OPCOACH

ECLIPSE TRAINING AND CONSULTING

Singapore Eclipse Meetup

Eclipse Overview

February 28th 2017

OPC 17 MTU PRE ECL 01 A

Table des matières

Introduction

This talk is about :

- Eclipse Foundation
- Eclipse IDE
- OSGi Architecture
- Eclipse RCP Architecture
- Modeling projects

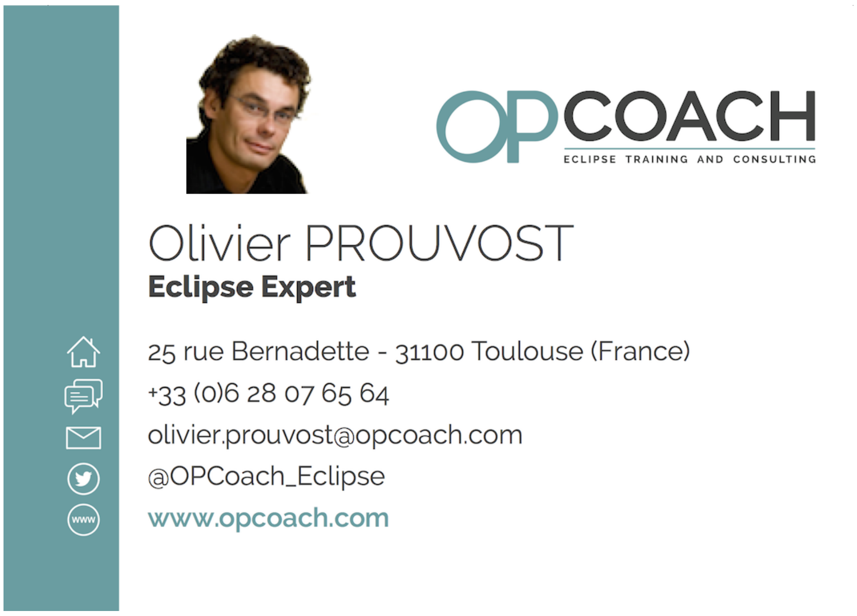
The goals of the talk are :

- to give you information about Eclipse projects
- to give you ideas to enhance your software development factory
- to start discussion around these technologies

Presentation

I

OPCoach



OPCOACH
ECLIPSE TRAINING AND CONSULTING

Olivier PROUVOST
Eclipse Expert

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**
- Web site : *http://www.opcoach.com/en*[1]
- Twitter : **@OPCoach_Eclipse**

1 - http://www.opcoach.com/en

Eclipse Eco system

II

Eclipse Eco System

- Foundation and goal
- Some figures
- Different groups

Eclipse Foundation

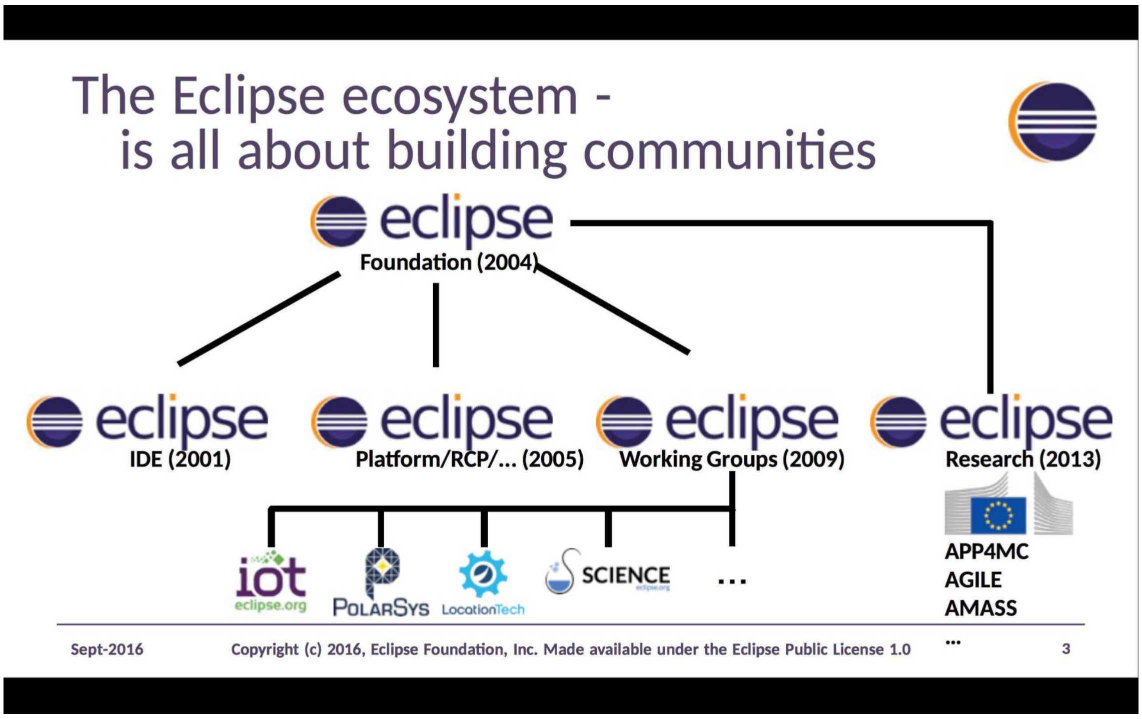
Goals :

- 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
eclipse Foundation (2004)
eclipse IDE (2001)
eclipse Platform/RCP/... (2005)
eclipse Working Groups (2009)
eclipse Research (2013)
iot eclipse.org
PolarSys
LocationTech
SCIENCE
...
APP4MC
AGILE
AMASS
...
Sept-2016
Copyright (c) 2016, Eclipse Foundation, Inc. Made available under the Eclipse Public License 1.0
3

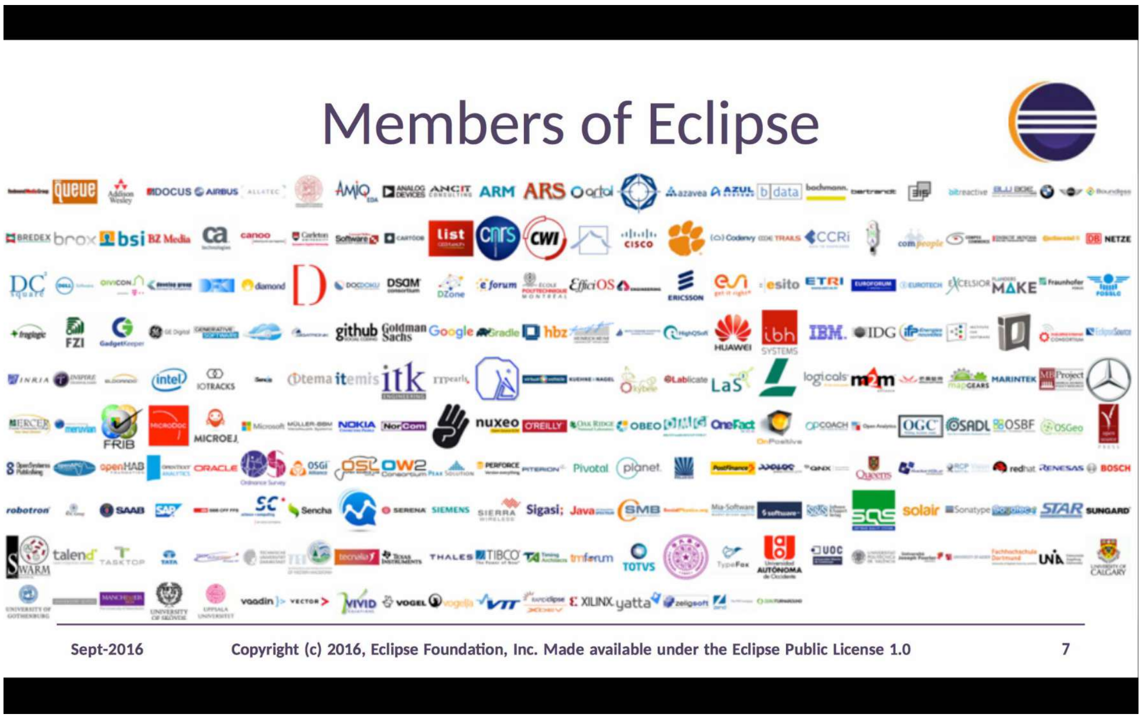
Some figures

- 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"**

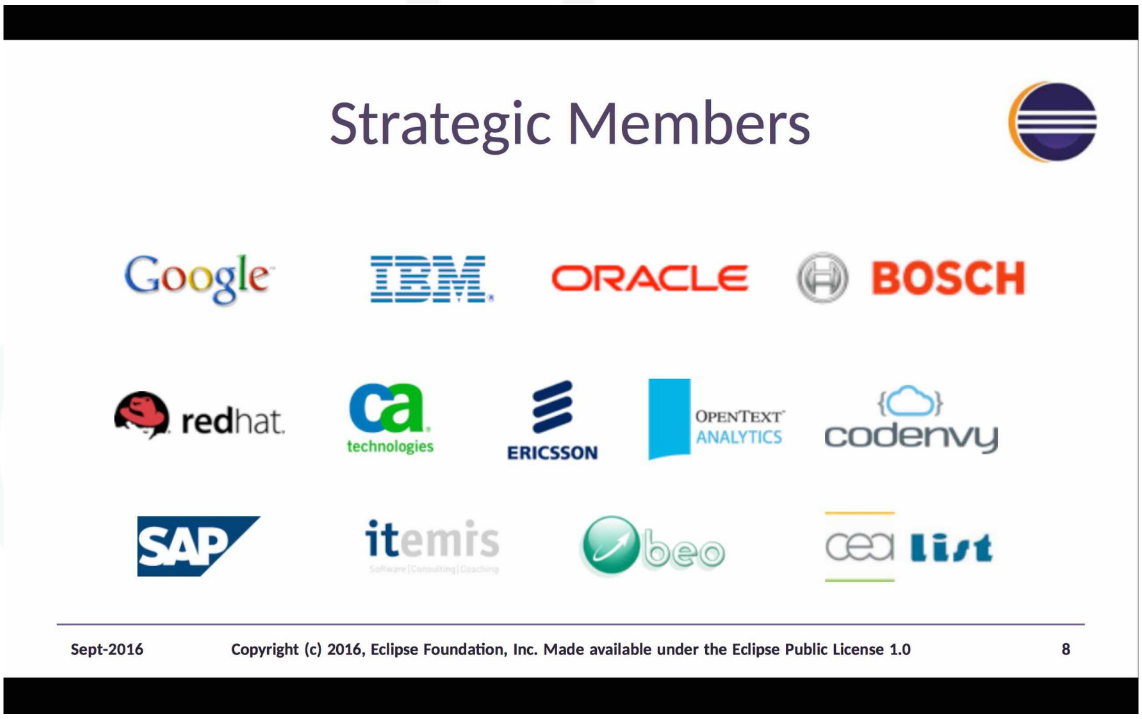
Members



Members of Eclipse

Sept-2016 Copyright (c) 2016, Eclipse Foundation, Inc. Made available under the Eclipse Public License 1.0 7

