

TP 2: Transformation

Cette manipulation consiste en la création de deux méta modèles définir les règles de transformation nécessaires pour le passage entre les deux utilisant l'approche par programmation.

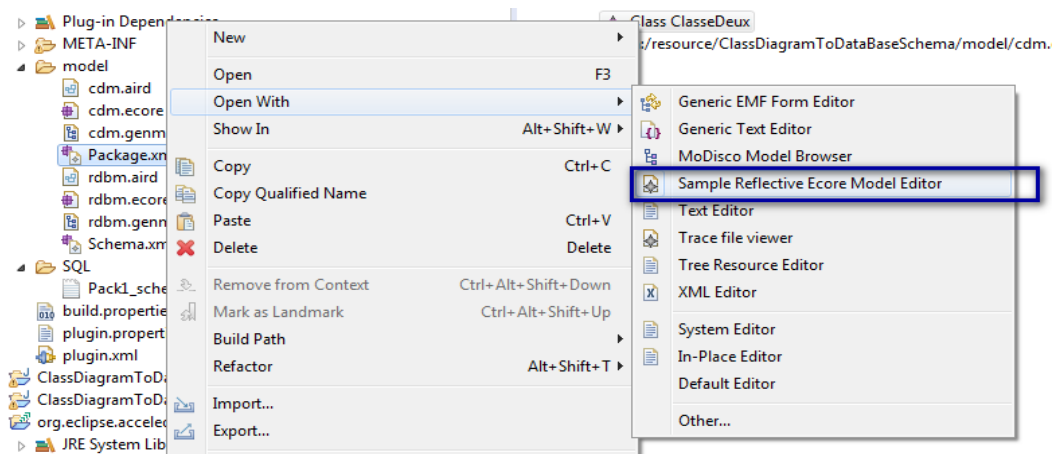
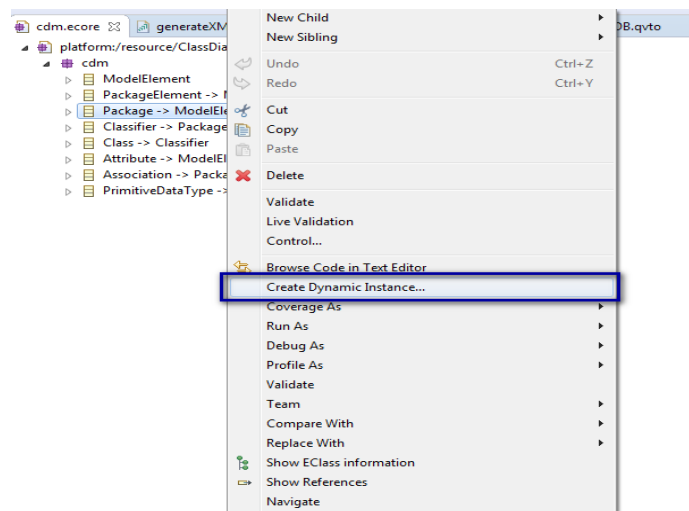
1. Créer un projet EMF vide ClassDiagramToDataBaseSchema
2. Importer les deux méta modèles fournis cdm.ecore et rdbm.ecore
3. Créer le générateur de modèle EMF

A partir des méta modèles définis, générer le générateur de modèle en chargeant les deux méta modèles, de ce fait, on aura deux .genmodel.

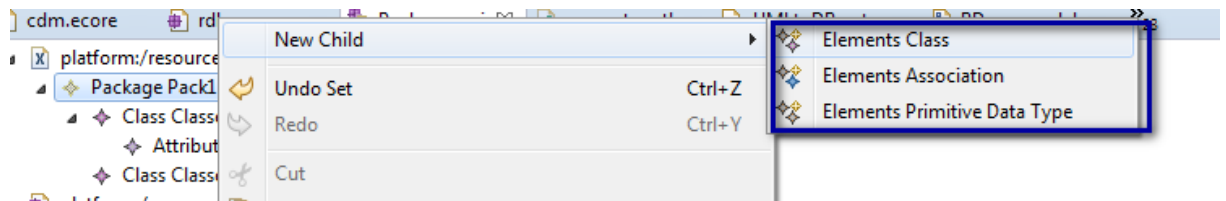
4. Instanciation des méta modèles

Clic droit sur l'élément
package de votre .ecore puis
sélectionner « create dynamic
Instance »

Une fois le fichier .xmi est créé
clic droit sur le fichier puis
choisir l'éditeur ecore

