# Eclipse Modeling Framework (EMF) - Tutorial

Lars Vogel

Version 3.4

Copyright © 2007-2015 vogella GmbH

02.09.2015

**Eclipse EMF**

This tutorial describes the usage of Eclipse EMF, a framework for modeling your data model and creating Java code from it. This tutorial is based on Eclipse 4.5 (Mars).

## Table of Contents

1. Models and Eclipse EMF
- 1.1. Data model
- 1.2. Eclipse Modeling Framework (EMF)
- 1.3. Generate data from an EMF model
- 1.4. Meta Models - Ecore and Genmodel
- 1.5. Ecore description file
- 1.6. Advantages of using EMF
- 1.7. UML

2. Installation
3. Define EMF models
- 3.1. Project and initial model creation
- 3.2. View Ecore diagram
- 3.3. Set the package

4. Generating the domain classes
- 4.1. Generating Java code

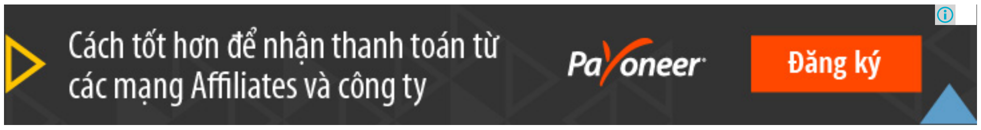
5. Create EMF Editor plug-ins
- 5.1. Generating edit / editor code
- 5.2. Run your plugins
- 5.3. Create your model
- 5.4. Edit your model

6. Using the model code
- 6.1. Overview
- 6.2. Example

7. Creating JavaDoc
8. Generating methods
9. Extending an EMF Ecore model (inheritance)
- 9.1. Overview
- 9.2. Example

**10. Setting the empty string as default value**
**11. Create EMF Generator Model**
**12. Next steps**
**13. About this website**
**14. Links and Literature**

**14.1. EMF Resources**
**14.2. vogella GmbH training and consulting support**


Cách tốt hơn để nhận thanh toán từ các mạng Affiliates và công ty
Payoneer
Đăng ký

# 1. Models and Eclipse EMF

## 1.1. Data model

A *data model*, sometimes also called *domain model*, represents the data you want to work with. For example, if you develop an online flight booking application, you might model your domain model with objects like `Person`, `Flight`, `Booking` etc.

A good practice is to model the data model of an application independently of the application logic or user interface. This approach leads to classes with almost no logic and a lot of properties, e.g., `Person` would have the properties *firstName*, *lastName*, *Address*, etc.

## 1.2. Eclipse Modeling Framework (EMF)

The *Eclipse Modeling Framework* (EMF) is a set of plug-ins which can be used to model a data model and to generated code or other output based on this model. EMF has a distinction between the meta-model and the actual model. The meta-model describes the structure of the model. A model is a concrete instance of this meta-model.

EMF allows the developer to create the meta-model via different means, e.g., XMI, Java annotations, UML or an XML scheme. It also allows to persists the model data; the default implementation uses a data format called *XML Metadata Interchange*.

## 1.3. Generate data from an EMF model

The information stored in the EMF models can be used to generate derived output.

A typical use case for EMF is that you specify a meta-data which represents the domain model of your application and that you use EMF functionality to generate corresponding Java implementation classes from this model.

The EMF framework supports that the generated code can be safely extended by hand.

Alternatively, the EMF model (which holds real data based on the model structure) can also be used to generate output, or it can be interpreted at runtime within an application.

**Note:** This output generation is not limited to Java classes. For example, the vogella.com webpage uses an EMF model to define the logical structure of the webpage and a hand-written Java program uses this model to generate HTML pages from it, using the **FreeMarker** template engine.

## 1.4. Meta Models - Ecore and Genmodel

As discussed earlier, EMF has a meta-model. This meta-model consists of two parts; the *ecore* and the *genmodel* description files.

The *ecore* file contains the information about the defined classes. The *genmodel* file contains additional information for the code generation, e.g., the path and file information. The *genmodel* file also contains the control parameter how the code should be generated.

## 1.5. Ecore description file

The *ecore* file allows to define the following elements.

- `EClass` : represents a class, with zero or more attributes and zero or more references.

- `EAttribute` : represents an attribute which has a name and a type.
- `EReference` : represents one end of an association between two classes. It has flags to indicate if it represents a containment and a reference class to which it points.
- `EDataType` : represents the type of an attribute, e.g., `int`, `float` or `java.util.Date`

The *Ecore* model shows a root object representing the whole model. This model has children which represent the packages, whose children represent the classes, while the children of the classes represent the attributes of these classes.

### 1.6. Advantages of using EMF

With EMF you define your domain model explicitly. This helps to provide clear visibility of the model. The code generator for EMF models can be adjusted and in its default setting. It provides change notification functionality to the model in case of model changes. EMF generates interfaces and a factory to create your objects; therefore, it helps you to keep your application clean from the individual implementation classes.

Another advantage is that you can regenerate the Java code from the model at any point in time.

### 1.7. UML

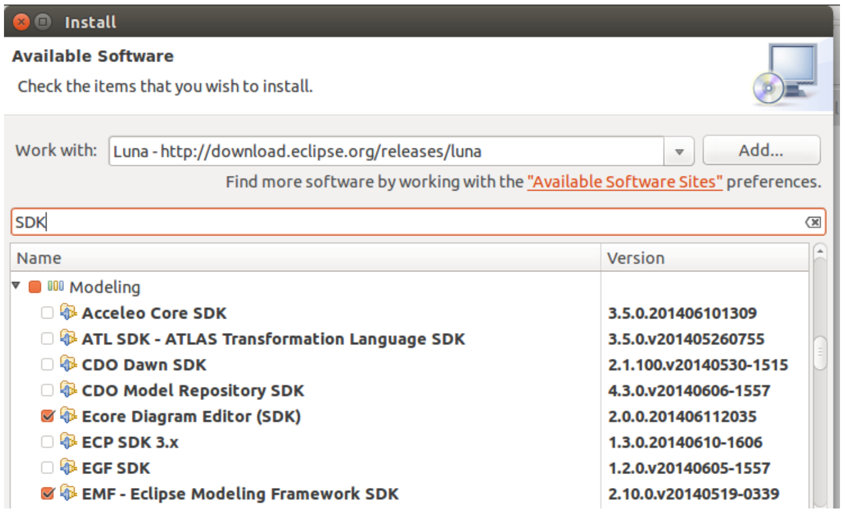
The EMF tooling allows you to create UML diagrams. The Unified Modeling Language (UML) is a visual language for capturing software designs and patterns. The first version of UML was defined 1994 and released by the Object Management Group (OMG) in 1997 as UML v.1.1. The syntax and a semantic of UML is defined by the OMG.

The basic building block for UML is a diagram. UML divides diagrams into ***structural diagrams*** and ***behavioral diagrams***

The latest version UML 2 has the target to add the ability for modelers to capture more system behavior. UML 2 has the target to support model driving architectures (MDA). MDA has the target to create automatically a software program from several models.

# 2. Installation

Install *EMF* via the *Eclipse Update manager* from *Help → Install New Software...*. Select *Modeling* and install *EMF - Eclipse Modeling Framework SDK* and the *Diagram Editor for Ecore (SDK)*. The second entry allows you to create models based on diagrams.


Install
Available Software
Check the items that you wish to install.
Work with:
Add...
Find more software by working with the "Available Software Sites" preferences.

Luna - http://download.eclipse.org/releases/luna

SDK

| Name | Version |
|---|---|
| Modeling | |
| ☐ Acceleo Core SDK | 3.5.0.201406101309 |
| ☐ ATL SDK - ATLAS Transformation Language SDK | 3.5.0.v201405260755 |
| ☐ CDO Dawn SDK | 2.1.100.v20140530-1515 |
| ☐ CDO Model Repository SDK | 4.3.0.v20140606-1557 |
| ☑ Ecore Diagram Editor (SDK) | 2.0.0.201406112035 |
| ☐ ECP SDK 3.x | 1.3.0.20140610-1606 |
| ☐ EGF SDK | 1.2.0.v20140605-1557 |
| ☑ EMF - Eclipse Modeling Framework SDK | 2.10.0.v20140519-0339 |

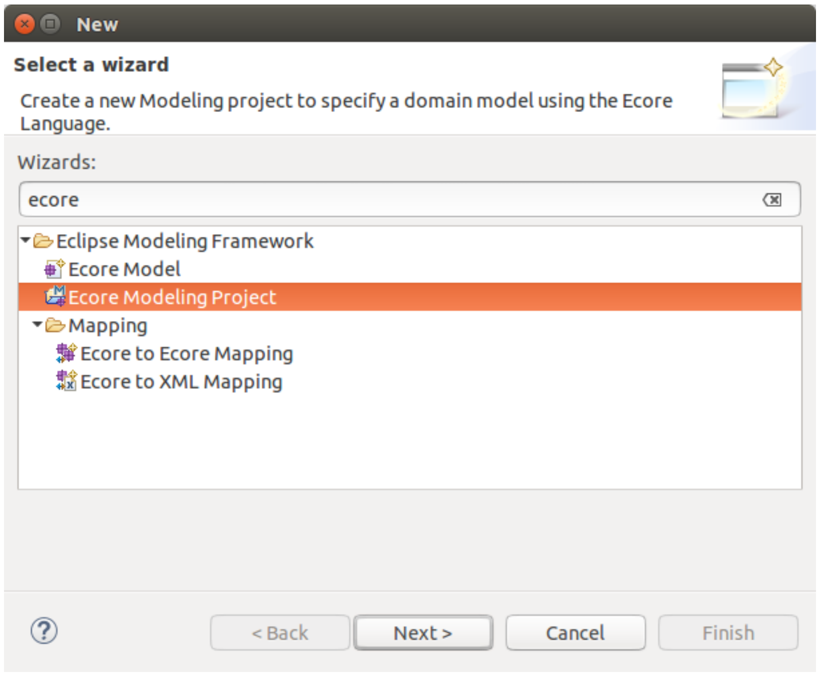
Restart your Eclipse IDE after the installation.

