Université Abdelhamid Ibn Badis Mostaganem

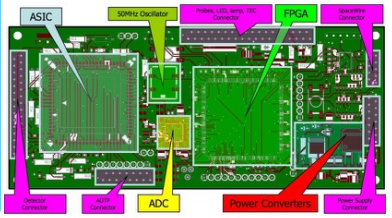
Faculté des sciences exactes et informatique

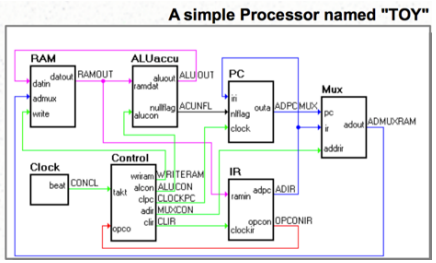
Ingénierie Dirigée par les Modèles (IDM)

Méta-modélisation (EMF)

Modélisation

- Abstraire un système (selon le but de notre modèle l'abstraction sera différente)
- Représenter un système (pour communiquer, analyser, générer du code, documenter, ...)



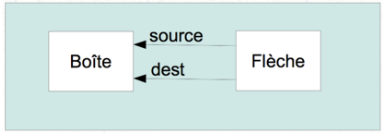


Méta-Modélisation

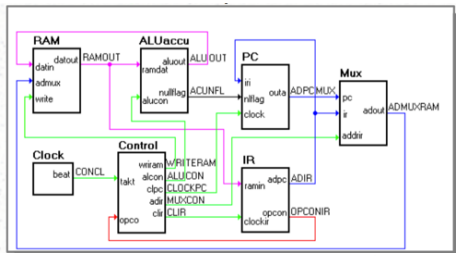
- Représenter un domaine (architecture, télécommunication, robotique, ...)
- Pour faciliter la manipulation de modèles (éditeurs, transformations, ...)

Métamodèle

Doit être fait par un spécialiste du domaine !



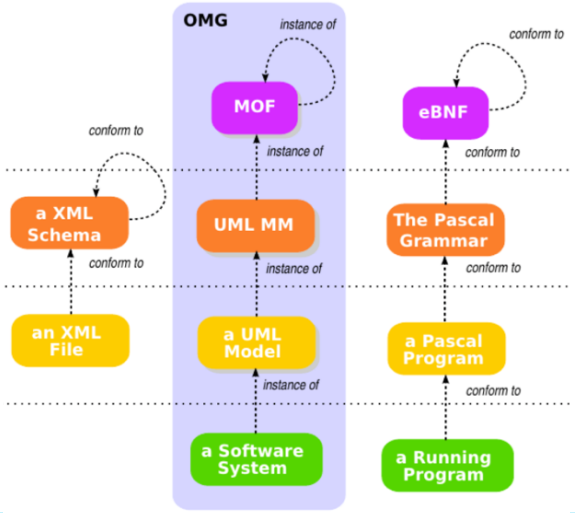
Modèle



Modèle et Métamodèle

- Un **modèle** est :
  - Une instance d'un métamodèle
  - *Conforme à* son métamodèle
  - Une syntaxe abstraite de l'entité modélisée
- Un **métamodèle** est:
  - Un modèle
  - La définition des concepts et des relations des instances qui lui sont conformes (proche de la définition d'une grammaire)

Modèle et Métamodèle



Métamodelisation

- Les langages de modélisation sont souvent dédiés à un domaine particulier (ou DSML pour domain specific modelling langage).
- L'un des défis de l'IDM est de pouvoir définir un DSML à travers la métamodélisation.
- **Définition:** La **méta-modélisation** est une activité de modélisation qui consiste à mettre en oeuvre un **langage**

Métamodelisation

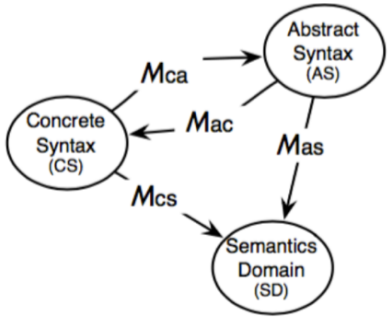
- La mise en oeuvre d'un **langage** consiste à:
  - produire des méta-modèles
  - définir la sémantique du langage
  - construire un ensemble d'outils exploitant les méta-modèles (analyseurs, compilateurs des générateurs de code, etc)

Qu'est-ce qu'un langage ?

- Un **langage** (en linguistique ou informatique) est caractérisé en général par une syntaxe et une sémantique.
  - La syntaxe permet de décrire les éléments et les règles qui constitue le langage
  - La sémantique donne un sens à chaque élément du langage.
- **Définition:** Un langage (L) est défini selon le tuple {S, Sem} ou S est sa syntaxe et Sem sa sémantique.