Strategic Members



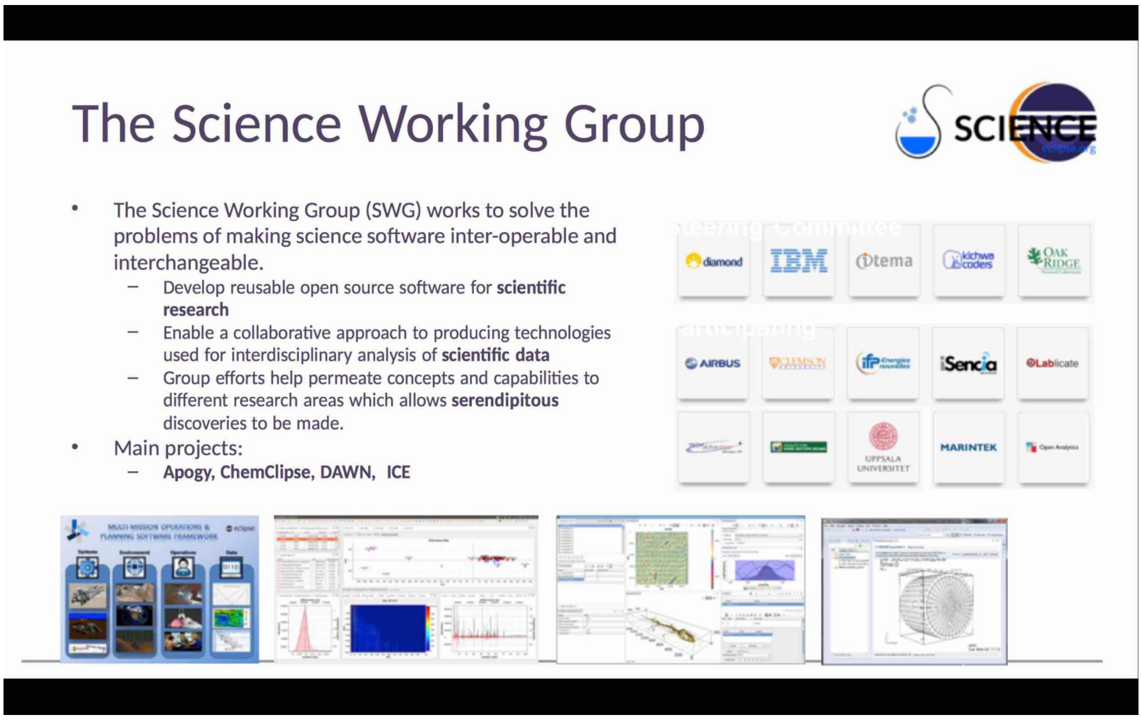
Strategic Members

Sept-2016 Copyright (c) 2016, Eclipse Foundation, Inc. Made available under the Eclipse Public License 1.0 8

Eclipse groups

Eclipse foundation leads several technical groups

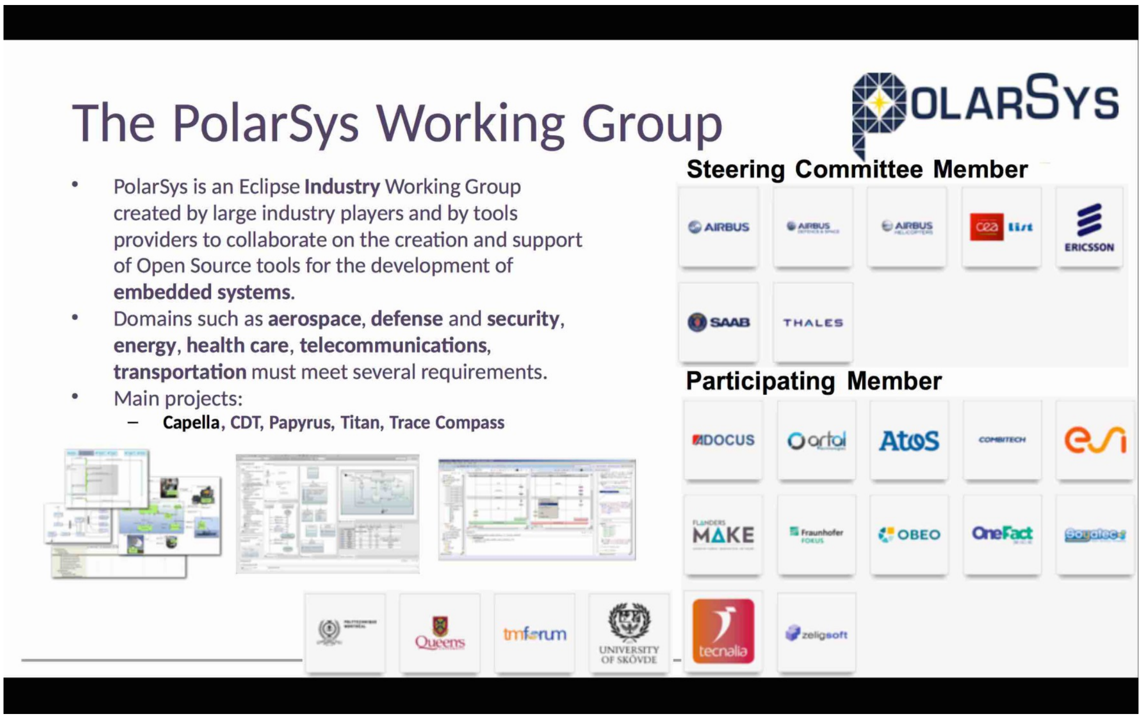
Science Group



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**

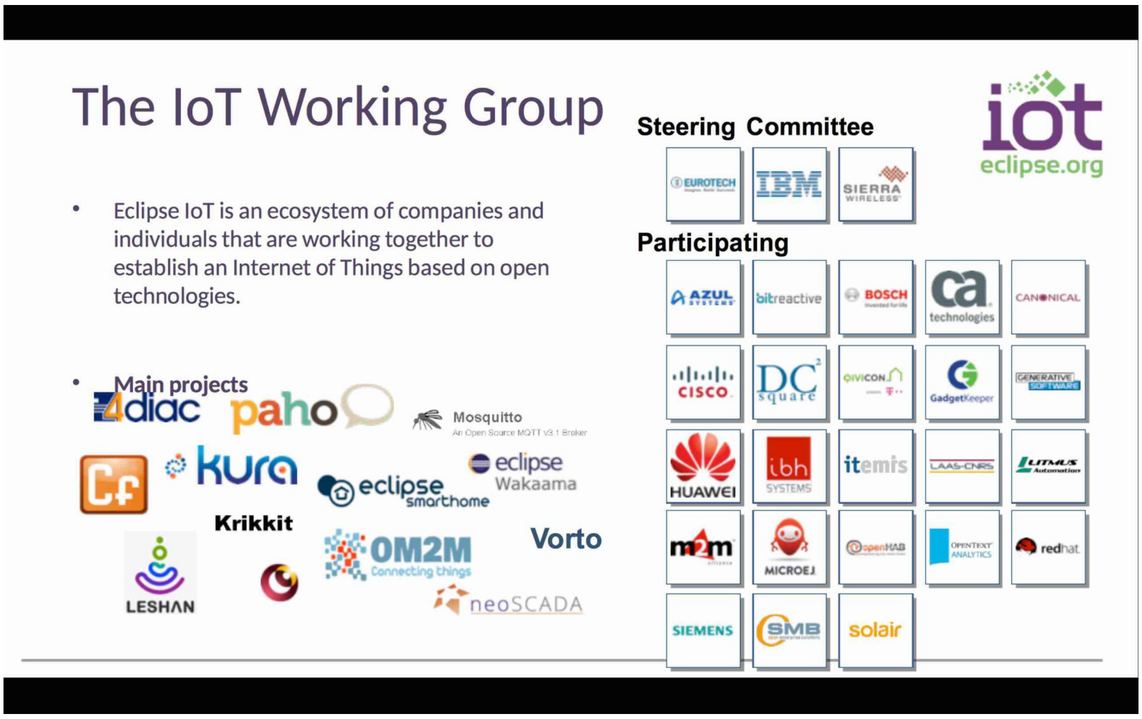
PolarSys Group



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**

IOT Group

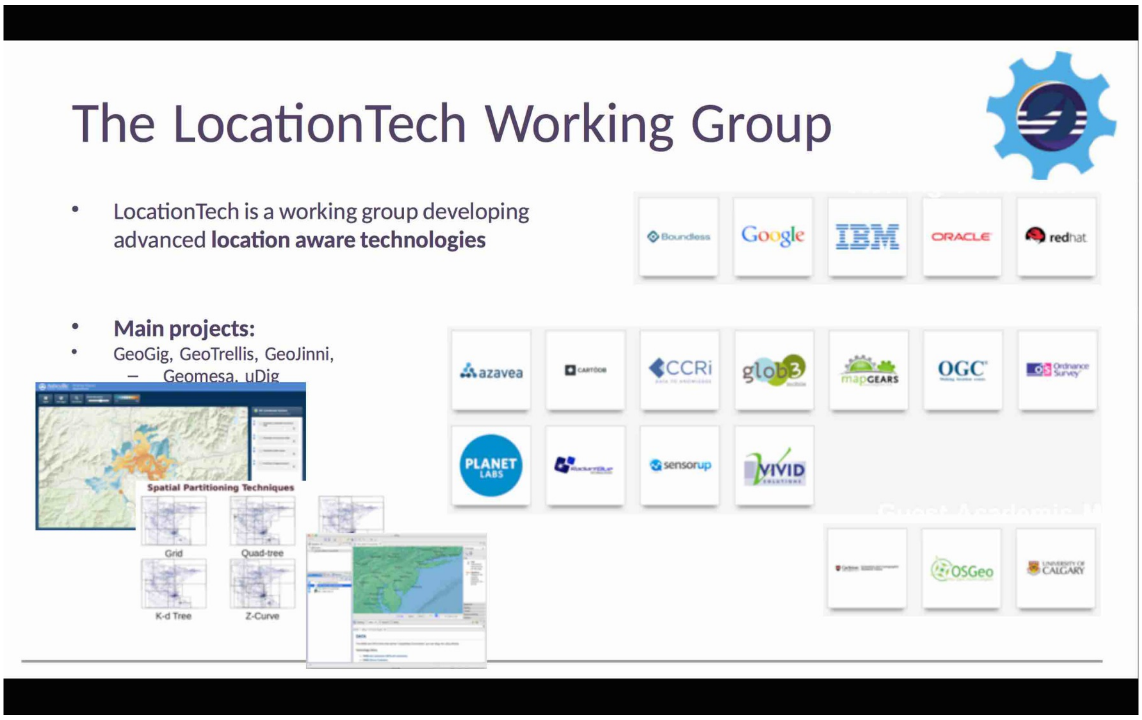


The IoT Working Group

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

4diac
paho
Mosquitto
An Open Source MQTT v3.1 Broker
Cf
kura
eclipse smarthome
eclipse Wakaama
Krikkit
OM2M Connecting things
Vorto
LESHAN
neoSCADA
Steering Committee
EUROTECH
IBM
SIERRA WIRELESS
Participating
AZUL SYSTEMS
bitreactive
BOSCH
ca technologies
CANONICAL
CISCO
DC square
OIVICON
GadgetKeeper
GENERATIVE SOFTWARE
HUAWEI
ibh SYSTEMS
itemis
LAAS-CNRS
LITMUS Automation
m2m alliance
MICROEJ
openHAB
OPENTEXT ANALYTICS
redhat
SIEMENS
SMB
solair
iot eclipse.org

Location Tech Group



The LocationTech Working Group

- LocationTech is a working group developing advanced **location aware technologies**
- **Main projects:**
- GeoGig, GeoTrellis, GeoJinni,
  - Geomesa, uDig

