zeligsoft

The Eclipse Project

An introduction to the Eclipse Project

zeligsoft

Overview

In this module we will explore the Eclipse Project including the different aspects of the Eclipse Workbench.

We will also explore how to develop and deploy plug-ins to extend the Eclipse Workbench.

Some content taken from
Developing Plug-ins for Eclipse (http://www.eclipse.org/resources/)
by Dwight Deugo and Nesa Matic

zeligsoft

Goals

- Following the completion of this module and its exercises you will
  - Have an understanding of the Eclipse Workbench
  - Know some of the terminology used within the Eclipse Workbench
  - Have an understanding of the Eclipse architecture
  - Have an understanding of how Eclipse is extended
  - Be able to develop and deploy a basic plug-in

zeligsoft

Agenda

- What is Eclipse?
- The Eclipse Workbench
- The Eclipse Architecture
- Extending the Workbench

zeligsoft

What is Eclipse?

- Eclipse is an open source project
  - http://www.eclipse.org
  - Consortium of companies, including IBM
  - Launched in November, 2001
  - Designed to help developers with specific development tasks

zeligsoft

Projects

- On topic
  - Business Intelligence and Reporting Tools (BIRT)
  - Device Software Development Platform
  - Eclipse Project
  - Eclipse Modeling Project
- Off topic
  - Data Tools Platform
  - Eclipse RT
  - SOA Tools
  - Eclipse Technology Project
  - Tools Project
  - Test and Performance Tools Platform Project
  - Eclipse Web Tools Platform Project

http://www.eclipse.org/projects/listofprojects.php

zeligsoft

Brief History of Eclipse

- 1994
  - VisualAge for Smalltalk
- 1996
  - VisualAge for Java
- 1996-2001
  - VisualAge Micro Edition
- 2001
  - Eclipse Project

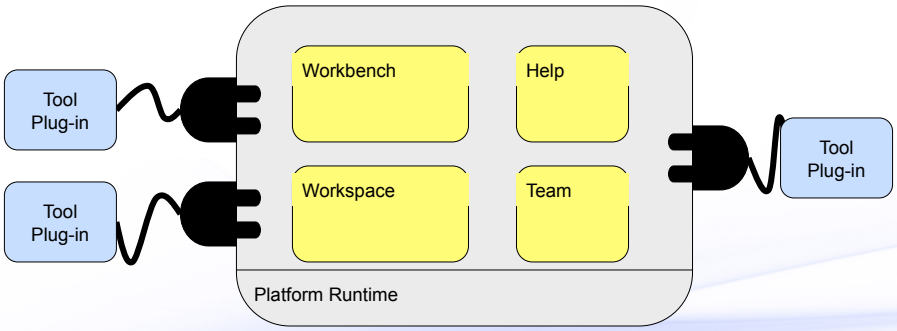
zeligsoft

The Motivation Behind Eclipse

- Support for the construction of application development tools
- Support for the development of GUI and non-GUI application development
- Support for multiple content types
  - Java, HTML, C, XML
- Facility the integration of tools
- Cross-platform support

zeligsoft

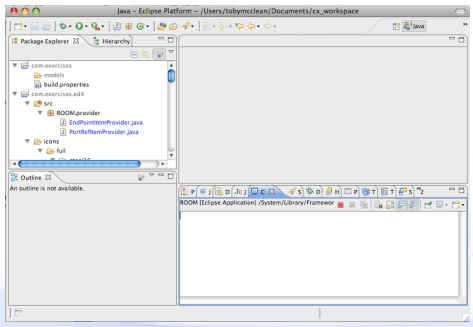
Eclipse Architecture



zeligsoft

Workbench

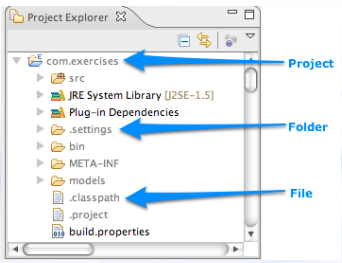
- The desktop development environment
  - Tools for resource management
  - Common of navigating through resources
- Multiple workbenches can be opened at the same time
  - Different workspaces



zeligsoft

Workspace

- Represents the user data
- A set of resources
  - Projects
    - Collection of files and folders
  - Folders
    - Contain other folders or files
  - Files



zeligsoft

Help

- For the creation and publishing of documentation
- Supports both
  - User guides
  - Programmer guides
- Created using
  - HTML for content
  - XML for navigation
- Support for context sensitive help

zeligsoft

Team

- Provides support for
  - Versioning
  - Configuration management
  - Integration with team repository
- Allows team repository provider to hook into environment
  - Team repository providers specify how to intervene with resources
- Has optimistic and pessimistic locking support

zeligsoft

How is Eclipse Used?

