Introduction to Eclipse

Ed Gehring

Using (with permission) slides developed by—
Dwight Deugo (dwight@espirity.com)
Nesa Matic (nesa@espirity.com)

Overview

1. Overview
2. Installing and Running
3. Building and Running Java Classes
4. Debugging

Module Road Map

1. Overview
   • Background
   • Architecture
   • Components
   • Usage scenarios
2. Installing and Running
3. Building and Running Java Classes
4. Debugging
What is Eclipse?

- Eclipse is an open source project
  - [http://www.eclipse.org](http://www.eclipse.org)
  - Consortium of companies, including IBM
  - Launched in November 2001
  - Designed to help developers with specific development tasks
- Consists of four separate projects:
  - Eclipse Project
  - Eclipse Tools Project
  - Eclipse Technology Project
  - Eclipse Web Tools Platform Project

IBM’s IDE History

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

The Eclipse Platform Motivation

- Application development tools construction support
- Independent tool vendors support
- GUI and non-GUI application development support
- Numerous content types support
  - Java, HTML, C, XML, ...
- Easy integration of tools
- Use of Java language for writing the tools
- Multiple operating systems support
Plug-in Architecture

- Eclipse Platform
  - Workbench
  - Help
  - Workspace
  - Team

- Tool [plug-in]
- Tool [plug-in]

- Multiple workbenches can be opened at the same time

Workbench

- Represents the desktop development environment
  - It contains set of tools for resource management
  - It provides common way of navigating through the resources

- Multiple workbenches can be opened at the same time

Workspace

- Represents users data
- It is a set of user defined resources
  - Files
    - Contain arbitrary number of bytes
  - Folders
    - Contain other folders or files
  - Projects
    - Collections of files and folders
Help

- Used for creating and publishing documentation
- There are two different documentation styles:
  - Help style documentation is published in the user guide
  - API documentation is published in the programmer guide
- Help content is in HTML format
- Help navigation is in XML format

Team

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

How is Eclipse Used?

- As an IDE - Integrated Development Environment
  - Supports the manipulation of various content types
  - Used for writing code
- As a product base
  - Supported through plug-in architecture and customizations
Eclipse as an IDE

- Java Development Tooling (JDT) is used for building Java code
- Provides set of workbench plug-ins for manipulating Java code
  - Java projects, packages, classes, methods, ....
- Java compiler is built in
  - Used for compiling Java code
  - Creates errors (special markers of code) if compilation fails

Eclipse as a Product Base

- Eclipse can be used as a Java product base
- Its flexible architecture used as a product framework
  - Reuse plug-in architecture
  - Create new plug-ins
  - Customize the environment

Module Road Map

1. Overview
2. Installing and Running Eclipse
   - Where to get Eclipse?
   - What is the support for Eclipse?
   - Installing Eclipse
   - Running Eclipse
3. Building and Running Java Classes
4. Debugging
Getting Eclipse

http://www.eclipse.org

- Main point for finding Eclipse resources
  - Downloads
  - Articles
  - Discussion groups
  - Bugs
- Contains various resources for using Eclipse
- Contains references to other Eclipse related sites

Downloading Eclipse’s Install Zip File

- Click on the Download from the main page on http://www.eclipse.org
  - Choose the closest site from which to download (geographical sites hosting the download)
  - Choose the build for download (usually the latest build)
  - Choose the platform for download and type of download (http or ftp)
  - Specify where to save the download locally
Installing the Eclipse

- Unzip the downloaded file to the directory of your choice

![Image of Eclipse installation](img1.png)

---

eclipse.exe

- Eclipse is run by double-clicking on the `eclipse.exe` file
- The file is located in the Eclipse installation directory
- If there is no path set for `javaw.exe` program, the following dialog will come up

![Image of Eclipse dialog](img2.png)

---

Missing a Java VM? ...

