Model Driven Engineering

Assignment 4 EMF

Submitted to

Juri Di Rocco

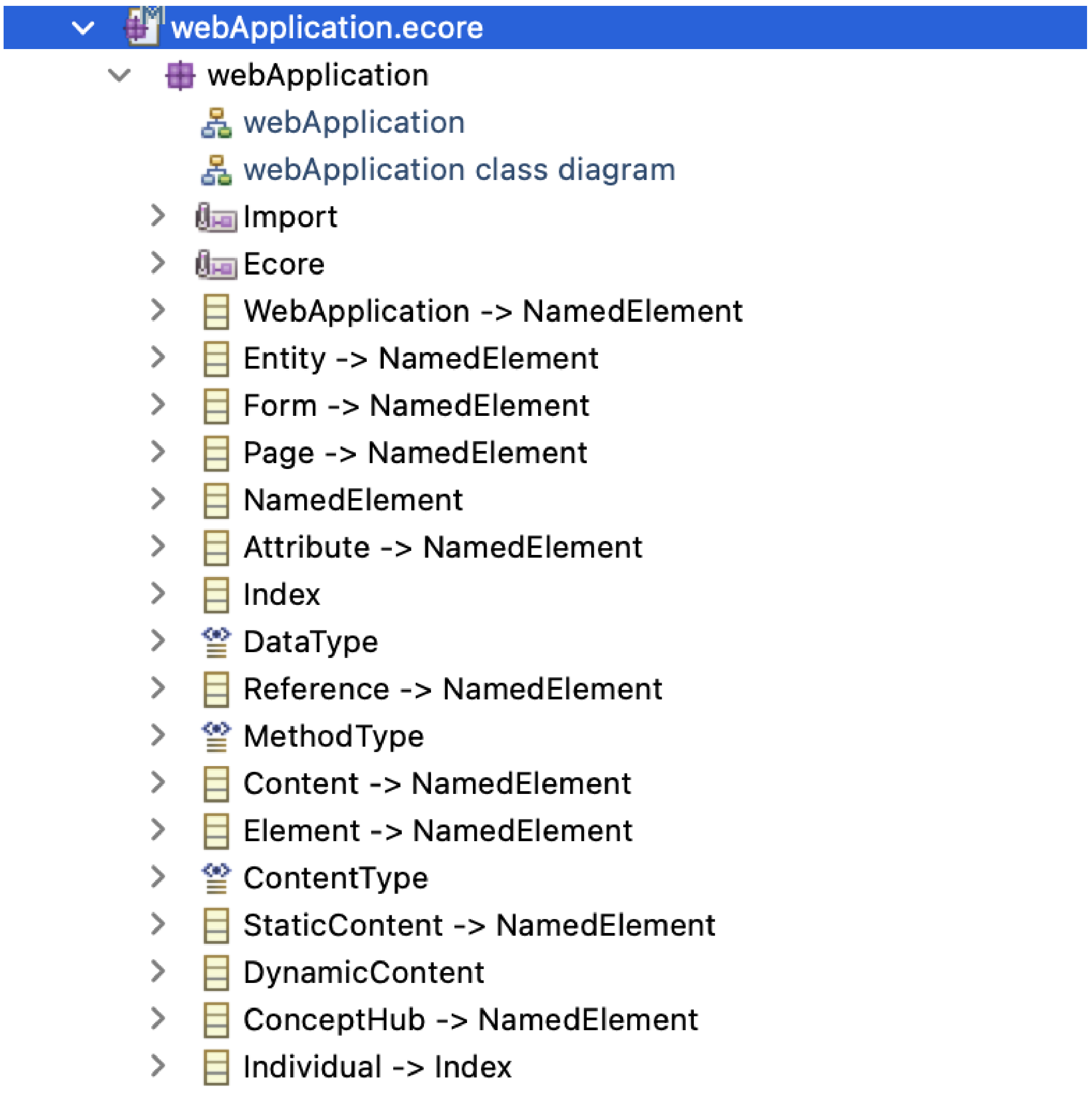
Submitted by

Team: Ctrl_C Ctrl_V

Md Ariful Islam, Sadrul Mizan Nashid, Elmar Karimov

In this assignment we defined “Glot” (DSL) specified for Web Applications (developed in HW2 and HW3) on the EMF platform by using **OCLInEcore**.

