Department of Computer Science

Cover Sheet for Coursework

Module Code: CO7517

Assignment: "Modelling the Navigational Model of Cloud-based Applications"

Surname (in CAPITALS): MUHAMMAD

First name (in CAPITALS): MOBIN

CFS ID: mimm1

I understand that this is a piece of coursework. I confirm that I handed in a signed Declaration of Academic Honesty Form (available at https://campus.cs.le.ac.uk/ForStudents/plagiarism/) and that I am fully aware of the statements contained therein.

Date: 7th Jan, 2016

Signature: [illegible]

Domain-Specific Languages

(CO7517)

(Mini Project)

Modelling the Navigational Model of Cloud-based Applications

Conveyor
Dr. Artur Boronat
Muhammad Taimoor

Index

Part 1: Design and implement the syntax of the DSL

Introduction

The **WebDSL** is a navigational model for web applications that can be deployed on to a cloud platform. The developers can use the compiler to generate the collection of HTML pages using **ANT** script in terminal console or using **DSL editor** that has features like code suggestion and visual validations. Transformation and the code generation using **XTEXT** and **XTEND** [1].

Modeling Concepts

In the analysis phase the following concepts are identified, the modeling concepts table represents the concepts used in the design, it also includes the intrinsic and extrinsic properties of the model.

Concept	**Intrinsic Properties**	**Extrinsic Properties**
WebApp	name : EString	one DataLayer one NavigationLayer one controllerLayer
NavigationLayer	none	one | many WebPage one Homepage
DataLayer	Class -> name: EString Attribute - > name: EString Type : [String | integer …]	dataLayer contains classes classes contains Attributes
WebPage	name: EString ServerPage | Client Page	one | many Links
Class	name: EString	one to many Classes
Attribute	name: EString Type : [String | integer …]	one to many Attributes
ServerPage	name: EString	zero | many Build ClientPage one Display class
ClientPage	name: EString baseTag: EString	zero | many TextArea zero | many Forms zero | many FrameSet

Form	name: EString method :[GET | POST] encType: ESting accept_charset: EString autocomplete: Boolean nonvalidate : Boolean action : ServerPage	zero | many InputElements zero | many SelectElement zero | one Target to ServerPage zero | many TextArea
FrameSet	Rows : EInt Cols : EInt	zero | many Frame zero | many FrameSet
Frame	name: EString scrolling: Boolean FrameBorder: EInt noResize: Boolean	zero | one source to ClientPage

Target	target=[Frame] |
	targetframe=frames
		zero | one target to Frame

Link	url=EString
	page=[WebPage|ID]
		one Webpage
		zero | many Link_Parameters

TargetedLink	name:EString	zero | one Link to Frame
FrameContent	Source = [ClientPage]	zero | one source to ClientPage
Submit	action=[ServerPage]	zero | one action from ClientPage to ServerPage
Build	build=[ClientPage]	ServerPage generates ClientPages
Redirect	redirect=[ClientPage]	Bidirectional relation between client and server pages
InputElement	name:EString value=EString Type -> [text | radio | Submit …]	Form contains several input elements
SelectElement,	name:EString size : EInt Multiple: Boolean Option -> value: EString name: EString	Form contains several select elements Each select element have several options
TextAreaElement	name:EString test: EString rows: EInt cols: EInt	Form contains several TextArea elements ClientPage contains several TextArea elements

Table 1: Modelling Concepts

1-(a) Domain model as an EMF metamodel

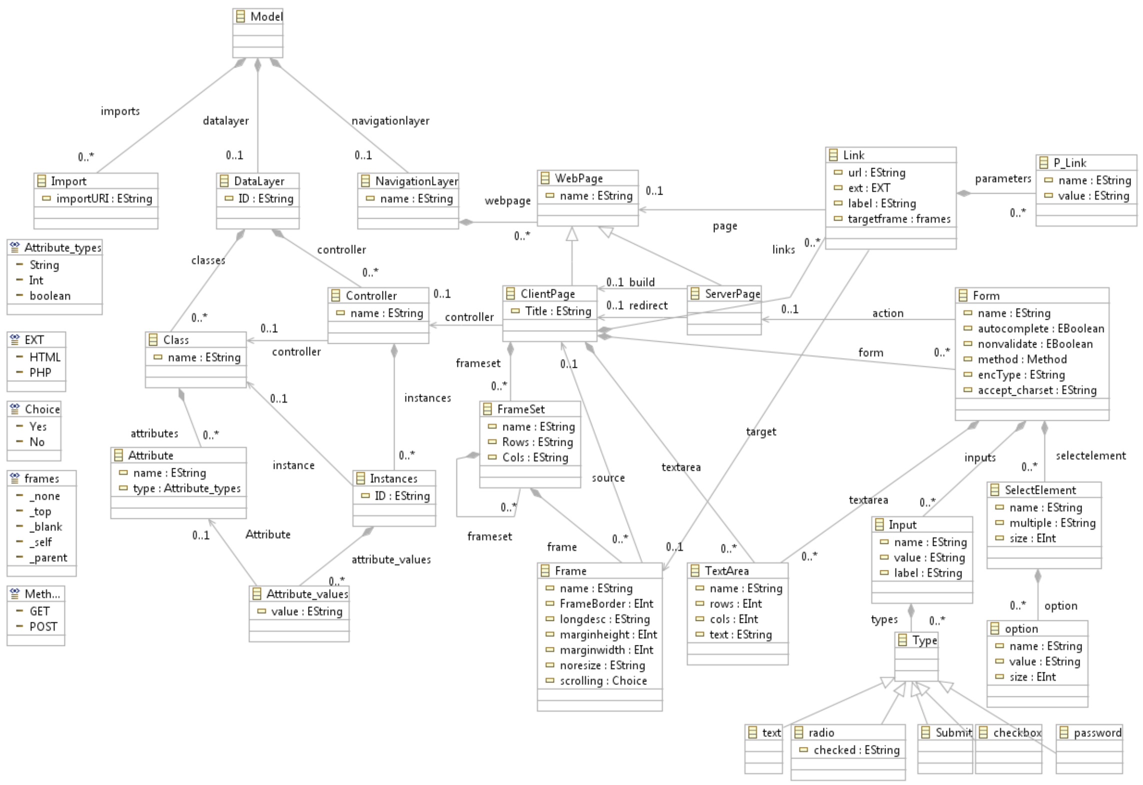


Figure 1: Class Diagram of Wdsl (Large View)

1-a-(1) - OCL Constraints

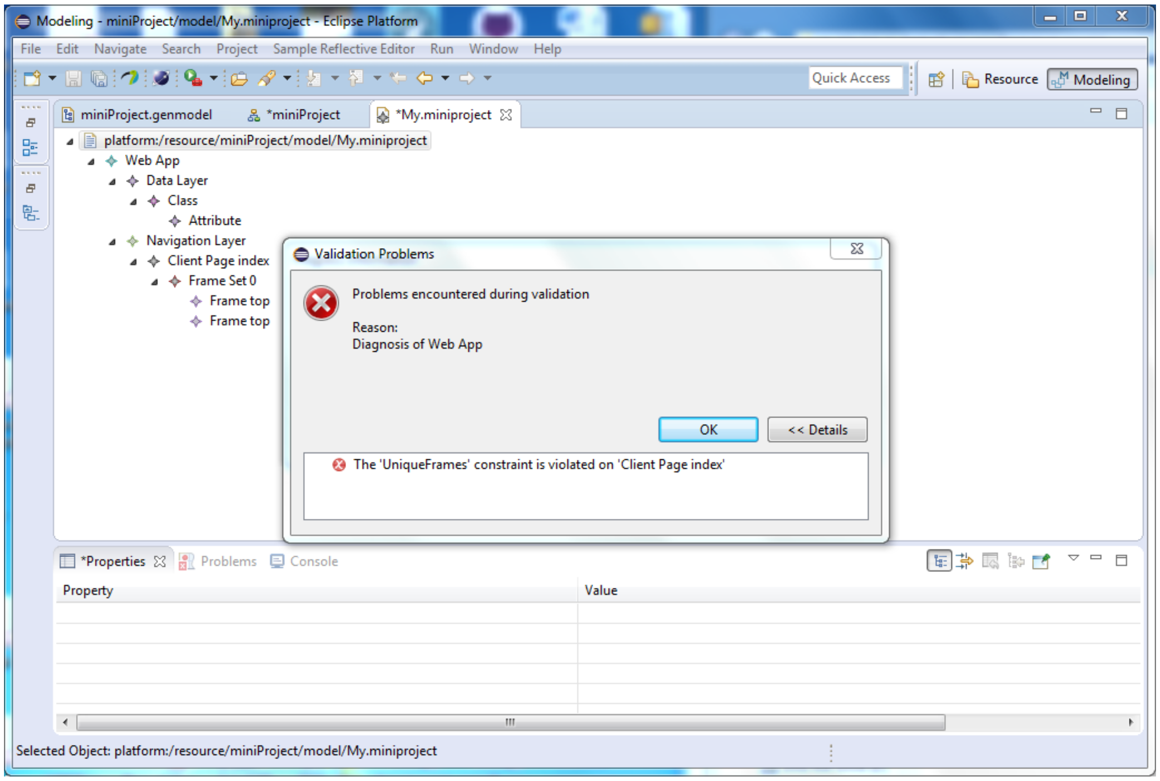
Concept	OCL / Description
Server	The name of the server page must be unique in a navigational layer . `Context: NavigationLayer` `invariant UniqueServerPage` *`serverpage->isUnique(name);`*
Client Page	The name of the client page must be unique in a navigational layer `Context: NavigationLayer` `invariant UniqueClientPage` *`clientpage->isUnique(name);`*
Frame	`The frame defined in a particular frameset must be unique.` `Context: ClientPage` `invariant UniqueTargets` *`frameset.frame->isUnique(name);`*

OCL Constraints (cont.)

