Eclipse Rich Client Programming with Eclipse 4

Michael Vierhauser

Based on Eclipse Tutorials by Stefan Wallner, Siemens VAI & Wolfgang Heider, JKU

UNIVERSITY OF NOTRE DAME

What is Eclipse ?

UNIVERSITY OF NOTRE DAME

✓ A Development Environment ?

➡ A Framework?

➡ A Desktop Application Platform?

"Eclipse means a lot of different things to different people.
To some Eclipse is a free, state-of-the-art *Java development environment*.
To others, Eclipse is *a flexible environment* *to experiment with new computer languages or extensions to existing languages.*
To yet others, Eclipse is a comprehensive *framework that deploys* *many advanced and modern* *software design and implementation techniques*.*"*

[http://eclipse.org]

- **Eclipse provides IDEs and platforms for nearly every language and architecture**

  Java IDE, C/C++, JavaScript, PHP,……

Platform

Provide base functionality, APIs, and interfaces for frequently used tasks

Consistent look and feel for all tools / views / editors within the platform

Allows the implementation of code that uses functionality that is not known during design time

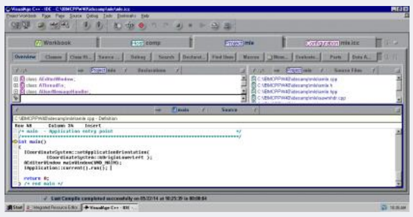
The own components offer extensions points (API) where other components may contribute additional functionality

Platform Tasks

- **Identify, register, load components**
- **Manage component lifecycle (install / uninstall)**
- **Register of extension points and extensions**
- **Resource access (file system, workspace, project, files)**
- **GUI (workbench, views, actionBars, menus, …)**

A Brief History of Eclipse

UNIVERSITY OF NOTRE DAME



- **1980s IBM Visual Age**

.

.

.

- **Nov. 7th 2001 - Eclipse 1.0**

.

- **Feb. 2nd 2004 - Eclipse Foundation was founded**

June – Eclipse 3.0

“A shift toward giving greater power and flexibility to applications built on the Eclipse infrastructure”.

-> Versions: Callisto, Europa, Ganymede, Galileo, Helios, Indigo, Juno.

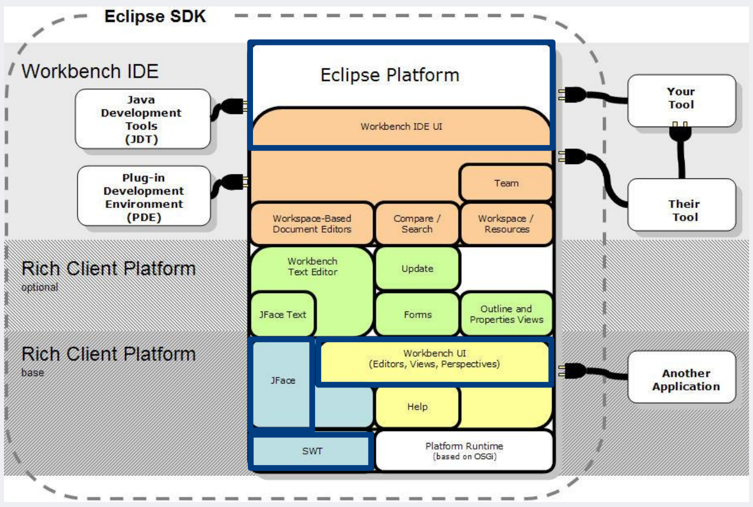
.

- **July 2010 – E4**

-> Versions: Juno, Kepler, Luna, Mars, **Neon**

Eclipse SDK

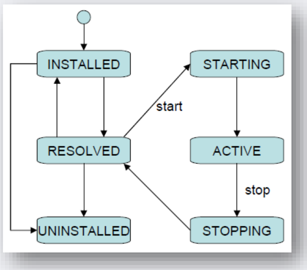
UNIVERSITY OF NOTRE DAME

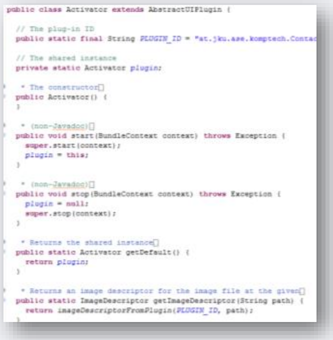


Glossary & Tutorials

OSGi (Open Services Gateway initiative)

- **OSGi specifies a dynamic module system for Java** (http://www.osgi.org)
  - Implementations available from different vendors
- **Eclipse is based on an implementation of the OSGi specification**
  - Since Eclipse 3.0: Equinox project
- **Eclipse plug-in = OSGi bundle**





```
public class Activator extends AbstractUIPlugin {

  // The plug-in ID
  public static final String PLUGIN_ID = "at.jku.ase.komptech.Contac

  // The shared instance
  private static Activator plugin;

   * The constructor
  public Activator() {
  }

   * (non-Javadoc)
  public void start(BundleContext context) throws Exception {
    super.start(context);
    plugin = this;
  }

   * (non-Javadoc)
  public void stop(BundleContext context) throws Exception {
    plugin = null;
    super.stop(context);
  }

   * Returns the shared instance
  public static Activator getDefault() {
    return plugin;
  }

   * Returns an image descriptor for the image file at the given
  public static ImageDescriptor getImageDescriptor(String path) {
    return imageDescriptorFromPlugin(PLUGIN_ID, path);
```

Eclipse RCP

A platform for developing desktop applications provides a (huge) framework for building your own custom tools, editors, visualizations.....

Most parts of the eclipse IDE, components, views... are rather generic and can be easily adapted for your own applications

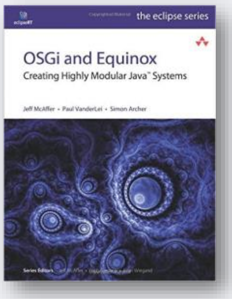
- Extension based on the Eclipse RCP-platform can work together without knowing each other
- Applications can be deployed in various variants with different plug-ins and functionality included
- Tools to implement Eclipse RCP plug-ins / extensions are integrated within the Eclipse IDE (*Eclipse for RCP/Plug-in Developers*).
- Eclipse RCP is under the Eclipse Public License (EPL), an open source license, commercial use of the framework is allowed.