Eclipse IDE

III

The different versions

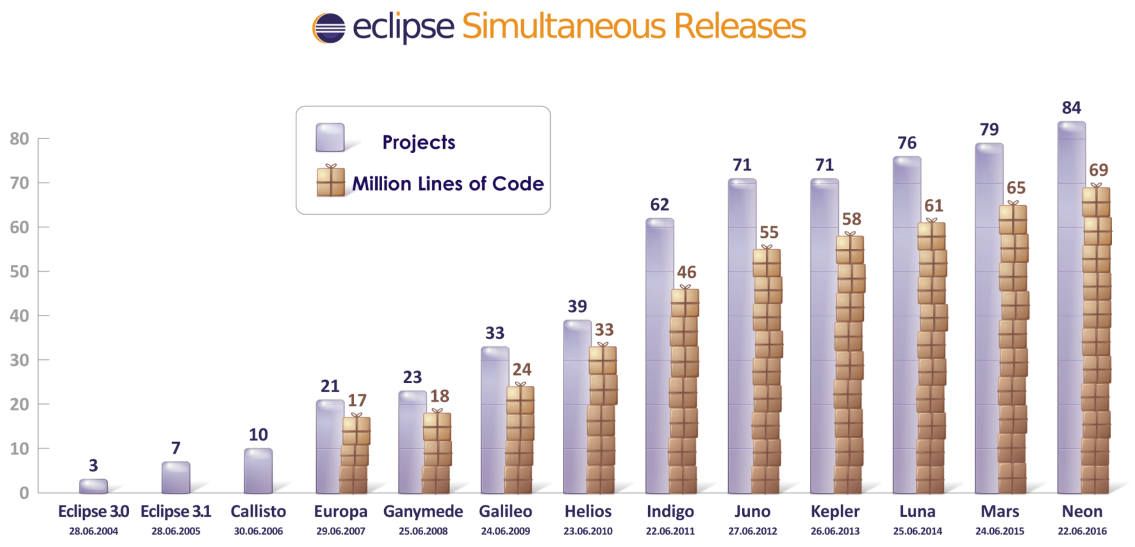


Image 1

Strengths

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

OSGi

IV

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

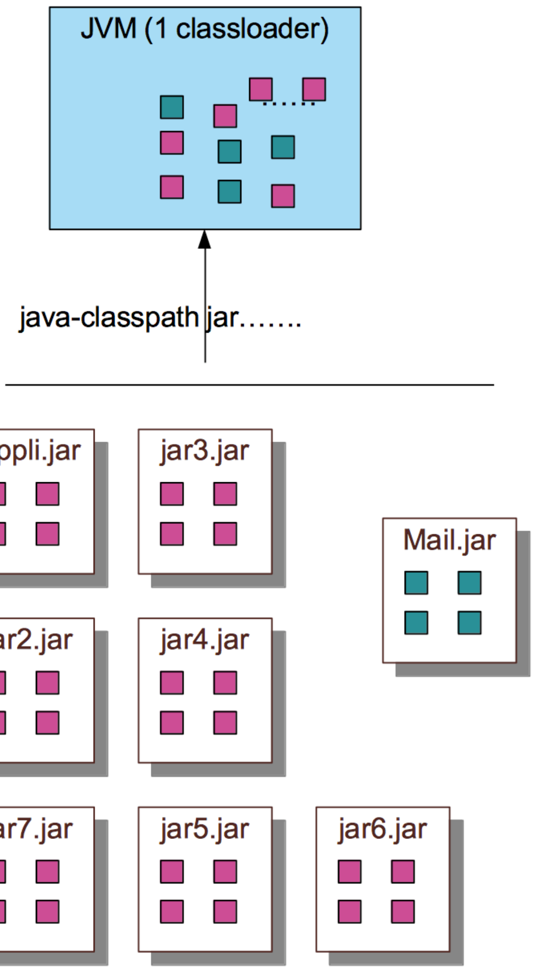
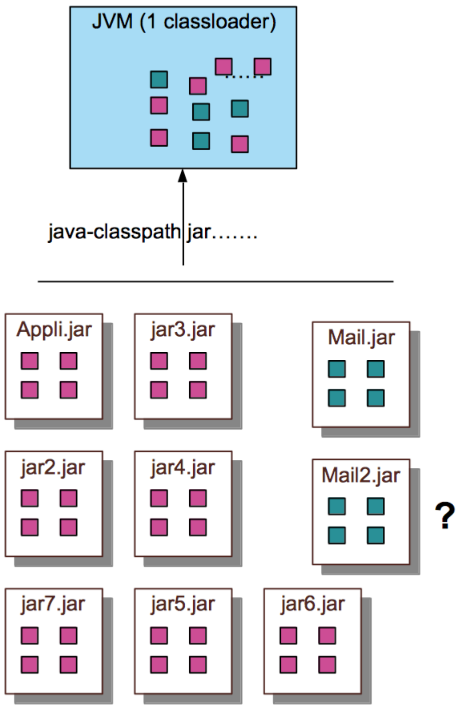


Image 2

Updating the application

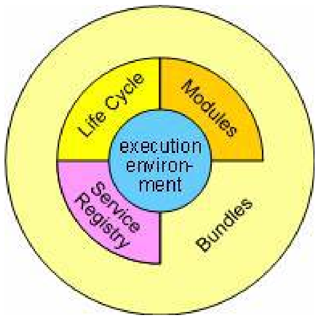
- The mail.jar library can not send attached files
- Another component needs to send attached files
- Should we update all dependencies ?



- 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 bundle activation
- OSGi manages services (Service Registry)



OSGi Framework

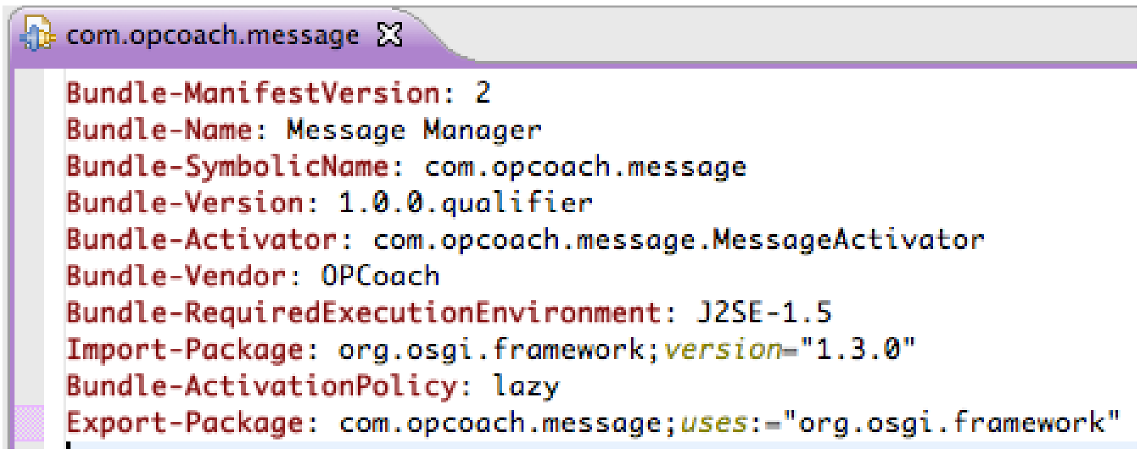
OSGI defines a model to manage java software modules
OSGi is a specification managed by the OSGi Alliance : *http://www.osgi.org*[2]

2 - http://www.osgi.org

It has been started in 1999 for Java embedded devices

The bundle

It is defined by a MANIFEST:



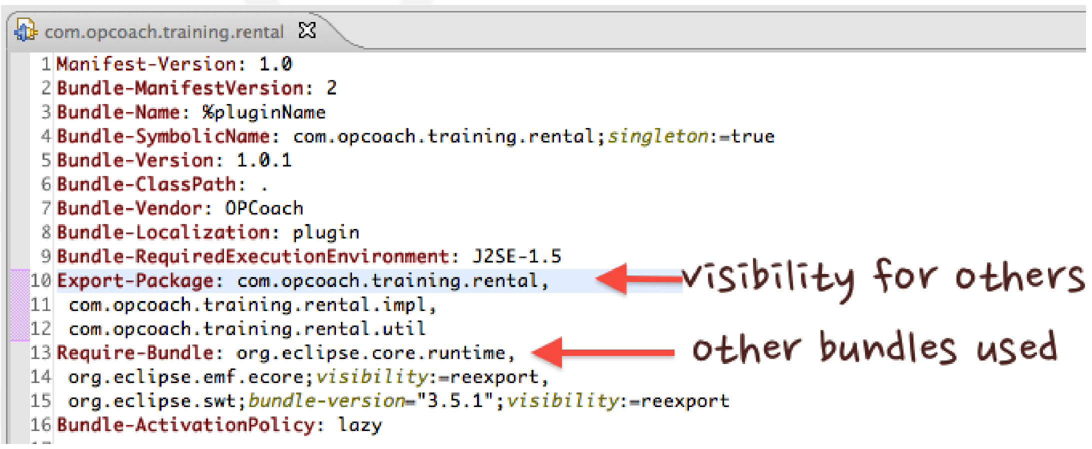
```
Bundle-ManifestVersion: 2
Bundle-Name: Message Manager
Bundle-SymbolicName: com.opcoach.message
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: com.opcoach.message.MessageActivator
Bundle-Vendor: OPCoach
Bundle-RequiredExecutionEnvironment: J2SE-1.5
Import-Package: org.osgi.framework;version="1.3.0"
Bundle-ActivationPolicy: lazy
Export-Package: com.opcoach.message;uses:="org.osgi.framework"
```

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
  - optionaly
  - depending on version numbers
- The bundle exports the packages it wants to make visible
- **The classpath is computed automaticaly** -> No more jar hell !



```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %pluginName
Bundle-SymbolicName: com.opcoach.training.rental;singleton:=true
Bundle-Version: 1.0.1
Bundle-ClassPath: .
Bundle-Vendor: OPCoach
Bundle-Localization: plugin
Bundle-RequiredExecutionEnvironment: J2SE-1.5
Export-Package: com.opcoach.training.rental,
 com.opcoach.training.rental.impl,
 com.opcoach.training.rental.util
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.emf.ecore;visibility:=reexport,
 org.eclipse.swt;bundle-version="3.5.1";visibility:=reexport
Bundle-ActivationPolicy: lazy
```

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**

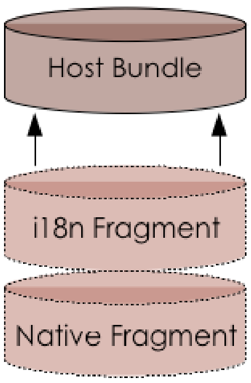


Image 5 Fragment

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

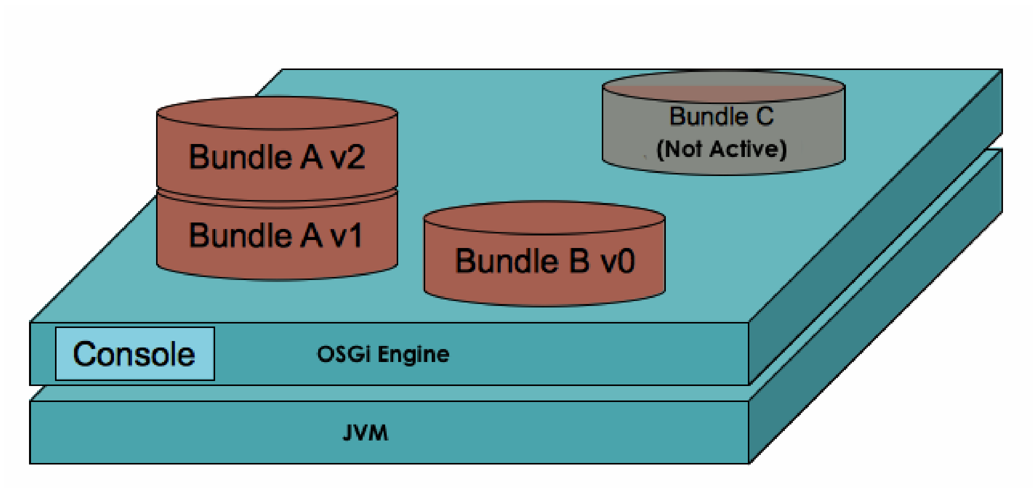


Image 6 OSGi Engine

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
  - automaticaly created with the **@Component** class annotation