# 3. Define EMF models

### 3.1. Project and initial model creation

Create a new project called *com.vogella.emf.webpage.model* via *File → New → Project... → Ecore*

*Modeling Project*.


New

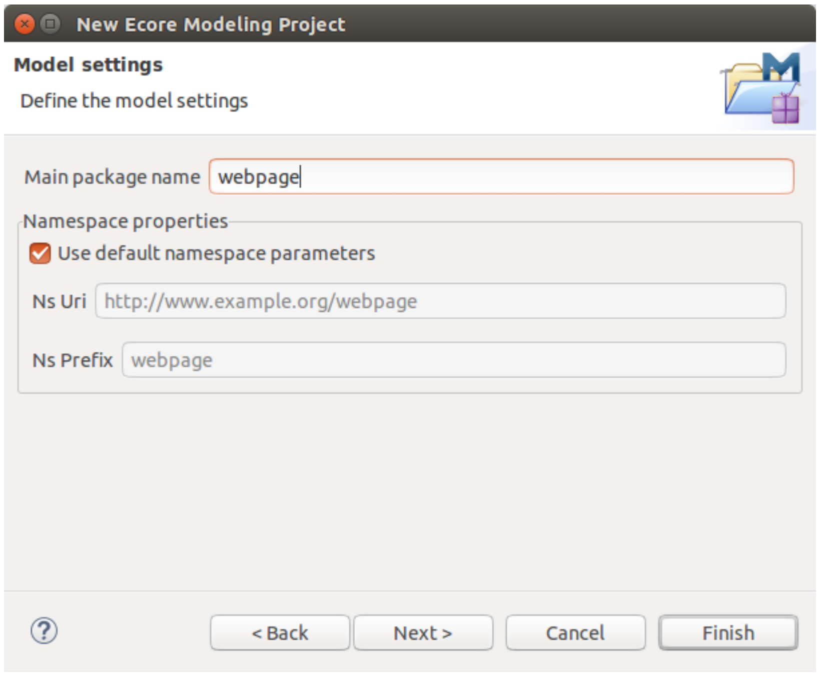
**Select a wizard**

Create a new Modeling project to specify a domain model using the Ecore Language.

Wizards: ecore

- Eclipse Modeling Framework
  - Ecore Model
  - Ecore Modeling Project
  - Mapping
    - Ecore to Ecore Mapping
    - Ecore to XML Mapping

Enter *`webpage.ecore`* as the *`Domain File Name`* parameter.


New Ecore Modeling Project

**Model settings**

Define the model settings

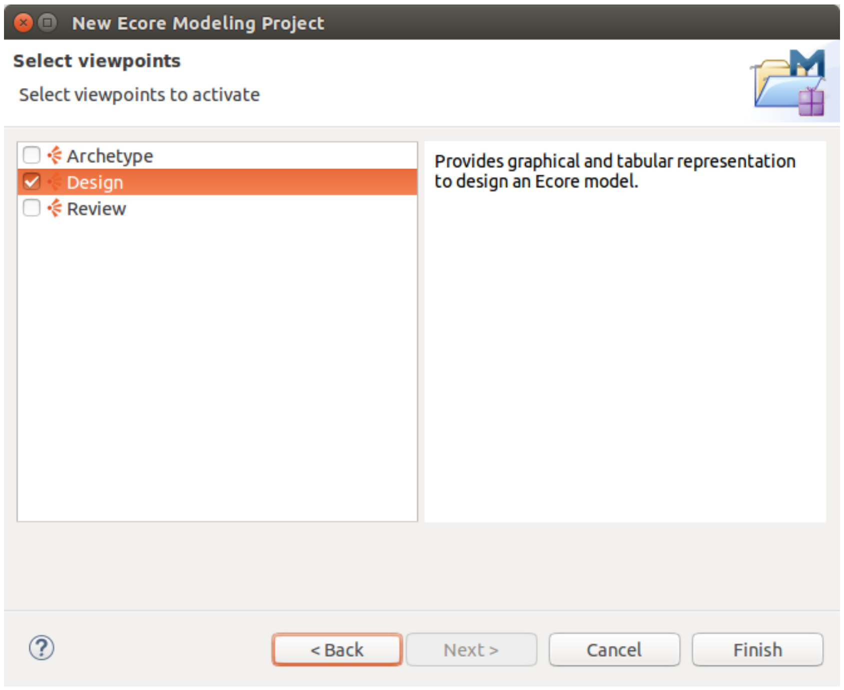
Main package name: webpage

Namespace properties

[✓] Use default namespace parameters

Ns Uri: http://www.example.org/webpage

Ns Prefix: webpage



New Ecore Modeling Project

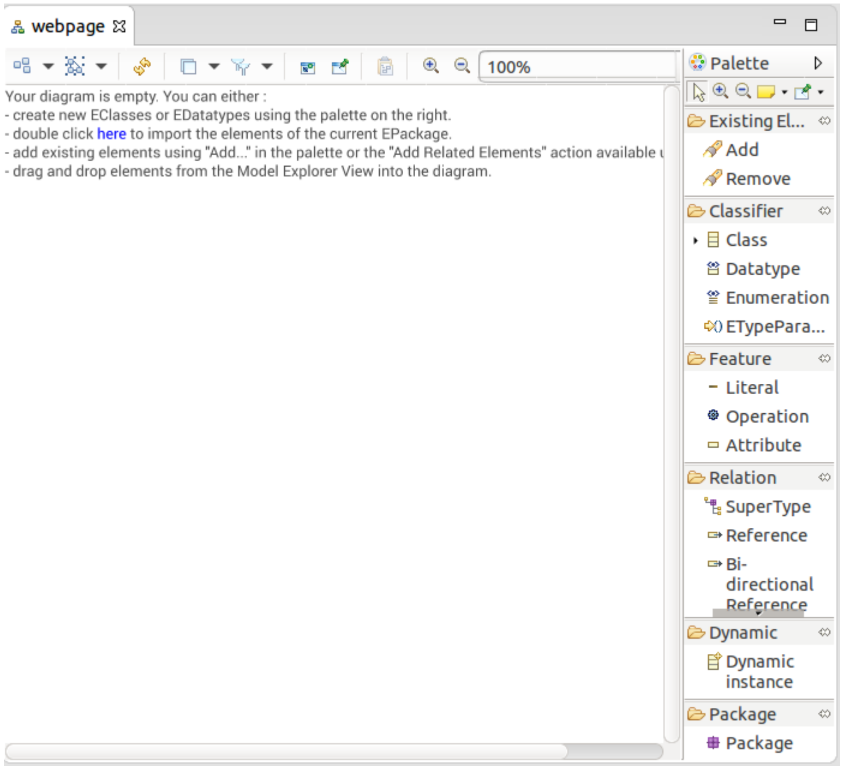
Select viewpoints

Select viewpoints to activate

- [ ] Archetype
- [x] Design
- [ ] Review

Provides graphical and tabular representation to design an Ecore model.