- Download the latest Java Runtime Environment (JRE), e.g., v1.4.1_02 from [http://java.sun.com/j2se/](http://java.sun.com/j2se/)
- Click on the downloaded EXE file to install
- When given the option by the installer, identify a directory of your choice for the install files
- Update the PATH environment variable to include the JRE’s bin directory

---
Specifying JVM

- There are two ways of specifying JVM for Eclipse:
  - By installing JVM under the \(\text{jre}\) directory off the eclipse install directory
  - By specifying existing JVM in the PATH environment variable

Start -> Control Panel -> System -> Advanced -> Environment Variables

Running Eclipse

- When Eclipse is run, a Workbench opens

Default Workspace

- The default workspace is created when Eclipse runs for the first time
  - The workspace is located under \workspace directory of Eclipse installation directory
  - For example c:\eclipse\workspace
- The workspace contains user defined data – projects and resources such as folders and files
Running Different Workspace...

- It is possible to run workspace other than default
  - -data argument must be used with eclipse.exe
  - Workspace location must be specified
- Useful for grouping project specific data
- Multiple workspaces can run at the same time

... Running Different Workspace

- Customize your working directory by creating a shortcut identifying the eclipse.exe and the working directory

Module Summary

- In this module you have learned:
  - What Eclipse is, its background and components
  - How Eclipse is used
  - How to download, install and run Eclipse
  - How to create and run multiple workspaces with Eclipse
Module Road Map

1. Overview
2. Installing and Running
3. Building and Running Java Classes
   - Developing Java applications
   - Projects, packages, classes
   - Browsing Java code
   - Searching Java code
   - Organizing Java code
   - Using Code Assist
   - Running Java applications
   - Scrapbook
4. Debugging

Java Development Tooling - JDT

- Eclipse's Java Development Environment is often referred to as JDT – Java Development Tooling
  - Using the JDT you can do the following with the Java programs:
    - Write
    - Compile
    - Test
    - Debug

Perspectives

- When developing Java code commonly used perspectives are:
  - Java Perspective
    - Designed for working with Java projects
  - Java Browsing Perspective
    - Designed for browsing structure of Java projects
  - Java Type Hierarchy Perspective
    - Designed for exploring type hierarchy
  - Debug Perspective
    - Designed for debugging Java programs
Java Perspective

- Contains:
  - Editor area
  - Package Explorer View
  - Hierarchy View
  - Outline View
  - Tasks View

Java Browsing Perspective

- Contains:
  - Editor area
  - Projects View
  - Packages View
  - Types View
  - Members View

Java Type Hierarchy Perspective

- Contains editor area and Hierarchy View
New Project Preferences

- You can set global preferences for a project
- Select Window -> Preferences to get Preferences View
- Good idea to separate your Java files into source and compiled directories (src and bin)
- This action only needs to be done once
- Done for all subsequent projects

Creating Java Project

- Project used to organize resources (source, class, icons) for a project
- To create Java project:
  - Select Window -> New -> Project... from the menu
  - The New Project wizard comes up
  - Select Java -> Java Project
  - Click Next

New Project Wizard

- Specify Project Name
- Click Next
Java Settings

- Specify Java settings
  - Output folder (where compiled files will be stored)
  - External jar files project depends on
  - Classes from other projects that are referenced in the project
- Click Finish

Project Properties

- You can change the Java build path at any time
  - Choose Properties from the context menu on the project

Creating Packages

- Package contains Java class files
- To create a package for a project:
  - Select the project
  - Choose New → Package from the context menu
  - Specify package name
  - Click Finish
Creating Classes

- To create a class in a package:
  - Select the package
  - Choose New → Class from the context menu
  - The Class wizard comes up
  - Specify class details
  - Click Finish

Using Code Assist

- When activated, code assist opens a list of available code completions
- Code Assist activates by Ctrl+Space
  - Activates automatically when a message needs to be sent to an object (after the dot is typed)

Using Quick Fix

- Useful if Java compiler shows errors
  - Gives options for fixing the errors
  - Activated through Edit → Quick Fix menu option
Searching for Java Classes

