Eclipse course day-wise schedule

Eclipse RCP and Plug-in development - 4 days

Day 01 - 4 Hours

1. Introduction - Eclipse Platform and its building blocks
2. Eclipse Workbench walkthrough
3. Eclipse Standard Widget Toolkit (SWT)
    a. Introduction
    b. Basic SWT application
    c. Controls - basic and advance
    d. Layout and Layout managers
        i. Fill layout
        ii. Row layout and Row layout manager
        iii. Grid layout and Grid layout manager
        iv. Stack layout
    e. Layout manager factories
    f. Exercise - SWT Layout and Layout managers
    g. Event handling in the SWT widgets

Day 01 - 4 Hours

1. Eclipse JFace UI framework
    a. Introduction
    b. Viewers
        i. Tree, Table, and List
        ii. Content and Label providers
    c. Views
        i. IViewPart introduction
        ii. Creating a plug-in that contributes a view using an extension point
        iii. Specifying view categories
        iv. Embedding viewers (tree/table/list) inside the custom view
    d. Editor
        i. IEditorPart introduction
        ii. Creating a plug-in that contributes an editor using an extension point
        iii. Strategy for handling dirty state of the editor
        iv. Embedding viewers (tree/table/list) inside the custom editor

Day 02 - 8 Hours

1. Dialogs
    a. Making use of out-of-the-box provided Dialogs
    b. Creating custom dialogs
    c. Adding controls to a custom dialog
    d. Launching a dialog
2. Wizard
    a. Wizard, Wizard pages, and Wizard Dialog
    b. Creating a custom wizard
    c. The new resource creation wizard
    d. Import and Export wizards
    e. Making use of existing wizards
    f. Launching a wizard
3. Perspective
    a. Organizing views using Perspective
    b. Layouting viewers using predefined Layouts
    c. Adding Viewers to an existing Perspective by extending it
    d. Adding existing Eclipse views to our custom perspective
4. Preferences
    a. Creating custom Preferences
    b. Reading and storing values in Preferences

Day 03 - 8 Hours

1. Command framework
    a. Defining Commands using the “commands” extension point
    b. Contributing Command to Eclipse UI using “menus” extension point
    c. Command with no handler
    d. Command with a default handler
    e. Defining a specific Handler using the “handlers” extension point
    f. Associating multiple handlers to a single command
    g. A quick look at the Plug-in spy
    h. Placing commands in a different location in the Eclipse workbench using the “menus” extension point
    i. Programmatic execution of handlers
    j. Passing parameters to commands
    k. Programmatic definition of commands
2. Expression framework
    a. Understanding the use of expression framework
    b. Activation and deactivation of command handlers
    c. Controlling the visibility of commands using an expression

   d. Making use of some the following operations - adapt, and/or/not, count, equals, instanceof, iterate, reference...etc.
   e. Reusing expressions using the “definitions” extension point.
3. Jobs framework
   a. Using Jobs framework and its use
   b. Job API, States, and Priorities
   c. Hello world Job
   d. A job with progress feedback
   e. Handling Job cancellation
   f. Splitting job into sub-tasks
   g. Grouping multiple jobs using progress subgroups
   h. System jobs
   i. User jobs
   j. Rescheduling jobs
   k. Setting job priorities
   l. Job scheduling rules
   m. Scheduling jobs after a delay
   n. Using Job#jon() API

Day 04 - 8 Hours

1. Testing SWT applications using SWTBot
2. Branding and packaging applications
   a. Defining plugin Features
   b. Creating an update site
   c. Creating a Target platform
   d. Product configuration
   e. Creating a custom IDE (Optional depending upon the time we have)

Eclipse EMF - 2 days

Day 05 - 4 Hours

1. Understanding Modeling Driven Development and its benefits
2. Introduction to the Eclipse Modeling Framework (EMF)
3. EMF workflow
    a. Meta-Modeling - Creating a custom modeling using ECore
    b. Generating Runtime and UI projects
    c. Creating a model using the Reflective ECore editor
4. Code Generation
    a. Understanding the generated Factory, Package, AdapterFactory and Switch classes and their use
    b. Customizing the Generated code and retaining the same
5. Customizing the .genmodel by changing values of the following attributes - Base Package, Prefix, Initialize by loading, Literals interfaces...etc
6. Ecore Meta-modeling (**PS - Here we extend the model created above**)
    a. Adding EClasses to the model
    b. Understand EObject
    c. Adding model attributes - Single and Multi-valued
    d. Adding References - Non-Containment, Containment, Bidirectional, and Map
    e. Adding DataTypes
    f. Adding Operations
    g. Adding Annotations to the model

