

Xtext 学习笔记

前言

因为在 Xtext 的学习过程中发现国内的讲解 Xtext 的人太少了，连官网都得 fq，国外的教程对于 Ecore 模型在 Xtext 上的应用也比较少，所以这里在我们的课题结束之后对我们所学的对 Xtext 的理解和查到的资料进行一个汇总总结，希望能让大家少走一些弯路。

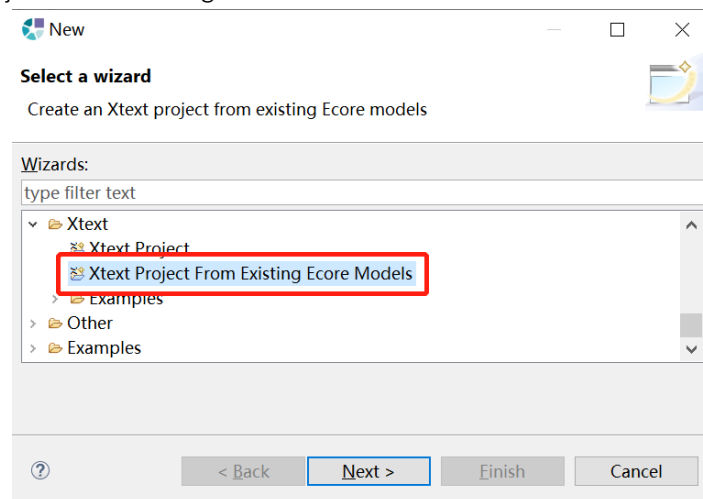
理解

Xtext 是一种用于定义自定义的领域特定语言 (DSL) 的框架，Xtend 是一种自定义语法框架的语言，我们可以通过已建立好的 Ecore 模型生成 Xtext 项目，使用 Ecore 模型中的结构，定义 DSL，功能强大。

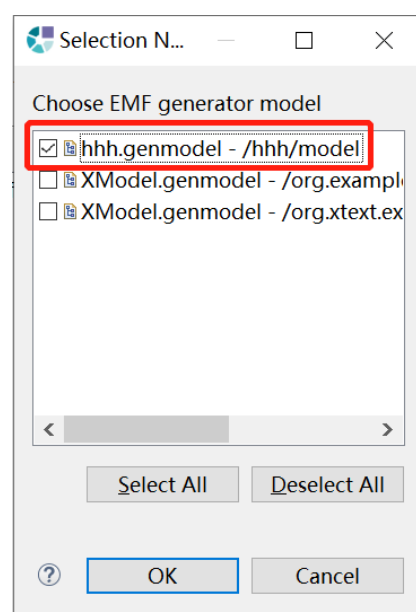
操作

1. 由于我们软工课设完成的是基于已建立好的 Ecore 模型生成 Java 代码，需要使用 Xtext Project From Existing Ecore Models 来建立 Xtext 项目。

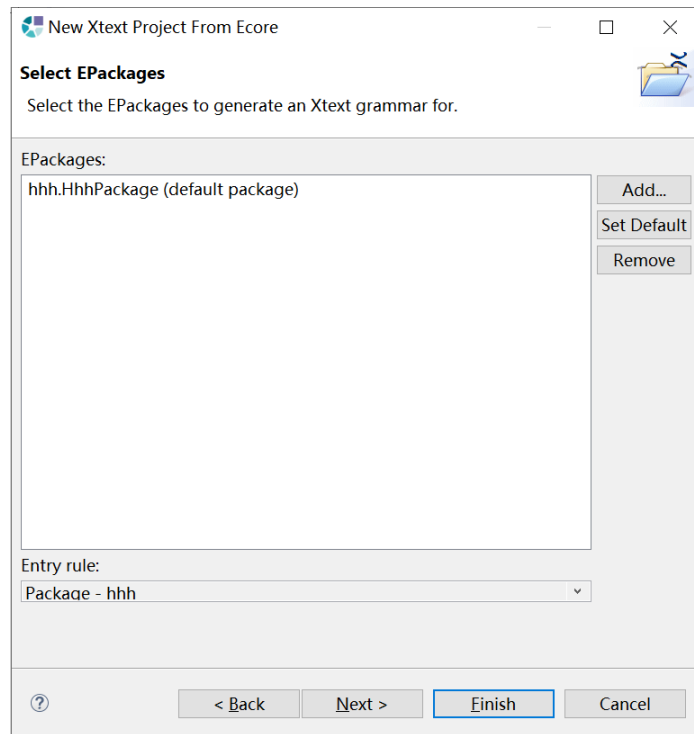
选择 Xtext Project From Existing Ecore Models:



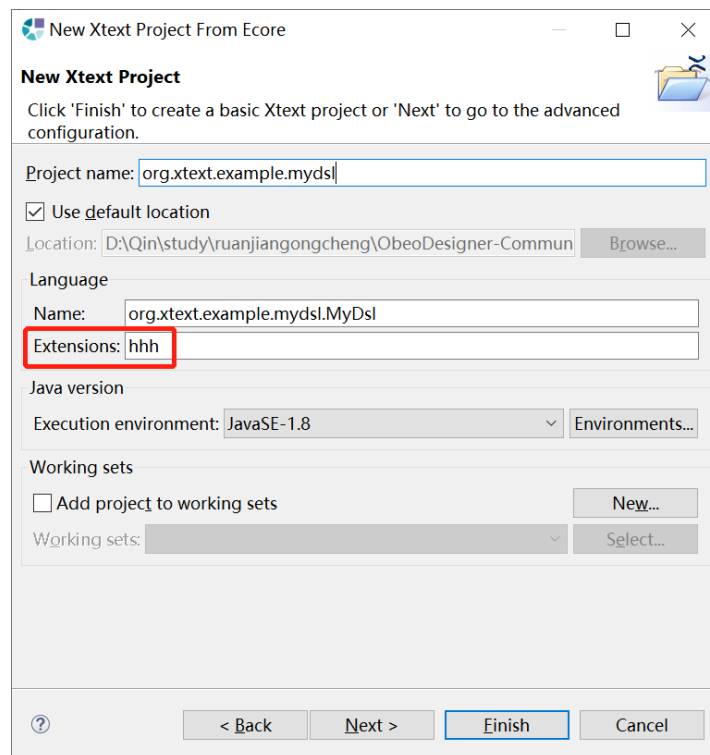
导入 ecore 模型对应的 genmodel 文件:



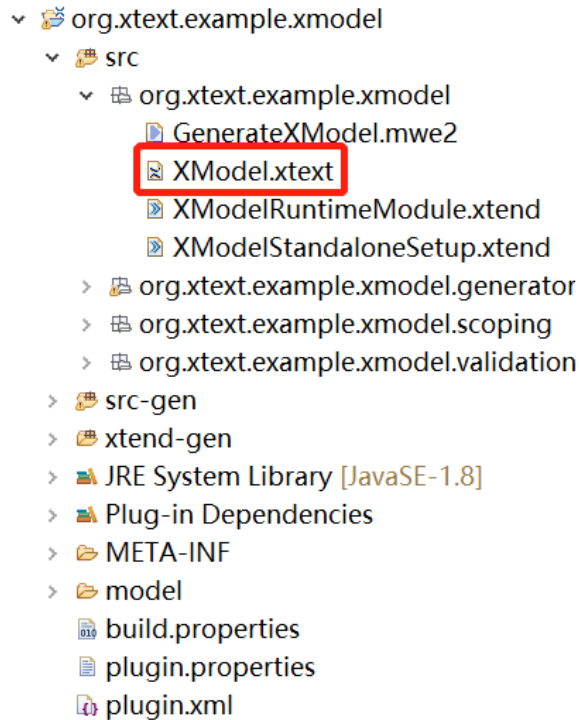
Entry rule 选择 Package (一定要选择最顶层的 EClass!!!):



