Actividad 03 - DMSS

Leonardo Rossi

May 18, 2014

Contents

1 Metamodelo MM_ModeloDatos **1**

2 Metamodelo HtmlFormMM **2**

3 Transformacion M2M (ATL) **4**

4 Transformacion M2T (Acceleo) **6**

5 Ejemplo de uso de DataModel **7**

5.1 Formulario de Contacto 7

5.2 Ejemplo general 7

6 Codigo HTML Obtenido **9**

6.1 Formulario de Contacto 9

6.2 Ejemplo general 9

Abstract

El objetivo de esta actividad es implementar una cadena de transformaciones M2M (Modelo a Modelo) y M2T (Modelo a Texto). Concretamente, se desea obtener formulario HTML a partir de modelos basados en el meta-modelo denominado ModeloDatos. La primera transformacio se va implementar con ATL y la otra con Acceleo.

Chapter 1

Metamodelo MM_ModeloDatos

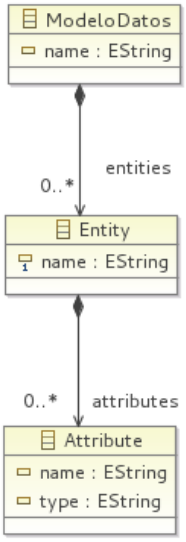

Figure 1.1: Meta-modelo de ModeloDatos

El meta-modelo de figura [1.1] tiene una representación muy genérica de los datos.

Eso tiene un modelo de los datos que hace de raíz y que contiene todos las entidades. A su vez, cada entidad contiene atributos que describe ella.

Chapter 2

Metamodelo HtmlFormMM

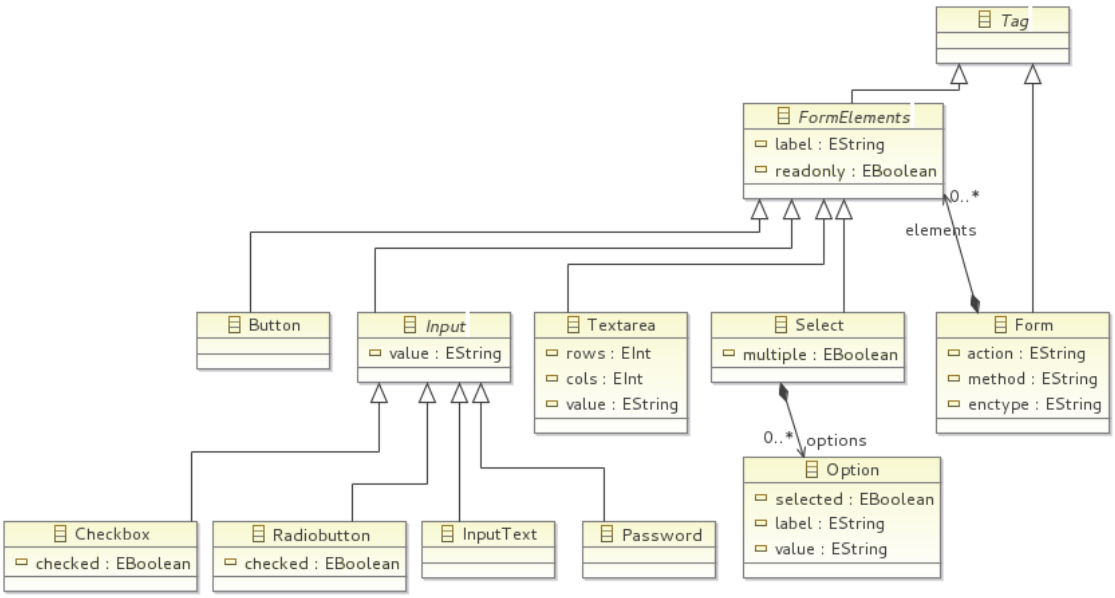

Figure 2.1: Meta-modelo de Formulario HTML

Esto meta-modelo de la figura [2.1] es mas especializado que *ModeloDatos*, descrito previamente. Eso representa un formulario en Html.

Ya que estamos hablando de Html, se ha definido una abstracción mas general de Tag, que todos van a extender.

El elemento que hace de raíz para todos los elemento concreto es *Form*. Eso tiene su atributos:action, que define dónde enviar los datos cuando se envía el formulario, method, que define el método de envío de los datos (get o post) y enctype, que define la forma en que los datos del formulario deben estar codificados cuando se envían.

Cada elemento del formulario es modelado como *FormElements*, que también extiende *Tag*. Eso tiene dos propiedades: *label*, que es un texto que describe el elemento en el formulario y *readonly*, un boolean para definir si es de sola lectura.