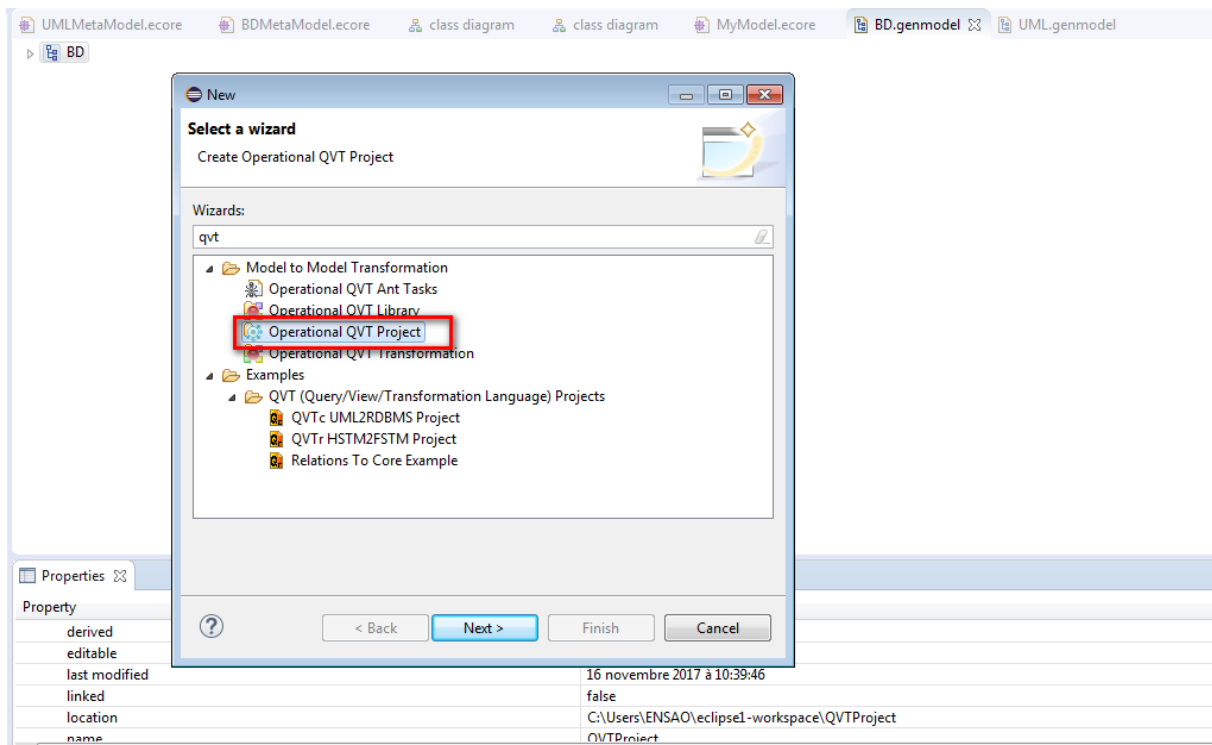


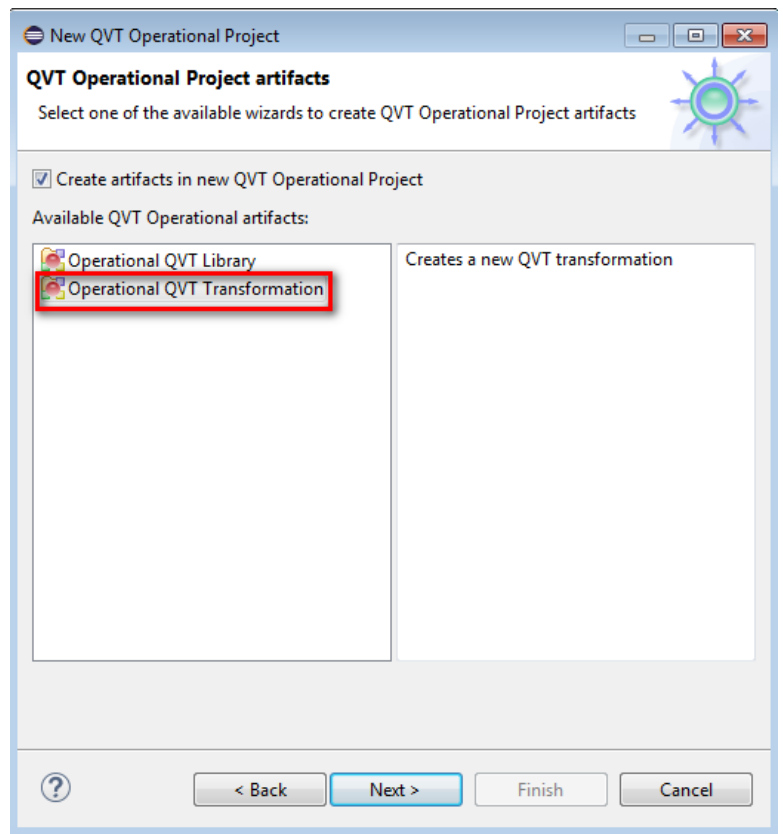
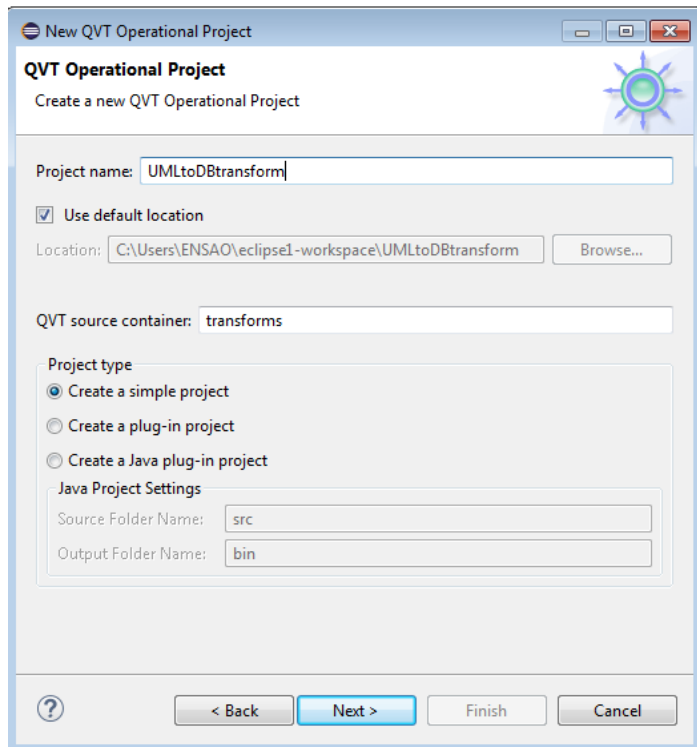
Créer une instanciation de méta modèle



