



Développement de clients riches : Plateforme Eclipse RCP

Chapitre 4 : Modélisation

Modélisation via EMF

Mickaël BARON - 2012

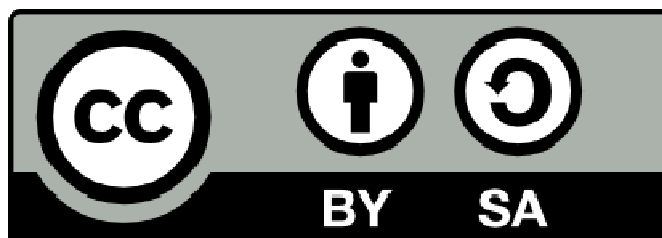
<mailto:baron.mickael@gmail.com> ou <mailto:baron@ensma.fr>

Creative Commons

Contrat Paternité

Partage des Conditions Initiales à l'Identique

2.0 France



<http://creativecommons.org/licenses/by-sa/2.0/fr>

Déroulement du cours

➤ Pédagogie du cours

- Illustration avec de nombreux exemples qui sont disponibles à l'adresse *mbaron.developpez.com/eclipse/emf*
- Des bulles d'aide tout au long du cours

➤ Logiciels utilisés

- Eclipse 3.7.1 Indigo (Modeling Tools)



Ceci est une astuce

➤ Prérequis

- Manipuler l'API SWT, JFace et les UI-Forms
- Développer un plugin

Ceci est une alerte



➤ Des articles sur EMF

- *Aide Eclipse (EMF Developer Guide, EMF Model Transaction Developer Guide)*
- *www.devx.com/java/Article/29093/1954*
- *www.vogella.de/articles/EclipseEMF/article.html*
- *eclipsesource.com/blogs/2011/03/22/what-every-eclipse-developer-should-know-about-emf-part-1*
- *eclipsesource.com/blogs/2011/03/31/what-every-eclipse-developer-should-know-about-emf---part-2*
- *www.vogella.de/articles/EclipseEMFNotification/article.html*
- *ed-merks.blogspot.com/2009/01/emf-ultra-slim-diet.html*
- *refcardz.dzone.com/refcardz/essential-emf*

➤ Des supports de cours

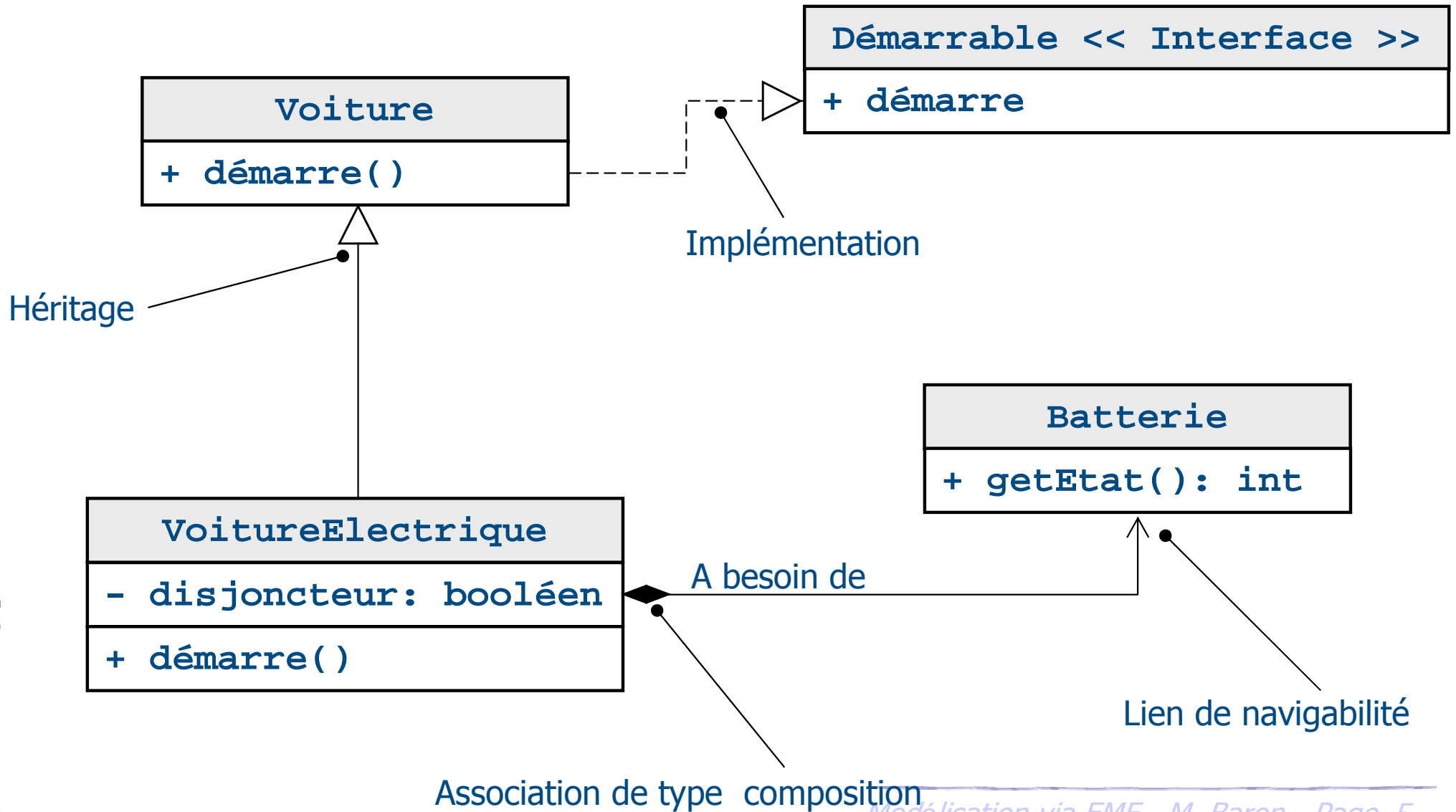
- *www.lisyc.univ-brest.fr/pages_perso/babau/cours/coursemf.pdf*
- *m2chm.univ-lemans.fr/modules/UE5/Fichiers/CoursLaforcade2.pdf*
- *anubis.polytech.unice.fr/cours/_media/2008_2009:sj5:idm:td:coursecore.pdf*

➤ Des livres

- *EMF : Eclipse Modeling Framework, 2nd Edition, 2008, ISBN-10: 0-321-33188-5*

Déroulement du cours

➤ Rappel pour le schéma UML (diagramme de classes)



Association de type composition

Organisation du cours ...

- Généralités
- Modèle Ecore
- Définir un modèle EMF
- Instancier un modèle
- Sauvegarder et charger les instances d'un modèle
- Manipuler le métamodèle
- Utiliser EMF sans conteneur OSGi
- Notification
- Transactions

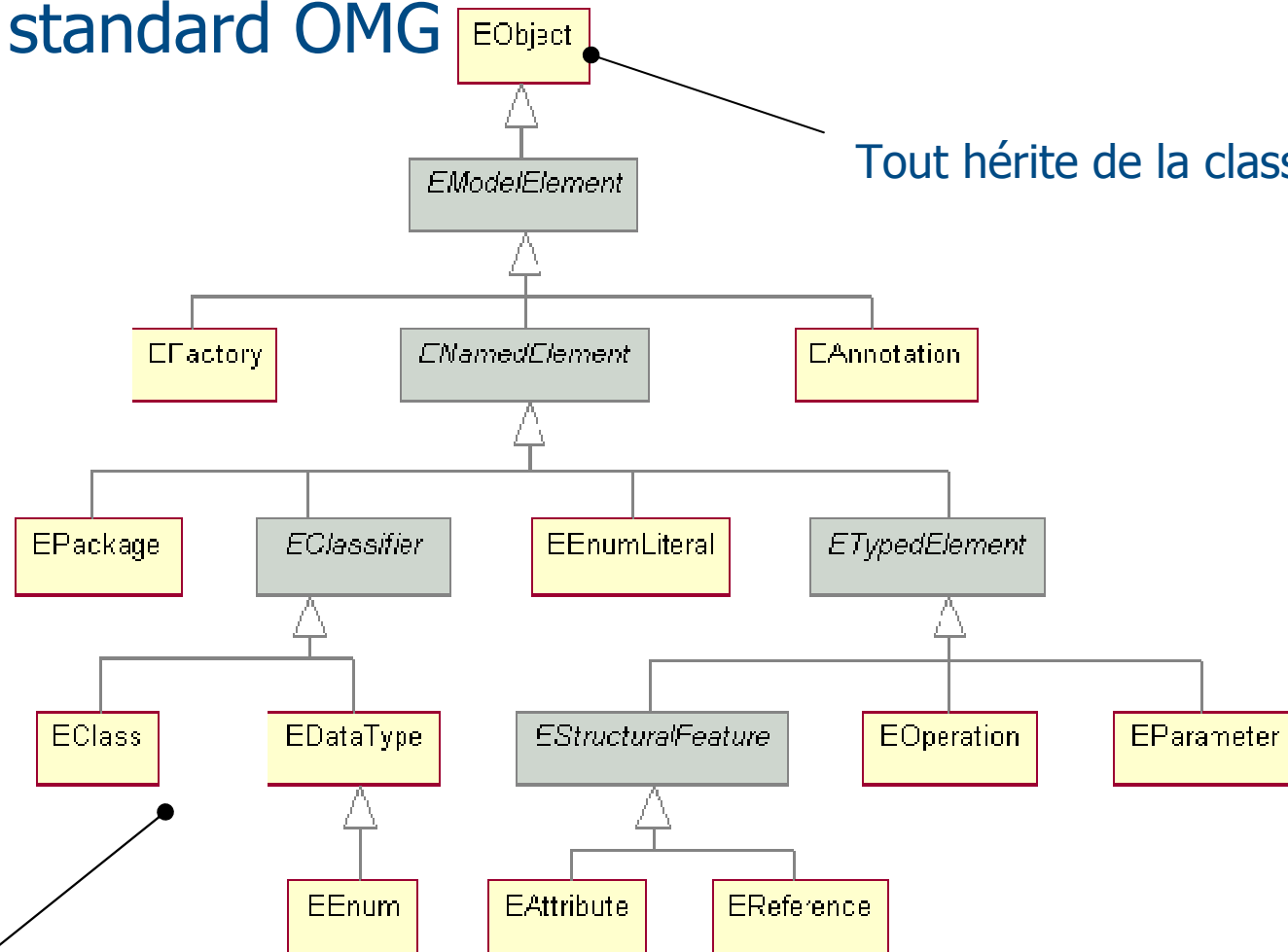


Modélisation avec EMF

- **Eclipse Modeling Framework (EMF)** est un framework Java et un outil de génération de codes pour construire des applications basées sur des modèles
- Lien du projet : *eclipse.org/modeling/emf/*
- EMF vous permettra de
 - Générer du code Java
 - Manipuler des modèles dynamiques (pas besoin de codes générés)
 - Interroger, créer et mettre à jour des instances de modèles
 - Sérialiser et désérialiser des instances
 - Valider des instances
 - Écouter les changements des instances

Modélisation avec EMF : Modèle Ecore

- EMF s'appuie sur le métamodèle Ecore qui respecte les principes définis par le eMOF (Extended Meta Object Facility) qui est un standard OMG



Tout hérite de la classe *EObject*

Modélisation avec EMF : Modèle Ecore

- **EClass** : désigne les classes des modèles, identifiées par un nom, peuvent contenir des *StructuralFeatures* (attributs ou références). Supporte l'**héritage multiple**, peut être **abstrait** (pas d'instance possible) ou une **interface** (pas d'implémentation générée)
- **EAttribute** : identifié par un **nom** et un **type**. Bornes mini et maxi sont utilisées pour la cardinalité
- **EReference** : association entre deux classes, identifiée par un nom et un type (une classe). **Relation inverse possible** (opposite). Bornes mini et maxi sont utilisées pour la cardinalité. Association de type **composition** autorisé (containment)

Modélisation avec EMF : Modèle Ecore

- **EDataType** : type primitif ou type objet défini par Java
- **EPackage** : désigne les packages des modèles qui sont des **conteneurs** de *classifiers* (classes et types). Défini par un nom de package (unique) et une URI pour l'identification lors de la sérialisation
- **EOperation** : désigne les opérations d'une classe pouvant être invoquées. Identifiée par un **nom**, un **type de retour** et des **paramètres**. Autorise les exceptions
- **EEnum** : désigne le types énumérés parmi un ensemble de **EEnumLiteral**

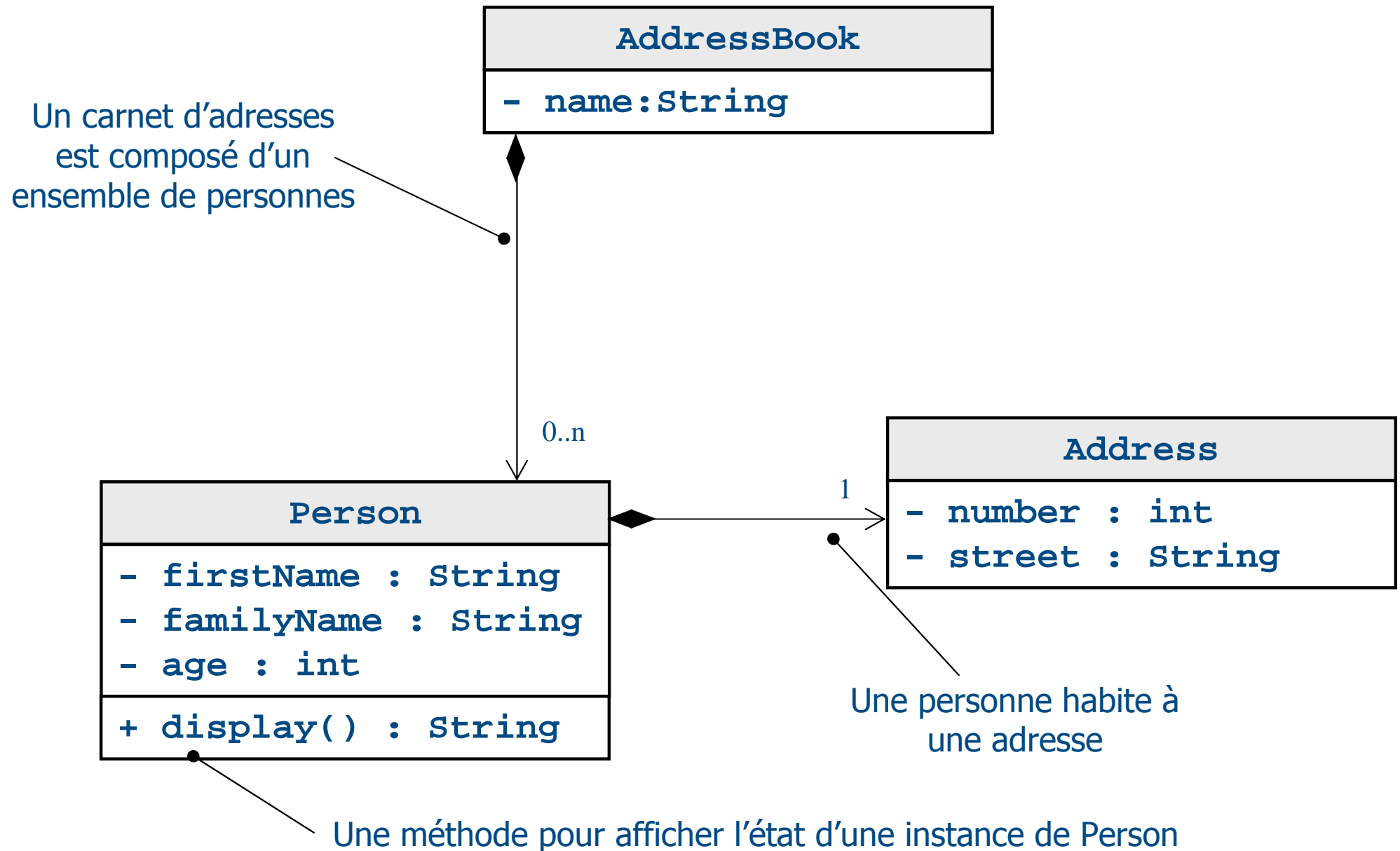
Modélisation avec EMF : Modèle Ecore

- Les éléments de type *StructuralFeatures* (attributs et références) contiennent des paramètres pour contrôler le code généré
 - *Unsettable* (true, **false**) : précise qu'une valeur d'un attribut n'a pas encore été déterminée (exemple : booléen true/false/undetermined)
 - *Containment* (true, **false**) : l'association est une composition
 - *Unique* (**true**, false) : pour les cardinalités multiples, précise qu'il ne peut y avoir la même valeur d'objet
 - *Changeable* (true, **false**) : valeur ne peut changer (pratique pour les relations inverses)
 - *Volatile* (true, **false**) : ne génère pas l'attribut pour stocker l'état, le corps de la méthode est également laissé à vide
 - *Transient* (true, **false**) : non persisté
 - *Derived* (true, **false**) : calculé à partir d'autres *StructuralFeatures* (attribut généralement marqué Volatile et Transient)

Modélisation avec EMF : Formats

- Pour construire un modèle EMF plusieurs formats disponibles
 - Modèle Ecore (voir la suite)
 - Classes Java annotées
 - Modèle de classes Rose
 - Modèle UML
 - XML Schema : les instances XMI seront conformes à l'XML Schema de départ
- Nous utiliserons par la suite un modèle Ecore puisque l'outillage fourni par EMF facilite la construction

Exemple : un carnet d'adresses



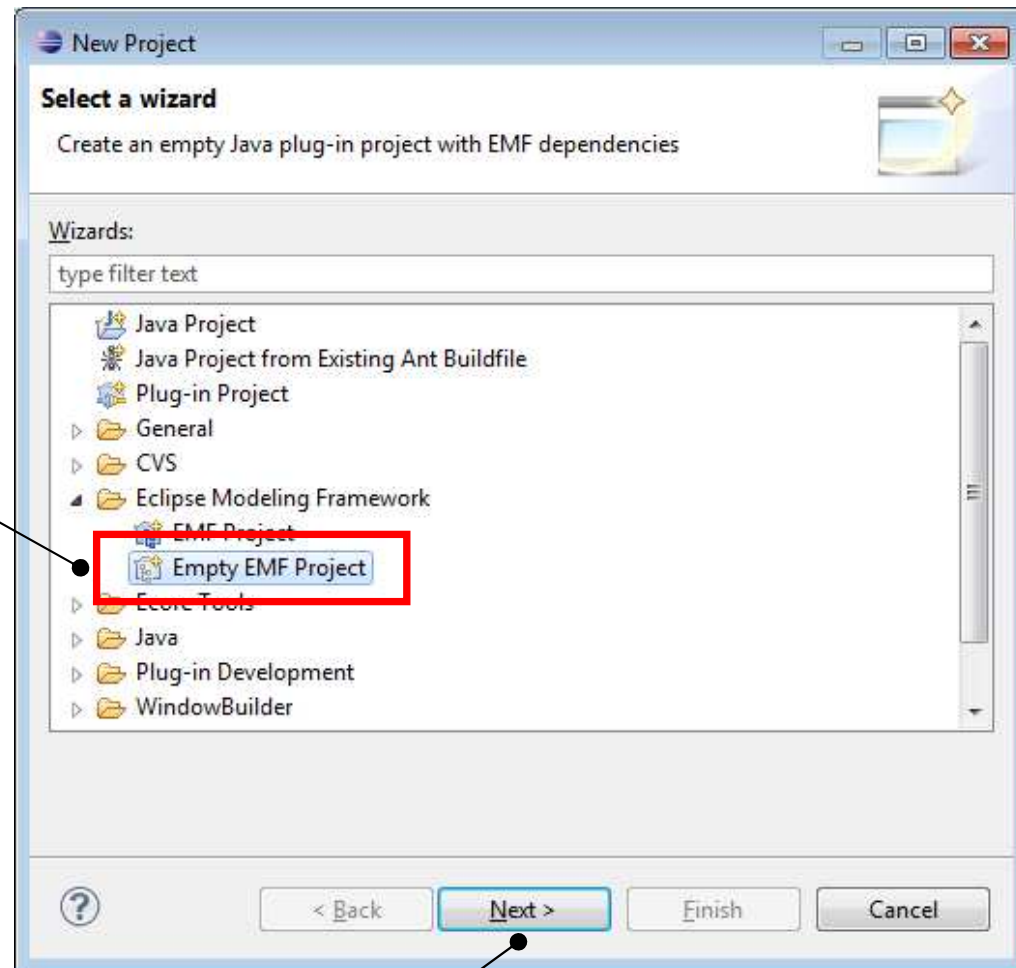
Définir un modèle EMF : Étapes de modélisation

