Introduction à XText pour la création d'un langage spécifique à la description de graphe

par Serge BACHMANN (Page perso)

Date de publication : 28/10/2011

Dernière mise à jour : 7/11/2011

Ce tutoriel s'intéresse à l'utilisation du framework Java XText via la plateforme Eclipse pour la création d'un DSL spécifique. Un langage de description de graphe est défini à partir d'un méta langage. Un éditeur syntaxique coloré est créé à partir de ce langage spécifique. Un graphe est édité, on accède à la description du graphe généré.

0 - Pré-requis logiciels....3
1 - Introduction....3
1.1 - Spécification....3
1.2 - Lancer la plateforme....3
1-3 - Installation de XText....3
1-4 - Perspective « Plugin Development »....6
2 - Projet « ... graphemodel »....6
2-1 - Notre grammaire....8
2-2 - Génération....9
2-3 - Edition du fichier MANIFEST....11
2-4 - Le fichier plugin.xml....12
2-5 - Le package « org.eclipse.xtext.example.graphemodel »....12
3 - Projet « ... graphemodel.ui »....12
3-1 - Commande View Model....13
3-2 - Définition du « Handler »....14
4 - Test....17
4-1 - L'éditeur....18
4-2 - La commande « View Model »....18
5 - Une autre grammaire....18
6 - Conclusion....19

0 - Pré-requis logiciels

La version Eclipse utilisée dans ce tutoriel est Eclipse Indigo 3.7.1 en incluant la distribution Eclipse Modeling Tools.

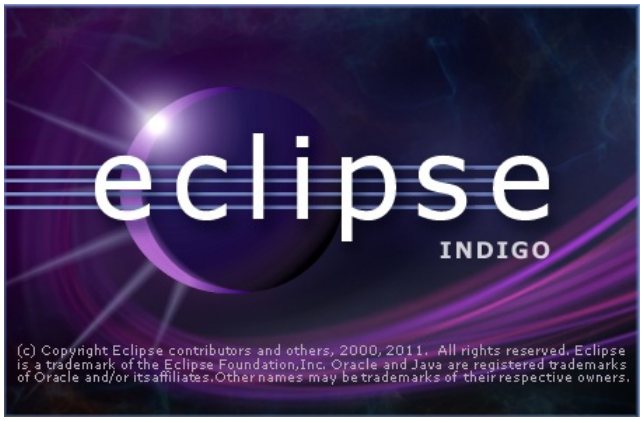
1 - Introduction

1.1 - Spécification

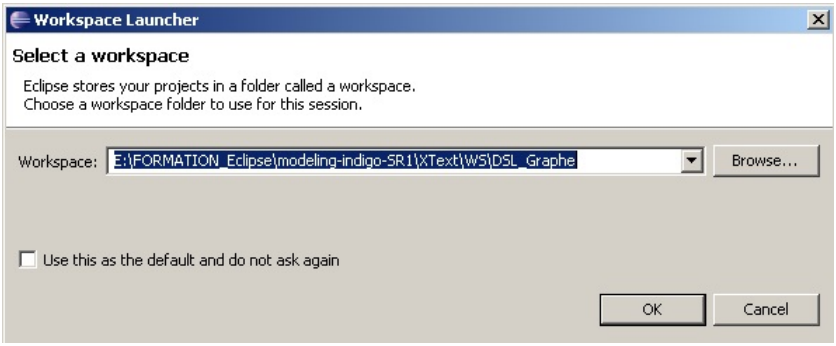
Construire un éditeur syntaxique d'un graphe.

1.2 - Lancer la plateforme

Double cliquer : *eclipse.exe* ou le raccourci vers cet exécutable si vous l'avez créé dans le répertoire destiné à recevoir les « workspaces ». La plate-forme « Eclipse » est lancée :



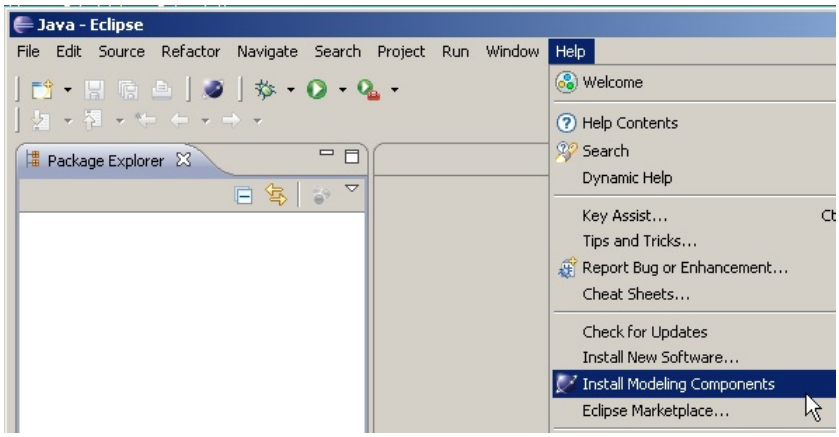
On choisi le « Workspace » :



Cliquer « OK ». Fermer la fenêtre « Welcome ».

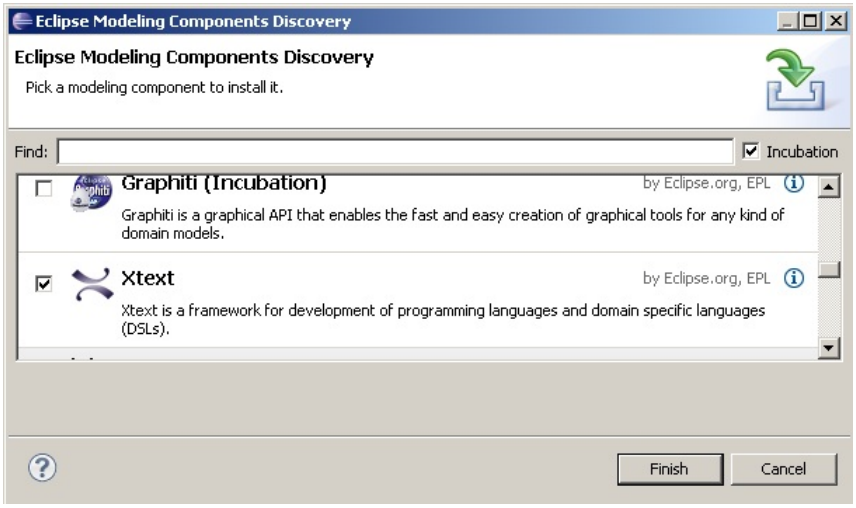
1-3 - Installation de XText

Sélectionner la commande « Help Install Modeling Components »