El meta-modelo define cuatro elemento que pueden estar en el formulario:

Button - Que es la abstracción de un botón

Input - Que es la abstracción de un elemento <*input*>y que puede ser de varios tipos.

Textarea - Que es la abstracción de un elemento <*textarea*>.

Select - Que es la abstracción de un elemento <*select*>.

El elemento *Button* extiende el *FormElements* sin añadir nada nuevo.

El elemento *Input* es un elemento abstracto que extiende *FormElements*, añadiendo la propiedad *value*. Eso se extendía desde los siguientes elementos:

Checkbox - abstracción de un elemento *Checkbox*. Tiene la propiedad *checked* para definir si esta seleccionado por defecto.

Radiobutton - abstracción de un elemento *Radiobutton*. Tiene la propiedad *checked* para definir si esta seleccionado por defecto.

InputText - abstracción de un elemento de texto.

Password - abstracción de un elemento de texto que contiene una password.

El elemento *Textarea* define propiedades para definir cuantas filas y columnas tienes. También define un texto por defecto.

El elemento *Select* define una propiedad *multiple* para especificar si permite la selección múltiple de los elementos añadidos. Eso contiene diferentes opciones.

Cada *Option* define tres propiedad: un valor *selected* para especificar si esta seleccionado por defecto, una *label* para definir un texto que la describe y un *value* para definir su valor.

Chapter 3

Transformacion M2M (ATL)

El elemento raíz del meta-modelo ModeloDatos [1.1] representa el elemento raíz del meta-modelo de formulario Html [2.1]. Con eso se puede detallar el elemento *Form* del formulario. Para explicitar su propiedades (véase la figura [3.1]), se añade un elemento Entity que contiene un Attribute del tipo "form". Cada Attribute de esta entidad van a determinar las propiedades del *Form*.

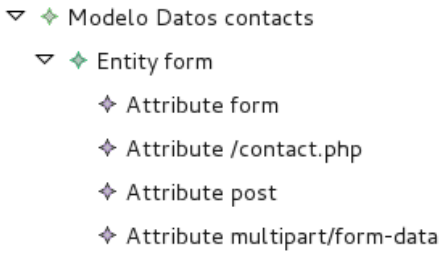

Figure 3.1: Definición de un formulario y su propiedades en el meta-modelo de origen.

Después, cada elemento que extiende FormElements del formulario Html corresponde a una Entidad del primero meta-modelo. A través de los Attribute en el primero meta-modelo, se definen todas la propiedades del elemento del formulario en el meta-modelo de destino.

Es figura [3.2] se puede ver como se transforma un elemento del modelo de origen en un elemento en el modelo de destino. El atributo de tipo "type" define el tipo de elemento que se entiende crear en el meta-modelo de salida. Todos los otros atributos van a definir las propiedades del elemento del formulario.

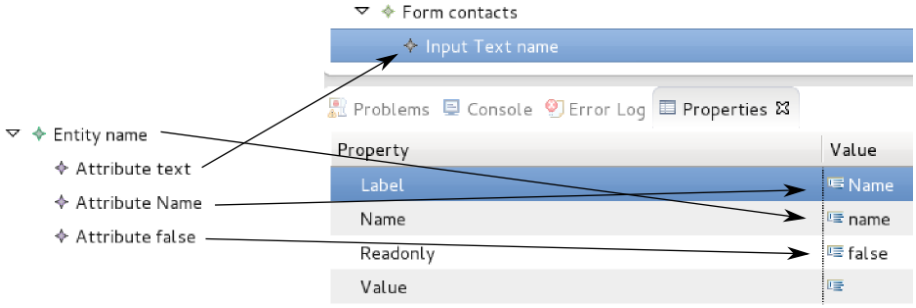

Figure 3.2: Correspondencia de una entidad en el modelo origen en un elemento *InputText* del formulario Html

Al final, vamos a hablar del elemento *Select* en el meta-modelo de destino, porque es un poco especial: tiene su propiedades como otros elementos del formulario, pero tiene también añadidos elementos *Option*. En figura [3.3] se puede ver un ejemplo de como esta modelado en el meta-modelo de origen y de como es la correspondencia con el meta-modelo de destino.

