Eclipse Plug-ins Development

Radu Farcas – Eclipse Plug-Ins Developer
Catalin Udma – Linux Developer

02-July-2013

AGENDA

- Eclipse Overview
  - Eclipse Project Introduction
  - Eclipse Development Tools
    - Eclipse Projects
  - Eclipse IDE Tutorial
  - Eclipse as a Rich Client Platform
- Linux Tools
  - Presentation of Linux Tools
  - Eclipse Integration
- Eclipse Plug-ins Development
  - Eclipse Plug-in Mechanism
  - Write your first Eclipse Plug-in

AGENDA (2)

- Visualization Solution
  - Using strace and ptrace on x86
  - Developing Eclipse Visualization Plug-in for strace and ptrace on x86
- Building Domain Specific Languages
  - Eclipse Modeling Framework
- Data binding in Eclipse
- Eclipse Community
  - Contributing to Eclipse Community

Goals

- Get to know Eclipse
- General things about development tools
- User-level presentation of Eclipse
- Working with Linux Tools
- Theoretical concepts about plug-ins development
- Development of a dedicate plug-in for starce/ptrace
- Advanced Eclipse topics: Databinding and DSL
- Learn how to contribute to an open source project

- **Prerequisites**
  - Install JRE/JDK
  - Install Eclipse for JDT/Plug-Ins development (aka Standard or SDK)
  - Explore Eclipse site – navigate through hundreds of projects

Eclipse Overview

- **Eclipse** - a multi-language Integrated development environment (IDE)
  - Made of a base workspace and an extensible plug-in system
- Created by an Open Source community
- It is written mostly in Java
- It can be used to develop applications in Java
  - Also Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy, Scheme, and Erlang
- Initial codebase originated from IBM VisualAge

Eclipse Overview (2)

• Today it is the leading development environment for Java with a market share of approximately 65%

•The Eclipse Open Source community has over 200 Open Source projects covering different aspects of software development

•The Eclipse IDE can be extended with additional software components

- •With the exception of a small run-time kernel, everything in Eclipse is a plug-in

•The Eclipse projects are governed by the *Eclipse Foundation*.

- • Non-profit, member supported corporation that hosts the Eclipse Open Source projects

Download Eclipse

• Eclipse requires an installed Java runtime.

•Java can be downloaded in two flavors, a *JRE* (Java runtime environment) and a *JDK* (Java development tools) version

•The Eclipse IDE contains its own Java compiler hence a JRE is sufficient for most tasks with Eclipse

•JRE:

