CS 297 Report
Paperful to Paperless Office Forms Integration Framework

Madhuri Potu
(madhuri5006@yahoo.com)

Advisor: Dr. Chris Pollett
December 15, 2003

Table of Contents

Abstract:

Filling Forms, any type of forms, office forms, web transaction forms, is the most important part of our daily routine. For example, consider web transactions, forms have become their most important part, enabling them to communicate with the users. HTML was/is our solution to web applications. Advancements in technology have replaced HTML by a more reliable, independent technology called XForms. Currently, XForms are still in the developmental stage, but once they are able to support all types of browsers they might replace HTML.

The main goal of this project is to produce a framework that contains some templates of sample office forms and can identify the forms associated with any person in a database. A person working on a new form can choose from the existing templates and customize it as required. A database of forms and the details of the people associated with it will be maintained. If a new form is requested of a person for which, data has already been successfully gathered by another, then our system will be clever enough to modify the current form so that when it is printed it only requests a fixed ID together with the data that has not been previously requested.

We intend our set-up to be flexible enough so that if two departments of a company do mass mailings to request data of the same individual. Then the data requested will be still be individualized to a given person. Data can also collected by either paper or web-based forms in a seemless manner. The framework that we will develop in this project will communicate with a database using an XML format with respect to an XML schema for that specifies what the data looks like. Forms will be created and stored using the new XML Forms language.

Introduction:

Most of the user transactions that take place on the web are entirely dependent on forms. For example even the smallest transaction like conducting a survey is done through forms. Currently most of the applications use HTML. However, there are some applications that put a limit on the HTML forms, making it difficult for the user to communicate. XForms is a new technology that is considered to be the future of web based forms. These XForms can help us overcome the limitations of HTML forms, various advantages of XForms are listed below. The main goal of this project is to use these XForms to develop a framework that contains templates of sample Forms and which lets users customize, edit and create new Forms.

The following are the main features of XForms:

1. The data and logic in XForms are independent of the presentation, i.e., the user interaction with the web forms is separated form the data in the form.
2. The data in XML is described using XML -- this integration with XML was not possible with HTML -- even for validation.
3. XForms also send the data using XML i.e., data transmission over the internet.
4. XForms are not dependent on a particular device, eg: can be used on desktop, PDAs, paper etc. This is possible because the data is independent of presentation.
5. It reduces scripting because all the validations and calculations are handled by the forms themselves.

6. Re-use of Schemas -- If we make changes to the logic of the Form, then the constraints need not be changed because of the changes in the validation rules

i. Overview of XForms

The following is an illustration of XForms model:

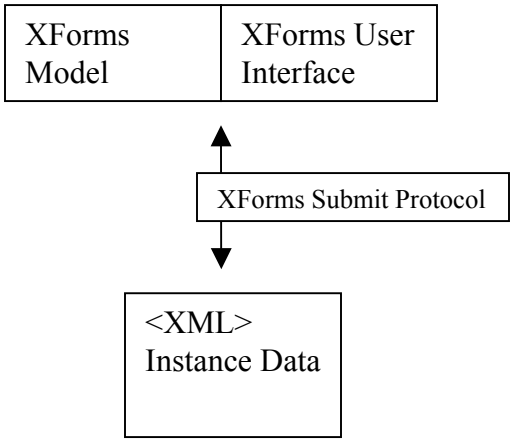

Figure 1: XForms Model
Source: http://www.w3.org/MarkUp/Forms/

XForms are comprised of three parts: User Interface, XForms Model, and Instance Data. The XForms model can work with any type of User interface – WML, XForms User Interface, XHTML or Proprietary User Interfaces. The data in the form is described by the XForms Model. The presentation is separated from the data and described in a separate section. The XForms User Interface has a set of controls that can be used in any type of device or platform. There are binding attributes to bind these controls with the XForms Model. e.g: 'reference' attribute. The Instance Data is the data

submitted to or collected by the forms; this data can be sent or received using the XForms Submit Protocol and can be directly submitted by the XML processor.

The main objective of this project is to develop a framework that contains a set of form templates. These can be changed according to a user's needs and new forms can be created using this framework. A database of the forms and the people associated with the forms will be developed for this project. The framework will be developed will communicate with the database using an XML format with respect to an XML schema, that specifies what the data looks like. Forms will be created and stored using XForms.