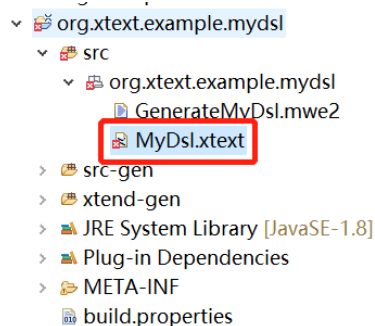
Extensions 部分改成自定义 ecore 模型的后缀：



生成的目录如下，其中.xtext 文件是根据选择的.ecore 文件自动生成的 DSL 语法：



若产生报错:



```
// automatically generated by Xtext
grammar org.xtext.example.mydsl.MyDsl with org.eclipse.xtext.common.Terminals

import "http://www.example.org/hhh"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore
```

Package returns Package:

```
{Package}
'Package'
name=EString
'{'
    ('class' '{' class+=XClass ( "," class+=XClass)* '}' )?
'}';
```

可尝试以下解决办法:

- 1) 右键单击包含要导入的 ecore 模型的项目: Configure >> Convert to Xtext project
- 2) 将项目添加到我的项目构建路径 (Java Build Path) 中: right-click >> Properties >> Java Build Path >> Projects >> Add.. , 然后只需选择包含要导入的模型的项目即可。

来源 ↓

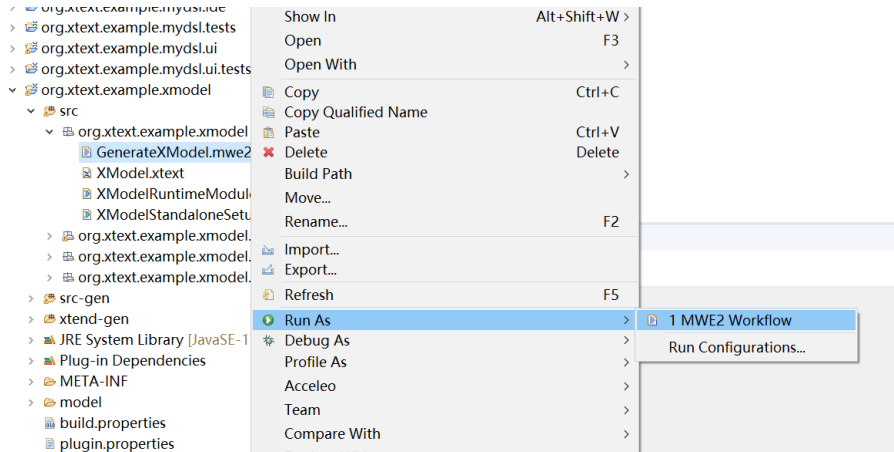
<https://stackoverflow.com/questions/49687485/xtext-import-an-existing-ecore-model>

如果你需要从头开始（不使用现有 ecore 模型），请参考 ↓

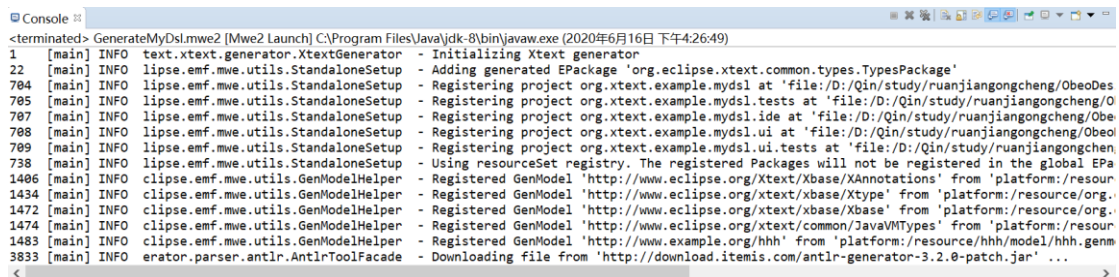
https://www.eclipse.org/Xtext/documentation/301_grammarlanguage.html

注：对于 Xtext 文件语法语言的描述。

2. 运行 MWE2 流程



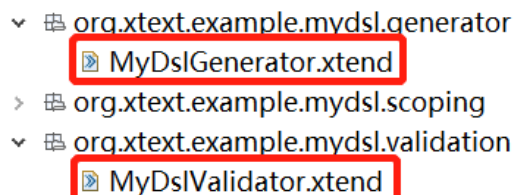
运行流程时可能会卡顿在下载一个 jar 包的地方，



这个 jar 包我们找到了，但是不知道放在哪儿，下载链接 ↓

https://bugs.eclipse.org/bugs/show_bug.cgi?id=470799

3. 根据官方的 15 Minutes Tutorial – Extended 教程编写代码生成器 Generator 类的 doGenerate 函数和创建自定义验证规则 Validator 类：