- When developing Java application a good search mechanism is very important
  - You often search for class, method declarations, and references
  - It is important to be able to find things quickly
- Eclipse Java Search mechanism is very extensive
- It allows you to search for:
  - Types, methods, constructors, packages, and fields
  - Declarations, Implementers, References
  - In the scope of Workspace, Working Set, or Selected Resources

Organizing Java Code

- Eclipse comes with extensive support for organizing and refactoring Java code
- It is possible to:
  - Generate getters and setters for the fields
  - Organize missing import statements
  - Move fields, methods, classes
  - Rename methods, classes, packages

Generating Getters and Setters

- Available for creating get and set methods on the fields
  - It supports encapsulation
  - Activated by choosing Source → Generate Getter and Setter from the editor’s context menu
Refactoring

- Available from the Refactor context menu option in the editor
- Helps with refactoring Java code
- Allows for:
  - Renaming
  - Methods, fields, packages, projects, parameters, or local variables
  - Changing of method signature
  - Pull up a field or method (into super class)
  - Push down a field or method (into sub class)
  - Encapsulate field (generate getter and setter)
  - Extract method, local variable, or constant from an expression

Running Java Classes

- To Run Java application
  - Choose Run → Run... from the menu

Console View

- Represents standard Java console
- Opens by default when standard Java output is used
- Can also be open from Window → Show View menu

```java
System.out.println("Hello World");
```
Scrapbook...

- Allows for writing and executing of Java code
  - Very useful for quick test of Java code that you write
- The Java code in the Scrapbook can be:
  - Displayed as a string when evaluated
  - Inspected when evaluated
    - Opens an Inspector view where you can see returning object from evaluation and all containing fields
  - Executed

...Scrapbook...

- It is created by selecting a project and choosing New \text{\textarrow{\text{New}}} \text{\textarrow{\text{Scrapbook Page}}} from the Package Explorer’s context menu and then entering the name of the page
- Your scrapbook page will become a resource in your project

... Scrapbook...

- To open the scrapbook page just click on the resource
- It opens up like a Java source file editor
- Type Java code and select the context menu to Display or Inspect
...Scrapbook

- Class names must be fully qualified in what you type
- Set imports to make life easier
- Think of your scrapbook as a page that Eclipse will take the source you type, wrap it in a class with your source in the main menu, then compile and execute

Summary

- You have learned:
  - How to create projects, packages and classes
  - How to browse Java code and search for Java code
  - How to use coding assistance
  - How to organize Java code
  - How to run Java code
  - How to use the Scrapbook

Module Road Map

1. Overview
2. Installing and Running
3. Building and Running Java Classes
4. Eclipse Debugging
Module Road Map

1. Eclipse Debugging
   - Debug Perspective
   - Debug Session
   - Breakpoint
   - Debug Views
   - Breakpoint Types
   - Evaluating and displaying expressions

Debugging in Eclipse

- The Java Debugger
  - Part of Eclipse Java Development Tools (JDT)
  - More than System.out.println(error)
  - Detects errors as code executes
  - Correct errors as code executes
  - Actions you can perform debugging include:
    - Control Execution
    - Set simple breakpoints
    - Set conditional breakpoints
    - Review and change variable values
    - Hot code replace (feature new to JRE 1.4)
Simple Breakpoint

- Breakpoint
  - Stops the execution of a program at the point
  - Thread suspends at the location where the breakpoint is set
- Setting a breakpoint
  - CTRL+Shift+B at current point in editor line
  - Double click in editor marker bar a current line

Starting a Debugging Session

- Select Java class containing the following:
  - main() method
  - Resulting execution will pass breakpoint
- Select Run → Debug As… → Java Application
- Or Select Debug As… → Java Application from the debug menu

Debug Session

- Execution suspends prior to the line with a breakpoint
- You can set multiple breakpoints
Deleting Breakpoints

