Améliorer votre productivité avec Eclipse Modeling

POSS 2018

06 décembre 2018

Table des matières

I - Presentation *5*

II - Eco-Système Eclipse *7*

III - Eclipse Modeling *9*

Presentation

OPCoach

- **Olivier Prouvost**
- **Formation Eclipse** : RCP, E4, Modeling, Build, GIT, en Français, Anglais, Espagnol
- **Expertise Eclipse**
- **Sourcing de profils Eclipse**
- **Web** : [1]
- **Twitter** : **@OPCoach_Eclipse**
- **Certifié ICPF & PSI et inscrit au datadock**

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

Eco-Système Eclipse

II

La fondation Eclipse

Objectifs :

- Fournir une gouvernance pour l'eco-système Eclipse
  - gestion des projets
  - gestion des membres
  - gestion des groupes de travail
- Hébergement des projets
- Promouvoir les technologies Eclipse et les logiciels open source
- Aider les membres et les utilisateurs à faire des affaires avec Eclipse
- Organiser les conférences

Structure:

- La fondation Eclipse est basée à Ottawa et en Europe
- C'est une organisation indépendante financée par ses membres

Quelques chiffres

- 350 projects (dont EE4J)
- 162 MLOC (dont 28 millions dans Eclipse Modeling)
- > 1120 repositories GIT
- >1 400 committers
- > 5 millions d'utilisateurs de l'IDE Eclipse
- 1.5 million downloads/mois
- 2 millions visiteurs / mois
- > 5 millions $ de budget annuel
- > 280 membres (11 strategic)
- ~30 employés
- > 50 événements par an
  - EclipseCon Europe, France, Devoxx US...

Le secret du succès d'Eclipse

- Eclipse est un modèle logiciel indépendant pour la collaboration et l'innovation
- La collaboration dans l'industrie nécessitant :
  - Un modèle de license permettant le partage et le co-developpement
  - Une gestion de l'IP pour maximiser les opportunités commerciales
  - Une gouvernance pour permettre à tous les participants d'être impliqués
- **La fondation Eclipse apporte tous ces avantages**

Eclipse Modeling

III

Introduction

Les projets de Modeling permettent d'augmenter votre productivité de développement
Ils sont basés sur l'architecture RCP (Rich Client Platform) et OSGi
Les projets Modeling s'utilisent de deux manières :

- **MDA**: Model Driven Application:
  - le modèle est utilisé par l'application au runtime
- **MDD**: Model Driven Development:
  - les modèles sont utilisés par les outils de développement