教程 ↓

https://www.eclipse.org/Xtext/documentation/103_domainmodelnextsteps.html

注：编写代码生成器 Generator 类的 doGenerate 函数和创建自定义验证规则 Validator 类在 .xtend 文件中，不能写在 .java 文件中！

1) 编写代码生成器 Generator 类的 doGenerate 函数

a) 生成 java 文件，以类名为 java 文件名

```

override void doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) {
    for (e : resource.allContents.toIterable.filter(XClass)) {
        fsa.generateFile(
            e.fullyQualifiedName.toString("/") + ".java", e.compile)
    }
}

```

b) 编写 package 语句定义包

```

private def compile(XClass e) '''
    «IF e.eContainer.fullyQualifiedName != null»
        package «e.eContainer.fullyQualifiedName»;
    «ENDIF»

```

c) 编写 class 声明语句, 判断是否为抽象类以及是否存在继承关系, 若存在父类则 extends 对应的父类

```

public «IF e.isAbstract == true»abstract «ENDIF»class «e.name» «IF e.extend != null»extends «e.extend»«ENDIF» {

```

d) 编写注释语句的格式

```

«IF e.description != null» // «e.description»«ENDIF»

```

e) 对类中的每一个变量 (property) 和方法 (operation) 调用相应的语句规范

```

«FOR f : e.property»
    «f.compilePro»
«ENDFOR»

«FOR f : e.operation»
    «f.compileOp»
«ENDFOR»
...

```

f) 编写 property, 类中的变量声明语句, 以及每个变量对外的 get、set 方法, 方法名为 get/set 加上第一个字母大写的变量名, visibility 表示访问修饰符, xtype 表示变量类型, name 表示变量名字, description 表示对于这个变量的注释

```

private def compilePro(Property f) '''
«f.visibility» «f.xtype» «f.name»; «IF f.description != null» // «f.description»«ENDIF»

public «f.xtype» get«f.name.toFirstUpper»() {
    return «f.name»;
}

public void set«f.name.toFirstUpper»(«f.xtype» «f.name») {
    this.«f.name» = «f.name»;
}
...

```

g) 编写 operation, 类中的方法声明语句, description 表示对于这个方法的注释, visibility 表示访问修饰符, xtype 表示方法的返回值类型, name 表示方法名, 然后循环调用方法的形参 (parameter) 语句规范打印形参, 当打印到最后一个形参时不打印“,”, 并在方法体中打印“// to do”, 表示方法体待完成

```

private def compileOp(Operation f) '''
    «IF f.description != null» // «f.description»«ENDIF»
    «f.visibility» «f.xtype» «f.name» («FOR f1 : f.parameter» «IF f.parameter.Last != f1» «f1.compilePa», «ENDIF»
    «IF f.parameter.Last == f1» «f1.compilePa» «ENDIF» «ENDFOR» ) {
        // to do
    }
...

```

- h) 编写 parameter, 类中方法的形参, 即“类型 形参名”

```
private def compilePa(Parameter f) '''«f.xtype»«f.name»'''
```

- 2) 编写创建自定义验证规则 Validator 类

- a) 定义 6 种错误名

```
public static final String INVALID_CLASS_NAME = 'invalidClassName';
public static final String INVALID_PROPERTY_NAME = 'invalidPropertyName';
public static final String CLASS_NAME_EMPTY = "ClassNameEmpty";
public static final String PROPERTY_NAME_EMPTY = "PropertyNameEmpty";
public static final String OPERATION_NAME_EMPTY = "OperationNameEmpty";
public static final String PARAMETER_NAME_EMPTY = "ParameterNameEmpty";
```