Helpful Links & Tutorials

- **Tom Schindl**
  `https://tomsondev.bestsolution.at/category/eclipse/e4/tutorials-e4/`
  `http://github.com/tomsontom/e4demo/raw/master/tutorial.pdf`
- **Lars Vogel**
  `http://www.vogella.com/tutorials/`
- **Eclipse 4.0 Release Review**
  `http://archive.eclipse.org/projects/www/project-slides/eclipse4.0_release_review_20100721.pdf`
- **Eclipse RCP**
  `http://wiki.eclipse.org/Rich_Client_Platform`
- **Eclipse RCP Tutorials**
  `https://sites.google.com/site/tyroprogramming/java/rcp`
- **SWT**
  `http://www.eclipse.org/swt/snippets`
  `http://www.java2s.com/Code/Java/SWT-JFace-Eclipse/CatalogSWT-JFace-Eclipse.htm`
  `http://www.vogella.com/articles/SWT/article.html`

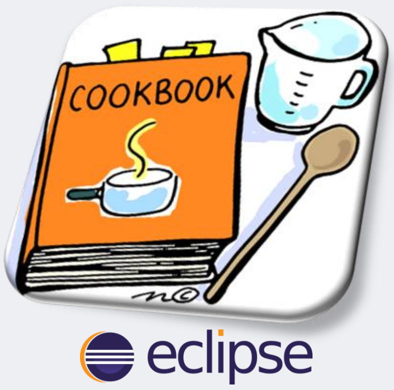
Helpful Links & Tutorials

- **Books (old! – Eclipse 3.x)**
  - OSGi and Equinox: Creating Highly Modular Java Systems
    Jeff McAffer
  - Eclipse Rich Client Platform
    Jeff McAffer


the eclipse series
OSGi and Equinox
Creating Highly Modular Java™ Systems
Jeff McAffer • Paul VanderLei • Simon Archer

the eclipse series
Eclipse Rich Client Platform
Second Edition
Jeff McAffer • Jean-Michel Lemieux • Chris Aniszczyk

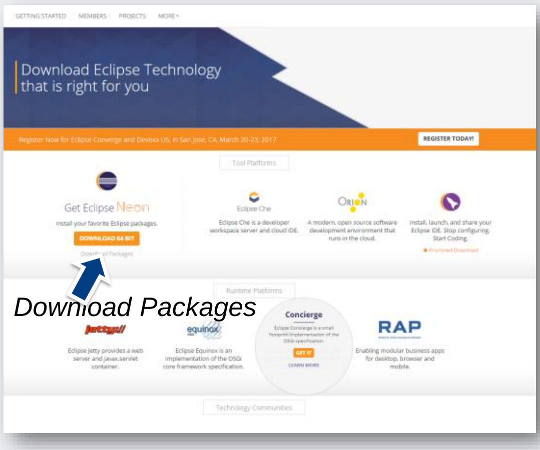
An Eclipse (RCP) Cookbook



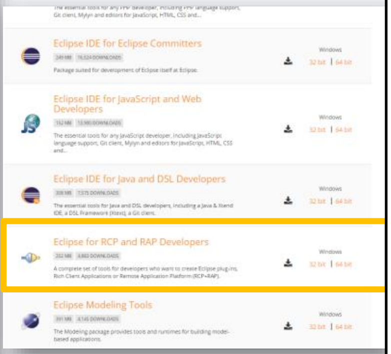
1 Select Eclipse Version

http://eclipse.org/downloads

Eclipse 4.6 Neon! (64 Bit)



Download Packages



The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and...

Eclipse IDE for Eclipse Committers
Package suited for development of Eclipse itself at Eclipse.

Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and...

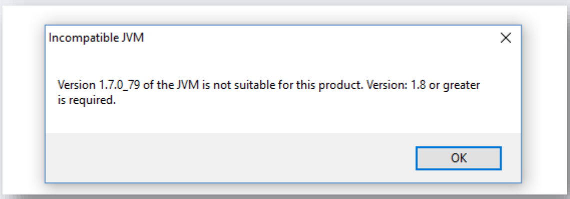
Eclipse IDE for Java and DSL Developers
The essential tools for Java and DSL developers, including a Java & Xtend IDE, a DSL Framework (Xtext), a Git client.

Eclipse for RCP and RAP Developers
A complete set of tools for developers who want to create Eclipse plug-ins, Rich Client Applications or Remote Application Platform (RCP+RAP).

Eclipse Modeling Tools
The Modeling package provides tools and runtimes for building model-based applications.

2 Start Eclipse

- Depending on selected Eclipse Version the used Java Version has to match (>= Java 8)
- Set Version for Eclipse: https://wiki.eclipse.org/Eclipse.ini
  - -vm path to your Java Version (e.g., -vm C:\Java\JDK\1.8\bin\javaw.exe)

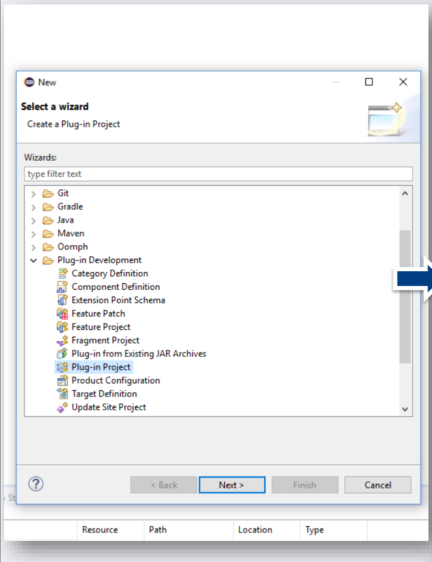


Incompatible JVM

Version 1.7.0_79 of the JVM is not suitable for this product. Version: 1.8 or greater is required.