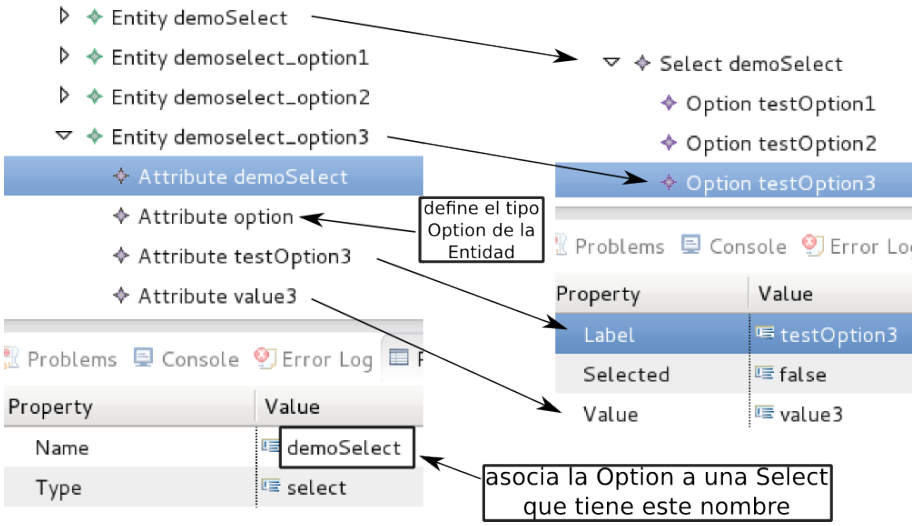

Figure 3.3: Correspondencia de entidades en el modelo origen en elementos *Select* y *Option* del formulario Html

Chapter 4

Transformacion M2T (Acceleo)

En figura [4.1] se puede ver como esta definido en el template general el formulario. Esta compuesto de un <*Form*>y después todos los elementos. Por cada elemento llama la función *generateHtmlCode()* para generar el correspondiente código *Html*.

```
<body>
  <h3>[aForm.name/]</h3>
  <form enctype="[aForm.enctype/]" class="[aForm.generateId()/]"
     id="form" method="[aForm.method/]" name="[aForm.name/]" action="[aForm.action/]">
    [aForm.elements.generateHtmlCode()/]
  </form>
</body>
```

Figure 4.1: Template del formulario Html

Cada tipo de elemento que extiende el tipo *FormElements*, va a implementar esta función para generar su correspondiente código *Html*.

Vamos a describir el elemento mas complejo: *Select*.

En figura [4.2] se puede ver que define una etiqueta y después el código del <*select*>. Dentro el <*select*>va a llamar una función especifica para sus elementos *Option* que genera el código *Html* para cada <*option*>.

```
[template public generateHtmlCode(element : Select) post(trim() + '\n<br>\n')]
[element.label/]
  <select name="[element.name/]" [if (element.multiple)]multiple[/if]>
    [element.options.generateHtmlCode(element.readonly)/]
  </select>
[/template]

[template public generateHtmlCode(element : Option, readonly: Boolean) post(trim() + '\n<br>\n')]
<option value="[element.value/]" [if (readonly)]disabled[/if]
   [if (element.selected)]selected="selected"[/if]>[element.label/]</option>
[/template]
```

Figure 4.2: Template del elemento <*select*>

Chapter 5

Ejemplo de uso de DataModel

5.1 Formulario de Contacto

En figura [5.1] se puede ver el primero ejemplo: la transformación *M2M* de un modelo de datos en un formulario *Html* y después una transformación *M2T* en código *Html*.

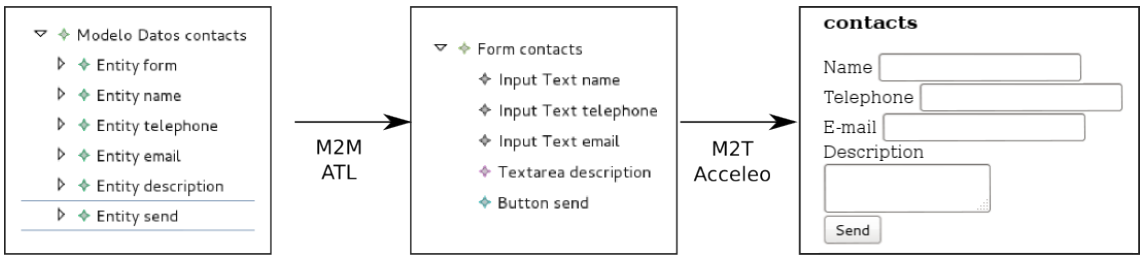

Figure 5.1: Primero ejemplo: transformación de un formulario de contacto

5.2 Ejemplo general

En figura [5.2] se puede ver un ejemplo que contiene todos los elementos definidos y vas a comprobar que todas las transformaciones funcionan bien.