Cartographie des principaux framework

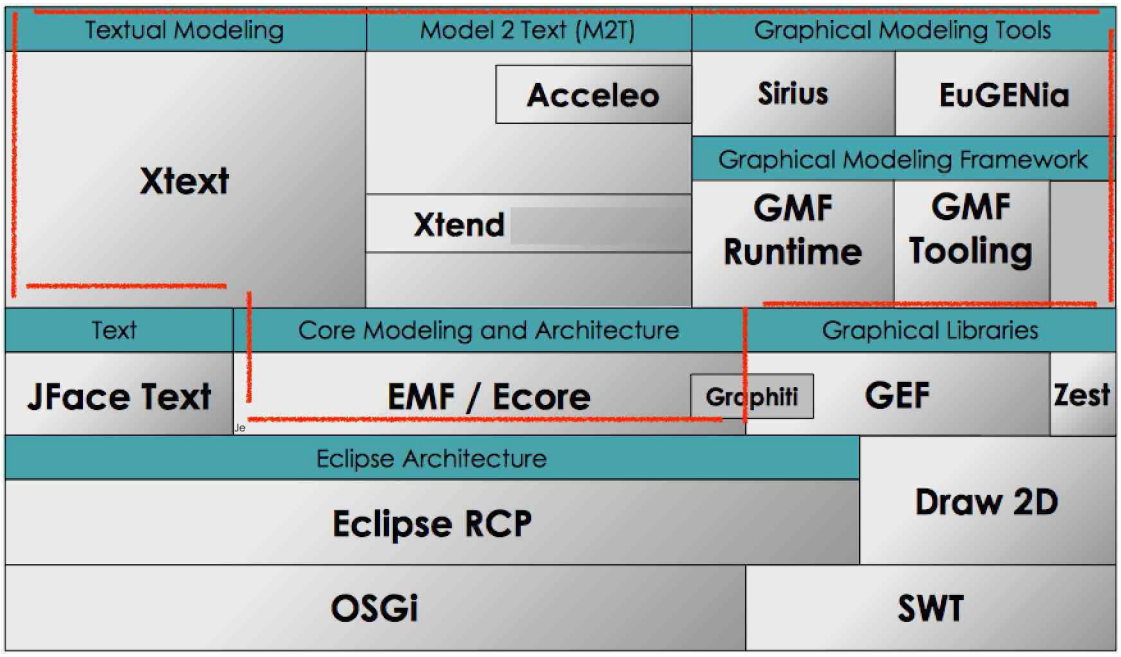
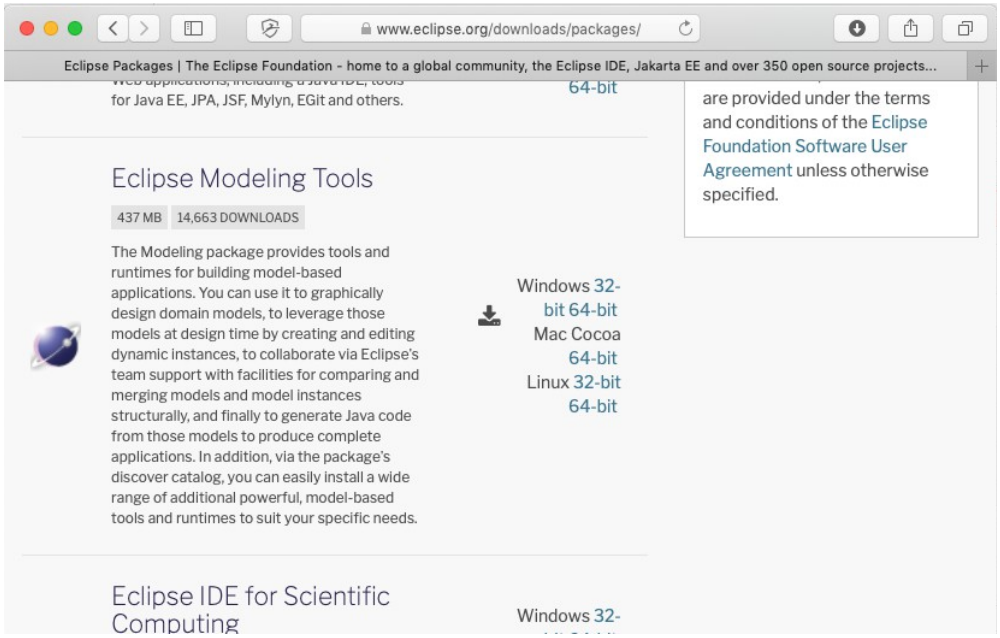


Image 1

Installation des outils

Comment installer les outils de modeling ?
Il suffit d'installer la version de l'IDE Eclipse pour Modeling

- puis d'utiliser l'Eclipse Market Place (Help -> MarketPlace)



Comment utiliser EMF ?