- 1 Création du modèle EMF (extension .ecore)
- 2 Création du modèle de génération (extension .genmodel)
- 3 Paramétrer le modèle de génération
- 4 Génération des codes Java et de l'éditeur graphique
- 5 Création d'une configuration d'exécution
- 6 Création des instances

Définir un modèle EMF : Création du modèle

- Création d'un projet EMF vide (*File -> New -> Project...*)

Choisir *Eclipse Modeling Framework*
puis *Empty EMF Project*

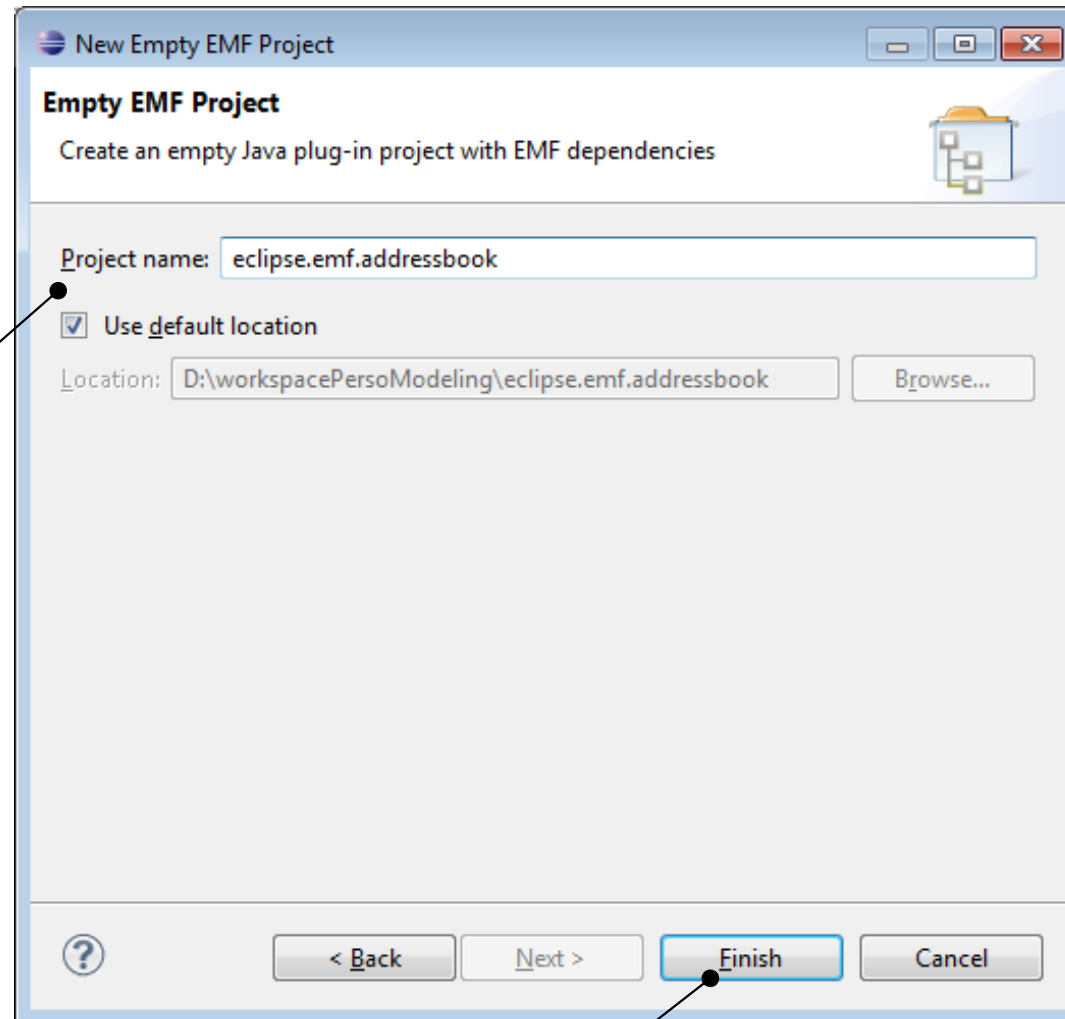


Puis faire *Next*

Définir un modèle EMF : Création du modèle

➤ Choisir le nom du projet EMF

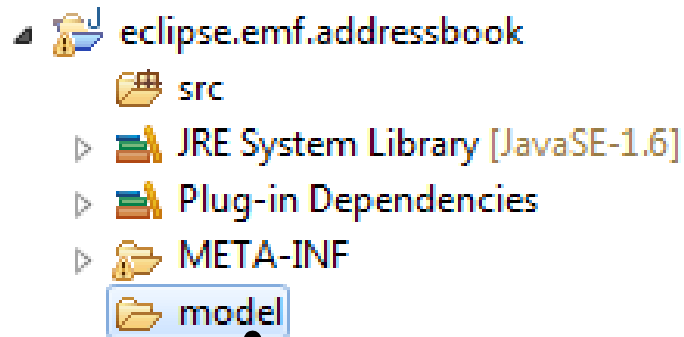
Choisir comme nom de projet
eclipse.emf.addressbook



Puis faire *Finish*

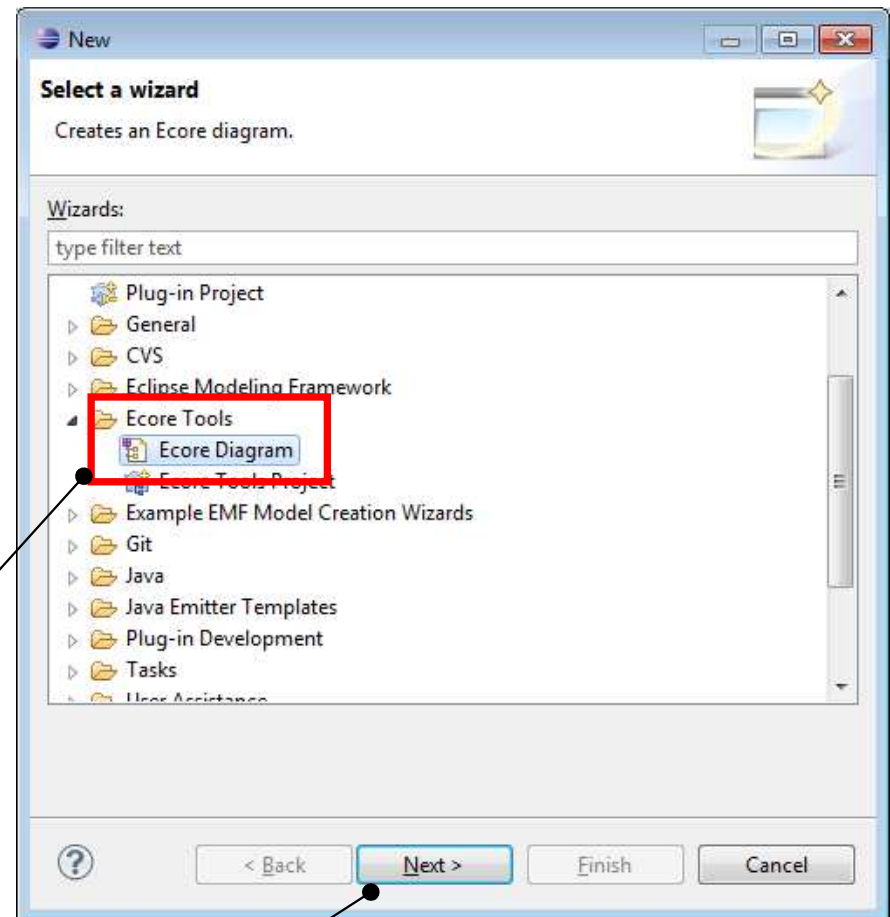
Définir un modèle EMF : Création du modèle

➤ Création d'un diagramme Ecore (inclus modèle Ecore)



Afficher le menu contextuel à partir du répertoire *model* puis faire *New -> Other ...*

Choisir *Ecore Tools* puis *Ecore Diagram*

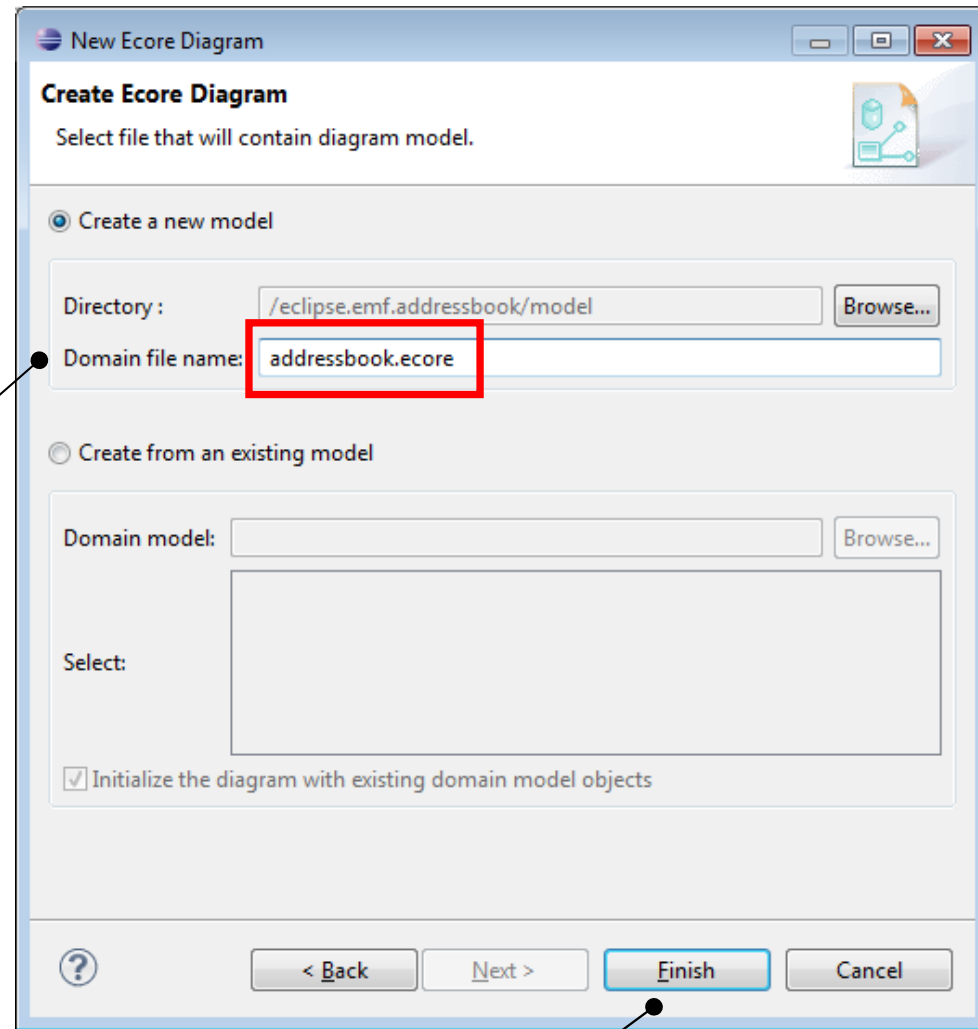


Puis faire *Next*

Définir un modèle EMF : Création du modèle

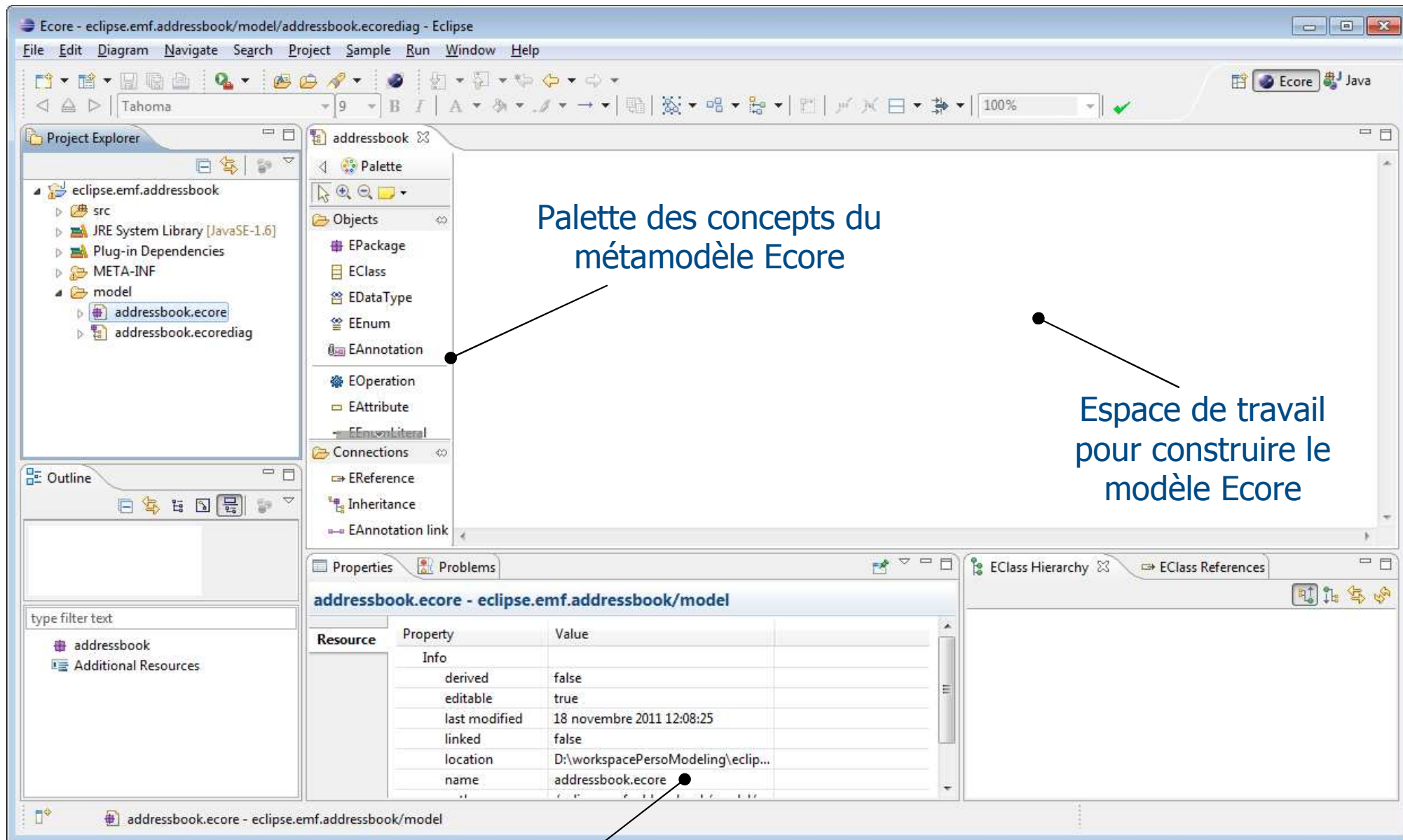
➤ Création d'un diagramme Ecore (suite)

Choisir comme nom de fichier de domaine *addressbook.ecore*



Puis faire *Finish*

Définir un modèle EMF : Création du modèle



Palette des concepts du métamodèle Ecore

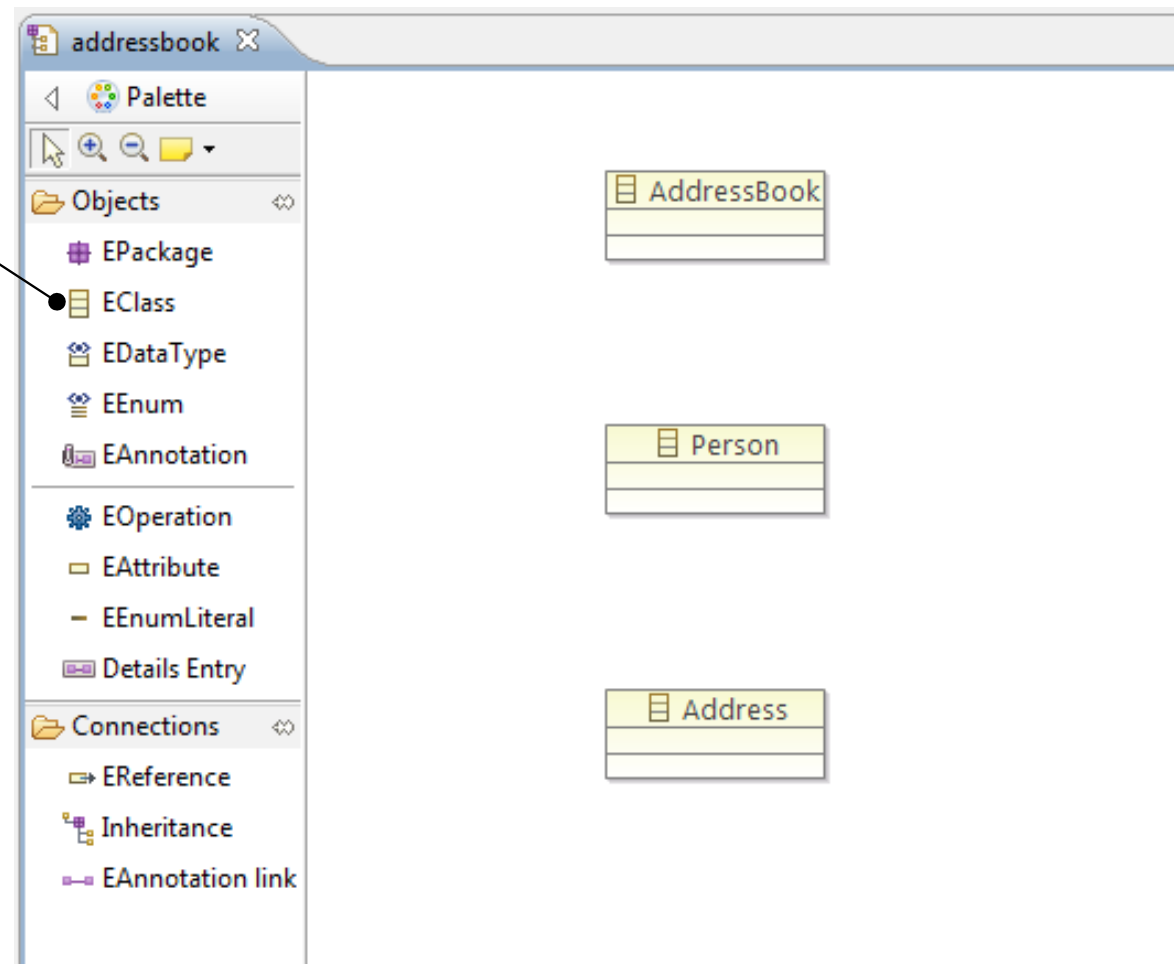
Espace de travail pour construire le modèle Ecore