3 Create Plug-in

UNIVERSITY OF NOTRE DAME


New

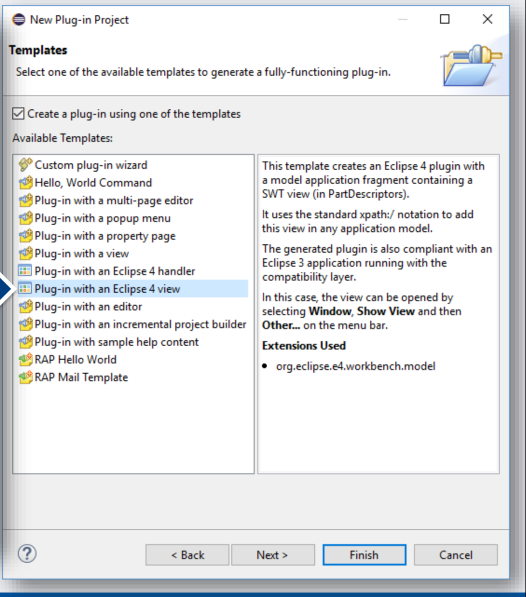
Select a wizard

Create a Plug-in Project

Wizards:

type filter text

- Git
- Gradle
- Java
- Maven
- Oomph
- Plug-in Development
  - Category Definition
  - Component Definition
  - Extension Point Schema
  - Feature Patch
  - Feature Project
  - Fragment Project
  - Plug-in from Existing JAR Archives
  - Plug-in Project
  - Product Configuration
  - Target Definition
  - Update Site Project


New Plug-in Project

Templates

Select one of the available templates to generate a fully-functioning plug-in.

☑ Create a plug-in using one of the templates

Available Templates:

- Custom plug-in wizard
- Hello, World Command
- Plug-in with a multi-page editor
- Plug-in with a popup menu
- Plug-in with a property page
- Plug-in with a view
- Plug-in with an Eclipse 4 handler
- Plug-in with an Eclipse 4 view
- Plug-in with an editor
- Plug-in with an incremental project builder
- Plug-in with sample help content
- RAP Hello World
- RAP Mail Template

This template creates an Eclipse 4 plugin with a model application fragment containing a SWT view (in PartDescriptors).

It uses the standard xpath:/ notation to add this view in any application model.

The generated plugin is also compliant with an Eclipse 3 application running with the compatibility layer.

In this case, the view can be opened by selecting **Window**, **Show View** and then **Other...** on the menu bar.

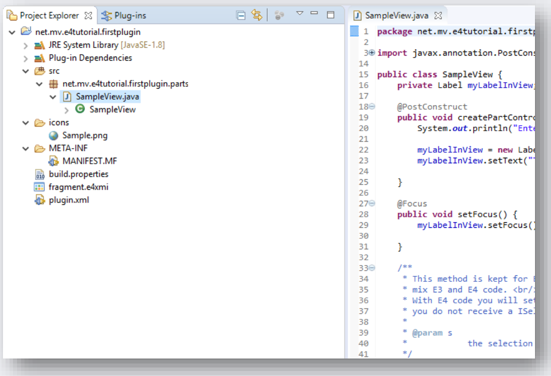
Extensions Used

- org.eclipse.e4.workbench.model

Important Artifacts

UNIVERSITY OF NOTRE DAME

- **Plug-in Manifest**
  - MANIFEST.MF
- **Plug-in Description**
  - plugin.xml
- **Workbench Model + Fragments (new in E4)**
  - xy.e4xmi
  - Name
  - ID
  - Version
  - Dependencies !
  - Extensions & Extension Points
  - Java Classpath !

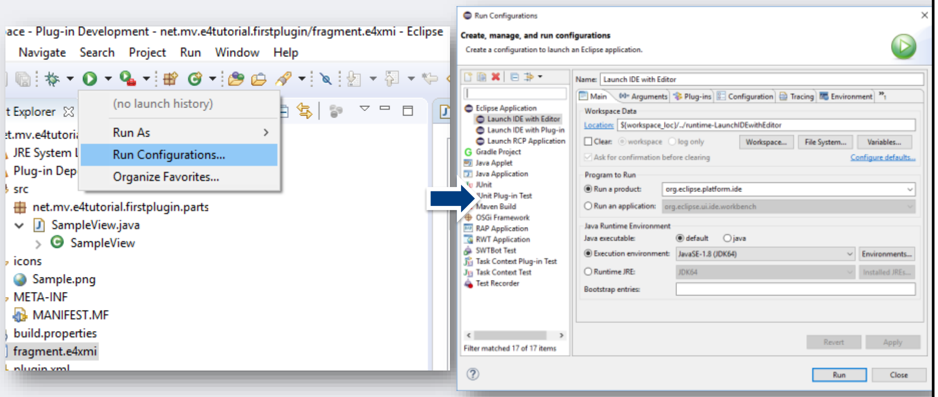


```
net.mv.e4tutorial.firstplugin
  JRE System Library [JavaSE-1.8]
  Plug-in Dependencies
  src
    net.mv.e4tutorial.firstplugin.parts
      SampleView.java
        SampleView
  icons
    Sample.png
  META-INF
    MANIFEST.MF
  build.properties
  fragment.e4xmi
  plugin.xml
```

```
package net.mv.e4tutorial.firstp

import javax.annotation.PostCons

public class SampleView {
    private Label myLabelInView;

    @PostConstruct
    public void createPartContro
        System.out.println("Ente

        myLabelInView = new Labe
        myLabelInView.setText("

    }

    @Focus
    public void setFocus() {
        myLabelInView.setFocus()

    }

    /**
     * This method is kept for 
     * mix E3 and E4 code. <br/
     * With E4 code you will se
     * you do not receive a ISel
     *
     * @param s
     *            the selection
     */
```

4 Run Plug-in

UNIVERSITY OF NOTRE DAME


ace - Plug-in Development - net.mv.e4tutorial.firstplugin/fragment.e4xmi - Eclipse

Run Configurations

Create, manage, and run configurations

Create a configuration to launch an Eclipse application.