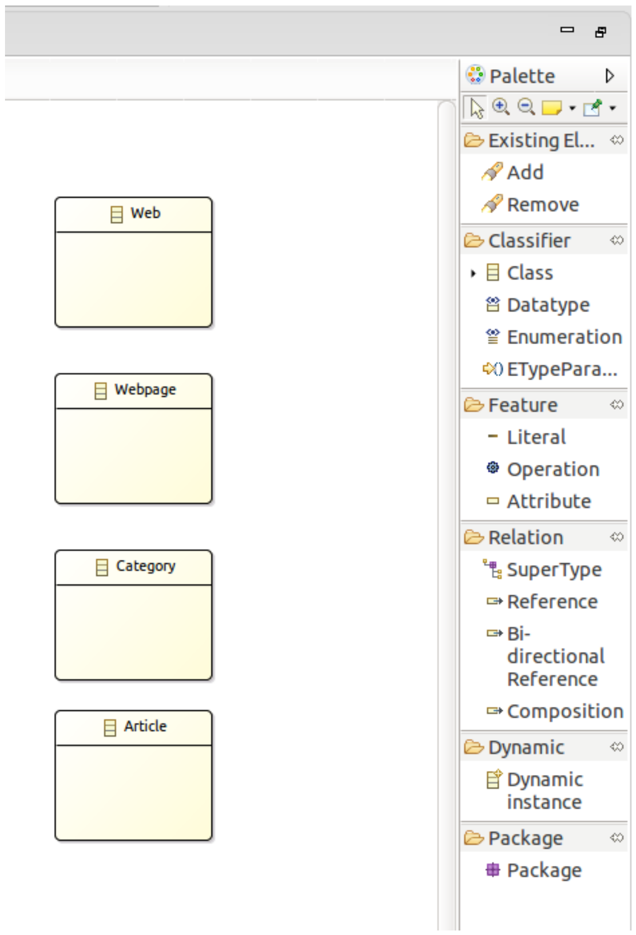
This should open a visual editor for creating EMF models.



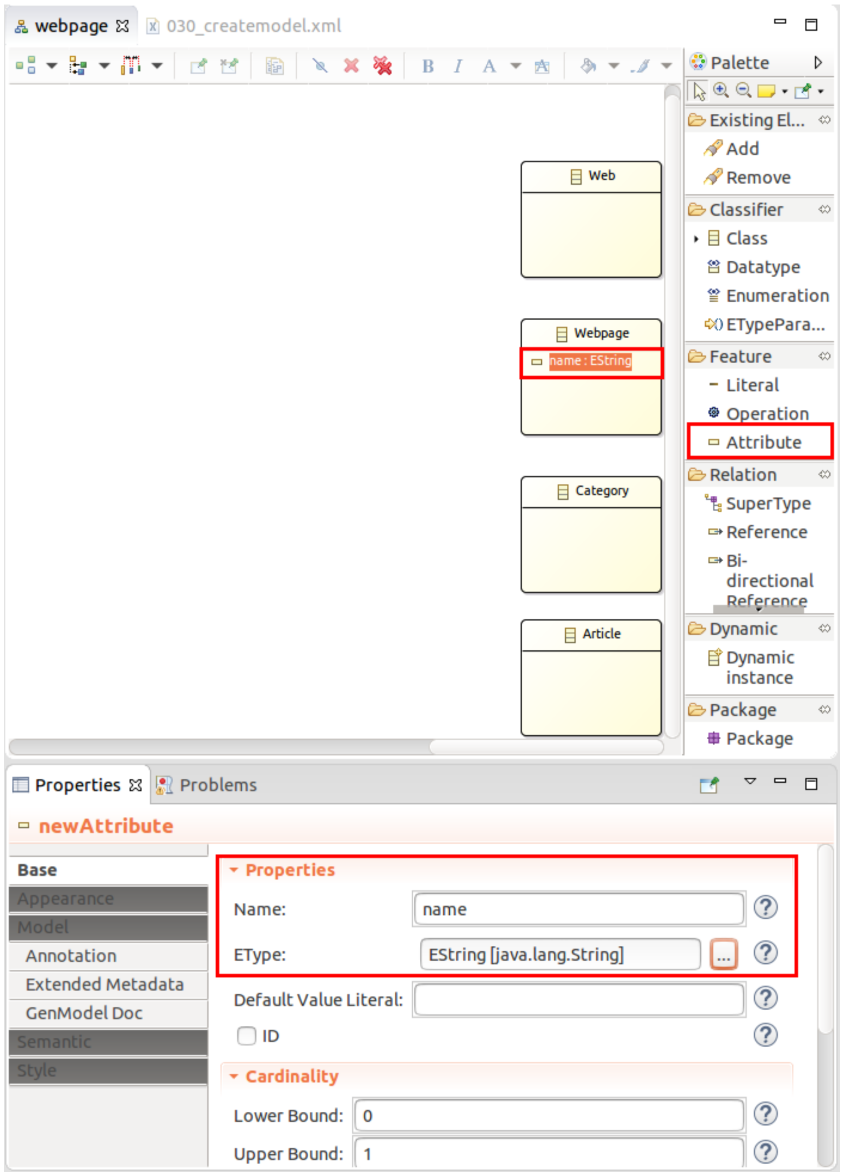
Your diagram is empty. You can either :
- create new EClasses or EDatatypes using the palette on the right.
- double click here to import the elements of the current EPackage.
- add existing elements using "Add..." in the palette or the "Add Related Elements" action available
- drag and drop elements from the Model Explorer View into the diagram.

Open the *Properties* view via the menu *Window → Show View → Other... → Properties*. This view allows you to modify the attributes of your model elements.

Click on *Class* and click into the editor to create a new class. Create the `MyWeb`, `Webpage`, `Category` and `Article` EClasses.



Use the *Attribute* node to assign the attribute called *name* to each object . This attribute should have the `EString` type.



newAttribute

Properties

Name: name

EType: EString [java.lang.String]

Default Value Literal:

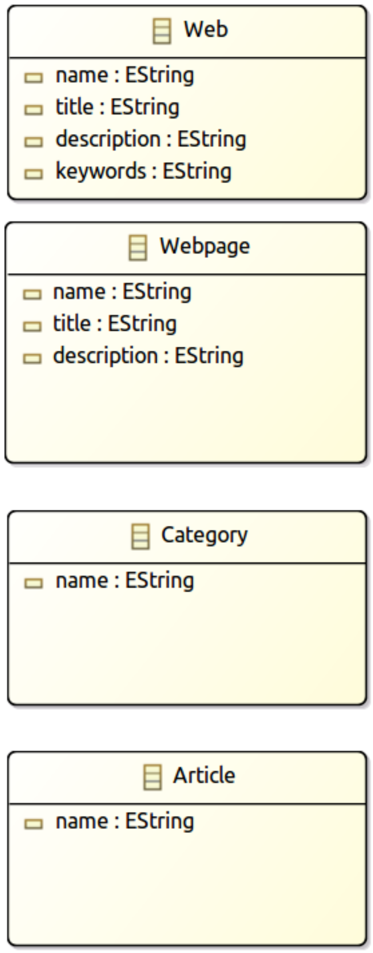
ID

Cardinality

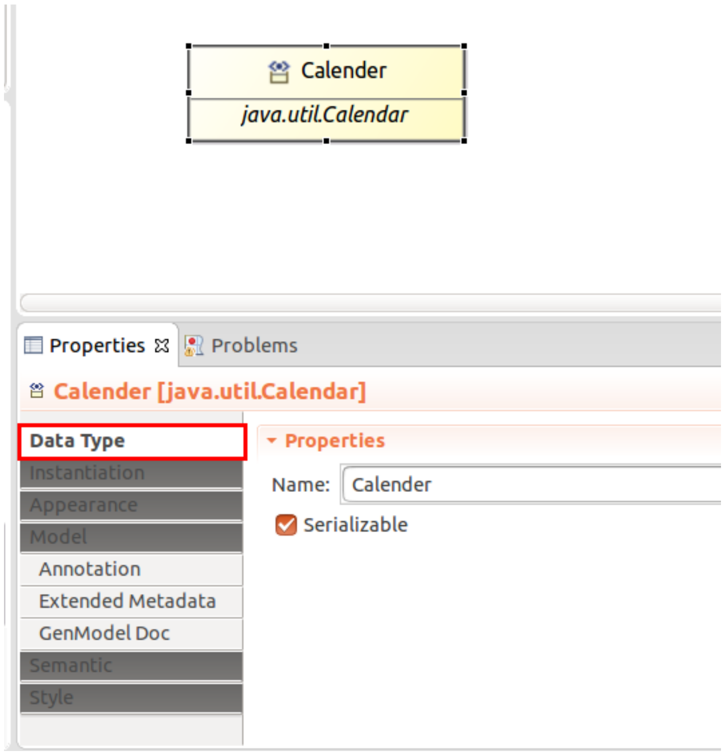
Lower Bound: 0

Upper Bound: 1

Add the `title`, `description` and `keywords` attributes to the *Web* and *Webpage* model elements.



We want to use the data type calendar in our model. Select `Datatype` and drag it into your model. Assign the name *Calendar* to it. Use `java.util.Calendar` as type parameter. Add a new Attribute called *created* to `Article` and use your new type.

