# Eclipse RCP and Plug-in development

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

# Acronyms

- RCP - Rich Client Platform
- PCF - Platform Command Framework
- PEF - Platform Expression Framework.

# JFace

- Platform Command framework
- Platform Expression framework
- Jobs framework

# 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
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