Name: Launch IDE with Editor

Workspace Data

Location: ${workspace_loc}/../runtime-LaunchIDEwithEditor

Clear: workspace log only

Ask for confirmation before clearing

Program to Run

[✓ Run a product:] org.eclipse.platform.ide

Run an application: org.eclipse.ui.ide.workbench

Java Runtime Environment

Java executable: [✓ default] java

[✓ Execution environment:] JavaSE-1.8 (JDK64)

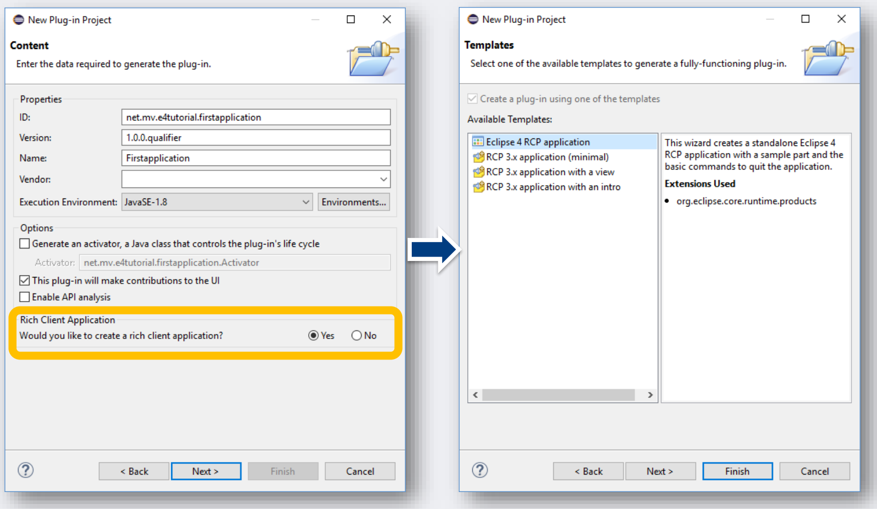
Runtime JRE: JDK64

Bootstrap entries:

Filter matched 17 of 17 items

5 Create your first RCP Application

UNIVERSITY OF NOTRE DAME


New Plug-in Project

Content

Enter the data required to generate the plug-in.

Properties

ID:	net.mv.e4tutorial.firstapplication
Version:	1.0.0.qualifier
Name:	Firstapplication
Vendor:	
Execution Environment:	JavaSE-1.8

Options

Generate an activator, a Java class that controls the plug-in's life cycle

Activator: net.mv.e4tutorial.firstapplication.Activator

[✓ This plug-in will make contributions to the UI]

Enable API analysis

Rich Client Application

Would you like to create a rich client application? [✓ Yes] No

New Plug-in Project

Templates

Select one of the available templates to generate a fully-functioning plug-in.

[✓ Create a plug-in using one of the templates]

Available Templates:

- Eclipse 4 RCP application
- RCP 3.x application (minimal)
- RCP 3.x application with a view
- RCP 3.x application with an intro

This wizard creates a standalone Eclipse 4 RCP application with a sample part and the basic commands to quit the application.

Extensions Used

- org.eclipse.core.runtime.products

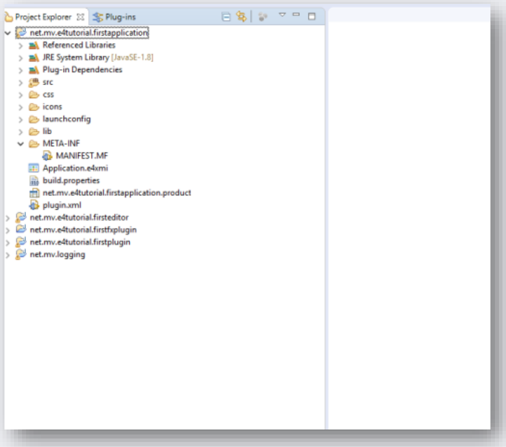
Important Artifacts

UNIVERSITY OF NOTRE DAME

- **Plug-in Manifest**
  - MANIFEST.MF
- **Plug-in Description**
  - plugin.xml
- **Workbench Model**
  - xy.e4xmi

Product File

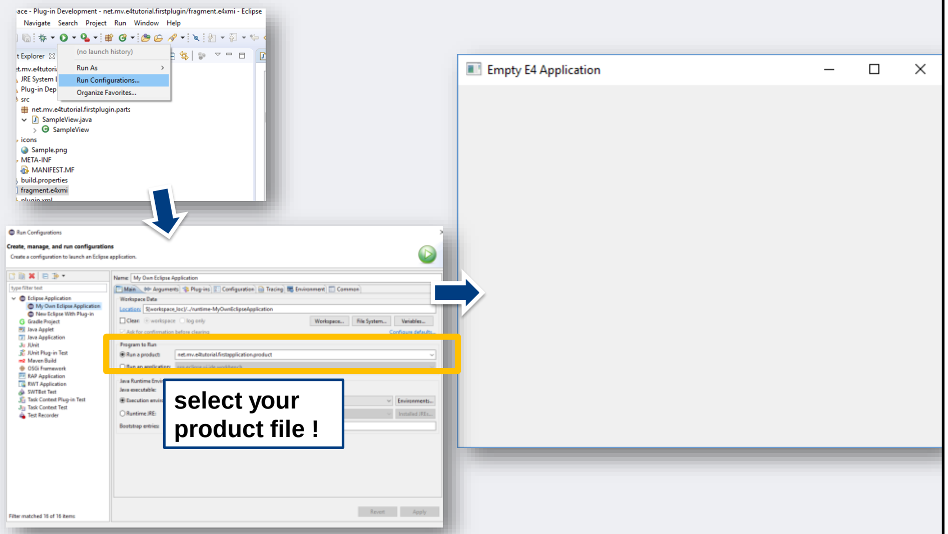
- xy.product



```
net.mv.e4tutorial.firstapplication
  Referenced Libraries
  JRE System Library [JavaSE-1.8]
  Plug-in Dependencies
  src
  css
  icons
  launchconfig
  lib
  META-INF
    MANIFEST.MF
  Application.e4xmi
  build.properties
  net.mv.e4tutorial.firstapplication.product
  plugin.xml
net.mv.e4tutorial.firsteditor
net.mv.e4tutorial.firstfxplugin
net.mv.e4tutorial.firstplugin
net.mv.logging
```