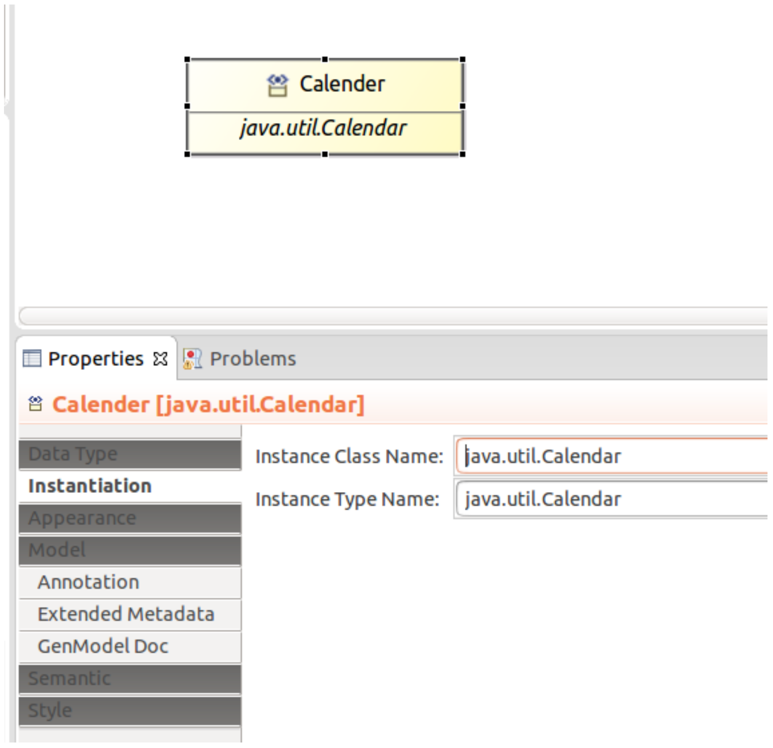


Calender [java.util.Calendar]

Properties

Name: Calender

[✓] Serializable

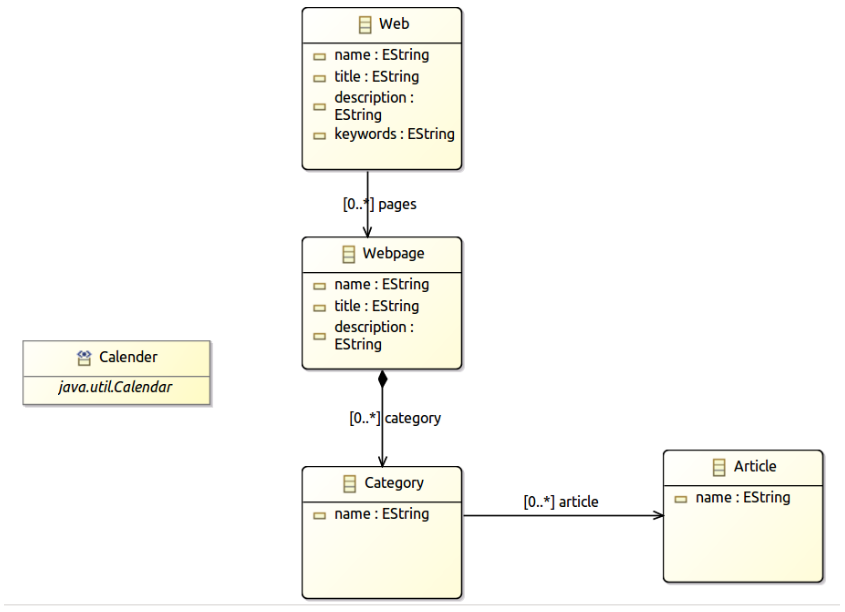


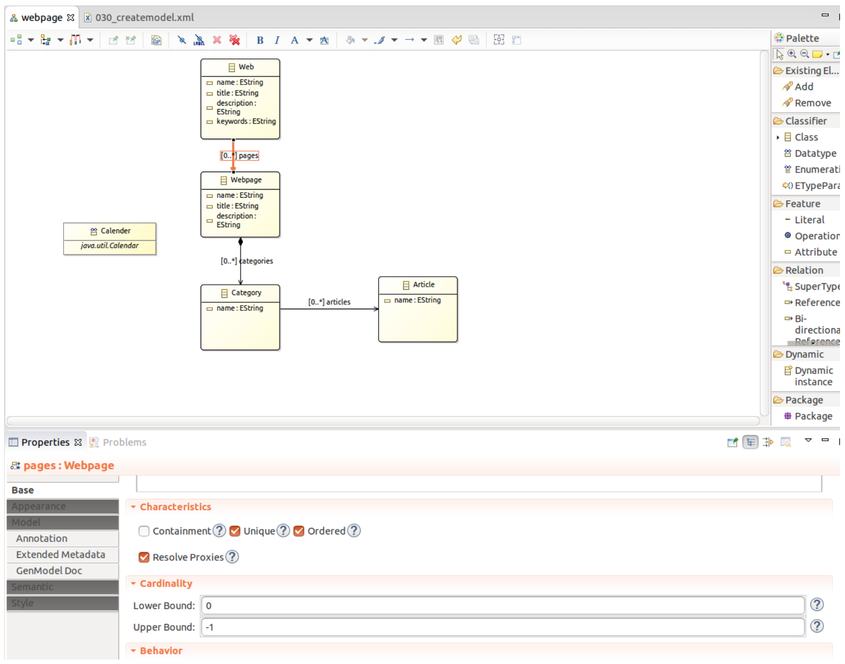
Calender [java.util.Calendar]

Instance Class Name: java.util.Calendar

Instance Type Name: java.util.Calendar

Select *References* and create an arrow similar to the following picture. Make sure the upper bound is set to "-1" (* on the user interface) and that the *Containment* property is flagged.





pages : Webpage

Characteristics

Containment [ ] Unique [✓] Ordered [✓]

Resolve Proxies [✓]

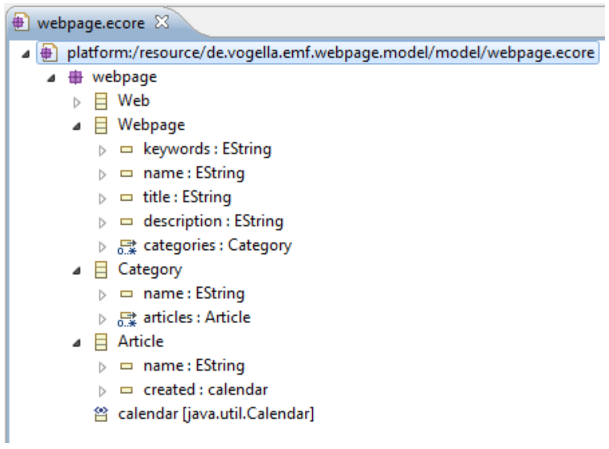
Cardinality

| | |
|---|---|
| Lower Bound: | 0 |
| Upper Bound: | -1 |

Behavior

## 3.2. View Ecore diagram

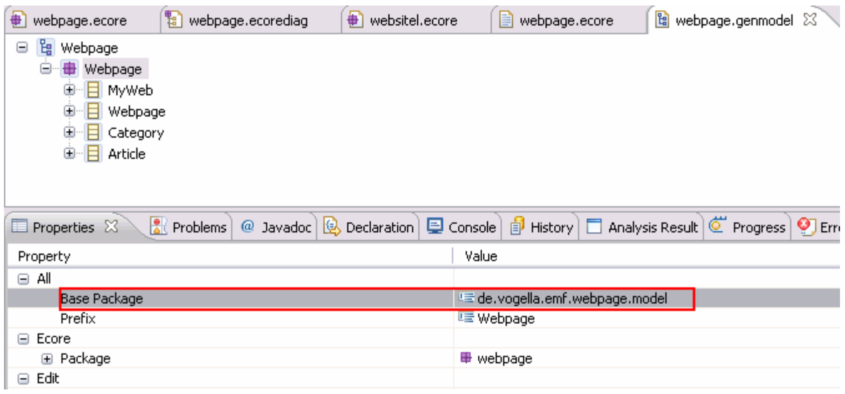
Close the diagram and open the *webpage.ecore* file. The result should look like the following screenshot.



```
webpage.ecore
platform:/resource/de.vogella.emf.webpage.model/model/webpage.ecore
  webpage
    Web
    Webpage
      keywords : EString
      name : EString
      title : EString
      description : EString
      categories : Category
    Category
      name : EString
      articles : Article
    Article
      name : EString
      created : calendar
    calendar [java.util.Calendar]
```

## 3.3. Set the package

Open the *webpage.genmodel* and select the *Webpage* node. Set the *base package* property to *de.vogella.emf.webpage.model*.



```
Webpage
  Webpage
    MyWeb
    Webpage
    Category
    Article
```

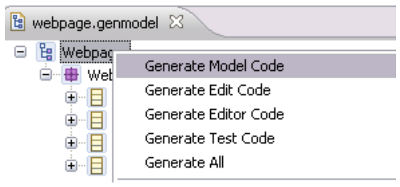
| Property | Value |
|---|---|
| All | |
| Base Package | de.vogella.emf.webpage.model |
| Prefix | Webpage |
| Ecore | |
| Package | webpage |
| Edit | |

# 4. Generating the domain classes

## 4.1. Generating Java code

Based on the *.genmodel* files, you can generate Java code.

