XForms - The Next Generation of Internet Interfaces

Christoph Hofmann
chofmann@rz.fh-augsburg.de

Gregor Liebermann
gregor.liebermann@fh-augsburg.de

Augsburg University of Applied Sciences
Dept. of Computer Science
Baumgartnerstr. 16
86161 Augsburg

ABSTRACT: XForms stand for the new standard in HTML development. This paper wants to give a first introduction into this new technique. It addresses to web-developers, who have already experience with the XML language but don’t yet know XForms and want to get a overview about this new technique. What is shown first are the main benefits of Xforms and the big lacks of actual HTML forms, which haven’t changed the last ten years. One main innovation is the separation between data and presentation layer. This technique is mainly based on the MVC pattern which will be explained in more detail. Next, the paper gives a little insight into the new programming elements and procedures. One of the new improvements is, that the design of an input element depends of the chosen data type. But the article also shows the practical aspects of XForms. Here the paper gives an introduction of two XForms implementations, the main XForms processors and browsers. At last two XForms editors are presented; a Java and an IDE based one.

1. Introduction

In the today’s working on the web, the use of forms on websites is taken for granted. Interactive tasks in hypertext-applications would be unthinkable without them. HTML forms offer a standardized interaction interface between users and servers in the background.

Today, about ten years after the W3C has released forms as an integral part of HTML, the web has become a place where millions of complex transactions starting to exceed the limitations of HTML forms. Not only browsers on PCs or workstations are used as end devices for hypertext applications, but the web-usage is more and more distributed by other devices, such as PDAs, Set-Top-Boxes, mobiles and, so on.

Instead of modifying the existing forms language that is part of HTML, the W3C saw the need of a new approach. In October 2003 the next generation of web forms, XForms 1.0 have reached the status of a W3C recommendation. The aim of XForms is to displace the traditional forms as defined in HTML and XHTML.
2. Motivation

Because HTML forms haven’t really changed the last ten years, they still consist of “name and value” pairs.

For example: `<input type="radio" name="as" value="cash" />

HTML forms have the following limitations:

- no validation and calculation server-sided
- no loops
- no data types supported
- tied to single language: HTML
- missing separated data model
- each form is bound to one single HTML page
- limited to one request/response cycles (only one submit “button”)"

If a programmer needs more complex actions he only can achieve this by using scripts (JavaScript), which reduce accessibility and maintenance. XForms break up with these limitations and offer a flexible technology, which will be explained in the next chapters.

The main common aspects of the Xforms technology are:

**Reusability**: A form developed can easily be reused in other applications, because its based on XML.

**Platform independency**: XForms provide an abstract visual presentation that enables the adaption to different end devices. This design is totally independent from the data model.

**Universality**: By separating the data from the presentation, forms are portable for usage in other applications, which can use the forms for different purposes.

The key to the power of XForms is that they can work with XML. This feature is accomplished through binding, the connection between the XForms model and the user interface.

3. What are XForms?

As mentioned before the XForms specification was motivated by the demand for extensible means to provide richer, more dynamic forms for HTML documents, while keeping in mind that web forms should be easier and faster to create. XForms extend the XHTML module (XML-compliant form of HTML). The XForms specification introduces several tags, elements and attributes.

Because of this, the learning of XForms is more a matter of understanding these individual elements and how they interrelate. As a benefit you can replace several tasks declaratively that would normally require complicated scripting, by putting the right elements in place. Browsers with an integrated XForms processor can render XForms documents on the client. By integrating events, input-validation and the bigger functional range, the client can already execute a lot of tasks.

This helps that developers can reduce accesses on the servers and so can improve the response time associated with server round-trips to update the web form. (What is also possible is a server side
XForms interpreter that translates the XForms documents into HTML.) Additionally you don’t have to contact a server, if you form stretches across more websites. A big advantage is that XForms strictly separate purpose (data processing logic) from presentation.

Most importantly, XForms splits a form's data model, view, and controller. These parts are further decomposed into even finer reusable levels. For instance, XForms differentiates a form's view into its presentation and purpose. Thus you can easily reuse different contexts as well as displaying them on different output devices.

Last but not least XForms bring the benefit of XML, because the entered data become transferred in XML. This enables to simply check the correctness and the validity by using XML Schema. XForms don’t represent an independent document type, but developers embed them in other documents, like XHTML or SVG.