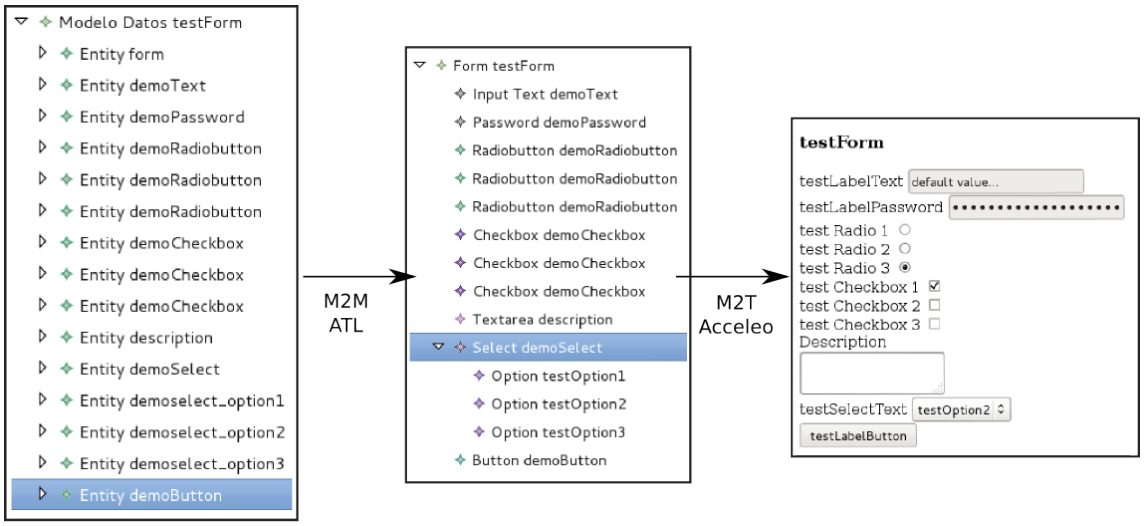

Figure 5.2: Segundo ejemplo: transformación de un formulario que define todos los elementos

Chapter 6

Codigo HTML Obtenido

6.1 Formulario de Contacto

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>contacts</title>
</head>
<body>
  <h3>contacts</h3>
  <form enctype="multipart/form-data" class="form-contacts" id="form" method="post" name="contacts" action="/contact.php">
    Name <input type="text" class="form-name" value="" name="name"  >
    <br>
    Telephone <input type="text" class="form-telephone" value="" name="telephone"  >
    <br>
    E-mail <input type="text" class="form-email" value="" name="email"  >
    <br>
    Description <br>
      <textarea type="text" class="form-description" name="description"  ></textarea>
    <br>
    <input type="submit" class="form-send" value="Send" name="send"  >
    <br>

  </form>
</body>
</html>
```

Figure 6.1: Primero ejemplo: código HTML

6.2 Ejemplo general

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>testForm</title>
</head>
<body>
  <h3>testForm</h3>
  <form enctype="multipart/form-data" class="form-testForm" id="form" method="post" name="testForm" action="/contact.php">
    testLabelText <input type="text" class="form-demoText" value="default value..." name="demoText" readonly >
    <br>
    testLabelPassword <input type="password" class="form-demoPassword" value="default password value..." name="demoPassword" readonly >
    <br>
    test Radio  1 <input type="radio" class="form-demoRadiobutton" value="default radio1 value..." name="demoRadiobutton" disabled  >
    <br>
    test Radio  2 <input type="radio" class="form-demoRadiobutton" value="default radio2 value..." name="demoRadiobutton"   >
    <br>
    test Radio  3 <input type="radio" class="form-demoRadiobutton" value="default radio3 value..." name="demoRadiobutton"  checked >
    <br>
    test Checkbox  1 <input type="checkbox" class="form-demoCheckbox" value="default checkbox1 value..." name="demoCheckbox"  checked >
    <br>
    test Checkbox  2 <input type="checkbox" class="form-demoCheckbox" value="default checkbox2 value..." name="demoCheckbox"   >
    <br>
    test Checkbox  3 <input type="checkbox" class="form-demoCheckbox" value="default checkbox3 value..." name="demoCheckbox" disabled  >
    <br>
    Description <br>
      <textarea type="text" class="form-description" name="description"  ></textarea>
    <br>
    testSelectText
      <select name="demoSelect" >
        <option value="value1"  >testOption1</option>
        <br>
        <option value="value2"  selected="selected">testOption2</option>
        <br>
        <option value="value3"  >testOption3</option>
        <br>

      </select>
    <br>
    <input type="submit" class="form-demoButton" value="testLabelButton" name="demoButton" readonly >
    <br>

  </form>
</body>
</html>
```

Figure 6.2: Segundo ejemplo: código HTML