Concept	OCL / Description
Build	The input argument of the build must be an instance of client page. `This constrained is captured is Xtexd Validation Class`
Redirect	The input argument of the Redirect must be an instance of Server page. `This constrained is captured is Xtexd Validation Class`

1-a-(2) - OCLinEcore

The OCL performed using OCLinEcore, the test results on two frames name "top" is a client page. A frame name "top" is valid in a different client page. It means that the OCL constraint duplicate frame names in a client page.


Modeling - miniProject/model/My.miniproject - Eclipse Platform
File Edit Navigate Search Project Sample Reflective Editor Run Window Help
Quick Access
Resource
Modeling
miniProject.genmodel
*miniProject
*My.miniproject
platform:/resource/miniProject/model/My.miniproject
Web App
Data Layer
Class
Attribute
Navigation Layer
Client Page index
Frame Set 0
Frame top
Frame top
*Properties
Problems
Console
Property
Value
Selected Object: platform:/resource/miniProject/model/My.miniproject

Validation Problems

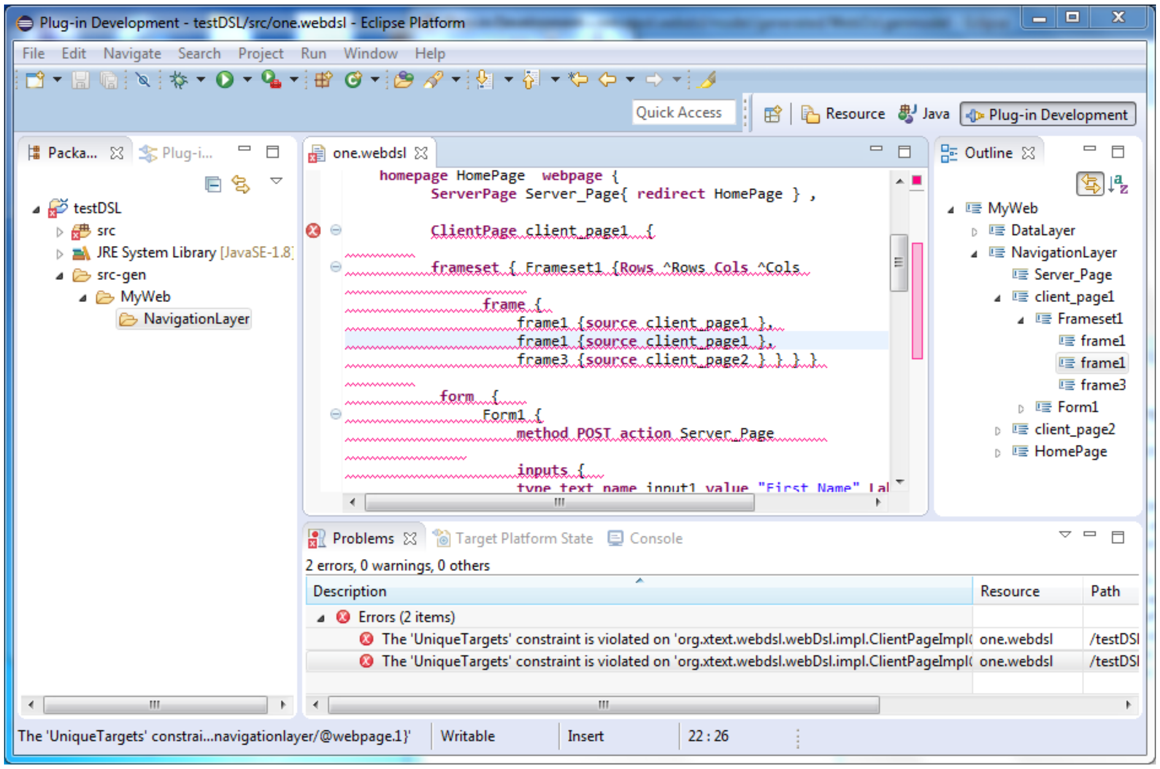
Problems encountered during validation

Reason:
Diagnosis of Web App

OK   << Details

The 'UniqueFrames' constraint is violated on 'Client Page index'

Figure 2: OCL Test for Unique Frames



```
homepage HomePage  webpage {
        ServerPage Server_Page{ redirect HomePage } ,

        ClientPage client_page1  {

        frameset { Frameset1 {Rows ^Rows Cols ^Cols

                frame {
                    frame1 {source client_page1 },
                    frame1 {source client_page1 },
                    frame3 {source client_page2 } } } }

         form  {
                Form1 {
                    method POST action Server_Page

                    inputs {
                    type text name input1 value "First Name" Lal
```

2 errors, 0 warnings, 0 others

Description	Resource	Path
Errors (2 items)		
The 'UniqueTargets' constraint is violated on 'org.xtext.webdsl.webDsl.impl.ClientPageImpl(	one.webdsl	/testDSl
The 'UniqueTargets' constraint is violated on 'org.xtext.webdsl.webDsl.impl.ClientPageImpl(	one.webdsl	/testDSl

The 'UniqueTargets' constrai...navigationlayer/@webpage.1}' | Writable | Insert | 22 : 26

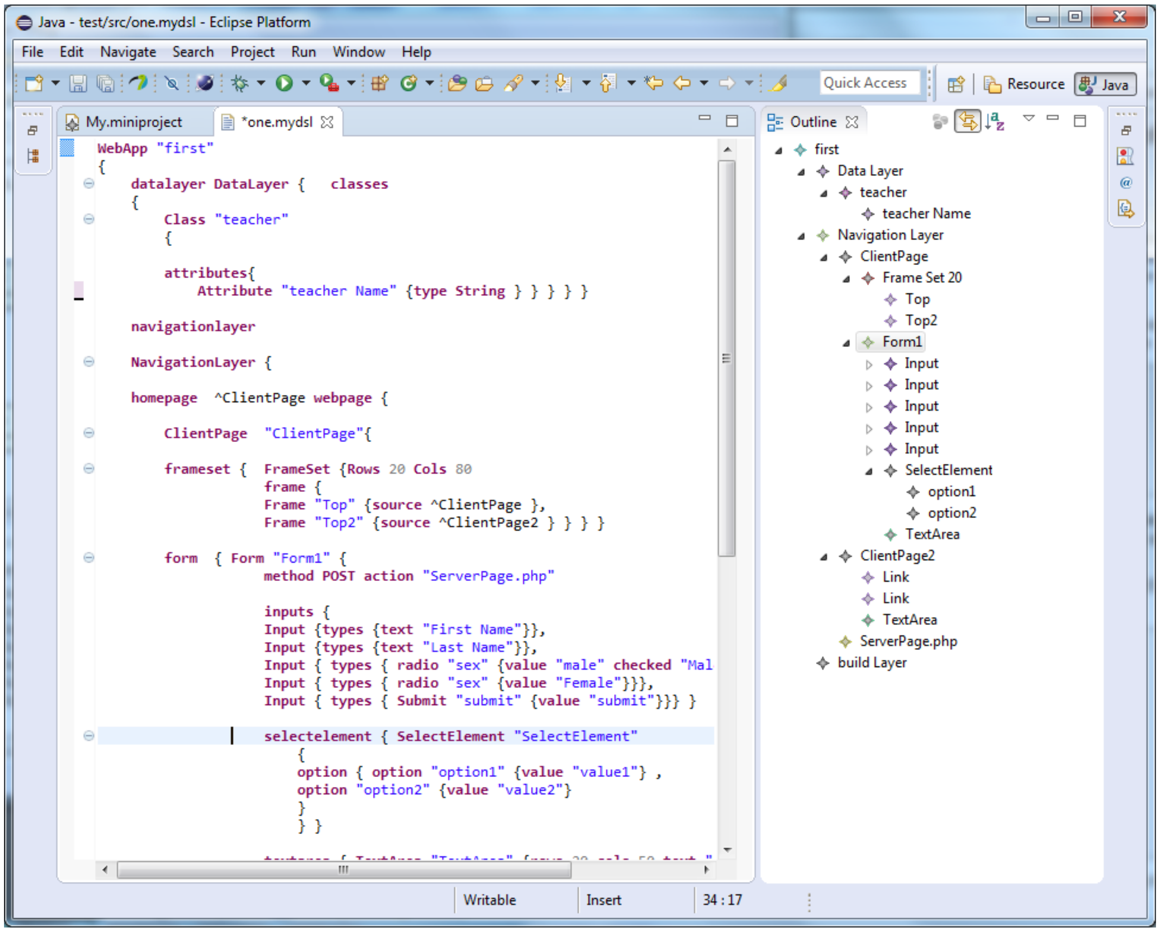
Figure 3: OCL test in Xtext

1-a-(3) -Validations in Xtend

Another option for validation is xtend based API inside XtendValidation Class, I have chosen this feature to validate in a given navigational layer the elements like Webpage, frames, forms etc. to detect the duplication of ID's. It can be easily done by setting the 'NamesAreUniqueValidator'. In the Mwe2 file.

```
// Xtend-based API for validation
    fragment = validation.ValidatorFragment auto-inject {
    // composedCheck = "org.eclipse.xtext.validation.ImportUriValidator"
    composedCheck = "org.eclipse.xtext.validation.NamesAreUniqueValidator"
    }
```

