Singapore Eclipse Meetup

Eclipse Overview

February 28th 2017
Table des matières

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>I - Presentation</td>
<td>7</td>
</tr>
<tr>
<td>II - Eclipse Eco system</td>
<td>9</td>
</tr>
<tr>
<td>III - Eclipse IDE</td>
<td>17</td>
</tr>
<tr>
<td>IV - OSGi</td>
<td>19</td>
</tr>
<tr>
<td>V - Eclipse Architecture</td>
<td>25</td>
</tr>
<tr>
<td>VI - Modeling Projects overview</td>
<td>31</td>
</tr>
<tr>
<td>VII - Conclusion</td>
<td>43</td>
</tr>
</tbody>
</table>
This talk is about:
- Eclipse Foundation
- Eclipse IDE
- OSGi Architecture
- Eclipse RCP Architecture
- Modeling projects

The goals of the talk are:
- to give you information about Eclipse projects
- to give you ideas to enhance your software development factory
- to start discussion around these technologies
OPCoach

Eclipse Expert

Olivier PROUVOST

25 rue Bernadette - 31100 Toulouse (France)
•+33 (0)6 28 07 65 64
olivier.prouvost@opcoach.com
@OPCoach_Eclipse
www.opcoach.com

➢ Eclipse Training : RCP, E4, Modeling, Build, given in French, English and ... Spanish
➢ Eclipse Consulting
➢ Recruitment service to link companies and job applicants
➢ Twitter : @OPCoach_Eclipse

1 - http://www.opcoach.com/en
Eclipse Eco system

Eclipse Eco System

- Foundation and goal
- Some figures
- Different groups

Eclipse Foundation

Goals:
- Providing a governance for Eclipse eco system:
  - managing projects
  - managing members
  - managing groups
- Hosting projects
- Promoting the Eclipse technologies and open source software
- Helping members and others to make business with Eclipse
- Organizing conferences

Structure:
- The Eclipse foundation is located in Ottawa and in Europe
- It is an independant organization founded by the members
Some History

The Eclipse ecosystem - is all about building communities

Some figures

- 300 projects (46 new in the past 12 months)
- 130 MLOC/year code change velocity
- >1 400 committers
- >5 million active users of Eclipse IDE
- 1.5 million downloads/month (average)
- 2 million unique visitors/month
- Leading IDE in Java, C/C++, PHP, ...
- >$5.0 million annual budget
- >240 members (13 strategic)
- ~30 staff
- >50 events per year
  - including 4 EclipseCon major events (US, France, Germany, India)

The Eclipse Secret

- Eclipse is the industry's best model for vendor-neutral collaboration on innovation
- Industry collaboration requires:
  - Licensing model for sharing co-developed innovation
  - IP management to maximize commercialization opportunities
  - Project model for coordinating investments and activities
  - Governance model to ensure a level playing field for all participants
- Eclipse gives you these "out of the box"
Members

Members of Eclipse

Strategic Members

Strategic Members

Eclipse overview, Meetup, Singapore, 28th February 2017
**Eclipse groups**

Eclipse foundation leads several technical groups

**Science Group**

---

**The Science Working Group**

- The Science Working Group (SWG) works to solve the problems of making science software inter-operable and interchangeable.
  - Develop reusable open source software for scientific research
  - Enable a collaborative approach to producing technologies used for interdisciplinary analysis of scientific data
  - Group efforts help permeate concepts and capabilities to different research areas which allows serendipitous discoveries to be made.
- Main projects:
  - Apogy, ChemClipse, DAWN, ICE
**PolarSys Group**

**The PolarSys Working Group**

- PolarSys is an Eclipse *Industry* Working Group created by large industry players and by tools providers to collaborate on the creation and support of Open Source tools for the development of embedded systems.
- Domains such as aerospace, defense and security, energy, health care, telecommunications, transportation must meet several requirements.
- Main projects:
  - Capella, CDT, Papyrus, Titan, Trace Compass

**Steering Committee Member**

**Participating Member**
IOT Group

The IoT Working Group

- Eclipse IoT is an ecosystem of companies and individuals that are working together to establish an Internet of Things based on open technologies.

Location Tech Group

The LocationTech Working Group

- LocationTech is a working group developing advanced location aware technologies

- Main projects:
  - GeoGig, GeoTrellis, GeoInfini, Geomesa, uDis
Eclipse IDE

The different versions

Image 1

Strengths

- RCP open architecture
- Adopted by all software vendors
- Rich environment and tools
- Lot of documentation
- Free
- Portable to all OS
- Community (foundation, members, ...)
- Stable and regular deliveries (1/year)
- Eclipse Public License (EPL)
- OSGi based architecture
Open Services Gateway initiative (OSGi)

Overview

**OSGi... why ??**

Java seems to be modular
- packages
- classes
- methods
- different jar files

All is well separated during development.
But what happens when you launch your java program ?