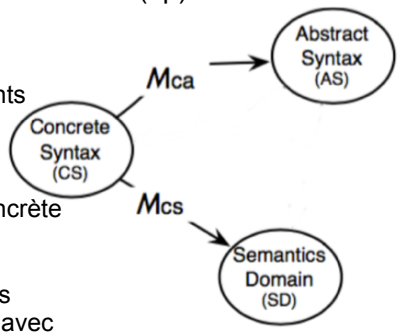
Composants d'un langage

- En informatique, le **langage** est caractérisé par deux types de syntaxes :
  - **la syntaxe concrète** (CS), manipulée par l'utilisateur du langage
  - et **la syntaxe abstraite** (AS) qui est la représentation interne (d'un programme ou d'un modèle) manipulée par l'ordinateur
  - Le **domaine sémantique** représente l'ensemble des états possibles du système



Langage de programmation

- **Définition:** Un **langage de programmation** (Lp) est défini selon le tuple {AS,CS,Mca,SD,Mcs}:
- Mca permet d'enlever tout les éléments syntaxiques inutiles à l'analyse du programme
- Mcs est le mapping de la syntaxe concrète vers le domaine sémantique
- La sémantique est donnée en liant les constructions de la syntaxe concrète avec l'état auquel elles correspondent dans le domaine sémantique (Mas).

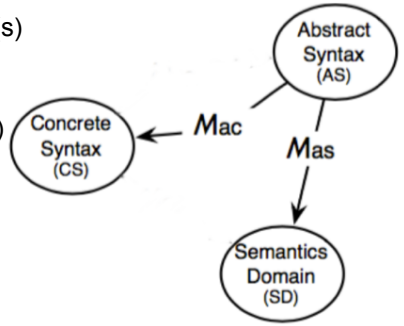


Langage de modélisation

- Pour répondre au critère de réutilisabilité, l'IDM se focalise sur la **syntaxe abstraite** pour décrire un langage de modélisation.
- La syntaxe concrète consiste à définir uniquement le lien entre les constructions de la syntaxe abstraite et leurs representations **textuelles** et **graphiques** (Mac).
- Ce changement de sens du lien par rapport aux langages de programmation permet de définir plusieurs syntaxes concrètes (Mac) pour une même syntaxe abstraite et par conséquent plusieurs représentations d'un même modèle.

Langage de modélisation

- **Définition:** Un **langage de modélisation** (Lm) est défini selon le tuple {AS,CS,Mac,SD,Mas} ou :
- CS est la(les) syntaxe(s) concrète(s)
- Mac est l'ensemble des mappings de la syntaxe abstraite vers la (les) syntaxe(s) concrète(s),
- Mas est le mapping de la syntaxe abstraite vers le domaine sémantique.



Les langages dédiés de modélisation - DSML

- IDM considère qu'un système complexe peut être modéliser à travers plusieurs DSMLs ayant des relations entre eux.
- Ces langages sont généralement de petite taille et doivent être facilement manipulables, transformables, combinables, etc.
- Nous introduisons dans ce qui suit quelques standards, et outils disponibles pour décrire les différentes principales parties d'un DSML(la syntaxe abstraite et la syntaxe concrète).

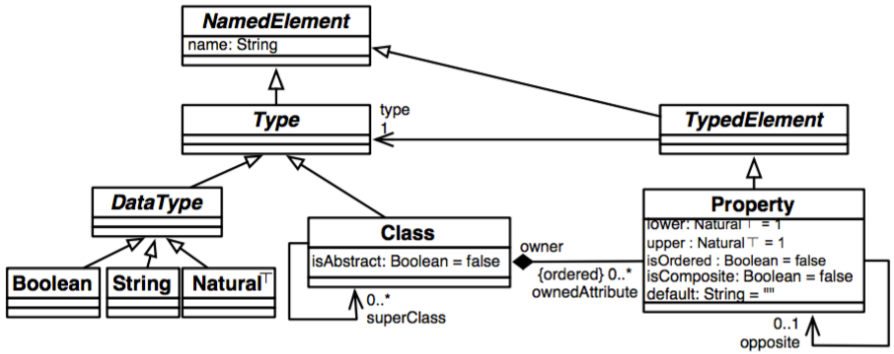
DSML - Syntaxe abstraite

- La syntaxe abstraite (AS) d'un langage de modélisation exprime, de manière structurelle, l'ensemble de ses concepts et leurs relations.
- Les langages comme le standard MOF de l'OMG permet de décrire un métamodèle représentant la syntaxe abstraite d'un langage de modélisation.
- Plusieurs environnements et langages de métamodélisation sont disponibles afin de décrire la syntaxe abstraite d'un langage de modélisation tels que :
  - Eclipse-EMF/Ecore
  - GME/MetaGME
  - AMMA/KM3,
  - Kermeta

DSML - Syntaxe abstraite

- Ces langages s'inspirent de l'approche orientée objet et reposent sur les mêmes constructions élémentaires.
- Les langages de métamodélisation objet offre le concept de classe (**Class**) pour définir les concepts d'un DSML.
- Une classe est composée de propriétés (**Property**) qui la caractérisent.
- Une **propriété** est appelée :
  - **référence** lorsqu'elle est typée (**TypedElement**) par une autre classe
  - et **attribut** lorsqu'elle est typée par un type de donnée (exp. booléen, chaine de caractère et entier).