Vue *Properties* qui permet d'éditer les éléments du modèle sélectionnés

Définir un modèle EMF : Création du modèle

➤ Création des classes

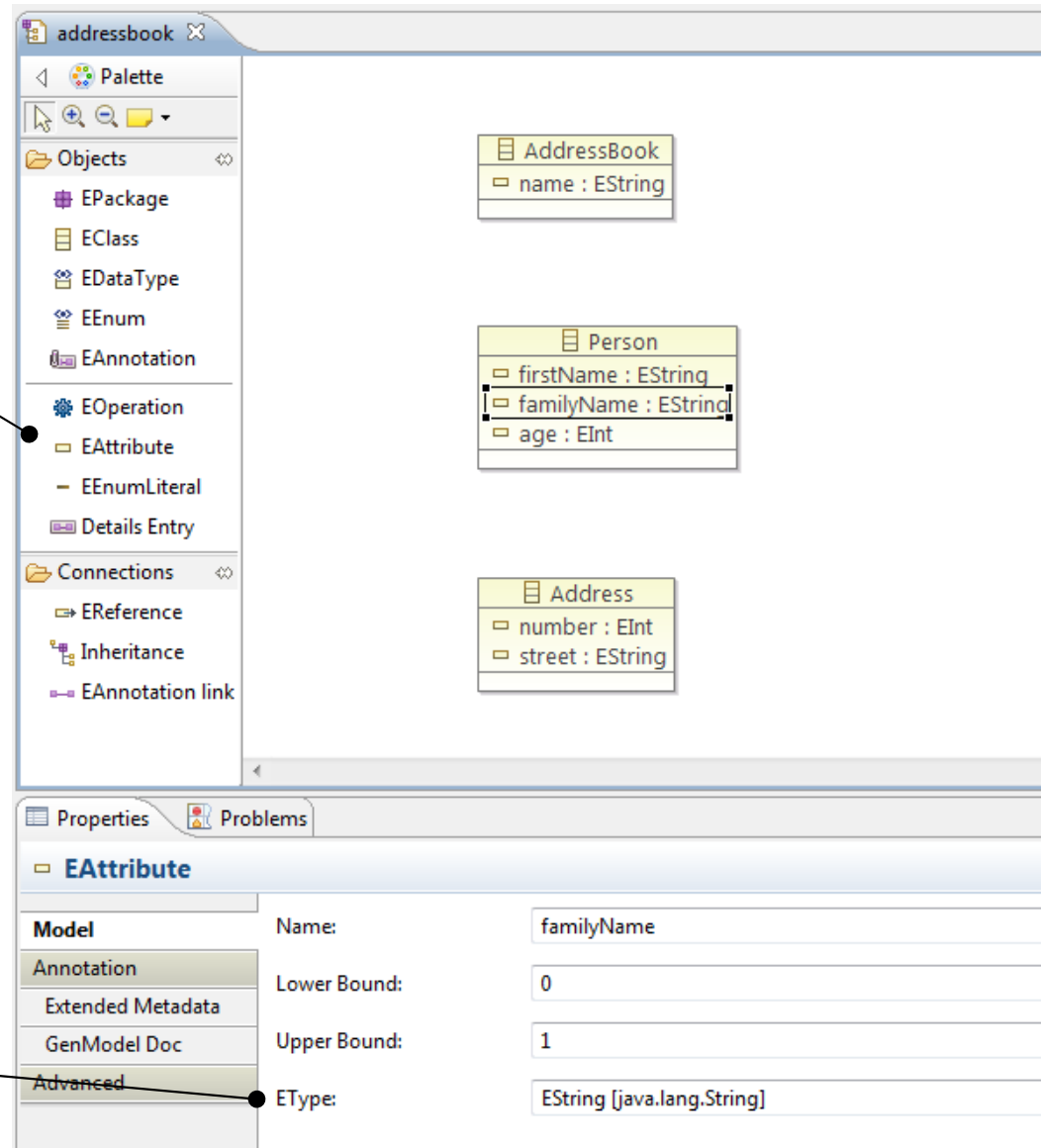
À partir de la palette
création de trois classes



Définir un modèle EMF : Création du modèle

► Création des attributs des classes

Pour chaque classe, création d'attributs



La vue *properties* permet de modifier le type de l'attribut sélectionné (*familyName*)

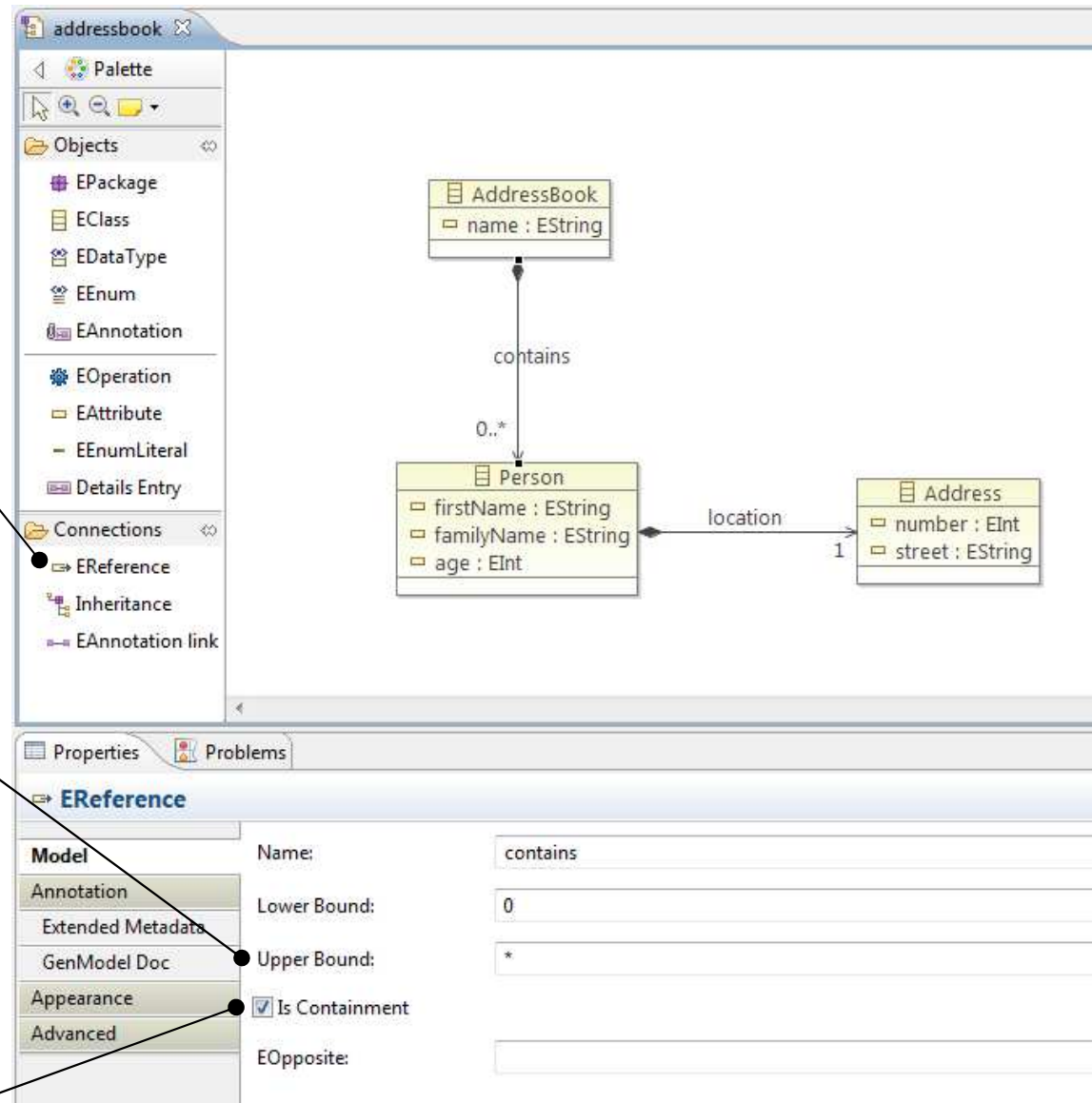
Définir un modèle EMF : Création du modèle

➤ Création des associations entre les classes

Création de deux associations par l'intermédiaire de *EReference*

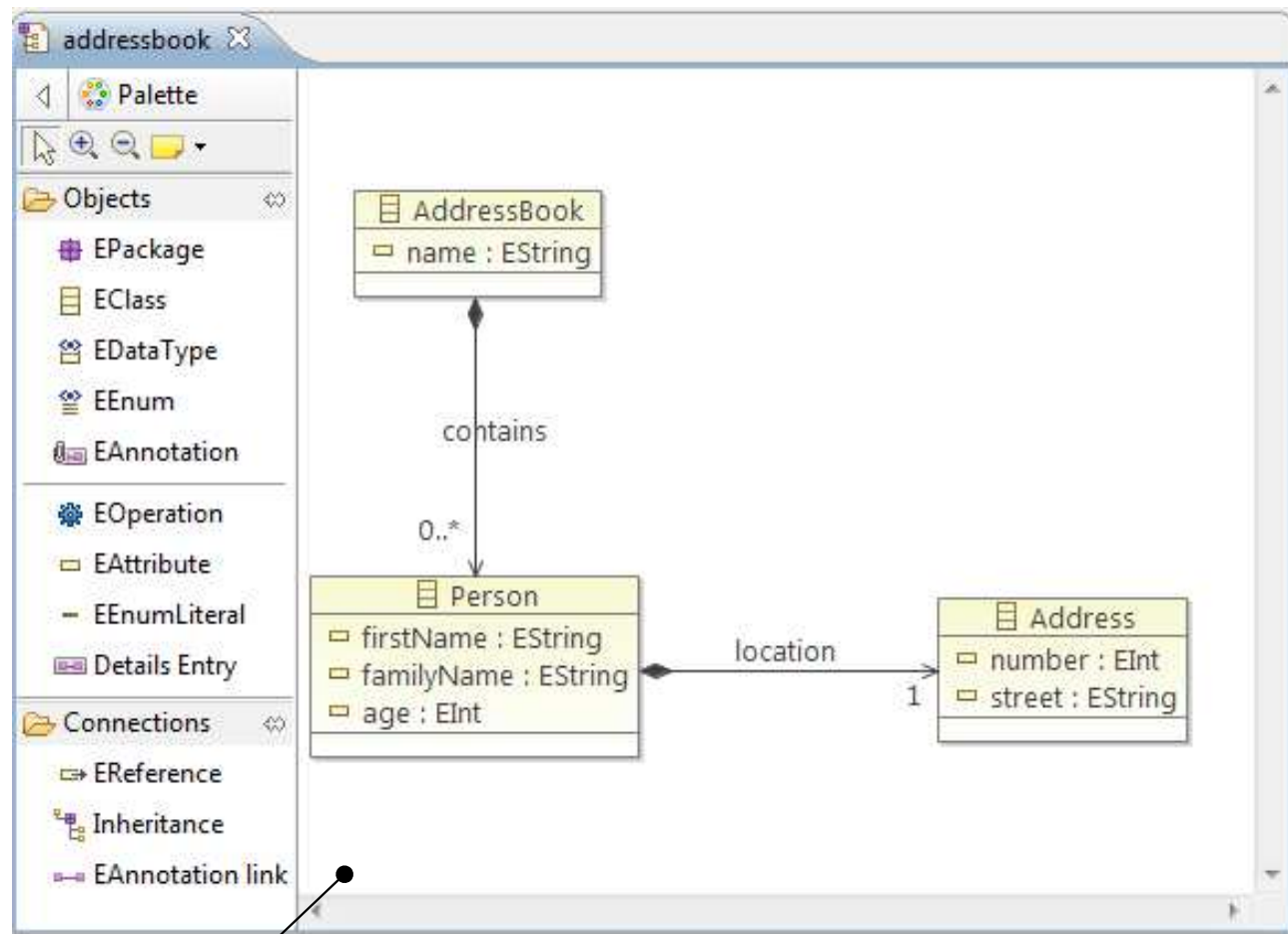
Pour préciser la cardinalité multiple placer -1 (traduit par *)

L'option *Is Containment* précise qu'il s'agit d'une composition



Définir un modèle EMF : Création du modèle

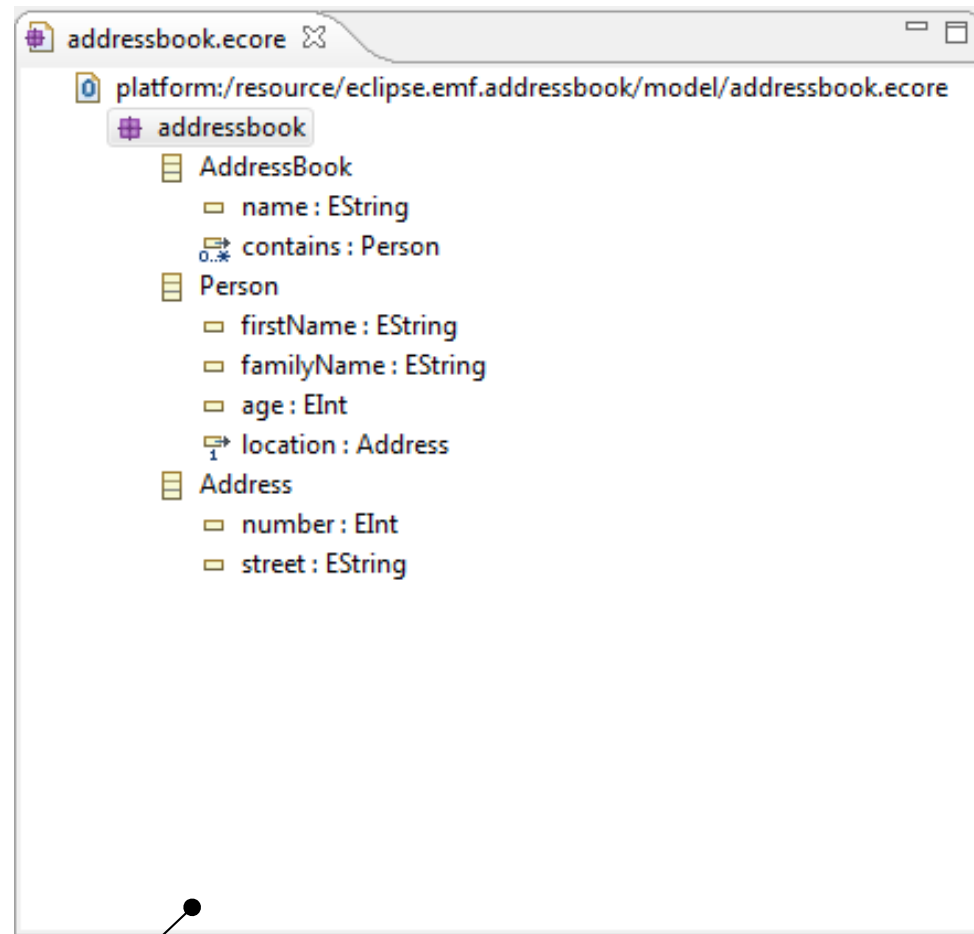
➤ Visualisation du modèle EMF : plusieurs représentations



Vue de type diagramme
(extension *ecorediag*)

Définir un modèle EMF : Création du modèle

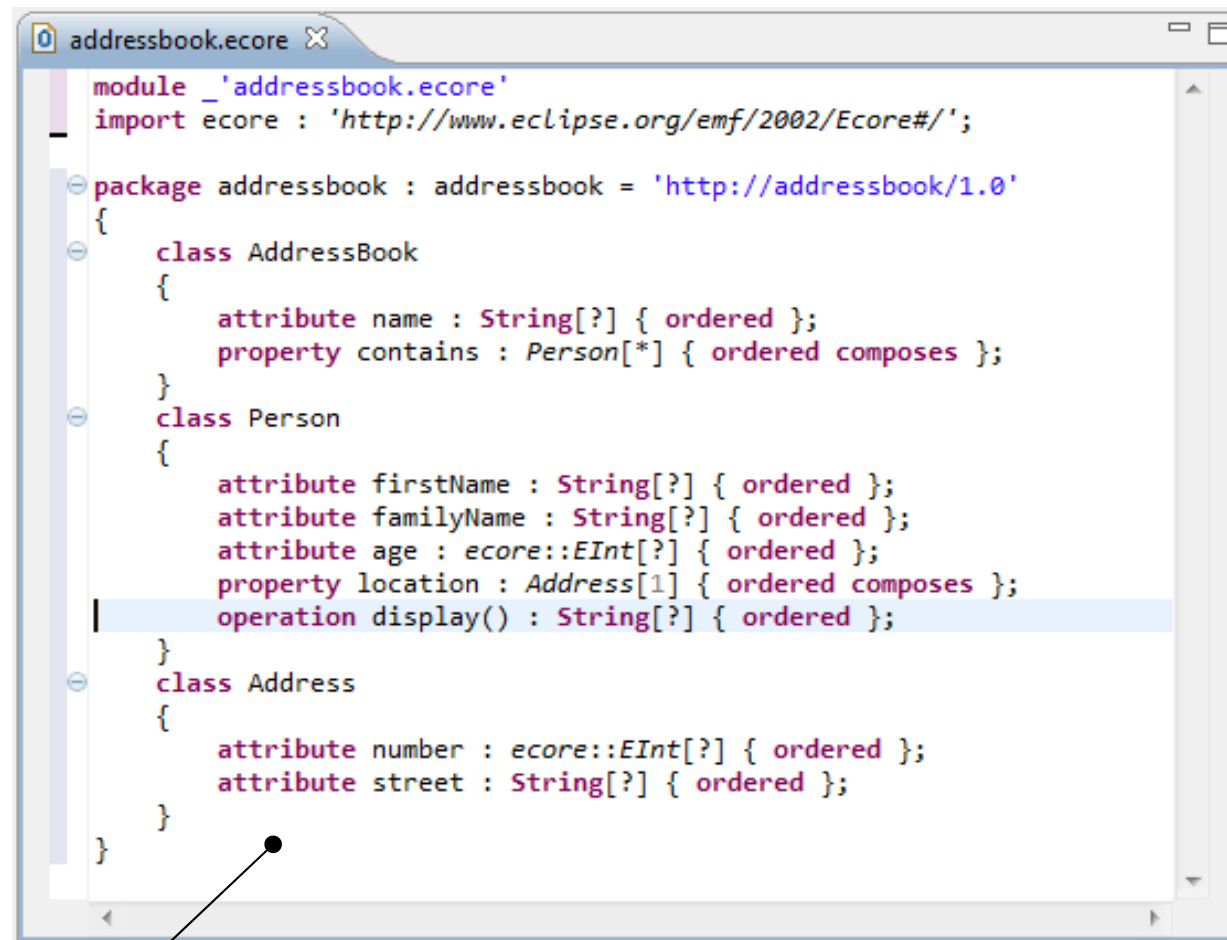
➤ Visualisation du modèle EMF (suite) : plusieurs représentations



