   </objectContribution>
</extension>
```

org.osdc.myapplication (plugin.xml)
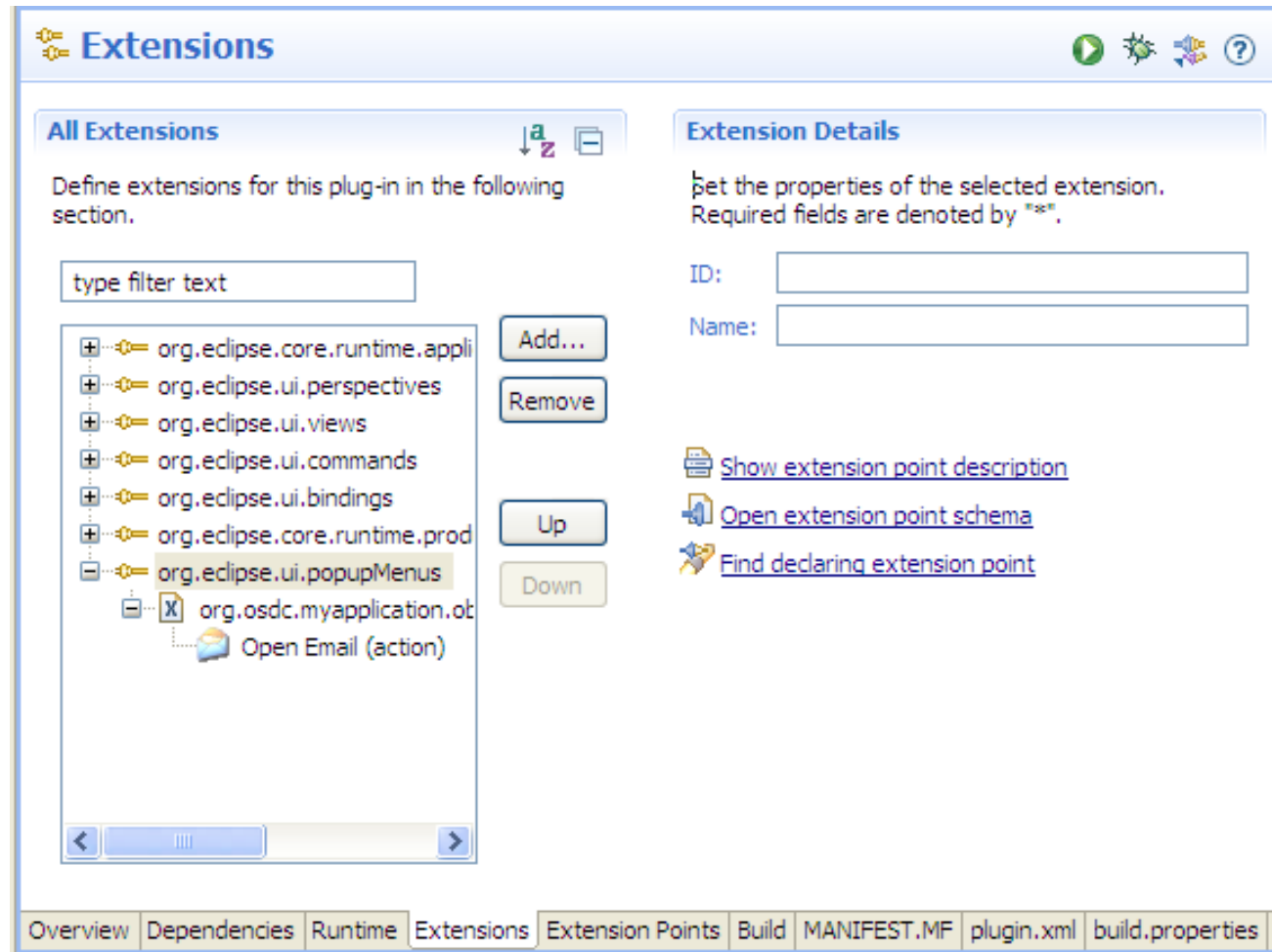
So you want to be an RCP developer…

- Boy there is a lot of XML to be written/maintained

- Dependency management/ identification

- Bundling all those plug-ins into a application

- Where to start?

- Can be done with a Java compiler, text editor and libraries, but 9/10 nutritionalists recommend….

http://www.eclipse.org/downloads/
Select Eclipse for RCP /Plug-in Developers

Plug-in Development Environment

Plug-in Development Environment

Plug-in Development Environment

- Compile time detection

    - of dependency problems

    - XML issues/mismatch

- Tools for creating new Extension Points etc.

- Tools for building and bundling whole applications, including continuous integration through headless builds

- Really makes development very easy

More Resources

- Get the paper

- http://www.eclipse.org/rcp/

- http://www.eclipse.org/resources/ - Over 300 articles on everything to do with RCP and Eclipse

- http://www.planeteclipse.org/planet/

- http://live.eclipse.org/

- http://www.eclipseplugincentral.com/

- Contact me: william.bath@gmail.com

That's All, Folks!

D otry it…