Day 05 - 4 Hours

1. Runtime Framework
    a. Notification and adapters
        i. Understanding EMF notification mechanism
        ii. Observing the model changes by attaching notification adapters
        iii. EContentAdapter - Definition and its use
        iv. Adapter dos and don’ts
    h. Persistence framework
        i. Persistence API - ResourceSet, Resource, and URI
        ii. EMF Package Registry
            1. Registring packages with the Local EMF Registry
            2. Registring packages with the Global EMF Registry
            3. Reding from the registry
            4. When to use the local and the global registries
            5. Automatic Vs manual EPackage registration
            6. EPackage registration usage dos and don’ts
    i. EMF Proxy resolution

i. Proxies in EMF
ii. ECoreUtil#resolveAll() API
iii. Influencing Proxy resolution
    1. Resolve Proxies property
    2. Containment Proxies property

j. Dynamic EMF
    i. Creating the above-created custom ECore model programmatically using the EMF Dynamic API
    ii. When to use EMF Dynamic API

Day 06 - 8 Hours

1. EMF Compare framework
    a. Understanding the EMF compare framework architecture
    b. Extending the framework to compare two versions of the above-created ECore model
        i. Defining a ResourceSet comparer
        ii. Defining a custom matching strategy
        iii. Defining a custom Diff strategy
        iv. Filtering models features from the comparison mechanism
        v. Adapting the result Compare model to fit our needs.
2. EMF Validation framework
    a. Introduction to the framework
    b. Defining constraints and invariants
    c. Invoking the validation

Eclipse Xtext - 3 days

Day 07 - 8 Hours

1. Textual modeling using the Xtext framework
2. Examples of Domain-Specific Languages (DSL's) bases on Xtext
3. Xtext infrastructure
    a. Creating the Xtext project using the project creation wizard
    b. Understanding the project layout - Runtime, UI, and Test
    c. .xtext and .MWE2 files
    d. Code Generation
    e. Understanding the generated code
    f. Customizing the code generator by adapting the .MWE2 file
4. Grammar definition
    a. Common Terminal grammar
    b. Defining a custom DSL for the financial domain
    c. Understanding EPackage declaration, Overriding grammar fuels, Enum definition, Terminal rules, Terminal fragment rules...etc
    d. Hidden tokes
    e. Overriding Parser and Terminal rules
    f. Grammar definition Dos and Don'ts
5. Grammar reuse and Mixins
    a. Reusing rules defining in a separate grammar
    b. Adapting the .MWE2 file

Day 08 - 8 Hours

1. Runtime customization
    a. Adding validation for our financial DSL
        i. Custom validation
        ii. Customizing the LinkingDiagnosticMessageProvider
    b. Understanding the Scope provider, customizing the same
        i. Local Scope provider
        ii. Global Scope provider
    c. Customizing the Formatter
    d. Providing a library.
2. UI customization
    a. Customizing the Proposal provider
    b. Customizing the Quick-fix provider.
3. Xtext index
    a. Understanding the Xtext index mechanism
    b. Defining a custom index specific to our language

    c. Defining a ResourceDescriptionStrategy for controlling what goes into the Xtext index
4. Qualified name providers
5. Testing Xtext DSL infrastructure components.

Day 09 - 8 Hours

1. Defining a Generator for our language
    a. Here we generate a representation of our models in a text file.
2. Xbase - Adding expression support for our language
    a. Enhancing our language to have expression support
    b. Using Xbase compiler/interpreter to execute the language at runtime
3. Code generation
    a. Understand Xtend syntax
    b. Defining a code generator for our language

Eclipse Sirius - 1 day

Day 10 - 8 Hours

1. Sirius overview
    a. Graphical modeling using Sirius framework
    b. Architecture overview
        i. Viewpoint and viewpoint registry
        ii. Session
        iii. Transaction editing domain
        iv. Dialects
    c. Example applications built using Sirius
2. Viewpoint specification model (VSM)
    a. Introduction
    b. Specifying viewpoints
        i. Container, Node, and Connections mappings
        ii. Specifying tools
        iii. Specifying Model operations
        iv. Specifying Filters
        v. Specifying Validation
3. Xtext Sirius integration
    a. Integrating our Xtext language with Sirius.