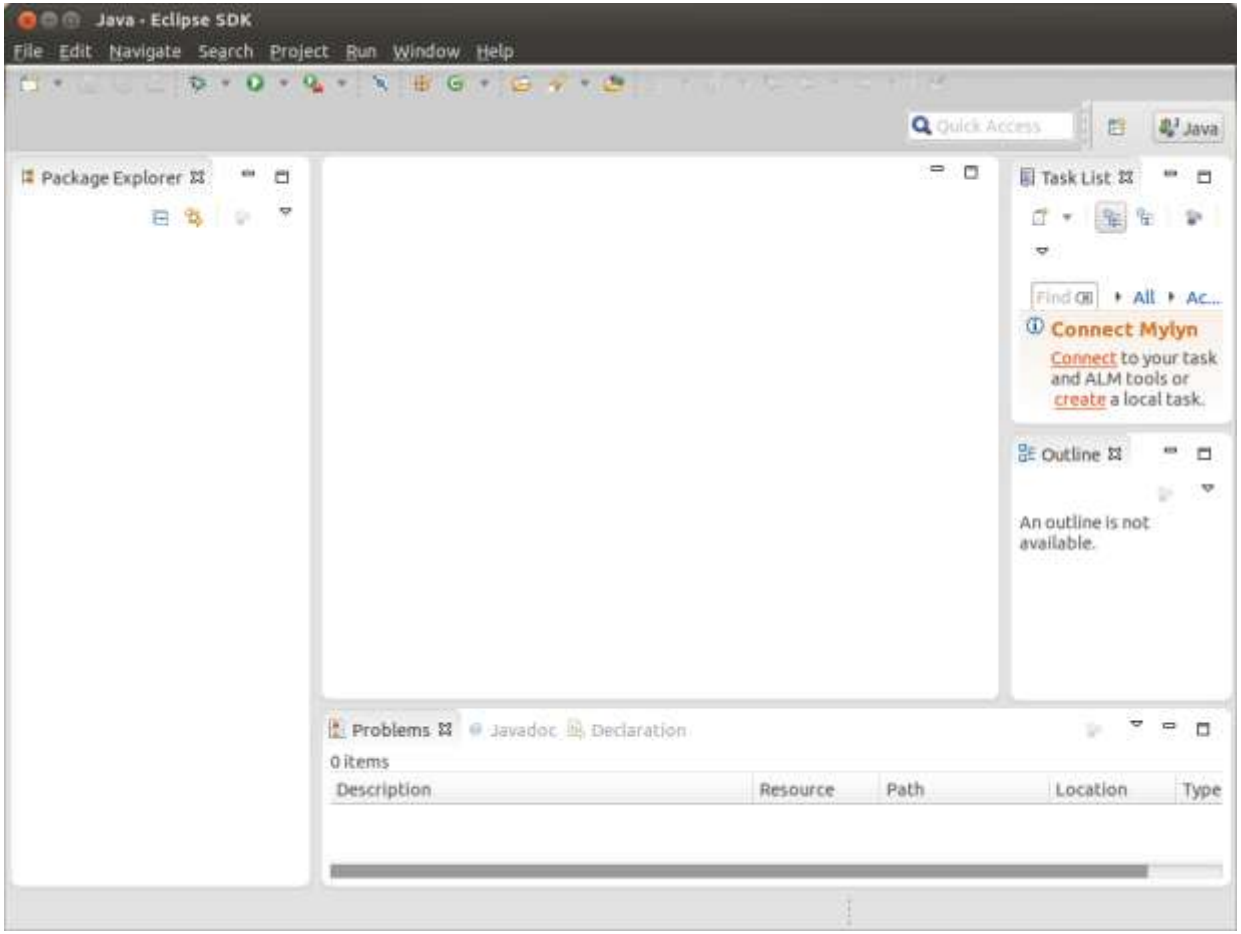
- •http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html

•The Eclipse.org website provides pre-packaged Eclipse distributions to provide downloads for typical use cases

- •The *Eclipse IDE for Java Developers* distribution is specifically designed for standard Java development: http://www.eclipse.org/downloads
- •The *Eclipse Classic/Standard* for plug-ins development

Starting Eclipse

•Specify an *workspace*. The *workspace* is the place in which you work



Eclipse Terminology

•The ***workspace*** is the physical location (file path) you are working in. Your projects, source files, images and other artifacts can be stored and saved in your workspace. The *workspace* also contains preferences settings, plug-in specific meta data, logs etc.

•You can choose the workspace during startup of Eclipse or via the menu (*File → Switch Workspace → Others*) .

•An **Eclipse project** contains source, configuration and binary files related to a certain task and groups them into buildable and reusable units. An Eclipse project can have *natures* assigned to it which describe the purpose of this project. For example the Java *nature* defines a project as Java project. Projects can have multiple natures combined to model different technical aspects.

Eclipse Workbench Elements

•**Views and editors - parts**
•*Parts* are user interface components which allow you to navigate and modify data. A *part* can have a menu and a toolbar. *Parts* are typically divided into *views* and *editors*

•A ***view*** is typically used to work on a set of data, which might be a hierarchical structure. If data is changed via the *view*, this change is typically directly applied to the underlying data structure. A *view* sometimes allows us to open an *editor* for a selected set of data

•***Editors*** are typically used to modify a single data element, e.g. a file or a data object. To apply the changes made in an editor to the data structure, the user has to explicitly save the editor content

•*Java editor* is used to modify Java source files. Changes to the source file are applied once the user selects the *Save* command. A dirty editor is marked with an asterisk

Eclipse Workbench Elements (2)

•A ***Perspective*** is a visual container for a set of *parts*. The Eclipse IDE uses *perspectives* to arrange *parts* and configure the menu and the toolbar for different development tasks. Open *editors* are shared between*perspectives*, i.e. if you have an *editor* open in the *Java perspective* for a certain class and switch to the *Debug*perspective, this *editor* stays open.

- *Save Perspective As..., Customize Perspective…,*
- *Window → Open Perspective → Java*.