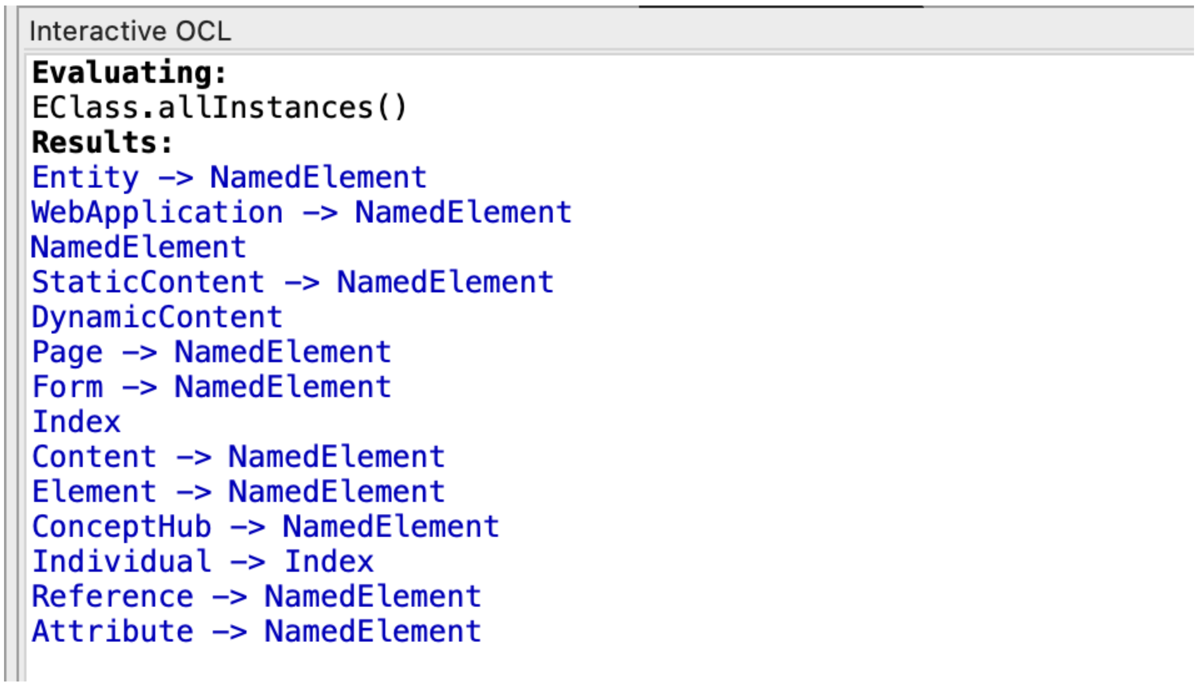
Task A4.1



```
webApplication.ecore
  webApplication
    webApplication
    webApplication class diagram
    Import
    Ecore
    WebApplication -> NamedElement
    Entity -> NamedElement
    Form -> NamedElement
    Page -> NamedElement
    NamedElement
    Attribute -> NamedElement
    Index
    DataType
    Reference -> NamedElement
    MethodType
    Content -> NamedElement
    Element -> NamedElement
    ContentType
    StaticContent -> NamedElement
    DynamicContent
    ConceptHub -> NamedElement
    Individual -> Index
```

Fig 1: Metaclasses

We defined the metamodel describing WebApplications domain satisfying the requirements given. Our metamodel consists of inheritance, relations, containment, non containment, different enumeration types. Metaclasses also have attributes and references.