1-(b) Concrete syntax for the language.



```
WebApp "first"
{
    datalayer DataLayer {   classes
    {
        Class "teacher"
        {

        attributes{
            Attribute "teacher Name" {type String } } } } }

    navigationlayer

    NavigationLayer {

    homepage  ^ClientPage webpage {

        ClientPage  "ClientPage"{

        frameset {  FrameSet {Rows 20 Cols 80
                    frame {
                    Frame "Top" {source ^ClientPage },
                    Frame "Top2" {source ^ClientPage2 } } } }

        form  { Form "Form1" {
                    method POST action "ServerPage.php"

                    inputs {
                    Input {types {text "First Name"}},
                    Input {types {text "Last Name"}},
                    Input { types { radio "sex" {value "male" checked "Mal
                    Input { types { radio "sex" {value "Female"}}},
                    Input { types { Submit "submit" {value "submit"}}} }

                    selectelement { SelectElement "SelectElement"
                        {
                        option { option "option1" {value "value1"} ,
                        option "option2" {value "value2"}
                        }
                        } }
```

```
first
    Data Layer
        teacher
            teacher Name
    Navigation Layer
        ClientPage
            Frame Set 20
                Top
                Top2
            Form1
                Input
                Input
                Input
                Input
                Input
                SelectElement
                    option1
                    option2
                TextArea
        ClientPage2
            Link
            Link
            TextArea
        ServerPage.php
    build Layer
```

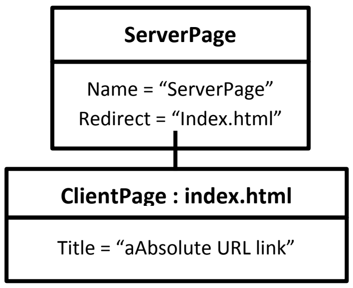
Java - test/src/one.mydsl - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Resource
Java
My.miniproject
*one.mydsl
Outline
Writable
Insert
34 : 17

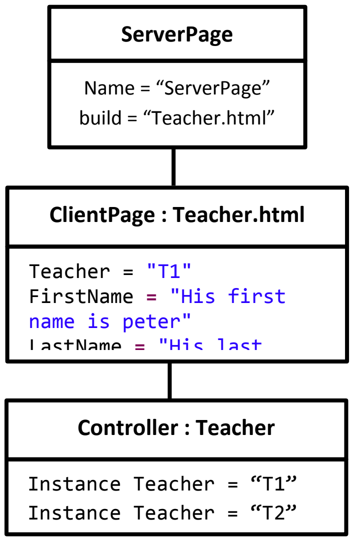
Figure 4 : Textual Concrete syntax

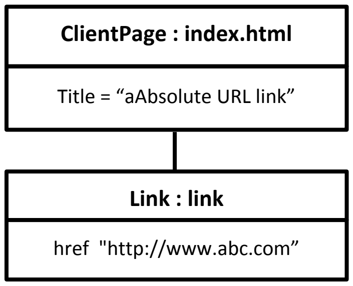
Part II - Implementation

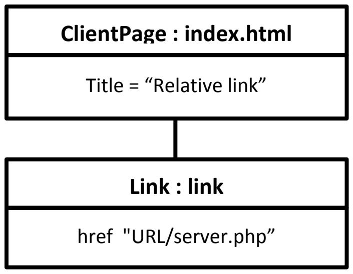
2 (a) - Translation process

MetaModel Pattern	Abstract Syntax	OCL/Description

 ServerPage
Name = "ServerPage"
Redirect = "Index.html"
ClientPage : index.html
Title = "aAbsolute URL link"
	"Redirect Concept"
	
	`«IF s.redirect != null»`
	`«s.redirect.name».html</h3>";`
	`header("refresh:10;`
	`«s.redirect.name».html");`
	`?>`
		`if (!(page.redirect instanceof ClientPage)){`
		`warning('A server can redirect to client page only',`
		`WDslPackage.Literals.SERVER_PAGE__REDIRECT)}`

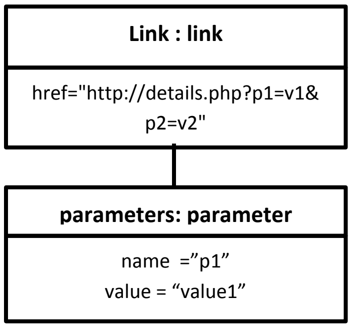
 ServerPage
Name = "ServerPage"
build = "Teacher.html"
ClientPage : Teacher.html
Teacher = "T1"
FirstName = "His first name is peter"
LastName = "His last
Controller : Teacher
Instance Teacher = "T1"
Instance Teacher = "T2"
	"Build Concept"
	
	`def render_cPage(ClientPage c)`
	`«FOR tr :`
	`c.controller.instances»`
	`«tr.instance.name»`
	`  «FOR td :`
	`    tr.attribute_values»`
	`    «td.attribute.name»`
	`    «td.value»`
	`  «ENDFOR»`
	`«ENDFOR»`
		The contraints of build is handled in Xtext grammar, The build object only accepts the client page as an input.
		
		The duplicate names of controller, server pages and client pages is handled in mwe2 workflow.

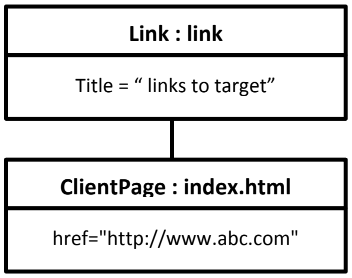
 ClientPage : index.html
Title = "aAbsolute URL link"
Link : link
href "http://www.abc.com"
	`«IF(link.url!= null &&`
	`link.page ==`
	`null)»«link.url»`
	`«ENDIF»`
		Client page has links to absolute URL

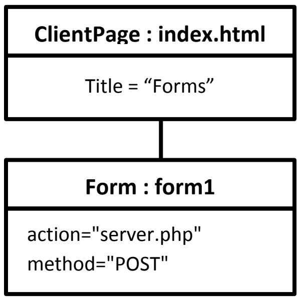
 ClientPage : index.html
Title = "Relative link"
Link : link
href "URL/server.php"
	`«IF(link.url!= null &&`
	`link.page !=`
	`null)»«link.url»/«link.page.name».«link.ext»`
		Link to relative a relative path, extension of the webpages are enum type
		
		`enum EXT:`
		`HTML = 'html' | PHP = 'php';`

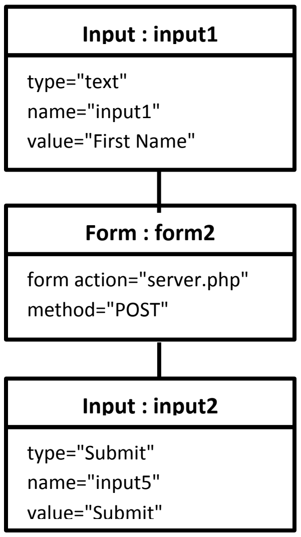
Translation process (Cont.)

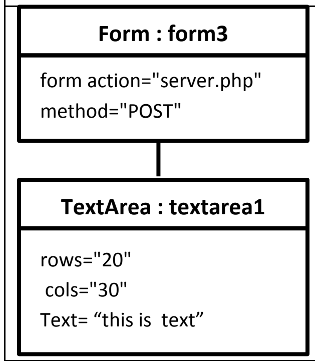
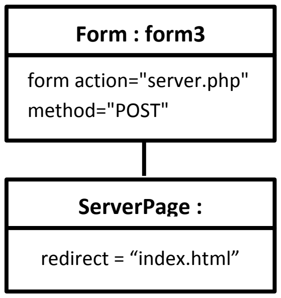
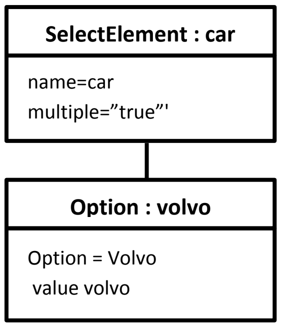
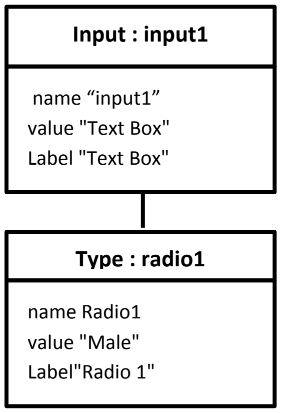
MetaModel Pattern	Abstract Syntax	OCL/Description

 Link : link
href="http://details.php?p1=v1&p2=v2"
parameters: parameter
name ="p1"
value = "value1"
	«FOR link : c.links»
	  FOR p : link.parameters SEPARATOR "&"»«p.name»=«p.value»
	 «ENDFOR»"
	«ENDFOR»"«
		Link passing parameters in URL.

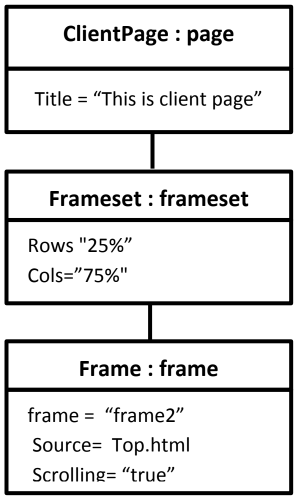
 Link : link
Title = " links to target"
ClientPage : index.html
href="http://www.abc.com"
	«IF link.target !=null» target="«link.target.name»"«END IF»«IF link.targetframe.literal !='_none'» target="«link.targetframe»"«END IF»	Link source to a webpage, either client to server page.