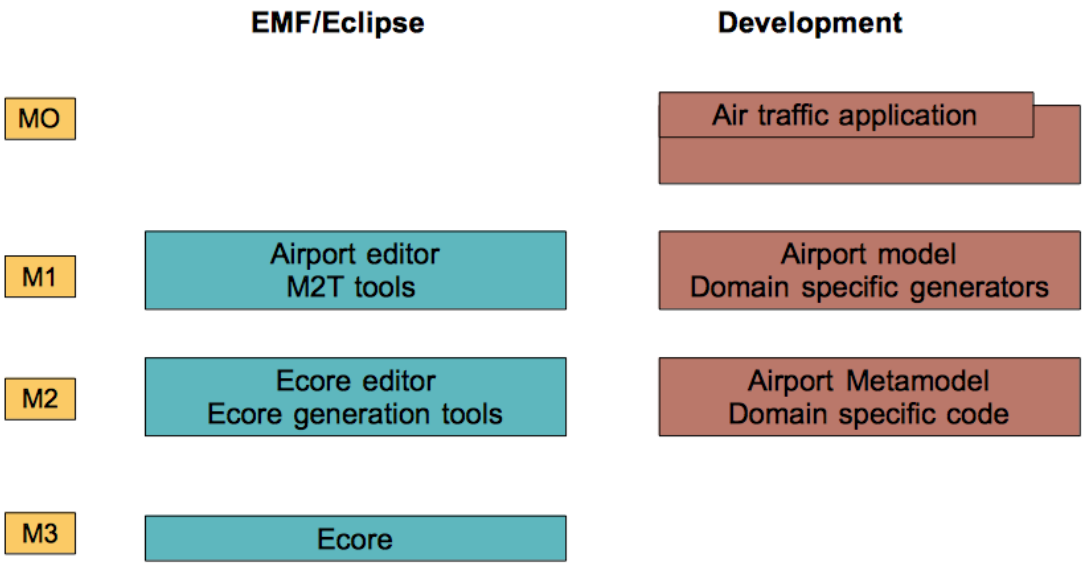


Image 2 EMF in the development process

model / meta model

La distinction entre Model et Meta Model

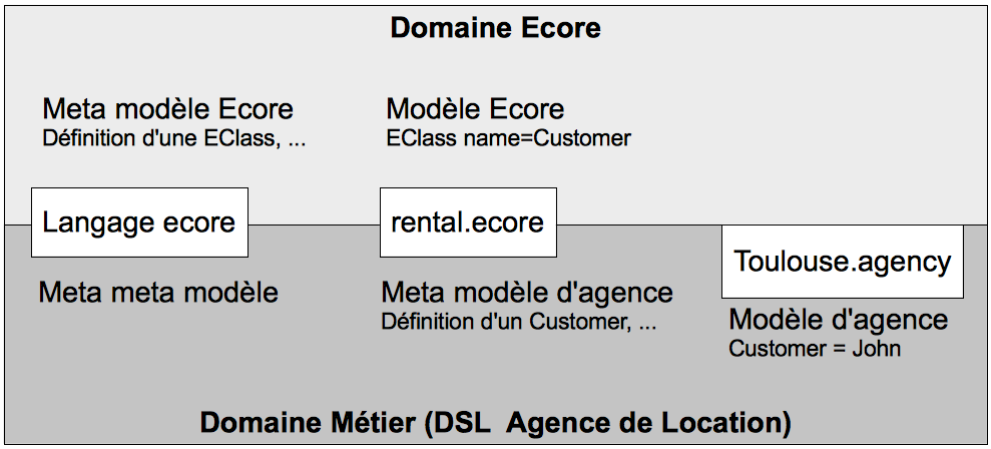
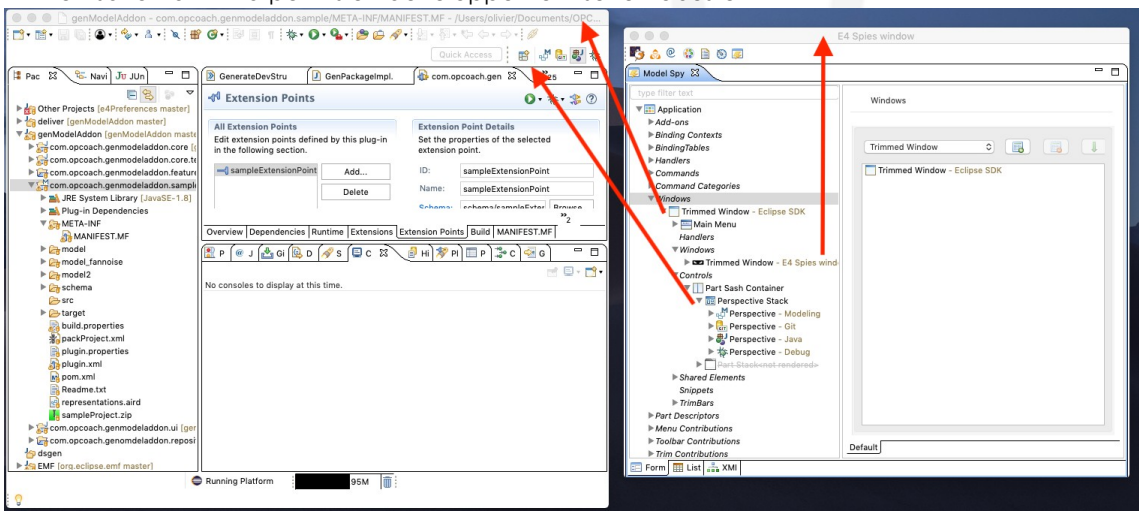