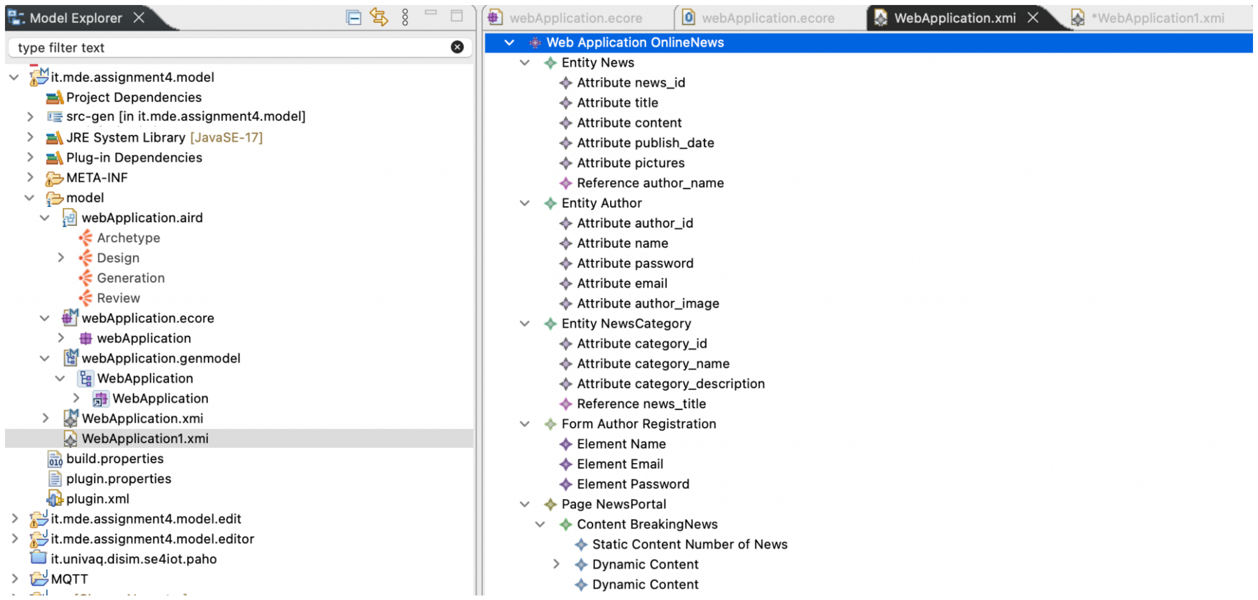


```
Interactive OCL
Evaluating:
EClass.allInstances()
Results:
Entity -> NamedElement
WebApplication -> NamedElement
NamedElement
StaticContent -> NamedElement
DynamicContent
Page -> NamedElement
Form -> NamedElement
Index
Content -> NamedElement
Element -> NamedElement
ConceptHub -> NamedElement
Individual -> Index
Reference -> NamedElement
Attribute -> NamedElement
```

Fig 2: All Instances

Task A4.2

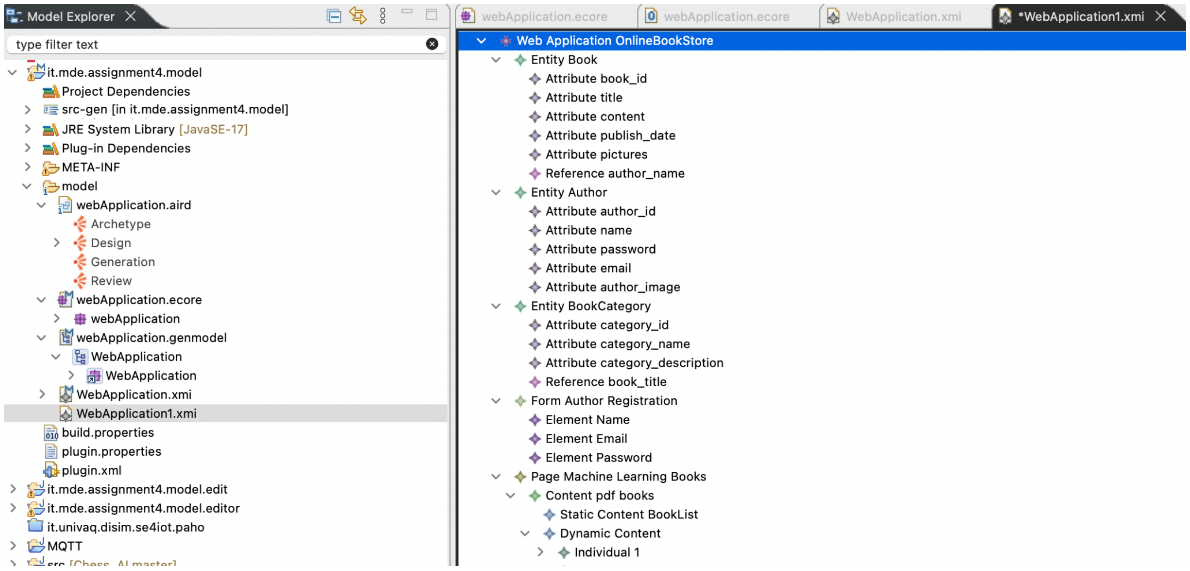
We created two concrete instances called Web Application OnlineNews and Online BookStore illustrated in figure 3 and 4 below, and each metaclasses can be instantiated at the metamodel level in our models.



```
Model Explorer
type filter text
it.mde.assignment4.model
    Project Dependencies
  src-gen [in it.mde.assignment4.model]
  JRE System Library [JavaSE-17]
  Plug-in Dependencies
  META-INF
  model
    webApplication.aird
      Archetype
      Design
      Generation
      Review
    webApplication.ecore
      webApplication
    webApplication.genmodel
      WebApplication
        WebApplication
    WebApplication.xmi
    WebApplication1.xmi
  build.properties
  plugin.properties
  plugin.xml
it.mde.assignment4.model.edit
it.mde.assignment4.model.editor
it.univaq.disim.se4iot.paho
MQTT
```

```
webApplication.ecore | webApplication.ecore | WebApplication.xmi | *WebApplication1.xmi
Web Application OnlineNews
  Entity News
    Attribute news_id
    Attribute title
    Attribute content
    Attribute publish_date
    Attribute pictures
    Reference author_name
  Entity Author
    Attribute author_id
    Attribute name
    Attribute password
    Attribute email
    Attribute author_image
  Entity NewsCategory
    Attribute category_id
    Attribute category_name
    Attribute category_description
    Reference news_title
  Form Author Registration
    Element Name
    Element Email
    Element Password
  Page NewsPortal
    Content BreakingNews
      Static Content Number of News
      Dynamic Content
      Dynamic Content
```