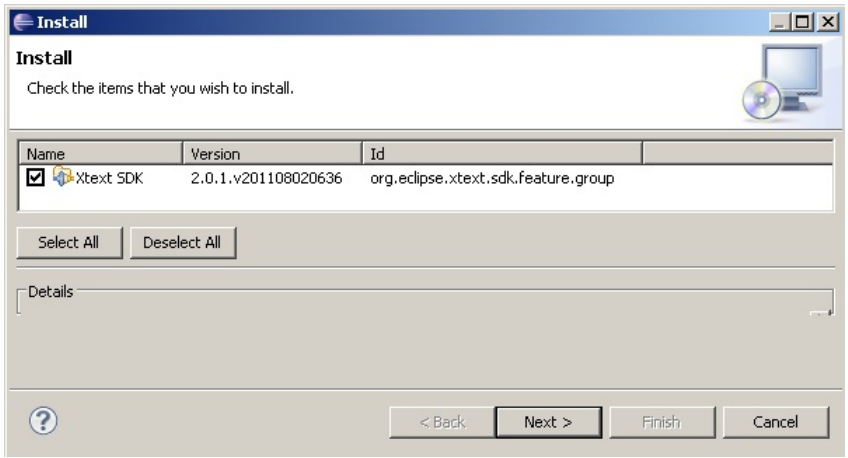
Dans « Eclipse Modeling Components Discovery □ Eclipse Modeling Components Discovery »

Sélectionner Xtext :

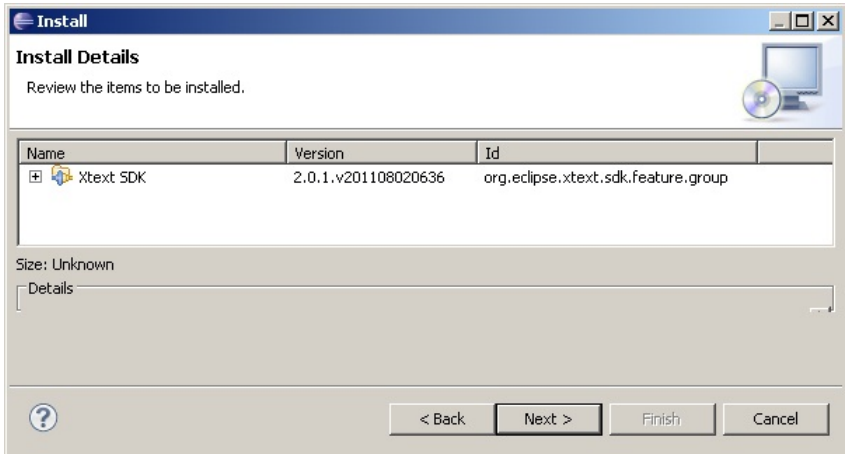


Cliquer « Finish ».

Confirmer l'installation de « XText »

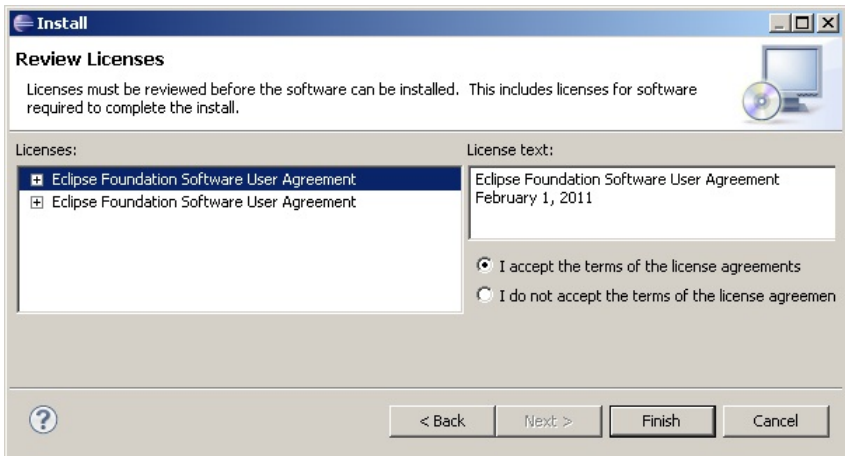


Cliquer « Next > »

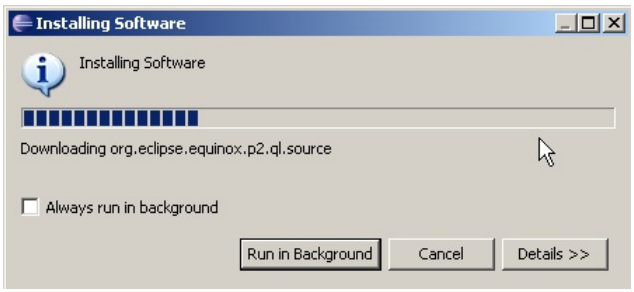


Cliquer « Next > »

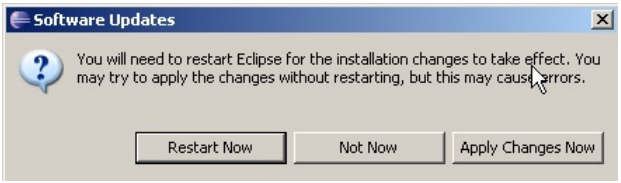
Accepter la licence :



Cliquer « Finish », Xtext s'installe.