- Double click on the breakpoint in the editor

---

_Control Execution From Breakpoint..._

- **Step Into or F5:**
  - For methods, execute method and suspend on first statement in the method
  - For assignments, similar to Step Over
  - For conditionals, similar to Step Over
- **Step Over or F6**
  - Execute next statement
- **Step Return or F7**
  - Resume execution to the end of the method on the next line after it was invoked

---

...Control Execution From Breakpoint

- **Resume or F8**
  - Continue execution until program ends or another breakpoint is reached
- **Terminate**
  - Stops the current execution thread
Variables and Fields

- To see the values bound to fields:
  - Use Variables View
  - Select variable in editor and select Inspect
  - Select variable in editor and select Display

Code Debugging in this Module

```java
public class Debug {
    private int something = 0;
    private Vector list = new Vector();
    public void firstMethod() {
        thirdMethod(something);
        something = something + 1;
    }
    public void secondMethod() {
        thirdMethod(something);
        something = something + 2;
    }
    public void thirdMethod(int value) {
        something = something + value;
    }
    public static void main(String[] args) {
        Debug debug = new Debug();
        debug.firstMethod();
        debug.secondMethod();
    }
}
```

Variables View

- Shows all fields of instance where breakpoint occurred
  - Select this to see all fields
  - Select any field to see value
  - If field is bound to an object, you can select inspect from the menu to view its fields and values
### Changing Field Values

- To change field value:
  - Select field in Variables view
  - Select Change Variable Value from the menu
  - Enter new value into Set Variable Value window
  - Click OK

![Set Variable Value](image1)

### Display View

- Displays the result of evaluating any expression in the current context
- Opens by:
  - Selecting a field in the editor or Variables View and choosing Display
  - Clicking on the Display tab

![Display View](image2)

### Expressions View

- Remembers all objects you have inspected
- Displays the fields of the object
  - You can see the values of the fields
  - You can Inspect the fields
- Opens when:
  - You Inspect an object
  - You click on the Expressions tab

![Expressions View](image3)
Breakpoint View

- Lists all available breakpoints
- Can be used for manipulating breakpoints (through the views menu):
  - Enabling
  - Disabling
  - Removing
- Also displays breakpoints properties
- Accessed like other debugging views

Debug View

- Shows:
  - Active threads
  - Current stack frame when execution has stopped
  - Previous stack frames
- Method and variables are shown in the editor for the selected frame
  - Update in the editor updates the source

Breakpoint Types

- Breakpoints can be set for the following Java entities:
  - Line (simple breakpoint)
  - Method
  - Field (Watchpoint)
  - Java Exception
- Each breakpoint is set a different way and has different properties
Method Breakpoints

- To set method breakpoints:
  - Select method in the Outline View
  - From context menu select Add/Remove Method Breakpoint
- To set breakpoint’s properties:
  - Select breakpoint in editor
  - Select Breakpoint Properties.. from context menu
  - Set properties as desired
    - Entry, exit, enable hit count
- Execution suspends on entry/exit into method

Field Breakpoints

- Also known as watchpoint
- To set the watchpoint:
  - Select field in the Outline View
  - From context menu select Add/Remove Watchpoint
- To set watchpoint’s properties:
  - Select breakpoint in editor
  - Select Breakpoint Properties.. from context menu
  - Set properties as desired
    - Access/modification, enable
- Execution suspends on access/modification of field

Java Exception Breakpoint

- To Add Java Exception Point:
  - Select Add Java Exception Point from menu
  - Enter exception type
  - Specify what triggers a breakpoint:
    - Caught exception
    - Uncaught exception
    - Both
How To Debug

- Here are simple steps for debugging in Eclipse:
  - Set your breakpoints
  - Hit a breakpoint during execution
  - Walk/step through code to other breakpoints
  - Follow along in editor
  - Inspect/Display interesting fields
  - Watch the Console for things to happen

Summary

- You have learned:
  - The views in the Debug Perspective
  - Typical debug session
  - How to use the Inspector
  - About the different types of breakpoints
  - How to set breakpoints
  - How step around your code doing debugging