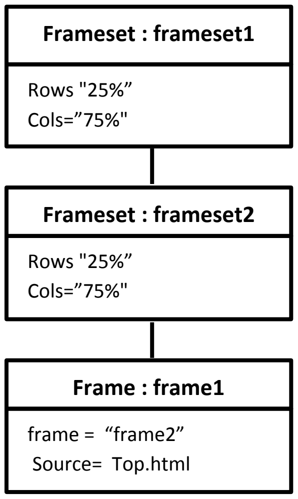
 ClientPage : index.html
Title = "Forms"
Form : form1
action="server.php"
method="POST"
	«FOR form : c.form»
	<h2>«form.name»</h2>
	<form action="«form.action.name».php" method="«form.method»">
	</form>
	«ENDFOR»
		A client page can submits form to a server page only.
		if ((!form.action instanceof ServerPage)) {
		warning('Form action must be a Server Page', WDslPackage.Literals.FORM__ACTION)
		}

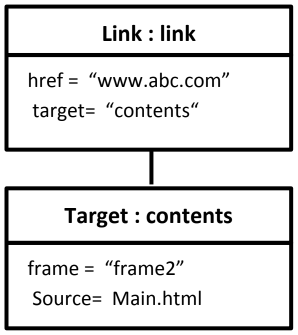
 Input : input1
type="text"
name="input1"
value="First Name"
Form : form2
form action="server.php"
method="POST"
Input : input2
type="Submit"
name="input5"
value="Submit"
	«FOR input : form.inputs»
	   «FOR type : input.types»
	«input.label» <input type="«type.eClass.name.toString»" name="«input.name»" value="«input.value»">
	
	   «ENDFOR»
	
	«ENDFOR»
		A form contains several inputs elements and each input can be of type, text, submit, radio, checkbox or password.

MetaModel Pattern	Abstract Syntax	OCL
 Form : form3 form action="server.php" method="POST" TextArea : textarea1 rows="20" cols="30" Text= "this is text"	`«FOR form : c.form» «FOR select : form.textarea»<textarea rows="«select.rows»" cols="«select.cols»"> «select.text»</textarea> «ENDFOR» «ENDFOR»`	A form contains one or more text area elements. Option to adjust rows and columns of textarea.
 Form : form3 form action="server.php" method="POST" ServerPage : redirect = "index.html"	`«FOR form : c.form» <form action="«form.action.name».php" method="«form.method»"> /form> «ENDFOR» Server Rule: '''<?php foreach($_POST as $key=>$value) { if($key != "Submit"){ echo "<h4>$key=$value</h4>"; echo "<br>";} } header("refresh:10; «s.redirect.name».html"); ?>'''`	`enum Method: GET = 'GET' | POST = 'POST';`
 SelectElement : car name=car multiple="true"' Option : volvo Option = Volvo value volvo	`«FOR select : form.selectelement» <select name= «select.name» «select.multiple»> «FOR option : select.option» <option value="«option.value»"> «option.name» </option> «ENDFOR» /<select> <br> «ENDFOR»`	If "multiple" propertiy is true, the list box appears and multiple selection is possible using control key.
 Input : input1 name "input1" value "Text Box" Label "Text Box" Type : radio1 name Radio1 value "Male" Label"Radio 1"	`«FOR input : form.inputs» «FOR type : input.types» «input.label» <input type="«type.eClass.name.toString»" name="«input.name»" value="«input.value»"><br> «ENDFOR» <br> «ENDFOR»`	The input types are `Type: text | radio | checkbox | Submit | password;`

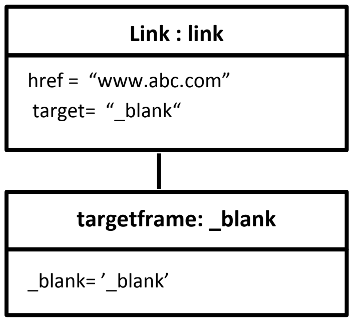
MetaModel Pattern	Abstract Syntax	OCL

 ClientPage : page
Title = "This is client page"
Frameset : frameset
Rows "25%"
Cols="75%"
Frame : frame
frame = "frame2"
Source= Top.html
Scrolling= "true"
	`«FOR fs : c.frameset»`
	`<frameset rows="«fs.rows»" cols="«fs.cols»">`
	`«FOR frame : fs.frame»`
	`<frame name="«frame.name»" src="«frame.source.name».html">`
	`«ENDFOR»`
	`</frameset>`
	`«ENDFOR»`
		A webpage should have unique frame names
		
		OCL:
		
		Context: WebPage
		invariant
		UniqueTargets:
		frameset.frame->isUnique(*name*);

 Frameset : frameset1
Rows "25%"
Cols="75%"
Frameset : frameset2
Rows "25%"
Cols="75%"
Frame : frame1
frame = "frame2"
Source= Top.html
	`«FOR fs : c.frameset»`
	`<frameset rows="«fs.rows»" cols="«fs.cols»">`
	`«FOR frame : fs.frame»`
	`<frame name="«frame.name»" src="«frame.source.name».html">`
	`«FOR nfs : fs.frameset»`
	`<frameset rows="«ns.rows»" cols="«nfs.cols»">`
	`«FOR nf : nfs.frame»`
	`<frame name="«nf.name»" src="«nf.source.name».html">`
	`«ENDFOR»`
	`«ENDFOR»`
	`«ENDFOR»`
	`</frameset>`
	`</frameset>`
	`«ENDFOR»`
		Nested Framesets
		
		Used in three frames webpages, top, left and right.
		
		nfs = nested frameset
		
		nf = frames of nested frame

 Link : link
href = "www.abc.com"
target= "contents"
Target : contents
frame = "frame2"
Source= Main.html
	`«IF link.target !=null»`
	`target="«link.target.name»"`
	`«ENDIF»`
		A target of a link could be a named frame.

MetaModel Pattern	Abstract Syntax	OCL

 Link : link
href = "www.abc.com"
target= "_blank"
targetframe: _blank
_blank= '_blank'
	`«IF link.targetframe.literal !='_none'» target="«link.targetframe»" «ENDIF»`	A target could be a link or a targeted link, Example of targeted links are `_top = '_top'|_blank = '_blank' |_self = '_self' |_parent='_parent';`

2 (b) Implementation

The objective of the WebDSL is an attempt to automate the navigational model of cloud-based applications. Jim Conallel in his book has explained the use of the DSL for designing web applications [2]. The tool supports the web designers by automating the generation process of HTML pages. The syntax of the language resembles HTML to which most web developers already familiar and they are not required to learn a new language.

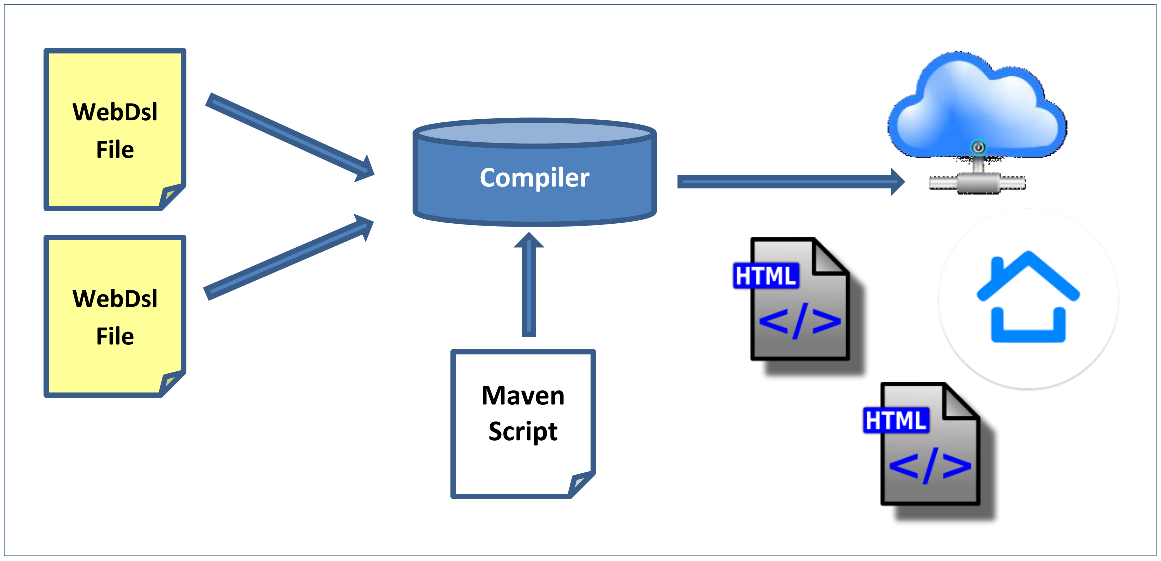


Figure 5: Overall concept of Cloud base Application using WebDSL

Client Pages

Xtend is used for code generation, doGenerate methods overrided in the IGenerator.xtend file. Xtend org.eclipse.xtext.generator.IGenerator Interface is useful for code generation. When a user save the file, this Generator is called by the Builder using org.eclipse.xtext.builder.IXtextBuilderParticipant.