Il faut relancer Eclipse:

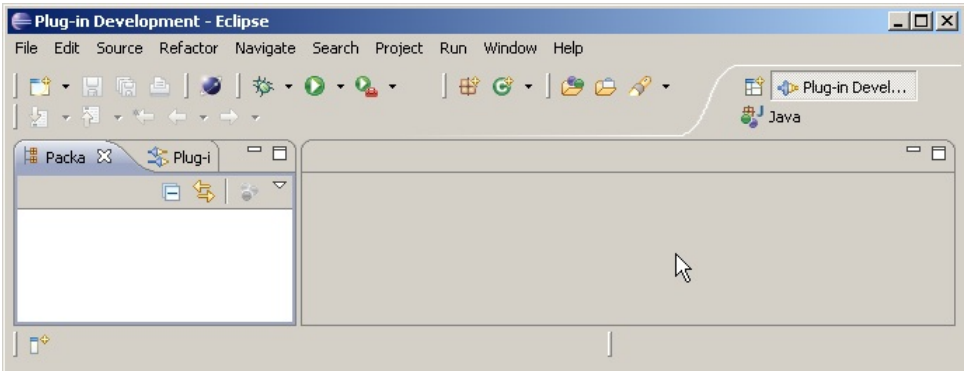


Cliquer sur le bouton « Restart Now ».

Nous conserverons le même « Workspace » puis cliquer sur « OK ».

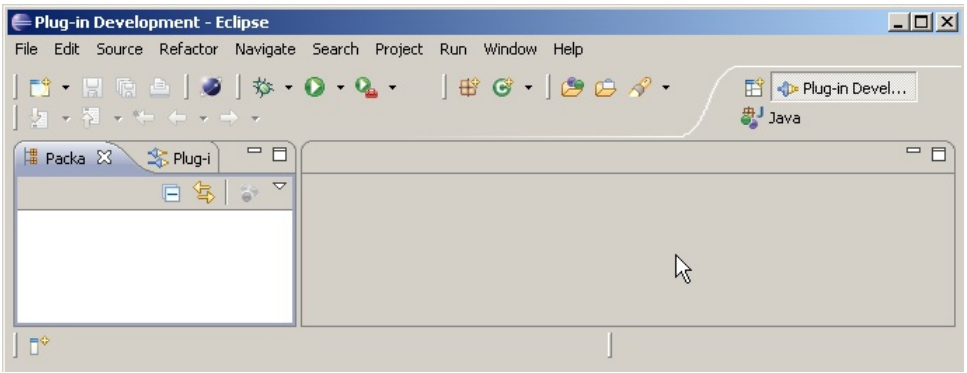
1-4 - Perspective « Plugin Development »

Sélectionner la commande « Window Open Perspective > Other... ». Dans « Open Perspective » sélectionner « Plug-in Development », fermer les vues inutiles ce qui donne :

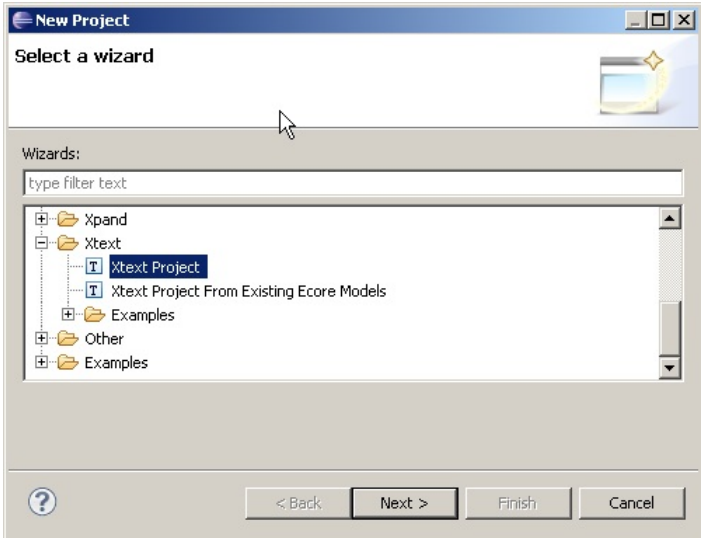


2 - Projet « ... graphemodel »

Sélectionner la commande « File > New > Project... »

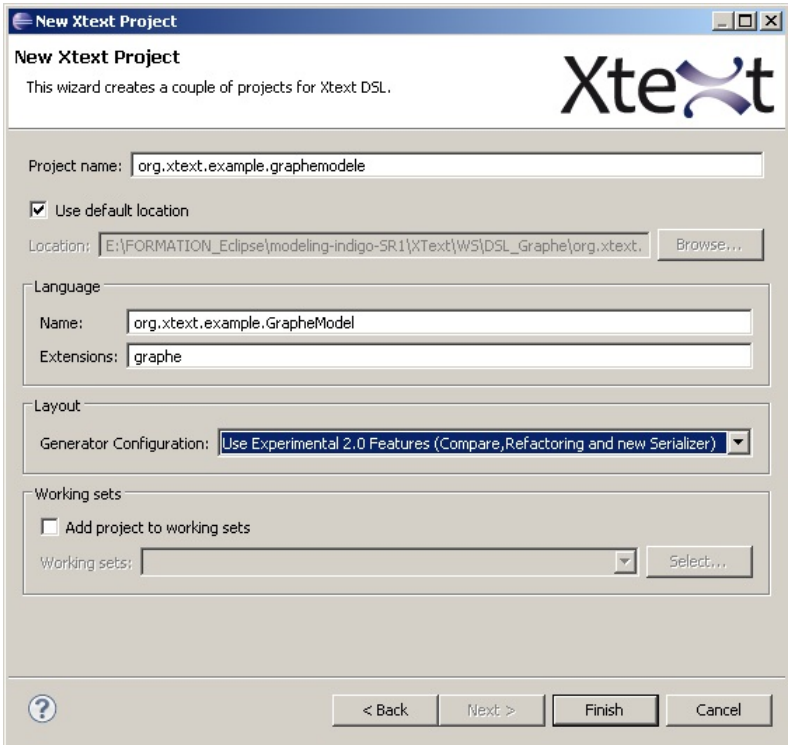


Dans « New Project Select a wizard » sélectionner « Xtext Project »