Right-click on the root node of the *.genmodel* file and select *Generate Model Code*. This creates the Java implementation of the EMF model in the current project.



## 4.2. Review the generated code

The generated code will consist of the following:

- *model* -- Interfaces and the Factory to create the Java classes
- *model.impl* -- Concrete implementation of the interfaces defined in model
- *model.util* -- The AdapterFactory

The central factory has methods for creating all defined objects via `createObjectName()` methods.

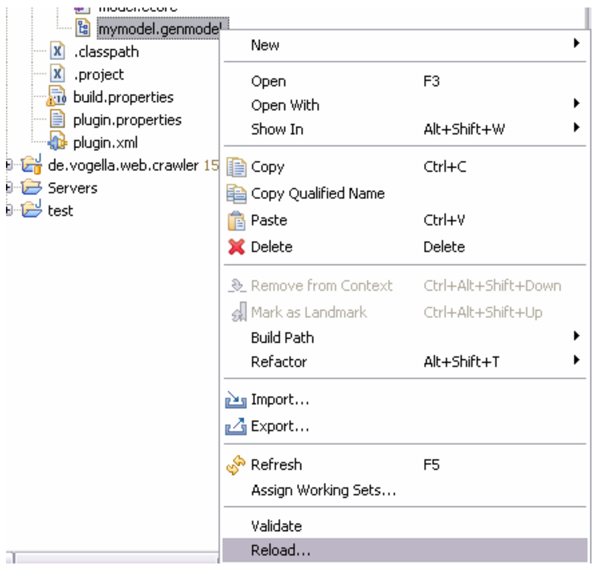
For each attribute the generated interface and its implementation contain `getter` and (if allowed in the model definition) `setter` methods. Each setter also has a generated notification to observers of the model. This means that other objects can attach them to the model and react to changes in the model.

Each generated interface extends the `EObject` interface. `EObject` is the base of every EMF class and is the EMF equivalent of `java.lang.Object`. `EObject` and its corresponding implementation class `EObjectImpl` provide a lightweight base class that lets the generated interfaces and classes participate in the EMF notification and persistence frameworks.

Every generated method is tagged with `@generated`. If you want to manually adjust the method and have EMF overwrite the method during the next generation run, you need to remove this tag.

## 4.3. Updating the model

If you change your *.ecore* model then you can update the *.genmodel* simply by reloading.



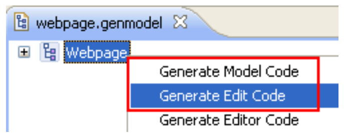
# 5. Create EMF Editor plug-ins

EMF can generate plug-ins which provide wizards for creating new model instances and an editor which allows you to enter your model information.

The following assumes that you have already have knowledge in developing Eclipse plug-ins. For more information about Eclipse plug-in development please see **Eclipse Plugin Tutorial**

## 5.1. Generating edit / editor code

Eclipse EMF allow you to create a editor for your model. Select your `.genmodel` file , right-click on it and select *Generate Edit Code* and afterwards *Generate Editor Code* .



Two **Eclipse plugin** projects have been created, "de.vogella.emf.webpage.model.edit" and "de.vogella.emf.webpage.model.editor".

## 5.2. Run your plugins

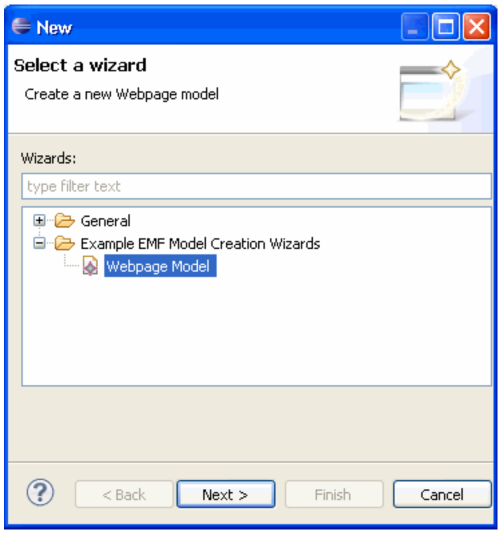
Select the `*.editor` project and start a new Eclipse instance with your new plug-in via right mouse-click on it and by selecting *Run-As → Eclipse application* .

This should start a new Eclipse runtime instance.

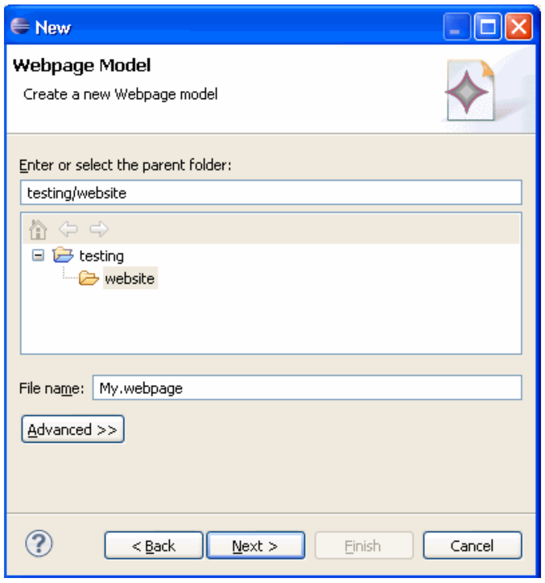
## 5.3. Create your model

In new Eclipse instance create a new project of type *General* called *testing* and a folder called `website`.

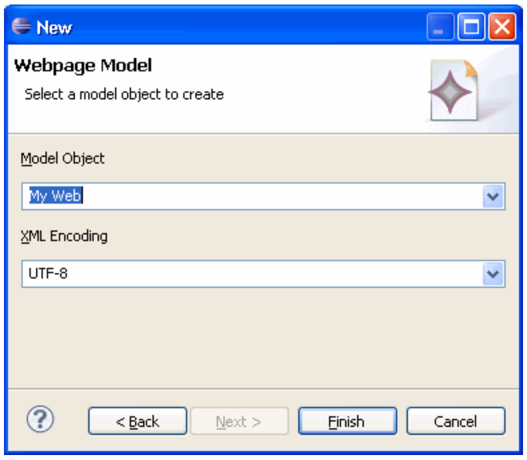
Select this folder, right click on it, select *New → Other... → Example EMF Model Creation Wizards → Webpage Model*.



Name your model *My.webpage*.

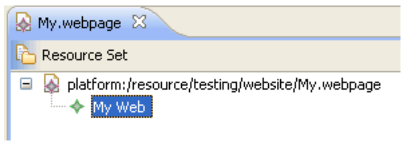


Select as the Model Object "My Web" and press finish.

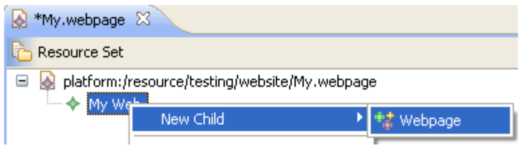


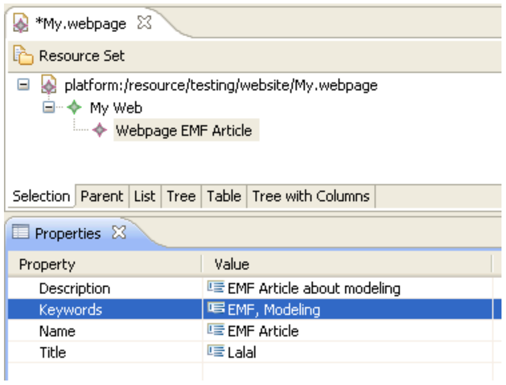
## 5.4. Edit your model

You should now see a editor for your website.model.



Right-click on "My Web" and create a new elements. To edit the elements use the "Properties View" which can be found under Window -> Show View -> Properties.





| Property | Value |
|---|---|
| Description | EMF Article about modeling |
| Keywords | EMF, Modeling |
| Name | EMF Article |
| Title | Lalal |

Save your created model.

# 6. Using the model code

## 6.1. Overview

The generated model code is standard Java code and can be used as such. The following demonstrates how you create objects based on the generated code.

## 6.2. Example

Create a new plug-in project called *de.vogella.emf.webpage.usingmodel*. Add the following dependency to your `MANIFEST.MF`.