Image 3 Model domains and meta models

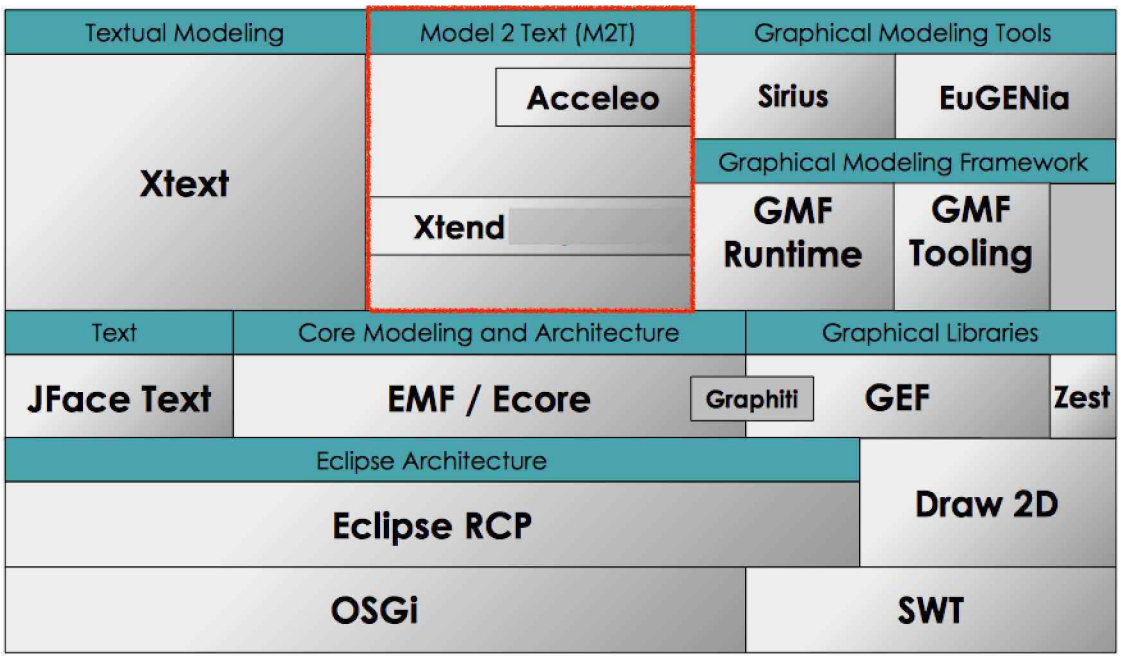
Les modèles dans l'IDE Eclipse

Le développement du moteur de l'IDE Eclipse repose à 100 % sur EMF et sur les modèles
Il est utilisé chaque jour par des millions d'utilisateurs
L'utilisation d'EMF a permis un développement sur et robuste

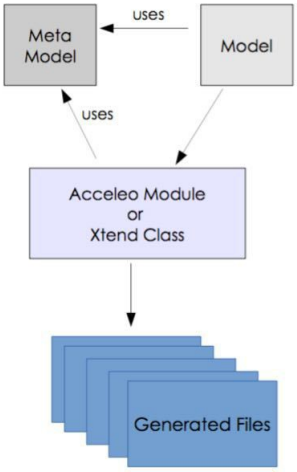


Model to Text (M2T)

Les projets M2T permettent d'utiliser les modèles pour générer du text (ie du code).



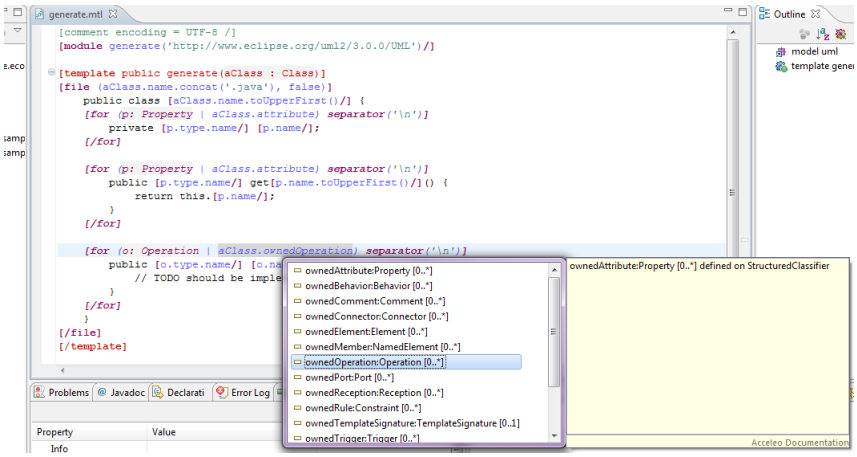
Principe de base de M2T



Le langage Acceleo

- Le projet Acceleo définit un langage de templates pour générer du texte à partir d'un modèle.
- Il est normalisé et reconnu comme langage par l'OMG
- Il est très facile d'accès (cf https://www.eclipse.org/acceleo/)
- Les outils sont bien intégrés dans Eclipse : wizards, launch configuration, vues et perspectives dédiées

- Pour installer Acceleo, utiliser le market place d'Eclipse (Help -> Eclipse Market Place)
- Un template acceleo va permettre de générer le texte voulu en utilisant des balises qui accèdent au modèle.



