Customizing and Extending Eclipse

Duration:	2 days
Type:	advanced

Description

Eclipse is now the dominant IDE within the Java community. However most developers using Eclipse are unaware that it is much more than a development tool.

Eclipse was designed from the beginning as a contribution based project where everything except for a small kernel would be a plug-in. Developers can add their own plug-ins, ranging from single buttons and menu items to complete wizards and power tools. Additionally Eclipse can be used as a framework for building rich client applications for any purpose.

This course provides complete coverage of the different ways in which Eclipse can be extended. Special consideration is given to writing plug-ins that contribute extra refactorings and code generation functionality.

Prerequisites

Delegates should have several years Java programming experience, understand XML and be familiar with developing Java applications using the Eclipse Java IDE. Previous experience of installing Eclipse plug-ins is helpful but not essential.

List of Modules

Introducing Eclipse from the Inside Out

- The evolution of Eclipse from Smalltalk
- SWT and JFace as an alternative to Swing
- The contribution based architecture of Eclipse
- The Eclipse workbench and perspectives
- Editors, actions and views within a perspective
- An overview of plug-in development
- An overview of rich client development

The Standard Widget Toolkit

- Differences between SWT and Swing/AWT
- Commercial and open source SWT tools
- Standard SWT widgets and Layout Managers
- Listeners and the event handling model
- Mapping data to widgets via viewers
- Adding support for drag and drop

Creating Plug-Ins

- The structure of a basic plug-in
- Adding a plug-in class with lifecycle methods
- Building and running a plug-in
- The lifecycle of plug-ins within Eclipse
- Commonly used extension points
- Defining new actions and views
- Base types used to create extensions
- Writing a plug-in with multiple extensions
- Making your plug-in extensible
- Debugging and unit testing plug-ins

Creating Refactoring Plug-Ins

- How Eclipse represents code within the Java IDE
- Obtaining a working copy of the code being edited
- Reconciling differences and merging compilation units
- Changing code via the modifier methods of *IType*
- Changing code by tokenizing the compilation unit
- Parsing the compilation unit into an Abstract Syntax Tree
- Changing code by altering the Abstract Syntax Tree
- Altering the Abstract Syntax Tree via the Visitor Pattern

The Rich Client Platform (RCP)

- Designing a new editor for Eclipse
- Creating dialog boxes and wizards
- Contributing new perspectives
- Adding properties and help pages
- Packaging an RCP application
- Deploying applications via WebStart