Vue de type arbre (extension *ecore*)

Définir un modèle EMF : Création du modèle

➤ Visualisation du modèle EMF (suite) : plusieurs représentations



```

module _'addressbook.ecore'
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#';

package addressbook : addressbook = 'http://addressbook/1.0'
{
    class AddressBook
    {
        attribute name : String[?] { ordered };
        property contains : Person[*] { ordered composes };
    }
    class Person
    {
        attribute firstName : String[?] { ordered };
        attribute familyName : String[?] { ordered };
        attribute age : ecore::EInt[?] { ordered };
        property location : Address[1] { ordered composes };
        operation display() : String[?] { ordered };
    }
    class Address
    {
        attribute number : ecore::EInt[?] { ordered };
        attribute street : String[?] { ordered };
    }
}
    
```

Vue de type texte avec l'éditeur
OCLinEcore

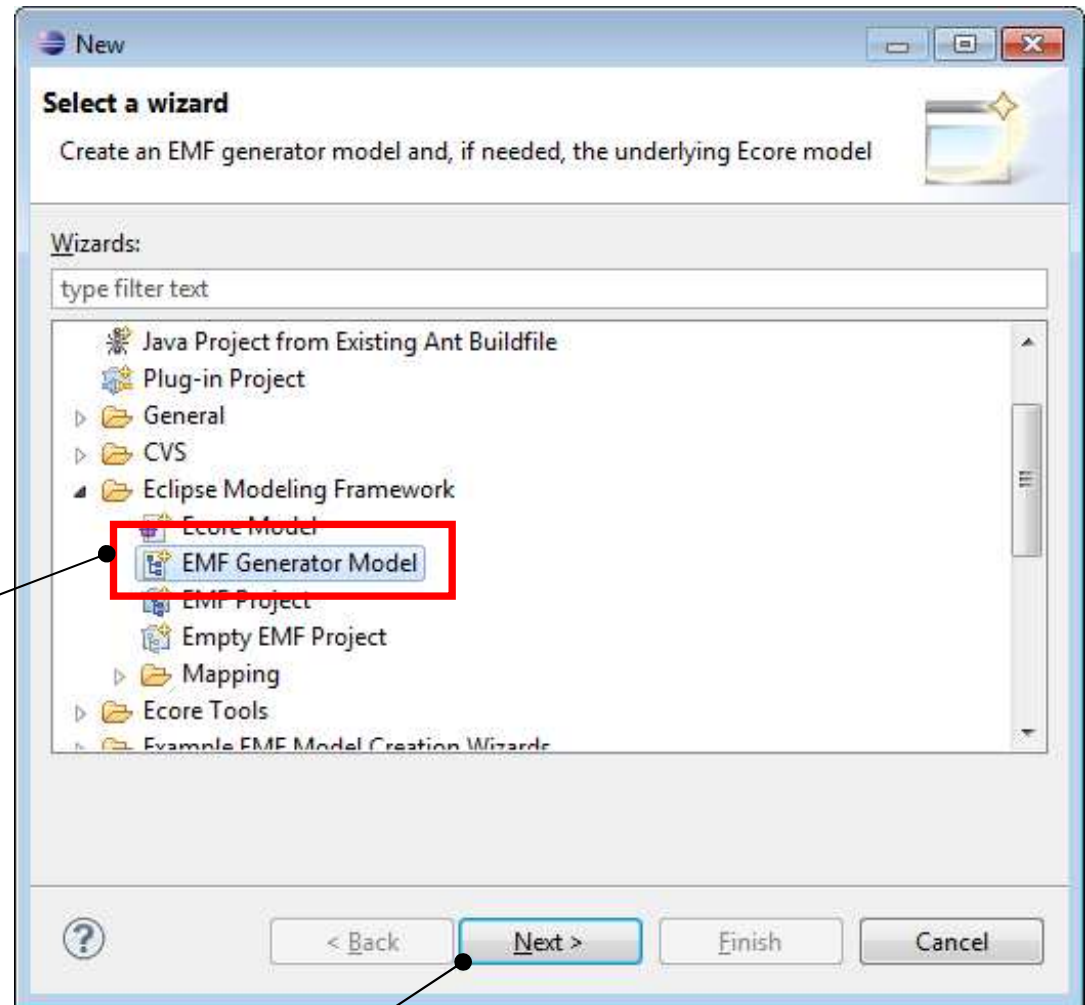
Définir un modèle EMF : Générer le code Java

- À partir du modèle défini précédemment, possibilité de générer du code Java dédié à la création des instances de ce modèle
- La génération de code nécessite la création d'un modèle de génération appelé *genmodel*
- Ce modèle contient des informations dédiées uniquement à la génération et qui ne pourraient pas être intégrées au modèle
 - Chemin de génération, package, préfixe...
- Le modèle *genmodel* est également un modèle EMF et chaque classe du modèle de génération est un décorateur des classes Ecore

Définir un modèle EMF : Générer le code Java

- Création d'un fichier *genmodel* utilisé pour la génération de code (*File -> New -> Other...*)

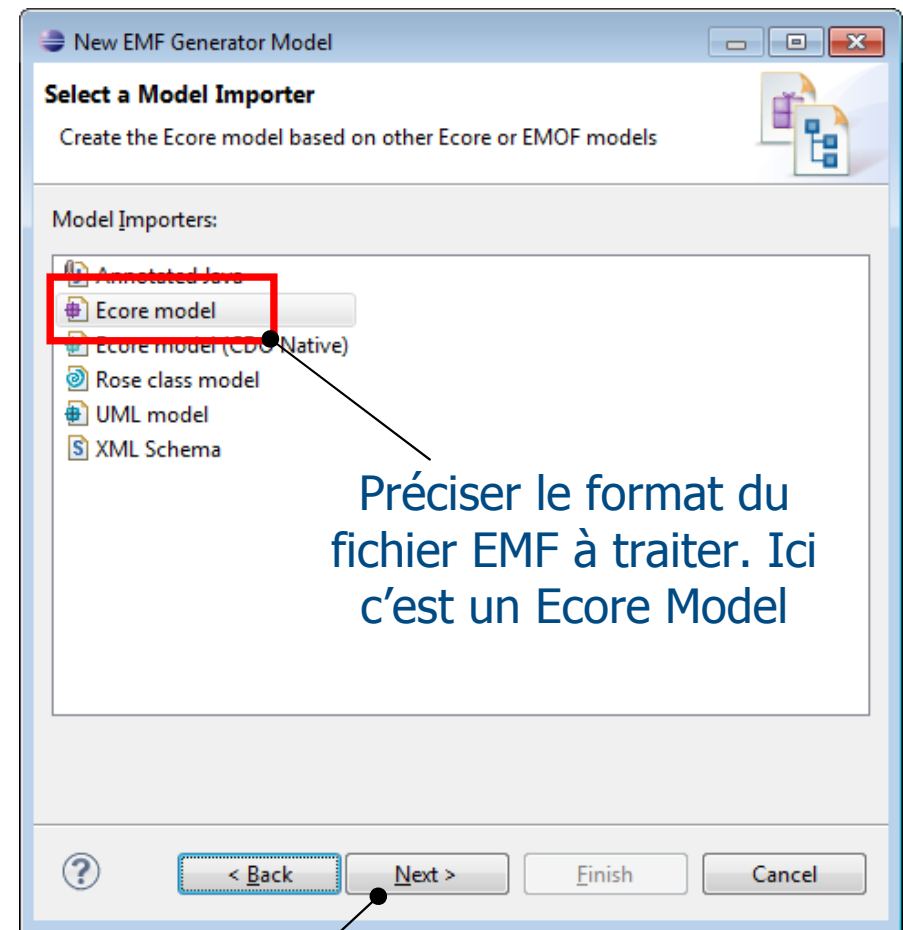
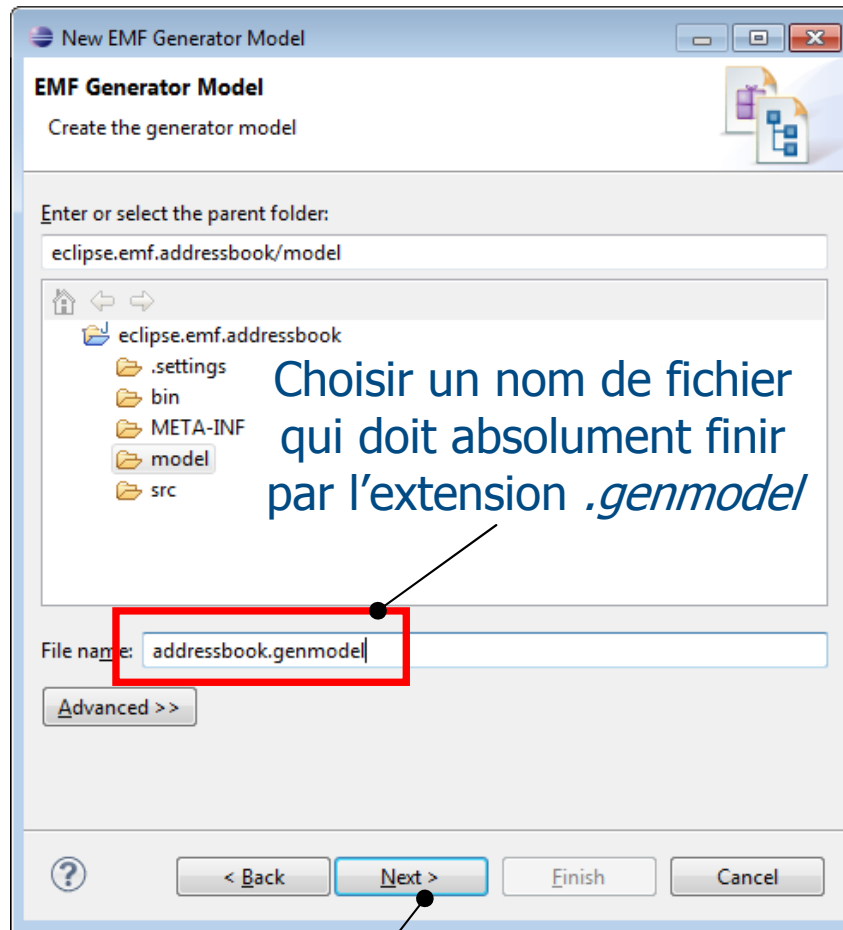
Choisir
Eclipse Modeling Framework
puis
EMF Generator Model



Puis faire *Next*

Définir un modèle EMF : Générer le code Java

- Création d'un fichier *genmodel* utilisé pour la génération de code (Suite)



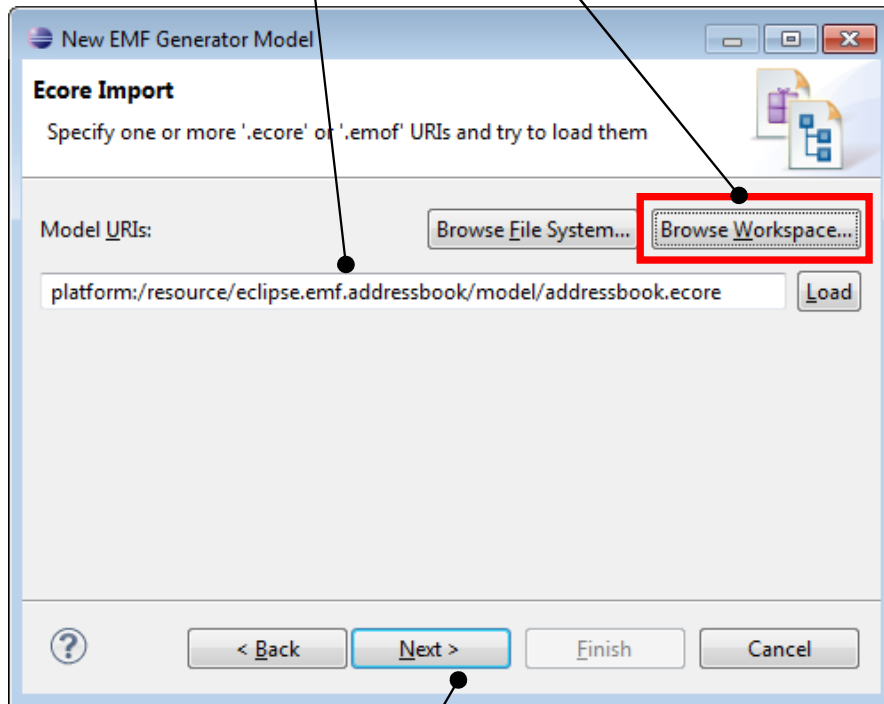
Puis faire *Next*

Puis faire *Next*

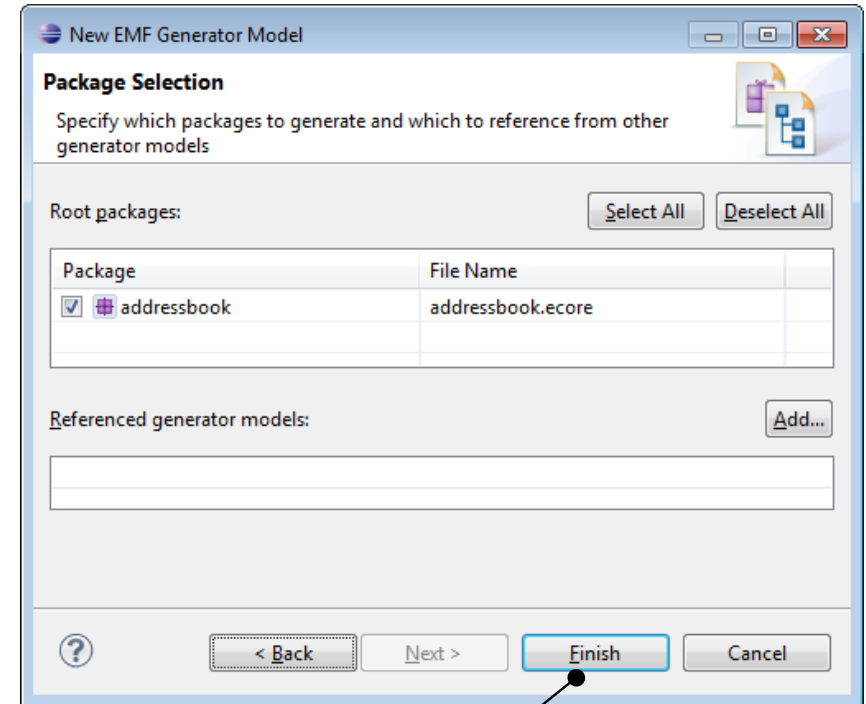
Définir un modèle EMF : Générer le code Java

- Création d'un fichier *genmodel* utilisé pour la génération de code (Suite)

Sélectionner le fichier *addressbook.ecore* en parcourant le contenu du Workspace



Puis faire *Next*



Enfin faire *Finish*

Définir un modèle EMF : Générer le code Java

➤ Configurer les informations de génération

Sélectionne l'élément package de plus haut niveau

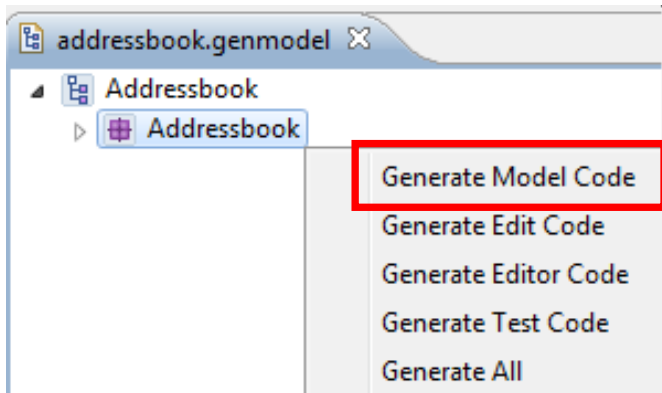
The screenshot shows the 'addressbook.genmodel' wizard. In the package list, the 'Addressbook' package is selected. The Properties view shows the 'Base Package' property set to 'eclipse.emf.addressbook.model'.