Sample OSGi Architecture

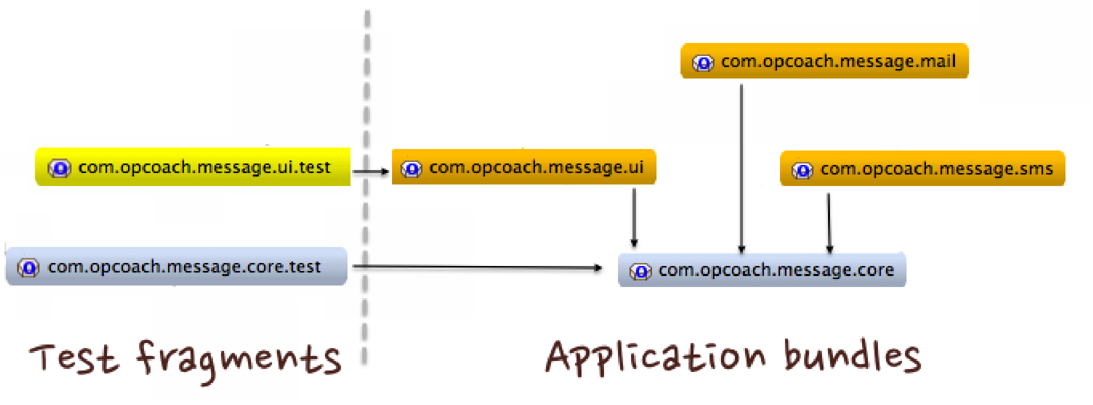


Image 7 OSGi architecture

Eclipse Architecture

V

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

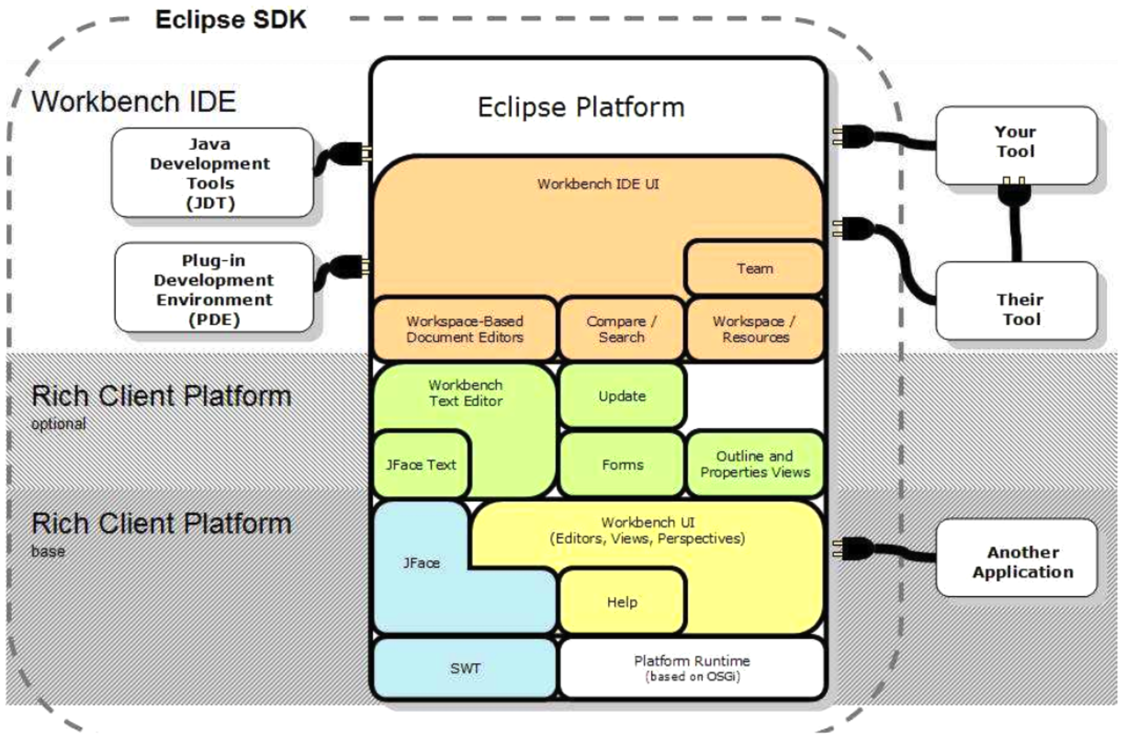


Image 8 Eclipse 3 Architecture

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

- 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

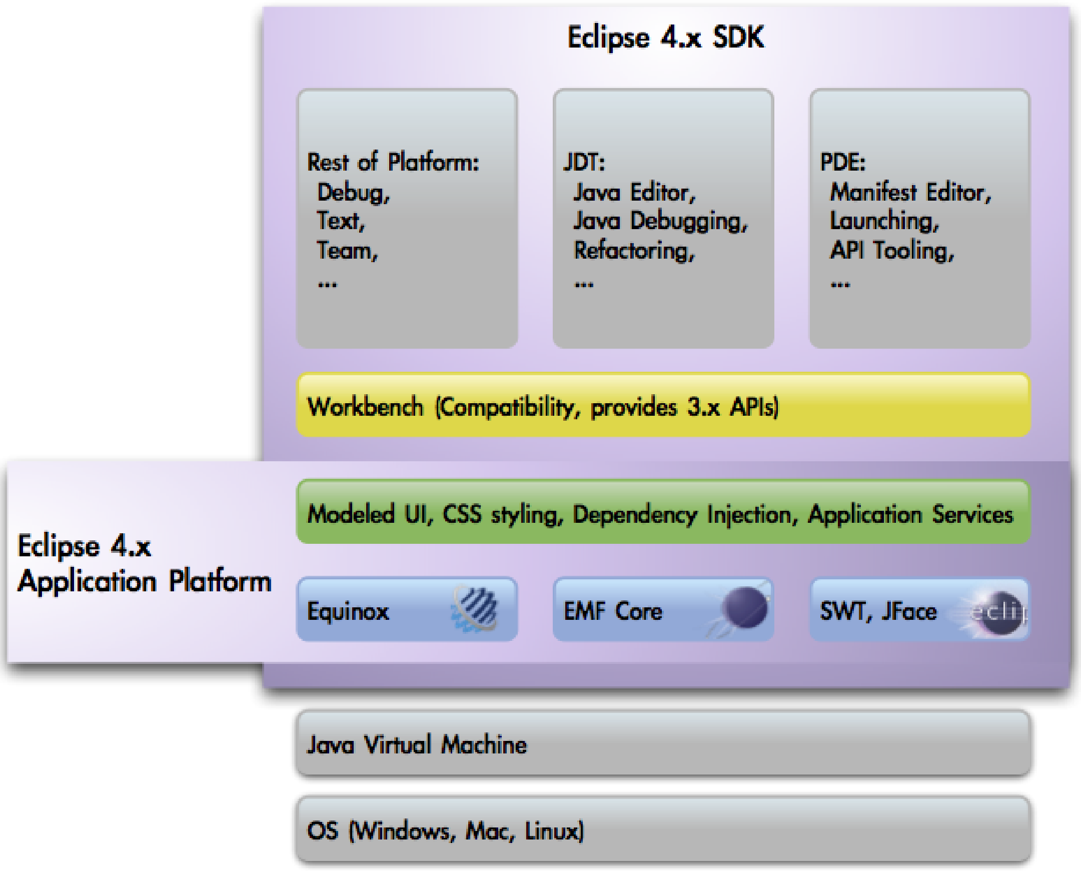
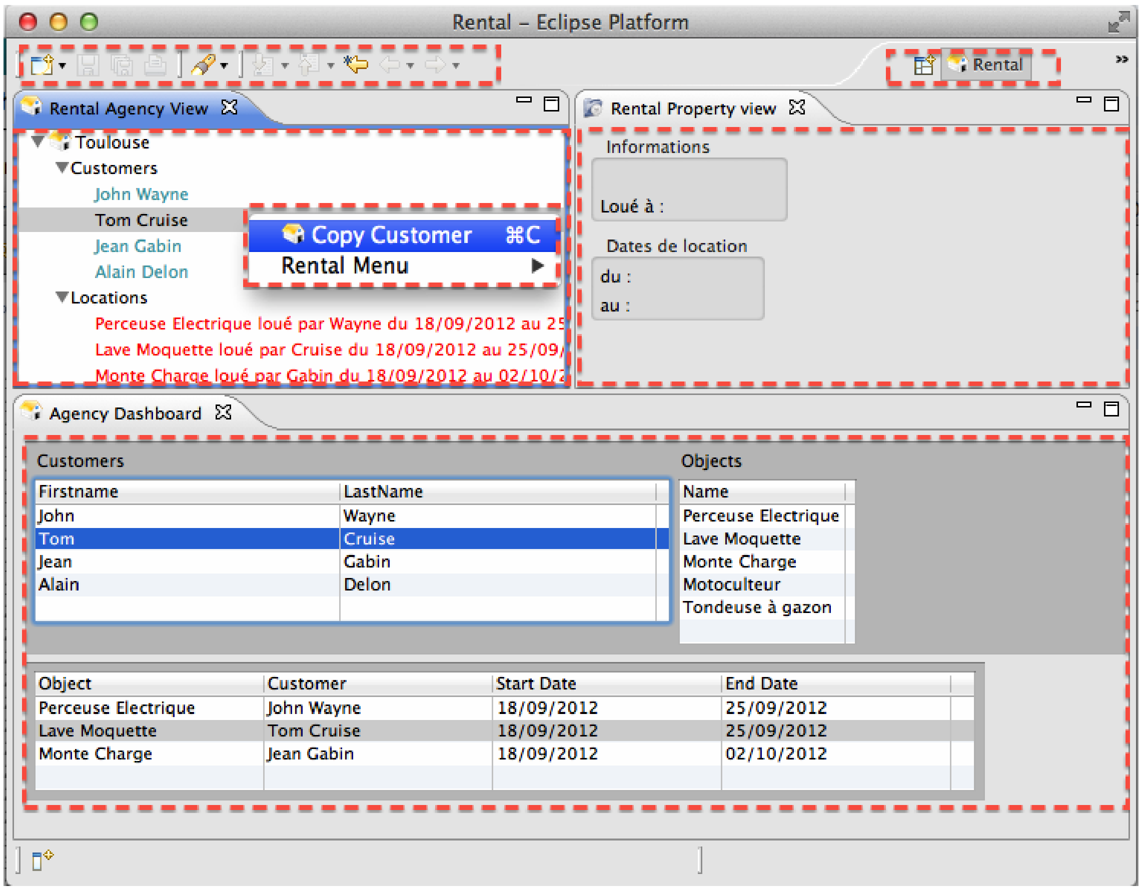

Eclipse 4.x SDK
Rest of Platform: Debug, Text, Team, ...
JDT: Java Editor, Java Debugging, Refactoring, ...
PDE: Manifest Editor, Launching, API Tooling, ...
Workbench (Compatibility, provides 3.x APIs)
Eclipse 4.x Application Platform
Modeled UI, CSS styling, Dependency Injection, Application Services
Equinox
EMF Core
SWT, JFace
Java Virtual Machine
OS (Windows, Mac, Linux)

Image 9 e4 Architecture

UI Model

Any user interface contains always the same elements:

views, perspectives, commands :