Fig 3: Web Application Online News



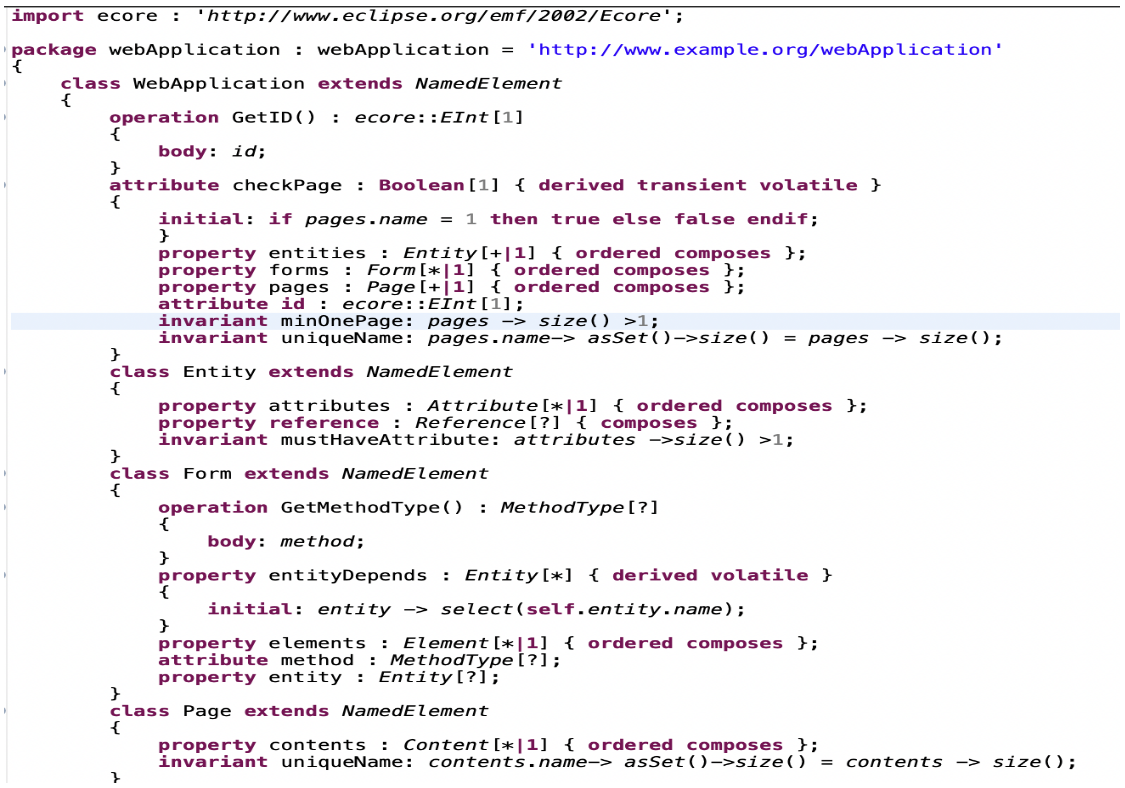
```
Model Explorer
type filter text
it.mde.assignment4.model
    Project Dependencies
  > src-gen [in it.mde.assignment4.model]
  > JRE System Library [JavaSE-17]
  > Plug-in Dependencies
  > META-INF
  v model
    v webApplication.aird
        Archetype
      > Design
        Generation
        Review
    v webApplication.ecore
      > webApplication
    v webApplication.genmodel
      v WebApplication
        > WebApplication
    > WebApplication.xmi
      WebApplication1.xmi
    build.properties
    plugin.properties
    plugin.xml
> it.mde.assignment4.model.edit
> it.mde.assignment4.model.editor
  it.univaq.disim.se4iot.paho
> MQTT
> src [Chess_AI master]
```

```
webApplication.ecore | webApplication.ecore | WebApplication.xmi | *WebApplication1.xmi
v Web Application OnlineBookStore
  v Entity Book
      Attribute book_id
      Attribute title
      Attribute content
      Attribute publish_date
      Attribute pictures
      Reference author_name
  v Entity Author
      Attribute author_id
      Attribute name
      Attribute password
      Attribute email
      Attribute author_image
  v Entity BookCategory
      Attribute category_id
      Attribute category_name
      Attribute category_description
      Reference book_title
  v Form Author Registration
      Element Name
      Element Email
      Element Password
  v Page Machine Learning Books
    v Content pdf books
        Static Content BookList
      v Dynamic Content
        > Individual 1
```