6 Run your Application

UNIVERSITY OF NOTRE DAME



Name: My Own Eclipse Application

Location: ${workspace_loc}/../runtime-MyOwnEclipseApplication

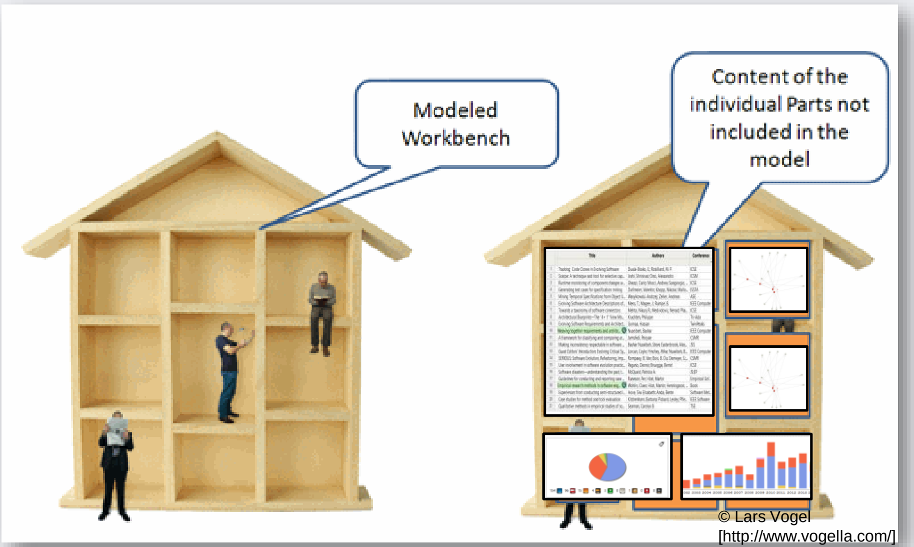
Program to Run

Run a product: net.mv.e4tutorial.firstapplication.product

select your product file !

Extending your Application

The Eclipse Workbench Model



© Lars Vogel [http://www.vogella.com/]

Views & Parts

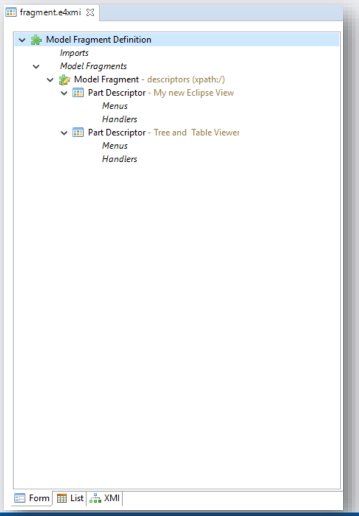
UNIVERSITY OF NOTRE DAME

- A **view** is a visual component within the Workbench.
  - Used to navigate a list or hierarchy of information
  - Modifications made in a view are saved immediately.
- An **editor** is also a visual component within the Workbench.
  - It is typically used to edit or browse a resource.
  - The visual presentation might be text or a diagram.
  - Typically, editors are launched by clicking on a resource in a view.
  - Modifications made in an editor follow an open-save-close lifecycle model.
- A **part** is a visual component in an RCP Application
  - Since Eclipse 4 basically everything is a part!

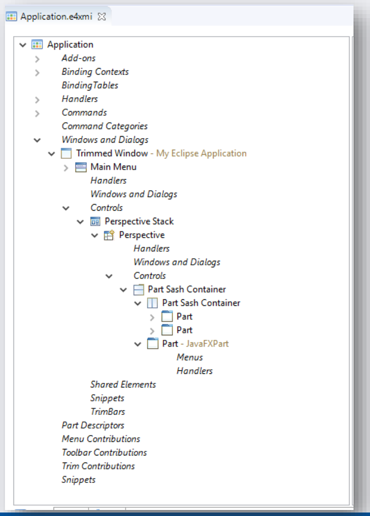
Views & Parts

UNIVERSITY OF NOTRE DAME

- **All of them are registered via the Workbench Model**



```
fragment.e4xmi
Model Fragment Definition
    Imports
    Model Fragments
        Model Fragment - descriptors (xpath:/)
            Part Descriptor - My new Eclipse View
                Menus
                Handlers
            Part Descriptor - Tree and  Table Viewer
                Menus
                Handlers
Form | List | XMI
```



```
Application.e4xmi
Application
    Add-ons
    Binding Contexts
    BindingTables
    Handlers
    Commands
    Command Categories
    Windows and Dialogs
        Trimmed Window - My Eclipse Application
            Main Menu
            Handlers
            Windows and Dialogs
            Controls
                Perspective Stack
                    Perspective
                        Handlers
                        Windows and Dialogs
                        Controls
                            Part Sash Container
                                Part Sash Container
                                    Part
                                    Part
                                Part - JavaFXPart
                                    Menus
                                    Handlers
            Shared Elements
            Snippets
            TrimBars
    Part Descriptors
    Menu Contributions
    Toolbar Contributions
    Trim Contributions
    Snippets
```

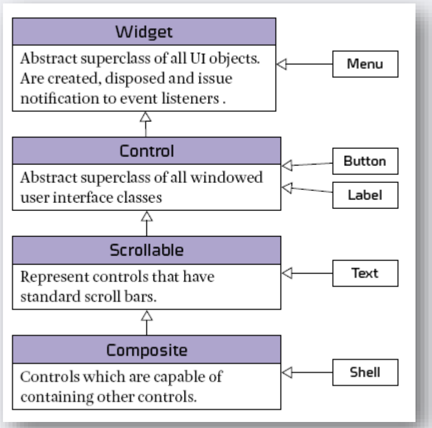
SWT – Standard Widget Toolkit

- SWT is an open source widget toolkit for Java
- Designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented.
- All standard SWT-widgets can be found at *org.eclipse.swt.widgets* and are derived from the base classes *Widget*, *Control*, *Scrollable* und *Composite*

[http://www.eclipse.org/swt]

SWT

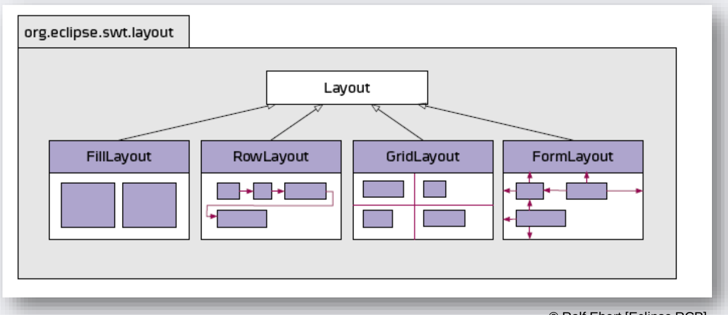
- http://www.eclipse.org/swt/snippets
- http://www.java2s.com/Code/Java/SWT-JFace-Eclipse/CatalogSWT-JFace-Eclipse.htm
- http://www.vogella.com/articles/SWT/article.html