XText Grammar

```
ClientPage:
      'ClientPage'name=ID
       '{'
             ('controller' controller=[Controller|ID])?
             ('Title' Title=STRING)?
             ( links+=Link (links+=Link)*)?
             ( 'frameset' frameset+=FrameSet*)?
             ( 'form' form+=Form*)?
             ( 'textarea' textarea+=TextArea*)?
       '}';
```

The templates for client and server pages defined in separate methods. The client page renders frames outside the <body> tag and inside it renders clientpage if a controller is attached to the page, it also render form and link in the html body tag.

```
override void doGenerate(Resource resource, IFileSystemAccess fsa) {
   for (c : resource.allContents.toIterable.filter(typeof(ClientPage))) {
      fsa.generateFile( c.name + ".html", c.generateClientPage)
   }
```

Template

```
<html>
«renderFrameset(c)»
<body>
       <h1>«c.name»</h1>
       «IF c.controller != null»
            «renderClientPage(c)»
       «ENDIF»
        «renderLinks(c)»
        «renderForm(c)»
</body>
</html>'''
```

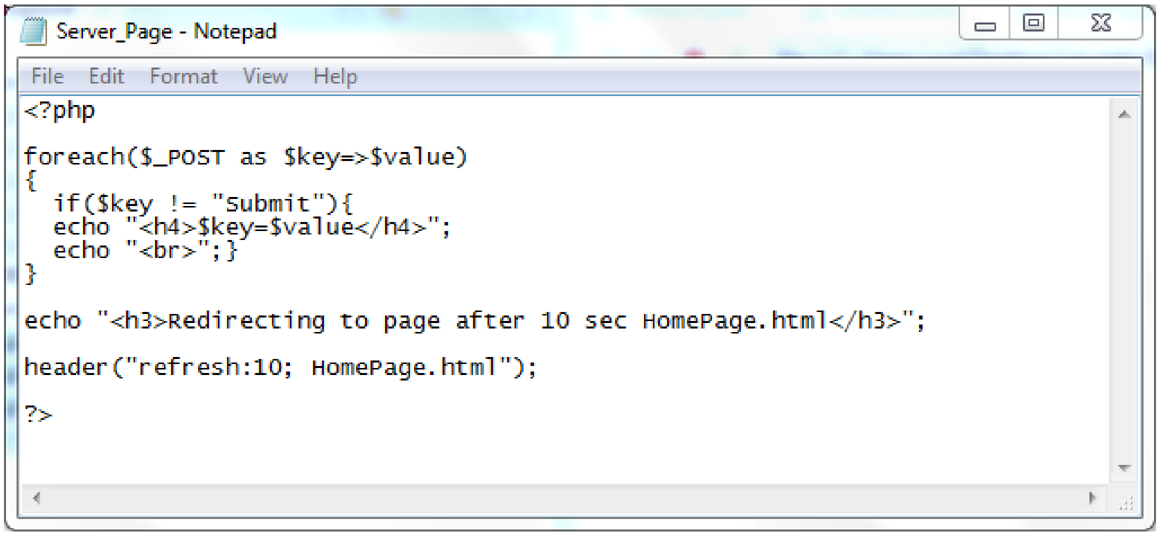
Server Pages

In cloud based web application the server has two roles, it build the client pages and redirect to a specific URL. The following section describes the generation of server pages. The following Xtend code generates the server pages.

```
override void doGenerate(Resource resource, IFileSystemAccess fsa) {
   for (s : resource.allContents.toIterable.filter(typeof(ServerPage))) {
      fsa.generateFile( s.name + ".php", s.generateServerPage)
   }
```

Template

```
def generateServerPage(ServerPage s) '''

<?php
foreach($_POST as $key=>$value)
{
   if($key != "Submit"){
   echo "$key=$value";
   echo "<br>";}
}

echo "<h3>Redirecting to page after 10 sec «s.redirect.name».html</h3>";

header("refresh:10; «s.redirect.name».html");

?>

'''
```



```
<?php

foreach($_POST as $key=>$value)
{
  if($key != "Submit"){
  echo "<h4>$key=$value</h4>";
  echo "<br>";}
}

echo "<h3>Redirecting to page after 10 sec HomePage.html</h3>";

header("refresh:10; HomePage.html");

?>
```

Figure 6: Generated PHP code

Build

Grammar

```
ServerPage:'ServerPage'
       name=ID
        ('build' build=[ClientPage])?
```

Template

```
def generateServerPage(ServerPage s) '''<?php
  «IF s.build != null»
  «renderClientPage(s.build)»
  «ENDIF»
'''

def renderClientPage(ClientPage c) '''
<!DOCTYPE html>

<html>
<body>
       «FOR tr : c.controller.instances»
       <table border = "1" style="width:100%">
       <tr>«tr.instance.name» <tr>
           «FOR td : tr.attribute_values»
               <td>«td.attribute.name»<td>
               <td>«td.value»<td>
           «ENDFOR»«»
       </table><br>
       «ENDFOR»
</body>
</html>'''
```

Cloud Applications

First created a “Teacher” class in the “Data layer”, Create a controller for Teacher Class and added two instances of teacher class. In the navigation layer created a “server page” and a “client page”. The controller Teacher defined in the client page. In the last used “build” statement in the server page to generate the client page based on the “teacher” class.

Programming in WebDSL

To implement the navigational model of cloud base web application the suggested framework uses model, view and controller based approach. Each page is assigned a controller. The model layer is the Data layer in which Classes and attributes can be are defined. In the following example using the webDSL languages programmer can define in a single program in few step.

The steps are simplified as:

Data Layer:

Step 1: Create classes and attributes (e.g. ClassTeacher with attribute “First and Last name”)

Step 2: Create a Controller

Step 3: Create instances of Controller (e.g. “Teacher1” and “teacher2”)

Navigation Layer:

Step 1: Create a Server Page

Step 2: Assign which page the server will build

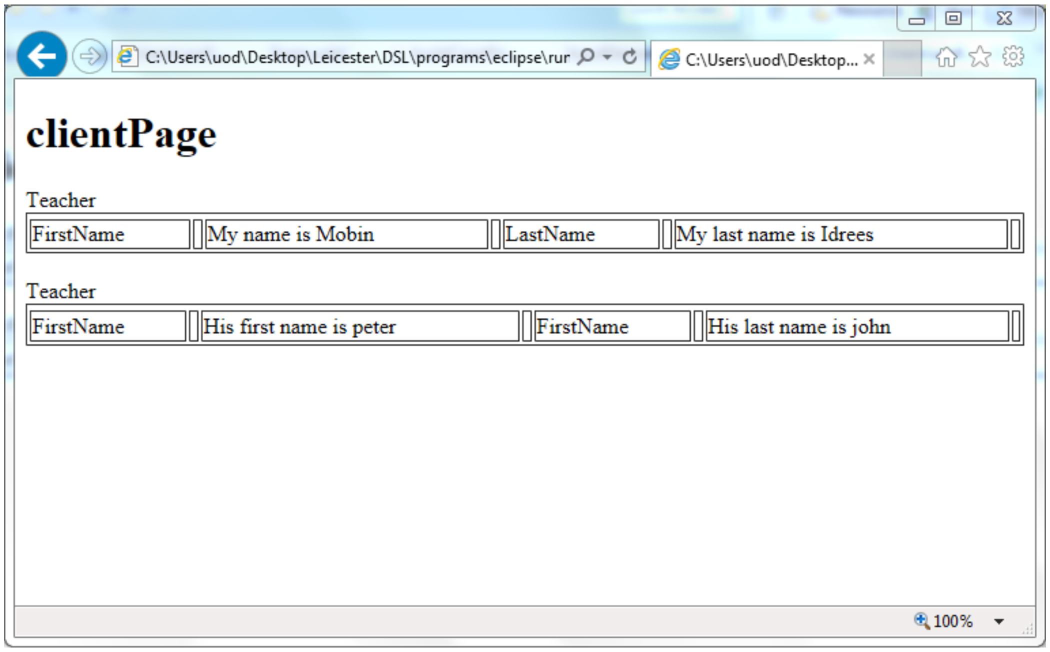
Step 3: Create Client page (e.g. “displayTeacher”)

Step 4: Set controller to be used for “displayTeacher” page

Step 5: Upload the generated pages to a Web Server.

