Eclipse RCP and Plug-in development

Gyaltso Technologies

www.gyaltso.com | info@gyaltso.com

About Me

- Neeraj Bhusare
- 12+ years of experience in product development using Java and Eclipse platform
- Founder of Gyaltso Technologies
- Trained 200+ students on various Eclipse technologies
- Clients - Nokia, Robert Bosch, Tata Consultancy Services (TCS), Tata Elxsi, KPIT, Cleo, GoldenSource, Atos Syntel, Bureau Veritas, IBM, AVIN Systems, Sapient, Majesco – Mastek, Aricent, Ancit Consulting.

Training Objective

- Eclipse IDE
- Eclipse Platform and its building blocks
- Standard Widget toolkit (SWT)
- JFace UI framework
- SWT-AWT Bridge.

Acronyms

- IDE - Integrated Development Environment
- RCP - Rich Client Platform
- SWT - Standard Widget Toolkit
- JFace - UI framework based on SWT
- Plug-in - A functional unit that integrated with the Eclipse IDE
- OSGi - Open Service Gateway Initiative.

Introduction

- Background
- Eclipse IDE
- Platform Architecture
- OSGi

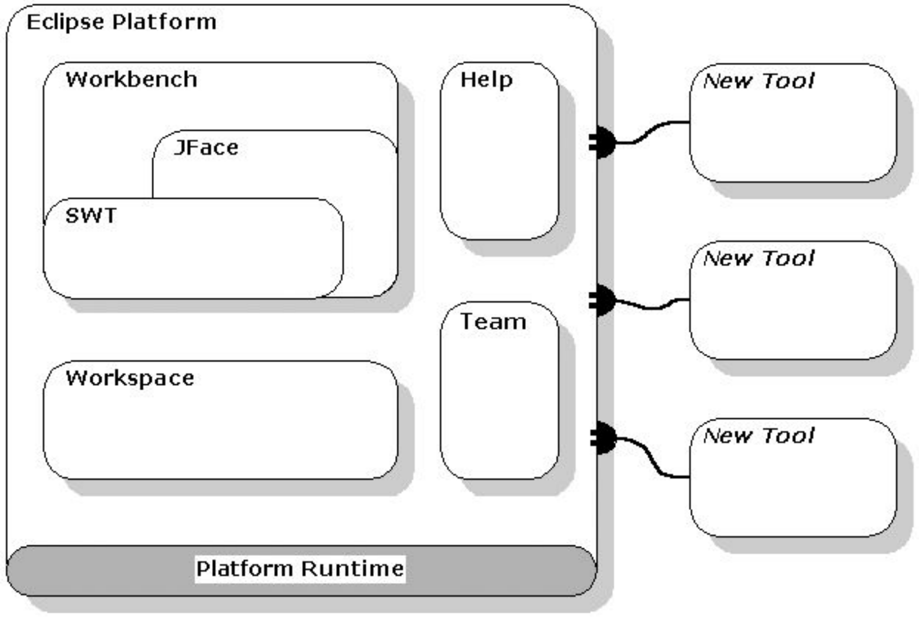
Introduction

- Eclipse is an IDE and a Platform for building RCP applications and Plug-ins
- License - Eclipse Public License (EPL)
- Latest release - Eclipse IDE 2020-03.

Eclipse IDE

- Eclipse Workbench
- Main menu bar, View menu, Context menu
- Main toolbar, View toolbar
- Perspective
- Preferences
- Views
- Editors and Editor area
- Help
- Workspace and the Eclipse file system (EFS)
- Versioning.

Platform architecture



OSGi

- Open Service Gateway Initiative
- OSGi is a specification for building “Modular” applications in Java
- Eclipse Plug-in’s or Modules are deployed into the OSGi runtime
- Implementations
    - Equinox (reference implementation)
    - Apache Felix
    - Knopflerfish.

Hello World RCP and Plug-in

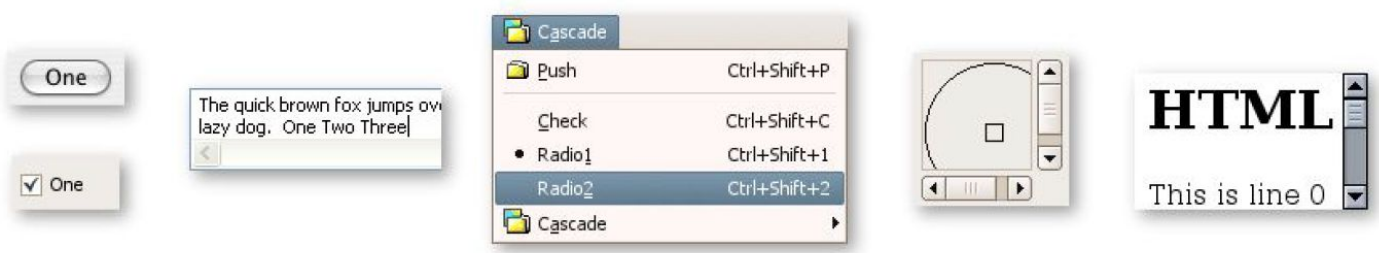
- Hello World RCP application
- Hello World Plug-in
- Launching the application
- Project walkthrough