DSML - Syntaxe abstraite



Concepts principaux de métamodélisation (EMOF 2.0)

DSML - Syntaxe concrète

- La syntaxe concrète (CS) d'un langage fournit à l'utilisateur un formalisme **graphiques** et/ou **textuels**, pour manipuler les concepts de la syntaxe abstraite afin de créer des "instances ".
- Différents outils **graphiques** sont basés sur EMF (Eclipse Modeling Framework) pour représenter la syntaxe concrète d'un langage. Par exemples :
  - GMF (Generic Modeling Framework) d'Eclipse,
  - TOPCASED
  - GEMS,
  - TIGER, etc.
  - D'autres outils sont plutôt textuels comme par exemple Sintaks et ATLAS.

Eclipse Modeling Framework

Eclipse Modeling Framework

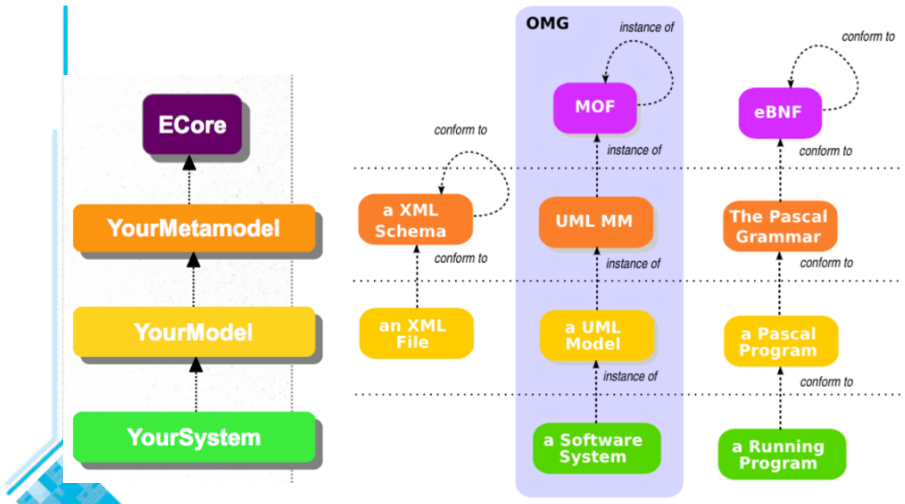
- Le projet EMF est un framework qui facilite la modélisation et la génération de code Java pour monter des applications et outils sur un modèle de données structuré.
- EMF = un ensemble de plug-ins utilisés pour modéliser et générer du code ou autres résultat en se basant sur ce modèle.
- Il permet aux développeurs de créer leur **méta-modèle** via différents moyens tels que XMI, les annotations Java, UML ou un schéma XML.

Eclipse Modeling Framework

- A partir d'une spécification de modèles décrite, EMF fournit des outils et un support d'execution afin de produire :
  - (1) un ensemble de classes Java pour le modèle,
  - (2) des adaptateurs de classes qui permettent la visualisation et l'edition du modèle
  - et (3) un éditeur de base.