- As an Integrated Development Environment
  - Supports the manipulation of multiple content types
  - Used for specifying and designing applications
    - Requirements, models, documentation, etc.
  - Used for writing code
    - Java, C, C++, C#, Python, ...
- As a product base
  - Supported through plug-in architecture and customizations

zeligsoft

Eclipse as an IDE

- Java Development Tooling
  - Editors, compiler integration, ant integration, debugger integration, etc.
- C/C++ Development Tooling
  - Editors, compiler integration, make integration, debugger integration, etc.
- Dynamic languages
  - Editors, interpreter and compiler integration, debugger integration, etc.
- Modeling projects

zeligsoft

Eclipse as a Product Base

- Eclipse can be used a Java product base
- Its flexible architecture can be used as product framework
  - Reuse plug-in architecture
  - Create new plug-ins
  - Customize the environment
- Support for branding
- Rich client platform support

zeligsoft

Rich Client Platform

- A minimal set of Eclipse plug-ins necessary for building a product
- Control over resource model
- Control over look and feel
- But still capable of leveraging existing plug-ins

zeligsoft

Summary

- In this module we explored
  - Eclipse, its background and the components that form its foundation
  - Eclipse use cases

zeligsoft

Agenda

- What is Eclipse?
- The Eclipse Workbench
- The Eclipse Architecture
- Extending the Workbench

zeligsoft

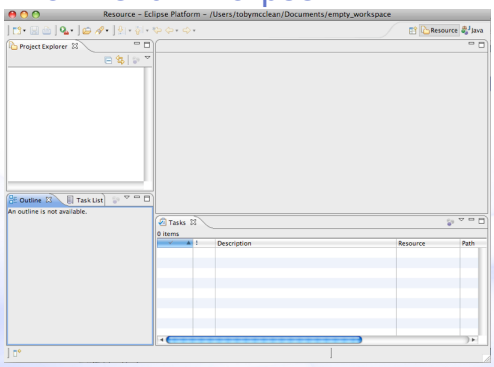
The Eclipse Project

The Eclipse Workbench

zeligsoft

What is the Workbench?

- The working environment in Eclipse



zeligsoft

Multiple Workbench Instances

- Instance of Workbench comes up when Eclipse is launched
- It is possible to open another window for the Workbench
  - Window → New Window
  - This opens up a new Workbench window
  - Can have different perspectives open in the different windows
  - Same result is not achieved by launching Eclipse twice

zeligsoft

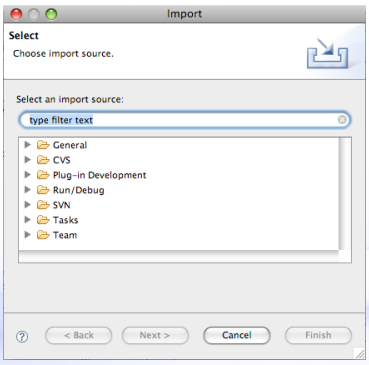
Resources in a Workbench

- When working in Eclipse, you work with Resources
- Resources are organized in file/directory structure in the Workbench
  - They correspond to the actual files and directories in the Workspace
  - There are four different levels of resources:
    - Workspace root
    - Projects
    - Folders
    - Files
- Resources form the file system can be dragged into the workbench

zeligsoft

Importing Resources

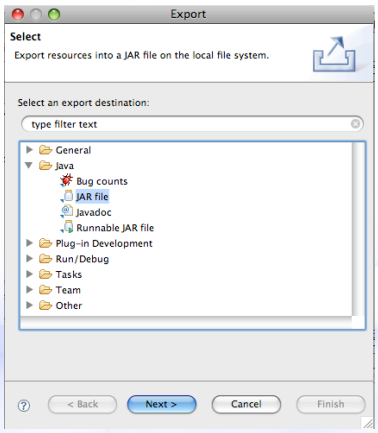
- Available through
  - Menu option **File → Import...**
  - Right-click menu **Import...** option on a Folder or Project



zeligsoft

Exporting Resources

- Available through
  - Menu option **File → Export...**
  - Right-click menu **Export...** option on a Resource in the Workbench