•*Views - Window → Show View → Other*
•*Package Explorer view*, which allows you to browse your *projects*
• Several *editors* are stacked in the same container

Creating the First Project

- Exercise – Create a new Java project

•The *Outline view* shows the structure of the currently selected source file
•The *Problems view* shows errors and warning messages.
•The *Javadoc view* shows the documentation of the selected element in the Java *editor*.

Import RSS Feed Client Project

-  Exercise – RSS Feed Client

- Run the project
- Create debug configuration
- Start debugging process

Explore Eclipse IDE Features

- Navigating through your project with the *Package Explorer*
- *Package Explorer* allows you to filter the resources which should be displayed
- Close projects via right-click and by selecting the *Close Project* menu entry
- *Link with Editor* button in the *Package explorer view*
- Open any class by positioning the cursor on the class in an editor and pressing **F3**.
- Open type dialog: press **Ctrl**+**Shift**+**T** dialog (enter the class name to open it)
- *Quick Outline* is **Ctrl**+**O**. *Quick Outline* shows only the direct members and fields of the class. **Ctrl**+**O** again for inherited members
- *Open Type Hierarchy*(Shortcut: **F4**) on NodeList or DocumentBuilder
- *Search* → *Search* menu (Shortcut: **Ctrl**+**H**) you can open the search dialog of Eclipse
  - Java, File, etc
  - **Ctrl**+**K** shortcut to activate *Find Next*. Repeat **Ctrl**+**K** in order to move to the next occurrences of the current search term.
- Activate the *breadcrumb* mode for the Java editor which allows you to navigate the source code directly from the Java editor (**Shift+Alt+B**)

Explore Eclipse IDE Features (2)

- Quick Access – **Ctrl+3 -** quick jump to any function (e.g. 'java class' for the class wizard)
- Open *Preferences → General → Keys* to find and redefine shortcuts at runtime
- *Content assist* is a functionality in Eclipse which allows the developer to get context sensitive code completion **Ctrl**+**Space**
- *Quick Fix* Select the underlined text and press **Ctrl**+**1** to see proposals how to solve this problem (check it with a compile error)
- Eclipse has several possibilities to generate code for you: *Generate Constructor from Fields*, *Generate Getter and Setter, Override/Implement Methods* ***(Source menu)***
- Refactoring
  - Rename refactoring: works nice inside the source file too (example)
  - Extract method (printElement in RSS Client)
  - Extract constants – create constants out of strings
- Code navigation:
  - Show references of a Java element **Ctrl+Shift+G**
  - Show Hierarchy for a Java type **Ctrl+Shift+H** (and choose a type)

Explore Eclipse IDE Features (3)