SWT

- Introduction
- SWT Widgets
- SWT Hello World
- SWT Layout and Layout Data
- SWT Data binding

Introduction

- SWT - Standard Widget Toolkit
- It is an open-source widget toolkit for Java which is portable across multiple operating systems
- The library provides widgets like Canvas, Shell, Button, Menu, Label, Text, Browser, to name a few.



The quick brown fox jumps ov
lazy dog. One Two Three

SWT Widgets

- Standard
  - Button, Menu, Colbo, Dialog, Canvas...etc
  - Provided by the **org.eclipse.swt.widgets** project
- Custom
  - CBanner, CCombo, CLabel, CTabFolder..etc
  - Provided by the **org.eclipse.swt.custom** project

SWT Hello World

- Shell - Container widget
- Display
    - A bridge between the SWT widget and the operating system resources.
    - It provides methods for accessing the OS resources
    - Provides inter thread communication methods
- Display#readAndDispatch()
    - It reads and dispatches the events from the operating systems event queue.

SWT Layout

- Fill Layout - Positions the children in a **single row** or a **column**
- Row Layout - Positions the children in a **horizontal row** or a **vertical column**
- Grid Layout - Places the children in a **grid**
- Stack Layout - Stacks the children on top of other and resizes all the controls to have the same size.

Exercise

1. Create SWT Hello World example
2. Enhance the Hello World example by adding Fill Layout
3. Enhance the Hello World example by adding Row Layout
4. Enhance the Hello World example by adding Grid Layout.

SWT Layout Data

- Row Data
  - It is used with the **Row layout**
  - It is used for configuring the **height** and the **width** of the widgets
- Grid Data
  - It is used with the **Grid layout**
  - It is used for configuring the attributes **height**, **width**, **alignment**, **span**, to name a few.

SWT Layout factory

- RowLayoutFactory
  GridLayoutFactory.fillDefaults().numColumns(2).equalWidth(false).applyTo(theComponent);

  GridDataFactory.fillDefaults().grab(true, false).align(SWT.FILL, SWT.BEGINNING).hint(200, 200)
  .applyTo(button);

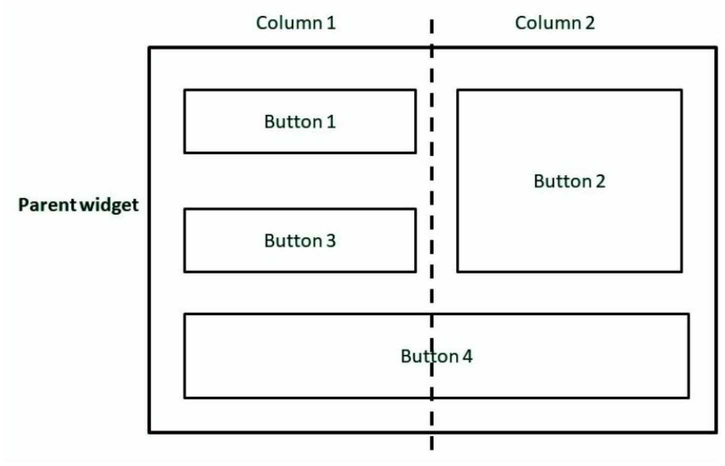
- GridLayoutFactory
  RowLayoutFactory.fillDefaults().type(SWT.VERTICAL).applyTo(theComponent);

  RowDataFactory.swtDefaults().hint(200, 200).applyTo(button);

Exercise 01

1. Enhance the Row Layout example by adding Row Data
2. Enhance the Row Layout example by adding RowDataFactory
3. Enhance the Grid Layout example by adding GridData
4. Enhance the Grid Layout example by adding GridDataFactory.