4. Architecture

4.1 Overview

Like many new technologies the XForms architecture is based on the MVC pattern. Model-View-Controller (MVC) is a “classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC pattern hinges on a clean separation of objects into one of three categories — models for maintaining data, views for displaying all or a portion of the data and controllers for handling events that affect the model or view(s).”[ENo02] A key feature of this MVC decomposition is a clear separation of the model from its final presentation.

4.2 Using MVC

By using the MVC pattern XForms will be portable from one server to another, serve different clients, provide rich validation and calculation and allow complex logic without a single line of scripting code. Furthermore XForms is maintainable through clear separation of logic, data and presentation which allows a role based development process. Because of the experiences by developing today’s web applications and the need to deliver such applications to an ever-increasing number of end-user devices, the overall XForms architecture has been divided into the following components:

- **Model** – Describes the data model of the form (structure and data types). All non presentational aspects of a web application are encapsulated by the XForms’ data model. The data model integrates an XML instance that collects the user input, the constraints used to validate this input and the necessary metadata about how this user input should be communicated to the Web server. It is even possible to construct dependencies between different data.
- **User Interface** – provides visual illustration for different devices. XForms defines a user interface vocabulary to create rich user interfaces. The user interface vocabulary is designed to capture the underlying intent of the user interaction, rather than its final presentation on any given device or in any specific modality. This makes it possible to deliver XForms-based Web applications to different devices and modalities.
- **Instance-Data** – represents the submitted data serialized in XML. This allows the web application author to specify where, how, and what pieces of the data should be submitted to the web server. It also permits the application developer to specify what actions to take upon receiving a response from the server. It is possible to use more than one submit “buttons”.

4.3 An XForms code example

The W3C XForms specification defines two XML dialects. One of them defines the **form model** and the other describes how the form will be displayed to the end user (**form control**) [Orb03]. This chapter shows how the three XForms elements, described in 4.2, are working together. In the following example the user has to fill out a simple form. This example is very easy to understand and should only show the separation of the different XForms components.

We want to collect four values:
- the “status” of the student (Master, Bachelor, no Student)
- the Name of the Student
- the ID Number of the Student
- the birthday of the student

**XForms Model**

This can be represented in the XForms model element. This element would typically be contained within the head section of an XHTML Site:

```xml
<head>
<!-- define the form model -->
<xforms:model id="register">                         <!-open XFORMs model -->
<xforms:instance>                           <!-one XFORMs model instance -->
    <student>                    <!- model elements | "how" the data will be collected -->
        <status/>
        <name/>
        <number/>
        <birthday/>
    </student>
</xforms:instance>
</xforms:model>                       <!—close XFORMs model -->
</head>
<body>
....
```

The XML section above describes the collection of three information pieces (status, name, number, birthday), and that they will be submitted using the URL in the action attribute. The XForms model is surrounded by the “<xforms:model>” tag and one model instance is defined via “<xforms:instance>”. After setting up the “skeleton” of the XForm the next step is to implement the controls, which are the interface for the user.

**XForms Controls**

XForms defines a set of form controls for “standard” use, like collecting names and numbers. The form controls are bound to the model element via a “ref” attribute, which is introduced by the XForms specification. In addition to the model, you need to define:
- What controls are to be used? For example a radio button for the status and text field for the number.
- How those controls are bound to the instance, such as binding the number text field with the number element of the instance.

The form controls typically appear within the body section, as the following code-fragment shows:

```xml
<body>
  <xforms:select1 ref="status">                 <!-opens the selection | the user can choose (radio) -->
    <xforms:label>Who are you?</xforms:label>             <!-the text in front of the selection -->
    <xforms:item>             <!-selection item one -->
      <xforms:label>Master</xforms:label>
      <xforms:value>master</xforms:value>
    </xforms:item>
    <xforms:item>          <!-selection item two -->
      <xforms:label>Bachelor</xforms:label>
      <xforms:value>bachelor</xforms:value>
    </xforms:item>
    <xforms:item>           <!-selection item tree -->
      <xforms:label>no Student</xforms:label>
      <xforms:value>none</xforms:value>
    </xforms:item>
  </xforms:select1>                                <!-closes the selection -->

  <xforms:input ref="name">       <!-input element | the user can type in his name -->
    <xforms:label>Name:</xforms:label>
  </xforms:input>

  <xforms:input ref="number">       <!-input element | the user can type in a number -->
    <xforms:label>ID Number:</xforms:label>
  </xforms:input>

  <xforms:input ref="birthday">         <!-input element | the user can choose a date -->
    <xforms:label>Your birthday:</xforms:label>
  </xforms:input>

  <xforms:submit submission="submit">            <!-the SUBMIT "button" -->
    <xforms:label>Submit</xforms:label>
  </xforms:submit>
</body>

As shown in this code section, there are different types of XForms elements to build an XForms controller. The “<xforms:label>TEXT</xforms:label>” element describes the content, which stands in the user interface in front of an XForms control item (e.g. the word “name” in front of an input field). Another aspect is that important tags are described with comments.
XForms Processor
An XForms processor built into the browser will be responsible for submitting the XForms data to a target. The data will be submitted as XML data and would look like this, if the user chooses the status “master”:

```xml
<student>  <!-- the structured data | “in which structure” will the data be transferred -->
    <status>master</status>
    <name>Peter O’Toole</name>
    <number>1234567890123456</number>
    <birthday>1979-07-12</expiry>
</student>
```

Of course there is the possibility to validate the XML parts with a schema, for example to check the length of the number, if it’s a correct ID.

The example shows the benefit that the usage of radio buttons is not hard coded in the user interface. Furthermore, form controls always have labels which are associated with them and there is no need for an enclosing form element.

5. The XForms language

5.1 Input elements

Within the document body web authors can put their actual input elements at any place and don’t have to embed them in `<form>`-tags, as in HTML.[Ric04] The only thing, developers have to determine is the way of input, and not the appearance. The design of an input element depends on the data type. Thus, an `<input>` element for a string can be displayed traditionally, whereas a calendar-control is used for data entries. Listing 1 shows a list of XForm specific input elements. Several new form elements have been introduced. One is the “range” element, which allows the selection of a value out of a specific area, e.g. in shape of a throttle.[Dub03] In addition to allowing file system uploads, this form-control specifies connections to scanners, digital cameras, microphones, pens and digitizers, and on platforms with sufficient hardware support even video, 3d and other exotic input devices.

Listing 1 shows an extract of important XForms input elements. Some already exist in a certain way in HTML but there are several new items which offer a lot more functionality.

<table>
<thead>
<tr>
<th>XForms form control</th>
<th>Closest XHTML equivalent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>N/A</td>
<td>For inline display of any instance data</td>
</tr>
<tr>
<td>Range</td>
<td>N/A</td>
<td>For smooth &quot;volume control&quot; selection of a value. Indicates the minimal and maximal values of the inputs, plus the interval sizes</td>
</tr>
<tr>
<td>Upload</td>
<td><code>&lt;input type=&quot;file&quot;&gt;</code></td>
<td>For upload of file or device data</td>
</tr>
<tr>
<td>Select1</td>
<td><code>&lt;select&gt;</code> or multiple <code>&lt;input type=&quot;radio&quot;&gt;</code></td>
<td>For selection of just one option among several</td>
</tr>
<tr>
<td>Trigger</td>
<td>N/A</td>
<td>Possibility to activate a defined process</td>
</tr>
<tr>
<td>Help</td>
<td>N/A</td>
<td>A window that opens and which can be moved and has to be closed actively. The User can go on working while the help window is opened</td>
</tr>
<tr>
<td>Alert</td>
<td>N/A</td>
<td>Opens a hint-window. The user only can go, if he closes the window</td>
</tr>
</tbody>
</table>

Listing 1: Overview of some new XForm elements

Source: [Pru03][XIn04]
5.2 Properties

You can assign different properties to any object within the XForm model, where the data and objects are processed. The XForms 1.0 standard defines the following:[XIn04],[Mas04]

**required**
If it is set to true(), it describes whether a value is required or can be left blank. The developer has to proof that required data is visible.

**relevant**
Non-relevant instance data nodes are not serialized for submission

**readonly**
When set to true, it is a signal for the XForms processor, not to allow changes of the instance data.

**calculate**
Provides a XPath expression to calculate mathematical operations, which can include mathematical operators and function calls.

**constraint**
Specifies a full XPath expression that needs to be satisfied before the instance data submission is considered valid and before submission can succeed.

**type**
Associates a Schema data type with the attached instance data nodes.

5.3 Events

There are four kinds of Events that can occur: [Sche04]

- Initialization (dealing with the XForms engine)
- Interaction (indicates that something took place)
- Notification (triggers some kind of processing)
- Error Indications (when errors or unusual situations occur)

Each category possesses a number of events that are activated when certain actions occur in the XForms processor. When an event arises so called XForms-actions are initiated. XForms-actions are noting else than XML-elements which act as functions. These are used to modify changes in the data model or in the user interface.

For the code example in Chapter 4.3 which shows the $forms-select causing setvalue actions. The action is integrated in the $forms:item element. The $forms-select event occurs when the user selects a pay method. The $forms:setvalue action modifies the values of the Infobox where further information about the pay method can be displayed (identified by the bind attribute). [DSTC04]