Complete code used in the scenario.

```
DataLayer DataLayer {
Class Teachcer { Attribute FirstName  { type String }
                 Attribute LastName  { type String }  }
Controller Teachcer {
              instance Teachcer "teacher1"
                 Attribute FirstName value "My name is Mobin"
                 Attribute LastName value  "My last name is Idrees"
              instance Teachcer "teacher2"
                 Attribute FirstName value "His first name is peter"
                 Attribute FirstName value "His last name is john" } }
NavigationLayer {
             ServerPage Server_Page{
                 redirect homePage
                 build clientPage       }

             ClientPage displayTeacher  {
                 controller Teachcer    }

             ClientPage homePage  {}
}
```



clientPage

Teacher

FirstName	My name is Mobin	LastName	My last name is Idrees

Teacher

FirstName	His first name is peter	FirstName	His last name is john

Figure 7: Generate Client Page

Redirect

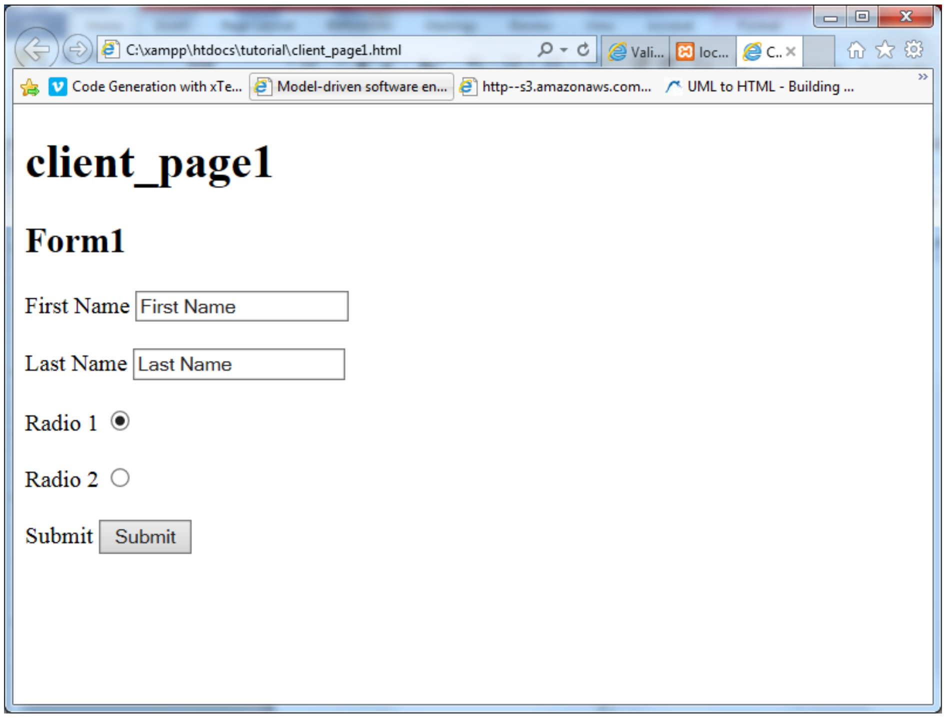
Grammar

```
ServerPage:
	'ServerPage' name=ID
	('redirect' redirect=[ClientPage])?
```

Template

```
«IF s.redirect != null»
echo "<h3>Redirecting to page after 10 sec «s.redirect.name».html</h3>";
header("refresh:10; «s.redirect.name».html");
?>
```

The following example illustrates the redirect process, the following generated client form submits to the server page using HTTP Post, the contents of the form are displayed and the server redirects to a different page after a certain time interval.



C:\xampp\htdocs\tutorial\client_page1.html

client_page1

Form1

First Name [First Name]

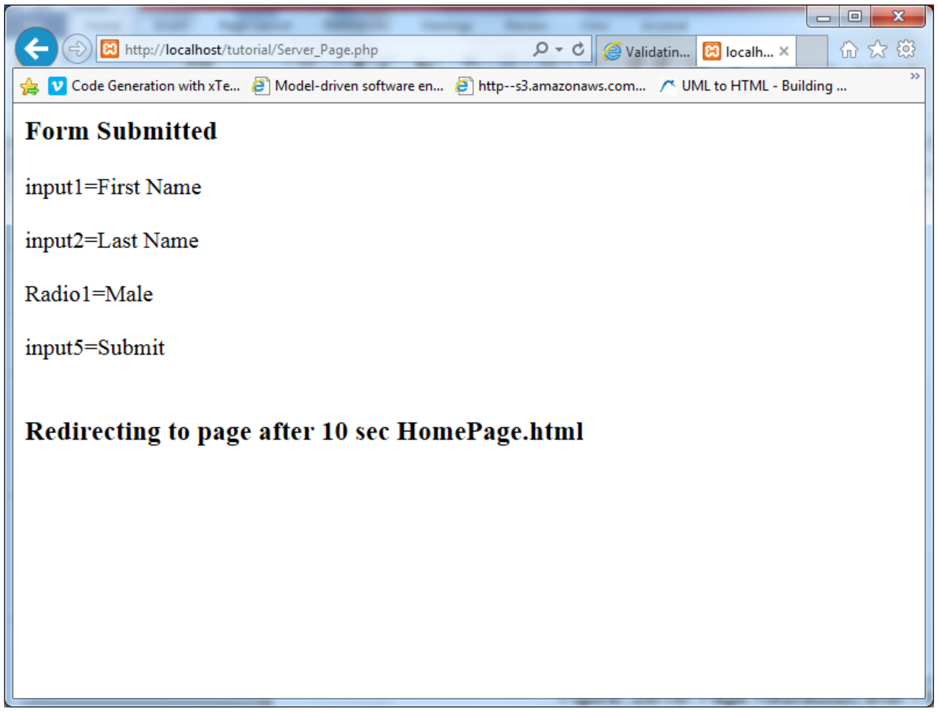
Last Name [Last Name]

Radio 1 ●

Radio 2 ○

Submit [Submit]

Figure 8: Client page submiting to the Server Page.



http://localhost/tutorial/Server_Page.php

Form Submitted

input1=First Name

input2=Last Name

Radio1=Male

input5=Submit

Redirecting to page after 10 sec HomePage.html

Figure 9: Server Page Redirection

Forms

Grammar

```
Form: {Form} name=ID
       'form'
       (autocomplete?='autocomplete')?
       (nonvalidate?='nonvalidate')?

       '{'
              ('method' method=Method)?
              ('encType' encType=ID)?
              ('accept_charset' accept_charset=ID)?
              ('action' action=[ServerPage|ID])?
              ('inputs' '{' inputs+=Input* '}' )?
              ('selectelement' '{' selectelement+=SelectElement* '}' )?
              ('textarea' '{' textarea+=TextArea* '}' )?
       '}';
```

Template

```
def renderForm(ClientPage c) '''
«FOR form : c.form»
       <h2>«form.name»</h2>
       <form action="«form.action.name».php" method="«form.method»">
       «FOR input : form.inputs»
              «FOR type : input.types»
               «input.label» <input type="«type.eClass.name.toString»"
name="«input.name»" value="«input.value»"><br>
              «ENDFOR»<br>
        «ENDFOR»<br>
        «FOR select : form.selectelement»
               <select name= «select.name» «select.multiple»>
        «FOR option : select.option»
               <option value="«option.value»"> «option.name» </option>
              «ENDFOR»
                     /<select> <br>
        «ENDFOR»<br>
        «FOR select : form.textarea»
              <textarea rows="«select.rows»" cols="«select.cols»">
               «select.text»
               </textarea>
        «ENDFOR»

  </form>
 «ENDFOR» '''
```

Submit

Grammar

```
Input: {Input}
      'type' types+=Type*
Type:
       text | radio | checkbox | Submit | password;

Submit: {Submit} 'submit';
enum Method:
       GET = 'GET' | POST = 'POST';
```

Template Server

```
foreach($_POST as $key=>$value)
{
  if($key != "Submit"){
  echo "<h4>$key=$value</h4>";
  echo "<br>";} }
```

FrameSet

Grammar

```
FrameSet: name =ID
             ('Rows' Rows=STRING)?
             ('Cols' Cols=STRING)? ">"
             ('frame' (frame+=Frame)*)
             ('frameset' frameset+=FrameSet)?;
```

Template

```
def renderFrameset(ClientPage c) '''
 «FOR fs : c.frameset»
<frameset rows="«fs.rows»" cols="«fs.cols»">
                     «FOR frame : fs.frame»
                     <frame name="«frame.name»"
src="«frame.source.name».html">
                          «FOR fs2 : fs.frameset»
                          <frameset rows="«fs2.rows»" cols="«fs2.cols»">
                              «FOR frame2 : fs2.frame»
                           <frame name="«frame2.name»"
src="«frame2.source.name».html">
                              «ENDFOR»
                          «ENDFOR»
                      «ENDFOR»</frameset>
 </frameset>
«ENDFOR» '''
```

Links

Grammar

```
Link: {Link}
       'href' ('url' url=STRING)?
              ('page' page=[WebPage|ID])?('.' ext=EXT)?
              ('label' label=STRING)?
              ('target' target=[Frame])?
              ('target' targetframe=frames)?
              ('parameters' parameters+=P_Link*)?
```

Template

```
def renderLinks(ClientPage c) '''

«FOR link : c.links»
 <a href="«IF(link.url!= null && link.page ==
null)»«link.url»«if(link.parameters.isEmpty())'"'»«ENDIF»«IF(link.url!= null
&& link.page !=
null)»«link.url»/«link.page.name».«link.ext»«if(link.parameters.isEmpty())'"'»
«ENDIF»«IF(link.url== null && link.page !=
null)»«link.page.name».«link.ext»«ENDIF»«IF !(link.parameters.isEmpty())»?«FOR
p : link.parameters SEPARATOR "&"»«p.name»=«p.value»«ENDFOR»"«ENDIF»
 «IF link.target !=null» target="«link.target.name»"«ENDIF»«IF
link.targetframe.literal !='_none'» target="«link.targetframe»"«ENDIF»
>«link.label»</a><br>
«ENDFOR» '''
```

Target

Grammar

```
('target' targetframe=frames)?
('target' target=[Frame])?

Frame: name=ID
             'source' source=[ClientPage|ID]
;
enum frames returns frames:
_none = '_none'|_top = '_top'|_blank = '_blank' |_self = '_self'
|_parent='_parent';
```

TargetLink

Grammar

```
Link: {Link}
      'href' ('url' url=STRING)?
             ('target' targetframe=frames)?
```