Exercise 02


Column 1
Column 2
Parent widget
Button 1
Button 2
Button 3
Button 4

JFace

- Views and Editors
- JFace Viewers
- Platform Command and Expression framework
- Jobs framework
- Wizards
- Dialogs
- Perspective
- Preferences

Views and Editors

Introduction - View part

- The View class typically implements from the `IViewPart` interface or extends from the `ViewPart` class
- The extension point **org.eclipse.ui.views** is used for contributing Editors to the Eclipse workbench
- An View class should implement the following methods
  - `IViewPart.init`
  - `IViewPart.createPartControl`

JFace Viewers

- **ListViewer**: Used for displaying items in a List
  - Created using `new ListViewer(shell)` constructor
  - Content provider implements the `IStructuredContentProvider`, and Label provided extends the `LabelProvider` class
- **TreeViewer**: Used for displaying items in a Tree
  - Created using `new TreeViewer(shell)` constructor
  - Content provider implements the `ITreeContentProvider`, and Label provided extends the `LabelProvider` class
- **TableViewer**: Used for displaying items in a Table
  - Created using `new TableViewer(shell)` constructor
  - Content provider implements the `IStructuredContentProvider`, and Label provided extends the `LabelProvider` class

Exercise

1. Create an Empty View
2. Use the ListViewer to display items in a list format
3. Use the TreeViewer to display items in a tree format
4. Use the TableViewer to display items in a table format
5. Enhance the above example to show multiple columns in a TableViewer.

Introduction - Editor part

- The Editor class typically implements from the **`IEditorPart`** interface or extends from the **`EditorPart`** class
- The extension point **org.eclipse.ui.editors** is used for contributing Editors to the Eclipse workbench
- An Editor class should implement the following methods
    - **`IEditorPart.init`**
    - **`IWorkbenchPart.createPartControl`**
    - **`IWorkbenchPart.setFocus`**
    - **`IEditorPart.isDirty`**
    - **`IEditorPart.doSave`**
    - **`IEditorPart.doSaveAs`**
    - **`IEditorPart.isSaveAsAllowed`**

Exercise

1. Create a Simple Editor
2. Enhance the editor to detect specific text in the input, and change the state of the editor to dirty
3. Enhance the editor to enable saving of the input text.

Platform Command framework

Introduction

- The Platform Command Framework (PCF) is used for managing the action contributions to the Eclipse workbench
- A **Command** provides the declaration and the description of an action
- The **Handler** performs the actual work
- Multiple handlers can be associated with a single command; however, at the max, one handler is active at a given point, else the command is disabled.

Command contribution using extension points

- The Command, Handler, and its location in the Eclipse UI is defined using extension points provided by the **org.eclipse.ui** plug-in
- A Command is defined using the **org.eclipse.ui.commands** extension point
- A Handler is defined using the **org.eclipse.ui.handlers** extension point
- The location of a command is specified using the **locationURI** field in the **org.eclipse.ui.menus** extension point.

Exercise

- Command with handler not specified
- Command with a Default handler specified
- Command with a Specific handler
- Placing command in the following UI locations
  - Main toolbar and Menubar
  - Popup menu of the Package explorer
  - View menu of the Package explorer
  - View toolbar of the Package explorer
  - Before the New menu item in the right click context menu
  - At the end of the New menu group item in the right click context menu
  - Before the Open group in the right click context menu.

Platform Expression framework

Introduction

- Core expressions defined in the plugin.xml file are used for controlling the state (enabled, active, visibility) of a Handler
- A handler can be enabled using the **enableWhen** and activated using the **activeWhen** expressions in the handler contribution. In the menus contribution, **visibleWhen** controls the visibility
- The evaluation of expression happens lazily at runtime
- The expressions can be reused using the **org.eclipse.core.expressions.definitions** extension point.

Exercise

- Command with one default and multiple specific handlers
- Command with a specific handler that is activated when the selection count is equal to 1
- Command with a specific handler that is activated when the selection count is more than 1, and all the selected objects are of type IFile, with .txt file extension.

Jobs framework

Introduction

- The Jobs framework provides API for running tasks in the background, and provide visual feedback of the same
- The `Job` interface represents the unit of runnable work. It can be in the following states - **RUNNING**, **SLEEPING**, **WAITING**, and **NONE**. It can have the following priorities - **INTERACTIVE**, **SHORT**, **LONG**, **BUILD**, and **DECORATE**, where LONG is the default job priority
- A Job Manager (`IJobManager`) performs scheduling of the jobs. It maintains a queue to which a new job gets added on the execution of `Job#schedule()` API
- `IProgressMonitor` is the interface for showing progress feedback of a given job to the user.

Exercise

1. A simple Hello World Job
2. A job with Progress feedback, and handling job cancellation
3. Splitting a given job into sub-tasks
4. Grouping multiple jobs using Progress groups
5. Creating a System job
6. Creating a User job
7. Rescheduling jobs
8. Setting job priorities
9. Scheduling the jobs to run after some delay
10. Using Job#join() API

JFace Wizards

JFace Preferences

Resource pointers

- Gyaltso Github repo - https://github.com/gyaltso
- Training slides - https://github.com/gyaltso/Eclipse-Training-Slides
- Training Example source code - https://github.com/gyaltso/Eclipse-Training-Sources
- Training schedules - https://github.com/gyaltso/Eclipse-Training-Schedules

Thank you

Gyaltso Technologies
www.gyaltso.com | info@gyaltso.com