Le langage Xtend

- Xtend est un langage simplifiant Java qui contient également un langage de templates

```
GenerateBean.xtend
1  package com.opcoach.training.rental.xtend
2
3  import org.eclipse.emf.ecore.EReference;
4  import org.eclipse.emf.ecore.EAttribute;
5  import org.eclipse.emf.ecore.EClass;
6
7  class GenerateBean {
8
9      def generateCode(EClass c,  String packName) '''
10         package «packName»;
11         public class «c.name»
12         {
13             «FOR f : c.EAllStructuralFeatures»
14             private «f.type» _«f.name»;
15             «ENDFOR»
16
17             «FOR f : c.EAllStructuralFeatures»
18             «f.generateGetter»
19             «ENDFOR»
20
21         }
22     '''
23
24     def dispatch generateGetter(EAttribute att) '''
25     public «att.type» get«att.name.toFirstUpper»()
26     {
27         return _«att.name»
28     }
29     '''
30     def dispatch generateGetter(EReference ref) '''
31     public Collection<«ref.type»> get«ref.name.toFirstUpper»()
32     {
33         return _«ref.name»
34     }
35     '''
36
37     def dispatch getType(EAttribute att)    { att.EType.instanceClassName   }
38     def dispatch getType(EReference ref)    { ref.EType.name     }
39 }
```

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 4 Xtend generator sample

Comment utiliser M2T ?

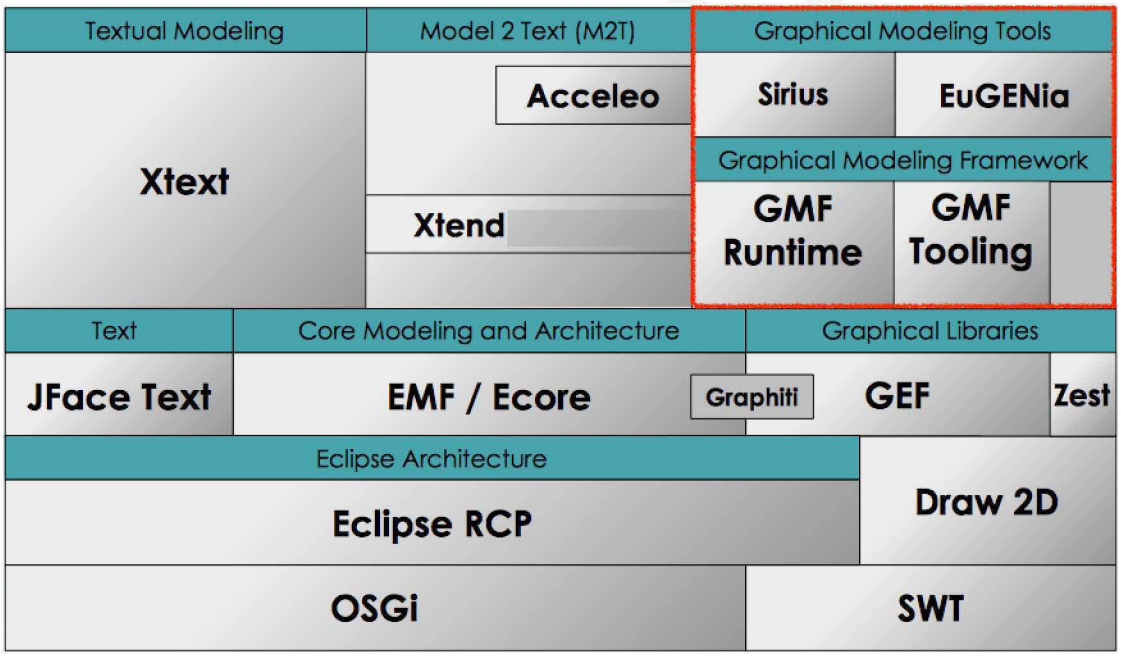
Ecrire les générateurs de code :

- compatibles avec votre modèle/meta modèle (-> erreurs détectées en amont)
- générer le code pénible à écrire et toujours identique :
  - mapping SQL
  - mapping coté Web (php, angular, node.js..)
  - utilisation de vos librairies de code à appeler (ex : librairies de simulation)
  - ...