**The jar hell**

- Jar files do not have a version (it is only for documentation)
- Jar files are not bound together
- We have to know all the dependencies
- Classpath must be set manually
- We have to deal with : LinkageError, ClassNotFoundException, NoClassDefFoundError
- Two jar releases can not be mixed
- Managing the jar files becomes tough, this is the jar hell

**Launch Java**

- The JVM starts with a single class loader.
- The ‘modularity’ is dissolved.
- Everything must be configured manually
Updating the application

➢ The mail.jar library cannot send attached files
➢ Another component needs to send attached files
➢ Should we update all dependencies?
➢ OSGi will be a solution
➢ OSGi simplifies modularity and evolutions of your application

**Presentation**

➢ OSGi defines the 'Bundle' concept
➢ OSGi manages versions of bundles
➢ OSGi manages bundles activation
➢ OSGi manages services (Service Registry)

**OSGi Framework**

OSGi defines a model to manage java software modules
OSGi is a specification managed by the OSGi Alliance: [http://www.osgi.org](http://www.osgi.org)

2 - [http://www.osgi.org](http://www.osgi.org)
It has been started in 1999 for Java embedded devices

**The bundle**

It is defined by a MANIFEST:

Image 3 Manifest sample

- It has a unique ID and a version
- It may publish or consume services
- It will have its own class loader at runtime
- It can be defined for a specific platform (Windows, Linux ...)

**Relations with other bundles**

- The bundle may depend on other bundles
  - optionally depending on version numbers
- The bundle exports the packages it wants to make visible
- The classpath is computed automatically -> No more jar hell!

Image 4 Bundle relations

**Remark**: some of these concepts are redefined in the Java 9 Jigsaw Projects!
**Bundle Fragment**

- The fragment is a complement for a bundle (host bundle)
- The fragment is used to:
  - manage native code for a platform (windows, linux, mac..)
  - complete a bundle with additional files (i18n, ...)
  - write tests (has visibility on the bundle)
- The fragment is included in the host bundle at runtime

---

**Image 5 Fragment**

**Dynamic behavior**

The lifecycle of the bundle is managed by the OSGi engine:

- It can be started when necessary
- It can be stopped at any time
- It can be replaced by a more recent version
- Two different versions can coexist
- New services can be published during runtime
Running an OSGi application

Services
- A bundle can publish services
- A service is represented by a java interface
- It can be implemented by one or more bundles.
- Services are created with 'component' files
  - manually created with the 'new Component' wizard
  - automatically created with the @Component class annotation

Sample OSGi Architecture

Test fragments

Application bundles

Image 6 OSGi Engine

Image 7 OSGi architecture
**Eclipse 3 Architecture**

- The Eclipse architecture is based on the notion of plug-in.
- A plug-in is a software component element
- A plug-in is like an OSGi bundle

**Plug-in vs Bundle**

The plug-in defines two additional concepts on a bundle:

- **Extension point**: Defines a model of a concept that the plugin can handle (a menu, a driver, a business concept for you...).
- **Extension**: Defines an instance for the model described by the extension point (the navigator view, the driver Z, your business implementation...)

**Eclipse 4 Architecture**

- In 2012, the Eclipse foundation provided a new RCP runtime (E4)
- This new architecture is fully compliant with the Eclipse 3 concepts
- The major part of Eclipse is still written using Eclipse 3

Image 8 Eclipse 3 Architecture
Basically E4 provides;
- an application model
- a UI renderer (SWT or Java Fx)
- Injection
- CSS
- Spies
- A compatibility layer to be compliant with Eclipse 3 concepts

This new runtime is based on OSGi and distributed under EPL license

Image 9 e4 Architecture

**UI Model**

Any user interface contains always the same elements:
views, perspectives, commands:

Image 10 UI anatomy
In Eclipse 4, these elements are defined in application model:

- It is defined using Modeling technologies
- It is a live model
  - read when application starts
  - saved when application ends
- UI Agnostic
- Rendered by a renderer (SWT or JavaFx)
E4 Injection

Eclipse 4 provides a specific injector using standard annotations
➢ @Inject, @PostConstruct, ...