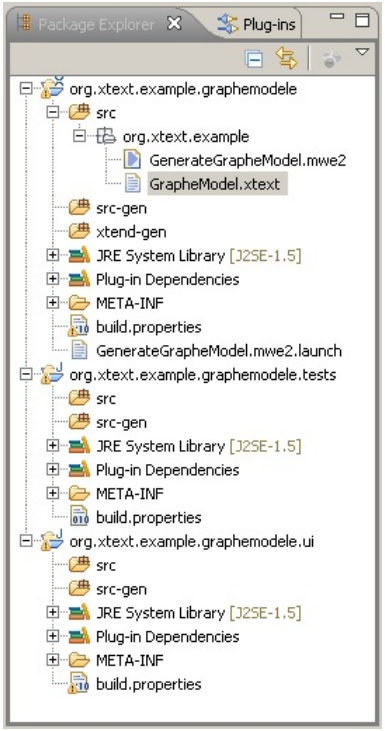
Cliquer « Next > »

Initialiser: « New Xtext Project New Xtext Project »



Cliquer « Finish »

Les projets suivants sont créé:



Dans *org.xtext.example.graphemodele* > *src* > *org.xtext.example* le fichier *GrapheModel.xtext* définit la grammaire de notre langage.

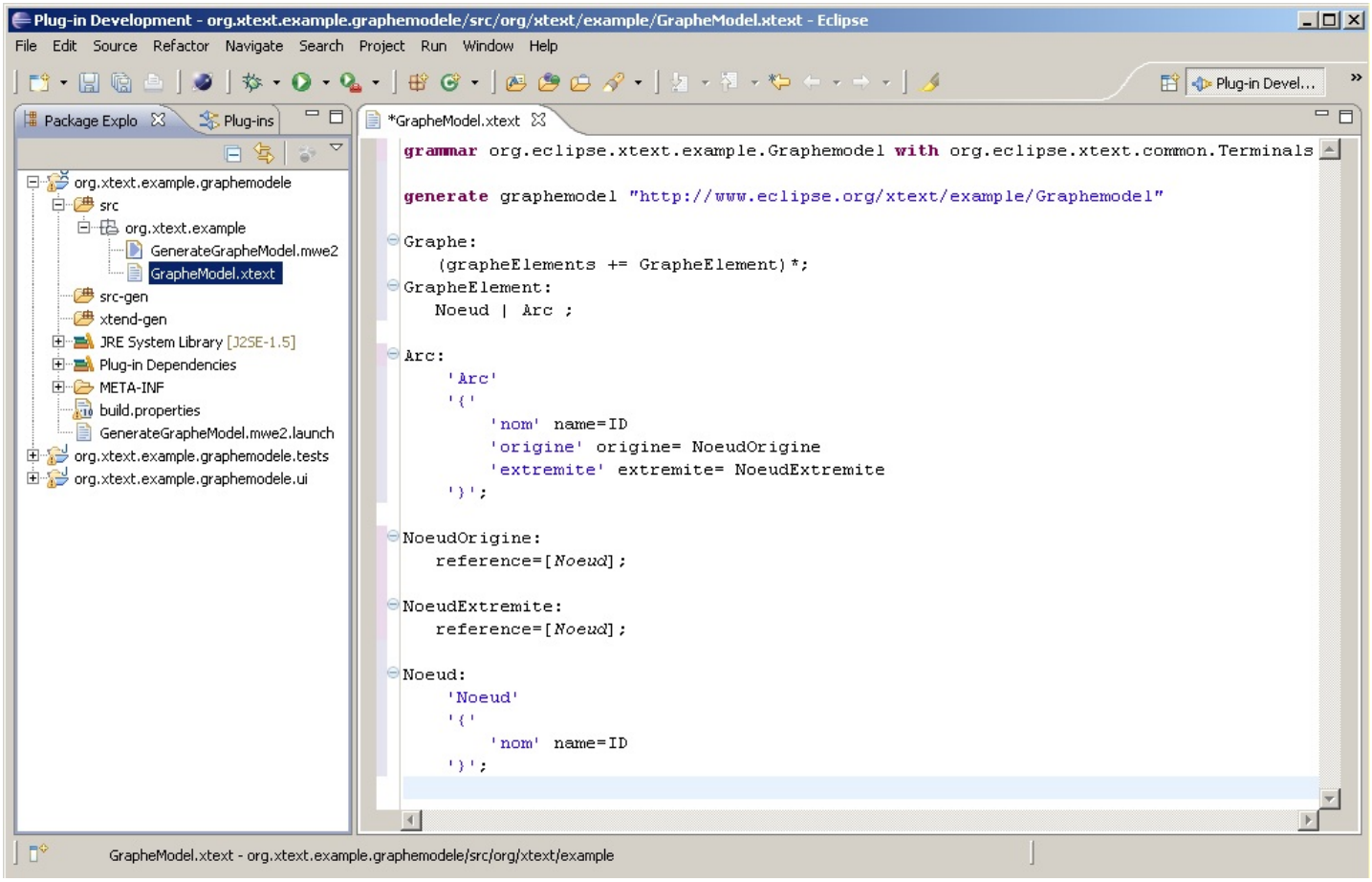
⚠ *Chaque projet contient les répertoires src et src-gen. Nous ne devrons intervenir que sur les répertoires src.*

2-1 - Notre grammaire

On édite le fichier « GrapheModel.xtext » :

```
grammar org.eclipse.xtext.example.Graphemodel with org.eclipse.xtext.common.Terminals

generate graphemodel "http://www.eclipse.org/xtext/example/Graphemodel"

Graphe:
   (grapheElements += GrapheElement)*;
GrapheElement:
   Noeud | Arc ;

Arc:
 'Arc'
 '{'
  'nom' name=ID
  'origine' origine= NoeudOrigine
  'extremite' extremite= NoeudExtremite
 '}';

NoeudOrigine:
 reference=[Noeud];

NoeudExtremite:
 reference=[Noeud];

Noeud:
 'Noeud'
 '{'
  'nom' name=ID
 '}';
```