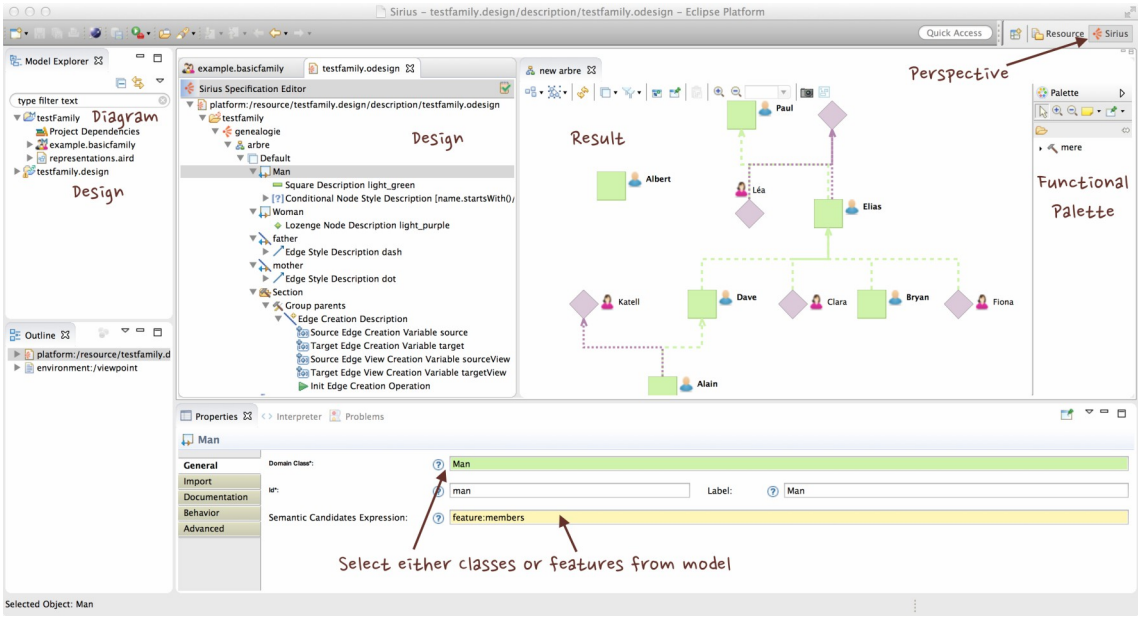
Remarque : si vous décrivez plusieurs fois la même information dans du code, il faudrait le générer !

Les éditeurs graphiques de modèles

- Editer un modèle graphiquement est intéressant quand il y a beaucoup de liens
- Sirius permet de décrire l'éditeur de manière simple.
- Sirius s'installe via le market place
- Les éditions graphiques et le modèle sont synchronisées.



- Sirius est une chaine d'outils permettrant de décrire des éditeurs graphiques :
  - diagram : noeuds et liens
  - tables : lignes d'instances du modèle avec les propriétés en colonne
  - matrices : lignes et colonnes d'instances avec des valeurs dans les cellules
- La description est simple est le résultat s'utilise sans génération de code

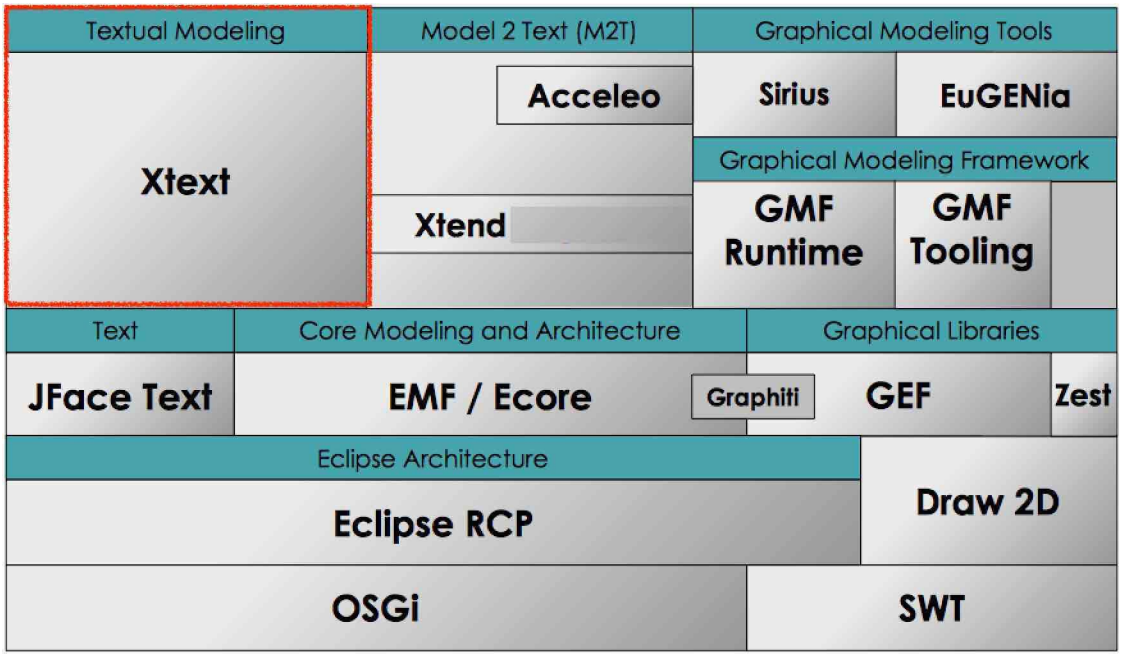


Sirius overview

Remarque : l'éditeur graphique pour les modèles ecore a été réécrit avec Sirius

Les éditeurs textuels de modèle

- On utilise les éditeurs textuels pour les modèle :
  - de code
  - de descriptions
  - contenant peu de liens à représenter graphiquement (mais navigation possible)
- Les éditeurs textuels
  - permettent de contrôler la saisie (validation syntaxique et sémantique)
  - d'appliquer un générateur de code
- On utilise les éditeurs de texte de modèle en :
  - MDD : pour décrire l'application à générer partiellement ou totalement
  - MDA : pour fournir l'éditeur de texte à l'utilisateur final.