- b) 验证实例类名是否以大写字母开头并报错'Name should start with a capital!'

```
@Check
def checkClassStartsWithCapital(XClass classes) {
  if (!Character.isUpperCase(classes.name.charAt(0))) {
    warning('Name should start with a capital!',
      XModelPackage.Literals.XCLASS__NAME,
      INVALID_CLASS_NAME)
  }
}
```

- c) 验证变量是否以小写字母开头并报错'Name should start with a lowercase!'

```
@Check
def checkPropertyStartsWithLowercase(Property pro) {
  if (!Character.isLowerCase(pro.name.charAt(0))) {
    warning('Name should start with a lowercase!',
      XModelPackage.Literals.PROPERTY__NAME,
      INVALID_PROPERTY_NAME)
  }
}
```

- d) 验证实例类名是否为空并报错"类名不可为空: ClassName cannot be empty!"

```
@Check
def checkClassNameIsNotEmpty(XClass classes) {
  if(classes.getName().isEmpty()) {
    error("类名不可为空: ClassName cannot be empty!",
      XModelPackage.Literals.XCLASS__NAME,
      CLASS_NAME_EMPTY);
  }
}
```

- e) 验证变量名是否为空并报错"变量名不可为空: Property Name cannot be empty!"

```
@Check
def checkPropertyNameIsNotEmpty(Property pro) {
  if(pro.getName().isEmpty()) {
    error("变量名不可为空: Property Name cannot be empty!",
      XModelPackage.Literals.PROPERTY__NAME,
      PROPERTY_NAME_EMPTY);
  }
}
```

- f) 验证函数名是否为空并报错"函数名不可为空: Operation Name cannot be

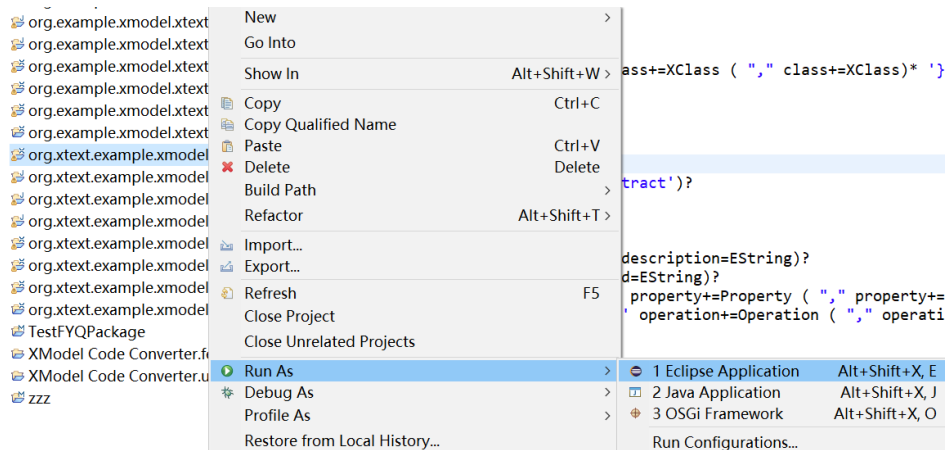
empty!"

```
@Check
def checkOperationNameIsNotEmpty(Operation op) {
  if(op.getName().isEmpty()) {
    error("函数名不可为空: Operation Name cannot be empty!",
      XModelPackage.Literals.OPERATION_NAME,
      OPERATION_NAME_EMPTY);
  }
}
```

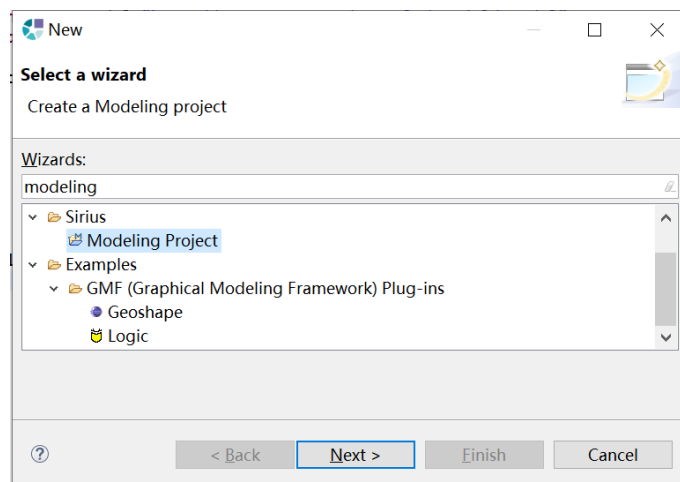
- g) 验证参数名是否为空并报错"变量名不可为空: Parameter Name cannot be empty!"

```
@Check
def checkParameterNameIsNotEmpty(Parameter pa) {
  if(pa.getName().isEmpty()) {
    error("变量名不可为空: Parameter Name cannot be empty!",
      XModelPackage.Literals.PARAMETER_NAME,
      PARAMETER_NAME_EMPTY);
  }
}
```