Fig 4: Web Application Online BookStore

Task A4.3

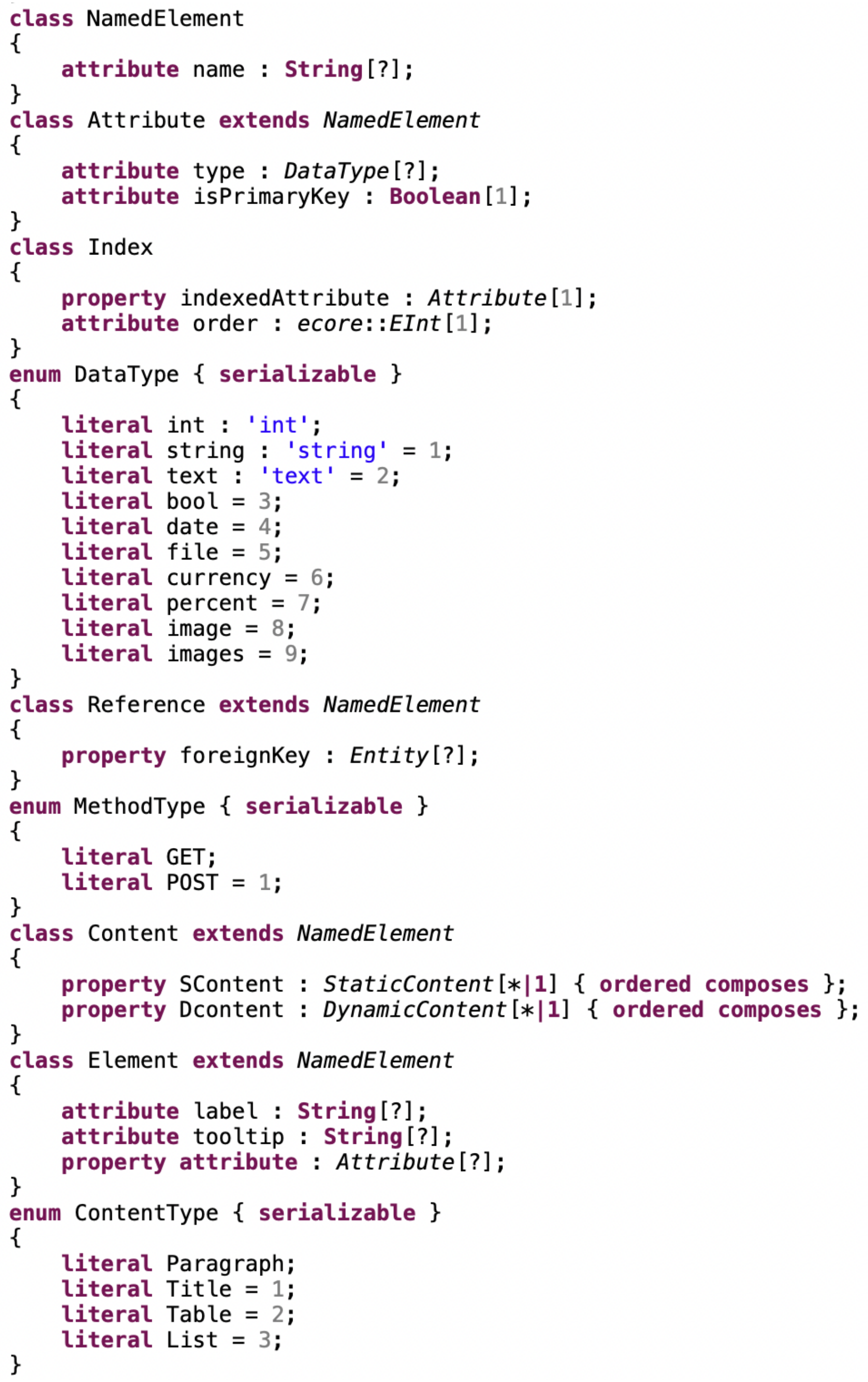
We have added several constraints to validate our models. Developed a few operations as well. As illustrated in the figures below:



```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore';

package webApplication : webApplication = 'http://www.example.org/webApplication'
{
    class WebApplication extends NamedElement
    {
        operation GetID() : ecore::EInt[1]
        {
            body: id;
        }
        attribute checkPage : Boolean[1] { derived transient volatile }
        {
            initial: if pages.name = 1 then true else false endif;
            }
            property entities : Entity[+|1] { ordered composes };
            property forms : Form[*|1] { ordered composes };
            property pages : Page[+|1] { ordered composes };
            attribute id : ecore::EInt[1];
            invariant minOnePage: pages -> size() >1;
            invariant uniqueName: pages.name-> asSet()->size() = pages -> size();
        }
        class Entity extends NamedElement
        {
            property attributes : Attribute[*|1] { ordered composes };
            property reference : Reference[?] { composes };
            invariant mustHaveAttribute: attributes ->size() >1;
        }
        class Form extends NamedElement
        {
            operation GetMethodType() : MethodType[?]
            {
                body: method;
            }
            property entityDepends : Entity[*] { derived volatile }
            {
                initial: entity -> select(self.entity.name);
            }
            property elements : Element[*|1] { ordered composes };
            attribute method : MethodType[?];
            property entity : Entity[?];
        }
        class Page extends NamedElement
        {
            property contents : Content[*|1] { ordered composes };
            invariant uniqueName: contents.name-> asSet()->size() = contents -> size();
        }
```