zeligsoft

Refreshing Workbench

- Used for refreshing resources that change in the Workspace outside the workbench
- For example, if a file added to a directory in the Workspace
  - Select the project or directory in the workbench
  - Choose **Refresh** from its context menu
  - Alternatively you can press the **F5 button** on your keyboard
- Best practice
  - Work with the resources from within the Workbench when possible

zeligsoft

Resource History

- Changing and saving a resource results in a new version of the resource
  - All resource versions are stored in local history
  - Each resource version is identified by a time stamp
  - This allows you to compare different versions of the resource
- There are two actions to access the local history of a resource, from its context menu
  - **Compare With → Local History...**
  - **Replace With → Local History...**

zeligsoft

Workbench Components

- The Workbench contains Perspectives
- A Perspective has Views and Editors

zeligsoft

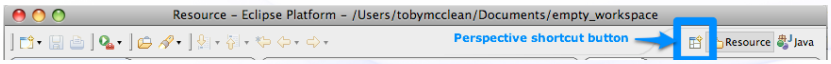
Perspectives

- Perspective defines the initial layout of views in the Workbench
- Perspectives are task oriented, i.e. they contain specific views for doing certain tasks

zeligsoft

Choosing a Perspective

- To change the current perspective of the Workbench
  - **Window → Open Perspective →**
  - Clicking on the Perspective shortcut button



zeligsoft

Saving a Perspective

- Arrangement of views and editors can be modified and saved for perspectives
  - **Window → Save Perspective As...**
  - It can be saved under an existing name or a new name creating a user-defined perspective

zeligsoft

Resetting a Perspective

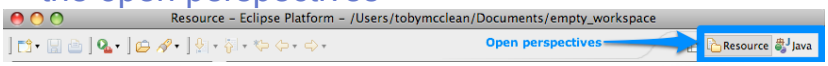
- Sometimes views and editors for a perspective need to be reset to the defaults
  - **Window → Reset Perspective...**
- This only applies to default Eclipse perspectives not user-defined ones

Customizing Perspectives

- The shortcuts and commands for the current perspective can be customized
  - **Window → Customize Perspective...**
- This is in addition to the customization of the views that are available in the perspective

Closing Perspectives

- The upper right corner of the workbench shows the open perspectives



- It is possible to close a perspective
  - **Window → Close Perspective**
  - **Window → Close All Perspectives**

zeligsoft

Editors

- An editor for a resource opens when you double-click on a resource
  - Editor type depends on the type of resource
  - An editor stays open when the perspective is changed
  - Active editor contains menus and toolbars specific to the editor
  - When a resource has been changed an asterisk in the editor's title bar indicates unsaved changes

zeligsoft

Editors and Resource Types

- It is possible to associate an editor with a resource type by the following actions
  - **Window → Preferences**
  - Select **General**
  - Select **Editors**
  - Select **File Associations**
  - Select the resource type
  - Click the **Add** button to associate it with a particular editor
- In same dialog the default editor can be set
- Others are available from **Open With** in the resources context menu

zeligsoft

Views

- The main purpose of a view is
  - Support editors
  - Provide alternative presentation and navigation
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are only available in that view

zeligsoft

More Views

- Views can
  - Appear on their own
  - Appear stacked with other views
- Layout of views can be changed by clicking on the title bar and moving views
  - Single views can be moved together with other views
  - Stacked views can be moved to be single views

zeligsoft

Adding Views to the Perspective

- To add a view to the current perspective
  - **Window → Show View → Other...**
  - Select the desired view
  - Click the **Ok** button

zeligsoft

Stacked Views and Stacking Views

- Stacked views appear in a notebook form
  - Each view is a page in the notebook
- A view can be added to a stack by dragging into the area of the tab area of the 'notebook'
- Similarly a view can be removed from a stack by dragging its tab away from the 'notebook'

zeligsoft

Fast Views

- A fast view is hidden and can be quickly opened and closed
- Created by
  - Dragging an open view to the shortcut bar
  - Selecting Fast View from the view's menu
- A fast view is activated by clicking on its Fast View pop-up menu option
  - Bottom right of the workbench
- Deactivates as soon as focus is given to another view or editor

zeligsoft

Summary

- In this module we explored
  - Components of the Eclipse workbench
    - Perspectives;
    - Editors; and
    - Views

zeligsoft

Agenda

- What is Eclipse?
- The Eclipse Workbench
- The Eclipse Architecture
- Extending the Workbench