Xtext

- Projet Eclipse permettant de générer un éditeur de texte pour un modèle
- 'Language IDE framework'
- Le code est généré à partir d'une grammaire ou d'un meta modèle exprimé en ecore.
- XText peut générer un éditeur pour IntelliJ, ou pour le web

Site web : *http://www.eclipse.org/Xtext/* [2]

Caractéristiques des éditeurs XText

Tous les éditeurs basés sur XText ont les propriétés suivantes :

- syntax highlighting
- auto completion
- index de recherche
- validation et quickfixes
- gestion des erreurs et warning
- templates
- outline
- definition des hyperlink (cross languages)
- branchement d'un générateur automatique à la sauvegarde

Remarque : utiliser XText est un moyen d'avoir l'équivalent de l'éditeur Java mais pour votre propre langage.

2 - http://www.eclipse.org/Xtext/

Exemple d'éditeur

test.course

```
Course "Eclipse Rich Client Application"
    Id RCP Year 2016 Reference "OPC 16 XXX XXX XXX" Language French
    Description "Descriptions" Duration 4

    Objectives
        - Find a solution
        - knowing A, knowing B
        - knowing C and D

    Audience
        - Project manager
        - Développer

    Prerequisites
    - knowing Java
    - knowing Java
    - knowing Java

    Title "Introduction" Timing 6
    - Architecture OSGi et Eclipse
```

Text after format

Comment utiliser XText ?

- Suivre les tutoriaux (5mn et 30 mn sur le site)
- Définir une grammaire et générer le code
- Lancer les plugins Eclipse générés
- Utiliser Xtend pour brancher un générateur de code