```xml
<xforms:select ref="method"/>
  <xforms:label>Who are you?</xforms:label>
  <xforms:item>
    <xforms:label>Bachelor</xforms:label>
    <xforms:value>bachelor</xforms:value>
    <xforms:setvalue ev:event="xforms:select" bind="Infobox">Informations about the selected degree</xforms:setvalue>
  </xforms:item>
```
6. Implementations

6.1 Overview

A complete list of current implementations can be found at [Mas04]. We want to introduce only three implementations in the following sections:

- X-Smiles, a Java-based XML browser from Helsinki University of Technology
- formsPlayer, an XForms processor plug-in for Internet Explorer 6 SP1
- Chiba project, an implementation of the W3C XForms standard, thereby delivering generic, XML-based form-processing for the web

6.2 Browser X-Smiles

X-Smiles is a Java based XML browser. It is intended for both: desktop use and embedded network devices. X-Smiles supports multimedia services too. The main advantage of the X-Smiles browser is that it supports several XML related specifications and is still suitable for embedded devices supporting the Java environment.

X-Smiles is the first browser implementation for XForms and supports most of the XForms features [XSmi04], e.g. validation, calculations, form controls, data type adaption, events, dynamic binding, and many more.

X-Smiles has a component-based architecture that allows the adoption in many different user interface environments. In addition, XML vocabularies are rendered by Java components, which can be plugged into the browser. XML documents can also be displayed by using XSLT to transform them into an already supported vocabulary. X-Smiles currently supports rendering of SMIL, XSL-FO, XForms, and SVG, and it can use JAXP to select an XML parser. It also has some support for using CSS to style documents.

The X-Smiles project is a "prototype of XML multimedia browser for exotic devices"[XSmi04]. That means you have a browser which doesn't even aim to support HTML, has good multimedia-support and is expected to be tried out and experimented with on set-top boxes, handhelds etc. Normally it is written in Java, so it doesn't actually run on most handhelds. “iPaq is one exception, hopefully, and Linux and KaffeVM are likely to be instrumental in this process.”[Ukn04]

The use of X-Smiles is very simple, if Java is installed on the platform. All XForms exampled can be run. But besides some bug, which can be sorted out, the layout of this browser implementation is not up to date. Of course, the advantage of using this browser on many different devices is very nice and powerful, but for desktop use the performance is very slow and the look of the buttons and navigation bars is so cute, that some users may be deterred.

6.3 FormsPlayer

A user does not need to change his web browser, but can update his Internet Explorer 6 (IE6) with the formsPlayer plugin. So he is able to browse some XForms-using web sites. “formsPlayer is a set of modules designed to make it easy to build XForms processors, editors and debuggers. These processors can run on a variety of platforms, using a range of user interfaces.”[XPor04] As said
above the formsPlayer for IE6 is a complete XForms processor built using the formsPlayer XForms Toolkit. This processor enables to incorporate the entire range of XForms features when a user browses HTML pages with the IE6. The usage is comparable to the “macromedia flashplayer plugin”. In the HTML code a formsPlayer object must be defined, which the formsPlayer plugin can process and visualize in the IE6.

As shown in the example above a common scenario is to embed XForms elements inside XHTML documents. The standard browsers today do not understand XHTML in a very sophisticated way. “However, as XHTML becomes increasingly modularised, and browsers become capable of dealing with elements from other namespaces within XHTML documents, XForms will become the standard form-handling module for XHTML.”[Bir04] The formsPlayer plug-in runs only inside IE6 and the XForms tags are embedded in an ordinary HTML document.

The use of the formsPlayer plugin is very simple. After downloading and installing the plugin, all formsPlayer examples can be browsed with the normal IE6. But this is also a big disadvantage: the plugin only works with the IE6 and only works, if a formsPlayer object is initiated in the XHTML source code. All other XForms can not be processed with this implementation, other than by using X-Smiles.

6.4 Chiba project

Chiba is an open source Java implementation of the W3C XForms standard. Thus, with the XForms-processor it supports the interaction between users and a commonly distant agent, by implementing the XForms-specification.