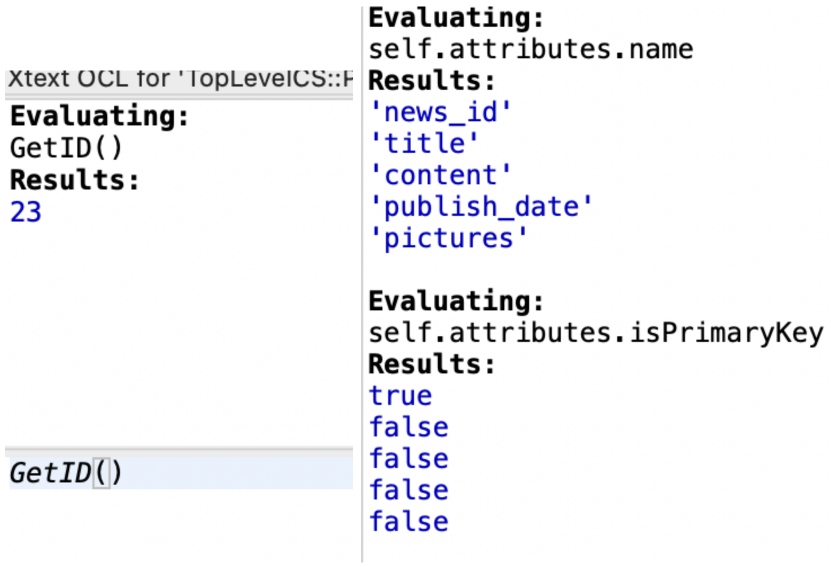
Fig 5: Constraints and operations

Constraints are represented as Invariants in OCL and it specifies all kinds of conditions that the system must comply with as illustrated above.