Widget
Abstract superclass of all UI objects. Are created, disposed and issue notification to event listeners .
Menu
Control
Abstract superclass of all windowed user interface classes
Button
Label
Scrollable
Represent controls that have standard scroll bars.
Text
Composite
Controls which are capable of containing other controls.
Shell

© Ralf Ebert [Eclipse RCP]

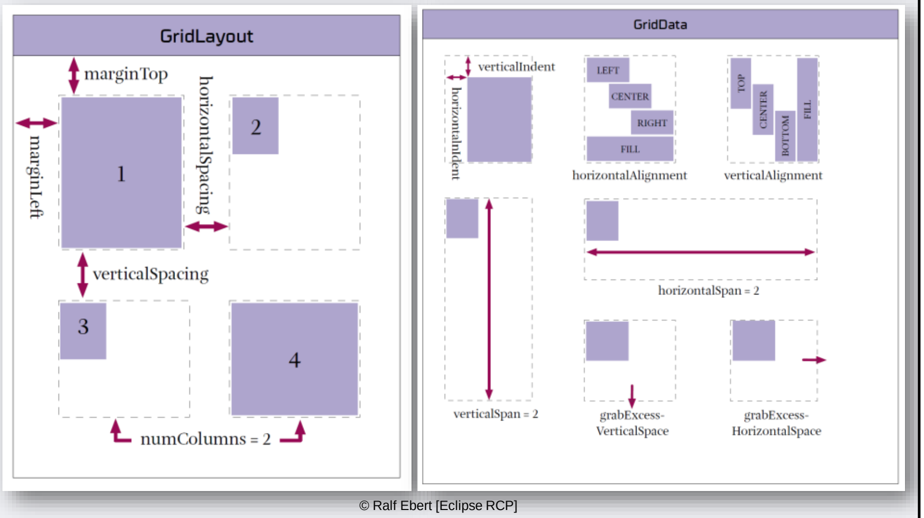
SWT - Layouts

- A layout manager is responsible for arranging the UI components of a container (Composite) on the screen.
- *SWT* offers several standard layout managers.



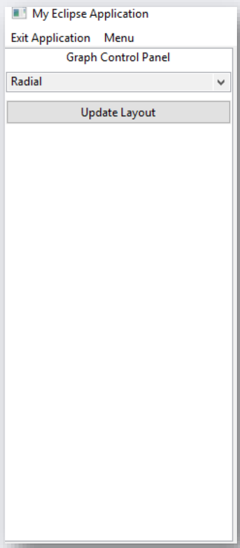
© Ralf Ebert [Eclipse RCP]

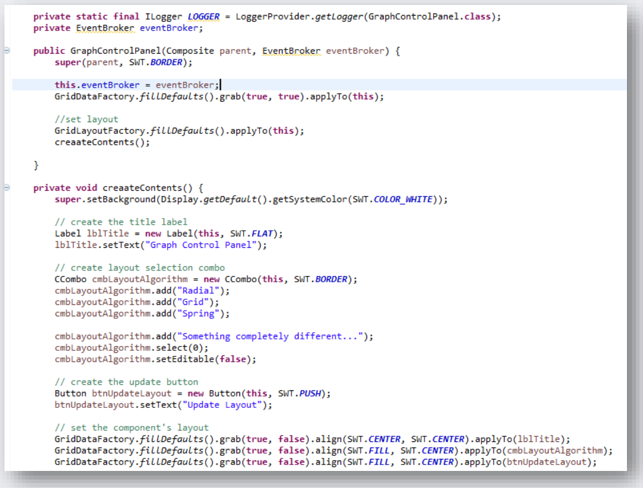
Grid Layout



© Ralf Ebert [Eclipse RCP]

Grid Layout Example





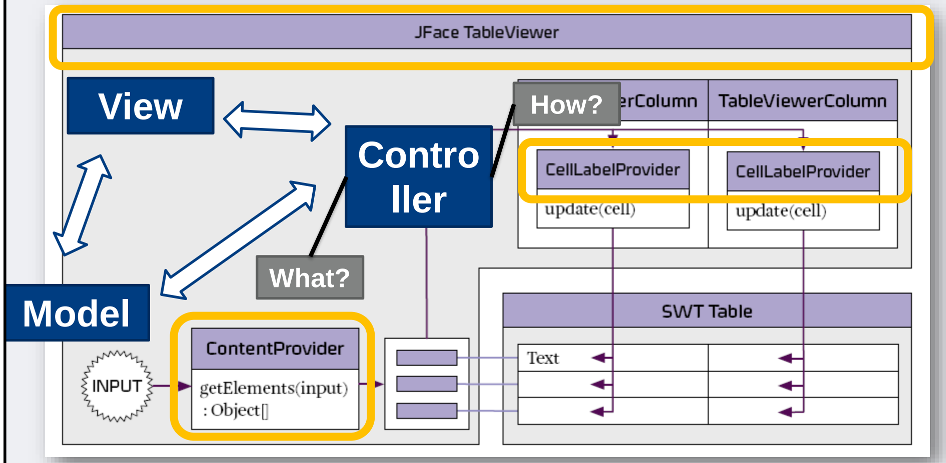
```java
    private static final ILogger LOGGER = LoggerProvider.getLogger(GraphControlPanel.class);
    private EventBroker eventBroker;

    public GraphControlPanel(Composite parent, EventBroker eventBroker) {
        super(parent, SWT.BORDER);

        this.eventBroker = eventBroker;
        GridDataFactory.fillDefaults().grab(true, true).applyTo(this);

        //set layout
        GridLayoutFactory.fillDefaults().applyTo(this);
        creaateContents();

    }

    private void creaateContents() {
        super.setBackground(Display.getDefault().getSystemColor(SWT.COLOR_WHITE));

        // create the title label
        Label lblTitle = new Label(this, SWT.FLAT);
        lblTitle.setText("Graph Control Panel");

        // create layout selection combo
        CCombo cmbLayoutAlgorithm = new CCombo(this, SWT.BORDER);
        cmbLayoutAlgorithm.add("Radial");
        cmbLayoutAlgorithm.add("Grid");
        cmbLayoutAlgorithm.add("Spring");

        cmbLayoutAlgorithm.add("Something completely different...");
        cmbLayoutAlgorithm.select(0);
        cmbLayoutAlgorithm.setEditable(false);

        // create the update button
        Button btnUpdateLayout = new Button(this, SWT.PUSH);
        btnUpdateLayout.setText("Update Layout");

        // set the component's layout
        GridDataFactory.fillDefaults().grab(true, false).align(SWT.CENTER, SWT.CENTER).applyTo(lblTitle);
        GridDataFactory.fillDefaults().grab(true, false).align(SWT.FILL, SWT.CENTER).applyTo(cmbLayoutAlgorithm);
        GridDataFactory.fillDefaults().grab(true, false).align(SWT.FILL, SWT.CENTER).applyTo(btnUpdateLayout);
```