zeligsoft

The Eclipse Project

Eclipse Architecture

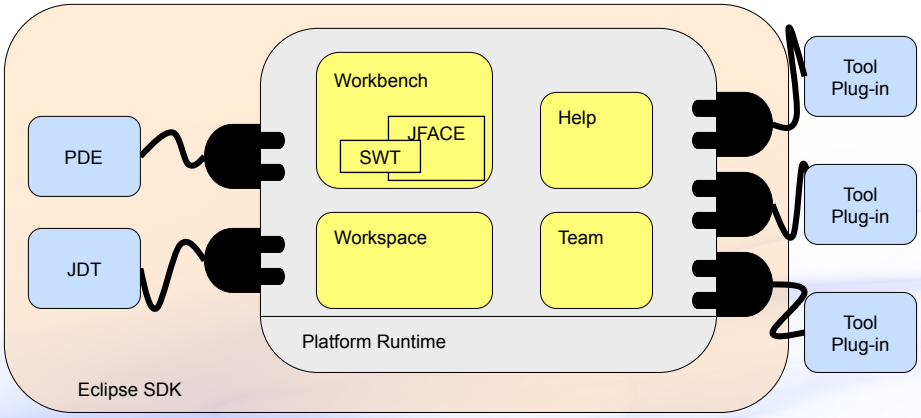
zeligsoft

Eclipse Architecture

- Flexible, structured around
  - Extension points
  - Plug-ins
- Architecture allows for
  - Other tools to be used within the platform
  - Other tools to be extended
  - Integration between tools and the platform

zeligsoft

More Eclipse Architecture



zeligsoft

Platform Runtime

- Everything is a plug-in except the Platform Runtime
  - A small kernel that represents the base of the Platform
- All other subsystems build up on the Platform Runtime following the rules of plug-ins
  - i.e. they are plug-ins themselves

zeligsoft

Extension Points

- Describe additional functionality that could be integrated with the platform
  - Integrated tools extend the platform to bring specific functionality
- Two levels of extending Eclipse
  - Extending the core platform
  - Extending existing extensions
- Extension points may have corresponding API interface
  - Describes what should be provided in the extension

zeligsoft

Plug-ins

- External tools that provide additional functionality to the platform
- Define extension points
  - Each plug-in defines its own set of extension points
- Implement specialized functionality
  - Usually key functionality that does not already exist in the platform
- Provide their own set of APIs
  - Used for further extension of their functionalities

zeligsoft

More Plug-ins

- Plug-ins implement behavior defined through extension point API interface
- Plug-in can extend
  - Named extension points
  - Extension points of other plug-ins
- Plug-in can also declare an extension point and can provide an extension to it
- Plug-ins are developed in the Java programming language

zeligsoft

What Makes Up a Plug-in?

- Plug-ins consist of
  - Java code
    - Binaries
    - Source (optional)
  - plugin.xml
    - Defines plug-in extensions, and
    - Declares plug-in extension points
  - plugin.properties
    - For localization and configuration
  - manifest.mf
    - describes the plug-in

zeligsoft

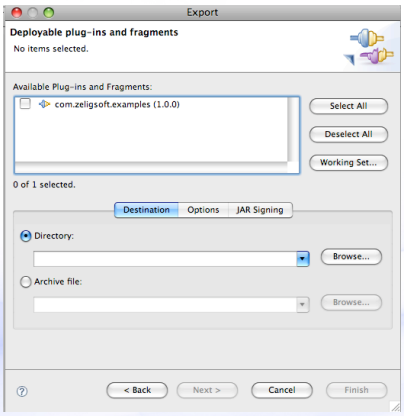
Publishing Plug-ins

- Prepares plug-in for deployment on a specific platform
- Manual publishing
  - Using Export Wizard
  - Using Ant Scripts
- Automatic publishing is available by using PDE Build

zeligsoft

Manually Publishing a Plug-in

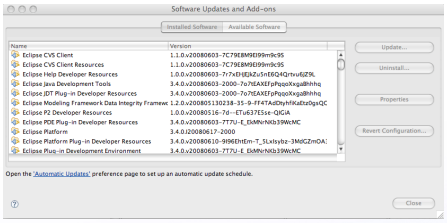
- A plug-in can be published manually by
  - **File → Export...**
  - Select Deployable plug-ins and fragments
  - Click **Next >** button
- Parameters configure how the plug-in published
- Can be saved as an Ant script



zeligsoft

Installing Plug-ins

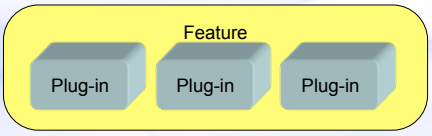
- Plug-ins are installed under \plugins\ directory of Eclipse installation
- Plug-ins can be installed from a update site
  - **Help → Software Updates..**
  - Typically as part of a feature
- Restart workbench