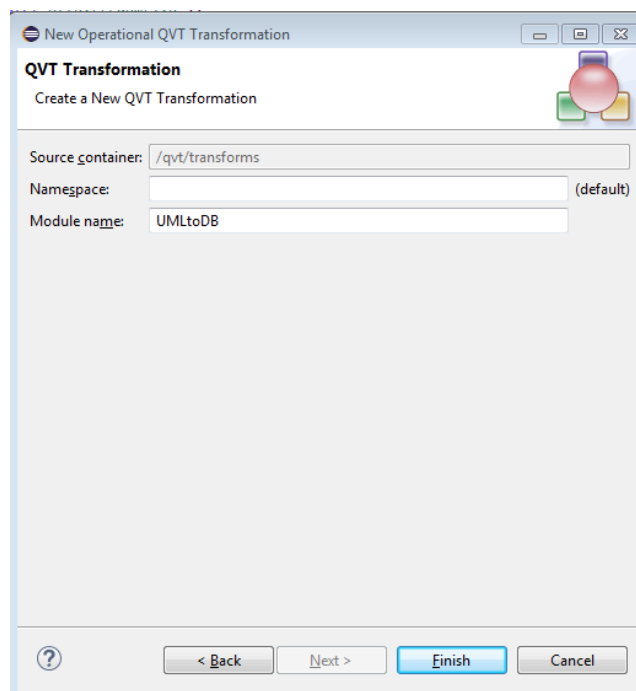
5. Créer un projet de transformation QVT

Une fois le générateur est créé, il est maintenant temps de créer un projet de transformation
(new → other → Operationnal QVT Project)