Template

```
def renderLinks(ClientPage c) '''
```

FrameContent

Grammar

```
Frame: name=ID
             'source' source=[ClientPage|ID]
```

Template

```
<frame name="«frame.name»" src="«frame.source.name».html">
```

InputElement

Grammar

```
Input: {Input}
      'type'  types+=Type* ;
Type:
      text | radio | checkbox | Submit | password;
text: {text} 'text';
radio: {radio} 'radio' ('checked' checked=ID)? ;
Submit: {Submit} 'submit';
checkbox: {checkbox} 'checkbox';
password: {password} 'password';
```

Template

```
«FOR input : form.inputs»
    «FOR type : input.types»
              «input.label» <input type="«type.eClass.name.toString»"
name="«input.name»" value="«input.value»"><br>
    «ENDFOR»<br>
«ENDFOR»<br>
```

SelectElement

Grammar

```
SelectElement:name=ID
	(multiple='multiple')?
	('size' size=INT)?
	('option' '{' option+=option* '}' )?;
option:
	'option' name=ID
	('value' value=ID)?
	('size' size=INT)?
```

Template

```
«FOR select : form.selectelement»
     <select name= «select.name» «select.multiple»>
     «FOR option : select.option»
     <option value="«option.value»"> «option.name» </option>
     «ENDFOR» /<select> <br>
«ENDFOR»<br>
«FOR select : form.textarea»
     <textarea rows="«select.rows»" cols="«select.cols»">
     «select.text» </textarea>
«ENDFOR»
```

TextAreaElement

Grammar

```
TextArea: {TextArea} name=ID
	'textarea'
	'{'
		('rows' rows=INT)?
		('cols' cols=INT)?
		('text' text=STRING)?'}
```

Template

```
«FOR select : form.textarea»
  <textarea rows="«select.rows»" cols="«select.cols»">
   «select.text»
  </textarea>
«ENDFOR»
```

Automated Code Generation

The goal is to develop a DSL where we have a .xtext grammar file and xtend templates. Taking these files as an input and running Maven would execute xtext, generate all src-gen and xtend-gen files, perform all automated task like compile and build an executable jar. The executable jar would be able to compile end-user webDSL files to generate the HTML files.

```
fragment = generator.GeneratorFragment {
      generateJavaMain = true
}
```

A new file "Main.java" will be created in the generator folder

This main file takes argument to parse an input file, validate or generate code.

POM. xml in the "cloud.mini.wdsl"

```
<artifactId>maven-assembly-plugin</artifactId>
     <version>2.5.5</version>
     <configuration>
         <descriptors>
            <descriptor>jar-with-ecore-model.xml</descriptor>
         </descriptors>
     <descriptorRefs>
         <descriptorRef>jar-with-dependencies</descriptorRef>
     </descriptorRefs>
     <appendAssemblyId>false</appendAssemblyId>
     <archive>
        <manifest>
            <mainClass>cloud.mini.wdsl.generator.Main</mainClass>
        </manifest>
     </archive>
 </configuration>
 <executions>
     <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
 </executions>
</plugin>
```

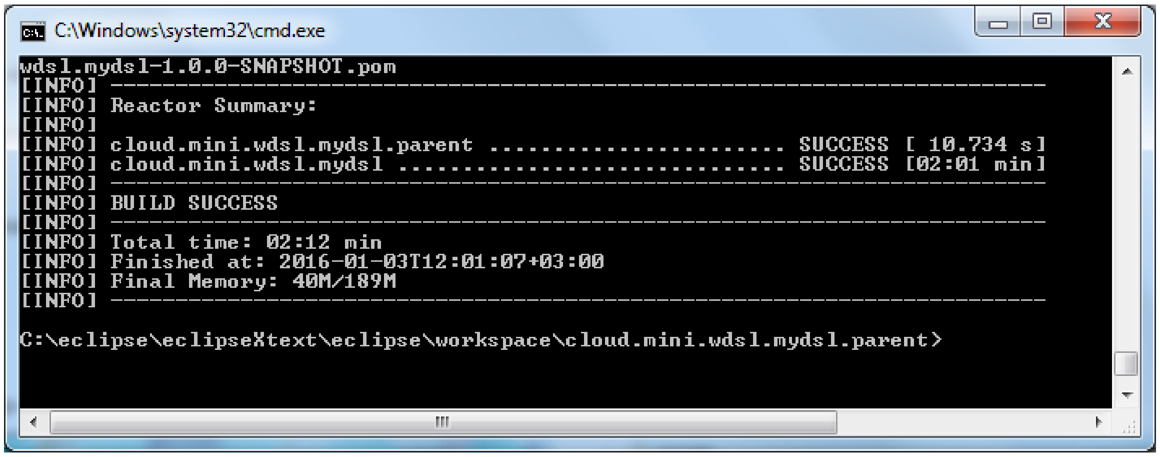
jar-with-ecore-model.xml

```
<assembly xmlns="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.3"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.3 http://maven.apache.org/xsd/assembly-1.1.3.xsd">
       <id>jar-with-ecore-model</id>
       <formats>
              <format>jar</format>
       </formats>
       <includeBaseDirectory>false</includeBaseDirectory>
       <fileSets>
              <fileSet>
                     <outputDirectory>/</outputDirectory>
                     <directory>target/classes</directory>
              </fileSet>
              <fileSet>
                     <outputDirectory>model/generated</outputDirectory>
                     <directory>model/generated</directory>
              </fileSet>
       </fileSets>

</assembly>
```

Compiler generation

The following command is used to generate the compiler.

mvn clean install



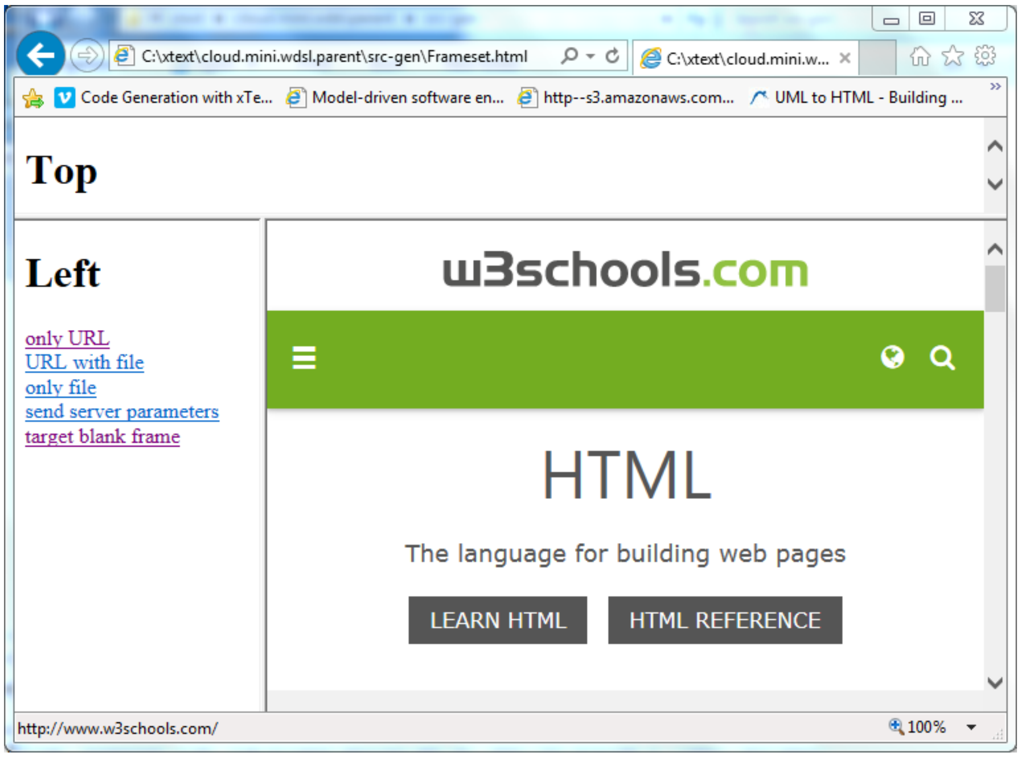
```
wdsl.mydsl-1.0.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] cloud.mini.wdsl.mydsl.parent ........................ SUCCESS [ 10.734 s]
[INFO] cloud.mini.wdsl.mydsl ............................... SUCCESS [02:01 min]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 02:12 min
[INFO] Finished at: 2016-01-03T12:01:07+03:00
[INFO] Final Memory: 40M/189M
[INFO] ------------------------------------------------------------------------

C:\eclipse\eclipseXtext\eclipse\workspace\cloud.mini.wdsl.mydsl.parent>
```

Figure 10: Generation of Compiler

Compiling (Automated)

java -jar cloud.mini.wdsl\target\cloud.mini.wdsl-1.0.0-SNAPSHOT.jar

cloud.mini.wdsl\src\wDsl\Frames.wdsl


C:\xtext\cloud.mini.wdsl.parent\src-gen\Frameset.html

Code Generation with xTe... Model-driven software en... http--s3.amazonaws.com... UML to HTML - Building ...

Top

Left

only URL
URL with file
only file
send server parameters
target blank frame

http://www.w3schools.com/ 100%

Figure 11: Generation of HTML Pages

Part 3 Tutorial on the use of DSL

Attached in a separate developers_guide.pdf

Link: Developers's guide

Part 4 MDD principles, standards, technology

In this project, I have used Model-Driven concepts particularly Domain Specific modeling at different level of abstraction by using concrete and abstract syntax for representation of the models. The model is the primary artifact in the development process and the code generation is automated. The Platform Independent Model (PIM) is used to describe the "Navigational" and "data layer". At the Platform Specific Model (PSM), the transformations are used that are based on rules that correspond to the processing of objects of elements of the model for generation of JAVA, HTML and PHP code.