zeligsoft

Features

- An Eclipse feature is a collection of plug-ins
  - Represents smallest unit of separately downloadable and installable functionality
- Installed into \features\ directory of Eclipse installation
- Features can be installed from an update site



zeligsoft

Product

- Stand-alone application built on Eclipse platform
- Typically include the JRE and Eclipse platform
- Defined by using an extension point
- Configurable
  - Splash screen
  - Default preference values
  - Welcome pages
- Customized installation

zeligsoft

Summary

- In this module we explored
  - Eclipse architecture
  - Extension points
  - Plug-ins
  - Features
  - Products

zeligsoft

Agenda

- What is Eclipse?
- The Eclipse Workbench
- The Eclipse Architecture
- Extending the Workbench

zeligsoft

The Eclipse Project

Plug-ins

zeligsoft

Integration Between Plug-ins

- Supported through the ability to contribute actions to existing plug-ins
  - New plug-in contributes an action to existing plug-in
  - Allows for tight integration between plug-ins
- There are many areas where actions can be contributed
  - Context menus and editors
  - Local toolbar and pull-down menu of a view
  - Toolbar and menu of an editor - appears on the Workbench when editor opens
  - Main toolbar and menu for the Workbench

zeligsoft

Extending Views

- Changes made to a view are specific to that view
- Changes can be made to view's
  - Context menu
  - Menu and toolbar

zeligsoft

Extending Editors

- When extending an editor changes can be made to
  - Editor's context menu
  - Editor's menu and toolbar
- Actions defined are shared by all instances of the same editor type
  - Editors stay open across multiple perspectives, so their actions stay the same
  - When new workspace is open, or workbench is open in a new window, new instance of editor can be created

zeligsoft

Action Set

- Allows extension of the Workbench that is generic
  - Should be used for adding non-editor, or non-view specific actions to the Workbench
  - Actions defined are available for all views and editors
- Allows customization of the Workbench that includes defining
  - Menus
  - Menu items
  - Toolbar items

zeligsoft

Choosing What to Extend

- Eclipse has a set of predefined extension points
  - Called Platform extension points
    - Comprehensive set of extension points
    - Detailed in the on-line help
- It is possible to create new extension points
  - Requires id, name, and schema to be defined
  - Done through Plug-in Development Environment (PDE)

zeligsoft

Some Common Extension Points

- Popup Menus for editors and views
  - org.eclipse.ui.popupMenus
- Menu and toolbar for views
  - org.eclipse.ui.viewActions
- Menu and toolbars for editors
  - org.eclipse.ui.editorActions
- Menu and toolbar for the Workbench
  - org.eclipse.ui.actionSets
- Complete set of extension points located in Eclipse help
  - Search on “Platform Extension Points”

zeligsoft

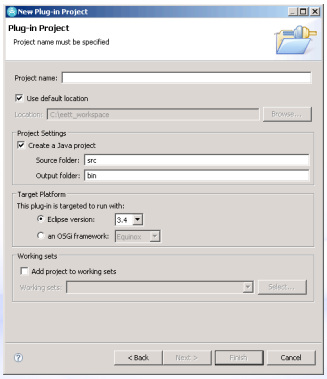
How to Extend the Workbench?

- Steps for adding a plug-in
  - Define Plug-in project
    - Plug-in nature
    - Java nature (optional)
  - Write the plug-in code
    - Create Java class
    - Add appropriate protocols to the class
  - Package the class
  - Create plugin.xml and MANIFEST.MF
  - Test the plug-in
  - Deploy the plug-in

zeligsoft

Creating Plug-in Project

- Plug-in project are created with the New Project wizard
  - File → New → Project…
- Project will contain
  - source code for the plug-in
  - manifest and property files

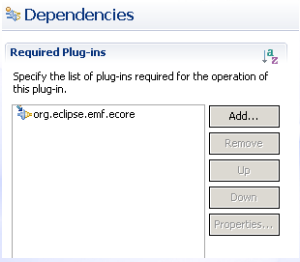


Implementation details

zeligsoft

Updating Project's Dependencies

- To make classes required for the plug-in visible for the project, update its dependencies
  - You add a dependency on another plug-in
  - You can add a jar that is not part of a plug-in to the build path