https ://www.eclipse.org/modeling/emf/

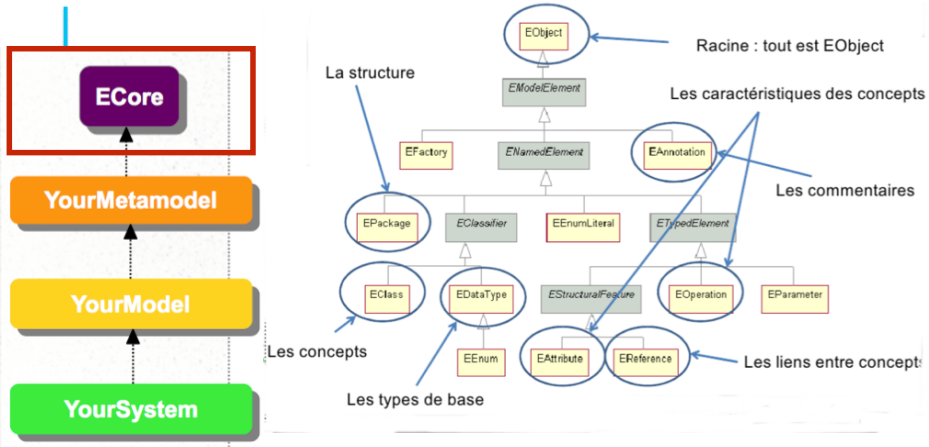
Eclipse Modeling Framework



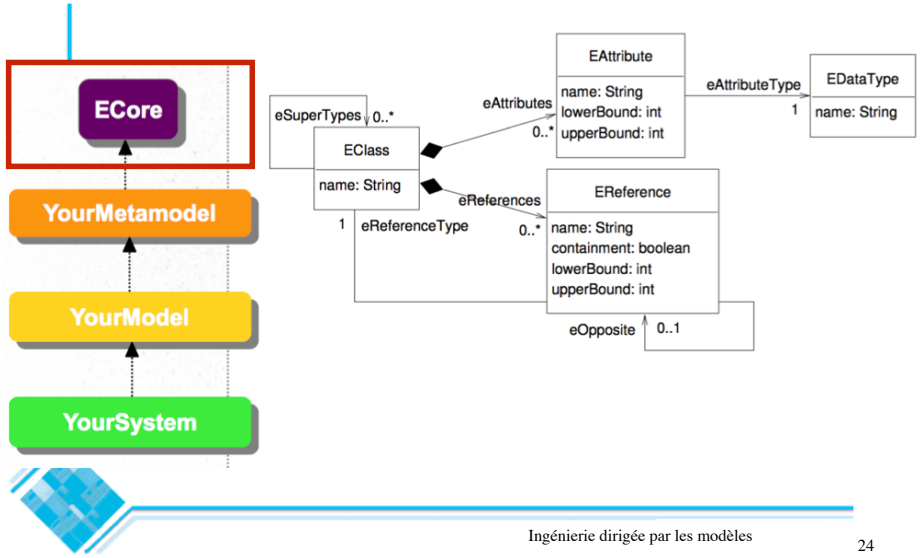
EMF (Core)

- **EMF (core)** est un standard de modélisation sur lequel se base plusieurs technologies et frameworks. Il se compose de trois principaux éléments :
  - **EMF** - Le noyau d'EMF inclut un méta-modèle (**Ecore**) pour décrire les modèles et les supports d'execution des modèles en incluant les API de manipulation des objets génériques et la persistance avec une sérialisation XMI.
  - **EMF.Edit** - ce framework se compose des classes génériques et réutilisable pour construire les éditeurs pour les modèles EMF.
  - **EMF.Codegen** - permet de générer tout ce qui est nécessaire pour construire un éditeur complet pour un modèle EMF.
    - Ceci inclut une interface (GUI) à partir de laquelle les options de génération peuvent êtres spécifier et les générateurs peuvent êtres invoqués.
    - La génération s'appuie sur sur le composant JDT (Java Development Tooling) d'Eclipse.