4. 选择模型所在的项目目录，右键>>Run as>>Eclipse Application，进行测试：

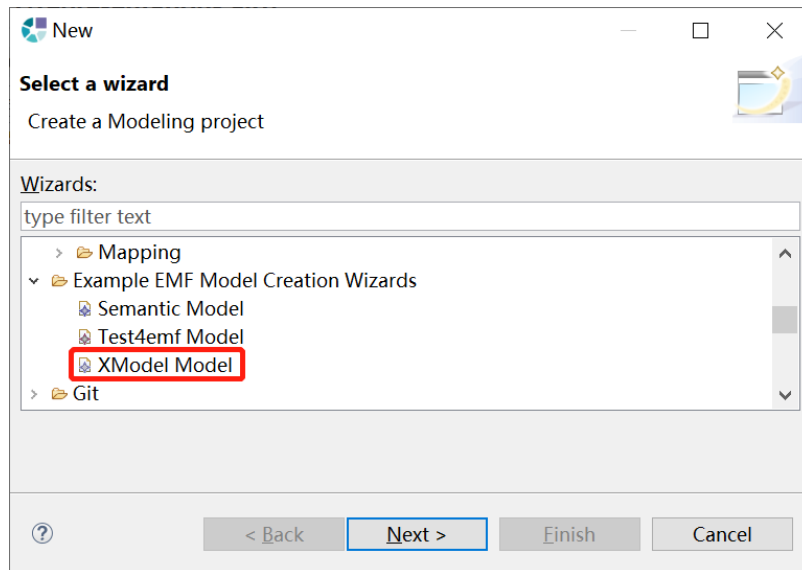


- 1) 新建一个建模工程或者 Java 工程，File>>new>>other>>Modeling Project/Java Project

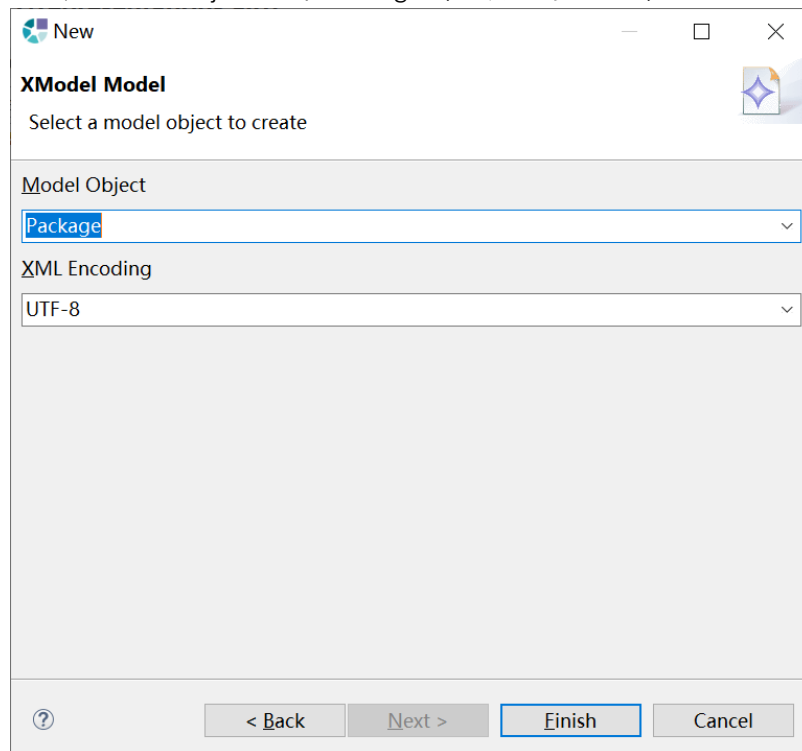




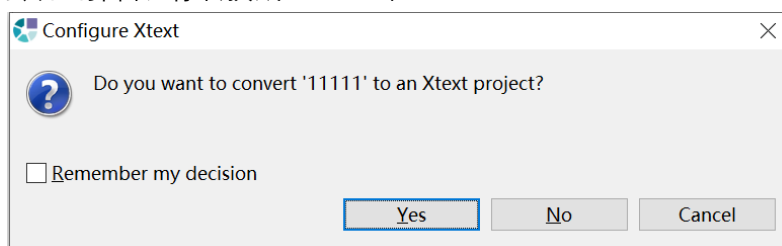
- 2) 选中生成的项目目录，右键>>New>>other>>XModel Model（自定义的 ecore 模型）