The following are the main goals for this semester:

1. To research on usability of forms. Read up on XML Forms and produce an example document.
2. To write a GUI program that has some sample template forms and allows you to create new forms
3. To write an XML Schema to describe the data associated with the forms
4. To match the forms to the people associated with them.

The GUI program developed in this project has a framework that contains a sample templates of XForms and can be customized according to the needs of the user. It can also create new XForms. After the XForms have been created, a generic XML Schema will be generated to describe the data associated in all the forms. An XSLT style sheet is used to display the form. An XForm contains the following: a

‘layout’ tag to describe the style of XForm, a ‘personinfo’ tag to match the forms to the people associated with them, and an ‘xformdocument’ tag to describe the XForms Document.

Future Work: Enhancements will be made to the Framework so that it is an application that manages the forms in a database, displays and customizes the forms. This set-up intended to be flexible enough so that if two departments of a company do mass mailings to request data of the same individual, then the data requested will still be individualized to a given person. We need to figure out how and when to merge forms. The application will be compared to other ways of maintaining data.

Deliverable 1:
i. XForms Example

Deliverable 1 is to research about XForms and produce an example document. In this deliverable an example XForm document is developed --- An entry form for a Dog Show. Given below is an example XForm document:

```
<head>
   <xforms:model>
      <xforms:instance>
            </name>
      </xforms:instance>
   </xforms:model>
</head>

<body>
      <xforms:input ref="name">
      <xforms:label>First Name:</xforms:label>
      </xforms:input>
 </body>
```

The ‘xforms:model’ element is used to represent the information needed in a form (in the above example: name). All this information is contained within the head tag. The content part is separated from the presentation making it easier for the forms to get only the data that is needed. XForms contain a set of controls that are described in the body section, these controls are bound to the ‘xforms:model’ element via the ‘ref’ attribute.

In the above example we described a form control called “xforms:input”, this control describes an input box and the ‘xforms:label’ associated with it is the label of the input box. Since we will only be representing text boxes, combo boxes and submit buttons in this project, we will deal only with those controls. The ‘xforms:selectone’, ‘xforms:choices’, ‘xforms:item’ are the controls used to represent combo boxes and the ‘xform:submit’ is the control used for the Submit button.

Here is an example showing the form control for combo boxes (choice for gender male or female):

```
<xform:selectOne xform="entryform" ref="gender">
        <xform:caption>Sex</xform:caption>
        <xform:choices>
                <xform:item value="male">
                 <xform:caption>Male</xform:caption>
                 </xform:item>
                <xform:item value="female">
                <xform:caption>Female</xform:caption>
                 </xform:item>
        </xform:choices>
 </xform:selectOne>
```

The ‘xforms:selectone’ does not just represent combo boxes, it can either be check boxes, radio buttons or combo boxes, depending on the person choosing it. ‘xforms:selectone’ only describes that we can choose any one among the given choices

(combo boxes or check boxes or radio buttons), giving the user the freedom to select any type of representation. An XForms Processor is used to validate the data and send it directly to the application. It is a built in processor enabled in browsers.

With the data separated from the presentation, XForms can be managed efficiently, making them device or platform independent. In the future all browsers will be enabled with the XForms plugin making the users not feel the difference except they will find more efficient web forms. This deliverable is for learning about XForms, their uses and how to develop them. This information is used in the next deliverable to develop a GUI program that can create XForms.

Deliverable 2:

Deliverable 2's purpose was to create a GUI program that has some sample XForms templates and that allows you to create new XForms documents. I developed a program called "MenuForm.java" that can display any XForm that contains textfields, comboboxes and submit buttons; it can also create new XForms and display them. The GUI Program displays four buttons: 'Form1', 'Form2', 'Form3' and 'New Form'. 'Form1', 'Form2' and 'Form3' are used to display sample template forms. These forms display textfields, comboboxes and submit buttons. The 'New Form' button helps us create new Form. New textfields, comboboxes and submit buttons can be added to the Form. It contains a menu and a SubmitForm button. The menu can be used to choose between adding a new textfield or a combo box. Once a form has been created, if Submit

button is clicked it displays a textfield to enter the name of the form. In this way a new XForm document can be created using the 'MenuForm.java' program.