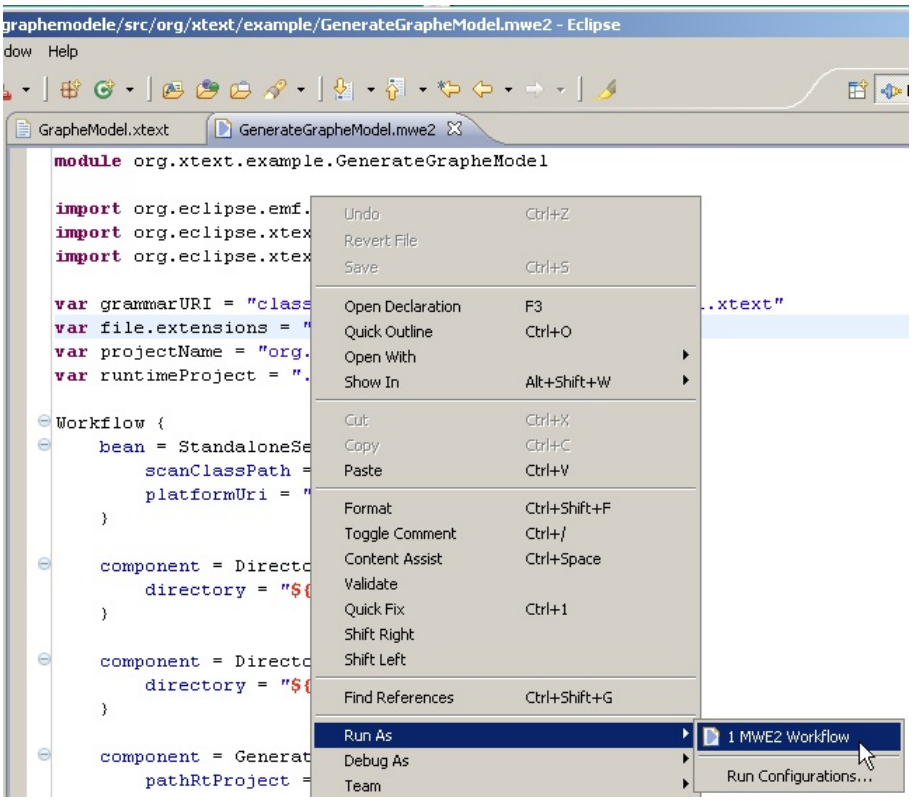
Ce qui donne après sauvegarde :



2-2 - Génération

Dans le projet « org.xtext.example.graphemodel », sous « src > org.xtext.example » ouvrir : « GenerateGrapheModel.mwe2 ».

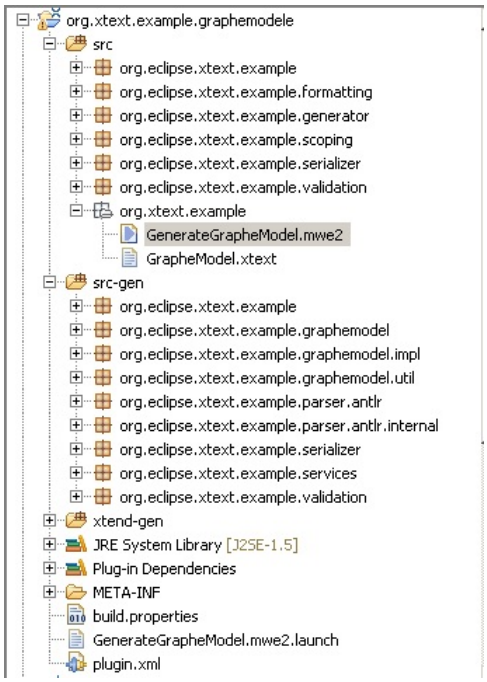
Dans l'éditeur faire un clic droit pour faire monter le menu contextuel et sélectionner « Run As > NWE2 Workflow » :



Le message suivant apparaît dans la vue « Console » saisir « y » puis « enter » comme demandé :

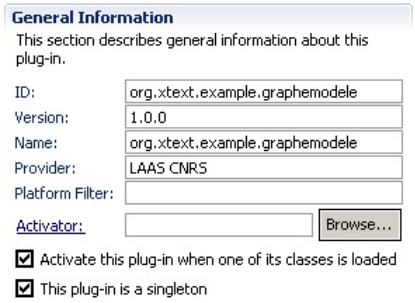
```
0    [main] INFO  lipse.emf.mwe.utils.StandaloneSetup  - Registering platform uri 'E:\ECLIPSE\XText
\WORKSPACES\DSL_Graphe'

*ATTENTION*
It is recommended to use the ANTLR 3 parser generator (BSD licence - http://www.antlr.org/
license.html).
Do you agree to download it (size 1MB) from 'http://download.itemis.com/antlr-generator-3.0.1.jar'?
 (type 'y' or 'n' and hit enter)y
```

Le parser « ANTLR 3 » est installé. Après génération le projet *org.xtext.example.graphemodele* devient :



2-3 - Edition du fichier MANIFEST

Dans le « Plugin manifest Editor » onglet « Overview » modifier le « provider »:



Faire une sauvegarde.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: org.xtext.example.graphemodele
Bundle-Vendor: LAAS CNRS
Bundle-Version: 1.0.0
Bundle-SymbolicName: org.xtext.example.graphemodele; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: org.eclipse.xtext;bundle-version="2.0.0";visibility:=reexport,
 org.apache.log4j;bundle-version="1.2.15";visibility:=reexport,
 org.apache.commons.logging;bundle-version="1.0.4";resolution:=optional;visibility:=reexport,
 org.eclipse.xtext.generator;resolution:=optional,
 org.eclipse.emf.codegen.ecore;resolution:=optional,
 org.eclipse.emf.mwe.utils;resolution:=optional,
 org.eclipse.emf.mwe2.launch;resolution:=optional,
 org.eclipse.xtext.util,
 org.eclipse.emf.ecore,
 org.eclipse.emf.common,
 org.antlr.runtime,
 org.eclipse.xtext.common.types
Import-Package: org.apache.log4j,
 org.apache.commons.logging,
 org.eclipse.xtext.xbase.lib,
 org.eclipse.xtext.xtend2.lib
```