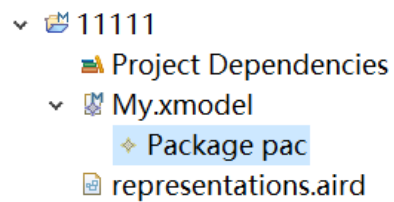
点击 Next，Model Object 选择 Package（最顶层的 EClass）



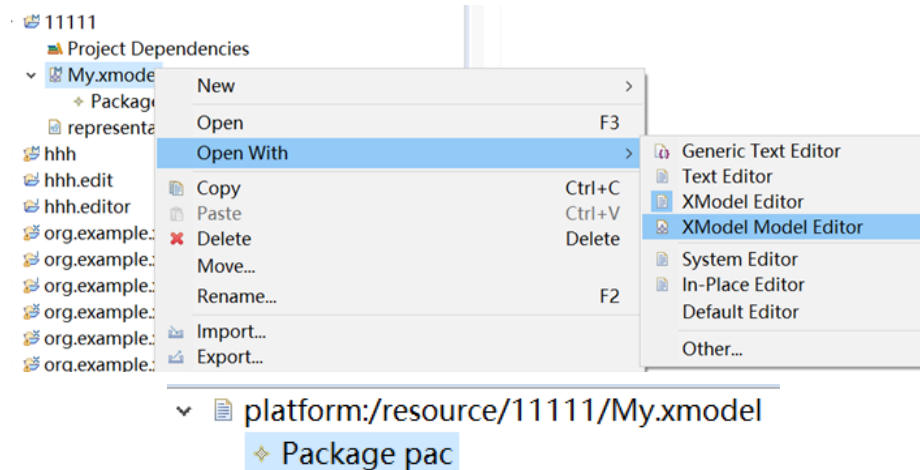
完成后会跳出弹窗让你转换成 Xtext 工程



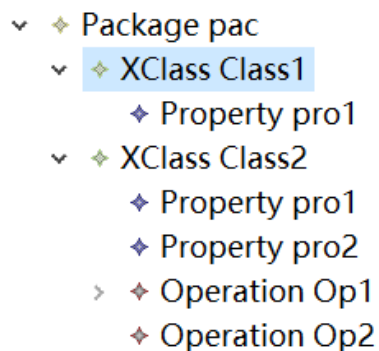
选择 Yes, 则可以转换成 Xtext 工程, 出现.xmodel 文件 (自定义 ecore 模型)



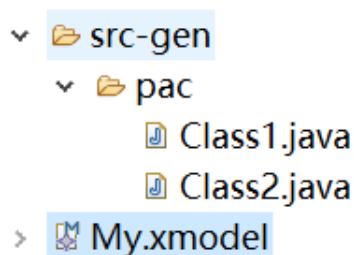
右键>>Open With>>XModel Model Editor, 以 XModel 模型编辑器方式打开