Property	Value
All	
Base Package	eclipse.emf.addressbook.model
Prefix	Addressbook
Ecore	
Package	addressbook
Edit	
Child Creation Extenders	false
Disposable Provider Factory	true
Extensible Provider Factory	false
Editor	
Generate Model Wizard	true

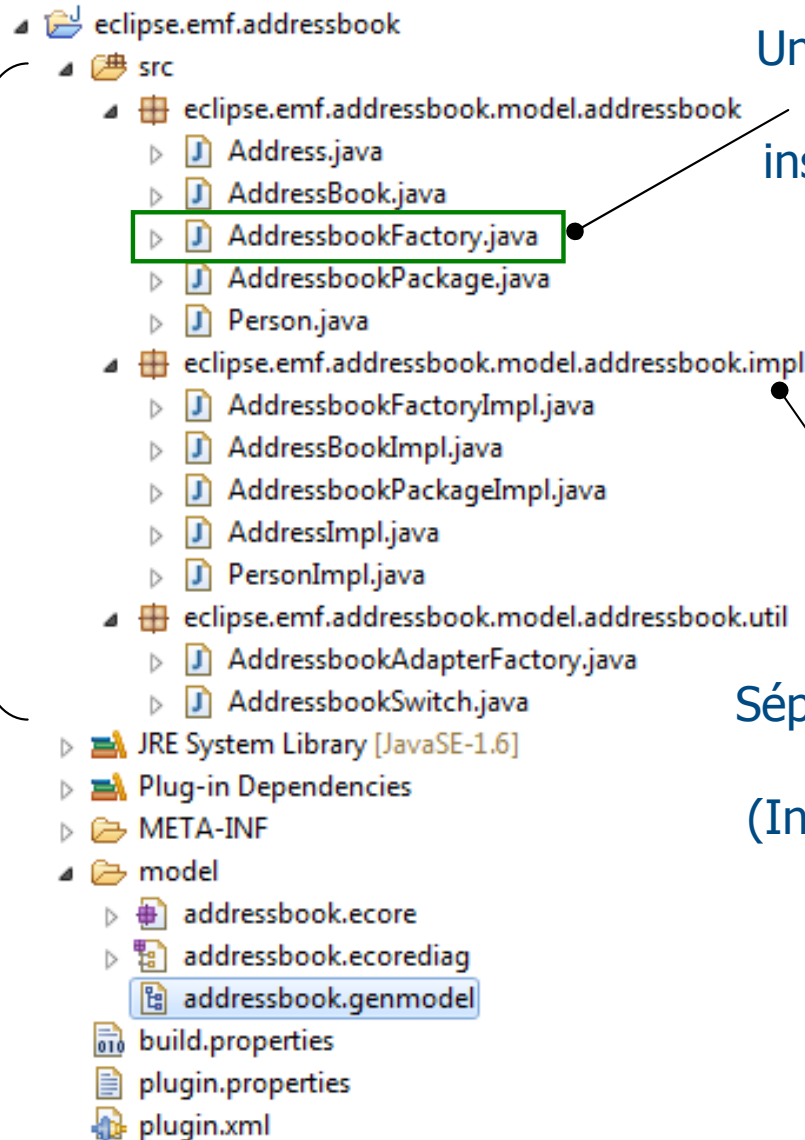
Pour préciser le package de génération (\$BASE_PACKAGE + addressbook)

Définir un modèle EMF : Générer le code Java

➤ Génération des classes du domaine (*Generate Model Code*)



Un ensemble de classes correspondant au modèle EMF a été généré



Une fabrique est utilisée pour construire des instances *AddressBook*, *Address* et *Person*

Séparation stricte entre la partie contrat (Interface) et la partie implémentation

Définir un modèle EMF : Mise à jour du code Java

- Si des modifications sont à apporter au modèle EMF, le code généré Java devra être mis à jour
 - Mise à jour du genmodel (*Reload*)
 - Génération du code Java (*Generate Model Code*)
- Seules les déclarations annotées (classe, interface, attribut, méthode) avec *@generated* seront régénérées

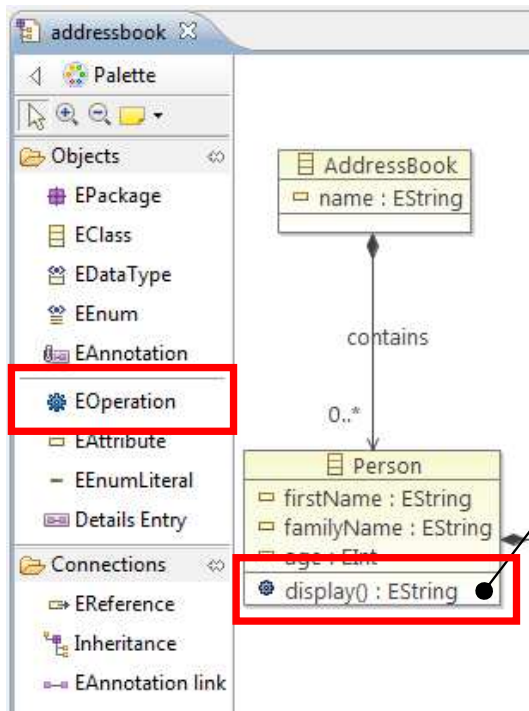
```
/**
 * ...
 * @see eclipse.emf.addressbook.model.addressbook.AddressbookPackage#getAddress()
 * @model
 * @generated
 */
public interface Address extends EObject {
    /**
     * ...
     * @model
     * @generated
     */
    int getNumber();
}
```

L'annotation *@generated* indique aux générateurs que le code associé peut être régénéré

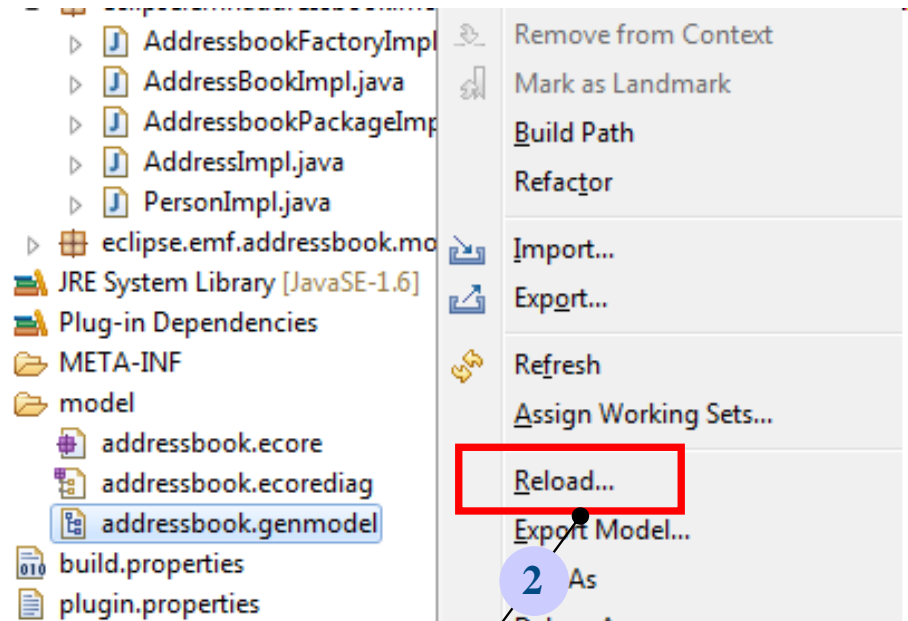
Interface *Address* du projet **addressbook**

Définir un modèle EMF : Mise à jour du code Java

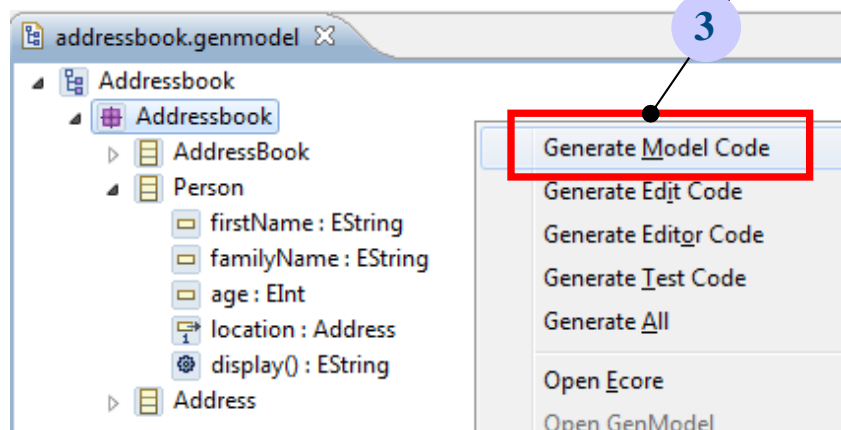
➤ Exemple : ajouter une opération *String display()* dans *Person*



Ajout d'une opération *String display()* dans la classe *Person*



Mise à jour de *genmodel* via la commande *Reload...*



Régénération du code Java

```
/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @model
 * @generated
 */
String display();
```

Définir un modèle EMF : Mise à jour du code Java

- Il peut être utile également d'apporter des modifications dans le code Java généré
 - Implémenter le corps d'une opération
 - Personnaliser l'affichage de l'état d'une classe
 - Ajouter de nouveaux attributs...
- Pour empêcher qu'une déclaration soit impactée par la mise à jour de la génération, plusieurs solutions :
 - Supprimer l'annotation *@generated*
 - Ajouter le mot NOT après *@generated* (*@generated NOT*)
 - Ajouter un suffixe **Gen** à la fin d'une méthode et implémenter sa propre version

Définir un modèle EMF : Mise à jour du code Java

➤ Exemple : ajouter son propre code dans la version générée

```
public class PersonImpl ... {  
    ...  
    public String displayBis() {  
        return this.toString();  
    }  
  
    /**  
     * ...  
     * @generated NOT  
     */  
    public String display() {  
        return this.toString();  
    }  
  
    /**  
     * ...  
     * @generated  
     */  
    public Address getLocationGen() {  
        return location;  
    }  
  
    public Address getLocation() {  
        System.out.println("PersonImpl.getLocation()");  
        return this.getLocationGen();  
    }  
}
```

Méthode ajoutée explicitement dans cette classe et ne sera pas impactée par la régénération

Méthode ajoutée par la génération. Le contenu ne sera pas mis à jour par la régénération

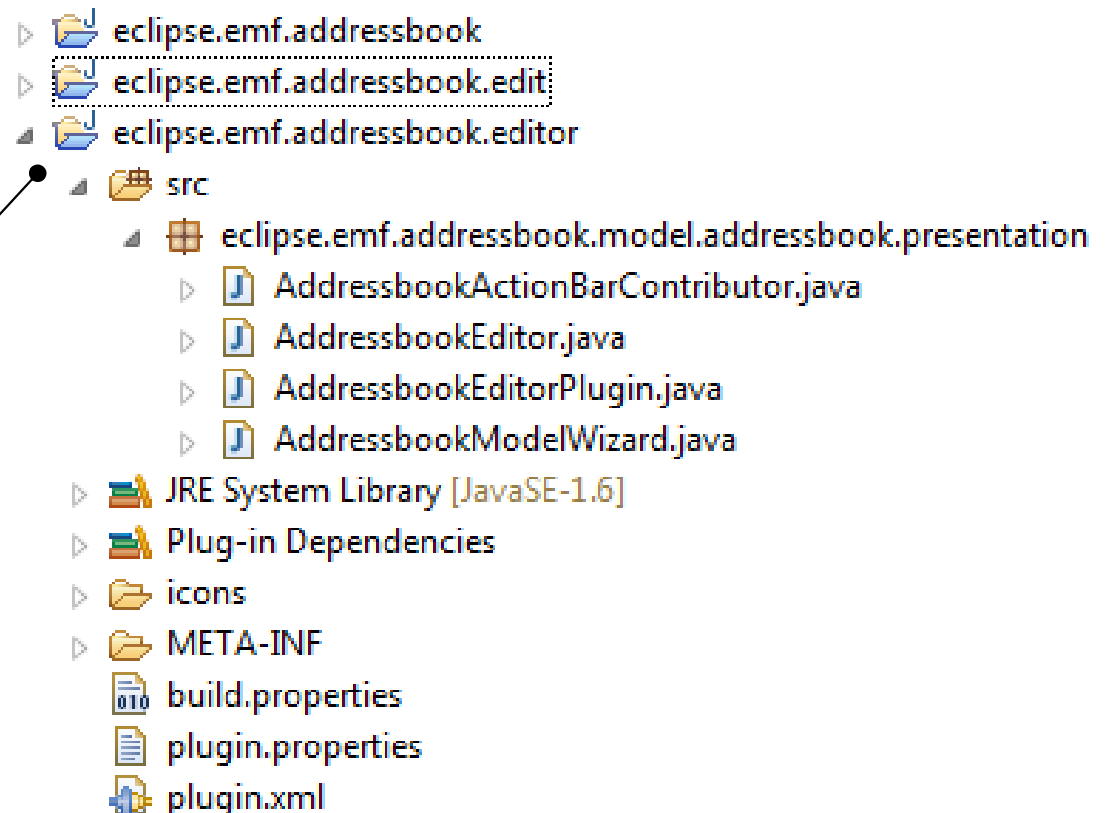
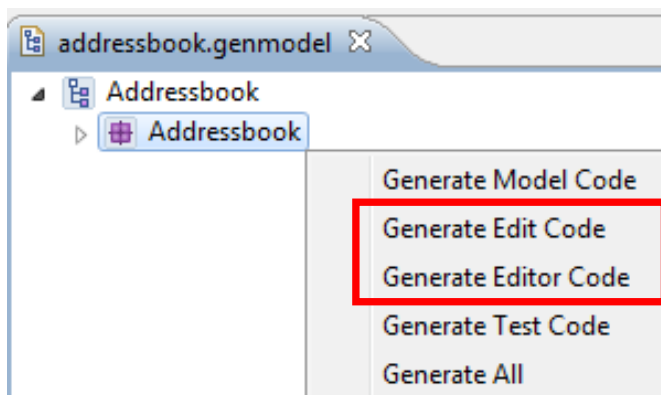
Méthode qui masque *getLocation()* et qui sera modifiée à chaque régénération

Méthode ajoutée explicitement et ne sera pas impactée par la régénération. Utilisation de la délégation pour accéder au contenu réel de *getLocation()*

Interface *PersonImp* du projet **addressbook**

Définir un modèle EMF : Générer le code de l'éditeur

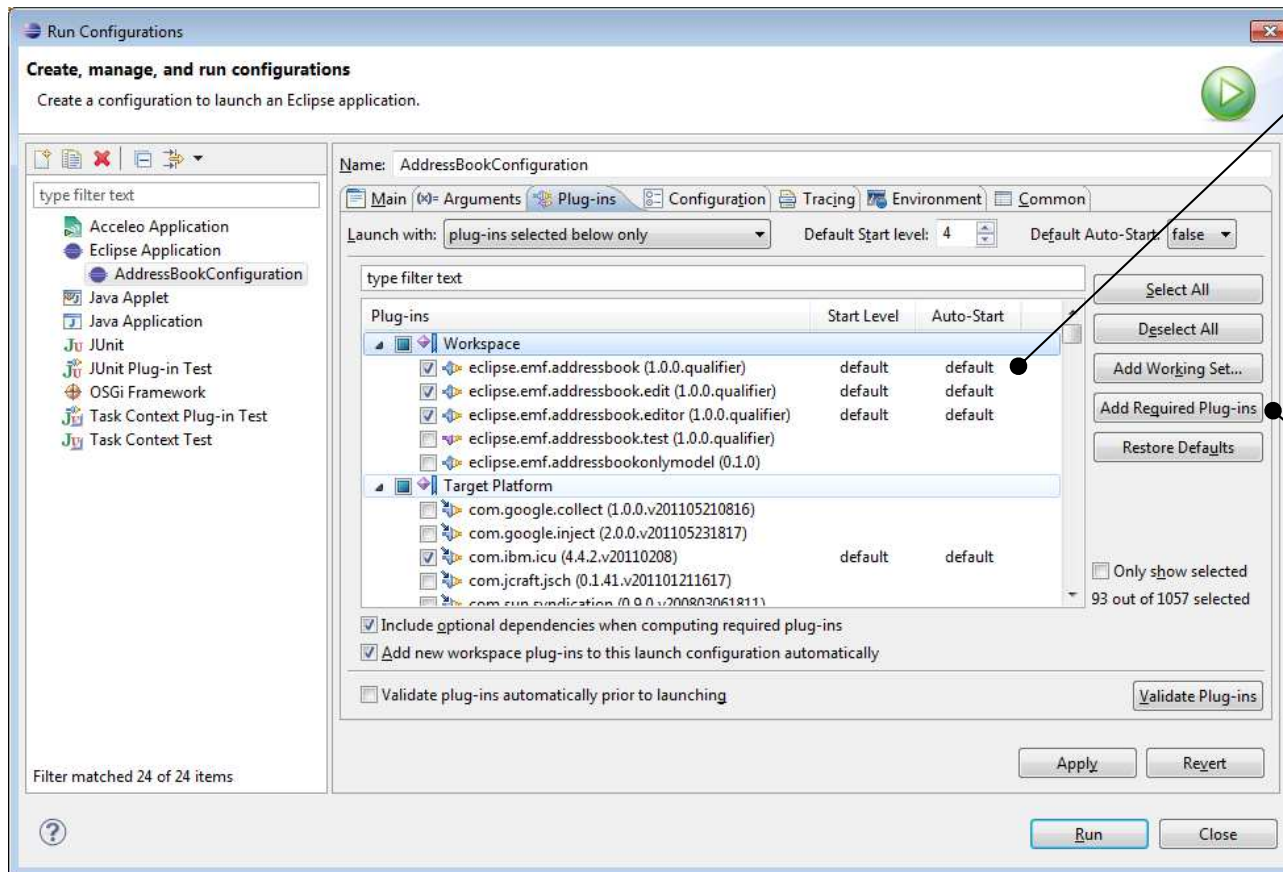
- Génération d'un éditeur (*Generate Edit and Editor Code*)
 - Les outils de la suite EMF permettent de générer automatiquement un éditeur pour construire les instances du modèle



Deux plugins ont été ajoutés et tous les codes Java ont été générés

Définir un modèle EMF : Exécuter le code de l'éditeur

- Création d'une configuration d'exécution en intégrant les trois plugins



Associer les trois plugins

N'oubliez pas de faire un
Add Required Plug-ins

Penser à ajouter le plugin
org.eclipse.ui.ide.workbench



Définir un modèle EMF : Exécuter le code de l'éditeur

► Exécution d'une configuration d'exécution

Création d'un simple projet

Création d'une instance du modèle via l'assistant

Les instances sont construites via l'éditeur

The screenshot displays the Eclipse IDE interface. The Project Explorer on the left shows a project named 'AddressBookInstancesSample' containing a resource 'First.addressbook'. The Resource Set on the right shows a tree structure for 'platform:/resource/AddressBookInstancesSample/First.addressbook' with a folder 'Address Book Mon Carnet d'Adresses' containing four instances: 'Person Mickael', 'Address 5', 'Person John', and 'Address 6'. The Properties view at the bottom right shows the 'Name' property of 'Address 5' set to 'Mon Carnet d'Adresses'. A 'New' wizard dialog is open in the foreground, showing the 'Addressbook Model' wizard selected under 'Example EMF Model Creation Wizards'.

Instancier un modèle par les API EMF

- Les instances de notre modèle peuvent être construites en utilisant directement le code généré
- Toutes les instances des classes *Address*, *AddressBook* et *Person* doivent être construites à partir de la **fabrique** *AddressBookFactory* (générée automatiquement)
- Il n'est pas recommandé d'utiliser directement les classes d'implémentation générées par Eclipse
- Toutes les classes du modèle générées héritent directement ou indirectement de la classe *EObject*

Instancier un modèle par les API EMF

➤ Exemple : instantiation d'un carnet d'adresses

```
@Test
public void createAddressBook() {
    AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
    createAddressBook.setName("Mon Carnet d'Adresses");

    Address createMBAddress = AddressbookFactory.eINSTANCE.createAddress();
    createMBAddress.setNumber(5);
    createMBAddress.setStreet("Rue des Javaistes");
    Person mickaelBaron = AddressbookFactory.eINSTANCE.createPerson();
    mickaelBaron.setAge(35);
    mickaelBaron.setFamilyName("BARON");
    mickaelBaron.setFirstName("Mickael");
    mickaelBaron.setLocation(createMBAddress);
    Assert.assertEquals("BARON", mickaelBaron.getFamilyName());

    Address createAddress = AddressbookFactory.eINSTANCE.createAddress();
    Person johnAaron = AddressbookFactory.eINSTANCE.createPerson();
    ...
    Assert.assertEquals("Rue des Espions", johnAaron.getLocation().getStreet());

    createAddressBook.getContains().add(mickaelBaron);
    createAddressBook.getContains().add(johnAaron);
    Assert.assertEquals(2, createAddressBook.getContains().size());
}
```

Instance de
AddressBook obtenue
via la factory

Classe *AddressBookTest* du fragment
addressbook.test

Sauvegarder et charger des instances du modèle

- EMF fournit une API pour persister les instances d'un modèle vers un fichier au format XMI
- L'unité de base pour la persistance est appelée *resource* qui est un conteneur pour des instances d'un modèle
- Une *resource* est manipulée via l'interface *Resource*
- Afin de gérer les relations entre les instances de différentes ressources (références), l'interface *ResourceSet* est utilisée comme conteneur de ressources
- Les API de persistance nécessitent une dépendance vers le plugin *org.eclipse.emf.ecore.xmi*

Sauvegarder et charger des instances du modèle

► Exemple : sauvegarder les instances d'un modèle

```
@Test
public void saveAndLoadAddressBook() {
    // Création des instances (voir exemple précédent)
    ...

    // Save Model to XMI
    ResourceSet resourceSet = new ResourceSetImpl();
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put("xmi", new XMIResourceFactoryImpl());
    URI uri = URI.createURI("file:/c:/addressbookinstances.xmi");
    Resource resource = resourceSet.createResource(uri);
    resource.getContents().add(createAddressBook);
    try {
        resource.save(null);
        Assert.assertEquals("Mon Carnet d'Adresses", createAddressBook.getName());
    } catch (IOException e) {
        e.printStackTrace();
    }
    ...
}
```

Obligatoire pour
déterminer le type de
ressource à charger

Classe *AddressBookTest* du
fragment **addressbook.test**

```
<?xml version="1.0" encoding="ASCII"?>
<addressbook:AddressBook xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:addressbook="http://addressbook/1.0" name="Mon Carnet d'Adresses">
    <contains firstName="Mickael" familyName="BARON" age="35">
        <location number="5" street="Rue des Javaistes"/>
    </contains>
    <contains firstName="John" familyName="AARON" age="14">
        <location number="6" street="Rue des Espions"/>
    </contains>
</addressbook:AddressBook>
```

addressbookinstances.xmi

Sauvegarder et charger des instances du modèle

➤ Exemple : charger les instances d'un modèle

```
@Test
public void saveAndLoadAddressBook() {
    // Création des instances (voir exemple précédent)
    ...

    // Save Model to XMI
    ...

    // Load model from XMI.
    resourceSet = new ResourceSetImpl();
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put("xmi", new XMIResourceFactoryImpl());
    uri = URI.createURI("file:/c:/addressbookinstances.xmi");
    resource = resourceSet.createResource(uri);
    try {
        resource.load(null);
        createAddressBook = (AddressBook)resource.getContents().get(0);
        Assert.assertEquals("Mon Carnet d'Adresses", createAddressBook.getName());
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Classe *AddressBookTest* du
fragment **addressbook.test**



**Suite dans le prochain
transparent**

Sauvegarder et charger des instances du modèle

➤ Exemple (bis) : charger les instances d'un modèle

```
@Test
public void saveAndLoadAddressBook() {
    // Création des instances (voir exemple précédent)
    ...