Implementation details

zeligsoft

Creating a Class

- For menu actions, define a class that
  - Subclasses client delegate class
  - Implements interface that contributes action to the Workbench

zeligsoft

Interface IWorkbenchWindowActionDelegate

- Extend IActionDelegate and define methods
  - init(IWorkbenchWindow) - Initialization method that connects action delegate to the Workbench window
  - dispose() - Disposes action delegate, the implementer should unhook any references to itself so that garbage collection can occur
- Interface is for an action that is contributed into the workbench window menu or toolbar

zeligsoft

Class ActionDelegate

- Abstract base implementation for client delegate action (defines same methods as interface)
- In addition it also defines
  - runWithEvent(IAction, Event)
    - Does the actual work when action is triggered
    - Parameters represent the action proxy that handles the presentation portion of the action and the SWT event which triggered the action being run
    - Default implementation redirects to the run() method
  - run(IAction)
    - Inherited method, does the actual work as it's called when action is triggered
  - selectionChanged(IAction, ISelection)
    - Inherited method, notifies action delegate that the creation in the Workbench has changed

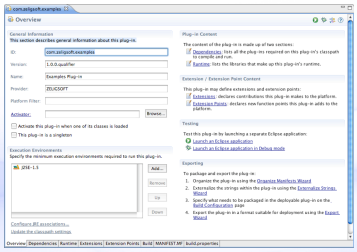
zeligsoft

Defining Manifest

- MANIFEST.MF file
  - In the META-INF directory
- Defines
  - Plug-in name
  - Plug-in version
  - Plug-in vendor/provider
  - Plug-in dependencies
  - Packages exported
- Manifest editor available

com.zeligsoft.examples

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Examples Plug-in
Bundle-SymbolicName: com.zeligsoft.examples
Bundle-Version: 1.0.0.qualifier
Bundle-Vendor: ZELIGSOFT
Bundle-RequiredExecutionEnvironment: J2SE-1.5
Require-Bundle: org.eclipse.emf;bundle-version="2.4.0"
Export-Package: com.zeligsoft.examples
Import-Package: org.eclipse.uml2.uml
```



zeligsoft

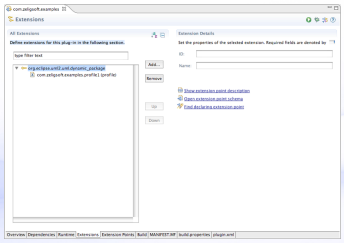
Defining plugin.xml

- In the root of the project
- Defines
  - Extensions
  - Extension points
- Extension editor
  - Includes wizards
- Extension point editor

com.zeligsoft.examples

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
   <extension
         point="org.eclipse.uml2.uml.dynamic_package">
      <profile
            uri="com.zeligsoft.examples.profile1">
      </profile>
   </extension>

</plugin>
```

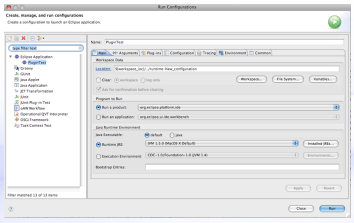


Implementation details

zeligsoft

Testing Plug-in from Workbench

- Available by opening new Workbench instance
  - Run → Run As → Eclipse Application
- Used when developing plug-in within the platform
- Can be configured
  - Run → Run Configurations...
  - Eclipse Application
- Debug available
  - Run → Debug As → Eclipse Application



Implementation details

zeligsoft

Getting Ready to Export Plug-in

- Create a build.properties file in your project

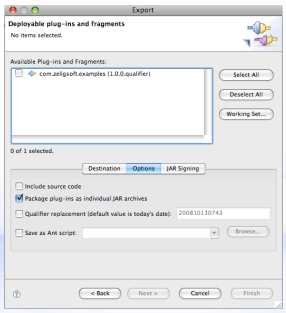
```
source.. = src/
output.. = bin/
bin.includes = META-INF, \
               .,\
               plugin.xml, \
               plugin.properties
```

- Build properties editor available
  - Open with → Build Properties Editor

zeligsoft

Exporting a Plug-in

- **File → Export...**
  - Deployable Plug-ins and Fragments
- Configure export
  - Archive vs. Directory
  - Packaging
  - Include source?
- Can save as ant script
- Ready to install



zeligsoft

Plug-in Registry

- Too see what plug-ins are registered within a Workbench instance
  - **Window → Show View → Other...**

Implementation details

zeligsoft

Summary

- In this module we explored
  - Plug-ins
  - MANIFEST
  - Plug-in descriptors
  - Testing plug-ins
  - Exporting plug-ins

zeligsoft

Questions