在 xmodel 文件的 package 中添加类、以及一些变量、方法、函数参数测试



则在目录中会自动生成 src-gen 目录, 并在其中生成以类名为名的 java 文件



```

package pac;

public class Class1 {

    public String pro1;

    public String getPro1() {
        return pro1;
    }

    public void setPro1(String pro1) {
        this.pro1 = pro1;
    }

}

package pac;

public class Class2 {

    public String pro1;

    public String getPro1() {
        return pro1;
    }

    public void setPro1(String pro1) {
        this.pro1 = pro1;
    }

    public String pro2;

    public String getPro2() {
        return pro2;
    }

    public void setPro2(String pro2) {
        this.pro2 = pro2;
    }

    public String Op1(String hhhh,String pa2,String pa3){
        // to do
    }
    public String Op2(){
        // to do
    }

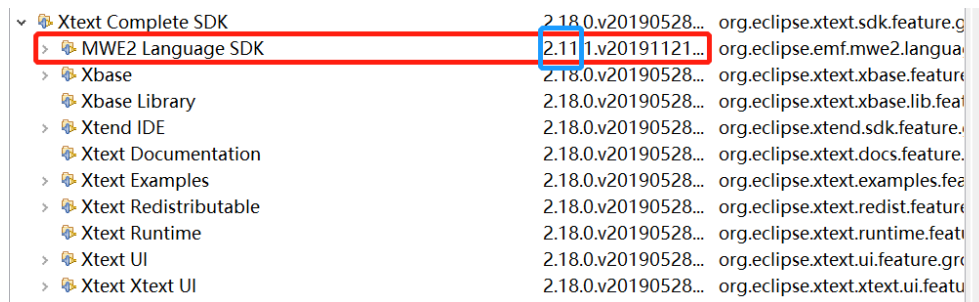
}

```

在测试过程中，有可能出现转换成 xtext 工程后没有生成 src-gen 目录和 java 文件的情况，可能原因是安装 Xtext 时，Xtext Complete SDK 中的 MWE2 Language SDK 版本为 2.11 以下 (2.10)

Xtext Complete SDK	2.18.0.v201905.
MWE2 Language SDK	2.10.0
MWE2 Launcher	2.10.0
MWE2 Launcher Developer Resources	2.10.0
MWE2 Runtime SDK	2.10.0
Xtext UI	2.18.0.v201905.
Xbase	2.18.0.v201905.
Xbase Library	2.18.0.v201905.
Xtend IDE	2.18.0.v201905.
Xtext Documentation	2.18.0.v201905.
Xtext Examples	2.18.0.v201905.
Xtext Redistributable	2.18.0.v201905.
Xtext Runtime	2.18.0.v201905.

正确版本应为 2.11 以上，主要还是需要与 Xtext Complete SDK 的版本适配



▼ Xtext Complete SDK	2.18.0.v20190528...	org.eclipse.xtext.sdk.feature.g
> MWE2 Language SDK	2.11.1.v20191121...	org.eclipse.emf.mwe2.langua
> Xbase	2.18.0.v20190528...	org.eclipse.xtext.xbase.feature
> Xbase Library	2.18.0.v20190528...	org.eclipse.xtext.xbase.lib.fe
> Xtend IDE	2.18.0.v20190528...	org.eclipse.xtend.sdk.feature.
> Xtext Documentation	2.18.0.v20190528...	org.eclipse.xtext.docs.feature.
> Xtext Examples	2.18.0.v20190528...	org.eclipse.xtext.examples.fe
> Xtext Redistributable	2.18.0.v20190528...	org.eclipse.xtext.redist.feature
> Xtext Runtime	2.18.0.v20190528...	org.eclipse.xtext.runtime.fe
> Xtext UI	2.18.0.v20190528...	org.eclipse.xtext.ui.feature.g
> Xtext Xtext UI	2.18.0.v20190528...	org.eclipse.xtext.xtext.ui.fe

在官网上可以查找到 Xtext Complete SDK 与 MWE2 Language SDK 的适配版本 ↓

<https://www.eclipse.org/Xtext/releasesnotes.html>

找到适配版本后可以在 install software 中粘贴 URL 选择适配版本并下载 ↓

<https://www.eclipse.org/xtend/download.html>