Rental – Eclipse Platform

Rental

Rental Agency View

- Toulouse
  - Customers
    - John Wayne
    - Tom Cruise
    - Jean Gabin
    - Alain Delon
  - Locations
    - Perceuse Electrique loué par Wayne du 18/09/2012 au 25
    - Lave Moquette loué par Cruise du 18/09/2012 au 25/09/
    - Monte Charge loué par Gabin du 18/09/2012 au 02/10/2

Rental Property view

Informations

Loué à :

Dates de location

du :

au :

Agency Dashboard

Customers

Firstname	LastName
John	Wayne
Tom	Cruise
Jean	Gabin
Alain	Delon

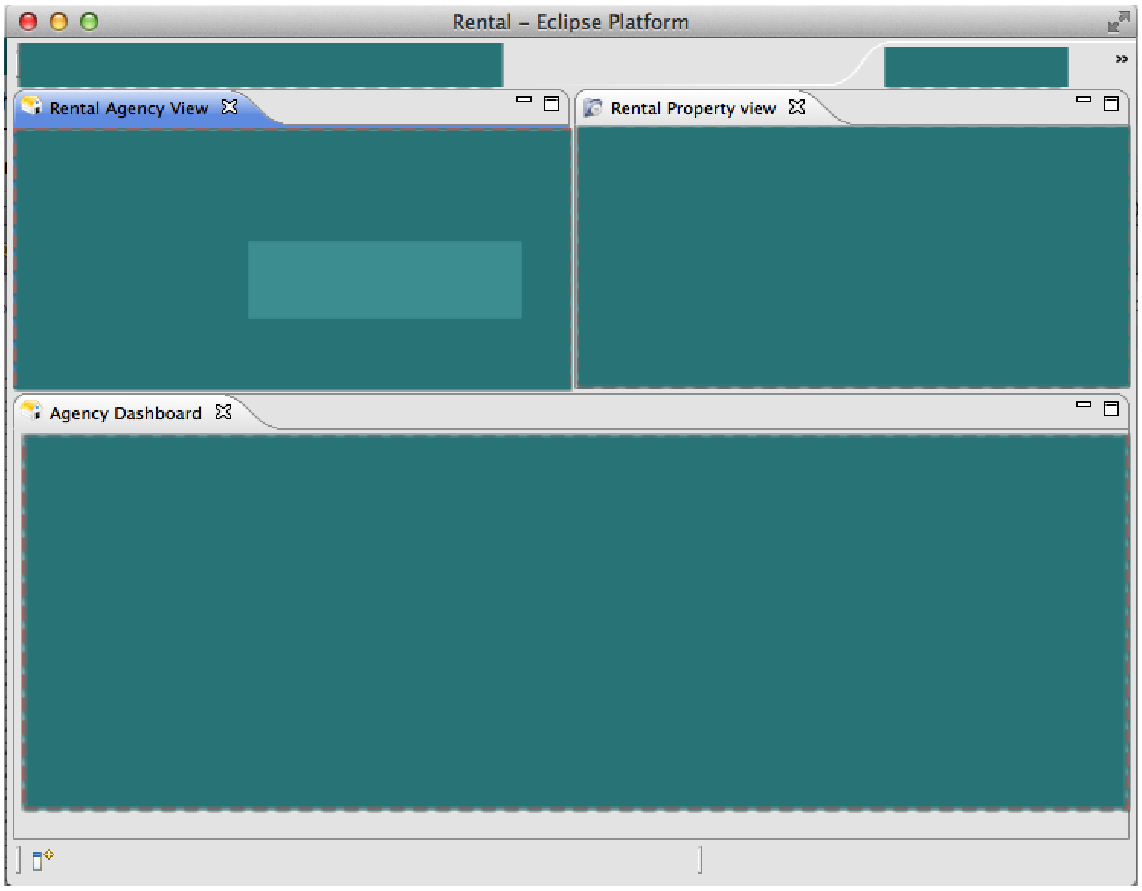
Objects

Name
Perceuse Electrique
Lave Moquette
Monte Charge
Motoculteur
Tondeuse à gazon

Object	Customer	Start Date	End Date
Perceuse Electrique	John Wayne	18/09/2012	25/09/2012
Lave Moquette	Tom Cruise	18/09/2012	25/09/2012
Monte Charge	Jean Gabin	18/09/2012	02/10/2012

Image 10 UI anatomy

In Eclipse 4, these elements are defined in application model :



UI in application model

The 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)

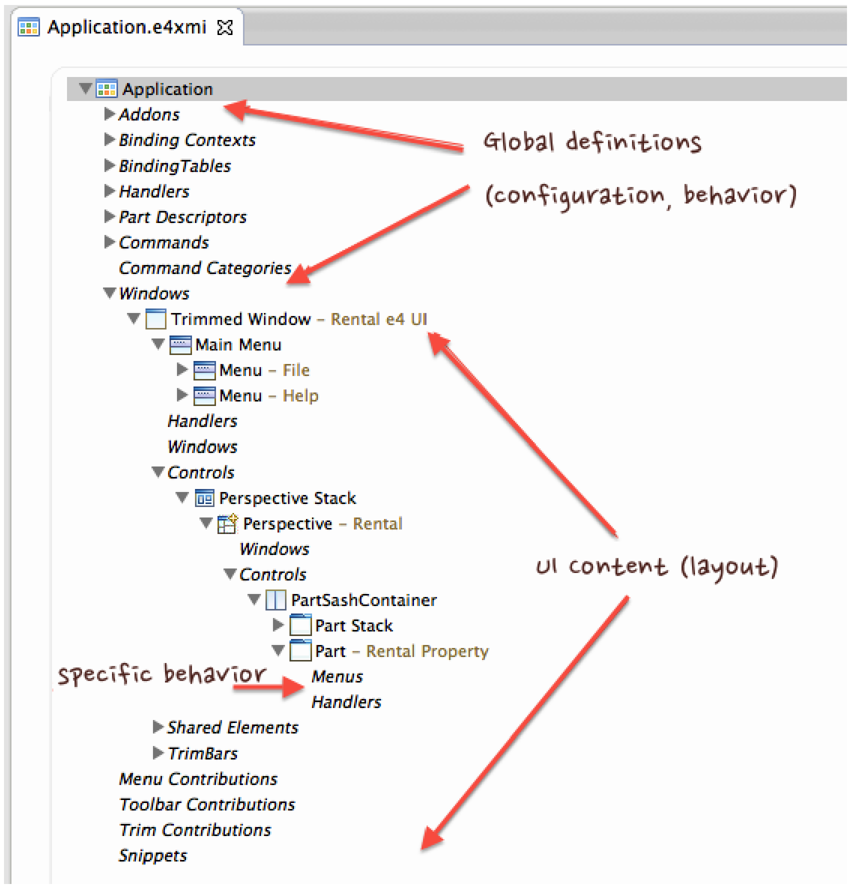


Image 11 Application Model

E4 Injection

Eclipse 4 provides a specific injector using standard annotations

- @Inject, @PostContruct, ...

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 automaticaly updated
- an injected method is automaticaly 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

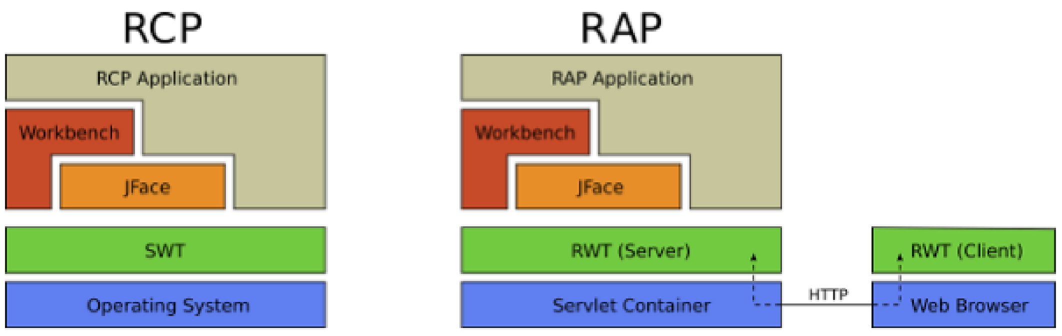


Image 12 RCP and RAP

- RAP allows you to write your application once for desktop and web.

Modeling Projects overview

VI

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 project is 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

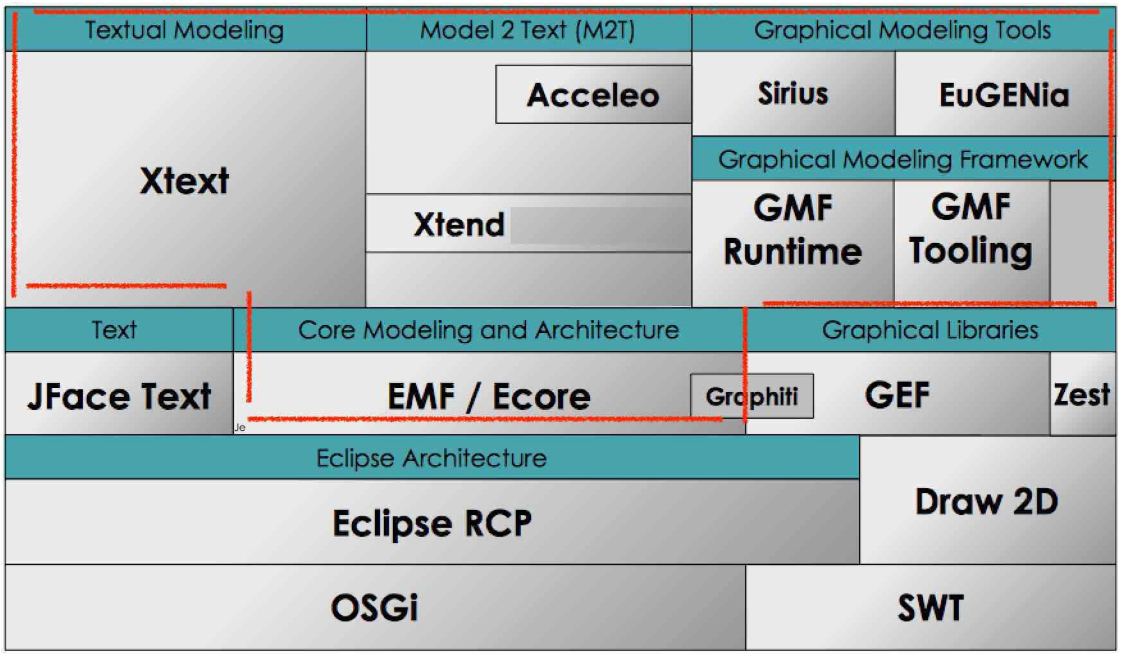


Image 13

What is EMF?