Eclipse Modeling Framework



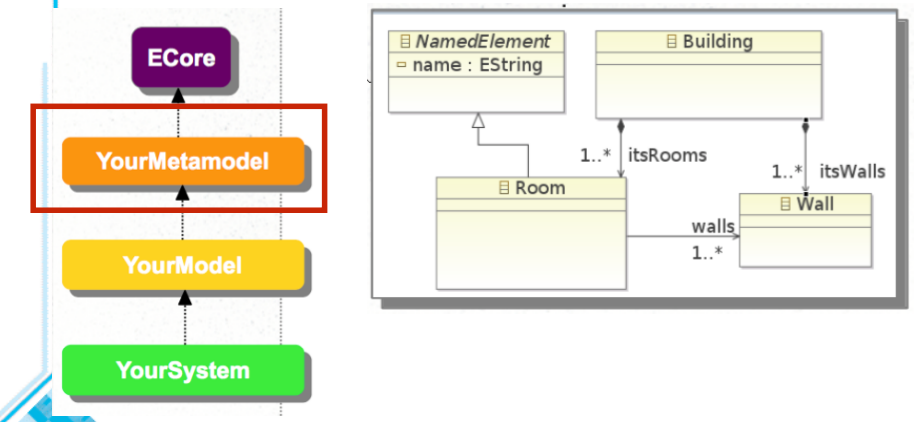
Eclipse Modeling Framework



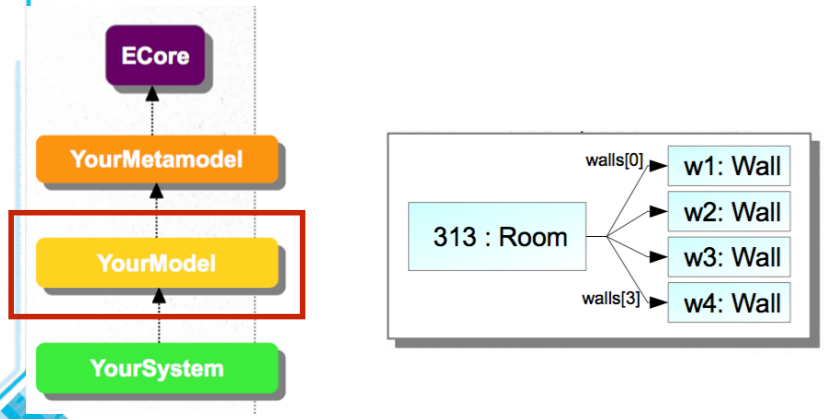
Ecore

- Le **fichier ecore** contient des informations sur les classes notamment :
  - **EClass** : représente la classe.
  - **EAttribute** : représente un attribut qui possède un nom et un type.
  - **EReference** : représente une association entre deux classes.
  - **EDataType** : représente le type d'un attribut, par exemple int, float, java.util.Date
- Le modèle **Ecore** montre un objet racine (root) représentant le modèle.
  - Ce modèle a des enfants qui sont les packages dont les enfants sont les classes.
  - Les classes possèdent des enfants qui sont les attributs.

Eclipse Modeling Framework



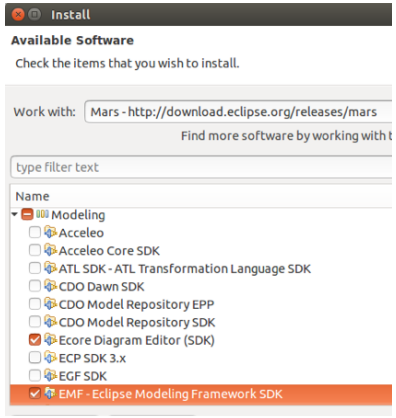
Eclipse Modeling Framework



Apprenant par l'exemple :
Créer un modèle EMF et générer du code Java

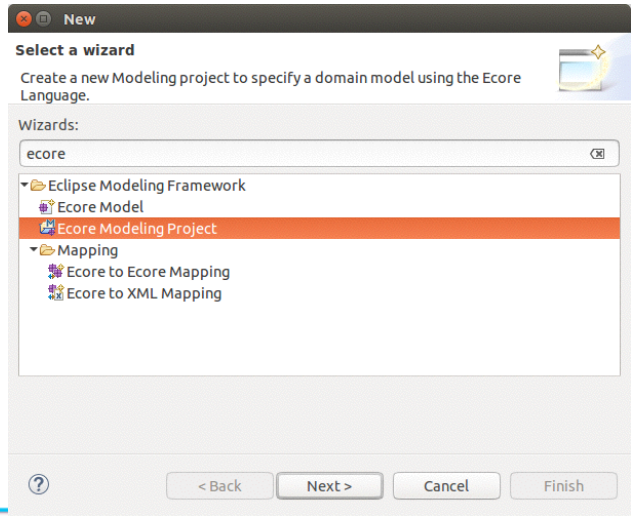
Installation

- Installer EMF via Eclipse Update manager à partir de Help > Install New Software...
- Sélectionner Modeling et installer:
  - **EMF - Eclipse Modeling Framework SDK** et
  - **Diagram Editor for Ecore (SDK)**, ce qui permet de créer des modèles via des diagrammes.
- Redémarrer Eclipse IDE après l'installation



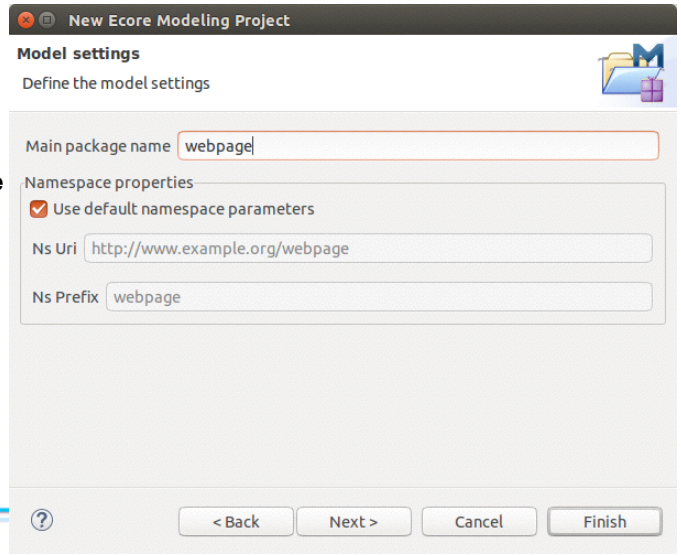
Création du projet Ecore

- Créer un nouveau projet appelé com.vogella.emf.webpage.model à travers
- **File > New > Project ... > Ecore Modeling Project.**