- org.eclipse.emf.ecore
- com.vogella.emf.webpage.model

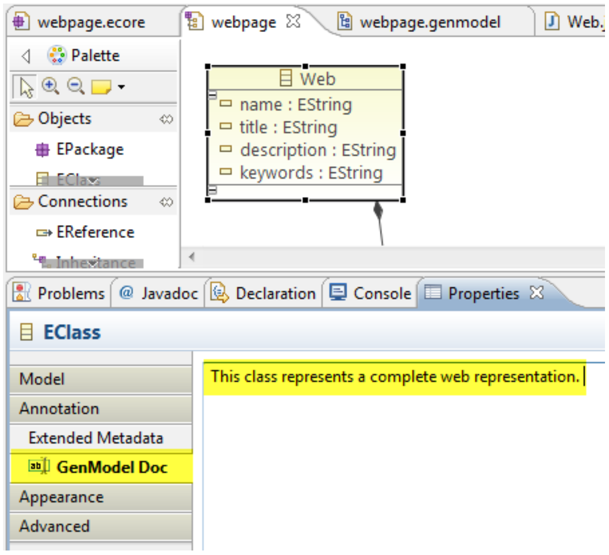
Create the following class.

```
package de.vogella.emf.webpage.usingmodel;

import de.vogella.emf.webpage.model.webpage.MyWeb;
import de.vogella.emf.webpage.model.webpage.Webpage;
import de.vogella.emf.webpage.model.webpage.WebpageFactory;
import de.vogella.emf.webpage.model.webpage.impl.WebpagePackageImpl;

public class UsingEMFModel {
  public static void main(String[] args) {
    WebpagePackage.eINSTANCE;
    // Retrieve the default factory singleton
    WebpageFactory factory = WebpageFactory.eINSTANCE;
    // create an instance of myWeb
    MyWeb myWeb = factory.createMyWeb();
    myWeb.setName("Hallo");
    myWeb.setDescription("This is a description");
    // create a page
    Webpage webpage = factory.createWebpage();
    webpage.setTitle("This is a title");
    // add the page to myWeb
    myWeb.getPages().add(webpage);
    // and so on, and so on
    // as you can see the EMF model can be (more or less) used as standard Java


  }
}
```

**Tip:** The `*PackageImpl.init()` method needs to be called before doing anything else as this method initializes the model and the listeners.

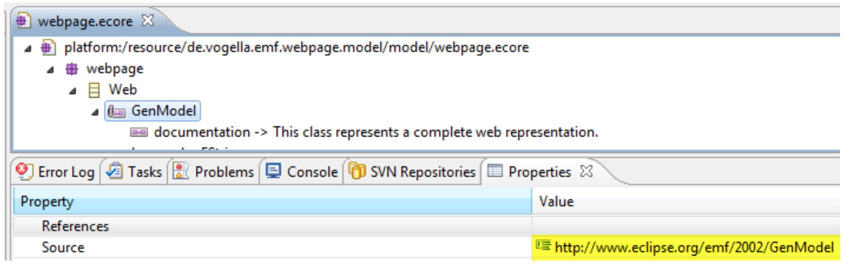
## 7. Creating JavaDoc

You can also generate Javadoc for your classes and methods. EMF uses annotations for this with a certain property key. The easiest way of adding this is again the diagram. Select a class and maintain the documentation in the "GenModel Doc".



This class represents a complete web representation.

The ecore model looks now like the following. The key in the annotation "http://www.eclipse.org/emf/2002/GenModel" is necessary and the key on the details enty must be "documentation".



platform:/resource/de.vogella.emf.webpage.model/model/webpage.ecore
webpage
Web
GenModel
documentation -> This class represents a complete web representation.

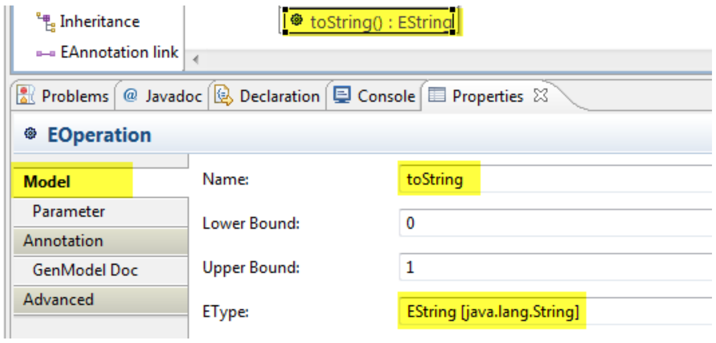
| Property | Value |
|---|---|
| References | |
| Source | http://www.eclipse.org/emf/2002/GenModel |

## 8. Generating methods

By default EMF generates getter and setter for every class. You can also add Operations or for example overwrite methods, e.g., the `toString()` method. For Article the following toString method was generated in "ArticleImpl

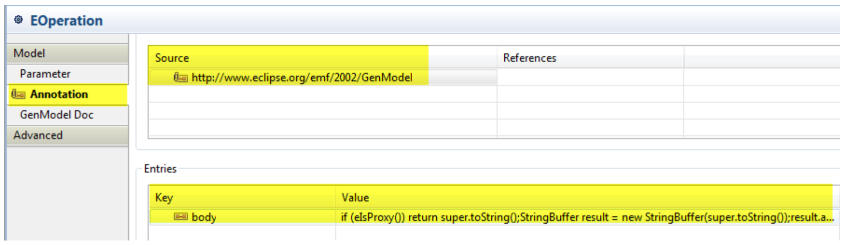
```
* @generated
   */
  @Override
  public String toString() {
    if (eIsProxy()) return super.toString();

    StringBuffer result = new StringBuffer(super.toString());
    result.append(" (name: ");
    result.append(name);
    result.append(", created: ");
    result.append(created);
    result.append(')');
    return result.toString();
  }
```

To overwrite this, add a "EOperation" to your model with the name toString. Maintain in the properties "EType" EString as return type.



Name: toString
Lower Bound: 0
Upper Bound: 1
EType: EString [java.lang.String]

Add an annotation with the source "http://www.eclipse.org/emf/2002/GenModel" and maintain an entry with the key "body", the value is the code that will be generated in to the method, you find it listed below.

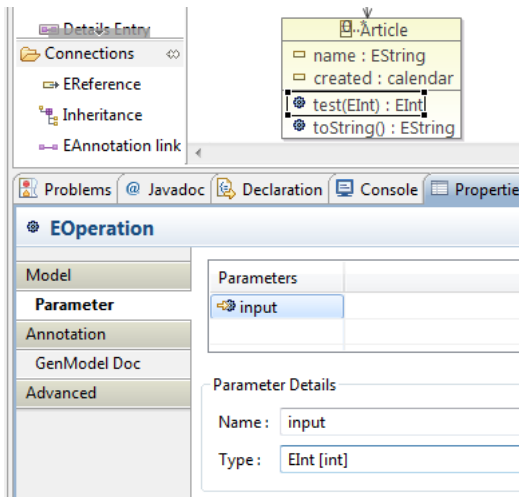


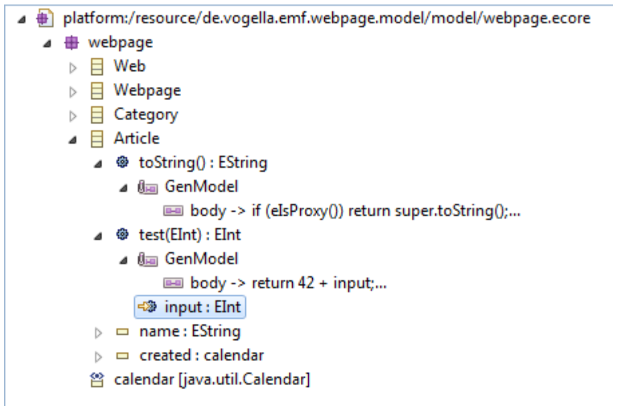
| Source | References |
|---|---|
| http://www.eclipse.org/emf/2002/GenModel | |

| Key | Value |
|---|---|
| body | if (eIsProxy()) return super.toString();StringBuffer result = new StringBuffer(super.toString());result.a... |

```
if (eIsProxy()) return super.toString();
    StringBuffer result = new StringBuffer(super.toString());
    result.append("Article: ");
    result.append(name);
    return result.toString();
```

You can also generate methods with input parameter, just add parameter with their type to your EOperation.




```
platform:/resource/de.vogella.emf.webpage.model/model/webpage.ecore
  webpage
    Web
    Webpage
    Category
    Article
      toString() : EString
        GenModel
          body -> if (eIsProxy()) return super.toString();...
      test(EInt) : EInt
        GenModel
          body -> return 42 + input;...
        input : EInt
      name : EString
      created : calendar
    calendar [java.util.Calendar]
```

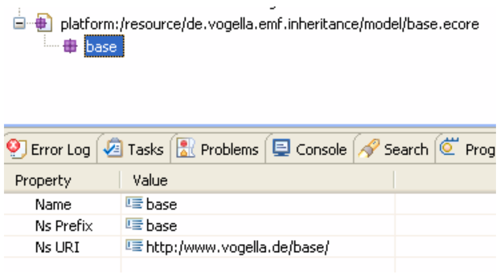
# 9. Extending an EMF Ecore model (inheritance)

## 9.1. Overview

EMF allows to extend existing models via inheritance. The following will define a base model and an extension based on this base model. This can for example be used to extend the **Eclipse e4** application model. It will also demonstrate how to work with EMF ecore models directly without using the ecore tools.