- EMF is the core of the modeling project.
- Basic modeling tools

- a modeling language (Ecore)
- default generators (bean, editor)

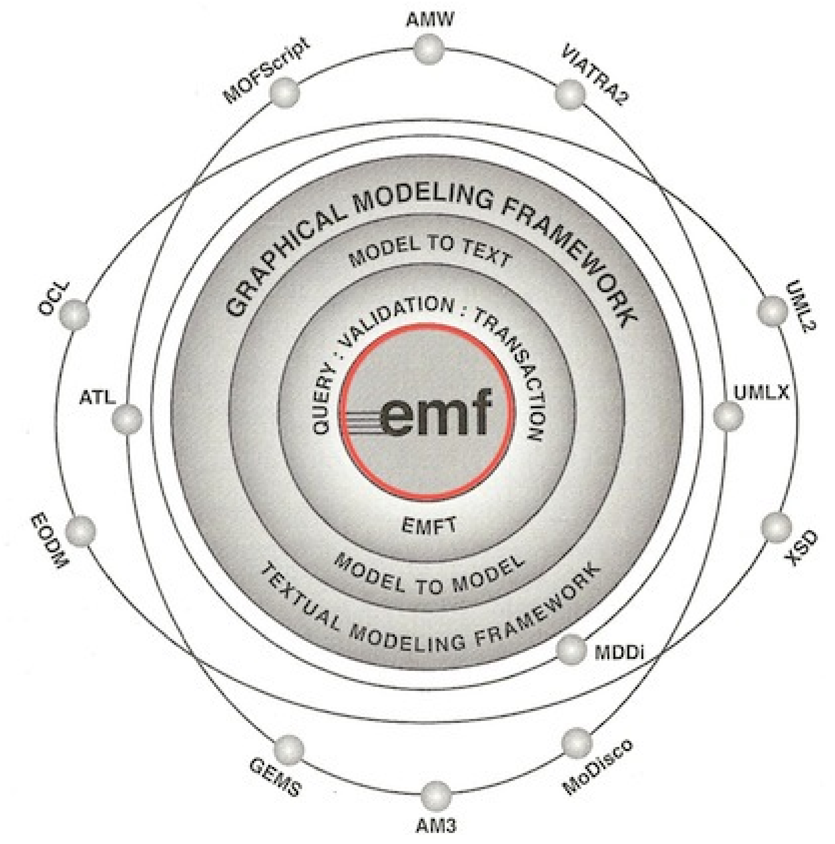


Image 14 EMF in the Modeling project

Modeling steps

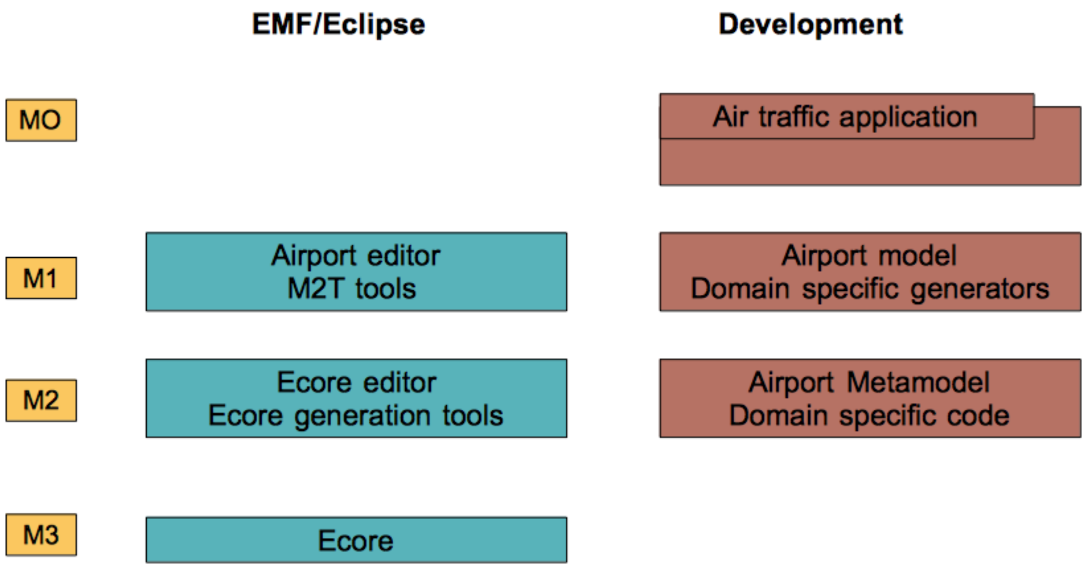


Image 15 EMF in the development process

model / meta model

Model and meta model are distinguished according to domain

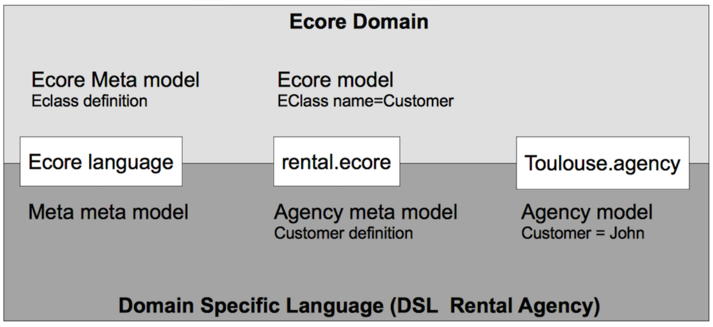
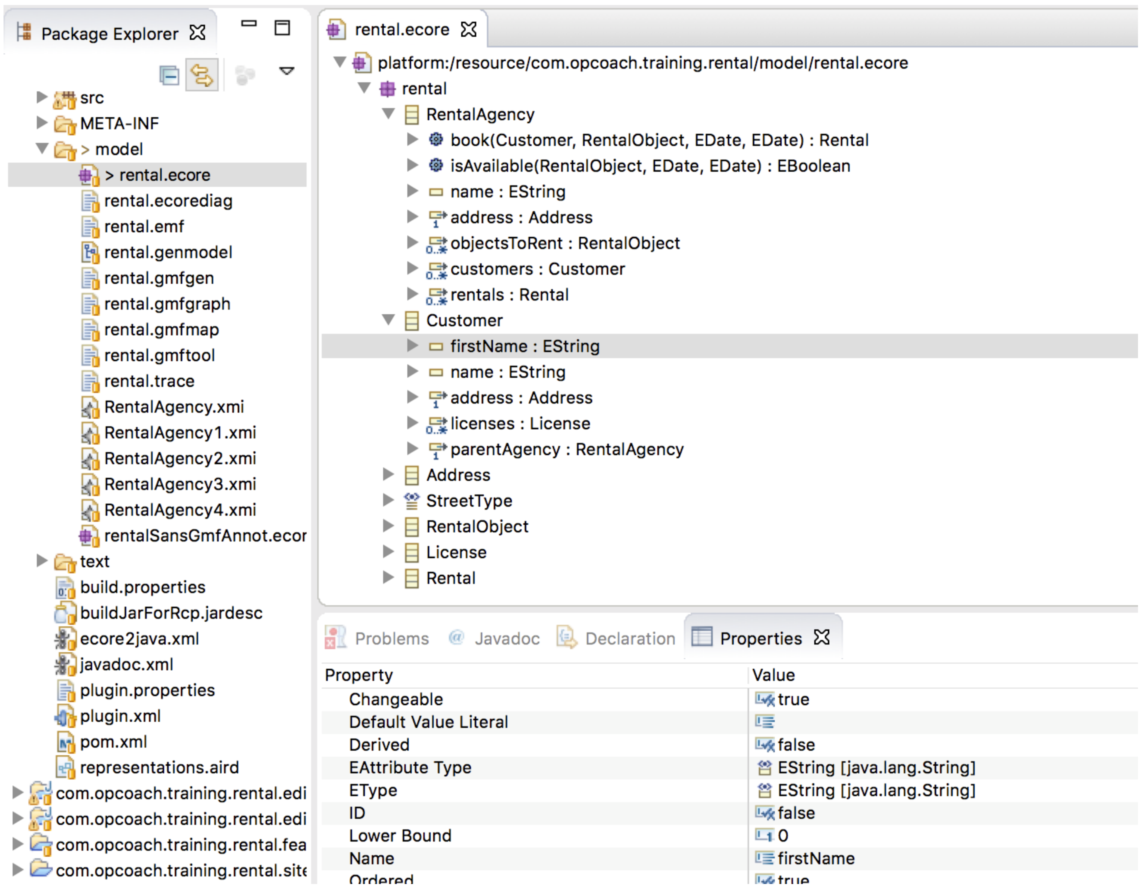


Image 16 Model and Meta model

Tree editor

- The tree editor is the default editor.
- Properties are edited in the Eclipse's default property view


Package Explorer

- src
- META-INF
- > model
  - > rental.ecore
  - rental.ecorediag
  - rental.emf
  - rental.genmodel
  - rental.gmfgen
  - rental.gmfgraph
  - rental.gmfmap
  - rental.gmftool
  - rental.trace
  - RentalAgency.xmi
  - RentalAgency1.xmi
  - RentalAgency2.xmi
  - RentalAgency3.xmi
  - RentalAgency4.xmi
  - rentalSansGmfAnnot.ecor
- text
- build.properties
- buildJarForRcp.jardesc
- ecore2java.xml
- javadoc.xml
- plugin.properties
- plugin.xml
- pom.xml
- representations.aird
- com.opcoach.training.rental.edi
- com.opcoach.training.rental.edi
- com.opcoach.training.rental.fea
- com.opcoach.training.rental.site

rental.ecore

- platform:/resource/com.opcoach.training.rental/model/rental.ecore
  - rental
    - RentalAgency
      - book(Customer, RentalObject, EDate, EDate) : Rental
      - isAvailable(RentalObject, EDate, EDate) : EBoolean
      - name : EString
      - address : Address
      - objectsToRent : RentalObject
      - customers : Customer
      - rentals : Rental
    - Customer
      - firstName : EString
      - name : EString
      - address : Address
      - licenses : License
      - parentAgency : RentalAgency
    - Address
    - StreetType
    - RentalObject
    - License
    - Rental

Problems | Javadoc | Declaration | Properties

Property	Value
Changeable	true
Default Value Literal	
Derived	false
EAttribute Type	EString [java.lang.String]
EType	EString [java.lang.String]
ID	false
Lower Bound	0
Name	firstName
Ordered	true

Image 17 Ecore Tree Editor

Graphical Editor

- The graphical editor is available in the eclipse modeling delivery :