Focusing on MDD Principles

- I tried to elevate the domain modeling task on a higher level of abstraction and used EMF models to represents the system elements.
- For developing the modeling language I found the Concrete and Abstract Syntax representation useful, used Diagram Definition (DD) specification for Concrete Syntax.
- Defined the Meta-model by analyzing the behavior and attributes and applying different rules.
- Selected the M3 Layer of the MDD i.e. Meta-Meta Model for language engineering and defining the Meta-Language
- Followed the OMG guidelines for defining the Meta-model using 'Meta Object Facility' (MOF Mode) from its language definition stack.
- Chosen Ecore as it is simple conceptual structural modeling language.
- Used incremental and iterative design approach, started by modeling domain analysis followed by language design and language validation.
- Used OCL and Xtend Validations for formalize the constraints of the models.

Used Features

The following are the main features used in the project from model transformation language to implement the mapping from your DSL to the target language

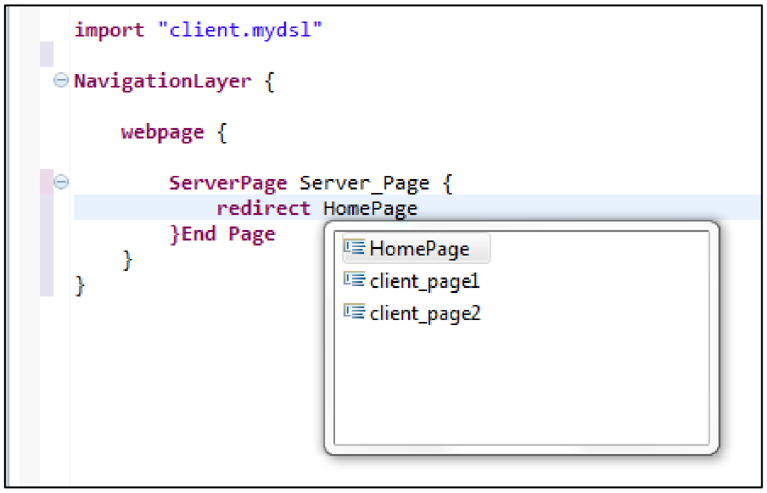
1. Validation

Xtext provides several methods validation techniques and the default validation checks the uniqueness of each IDs per element type. I have chosen this feature to validate in a given navigational layer the elements like Webpage, frames, forms etc. to detect the duplication. It can be easily done by setting the 'NamesAreUniqueValidator' in the mwe2 file.

Custom validation is useful for defining errors or warnings. I used custom validations for duplicate "WebPages" and for checking the relation of "Redirect" and "Form Action" in the WDslValidator class.

2. Referencing and Scoping

In this project there is an option to either use a single WebDSL file or multiple files. This enables the use of “import statements” in the dsl file. The objects defined in the Client Page will be available in the “ServerPage results in compact files. The following shows the use of this feature in the project.

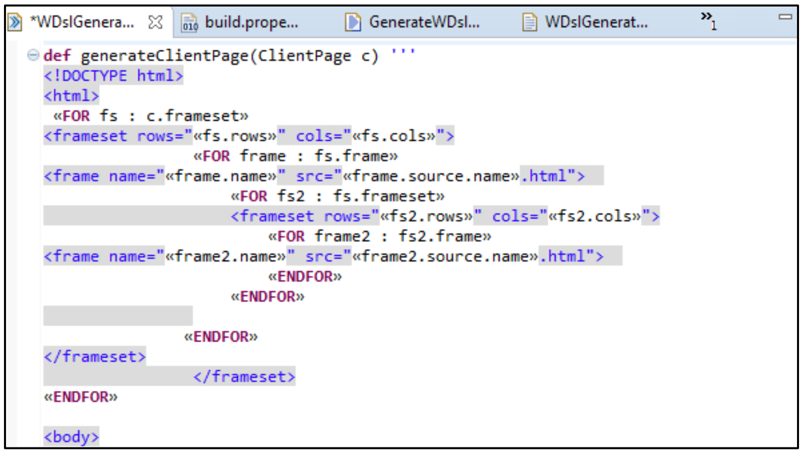


```
import "client.mydsl"

NavigationLayer {

    webpage {

        ServerPage Server_Page {
            redirect HomePage
        }End Page
    }
}
```

Figure 12: Referencing and scoping

3. Template Expressions

For mapping from DSL to the target language, xtend provides an option to create multiple templates and inject the data to process them inside the templates. This can also distinguish which indentation is related to the template and indentation related to the output.



```
def generateClientPage(ClientPage c) '''
<!DOCTYPE html>
<html>
 «FOR fs : c.frameset»
<frameset rows="«fs.rows»" cols="«fs.cols»">
                «FOR frame : fs.frame»
<frame name="«frame.name»" src="«frame.source.name».html">
                    «FOR fs2 : fs.frameset»
                    <frameset rows="«fs2.rows»" cols="«fs2.cols»">
                        «FOR frame2 : fs2.frame»
<frame name="«frame2.name»" src="«frame2.source.name».html">
                        «ENDFOR»
                    «ENDFOR»

               «ENDFOR»
</frameset>
                </frameset>
«ENDFOR»

<body>
```

Figure 13: Template expressions

4. OCLinEcore

OCL is short and easier than Xtend ustom validation, in Xtext the OCL can be applied to the generated ecore models using OCLinEcore editor, to use this option I had to modify the genmodel by setting the property "Model Operation Reflection", we have to be careful using it, If we modify the grammar and generate the artifacts the ecore will be regenerated and OCL part will be removed from the ecore.

Automated Code Generation

This feature provides option to generate the compiler and generate the target code in terminal console. First we have to set the "generateJavaMain = true" in the mwe2. This will result in "Main.java" in the generator folder, this file takes argument to parse an input file, validate and generate code.

The maturity of the technology / Eclipse Modeling Project

The Eclipse Modeling project [3] has become one of the vital projects in Model Driven Software Development. Object Management Group (OMG) introduced MDA in 2001, since then MDA and Eclipse served the software industry by providing open source platforms. The Eclipse Modeling Project provides variety of transformation tasks like Model-to-Model (M2M) and Model-to-Text (M2T). Altogether, these transformation tasks fulfill most of the requirement of MDA, MDSD and DSL. The following is a category of the transformation capabilities relevant within the current scope of the Eclipse Modeling Project.

Model to Model	**Model to Text**
Atlas Transformation Language(ATL)	MOF2Text
Epsilon	Xpand
Acceleo	Java Emitter Templates JET
Query/View/Transformation(QVT)	

Advantages

1. In Model refactoring we can define transformation on a specific model to produce a target model in such a way that the targets instance could be the same as the input mode.

2. We can merge different models and transforming models with multiple input and output models. We can use it for combining aspects of GMF's tooling and mapping.

3. When migrate a model to a future version we can include the version information in the URI. This enables a platform to distinguish different models versions and enables transformations.

4. Using higher order transformation (HOTs) a transformation model itself can be generated or modified by other transformations this is useful in improving internal structure of the model.

5. M2M supports automated generation of target model from source modes and it supports one-one, one-many and many to many transformations.

6. We can use out-place Transformations to create a new model from scratch or In-place Transformations to change specific parts of the model

7. Transformation chains can be used for modeling the orchestration of different model transformations by using conditional branches, loops or programming logics.

Disadvantages

1. When using Model to text transformation the templates can become complex for large projects and might become difficult to maintain if the source evolves.

2. ATL does not provide the option to modify the elements of the input model during transformation.

3. Bi-directional Transformations .i.e. source to target and target to source transformation is only supported in QVT Relational

Alternative Autmation approaches

Gradle would be an alternate tool in automating the implementation, it is build and automation system based on a Groovy-based domain specific language (DSL), plus it has option to integrate Apache Ant and Maven and support incremental build. The latest Xtext 2.9 has both Maven and Gradle build support for automating the application.

The UML-based Web Engineering (UWE [4] uses model-driven development approach and The UWE approach uses recently emerged technologies: model transformation languages like ATL3 and QVT4

One approach used in MDSE book Web applications uses MVC pattern and uses Java (Model), JSF (View), Servlets (Controller) and Apache Tomcat (Server) [5] Graph

Transformation approach is also helpful help in automated implementation as it has a model completion feature that can instantly introduce dynamic elements in the model.

Conclusion

Model-driven Web Engineering (MDWE) is the model-driven concept is useful as there is a frequent innovation of Web technologies and platforms. Several concepts used in UML for Web applications are addressed by using models e.g. for the data, navigation, controlling and presentation layers. An automated framework for transforming models into code with simplified validation control is significant for the software industry. Thanks to Dr. Artor and Mr.Taimoor for assisted me throughout the project and delivered us wonderful skills.

References

[1] Build your language using Xtext and Xtend www.xtext.org

[2] Jim Conallen. Building Web applications with UML. 2000. http://dl.acm.org/citation.cfm?id=326370

[3] The Eclipse Modeling project. www.eclipse.org

[4] Nora Koch. “Transformations Techniques in the Model-Driven Development Process of UWE”. Proc. 2nd Wsh. Model-Driven Web Engineering (MDWE’06), Palo Alto, 2006.

[5] Marco B, Jordi C, Model Driven Software Engineering in Practice, USA 2012