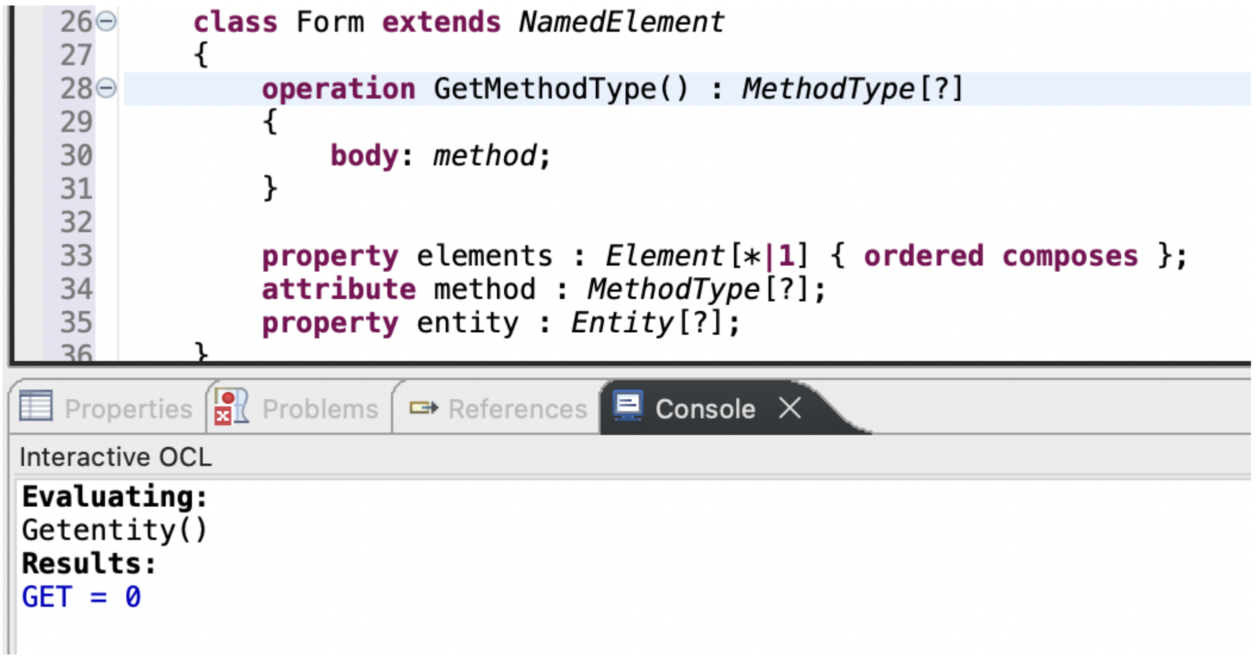
```
class NamedElement
{
    attribute name : String[?];
}
class Attribute extends NamedElement
{
    attribute type : DataType[?];
    attribute isPrimaryKey : Boolean[1];
}
class Index
{
    property indexedAttribute : Attribute[1];
    attribute order : ecore::EInt[1];
}
enum DataType { serializable }
{
    literal int : 'int';
    literal string : 'string' = 1;
    literal text : 'text' = 2;
    literal bool = 3;
    literal date = 4;
    literal file = 5;
    literal currency = 6;
    literal percent = 7;
    literal image = 8;
    literal images = 9;
}
class Reference extends NamedElement
{
    property foreignKey : Entity[?];
}
enum MethodType { serializable }
{
    literal GET;
    literal POST = 1;
}
class Content extends NamedElement
{
    property SContent : StaticContent[*|1] { ordered composes };
    property Dcontent : DynamicContent[*|1] { ordered composes };
}
class Element extends NamedElement
{
    attribute label : String[?];
    attribute tooltip : String[?];
    property attribute : Attribute[?];
}
enum ContentType { serializable }
{
    literal Paragraph;
    literal Title = 1;
    literal Table = 2;
    literal List = 3;
}
```

Fig 6: WebApplication.ecore



```
Xtext OCL for 'TopLevelCS::F
Evaluating:
GetID()
Results:
23

GetID()
```

```
Evaluating:
self.attributes.name
Results:
'news_id'
'title'
'content'
'publish_date'
'pictures'

Evaluating:
self.attributes.isPrimaryKey
Results:
true
false
false
false
false
```