In order to display and read the xforms documents simple file parsing is done using the SAX Parser. The source code below shows simple parsing:

```
try {
        InputSource input = Resolver.createInputSource (new File (form));
        XmlDocument doc = XmlDocument.createXmlDocument (input, false);
        doc.getDocumentElement ().normalize ();
        java.io.StringWriter fileData = new java.io.StringWriter();
        doc.write (fileData);
  } catch (SAXParseException err) {}catch (SAXException er) {} catch (Throwable t) {}
```

As seen above, an XML document is read (InputSource), parsed, and displayed using the write command. In the previous section we have seen how the GUI program can open existing XForms documents and create new XForms. The code above is used to open and read an existing XForm document.

The following is some sample code for creating a new XForm – it creates a document and then creates the root element – here the root element is 'html':

```
doc1 = new XmlDocument ();
root = (Element) doc1.createElement ("html");
root.setAttribute("xmlns:xform", "http://www.w3.org/2002/01/xforms");
doc1.appendChild(root);
```

Let us take a look at creating one of the controls while creating a New XForm document. The source code is given below:

```
text1.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e) {
```

```
            text = text1.getText();
            Element child = (Element)doc1.createElement("xform:input");
            root.appendChild(child);
            child.setAttribute("value", text);
            Text child1 = (Text)doc1.createTextNode(text);
            child.appendChild(child1);
          try{
            doc1.write(System.out);
          }catch (IOException io){}
      }
});
```

Once the request for creating a textfield is made, 'xform:input' is created, appended to the root of the document, and its value is set to the text label given by us. Text1 is the textfield where the name of 'xform:input' is entered. 'root' is the root of the XForm document created. This is the first element created. Later when the child nodes are created, they are appended to the 'root' using the command 'root.appendChild(child)'. 'Doc.createTextNode' creates the label or the attribute of the child described above. In this way the other controls 'combo box' and 'submit' are created and appended to the root so that a complete XForms Document can be created.

Deliverables 3 and 4:

In Deliverables 3 and 4, we develop an XML Schema to describe the data associated with forms and match the forms to the people associated with them. In this deliverable we developed an XML Schema (form.xsd), XForm documents and an XSLT stylesheet to transform the xform document. The XSLT stylesheet (formstylesheet.xsl)

can be used to transform any type of XForm document (in this deliverable form2.xml, form3.xml) that contain textfields, radio buttons and submit button.

The basic idea behind the deliverable can be shown in the diagram given below:

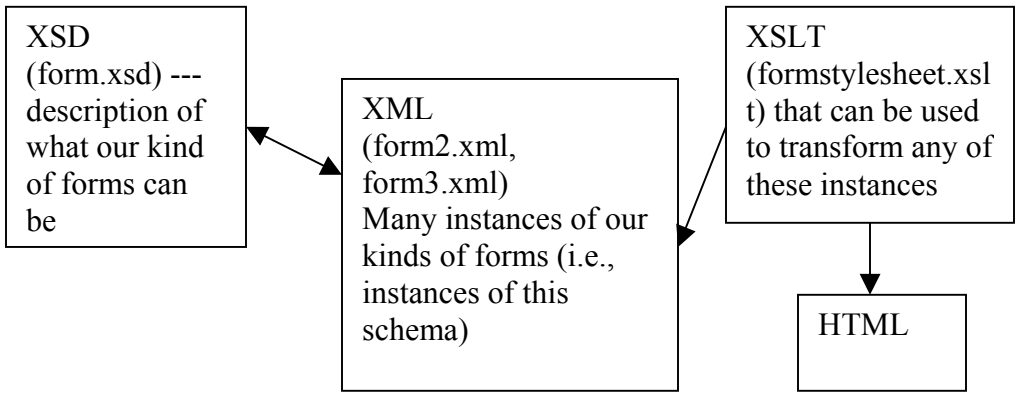

Figure 2: Diagram showing the idea behind Deliverables 3 and 4

i. XML Schema

XML Schema (form.xsd) describes the data associated with the forms, any type of forms with the following layout:

```
<layout>
      <inputbox>
<---! This element can be used to describe how the text box looks, the color etc. --->
      </inputbox>


       <fontsize>
<---! This element is also for the style of the form , to describe the font size and color of font ---->
       </fontsize>
</layout>
```