JFace – Viewer / Table / Tree

- JFace is a set of APIs which builds on top of SWT
- Provides higher level abstraction APIs and commonly used functions
- JFace does not hide the SWT API but extends it. Therefore it is important to have a solid understanding of SWT, even if JFace is used

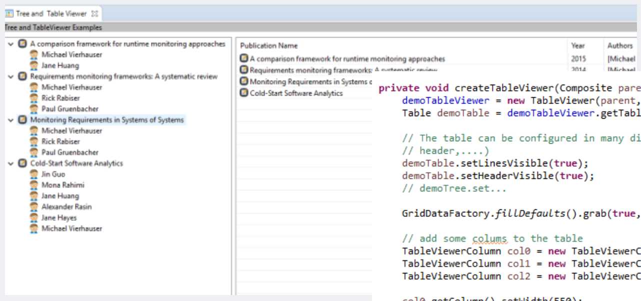
[https://wiki.eclipse.org/Jface]

JFace – Viewer / Table / Tree



© Ralf Ebert [Eclipse RCP]

Tree + Table Viewer Example


Tree and Table Viewer
Tree and TableViewer Examples

- A comparison framework for runtime monitoring approaches
  - Michael Vierhauser
  - Jane Huang
- Requirements monitoring frameworks: A systematic review
  - Michael Vierhauser
  - Rick Rabiser
  - Paul Gruenbacher
- Monitoring Requirements in Systems of Systems
  - Michael Vierhauser
  - Rick Rabiser
  - Paul Gruenbacher
- Cold-Start Software Analytics
  - Jin Guo
  - Mona Rahimi
  - Jane Huang
  - Alexander Rasin
  - Jane Hayes
  - Michael Vierhauser

Publication Name	Year	Authors
A comparison framework for runtime monitoring approaches	2015	[Michael
Requirements monitoring frameworks: A systematic review	2014	[Michael
Monitoring Requirements in Systems c		
Cold-Start Software Analytics		

```java
private void createTableViewer(Composite parent) {
    demoTableViewer = new TableViewer(parent, SWT.FULL_SELECTION | SWT.BORDER);
    Table demoTable = demoTableViewer.getTable();

    // The table can be configured in many different ways (color,
    // header,....)
    demoTable.setLinesVisible(true);
    demoTable.setHeaderVisible(true);
    // demoTree.set...

    GridDataFactory.fillDefaults().grab(true, true).applyTo(demoTableViewer.getTable());

    // add some colums to the table
    TableViewerColumn col0 = new TableViewerColumn(demoTableViewer, SWT.FLAT);
    TableViewerColumn col1 = new TableViewerColumn(demoTableViewer, SWT.FLAT);
    TableViewerColumn col2 = new TableViewerColumn(demoTableViewer, SWT.FLAT);

    col0.getColumn().setWidth(550);
    col1.getColumn().setWidth(60);
    col2.getColumn().setWidth(400);

    col0.getColumn().setText("Publication Name");
    col1.getColumn().setText("Year");
    col2.getColumn().setText("Authors");

    // 1 - each JFace Viewer requires a content provider responsible for
    // WHAT to show in the viewer
    demoTableViewer.setContentProvider(new DemoTreeViewerContentProvider());

    // 2 - each JFace Viewer requires a label provider responsible for HOW
    // to present thins in the viewer
    demoTableViewer.setLabelProvider(new DemoTableViewerLabelProvider());

    // Finally - to set some elements to the viewer - the viewer requires
    // some input!
    demoTableViewer.setInput(MyTreeTableDemoModel.getInstance());

}
```

JFace - Viewer/Table/Tree

UNIVERSITY OF NOTRE DAME

- **Viewer connect widgets to a model for the more complex SWT widgets like tables and trees**
- **One method for structure, labels, sorting and filtering**
- **Simple to update when domain objects are changed efficient handling included in viewer**

Yet another Design Pattern

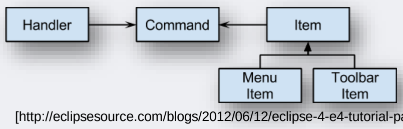
UNIVERSITY OF NOTRE DAME

Command Pattern

- **Decoupling producer from consumer**
  allows allows the requester of a particular action
  to be decoupled from the object that performs the action.

  Command declares an interface for all commands, providing a simple execute() method
  In Eclipse commands are defined in the workbench model with a unique id

  Handler is executed, performing the actual action.
  In Eclipse handler are defined in the workbench model and linked to a command



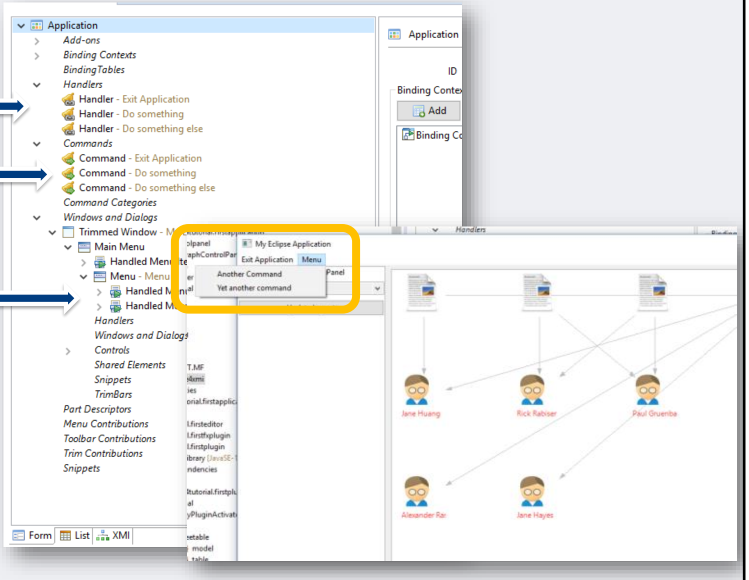
[http://eclipsesource.com/blogs/2012/06/12/eclipse-4-e4-tutorial-part-2]