La définition de la grammaire

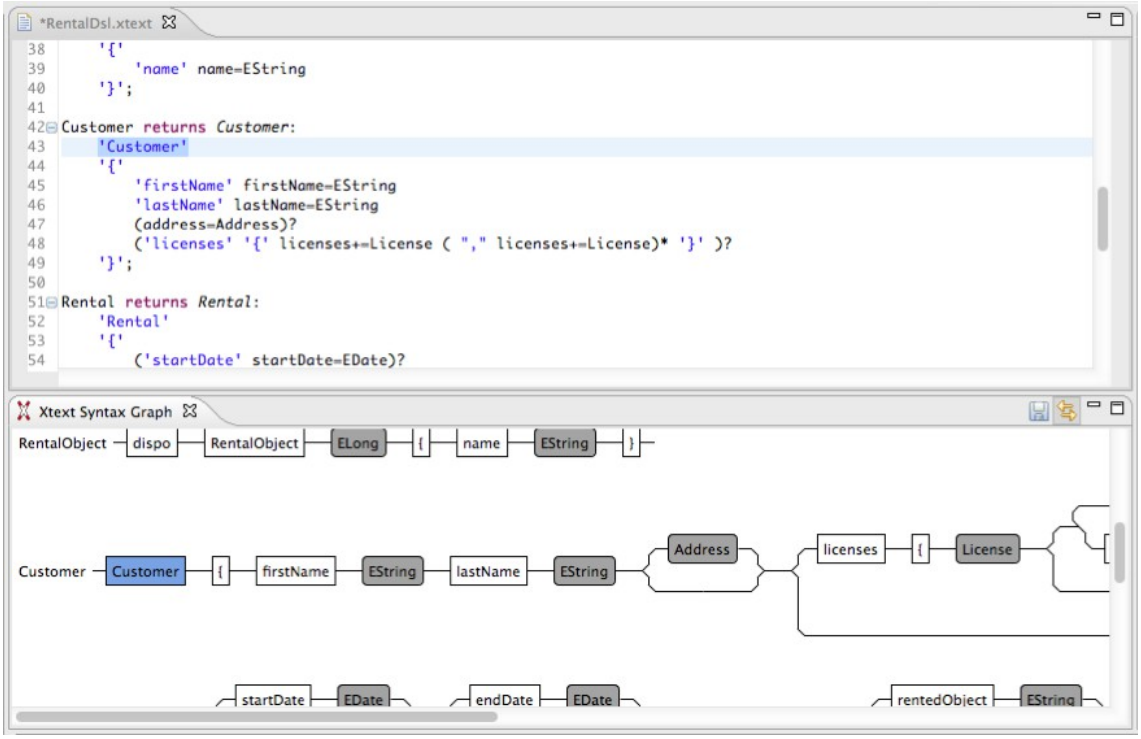


Image 5 XText Syntax Graph

L'éditeur généré

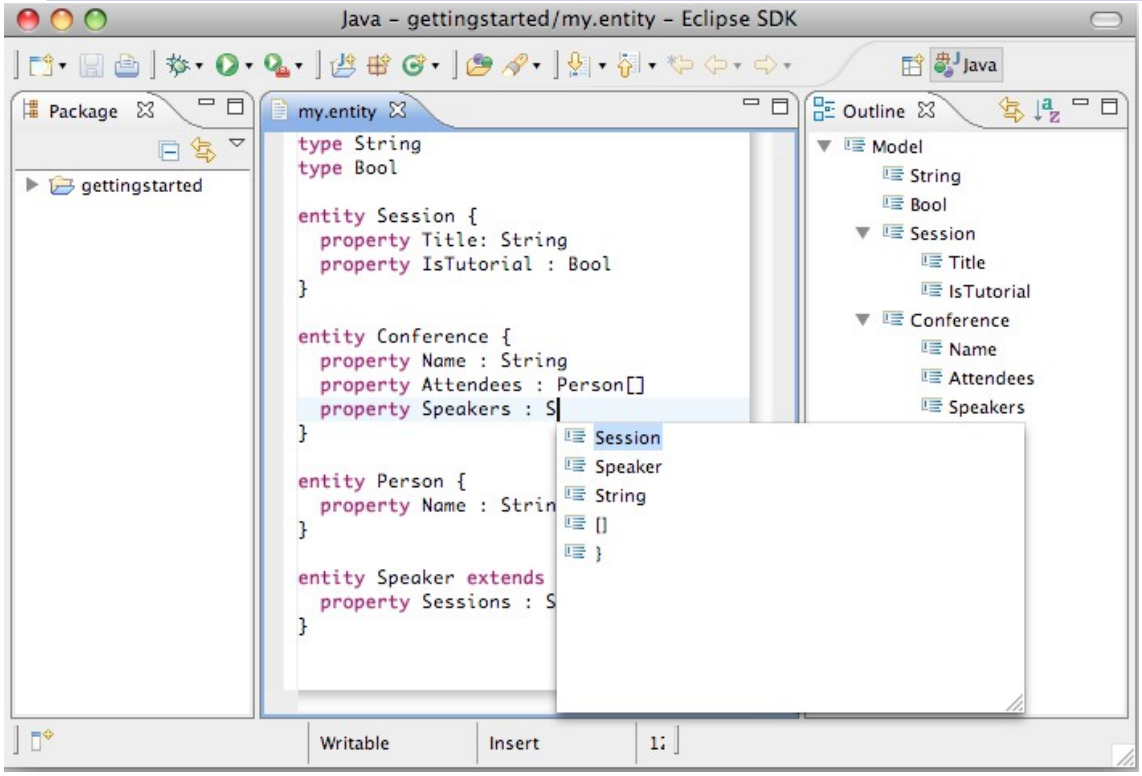


Image 6 Sample editor

Paramétrer votre éditeur

Il est très facile (et amusant !) de customiser votre éditeur :

- quickfixes (**@Fix** annotation)
- erreurs et warning (**@Check** annotation)
- auto completion (textuelle ou semantique)
- formattage du texte
- générateur de text (déclenché à la sauvegarde)
- templates (compatibles avec votre syntaxe)

Cas d'utilisation de XText

- Editeurs simples pour utilisateurs finaux :
  - Menus de restaurant éditables par le cuisinier et générés en HTML pour le site
  - Fichiers de configuration cohérents avec contrôles des saisies sémantiques
  - Re-engineering d'éditeurs de textes legacy (sans contrôles)
- Editeurs pour modéliser le développement (MDD)
  - Description de composants pour générer du code
    - Intégration des règles de codage dans les générateurs

Les autres projets modeling

- **EMF Forms** : permet de générer des formulaires de saisie à partir du modèle.
- **Edapt** : permet de gérer les évolutions d'un méta modèle pour mettre à jour les modèles
- **CDO** : pour stocker des objets du modèle dans des BDD

- **OCL** : pour exprimer des contraintes sur un modèle
- Cf : *https://www.eclipse.org/modeling/*[3]

Questions ?

Image 7

Pour des démos -> stand Eclipse

3 - https://www.eclipse.org/modeling/