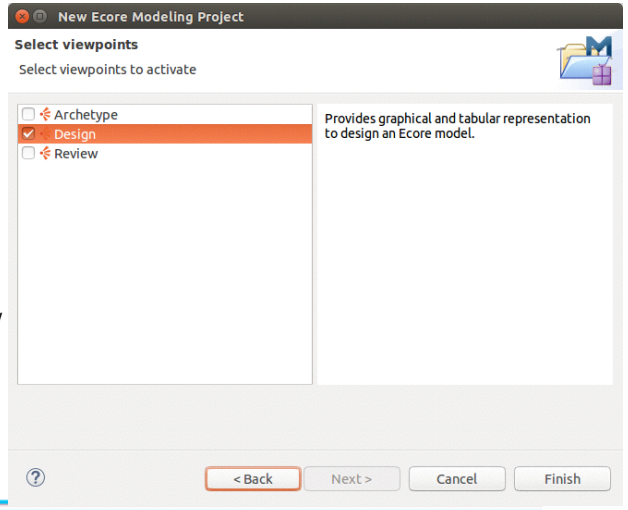
Création du projet Ecore

- Entrer **webpage.ecore** comme paramètre du nom de méta-modèle.



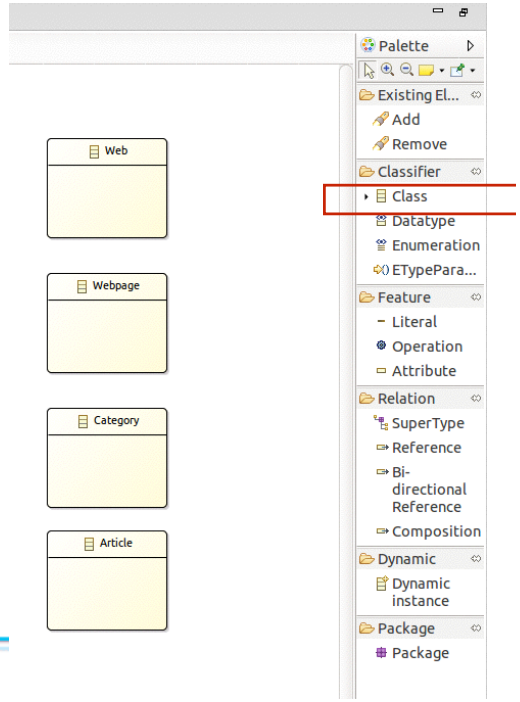
Création du projet Ecore

- Choisir le mode conception (**design**) qui permet d'ouvrir un éditeur de visualisation du modèle EMF.
- Ouvrir la vue **Properties** view via le menu Window > Show View > Other... > Properties. Cette vue permet de modifier les attributs de vos éléments du modèle.

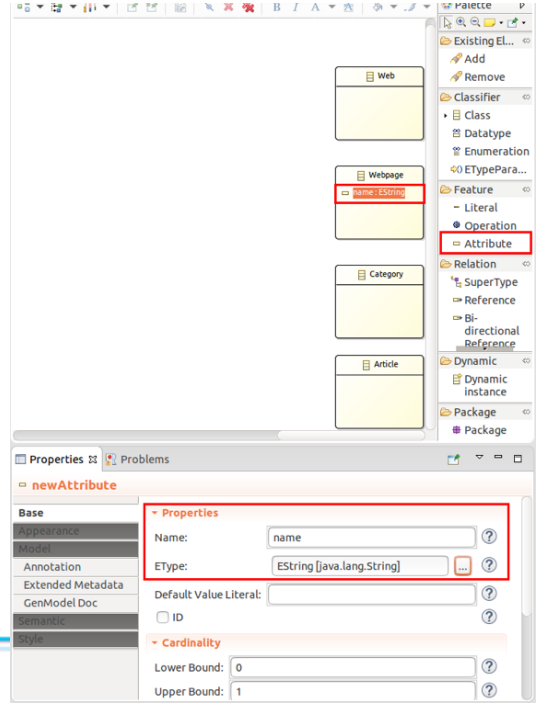


Création du modèle

Créez les **EClasses** : MyWeb, Webpage, Category et Article.