```
Bundle-RequiredExecutionEnvironment: J2SE-1.5
Export-Package: org.eclipse.xtext.example,
 org.eclipse.xtext.example.services,
 org.eclipse.xtext.example.graphemodel,
 org.eclipse.xtext.example.graphemodel.impl,
 org.eclipse.xtext.example.graphemodel.util,
 org.eclipse.xtext.example.serializer,
 org.eclipse.xtext.example.parser.antlr,
 org.eclipse.xtext.example.parser.antlr.internal,
 org.eclipse.xtext.example.validation
```

2-4 - Le fichier plugin.xml

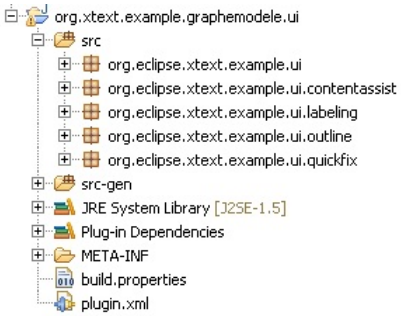
```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>
  <extension point="org.eclipse.emf.ecore.generated_package">
    <package
       uri = "http://www.eclipse.org/xtext/example/Graphemodel"
       class = "org.eclipse.xtext.example.graphemodel.GraphemodelPackage"
       genModel = "org/eclipse/xtext/example/Graphemodel.genmodel" />
  </extension>
</plugin>
```

2-5 - Le package « org.eclipse.xtext.example.graphemodel »

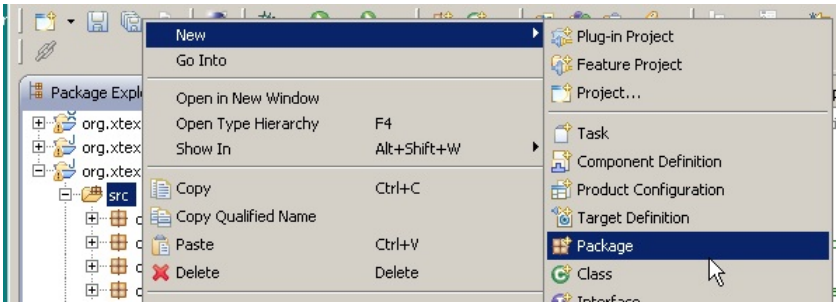
Dans le projet « org.xtext.example.graphemodele » sous « src-gen » le package ci-dessous est généré:

3 - Projet « ... graphemodel.ui »

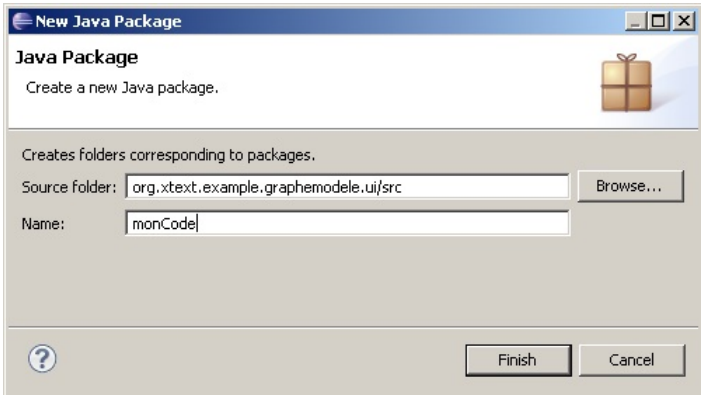
Ce projet défini l'interface utilisateur.



On crée un « package » pour recevoir notre code (Handler)



Nommer le « package » :

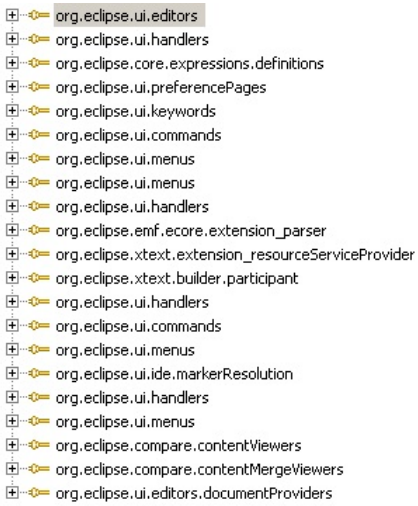


Cliquer « Finish ».

Le « package » est créé :

3-1 - Commande View Model

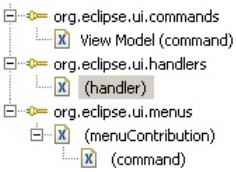
Une commande « View Model » est ajoutée pour visualiser le modèle édité. Ouvrir le « Plugin Manifest Editor » sélectionner l'onglet « Extension », initialement nous avons :



On édite le fichier « plugin.xml » pour ajouter une commande:

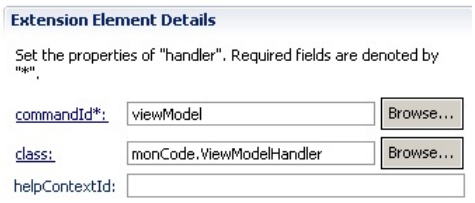
```
  <extension
      point="org.eclipse.ui.commands">
   <command
         id="viewModel"
         name="View Model">
   </command>
</extension>
<extension
      point="org.eclipse.ui.handlers">
   <handler
         class="monCode.ViewModelHandler"
         commandId="viewModel">
   </handler>
</extension>
<extension
      point="org.eclipse.ui.menus">
   <menuContribution
         allPopups="false"
         locationURI="menu:org.eclipse.ui.main.menu?after=additions">
      <command
            commandId="viewModel"
            style="push">
      </command>
   </menuContribution>
</extension>
```

Revenons à l'onglet « Extensions » on a ajouté :

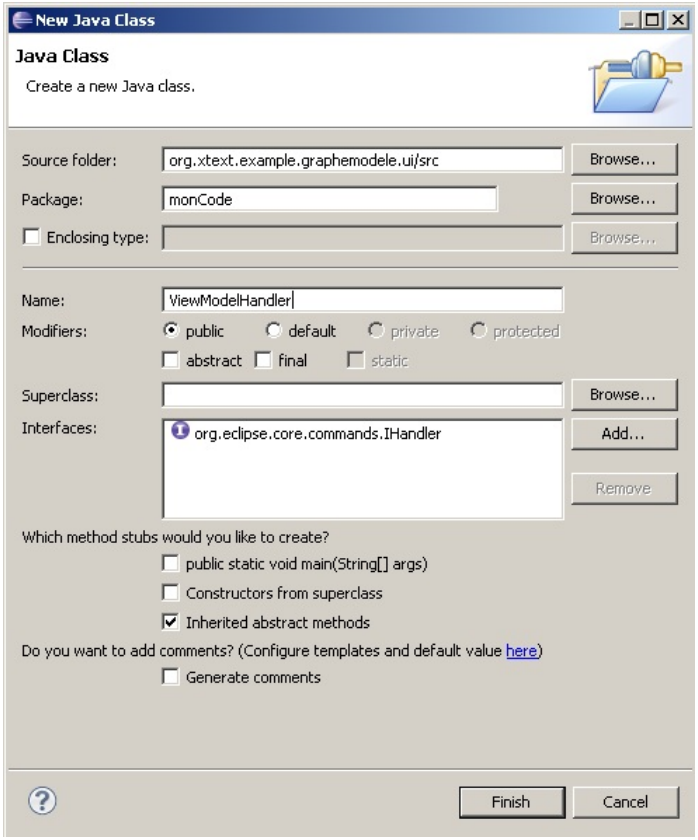


3-2 - Définition du « Handler »

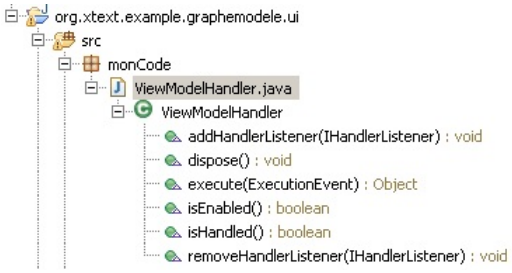
On sélectionne (handler) dans « Extension Element Details » cliquer « class: »



« New Java Class □ Java Class » est initialisé:



Cliquer « Finish » on obtient:



ce qui correspond au code:

```
package monCode;

import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.core.commands.IHandler;
import org.eclipse.core.commands.IHandlerListener;

public class ViewModelHandler implements IHandler {

 @Override
```

```
 public void addHandlerListener(IHandlerListener handlerListener) {
  // TODO Auto-generated method stub
 }

 @Override
 public void dispose() {
  // TODO Auto-generated method stub
 }

 @Override
 public Object execute(ExecutionEvent event) throws ExecutionException {
  // TODO Auto-generated method stub
  return null;
 }

 @Override
 public boolean isEnabled() {
  // TODO Auto-generated method stub
  return false;
 }

 @Override
 public boolean isHandled() {
  // TODO Auto-generated method stub
  return false;
 }

 @Override
 public void removeHandlerListener(IHandlerListener handlerListener) {
  // TODO Auto-generated method stub
 }
}
```

On édite le handler:

```
package monCode;

import java.util.ListIterator;

import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.core.commands.IHandler;
import org.eclipse.core.commands.IHandlerListener;
import org.eclipse.emf.common.util.EList;
import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.EObject;
import org.eclipse.emf.ecore.resource.Resource;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.handlers.HandlerUtil;
import org.eclipse.xtext.example.GraphemodelStandaloneSetup;
import org.eclipse.xtext.example.graphemodel.Arc;
import org.eclipse.xtext.example.graphemodel.Graphe;
import org.eclipse.xtext.example.graphemodel.GrapheElement;

public class ViewModelHandler implements IHandler {

 @Override
 public void addHandlerListener(IHandlerListener handlerListener) {
 }

 @Override
 public void dispose() {
 }

 @Override
 public Object execute(ExecutionEvent event) throws ExecutionException {
```

```
  Shell shell = HandlerUtil.getActiveShell(event);

  ISelection selection = HandlerUtil.getCurrentSelection(event);
  String nomFich = selection.toString();
  nomFich = nomFich.substring(2, nomFich.length()-1);
  //MessageDialog.openInformation(shell, "Working with EMF Models", nomFich);


  new GraphemodelStandaloneSetup().createInjectorAndDoEMFRegistration();
  ResourceSet rs = new ResourceSetImpl();

  Resource resource;
  try {
   resource = rs.getResource(URI.createURI("platform:/resource/"+nomFich),true);
   // autre possibilité:
   //resource = rs.getResource(URI.
   //    createPlatformResourceURI(nomFich,true), true);
  } catch (Exception e) {
   // e.printStackTrace();
   MessageDialog.openError( shell, "Working with EMF Models", "Select a file in Project Explorer");
   return null;
  }