You always have to keep in mind that there is still no industry standard for the Java-like APIs of XForms for XML-processing (JAXP, JAXM, …). That is the reason why the internal API of Chiba may be subjected by changes in the future. One of the primary aims of the Chiba XForms-framework is the flexibility which is based on Java-interfaces. Thus, you only have to implement the interfaces which correspond to your specific requirements. A disadvantage of increased flexibility is, that it brings more complexity.

Chiba’s presentation of forms and the processing of instance-data is mostly based on XSLT. Furthermore Chiba depends on a special XSLT-processor, XALAN. But a big advantage of server-side XSLT usage is the different design presentation which is independent from any application. Just by modifying the style sheets you can use the forms for mobile browsers or voice-operated browsers. [Mic03]

Here a short overview of the main features of Chiba:

- largely conform to the Xforms 1.0 W3C Recommendation
- supports most Xforms functionality to today’s browsers
- is not relied on scripting client-side scripting capabilities
- supports most Xforms Actions
- strong data-typing
- XForms calculation and validation
- based on Java2 and XSLT
- no client installation required [Tur03]
6.5 XFormation

This software unites Focus’s Software experience in providing XML toolkits and port’s.net knowledge in building XForms processors. x-port.net's XForms processor provides the formsPlayer, which is “one of the two fully compliant XForms processors that enabled the W3C to move the new standard to proposed recommendation on 1 August 2003 and then to a Full Recommendation on 14 October 2003.”[XMa04]

Mark Birbeck, CEO and CTO of x-port.net Ltd and an Invited Expert on two W3C Working Groups - XForms and HTML - said: "XFormation will be key in the adoption of the XForms standard and will provide the functionality required to use XForms to build New Generation Internet Applications. We have looked at various routes for providing an XForms editor to complement our processor and were delighted to be able to partner with Focus due to the richness of their tool."[XMa04] XFormation enables forms developers to work within a complete integrated development environment (IDE) for working with XForms solutions. “It will enable users to rapidly design, develop, manage, preview, and debug XForms.”[Foc04]

Here some key features:

- Guaranteed valid and fully compliant creation of XForms by context sensitive menus.
- XFormation provides a tree view, so that nodes can be added easily, but it also supports a color coded text view.
- Debugging functionality
- XFormation comes with formsPlayer (plug-in to Internet Explorer 6), a leading XForms processor. This enables the developer to process XForms embedded within HTML documents.
- Enables the creation of XForms directly from XML Schema.

7. Conclusion

If a developer has good basic knowledge about XML and HTML forms, it is no problem to acquaint himself with the matter of XForms. The earlier problems of the “old” forms were solved almost exclusively by this new technology. It was high time for a revolution in the "forms market". Because XForms provide a very simple syntax and separate the XForms model and the specification of data collection from the user interface, it is a very powerful, but although a very easy-to-use implementation. XForms examples can easily be run on test platforms after a few minutes.

The approach of XForms, in contrary to other platform independent user interface implementations, is to model the interface on the data side. On the one hand this is based on HTML forms; on the other hand, it eases to use applications with databases. The browser implementations like X-Smiles or the formsPlayer are in early state and not able to address a big user community, because they have some bugs and are not very easy to use. The XFormation software is a very powerful tool and is similar to use as XmlSpy.

But why does not XForm displace the “old” HTML forms till now? It is only a pity about the lack of the offered browsers, on which XForms can be tested. If a user wants to show a web page with XForms he cannot use the most common browsers like IE, Netscape or Opera, but must use exotic tools like X-Smiles or has to install different plugins. But if the next versions of the “standard” browsers will support an XForms processor, the revolution will make its way.
8. References


[Mas04]: Masayasu Ishikawa, 2004-06-11, http://www.w3.org/MarkUp/Forms/

[Mic03]: Mikhalenko, Peter V., XForms und Java Frameworks, 29.07.2003, CNET Networks Deutschland GmbH, http://www.zdnet.de/builder/program/0,39023551,20000910,00.htm