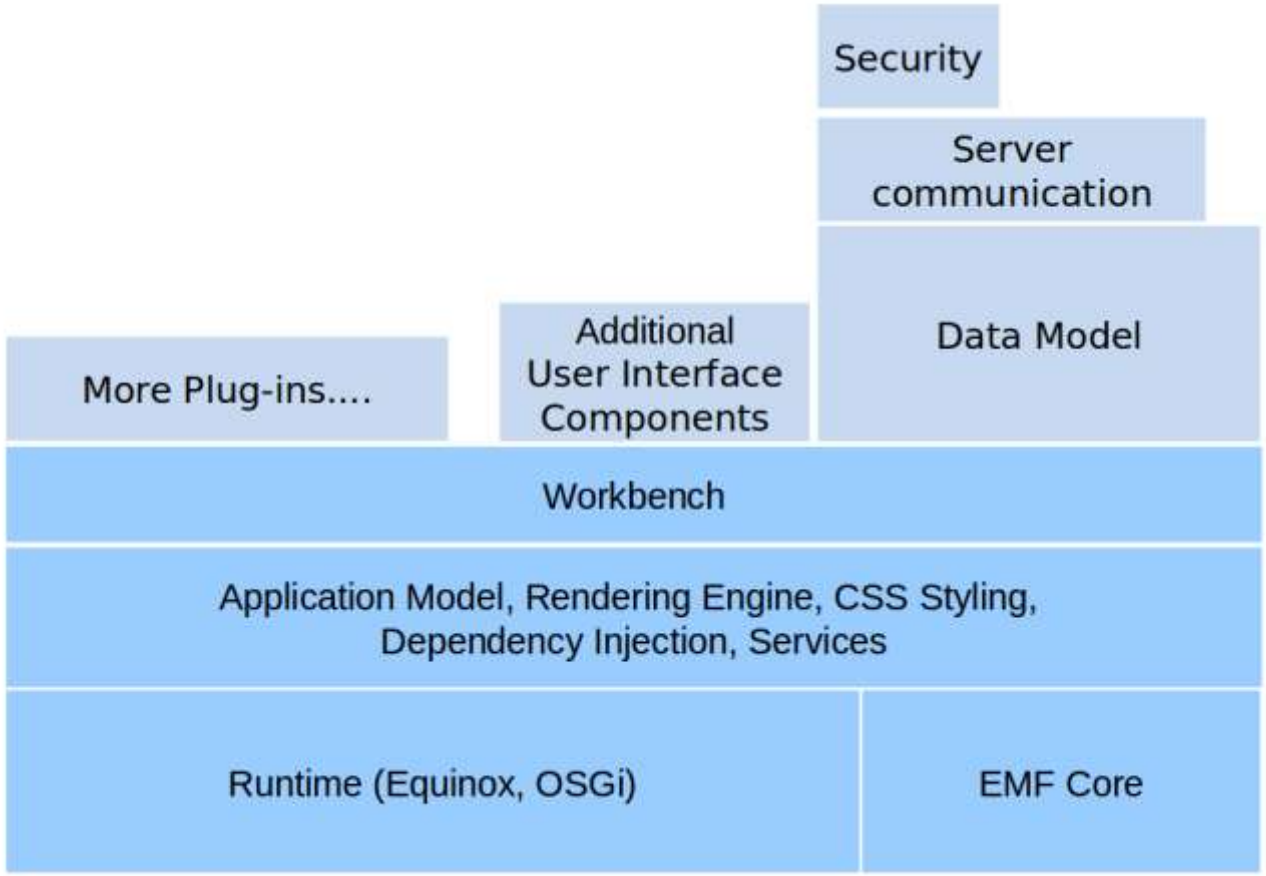
- Project dependencies ->*Java Build Path* and the *Projects* tab (e.g. First on RSS Client)
  - Only an Eclipse thing – outside Eclipse you need to create Java libraries for the projects
- Working with JAR files
  - Export a project as JAR (classes, sources and Eclipse project files too)
  - Add it as a library to the *Java Build Path*
- Eclipse update manager
  - Installable software components are called features and consist of plug-ins
    - *Help → Check for Updates* – updates for existing functionalities
    - *Help → Install New Software....* – for new functionality – looks into the available update sites
  - Uninstall features *Help → About Eclipse SDK → Installation Details*.
- Working with multiple JRE
  - Select active JRE
- Working sets and tasks
- Eclipse Java development preferences
  - JDT code checks

Eclipse Marketplace

- Included Eclipse *Marketplace client*
  - Search for components, discover popular extensions and see descriptions and ratings
  - You do not have to know the URL for the *software site*
- *Help → Eclipse Marketplace*.
  - Install Findbugs Eclipse plug-in http://en.wikipedia.org/wiki/FindBugs
    - Open source program which looks for Java bugs in code (static analysis)
  - Play with Findbugs on the RSS Client

Eclipse RCP

•Eclipse based applications which are not primarily used as software development tools are called Eclipse RCP applications



Eclipse SDK

- Eclipse software development kit (SDK), includes the Java development tools
  - Is meant for Eclipse plug-in or RCP developers
- Users can extend Eclipse abilities by installing plug-ins written for the Eclipse Platform
  - Development toolkits for other programming languages
  - Or, can write and contribute their own plug-in modules

- Offers an IDE with a built-in incremental Java compiler and a full model of the Java source files
- Eclipse implements widgets through a widget toolkit for Java called SWT (different than AWT or Swing)

Eclipse Licensing

- The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released

- The EPL is designed to be business-friendly
  - The consumer of EPL-licensed software can choose to use this software in closed source programs. Only modifications in the original EPL code must also be released as EPL code.

- The *Eclipse Foundation* also validates that source code contributed to Eclipse projects is free of Intellectual Property (IP) issues.
  - This process is known as IP cleansing.