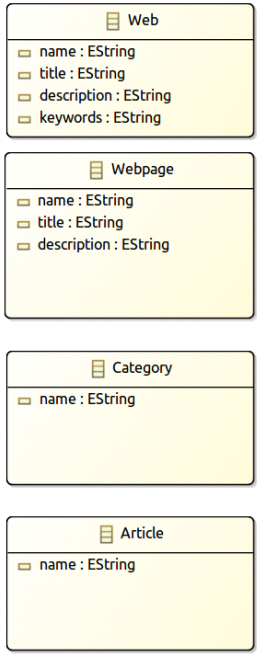


Associer l'attribut (**Attribute**) nom de type String à chaque objet



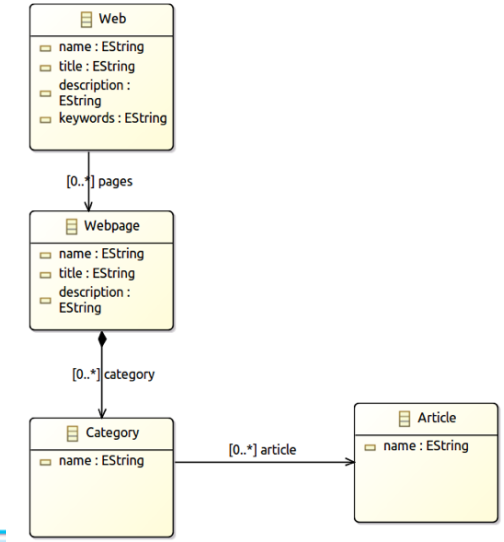
Création du modèle

Idem, ajoutez le titre, la description et les mots clés des éléments du modèle Web et Webpage.



Création du modèle

Ajoutez les associations

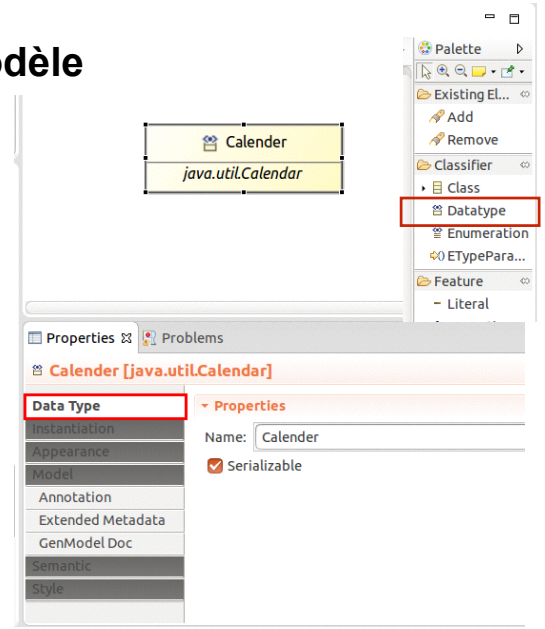


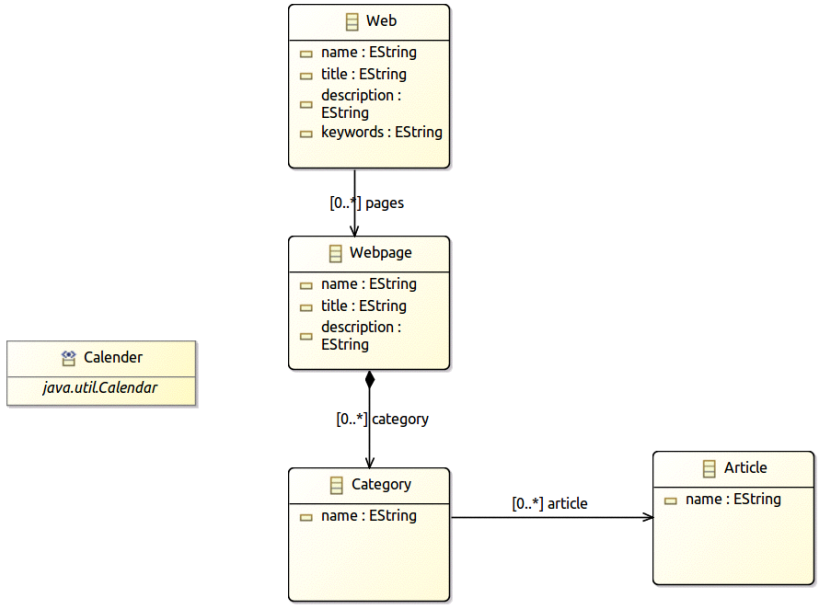
Création du modèle

Pour utiliser le type Calendar dans notre modèle. Sélectionnez **Datatype** et glissez le sur votre modèle.

Assignez le nom au Calendar et utilisez java.util.Calendar comme type.

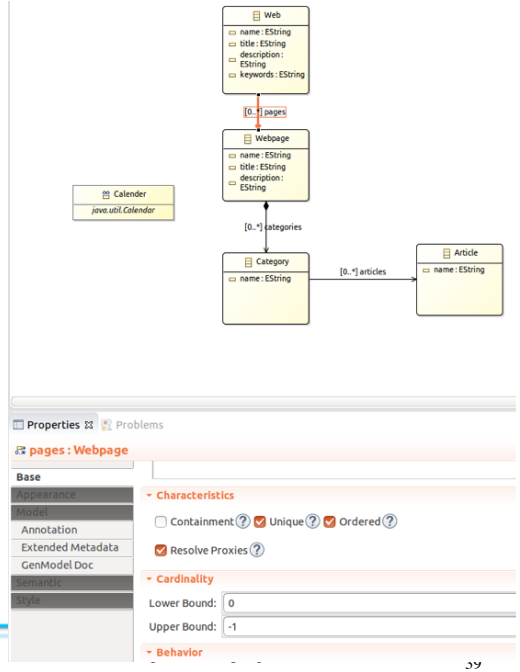
Ajoutez un Attribut nommé created à l'Article de type Calender