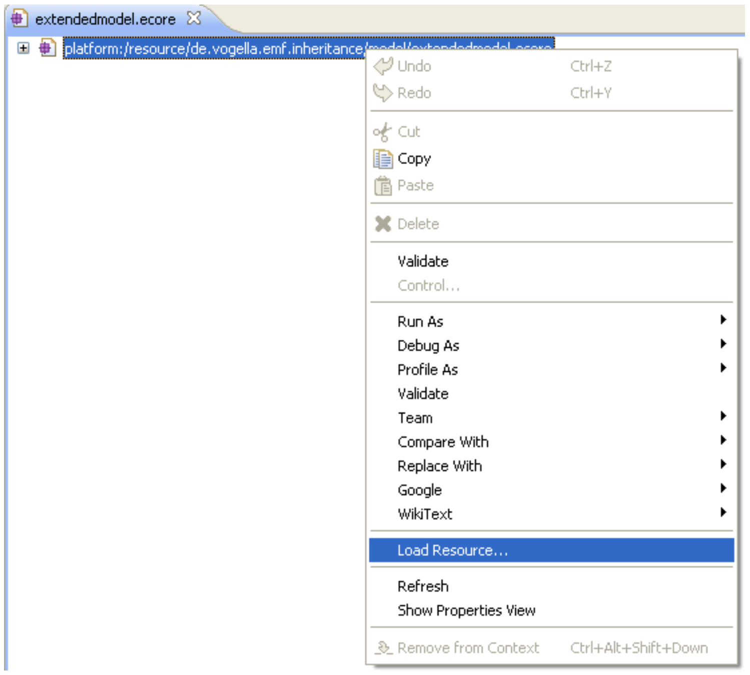
## 9.2. Example

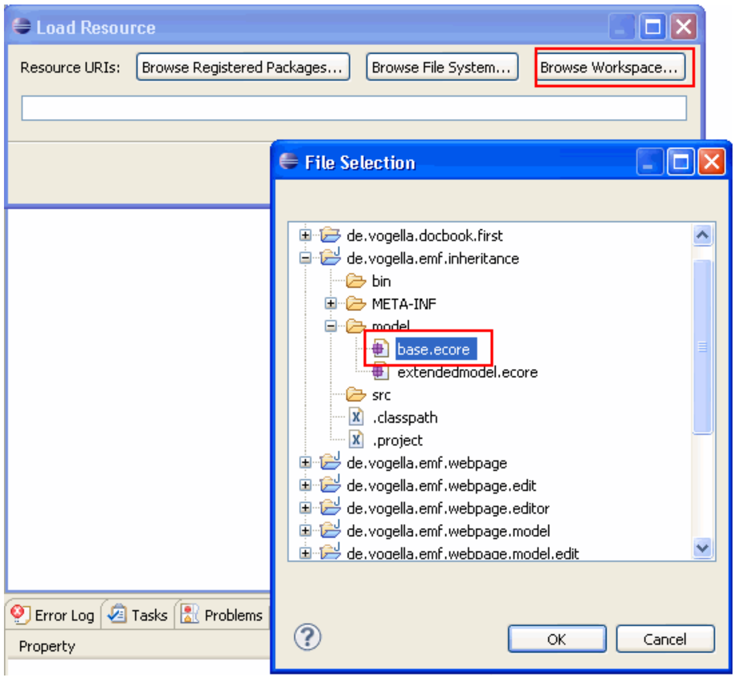
Create a new EMF project "de.vogella.emf.inheritance". Create a new model by selecting File -> New -> "Eclipse Modeling Framework" -> "Ecore Model". Name the model "base.ecore". Select "EPackage" as the basis and maintain the following properties for this package.

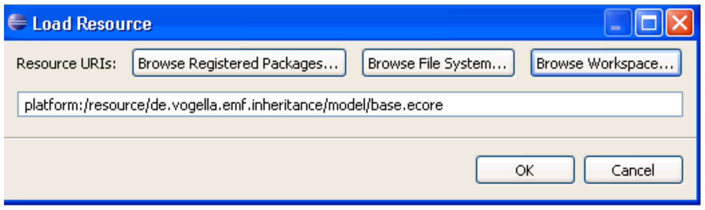

platform:/resource/de.vogella.emf.inheritance/model/base.ecore
base
Error Log
Tasks
Problems
Console
Search
Prog

| Property | Value |
|---|---|
| Name | base |
| Ns Prefix | base |
| Ns URI | http:/www.vogella.de/base/ |

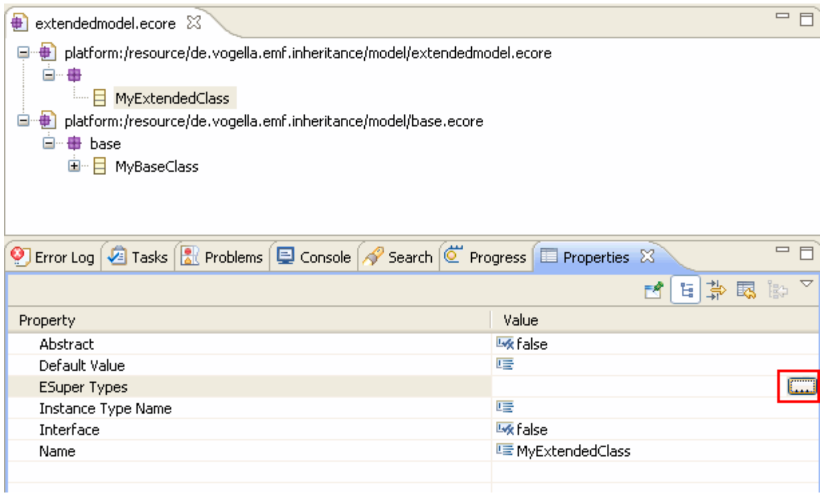
Right-click on the package and select New Child -> EClass. Maintain the class "MyBaseClass" with two "EAttributes" of type "EString". Create a new "Ecore" model "extendedmodel.ecore". Maintain "extended" as the package name. Right-click your model and select "Load resource".






Load Resource
Resource URIs:
Browse Registered Packages...
Browse File System...
Browse Workspace...
platform:/resource/de.vogella.emf.inheritance/model/base.ecore
OK
Cancel

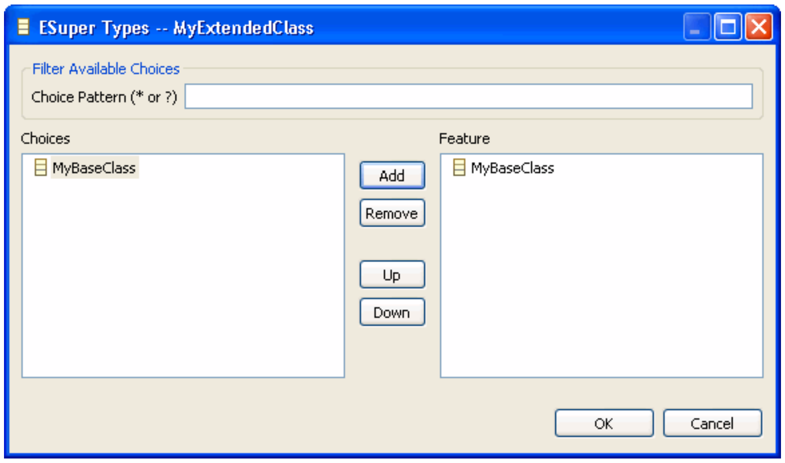
Create a new class "MyExtendedClass" and press "ESuperType".



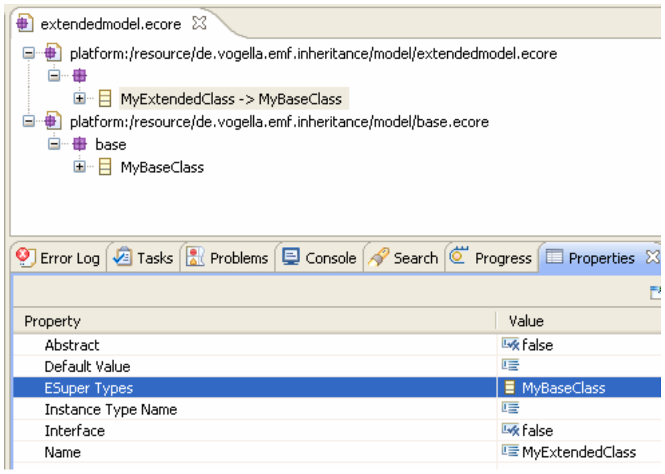
platform:/resource/de.vogella.emf.inheritance/model/extendedmodel.ecore
MyExtendedClass
platform:/resource/de.vogella.emf.inheritance/model/base.ecore
base
MyBaseClass

| Property | Value |
|---|---|
| Abstract | false |
| Default Value | |
| ESuper Types | |
| Instance Type Name | |
| Interface | false |
| Name | MyExtendedClass |

Add your "MyBaseClass".