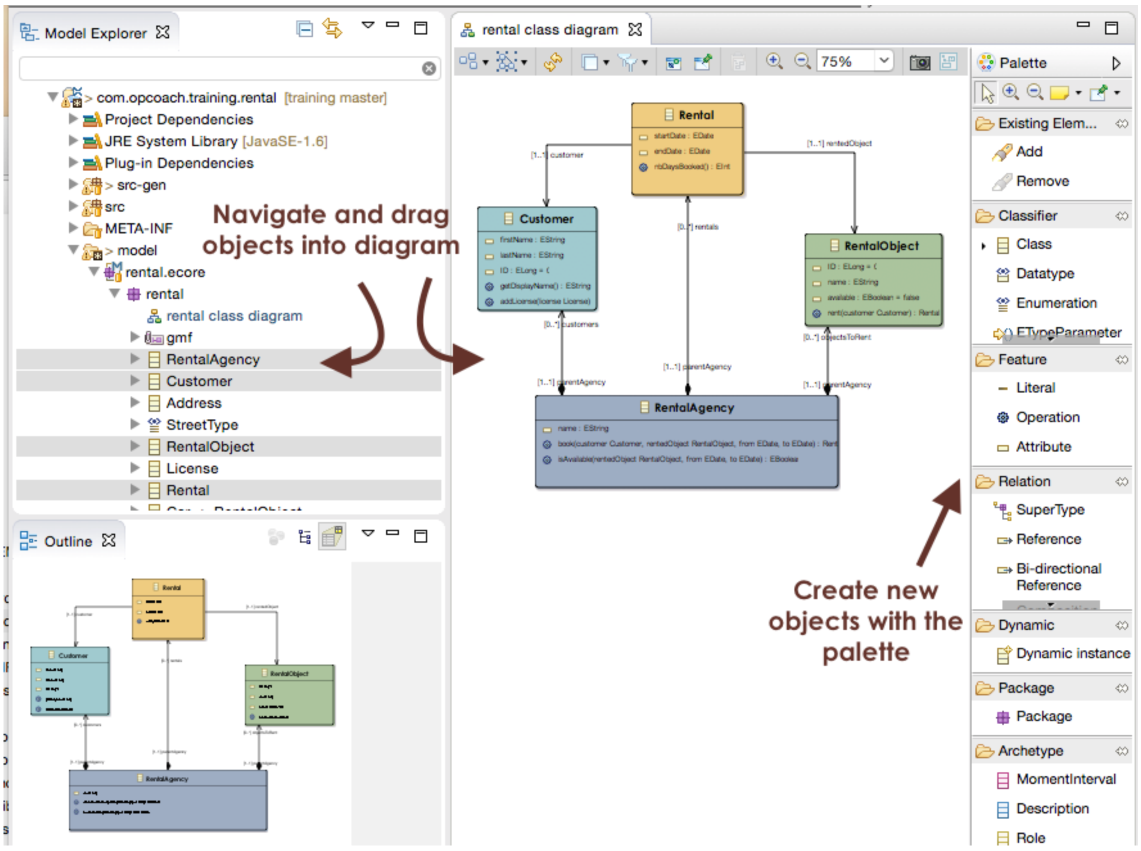
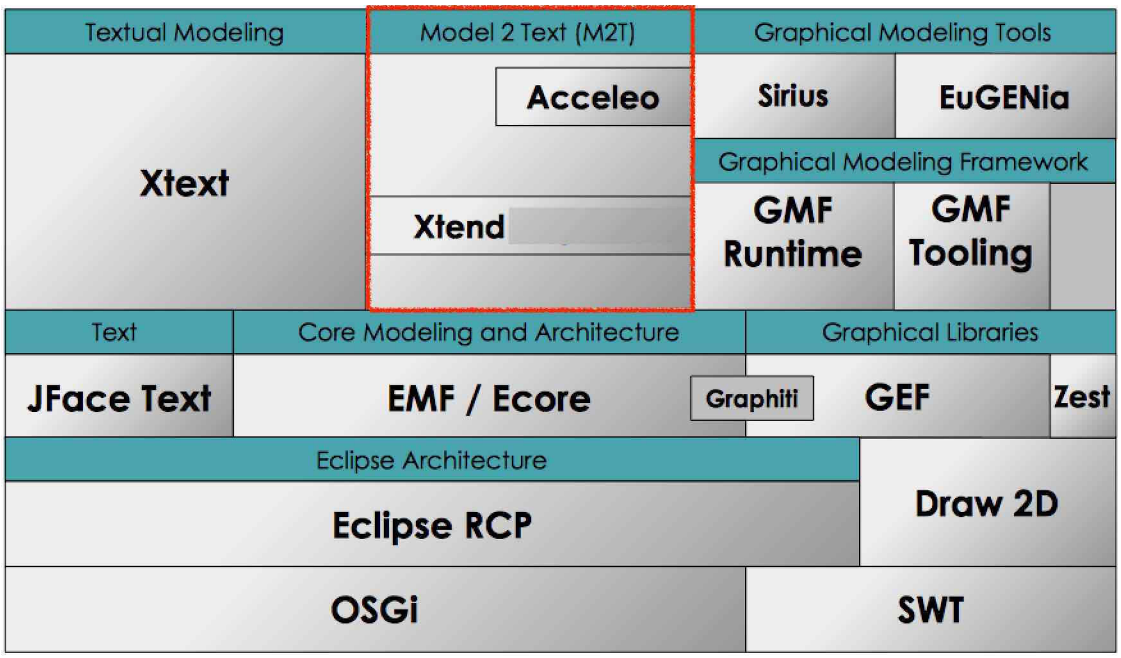


Image 18 Ecore tools

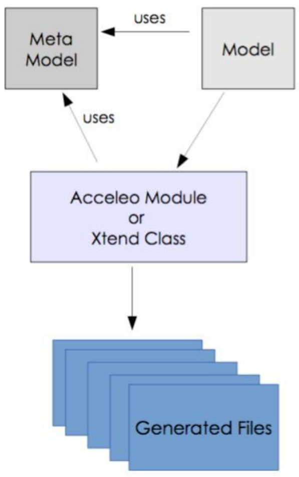
Model to Text (M2T)

How to use 'model to text' generators for your productivity

Once your model is defined, you can create code generators

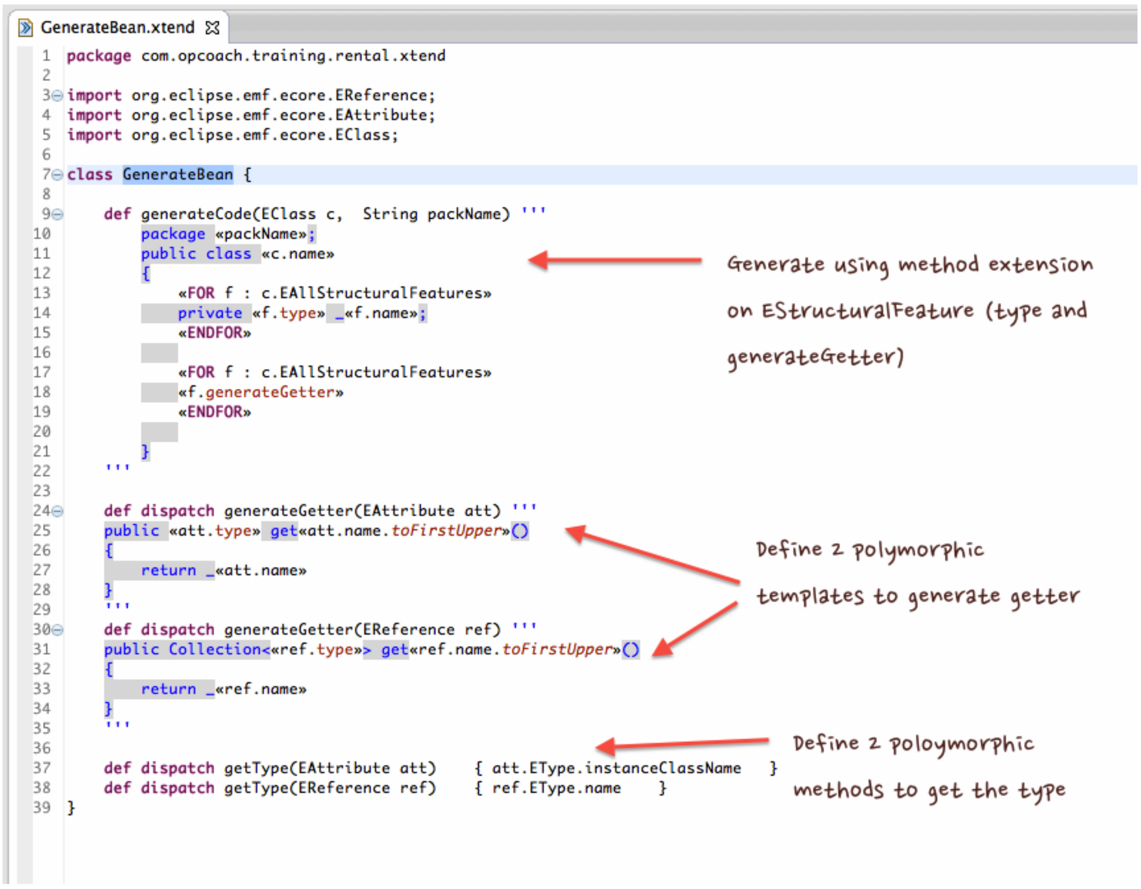


M2T Big picture



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.


GenerateBean.xtend

```
package com.opcoach.training.rental.xtend

import org.eclipse.emf.ecore.EReference;
import org.eclipse.emf.ecore.EAttribute;
import org.eclipse.emf.ecore.EClass;

class GenerateBean {

    def generateCode(EClass c,  String packName) '''
        package «packName»;
        public class «c.name»
        {
            «FOR f : c.EAllStructuralFeatures»
            private «f.type» _«f.name»;
            «ENDFOR»

            «FOR f : c.EAllStructuralFeatures»
            «f.generateGetter»
            «ENDFOR»

        }
    '''

    def dispatch generateGetter(EAttribute att) '''
    public «att.type» get«att.name.toFirstUpper»()
    {
        return _«att.name»
    }
    '''
    def dispatch generateGetter(EReference ref) '''
    public Collection<«ref.type»> get«ref.name.toFirstUpper»()
    {
        return _«ref.name»
    }
    '''

    def dispatch getType(EAttribute att)    { att.EType.instanceClassName   }
    def dispatch getType(EReference ref)    { ref.EType.name     }
}
```

Generate using method extension on EStructuralFeature (type and generateGetter)

Define 2 polymorphic templates to generate getter

Define 2 poloymorphic methods to get the type

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 !

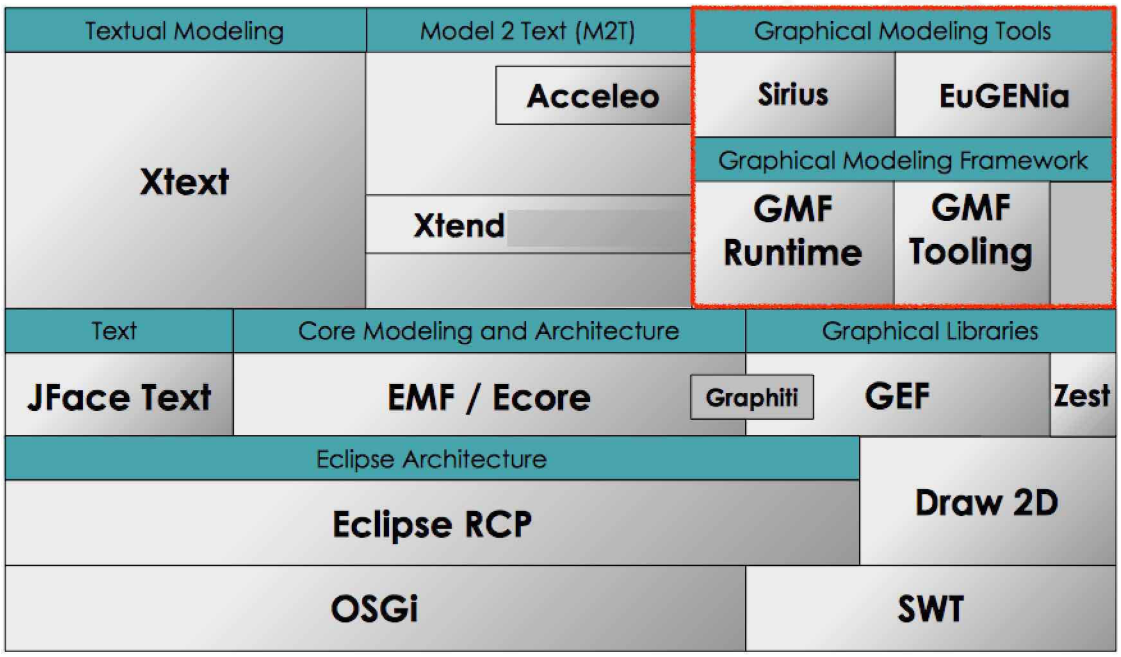
Model Edition

By default, the EMF framework can generates a Tree oriented editor for your model.
But it can be painful to use
You can also create your specific editor using:

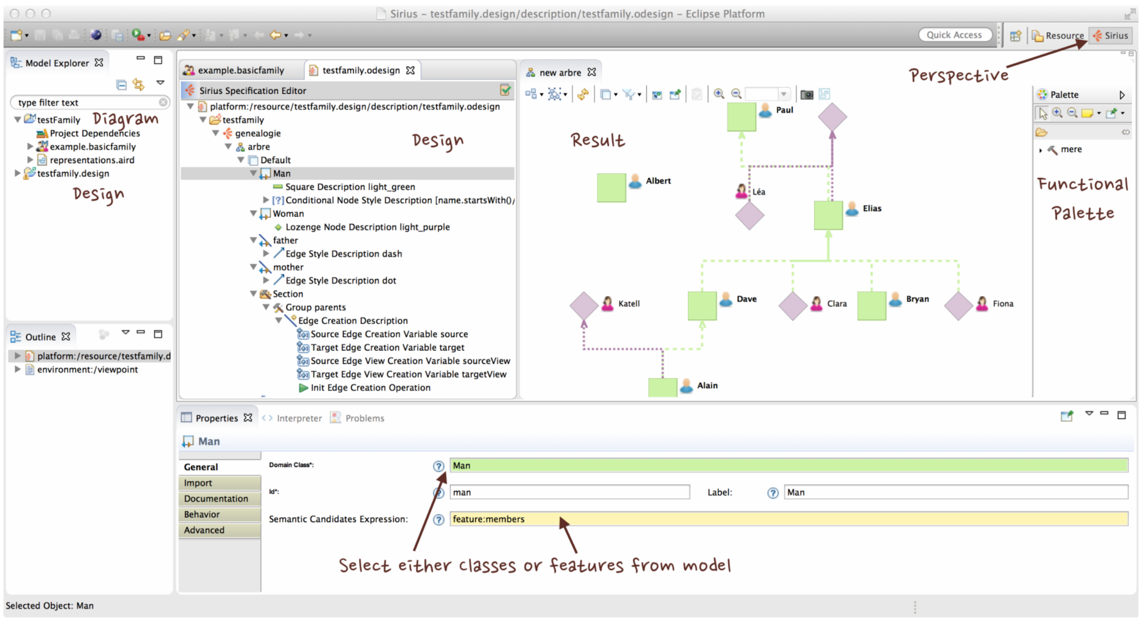
- sirius for graphical editors
- Xtext for textual editors

Graphical model editors

- Use the model editors when your model can be graphicaly 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**



- testFamily
  - Project Dependencies
  - example.basicfamily
  - representations.aird
- testfamily.design

- platform:/resource/testfamily.d
- environment:/viewpoint

- platform:/resource/testfamily.design/description/testfamily.odesign
  - testfamily
    - genealogie
      - arbre
        - Default
          - Man
            - Square Description light_green
            - [?] Conditional Node Style Description [name.startsWith()
          - Woman
            - Lozenge Node Description light_purple
          - father
            - Edge Style Description dash
          - mother
            - Edge Style Description dot
          - Section
            - Group parents
              - Edge Creation Description
                - Source Edge Creation Variable source
                - Target Edge Creation Variable target
                - Source Edge View Creation Variable sourceView
                - Target Edge View Creation Variable targetView
                - Init Edge Creation Operation

Man

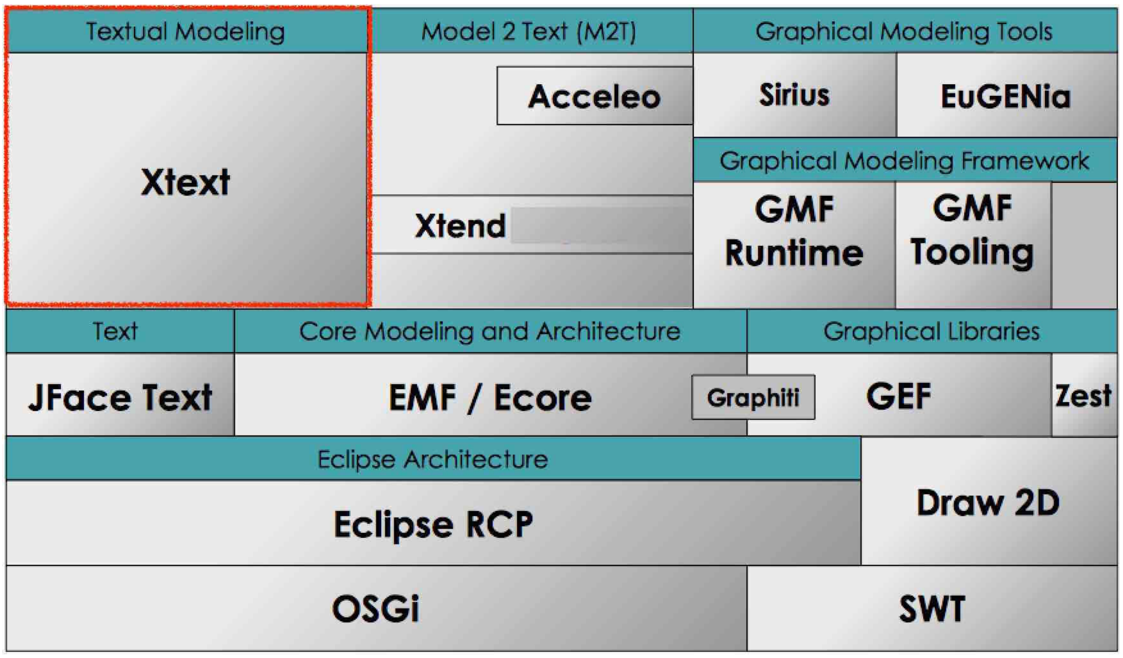
General	Domain Class*:	Man	
Import	Id*:	man	Label: Man
Documentation	Semantic Candidates Expression:	feature:members	
Behavior			
Advanced			

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 automaticaly (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/* [3]

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

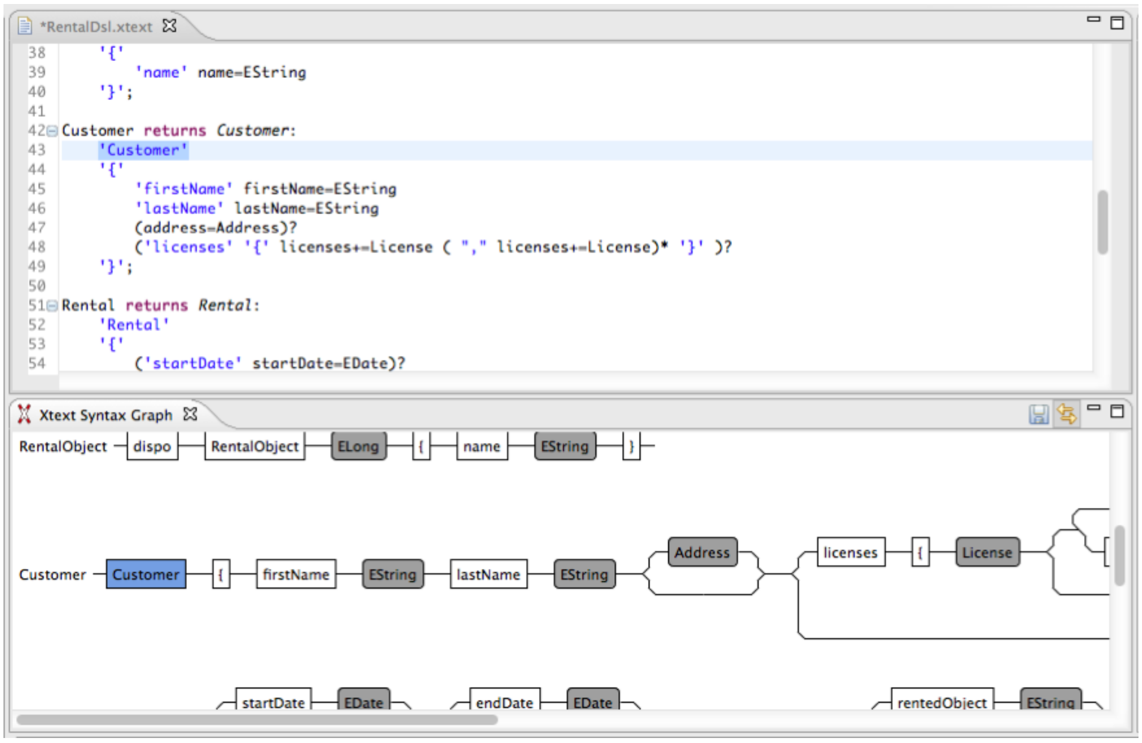
All editors generated with Xtext have the following features for free:

- 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

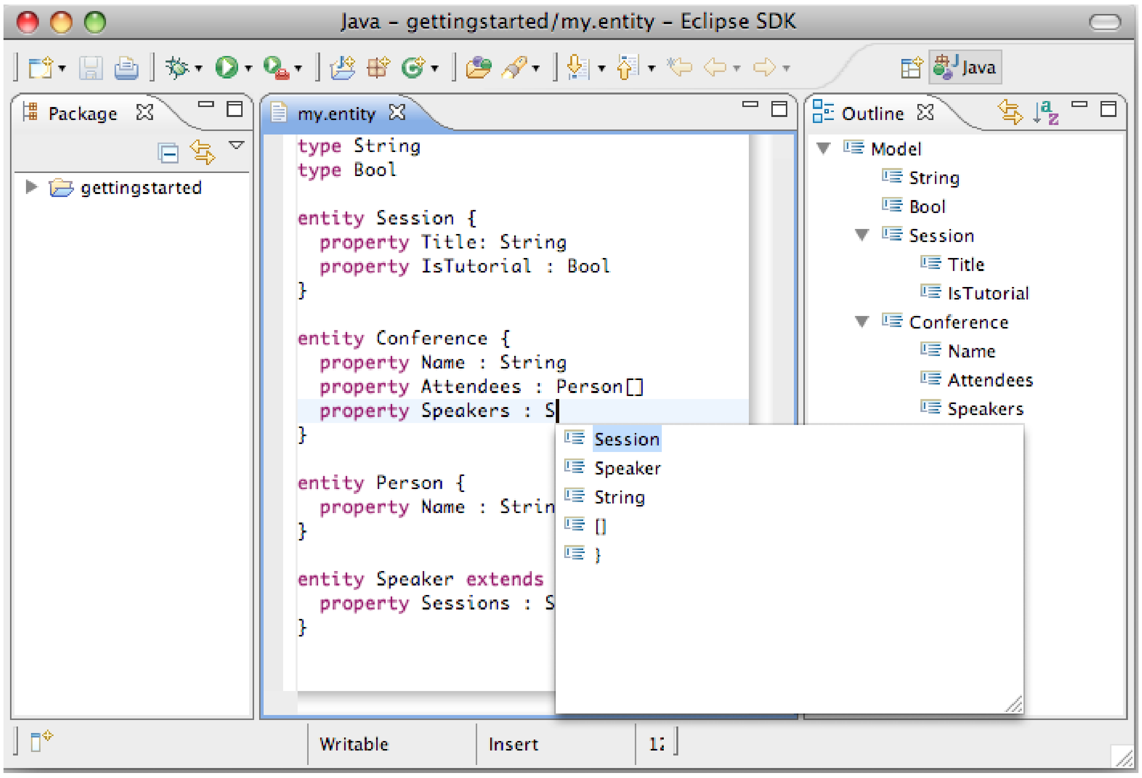


```
    '{'
        'name' name=EString
    '}';

Customer returns Customer:
    'Customer'
    '{'
        'firstName' firstName=EString
        'lastName' lastName=EString
        (address=Address)?
        ('licenses' '{' licenses+=License ( "," licenses+=License)* '}' )?
    '}';

Rental returns Rental:
    'Rental'
    '{'
        ('startDate' startDate=EDate)?
```

*RentalDsl.xtext
Xtext Syntax Graph
RentalObject dispo RentalObject ELong { name EString }
Customer Customer { firstName EString lastName EString Address licenses { License
startDate EDate endDate EDate rentedObject EString

Image 20 XText Syntax Graph

Generated editor



```
type String
type Bool

entity Session {
  property Title: String
  property IsTutorial : Bool
}

entity Conference {
  property Name : String
  property Attendees : Person[]
  property Speakers : S
}

entity Person {
  property Name : Strin
}

entity Speaker extends
  property Sessions : S
}
```

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

VII

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 ?