    // Save Model to XMI
    ...

    // Load model from XMI.
    ...

    // Load model from XMI in a second way.
    resourceSet = new ResourceSetImpl();
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put("xmi", new XMIResourceFactoryImpl());
    uri = URI.createURI("file:/c:/addressbookinstances.xmi");
    resource = resourceSet.getResource(uri, true);
    createAddressBook = (AddressBook)resource.getContents().get(0);
    Assert.assertEquals("Mon Carnet d'Adresses", createAddressBook.getName());
}
```

Classe *AddressBookTest* du
fragment **addressbook.test**

Effectue un chargement
de la ressource

Manipuler le métamodèle EMF

- Pour rappel un modèle EMF est une instance du métamodèle Ecore
- Il est donc possible d'interroger le métamodèle Ecore pour connaître la structuration d'un modèle EMF
- Par conséquent tous les accès se font par les classes du modèle Ecore (*EClass*, *EOperation*, *EAttribute*...)
- Pour accéder aux instances Ecore à partir des classes générées, il faut utiliser la classe décrivant le package
 - *AddressbookPackage.eInstance* : package spécifique au modèle
- Où *AddressbookPackage* est une classe héritant de *EPackage*
- Toutes les méthodes spécifiques à Ecore débutent par *getE...*

Manipuler le métamodèle EMF

► Exemple : interroger le métamodèle EMF

```
@Test
public void queryAddressBookStructure() {
    AddressbookPackage addressbookPackage = AddressbookPackage.eINSTANCE;
    EList<EClassifier> eClassifiers = addressbookPackage.getEClassifiers();
    for (EClassifier eClassifier : eClassifiers) {
        System.out.println(eClassifier.getName());
        System.out.print("  ");
        if (eClassifier instanceof EClass) {
            EClass eClass = (EClass) eClassifier;
            EList<EAttribute> eAttributes = eClass.getEAttributes();
            for (EAttribute eAttribute : eAttributes) {
                System.out.print(
                    eAttribute.getName() + "(" + eAttribute.getEAttributeType().getName() + ") ";
            }
            if (!eClass.getEAttributes().isEmpty() && !eClass.getEReferences().isEmpty()) {
                System.out.println();
                System.out.print("  Références : ");
            }
            EList<EReference> eReferences = eClass.getEReferences();
            for (EReference eReference : eReferences) {
                System.out.print(
                    eReference.getName() + "("
                    + eReference.getEReferenceType().getName()
                    + "[" + eReference.getLowerBound() + ".."
                    + eReference.getUpperBound() + "]" + ")";
            }
        }
    }
}
```

```
Console
<terminated> AddressBookTest.queryAddressBookStructure [JUnit] C
AddressBook
  name(EString)
  Références : contains(Person[0..-1])
Person
  firstName(EString) familyName(EString) age(EInt)
  Références : location(Address[1..1])
  Opérations : EString display
Address
  number(EInt) street(EString)
```

Classe *AddressBookTest* du fragment
addressbook.test

Manipuler le métamodèle EMF

- Pour l'instant nous avons vu que pour créer des instances du modèle nous utilisons les classes générées

```
AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
```

- Par réflexivité, il est possible de créer et modifier des instances d'un modèle sans avoir à manipuler explicitement les classes générées

```
EClass eClass = (EClass)ePackage.getEClassifier("AddressBook");
```

- L'intérêt est de pouvoir créer des instances sans avoir à générer les classes Java associées
- Le point de départ, comme vu précédemment, est le package

Manipuler le métamodèle EMF

➤ Exemple : créer et modifier des instances du modèle via le métamodèle sans génération de code

```
@Test
public void createAddressBookWithMetaModel() {
    Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
    Map<String, Object> m = reg.getExtensionToFactoryMap();
    m.put("ecore", new XMIResourceFactoryImpl());
    ResourceSet resourceSet = new ResourceSetImpl();
    URI fileURI = URI.createFileURI("model/addressbook.ecore");
    Resource resource = resourceSet.getResource(fileURI, true);

    EPackage ePackage = (EPackage) resource.getContents().get(0);

    EClass eAddressBook = (EClass) ePackage.getEClassifier("AddressBook");
    EReference eContains = (EReference) eAddressBook.getEStructuralFeature("contains");
    EAttribute eName = (EAttribute) eAddressBook.getEStructuralFeature("name");
    EObject addressBookInstance = ePackage.getEFactoryInstance().create(eAddressBook);
    addressBookInstance.eSet(eName, "Mon Carnet d'Adresses");

    EClass ePerson = (EClass) ePackage.getEClassifier("Person");
    EAttribute eFirstName = (EAttribute) ePerson.getEStructuralFeature("firstName");
    EAttribute eFamilyName = (EAttribute) ePerson.getEStructuralFeature("familyName");
    EObject personInstance = ePackage.getEFactoryInstance().create(ePerson);
    personInstance.eSet(eFirstName, "Mickael");
    personInstance.eSet(eFamilyName, "BARON");

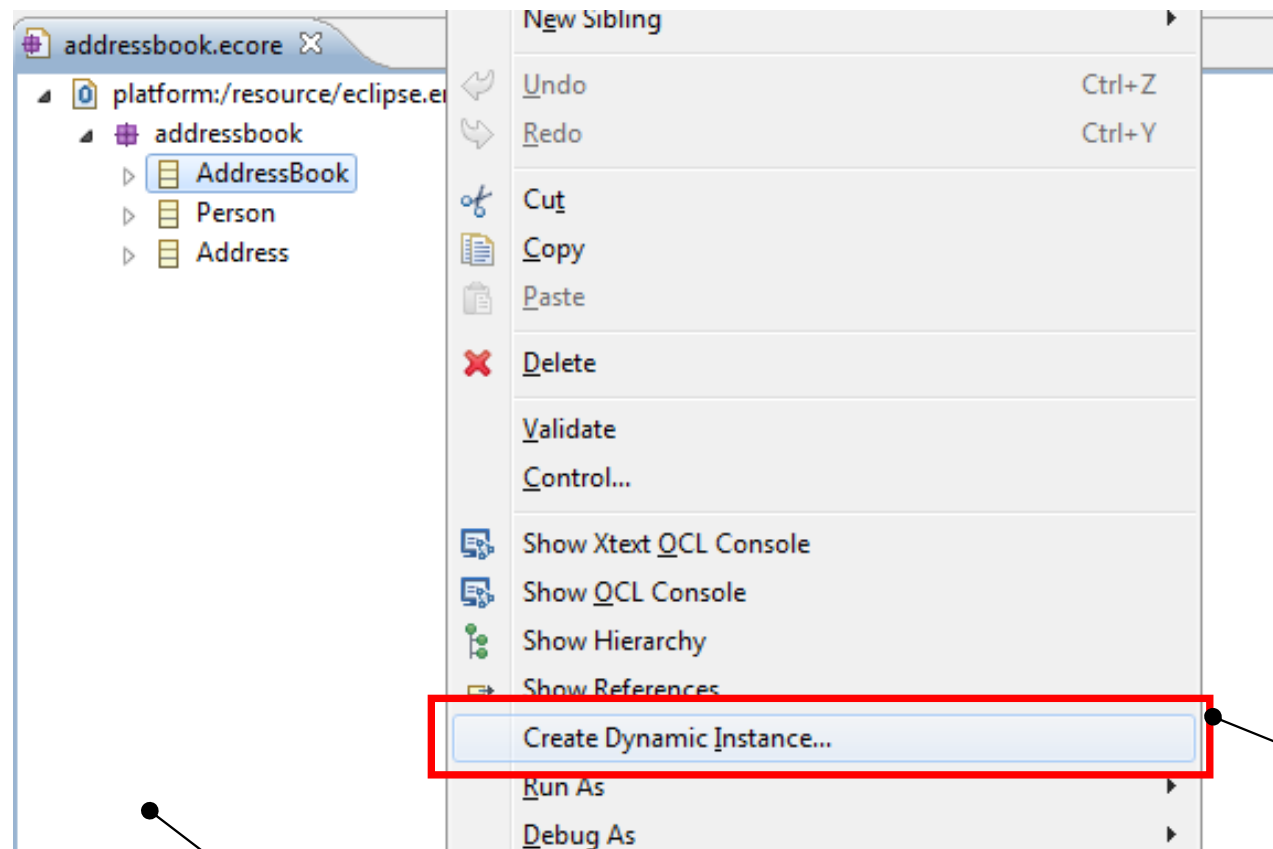
    List<EClass> containsList = new ArrayList<EClass>();
    containsList.add(ePerson);
    addressBookInstance.eSet(eContains, containsList);
}
```

Chargement du modèle *addressbook* afin de récupérer le package

Classe *AddressBookOnlyModelTest* du plugin **addressbookonlymodel**

Manipuler le métamodèle EMF

- Exemple : créer et modifier des instances du modèle via le métamodèle sans génération de code via les outils

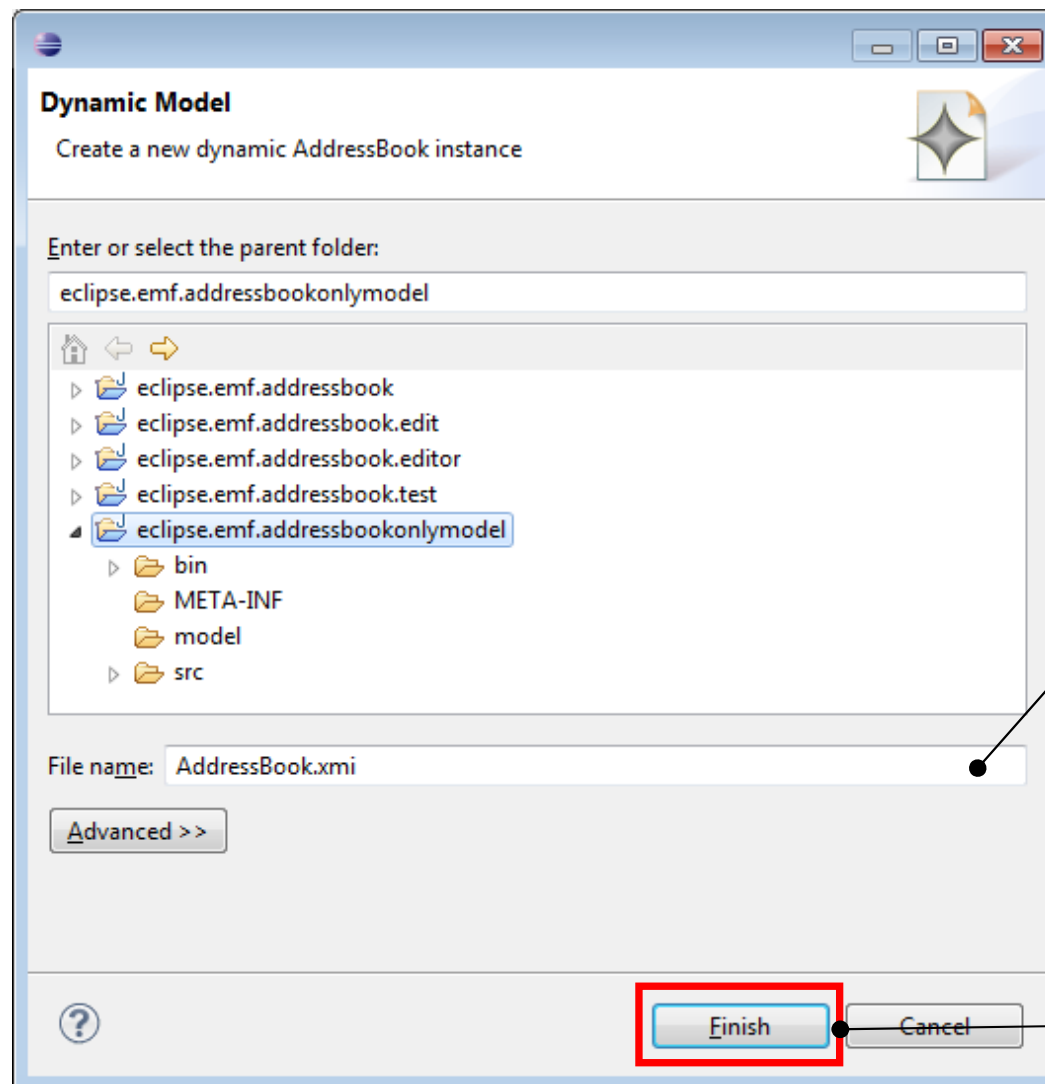


À partir de la classe
« englobante »
(*AddressBook*) création
des instances dynamiques

Modèle addressbook.ecore du plugin
addressbookonlymodel

Manipuler le métamodèle EMF

- Exemple (suite) : créer et modifier des instances du modèle via le métamodèle sans génération de code via les outils



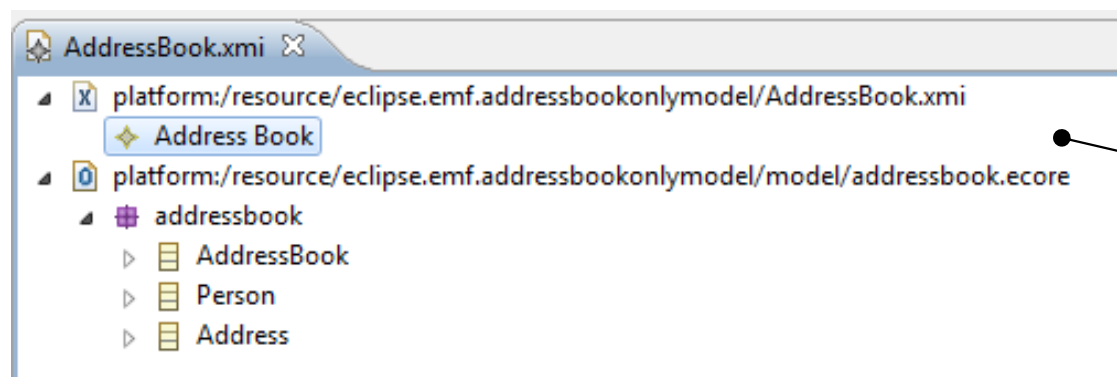
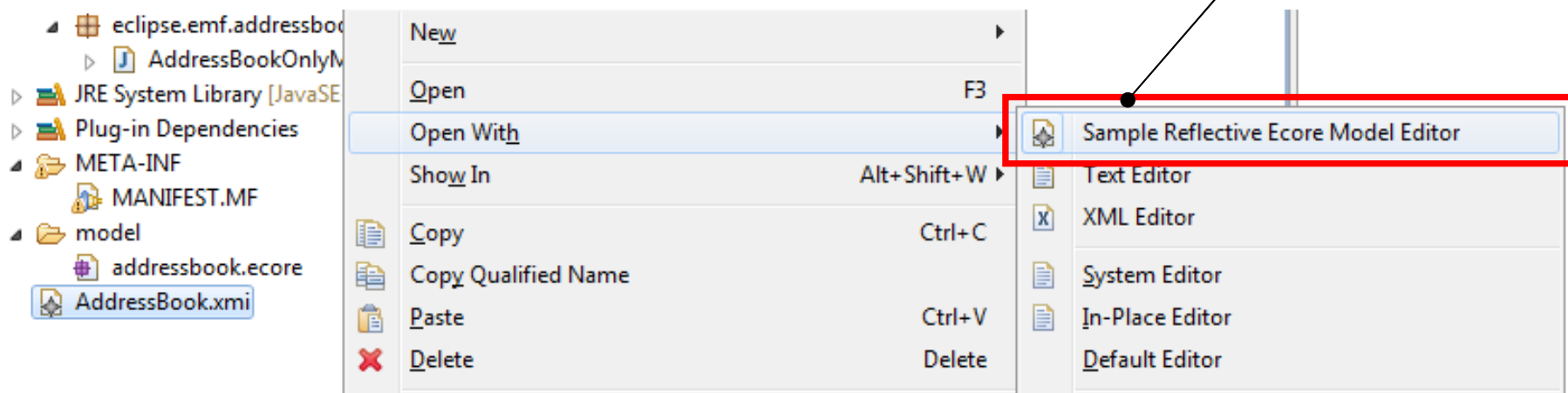
Création d'un fichier XMI contenant les futures instances

Puis faire *Finish*

Manipuler le métamodèle EMF

- Exemple (suite) : créer et modifier des instances du modèle via le métamodèle sans génération de code via les outils

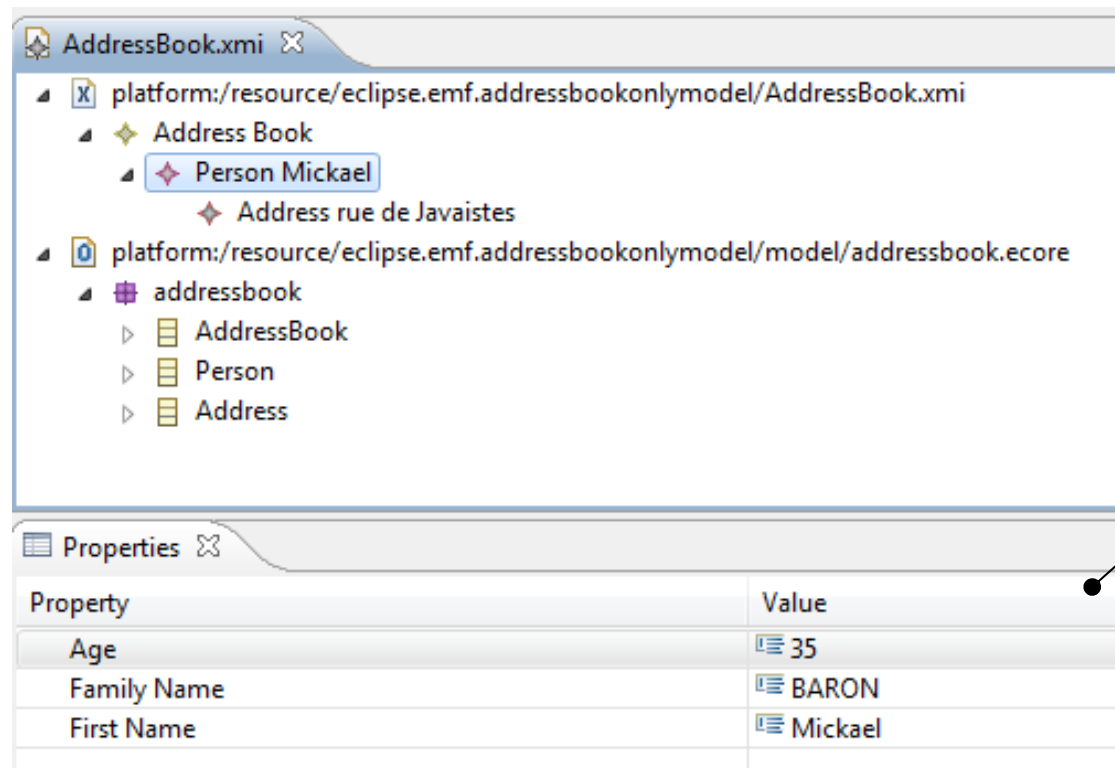
Ouvrir le fichier XMI avec l'éditeur
Sample Reflective Ecore Model Editor



Création des
instances via un
éditeur adapté

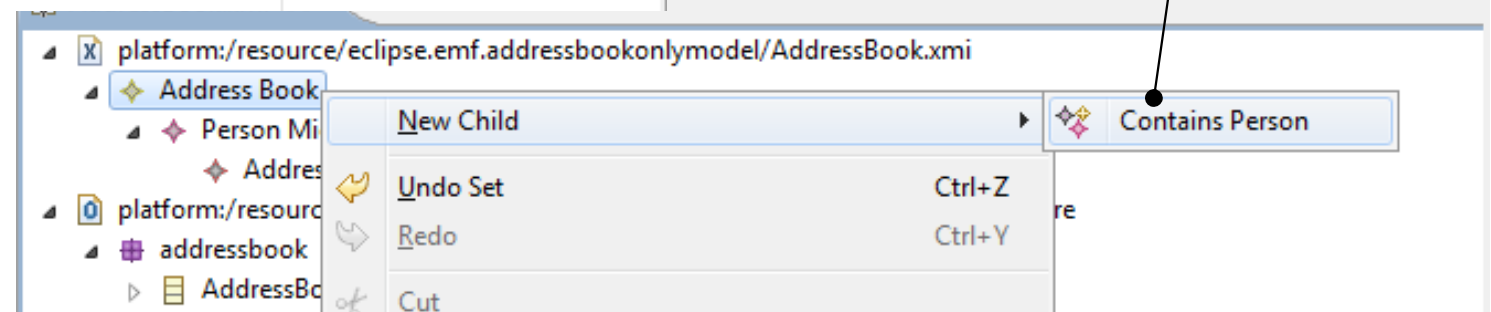
Manipuler le métamodèle EMF

- Exemple (suite) : créer et modifier des instances du modèle via le métamodèle sans génération de code via les outils



La vue *Properties* permet de modifier les valeurs des attributs

Pour chaque nœud possibilité de créer des références



Manipuler le métamodèle EMF

➤ Exemple : sauvegarder et charger des instances d'un modèle sans génération de code