  StringBuffer message = new StringBuffer("Modèle: "+ nomFich + "\n");
  EObject eobject = resource.getContents().get(0);
  Graphe graphe = (Graphe) eobject;
  EList<GrapheElement> grapheElementList = graphe.getGrapheElements();
  ListIterator<GrapheElement> li = grapheElementList.listIterator();
  while (li.hasNext()){
   GrapheElement grapheElement = li.next();
   if (grapheElement instanceof Arc){
    Arc arc = (Arc)grapheElement;
    message.append(arc.getName()
      + "("
      + arc.getOrigine().getReference().getName()
      + " -> "
      + arc.getExtremite().getReference().getName()
      + ")\n"
    );
   }
  }
  MessageDialog.openInformation(shell, "Working with EMF Models", message.toString());

  return null;
 }

 @Override
 public boolean isEnabled() {
  return true;
 }

 @Override
 public boolean isHandled() {
  return true;
 }

 @Override
 public void removeHandlerListener(IHandlerListener handlerListener) {
 }
}
```

Faire une sauvegarde générale.

4 - Test

Faire un clic droit sur le projet « org.xtext.exemple.graphemodel.ui » dans le menu contextuel sélectionner la commande « Run As > Run Configurations... ». Dans « Run Configuration » double cliquer « Eclipse Application ». Nommer la configuration « DSL_Graphe ». Faire « Apply » puis « Run »

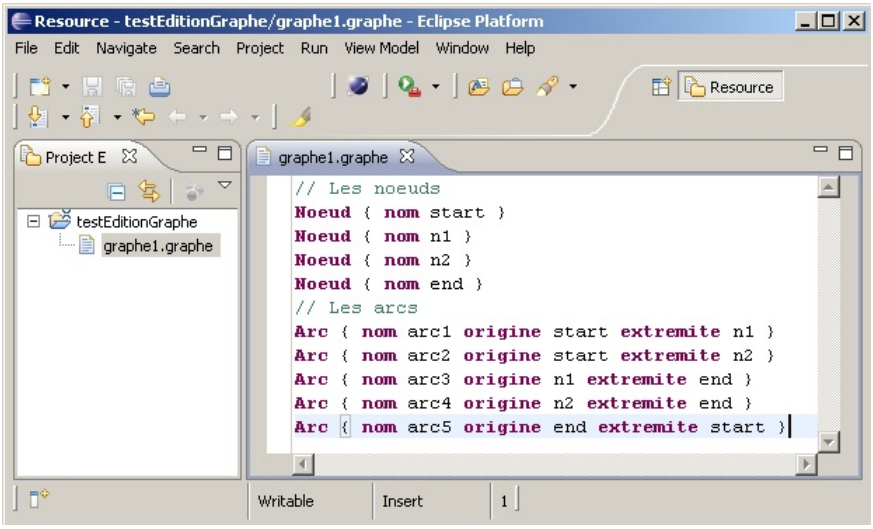
Sous la nouvelle plate-forme, on crée un simple projet de type « General > Project » de nom « testEditionGraphe ». Sous ce projet on crée un fichier « graphe1.graphe » le postfixe « graphe » induit le choix de l'éditeur.

4-1 - L'éditeur

L'éditeur est un éditeur syntaxique coloré.

Le source définissant une instance de graphe :

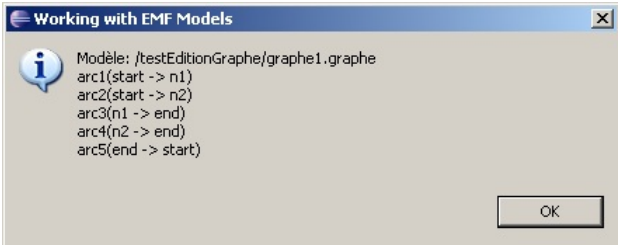
```
// Les noeuds
Noeud { nom start }
Noeud { nom n1 }
Noeud { nom n2 }
Noeud { nom end }
// Les arcs
Arc { nom arc1 origine start extremite n1 }
Arc { nom arc2 origine start extremite n2 }
Arc { nom arc3 origine n1 extremite end }
Arc { nom arc4 origine n2 extremite end }
Arc { nom arc5 origine end extremite start }
```



Faire une sauvegarde après l'édition.

4-2 - La commande « View Model »

Dans le « Project Explorer » sélectionner « graphe1.graphe ». La commande « View Model » accède à la représentation du modèle édité et en donne une représentation textuelle:

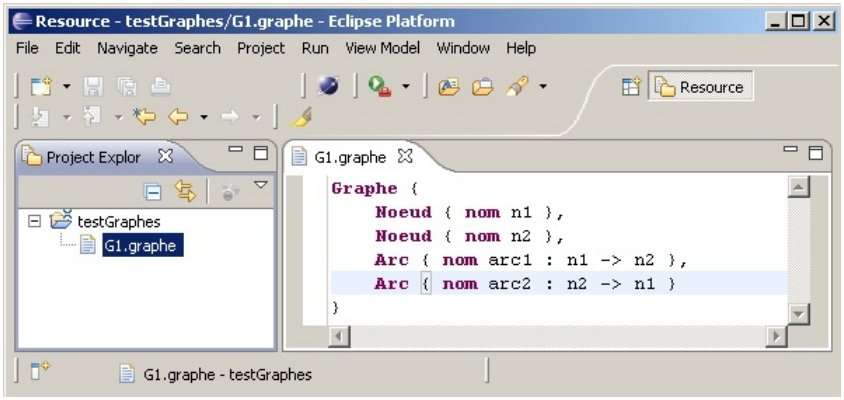


5 - Une autre grammaire

On peut définir une autre grammaire:

```
grammar org.eclipse.xtext.example.Graphemodel with org.eclipse.xtext.common.Terminals

generate graphemodel "http://www.eclipse.org/xtext/example/Graphemodel"

Graphe:
   {Graphe}
   'Graphe'
   '{'
      ( grapheElements += GrapheElement(',' grapheElements += GrapheElement)* )?
   '}'
   ;

GrapheElement:
   Noeud | Arc ;

Arc:
 'Arc'
 '{'
  'nom' name=ID
  ':'  origine= NoeudOrigine
  '->' extremite= NoeudExtremite
    '}';

NoeudOrigine:
   reference=[Noeud];

NoeudExtremite:
   reference=[Noeud];

Noeud:
 'Noeud'
 '{'
  'nom' name=ID
    '}';
```

ce qui donne la syntaxe ci-dessous:



La commande « View Model » reste valide.

6 - Conclusion

A partir d'une grammaire nous avons généré un éditeur syntaxique coloré. Nous avons utilisé Java pour accéder au modèle généré. D'autres solutions sont possibles en particulier pour écrire des générateurs de code à partir du langage spécialisé « M2T Xpand ».

Pour plus de précision voir:

- La documentation en ligne (Help > HelpContents)
- Le site officiel : http://www.eclipse.org/Xtext/