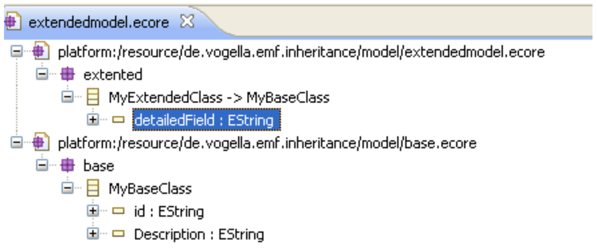
Choices: MyBaseClass
Feature: MyBaseClass



platform:/resource/de.vogella.emf.inheritance/model/extendedmodel.ecore
MyExtendedClass -> MyBaseClass
platform:/resource/de.vogella.emf.inheritance/model/base.ecore
base
MyBaseClass

| Property | Value |
|---|---|
| Abstract | false |
| Default Value | |
| ESuper Types | MyBaseClass |
| Instance Type Name | |
| Interface | false |
| Name | MyExtendedClass |

Maintain a new EAtribute "detailedField" on "MyExtendedClass".


```
extendedmodel.ecore
platform:/resource/de.vogella.emf.inheritance/model/extendedmodel.ecore
  extented
    MyExtendedClass -> MyBaseClass
      detailedField : EString
platform:/resource/de.vogella.emf.inheritance/model/base.ecore
  base
    MyBaseClass
      id : EString
      Description : EString
```

Create a new genmodel "extended.genmodel" based on extended.ecore. Generated Java code and you will see that the "MyExtendedClass" has extended "MyBaseClass".

## 10. Setting the empty string as default value

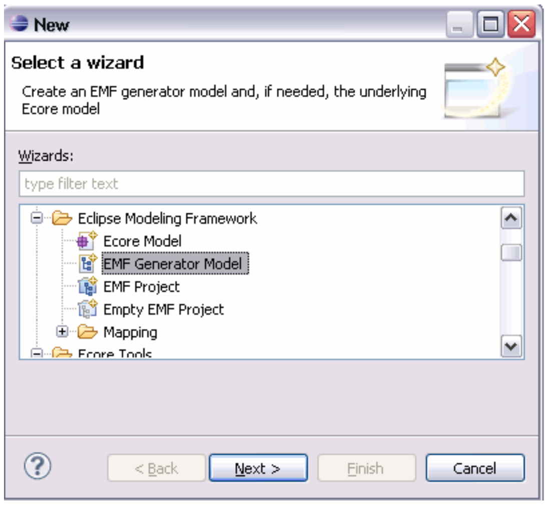
It is not obvious how to set an empty string as a default value for an EMF string attribute. To set an empty string as default value do the following.

- Select the Attribute
- Im the Property View click into the value field of "Default Value Literal"
- Do not enter something.

To remove this empty value again, click "Restore Default Value" in the toolbar.

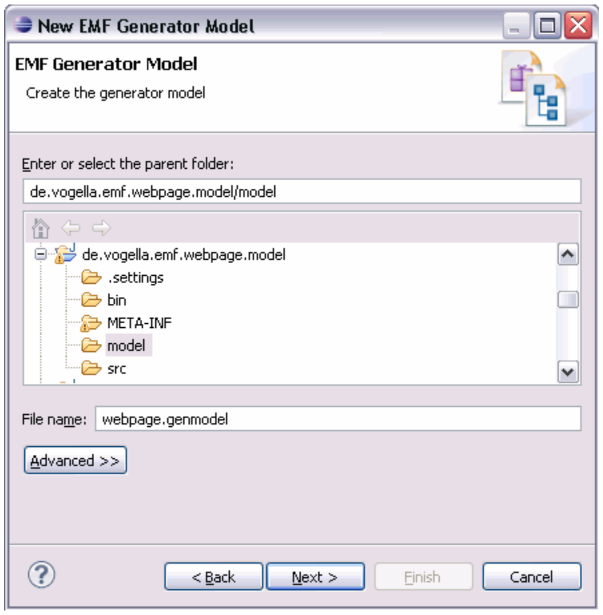
## 11. Create EMF Generator Model

If your EMF genmodel is missing you can create one. Right-click your `.ecore` file and select *File → New → Other... → EMF Generator model*. Create the `webpage.genmodel` file based on your *Ecore model*.


New

Select a wizard

Create an EMF generator model and, if needed, the underlying Ecore model



New EMF Generator Model

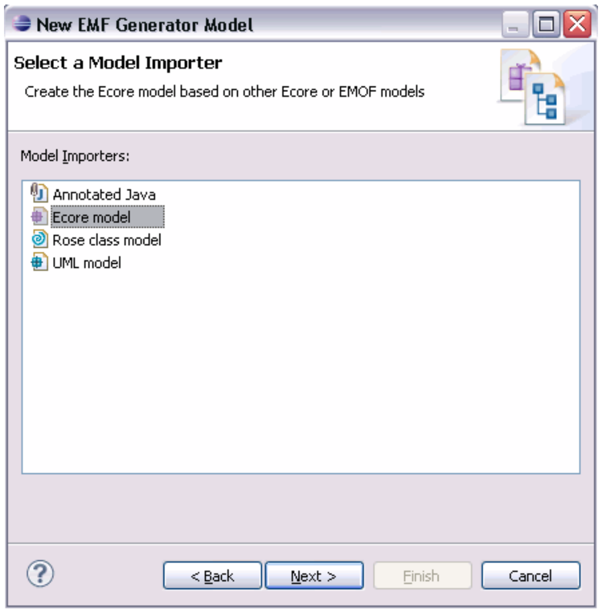
EMF Generator Model

Create the generator model

Enter or select the parent folder: de.vogella.emf.webpage.model/model

```
de.vogella.emf.webpage.model
    .settings
    bin
    META-INF
    model
    src
```

File name: webpage.genmodel



New EMF Generator Model

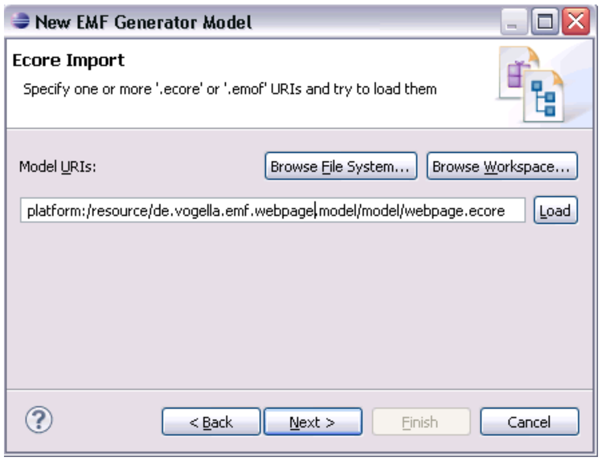
Select a Model Importer

Create the Ecore model based on other Ecore or EMOF models

Model Importers:

- Annotated Java
- Ecore model
- Rose class model
- UML model

Select your model and press load.

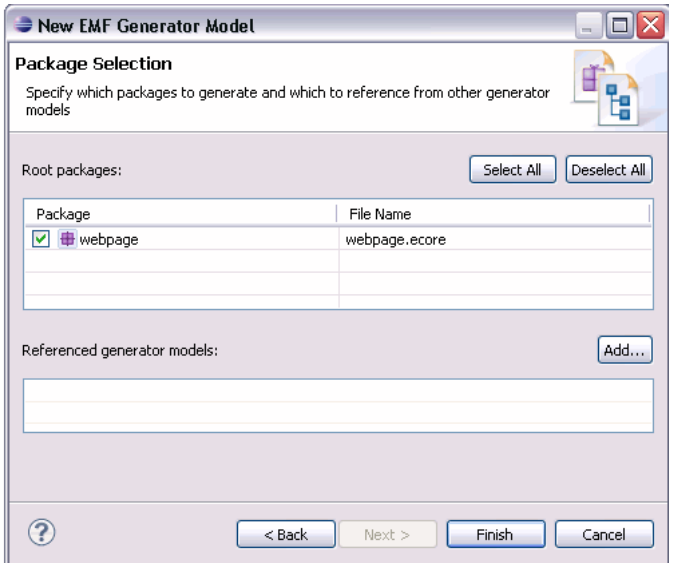


New EMF Generator Model

Ecore Import

Specify one or more '.ecore' or '.emof' URIs and try to load them

Model URIs: platform:/resource/de.vogella.emf.webpage.model/model/webpage.ecore



## 12. Next steps

Please check the appendix for pointers to more advanced Eclipse EMF topics.

## 13. About this website

## 14. Links and Literature

### 14.1. EMF Resources

Eclipse EMF Homepage

EMF Documentation

Ecore translation

Eclipse ATL - allows model to model transformation for EMF

### 14.2. vogella GmbH training and consulting support

TRAINING SERVICE & SUPPORT