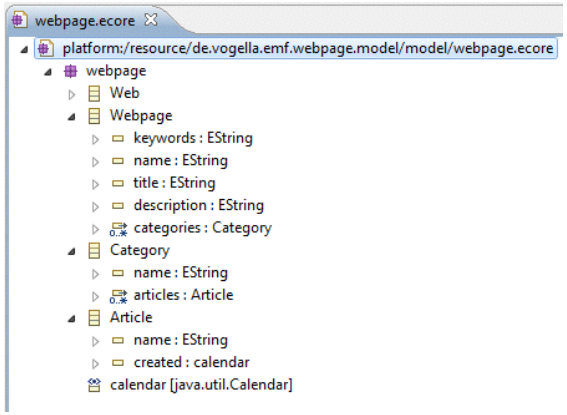
Création du modèle

Vérifier que upper bound est "-1" et que le propriété Containment est sélectionnée.



Visualiser le modèle Ecore

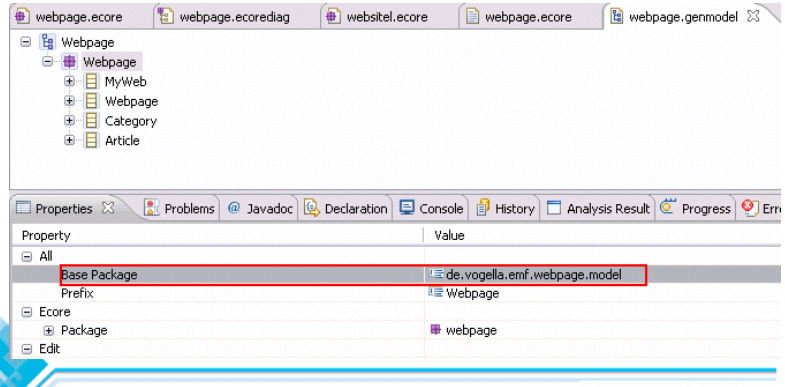
Fermez le diagramme et ouvrez le fichier **webpage.ecore**



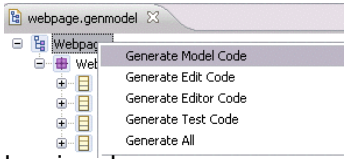
Déterminer le package

Ouvrez **webpage.genmodel** et sélectionner le noeud Webpage.

Mettre la propriété package à de.vogella.emf.webpage.model.



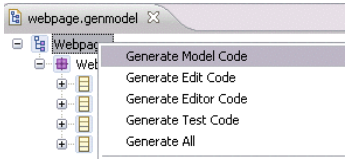
Génération du code



- Faites un clique droit sur le noeud racine du fichier **.genmodel** et sélectionner **Generate Model Code**.
- Le code généré se compose des éléments suivants :
  - model : les interfaces et Factory pour créer les classes Java.
  - model.impl : L'implémentation concrète des interfaces définies dans le modèle.
  - model.util : AdapterFactory.
    - La factory principale possède des méthodes pour créer tous les objets définis via les méthodes createObjectName().

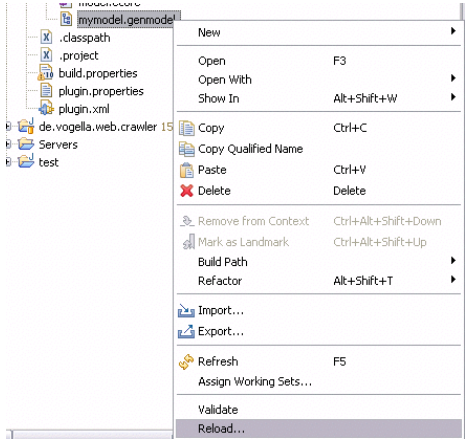
Génération du code

- Chaque interface généré hérite de l'interface EObject.
- EObject est la base de chaque classe EMF et c'est l'équivalent en EMF à java.lang.Object.
- Chaque méthode généré a un tag : @generated
  - Si vous souhaitez adapter manuellement la méthode vous aurez besoin de supprimer ce tag.



Modification du modèle

- Afin de modifier le modèle vous vous devez changer votre .ecore et faire un update du .genmodel en le rechargeant (reloading)



Conclusion

- La métamodelisation permet de:
  - Faciliter la manipulation de modèles (éditeurs, generation de code et plus tard les transformations, ...)
  - Accès aux outils basés sur les métamodèles:
    - Xtext
    - GMF
    - Transformation modèle à modèle
    - Transformations modèle à texte