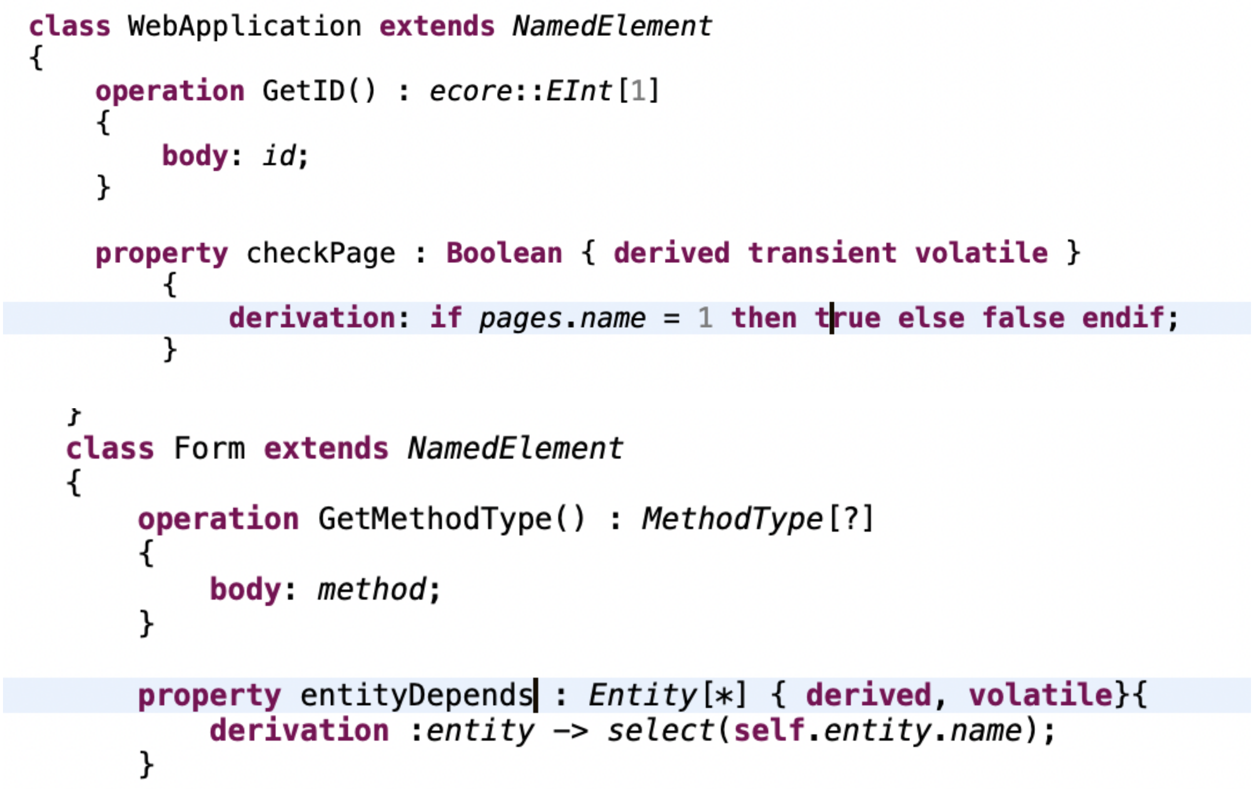
```
    class Form extends NamedElement
    {
        operation GetMethodType() : MethodType[?]
        {
            body: method;
        }

        property elements : Element[*|1] { ordered composes };
        attribute method : MethodType[?];
        property entity : Entity[?];
    }
```

```
Evaluating:
Getentity()
Results:
GET = 0
```

Fig 7: Operations Output

Derived fields:

We have implemented two simple derivation rules. The first one is checking if the pages have unique name access in our web application. The second one verifies which entity is dependent. These two are demonstrated below:



```
class WebApplication extends NamedElement
{
    operation GetID() : ecore::EInt[1]
    {
        body: id;
    }

    property checkPage : Boolean { derived transient volatile }
        {
            derivation: if pages.name = 1 then true else false endif;
        }

  }
  class Form extends NamedElement
  {
      operation GetMethodType() : MethodType[?]
      {
          body: method;
      }

      property entityDepends : Entity[*] { derived, volatile}{
          derivation :entity -> select(self.entity.name);
      }
```

Fig 8: Derived fields

There are lots of possibilities to improve our model. We hope to implement more constraints, operations and derivation rules to make our model more efficient.

References

1. https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.ocl.doc%2Fhelp%2FOCLInterpreterExample.html
2. https://modeling-languages.com/ocl-tutorial/