```
@Test
public void saveAndLoadAddressBookWithMetaModel() {
    Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
    Map<String, Object> m = reg.getExtensionToFactoryMap();
    m.put("ecore", new XMIRResourceFactoryImpl());

    ResourceSet resourceSet = new ResourceSetImpl();
    URI fileURI = URI.createFileURI("model/addressbook.ecore");
    Resource resource = resourceSet.createResource(fileURI);
    try {
        resource.load(null);
        EPackage ePackage = (EPackage) resource.getContents().get(0);

        EClass eAddressBook = (EClass) ePackage.getEClassifier("AddressBook");
        EAttribute eName = (EAttribute) eAddressBook.getEStructuralFeature("name");
        EObject addressBookInstance = ePackage.getEFactoryInstance().create(eAddressBook);
        addressBookInstance.eSet(eName, "Mon Carnet d'Adresses");

        EClass ePerson = (EClass) ePackage.getEClassifier("Person");
        EAttribute eFirstName = (EAttribute) ePerson.getEStructuralFeature("firstName");
        EAttribute eFamilyName = (EAttribute) ePerson.getEStructuralFeature("familyName");
        EObject personInstance = ePackage.getEFactoryInstance().create(ePerson);
        personInstance.eSet(eFirstName, "Mickael");
        personInstance.eSet(eFamilyName, "BARON");

        ...
    }
}
```

Classe *AddressBookOnlyModelTest* du
plugin **addressbookonlymodel**

Manipuler le métamodèle EMF

➤ Exemple (suite) : sauvegarder et charger des instances d'un modèle sans génération de code

```
@Test
public void saveAndLoadAddressBookWithMetaModel() {
    resourceSet = new ResourceSetImpl();
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put("xmi", new XMIResourceFactoryImpl());
    URI uri = URI.createURI("file:/c:/addressbookinstancesonlymodel.xmi");
    resource = resourceSet.createResource(uri);
    resource.getContents().add(addressBookInstance);
    resource.save(null);

    resourceSet = new ResourceSetImpl();
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
        .put("xmi", new XMIResourceFactoryImpl());
    Registry packageRegistry = resourceSet.getPackageRegistry();
    packageRegistry.put("http://addressbook/1.0", ePackage);

    uri = URI.createURI("file:/c:/addressbookinstancesonlymodel.xmi");
    resource = resourceSet.getResource(uri, true);
    resource.load(null);
    EObject test = resource.getContents().get(0);
    System.out.println(test);
} catch (IOException e) {
    e.printStackTrace();
    Assert.fail();
}
```

Permet d'associer le package chargé précédemment

addressbookinstancesonlymodel.xmi

```
<?xml version="1.0" encoding="ASCII"?>
<addressbook:AddressBook xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:addressbook="http://addressbook/1.0" name="Mon Carnet d'Adresses">
  <contains firstName="Mickael" familyName="BARON"/>
</addressbook:AddressBook>
```

Classe *AddressBookOnlyModelTest*
du plugin **addressbookonlymodel**

EMF sans conteneur OSGi

- Le framework EMF ne se limite pas seulement aux applications exécutées dans un conteneur OSGi
- Toute application Java peut embarquer des modèles générés avec EMF (GWT, Swing, Java FX, Servlet...)
- Les bibliothèques minimales du framework EMF (disponibles dans le répertoire plugins d'Eclipse) à ajouter dans le *classpath* sont :
 - *org.eclipse.emf.ecore*
 - *org.eclipse.emf.common*

EMF sans conteneur OSGi

► Exemple : projet Java « pur » intégrant les Jars EMF

```
public class AddressBookMainClass {
    public static void main(String[] args) {
        AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
        createAddressBook.setName("Mon Carnet d'Adresses");

        Address createMBAddress = AddressbookFactory.eINSTANCE.createAddress();
        createMBAddress.setNumber(5);
        createMBAddress.setStreet("Rue des Javaistes");
        Person mickaelBaron = AddressbookFactory.eINSTANCE.createPerson();
        mickaelBaron.setAge(35);
        mickaelBaron.setFamilyName("BARON");
        mickaelBaron.setFirstName("Mickael");
        mickaelBaron.setLocation(createMBAddress);

        Address createAddress = AddressbookFactory.eINSTANCE.createAddress();
        createAddress.setNumber(6);
        createAddress.setStreet("Rue des Espions");
        Person johnAaron = AddressbookFactory.eINSTANCE.createPerson();
        johnAaron.setAge(14);
        johnAaron.setFamilyName("AARON");
        johnAaron.setFirstName("John");
        johnAaron.setLocation(createAddress);

        createAddressBook.getContains().add(mickaelBaron);
        createAddressBook.getContains().add(johnAaron);

        System.out.println(createAddressBook.toString());
    }
}
```

Classe *AddressBookMainClass*
du plugin **addressbookonlymodel**

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
    <classpathentry kind="src" path="src"/>
    <classpathentry kind="con" path="..../JavacSE-1.6"/>
    <classpathentry kind="lib" path="C:/eclipseModelling/plugins/org.eclipse.emf.ecore_2.7.0.v20110912-0920.jar"/>
    <classpathentry kind="lib" path="C:/eclipseModelling/plugins/org.eclipse.emf.common_2.7.0.v20110912-0920.jar"/>
    <classpathentry kind="output" path="bin"/>
</classpath>
```

Notifier les changements des instances

- Toute classe EMF hérite de *EObject* qui est un *notifier*
- Il est donc possible d'écouter les changements réalisés sur une instance

```
eObject.eAdapters().add(new AdapterImpl() {  
    public void notifyChanged(Notification notification) {  
        // Ecoute les changements  
    }  
});
```

- Pour réaliser une écoute en profondeur (toutes les instances contenues) la classe *EContentAdapter* peut être utilisée
- Nous verrons dans la suite que les transactions offrent également un mécanisme de notification

Notifier les changements des instances

➤ Exemple : écouter les changements des instances

```
@Test
public void createAddressBookNotifier() {
    AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
    createAddressBook.eAdapters().add(new EContentAdapter() {
        public void notifyChanged(Notification notification) {
            super.notifyChanged(notification);
            System.out.println("(Global) Notification received from the data model : " +
                notification.getNewValue());
        }
    });
    createAddressBook.eAdapters().add(new AdapterImpl() {
        public void notifyChanged(Notification notification) {
            System.out.println("(Local) Notification received from the data model : " +
                notification.getNewValue());
        }
    });
    createAddressBook.setName("Mon Carnet d'Adresses");

    Person mickaelBaron = AddressbookFactory.eINSTANCE.createPerson();
    createAddressBook.getContains().add(mickaelBaron);
    mickaelBaron.setAge(35);
}
```

Classe *AddressBookTest*
du fragment **addressbook.test**

```
Console
<terminated> AddressBookTest.createAddressBookNotifier [JUnit] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (11 janv. 2012 18:52:04)
(Global) Notification received from the data model : Mon Carnet d'Adresses
(Local) Notification received from the data model : Mon Carnet d'Adresses
(Global) Notification received from the data model : eclipse.emf.addressbook.model.addressbook.impl.PersonImpl@aefdf (firstName: )
(Local) Notification received from the data model : eclipse.emf.addressbook.model.addressbook.impl.PersonImpl@aefdf (firstName: )
(Global) Notification received from the data model : 35
(Global) Notification received from the data model : BARON
(Global) Notification received from the data model : Mickael
(Global) Notification received from the data model : eclipse.emf.addressbook.model.addressbook.impl.AddressImpl@f73c1 (number: )
(Global) Notification received from the data model : 5
(Global) Notification received from the data model : Rue des Javaistes
(Global) Notification received from the data model : eclipse.emf.addressbook.model.addressbook.impl.PersonImpl@789144 (firstName: )
(Local) Notification received from the data model : eclipse.emf.addressbook.model.addressbook.impl.PersonImpl@789144 (firstName: )
```

Transactions EMF

- Le framework EMF fournit une API pour la gestion des transactions
- Pour utiliser les transactions EMF, il est nécessaire d'ajouter une dépendance sur le plugin *org.eclipse.emf.transaction*
- Les transactions permettent d'assurer :
 - Lecture et écriture des instances à partir de plusieurs Thread
 - Intégrité du modèle (*Rollback* automatique si non conforme)
 - Gestion automatique du mécanisme de *Undo / Redo*
 - Notification des changements (*pre-commit* et *post-commit*)

Transactions EMF : TransactionalEditingDomain

- La gestion des transactions est obtenue par un domaine d'édition transactionnel
- Un domaine d'édition transactionnel peut verrouiller une ressource de type *ResourceSet* (précédemment étudiée)
- La classe *TransactionalEditingDomain* définit le domaine d'édition transactionnel
- Deux solutions pour créer un *TransactionalEditingDomain*
 - Par une fabrique : à utiliser pour les applications non OSGi/Eclipse et/ou quand un seul client doit manipuler le domaine d'édition
 - Par le registre : où le domaine d'édition est défini via une extension Eclipse

Transactions EMF : TransactionalEditingDomain

► Exemple : créer *TransactionalEditingDomain* via la fabrique

```
@Test
public void createTransactionalEditingDomainByFactory() {
    // Définition d'une Resource.
    ResourceSet resourceSet = new ResourceSetImpl();
    Resource resource = resourceSet.createResource(URI.createURI("addressbookinstances"));
    // Création d'une instance sans transaction.
    AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
    resource.getContents().add(createAddressBook);

    TransactionalEditingDomain domain =
        TransactionalEditingDomain.Factory.INSTANCE.createEditingDomain(resourceSet);

    Command createCommand = domain.createCommand(
        SetCommand.class,
        new CommandParameter(
            createAddressBook,
            AddressbookPackage.Literals.ADDRESS_BOOK__NAME, "Mon Carnet d'Adresses"));
    CommandStack commandStack = domain.getCommandStack();
    commandStack.execute(createCommand);

    try {
        createAddressBook.setName("Mon Nouveau Carnet d'Adressess");
        Assert.fail();
    } catch (IllegalStateException p) {
    }
}
```

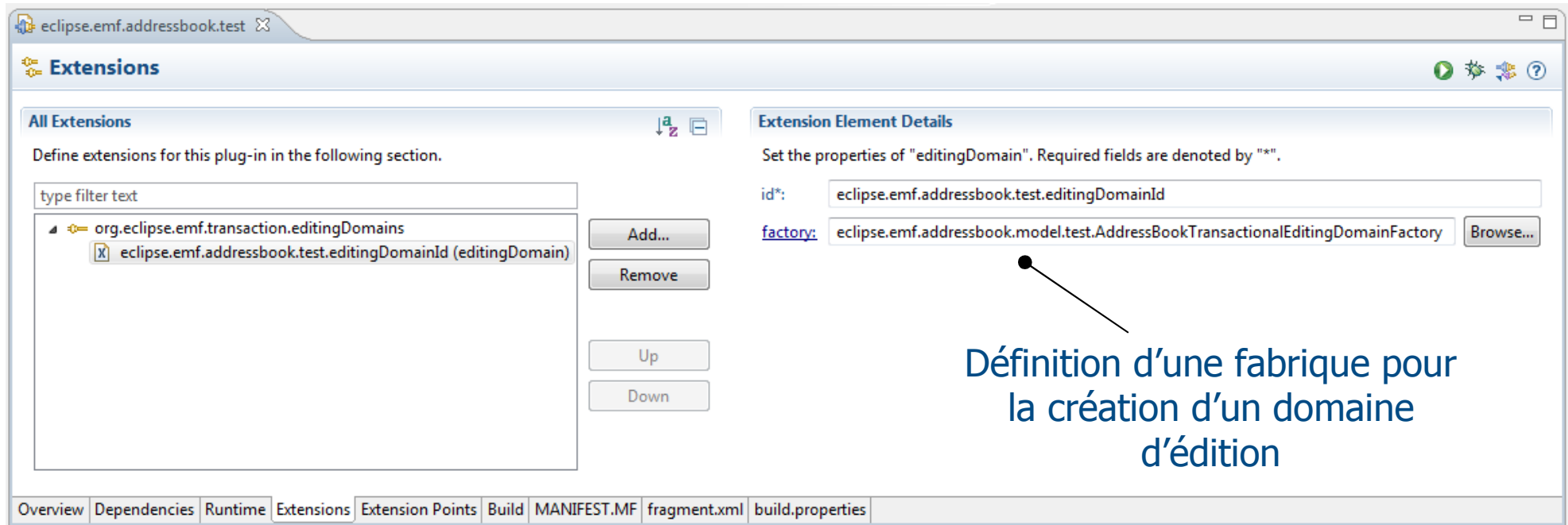
Création d'une transaction
(voir plus tard)

Ne peut modifier une ressource
sans passage par une transaction

Classe *AddressBookTest* du fragment
addressbook.test

Transactions EMF : TransactionalEditingDomain

➤ Exemple : créer *TransactionalEditingDomain* via le registre



Définition d'une fabrique pour la création d'un domaine d'édition

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<fragment>
  <extension
    point="org.eclipse.emf.transaction.editingDomains">
    <editingDomain
      factory="eclipse.emf.addressbook.model.test.AddressBookTransactionalEditingDomainFactory"
      id="eclipse.emf.addressbook.test.editingDomainId">
    </editingDomain>
  </extension>
</fragment>
```

Fichier *fragment.xml* du fragment **addressbook.test**

Transactions EMF : TransactionalEditingDomain

- Exemple (suite) : créer *TransactionalEditingDomain* via le registre

Récupération de
l'implémentation par défaut

```
public class AddressBookTransactionalEditingDomainFactory extends TransactionalEditingDomainImpl.FactoryImpl {  
    public TransactionalEditingDomain createEditingDomain() {  
        TransactionalEditingDomain transactionalEditingDomain = super.createEditingDomain();  
  
        // TODO specific configurations for this EditingDomain.  
  
        return transactionalEditingDomain;  
    }  
}
```

Classe *AddressBookTransactionEditingDomainFactory*
du fragment **addressbook.test**

Transactions EMF : TransactionalEditingDomain

- Exemple (suite) : créer *TransactionalEditingDomain* via le registre

```
@Test
public void createTransactionalEditingDomainByRegistry() {
    TransactionalEditingDomain domain = TransactionalEditingDomain.Registry.INSTANCE.getEditingDomain(
        "eclipse.emf.addressbook.test.editingDomainId");
    final Resource resource = domain.getResourceSet().createResource(URI.createURI("addressbookinstances"));

    RecordingCommand recordingCommand = new RecordingCommand(domain) {
        protected void doExecute() {
            AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
            createAddressBook.setName("Mon Carnet d'Adresses");
            resource.getContents().add(createAddressBook);
        }
    };
    CommandStack commandStack = domain.getCommandStack();
    commandStack.execute(recordingCommand);
}
```

Classe *AddressBookTest* du fragment
addressbook.test

Création d'une transaction
globale (voir plus tard)

Transactions EMF : TransactionalEditingDomain

► Exemple : désactivation d'un domaine d'édition

```
@Test
public void disableTransactionalEditingDomainByFactory() {
    ResourceSet resourceSet = new ResourceSetImpl();
    Resource resource = resourceSet.createResource(URI.createURI("addressbookinstances"));
    AddressBook createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
    resource.getContents().add(createAddressBook);

    TransactionalEditingDomain domain = TransactionalEditingDomain.Factory.INSTANCE.createEditingDomain(resourceSet);

    Command createCommand = domain.createCommand(
        SetCommand.class,
        new CommandParameter(
            createAddressBook,
            AddressbookPackage.Literals.ADDRESS_BOOK__NAME, "Mon Carnet d'Adresses"));
    TransactionalCommandStack commandStack = (TransactionalCommandStack)domain.getCommandStack();
    commandStack.execute(createCommand);

    domain.dispose();

    try {
        createAddressBook.setName("Mon Nouveau Carnet d'Adressess");
    } catch (IllegalStateException p) {
        Assert.fail();
    }
}
```

Désactivation du domaine
d'édition et faisant appel à
dispose()

Classe *AddressBookTest* du fragment
addressbook.test

Transactions EMF : Écriture

- Les modifications effectuées dans le domaine d'édition sont obtenues par des commandes (*Commands*)
- L'exécution d'une commande est réalisée dans une pile de commandes (*CommandStack*)
- Lors de l'exécution de la commande, une transaction peut aboutir à une exception (*Rollback*) si l'intégrité du modèle n'est pas respectée
- La *CommandStack* sera utilisée par la suite pour gérer automatiquement le mécanisme d'Undo / Redo (rappel des précédentes commandes)

Transactions EMF : Écriture

➤ Exemple : modifier la valeur d'un attribut d'une classe

```
@Test
public void createWriteTransactionWithCreateCommand() {
    ...