Commands & Handler

Handler

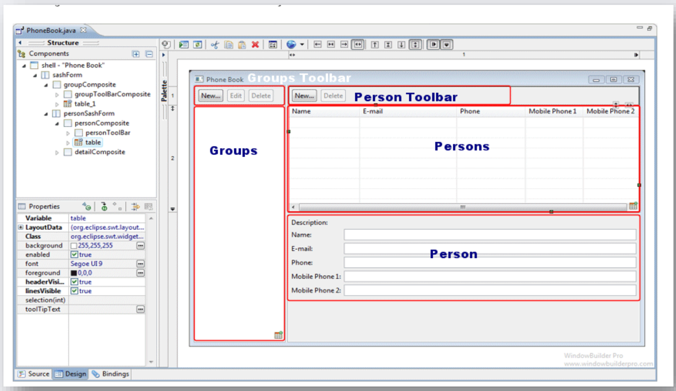
Commands

Item



Google Window Builder

http://www.eclipse.org/windowbuilder/download.php



Eclipse Tweaks & Tricks

UNIVERSITY OF NOTRE DAME

- Important Shortcuts

`Ctrl+Shift+R`	Search dialog for resources
`Ctrl+Shift+T`	Search dialog for Java Types
`Ctrl+F11`	Run last Launch Config
`Ctrl+Space`	Content Assist / Auto Complete
`Ctrl+Shift+F`	Format source code
`Ctrl+Shift+O`	Organize imports (add/remove/update)
`Ctrl+3`	Magic Key! – Search for any Component within Eclipse

- **Define your Cleanup / AutoFormat Rules**
  - **Preferences -> Java -> Code Style -> Cleanup**
  - **Preferences -> Java -> Code Style -> Formatter**

- **Define your Compiler Settings**
  - **Preferences -> Java -> Compiler -> Erros / Warnings**

More Eclipse Concepts…

UNIVERSITY OF NOTRE DAME

Images

- Image Descriptor
  - `ImageDescriptor colImage = ImageDescriptor.createFromURL(FileLocator.find(Activator.getDefault().getBundle(), new Path("icons\\viewIcon.png"),null));`

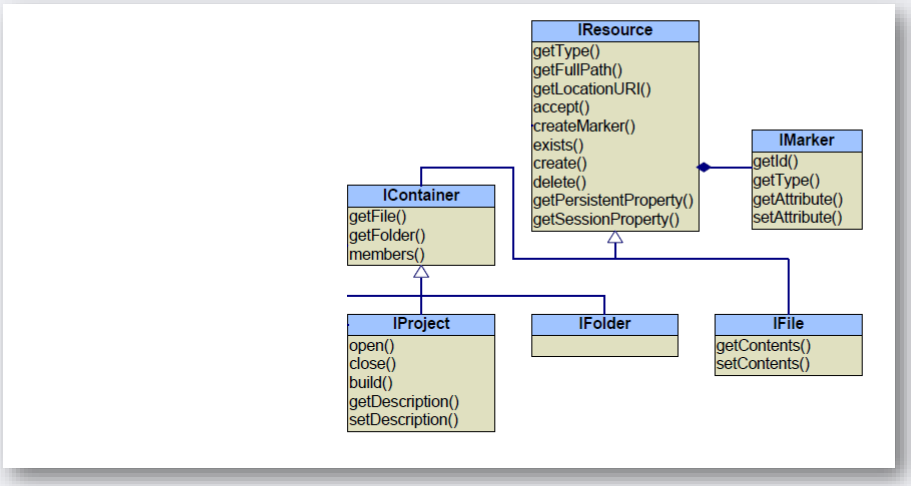
  - http://www.eclipse.org/articles/Article-Using%20Images%20In%20Eclipse/Using%20Images%20In%20Eclipse.html

- Eclipse Shared Images
  - http://shinych.blogspot.co.at/2007/05/eclipse-shared-images.html

Eclipse Editors

- Eclipse uses editors and views to maintain data.
- An editor is a workbench part that allows a user to edit an object (often a file).
- Editors operate in a manner similar to file system editing tools, except that they are tightly integrated into the platform workbench UI. An editor is always associated with an input object (IEditorInput).
- You can think of the input object as the document or file that is being edited. Changes made in an editor are not committed until the user saves them.
- An editor typically requires that the user explicitly select "save" to apply the changes to the data while a view typically changes the data immediately.

Resources



SWT –Threading Issues

- Each display is bound to a thread
  - Called the "user interface thread"
    - This thread executes the main event loop dispatch of operating system events
- SWT is not thread safe
  - Resource objects must only be accessed by user interface thread
    - "SWTException"when method called from wrong thread
    - Better than an unexpected behavior
- Execute code from a non UI thread
  - "display.syncExec(runnable)"or "display.asyncExec(runnable)"
  - The run method of the "Runnable"is executed in the UI thread

Views/Perspectives/Editors

- A **view** is a visual component within the Workbench.
  - used to navigate a list or hierarchy of information
  - Modifications made in a view are saved immediately.
- A **perspective** is a group of views and editors in the Workbench window.
  - One or more perspectives can exist in a single Workbench window.
  - Each perspective contains one or more views and editors.
- An **editor** is also a visual component within the Workbench.
  - It is typically used to edit or browse a resource.
  - The visual presentation might be text or a diagram.
  - Typically, editors are launched by clicking on a resource in a view.
  - Modifications made in an editor follow an open-save-close lifecycle model.

Feature

A Feature is used to package a group of plug-ins together into a single installable and updatable unit.

- Feature = installable package of multiple plug-ins
- Install and update mechanism depends on features
- Contains license and copyright information

  feature.xml describes the plug-ins that are contained
- Does not contain Java classes