<---! This layout element is to describe the style of the form --->

```
<personinfo>
            <department/>
```

```
            <id/>
</personinfo>
```

<---! The personinfo tag is to describe the person info so that we can match the forms associated with them – deliverable 4. ---->

```
<formdocument>
        <model>
                <instance>
                ------
                </instance>
        </model>
        <controlgroup>
    -------
        </controlgroup>
</formdocument>
```

<---! FormDocument is to describe an XForm document. It contains model, instance and a control group to describe the form controls --->

ii. XSLT Stylesheet

The XSLT stylesheet is used to transform the XForm documents. In this deliverable formstylesheet.xsl is the XSLT used to transform any type of the XForm documents with above described layout. Given below is a screenshot of form2.xml after being transformed using the formstylesheet.xsl

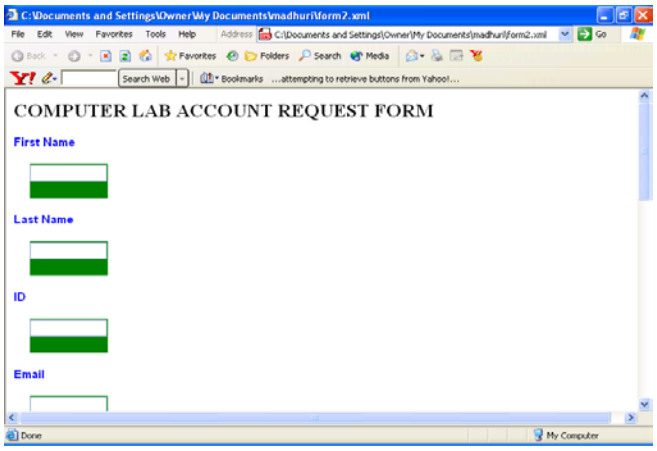

Figure 3: Screenshot of transformation of XForm using XSLT stylesheet

Conclusion:

XForms are the future of Web Forms; they are the successors of HTML forms. The user transactions are entirely dependent on the forms in some applications. HTML forms have been making all these transactions possible. XForms have many benefits over the existing web forms like: they separate the data and the presentation, uses XML to describe the data, less scripting, device independent, platform independence, and allows reuse.

In this project we will be developing an application that contains a framework to display XForms and create new XForms. A database of forms and the details of the people associated with it will be maintained. This semester we developed a GUI program, part of the application that can display sample XForms and create new XForms. We also developed an XML Schema that describes the data associated with all the forms. An XSLT stylesheet (generic) was developed for transforming these XForms.

The framework that we will develop in this project will communicate with the database using an XML format with respect to an XML schema for that specifies what the data looks like. We intend our set-up to be flexible enough so that if two departments of a company do mass mailings to request data of the same individual, then the data requested will be still be individualized to a given person.

References:

[HM02] Harold,R.E.,&, S.W. XML in a Nutshell (2nd ed.). O'Reilly. 2003.

[TM03] Thierry Michel. XForms - The Next Generation of Web Forms. W3C. http://www.w3.org/MarkUp/Forms/. 2003.

[ST03] C. M. Sperberg-McQueen and Henry Thompson. XML Schema. W3C. http://www.w3.org/XML/Schema. 2003.

[S01] Suciu, D. On Database Theory and XML.SIGMOD Record (8), pp.39--45. 2001.

[SYU99] T. Shimura, M. Yoshikawa, S. Uemura. Storage and Retrieval of XML Documents using Object-Relational Databases. Proceedings DEXA Conference Florence, Italy, 1999. pp 206--217.

[SKCC01] H. Su, D. Kramer, L. Chen, K. Claypool, and E. A. Rundensteiner XEM: Managing the evolution of XML Documents. Eleventh International Workshop on Research Issues in Data Engineering (RIDE 2001). Heidelberg, Germany, April 1-2, 2001.

[TVBSSZ02] Tatarinov, I., Viglas, D.S., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C. (2002,June). Storing and Querying Ordered XML using a Relational Database System. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 204--215.