    TransactionalEditingDomain domain = TransactionalEditingDomain.Factory.INSTANCE
        .createEditingDomain(resourceSet);

    Command createCommand = domain.createCommand(
        SetCommand.class,
        new CommandParameter(
            createAddressBook,
            AddressbookPackage.Literals.ADDRESS_BOOK__NAME, "Mon Carnet d'Adresses"));

    domain.getCommandStack().execute(createCommand);

    Assert.assertEquals("Mon Carnet d'Adresses", createAddressBook.getName());
}
```

Création d'une commande

De type modification

Instance de la classe à modifier

L'attribut qui sera modifié

La nouvelle valeur à appliquer

Classe *AddressBookTest* du fragment **addressbook.test**

Transactions EMF : Lecture

► Exemple : modifier un ensemble de valeurs

```
@Test
public void createWriteTransactionWithRecordingCommand() {
    TransactionalEditingDomain domain = TransactionalEditingDomain.Registry.INSTANCE
        .getEditingDomain("eclipse.emf.addressbook.test.editingDomainId");
    final Resource resource = domain.getResourceSet().createResource(
        URI.createURI("addressbookinstances"));

    RecordingCommand recordingCommand = new RecordingCommand(domain) {
        protected void doExecute() {
            createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
            createAddressBook.setName("Mon Carnet d'Adresses");
            resource.getContents().add(createAddressBook);

            Person firstPerson = AddressbookFactory.eINSTANCE.createPerson();
            Person secondPerson = AddressbookFactory.eINSTANCE.createPerson();

            createAddressBook.getContains().add(firstPerson);
            createAddressBook.getContains().add(secondPerson);
        }
    };
    CommandStack commandStack = domain.getCommandStack();
    commandStack.execute(recordingCommand);

    Assert.assertNotNull(createAddressBook);
    Assert.assertEquals("Mon Carnet d'Adresses", createAddressBook.getName());
    Assert.assertEquals(2, createAddressBook.getContains().size());
}
```

Création d'une commande
qui englobe plusieurs
modifications

Classe *AddressBookTest* du fragment
addressbook.test

Transactions EMF : Écouteurs

- Précédemment nous avons montré qu'EMF fournissait un mécanisme d'écouteurs (*Adapter*)
- Les transactions EMF fournissent également leur propre mécanisme d'écouteurs
 - **post-commit** : déclenché après la fin de la transaction
 - **pre-commit** (trigger) : déclenché avant la fin de la transaction
- Si une transaction aboutit à un échec (rollback) aucun événement n'est retourné puisqu'il n'y a pas de modification

Transactions EMF : Écouteurs

- L'abonnement aux écouteurs se fait par l'intermédiaire du domaine d'édition transactionnel (*addResourceSetListener*) via un *ResourceSetListener*
- L'événement retourné est de type *ResourceSetChangeEvent* et permet d'accéder aux informations suivantes :
 - *editingDomain* : domaine d'édition dans lequel la ressource a été modifiée
 - *notifications* : la liste des notifications (type, ancienne valeur, nouvelle valeur, attribut (*feature*) modifié...)
 - *transaction* : permet d'obtenir des informations sur la transaction

Transactions EMF : Écouteurs

► Exemple : mise en place d'un post-commit

```
@Test
public void createPostCommitListeners() {
    ...
    domain.addResourceSetListener(new ResourceSetListenerImpl() {
        public void resourceSetChanged(ResourceSetChangeEvent event) {
            System.out.println("Domain " + event.getEditingDomain().getID()
                + " changed " + event.getNotifications().size() + " times");

            List<Notification> notifications = event.getNotifications();
            for (Notification notification : notifications) {
                System.out.print("Type: " + notification.getEventType() +
                    " Old Value=" + notification.getOldValue() +
                    " New Value=" + notification.getNewValue());
            }
        }
    });
    recordingCommand = new RecordingCommand(domain) {
        protected void doExecute() {
            createAddressBook.setName("Mon Nouveau Carnet d'Adresses");
            Person firstPerson = AddressbookFactory.eINSTANCE.createPerson();

            createAddressBook.getContains().add(firstPerson);
        }
    };
    commandStack = domain.getCommandStack();
    commandStack.execute(recordingCommand);
}
```

Abonnement d'un écouteur

Type de la commande

Affichage de l'ancienne valeur et de la nouvelle valeur

```
Domain eclipse.emf.addressbook.test.editingDomainId changed 2 times
Type: 1 Old Value=Mon Carnet d'Adresses New Value=Mon Nouveau Carnet d'Adresses
Type: 3 Old Value=null New Value=eclipse.emf.addressbook.model.addressbook.impl.PersonImpl@168442e
```

Classe *AddressBookTest* du fragment
addressbook.test

Transactions EMF : Écouteurs

➤ Exemple : post-commit via *DemultiplexingListener*

```
@Test
public void createPostCommitListenersWithDemultiplexingListener () {
    ...

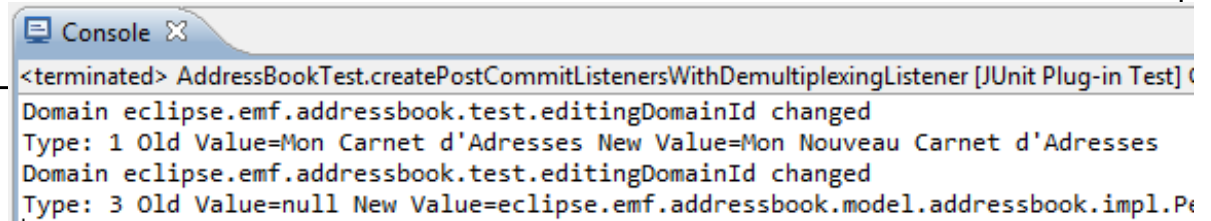
    domain.addResourceSetListener(new DemultiplexingListener() {
        protected void handleNotification(TransactionalEditingDomain domain, Notification notification) {
            System.out.println("Domain " + domain.getID() + " changed ");

            System.out.print("Type: " + notification.getEventType() +
                " Old Value=" + notification.getOldValue() +
                " New Value=" + notification.getNewValue());
        }
    });

    recordingCommand = new RecordingCommand(domain) {
        protected void doExecute() {
            createAddressBook.setName("Mon Nouveau Carnet d'Adresses");
            Person firstPerson = AddressbookFactory.eINSTANCE.createPerson();

            createAddressBook.getContains().add(firstPerson);
        }
    };
    commandStack = domain.getCommandStack();
    commandStack.execute(recordingCommand);
}
```

Cet écouteur permet de manipuler les notifications une par une



```
<terminated> AddressBookTest.createPostCommitListenersWithDemultiplexingListener [JUnit Plug-in Test]
Domain eclipse.emf.addressbook.test.editingDomainId changed
Type: 1 Old Value=Mon Carnet d'Adresses New Value=Mon Nouveau Carnet d'Adresses
Domain eclipse.emf.addressbook.test.editingDomainId changed
Type: 2 Old Value=null New Value=eclipse.emf.addressbook.model.addressbook.impl.Person
Domain eclipse.emf.addressbook.test.editingDomainId changed
Type: 3 Old Value=null New Value=eclipse.emf.addressbook.model.addressbook.impl.AddressBook
```

Classe *AddressBookTest* du fragment **addressbook.test**

Transactions EMF : Écouteurs

➤ Exemple : mise en place d'un trigger

```

@Test
public void createTriggers() {
    ...

    domain.addResourceSetListener(new ResourceSetListenerImpl() {
        public Command transactionAboutToCommit(ResourceSetChangeEvent event) throws RollbackException {
            List<Command> commands = new ArrayList<Command>();
            for (Notification notification : event.getNotifications()) {
                if (notification.getNotifier() instanceof AddressBook) {
                    AddressBook currentAB = (AddressBook)notification.getNotifier();
                    for(final Person currentPerson : currentAB.getContains()) {
                        commands.add(new RecordingCommand(event.getEditingDomain()) {
                            protected void doExecute() {
                                currentPerson.setFirstName("A FirstName");
                                currentPerson.setFamilyName("A FamilyName");
                                currentPerson.setAge(18);
                            }
                        });
                    }
                }
            }
            return commands.isEmpty()? null : new CompoundCommand(commands);
        }
    });
    ...
}

```

Implémenter la méthode transactionAboutToCommit

Possibilité de créer de nouvelles commandes avant la fin de la transaction

Console

```

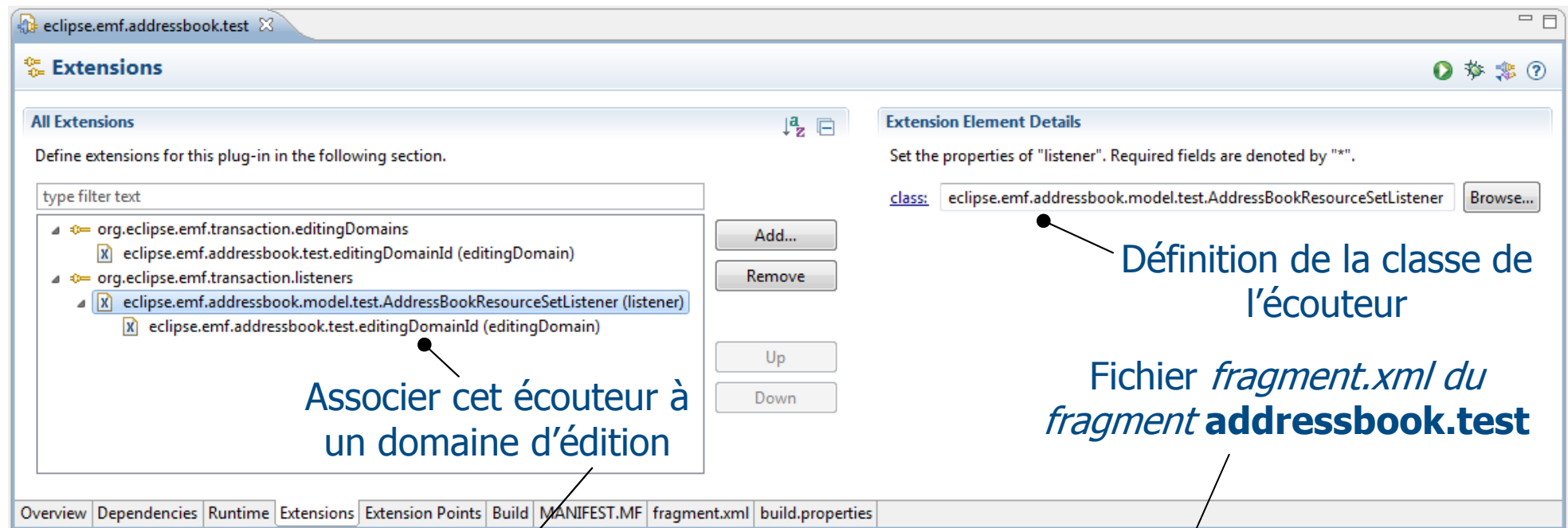
<terminated> AddressBookTest.createTriggers [JUnit Plug-in Test] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (5 ja
Domain eclipse.emf.addressbook.test.editingDomainId changed 5 times
Type: 1 Old Value=Mon Carnet d'Adresses New Value=Mon Nouveau Carnet d'Adresses
Type: 3 Old Value=null New Value=eclipse.emf.addressbook.model.addressbook.impl.PersonImp
Type: 1 Old Value=null New Value=A FirstName
Type: 1 Old Value=null New Value=A FamilyName
Type: 1 Old Value=0 New Value=18

```

Classe *AddressBookTest* du fragment
addressbook.test

Transactions EMF : Écouteurs

- Possibilité de définir un écouteur **statique** en utilisant l'extension (*org.eclipse.emf.transaction.listeners*)
- Exemple : définir un écouteur par extension



```
<fragment>
  ...
  <extension point="org.eclipse.emf.transaction.listeners">
    <listener class="eclipse.emf.addressbook.model.test.AddressBookResourceSetListener">
      <editingDomain id="eclipse.emf.addressbook.test.editingDomainId" />
    </listener>
  </extension>
</fragment>
```

Transactions EMF : Écouteurs

➤ Exemple (suite) : définir un écouteur par extension

```
public class AddressBookResourceSetListener extends ResourceSetListenerImpl {  
  
    public void resourceSetChanged(ResourceSetChangeEvent event) {  
        System.out.println("Domain " + event.getEditingDomain().getID() +  
            " changed " + event.getNotifications().size() + " times");  
  
        List<Notification> notifications = event.getNotifications();  
        for (Notification notification : notifications) {  
            System.out.print("Type: " + notification.getEventType() +  
                " Old Value=" + notification.getOldValue() +  
                " New Value=" + notification.getNewValue());  
  
            System.out.println();  
        }  
    }  
}
```

Classe *AddressBookResourceSetListener* du
fragment **addressbook.test**

Transactions EMF : Undo/Redo

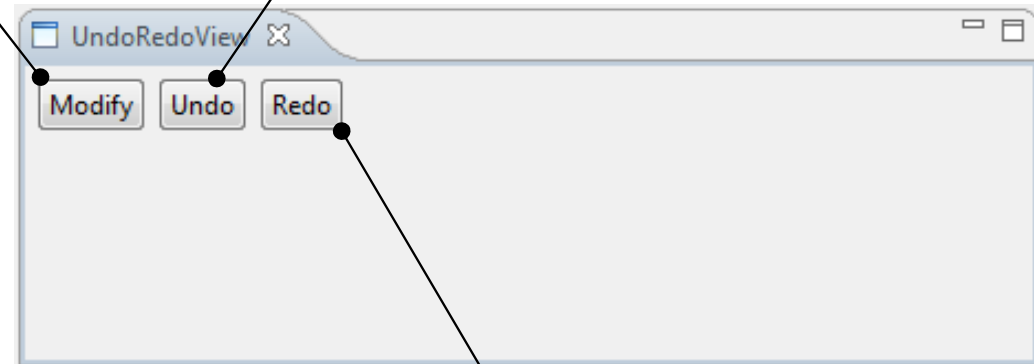
- Possibilité d'utiliser les transactions pour gérer automatiquement le mécanisme undo/redo
- Chaque commande invoquée est stockée dans une pile (*CommandStack*) récupérable via le domaine d'édition
- À partir d'un *CommandStack* possibilité d'invoquer une commande pour réaliser
 - *undo()* : procéder à un retour en arrière de l'état du modèle
 - *redo()* : procéder à un retour en avant de l'état du modèle
- À partir du *CommandStack* possibilité de vérifier s'il est possible de procéder à un undo ou un redo
 - *boolean canRedo()* : peut faire un redo
 - *boolean canUndo()* : peut faire un undo

Transactions EMF : Undo/Redo

➤ Exemple : utiliser le mécanisme undo/redo

Modification de la valeur de l'attribut *name*

Retour arrière sur la valeur de l'attribut *name*



Pour chaque modification de l'attribut *name* l'instance *AddressBook* est affichée

Retour avant sur la valeur de l'attribut *name*

```

Console X
AddressBookConfigurationUI [Eclipse Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (11 janv. 2012 17:18:13)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: kooBsserdda elpmas a si sihT)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: This is a sample AddressBook)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: kooBsserdda elpmas a si sihT)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: This is a sample AddressBook)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: kooBsserdda elpmas a si sihT)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: This is a sample AddressBook)
eclipse.emf.addressbook.model.addressbook.impl.AddressBookImpl@169bc15 (name: kooBsserdda elpmas a si sihT)

```

Transactions EMF : Undo/Redo

➤ Exemple (suite) : utiliser le mécanisme undo/redo

```
public class UndoRedoViewPart extends ViewPart {
    ...

    public void createPartControl(Composite parent) {
        ...
        ResourceSet resourceSet = new ResourceSetImpl();
        Resource resource = resourceSet.createResource(URI.createURI("addressbookinstances"));
        createAddressBook = AddressbookFactory.eINSTANCE.createAddressBook();
        createAddressBook.setName("This is a sample AddressBook");
        resource.getContents().add(createAddressBook);
        domain = TransactionalEditingDomain.Factory.INSTANCE.createEditingDomain(resourceSet);

        Button modify = new Button(parent, SWT.FLAT);
        modify.setText("Modify");
        modify.addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {
                RecordingCommand recordingCommand = new RecordingCommand(domain) {
                    protected void doExecute() {
                        StringBuffer sb = new StringBuffer();
                        sb.append(createAddressBook.getName());
                        createAddressBook.setName(sb.reverse().toString());
                    }
                };
                domain.getCommandStack().execute(recordingCommand);

                redo.setEnabled(domain.getCommandStack().canRedo());
                undo.setEnabled(domain.getCommandStack().canUndo());

                System.out.println(createAddressBook.toString());
            }
        });
    }
}
```

Modification de la valeur
de l'attribut *name*

Classe *UndoRedoViewPart* du plugin
addressbook.ui

Transactions EMF : Undo/Redo

➤ Exemple (suite) : utiliser le mécanisme undo/redo

```
public class UndoRedoViewPart extends ViewPart {
    ...
    public void createPartControl(Composite parent) {
        ...
        undo = new Button(parent, SWT.FLAT);
        undo.setText("Undo");
        undo.addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {
                domain.getCommandStack().undo();

                redo.setEnabled(domain.getCommandStack().canRedo());
                undo.setEnabled(domain.getCommandStack().canUndo());

                System.out.println(createAddressBook.toString());
            }
        });

        redo = new Button(parent, SWT.FLAT);
        redo.setText("Redo");
        redo.addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {
                domain.getCommandStack().redo();

                redo.setEnabled(domain.getCommandStack().canRedo());
                undo.setEnabled(domain.getCommandStack().canUndo());

                System.out.println(createAddressBook.toString());
            }
        });
    }
}
```

Appelle la commande *undo*

Appelle la commande *redo*

Classe *UndoRedoViewPart* du plugin
addressbook.ui

Conclusion

- Nous avons étudié dans ce cours
 - Construction d'un modèle EMF
 - Manipulation du métamodèle Ecore
 - Transactions
- Toutes les bases EMF sont acquises et nous étudierons dans les prochains cours
 - L'intégration d'EMF dans les IHM via l'utilisation de binding
 - La génération de formulaires automatiques avec Eclipse EEF
 - La persistance des modèles dans une base de données (CDO, Teneo)
 - La validation des modèles via le framework Validation et OCL