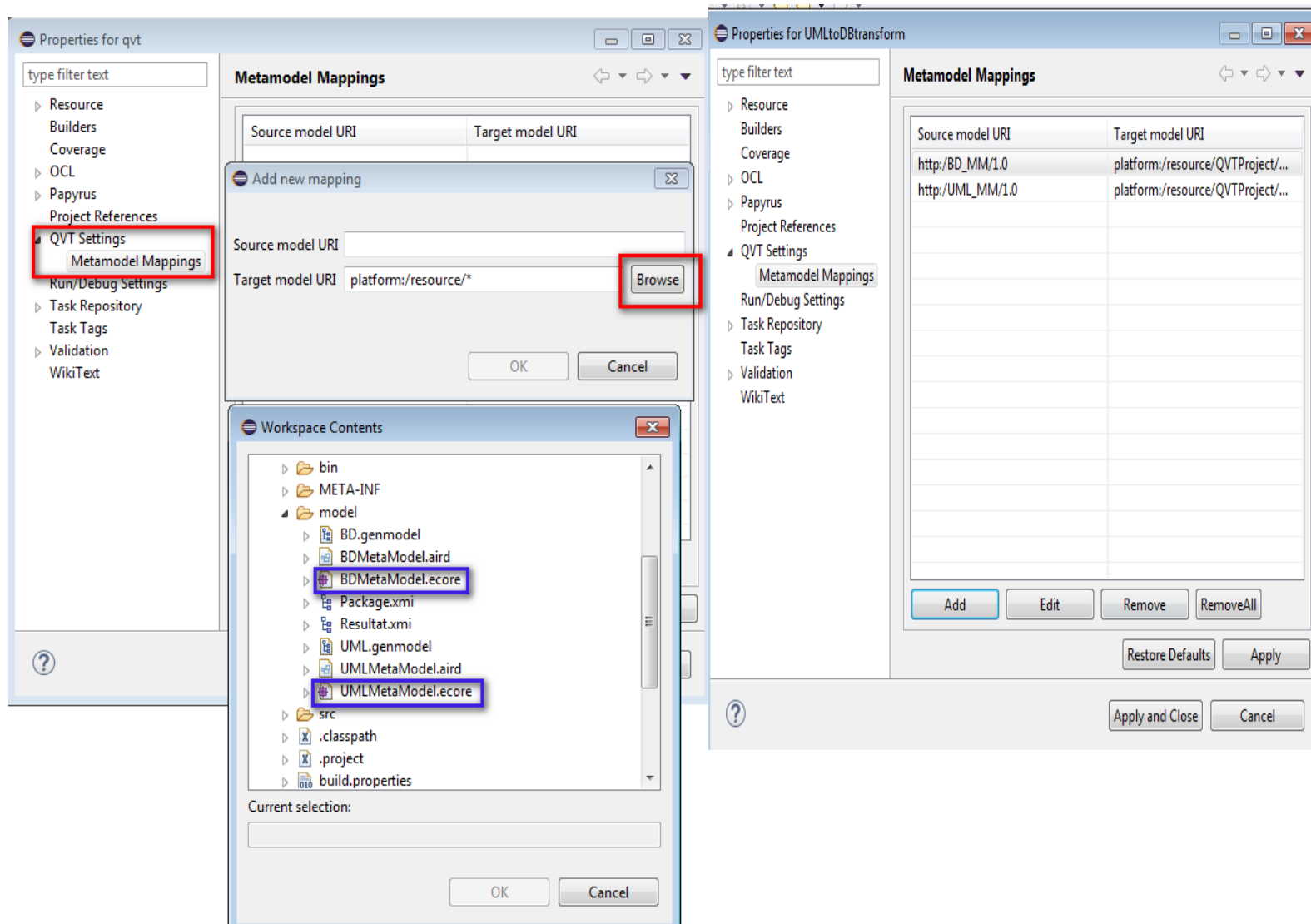


Cocher ensuite « Create artifacts in new QVT Operational Project » et sélectionner
« Operational QVT Transformation »
Enfin nommer votre fichier .qvto



6. Configuration de mapping : association de méta modèle

Clic droit sur le projet → propriété



The screenshot shows the Eclipse IDE interface for configuring metamodel mappings. The 'Properties for qvt' dialog is open, with the 'Metamodel Mappings' tab selected. The 'Add new mapping' dialog is also open, showing the 'Source model URI' and 'Target model URI' fields. The 'Workspace Contents' dialog is open, showing the project structure with 'BDMetaModel.ecore' and 'UMLMetaModel.ecore' selected. The 'Properties for UMLtoDBtransform' dialog is also open, showing the 'Metamodel Mappings' table.

Source model URI	Target model URI
http://BD_MM/1.0	platform:/resource/QVTProject/...
http://UML_MM/1.0	platform:/resource/QVTProject/...

```

1 modeltype UML uses "http://UML_MM/1.0";
2
3 modeltype DB uses "http://BD_MM/1.0";
4
5 transformation NewTransformation(in umlSource:UML, out DBTarget:DB);
6
7 main() {
8
9   umlSource.objects()[Package] -> map packageToSchema();
10
11 }
12
13 mapping Package::packageToSchema(): Schema {
14   name := "Schema";
15
16   table += umlSource.objects()[Class] -> map classToTable();
17
18 }
19
20 mapping Class::classToTable() : Table{
21   result.name := self.name;
22
23 }
24
  
```

6. Description des règles de transformations

- A tout package UML correspond un schéma
- A toute classe UML du package correspond une table dans le schéma
- A toute propriété UML d'une classe correspond une colonne dans la table
 - Si le type de la propriété est une classe UML, la colonne est une clé étrangère
 - Si le type de la propriété est un type de donnée UML, la colonne est du type correspondant
- A tout type de donnée UML d'un package correspond un type de donnée dans le schéma

7. Exécution de transformation

Clic droit sur votre fichier de configuration (.qvt) ➔ Run as ➔ Run Configuration