This injector is hierarchical:
➢ the values in context are associated to a level in the UI
➢ a view has its own values and can also get values from parent contexts

This injector is dynamic when a value changes:
➢ an injected field is automatically updated
➢ an injected method is automatically re-invoked on parameter’s value changes

Remote Application Platform (RAP)
➢ RAP is used to launch an RCP application in a web browser.
➢ RAP uses a specific SWT implementation
➢ RAP uses JSON to communicate between client and server
Image 12 RCP and RAP

- RAP allows you to write your application once for desktop and web.
**Modeling Projects overview**

*Modeling Projects*

The power of models for your productivity

**Overview**

This part is about Modeling projects. It will show you how you can increase your software productivity. Modeling projects are based on RCP architecture. They can be used in two ways:

- **MDA**: Model Driven Application: your model is used at runtime
- **MDD**: Model Driven Development: your model is used by your tooling.

---

**Map**

---

**What is EMF?**

- EMF is the core of the modeling project.
- Basic modeling tools
➢ a modeling language (Ecore)
➢ default generators (bean, editor)

Image 14 EMF in the Modeling project
**Modeling steps**

**EMF/Eclipse**

- MO
- M1
- M2
- M3

**Development**

- Air traffic application
- Airport model Domain specific generators
- Airport Metamodel Domain specific code

Image 15 EMF in the development process

**model / meta model**

Model and meta model are distinguished according to domain

Image 16 Model and Meta model

**Tree editor**

- The tree editor is the default editor.
- Properties are edited in the Eclipse's default property view
Image 17 Ecore Tree Editor

**Graphical Editor**
The graphical editor is available in the eclipse modeling delivery:

Image 18 Ecore tools

**Model to Text (M2T)**

How to use ‘model to text’ generators for your productivity
Once your model is defined, you can create code generators
Xtend language

- Xtend is a simplified Java language which is generated into real Java code
- It can also generate text code
- This is a really nice and easy language to learn.
Image 19 Xtend generator sample

**How can you use it?**

Create a meta model of your system:
- satellite description
- system description
- whatever description

Create a model of your business:
- the ABC satellite
- the ABC system

Write your code generator for your ABC system:
- it will be compliant with your meta-model/model
- produce the boilerplate code which is cumbersome to write
  - SQL mapping
  - Web side mapping (php, angular, node.js..)
  - Your framework compliant code mapping
  - ...

Remark: if you describe more than once the same information generate it from a model!
Model Edition

By default, the EMF framework can generate a Tree oriented editor for your model. But it can be painful to use.

You can also create your specific editor using:

- sirius for graphical editors
- Xtext for textual editors

Graphical model editors

- Use the model editors when your model can be graphically displayed
- Then bind your model to your code generator (MDD approach)
- Provide your model editor in your application (MDA approach)

Sirius is a tool to describe graphical representation for a model

- diagram: nodes and links
- table: lines of instances with properties in columns
- matrix: lines and columns of instances with values in cells

It allows to get very quickly a graphical editor for a model

The editor can be used directly without code generation
Sirius overview

Remark: the ecore graphical model editor is written using Sirius

Textual model editors
- Use textual model editors when you want to provide a DSL text editor
- Bind it to your code generator automatically (MDD approach)
- Provide your powerful text editor to your customer (MDA approach)
Xtext

➢ Eclipse project to generate a text editor for a model
➢ ‘Language IDE framework’
➢ Generates the editor from a grammar and an Ecore model

Xtext website: http://www.eclipse.org/Xtext/

How to use it?

➢ Follow the 5 mn and 30 mn tutorial.
➢ It can also be integrated with IntelliJ
➢ If it is relevant, you can bind your code generator
  ➢ Describe once and generate code in different languages

Xtext editors features

All editors generated with Xtext have the following features for free:

➢ syntax highlighting
➢ auto completion
➢ search indexes
➢ validation and quickfixes
➢ errors and warnings management
➢ templates
➢ outline
➢ definition of hyperlink (cross languages)
➢ binding with an optional Xtend generator

3 - http://www.eclipse.org/Xtext/
Remark: using XText is a way to get a ‘Java like’ editor but for your own business language!

Graphic visualization of the grammar

Image 20 XText Syntax Graph
Generated editor

Image 21 Sample editor

**Customize your editor**

With Xtext it is really easy (and fun!) to add your:

- quickfixes (@Fix annotation)
- errors and warnings (@Check annotation)
- auto completion (textual or semantic)
- text formatter
- code generator (triggered on save)
- templates (compliant with your syntax)
Conclusion

The Eclipse technologies can help you to:
- build your own tooling to increase your productivity
- make modular java applications without license constraints
- use for free so many components

How to be involved?
You can:
- Ask questions to the community using forum
- Use the bugzilla to open/find bugs
- Become a committer and enhance your resume
- Attend to Eclipse conferences
- Create your integrated tooling in the most used IDE on the market

How can you learn this?
- By yourself with google and the famous vogella web site
- Buy an Eclipse book
- Or contact me for:
  - A taylor made training
  - Order a consulting mission (on site or remotely).
- Slides of this presentation are available

Thank you
Any questions?