INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 4 (VIII): XForms

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of XForms
  - Learn how to use them in your web applications

- Outline
  - What is XSorms?
  - HTML Form Example
  - Problems with HTML Forms
  - How does XForms solve these problems?
  - XForms Design Goals
  - XForms Architecture
  - XForms Example
  - How It Works?
  - XForms Model
  - XForms Form Controls
  - XForms Deployment

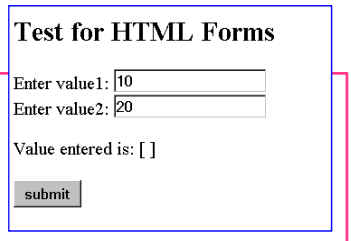
What is XForms?

- Forms are an important part of many web applications today.
- HTML forms became a part of the HTML standard in 1993
- An HTML form makes it possible for web applications to accept input from a user
  - Enhances online interaction with users
  - Allows electronic transactions
  - However, today web users do complex transactions that are starting to exceed the limitations of standard HTML forms
- To enhance the use of forms on the Web, the W3C sponsored the development of XForms
  - XForms 1.0 become an official Recommendation of the W3C in Oct 2003
    - http://www.w3.org/TR/2003/REC-xforms-20031014/
  - XForms 1.0 (Third Edition) was published on 29 Oct 2007
  - XForms 1.1 became a Candidate Recommendation in Nov. 2007
- Future e-commerce solutions are expected to demand the use of XForms-enabled browsers
  - all major browsers will support XForms in the near future

What is XForms? (cont.)

- XForms was designed to be the next generation of HTML/XHTML forms
  - rules for describing and validating data are expressed in XML.
  - the data displayed in a form are stored in an XML document, and the data submitted from the form, are transported over the internet using XML.
  - richer, more secure, more reliable, and more flexible than HTML forms
  - will be the forms standard in XHTML 2.0
  - platform and device independent
  - separates data definition and logic from its presentation, hence XForms data can be defined independent of how the end-user will interact with the application
  - stores and transports data in XML documents
  - contains features like calculations and validations of forms
  - reduces or eliminates the need for scripting

HTML Form Example




```
<html>
  <head>
        <script language="JavaScript">
        </script>
  </head>
  <body>
        <h2>Test for HTML Forms</h2>
        <form name="firstForm" action="">
              Enter value1: <input id="val1" value="10"/><br/>
              Enter value2: <input id="val2" value="20"/><br/>
              <div id="htmlOut"><p>Value entered is: [ ]</p></div>
              <input type="submit" value="submit"/>
        </form>
  </body>
</html>
```


HTML Form Example (cont.)

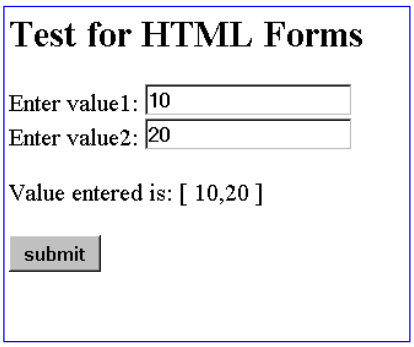
HTML Form with Script

```
<html>
<head>
<script language="JavaScript">
  function check() {
     if ((firstForm.val2.value - 0) > (firstForm.val1.value -0)) {
       htmlOut.innerHTML = "<p>Submitted values: "+firstForm.val1.value+","+firstForm.val2.value+"</p>";
       firstForm.submit();
     }
     else {
       htmlOut.innerHTML = "<p>Error: value 1 must be less than value 2 ( "
                                  +firstForm.val1.value+","+firstForm.val2.value+" )</p>";
      }
      alert("click to continue");
  }
  function display() {
     htmlOut.innerHTML = "<p>Value entered is: [ "+firstForm.val1.value+","+firstForm.val2.value+" ]</p>";
  }
  </script>
  </head>
```

HTML Form Example (cont.)

```
<body onload="display()">
        <h2>Test for HTML Forms</h2>
        <form name="firstForm" action="" onsubmit="check()">
                Enter value1: <input id="val1" value="10" onchange="display()"/><br/>
                Enter value2: <input id="val2" value="20" onchange="display()"/><br/>
                <div id="htmlOut"><p>Value entered is: [ ]</p></div>
                <input type="submit" value="submit"/>
        </form>
  </body>
</html>
```


HTML Form Example (cont.)



Problems with HTML Forms

- Some Assembly Required
- Primitive Data Representation
- Data & presentation intertwined
- Need Script to do anything…
  - Validations
  - Calculations
  - Dynamic Forms
- Medium specific
- High cost of application development and support

How does XForms solve these problems?

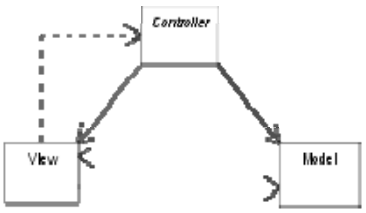
- Content is separated from the presentation
- Presentation is defined elsewhere in the document, it is only bound to the content
- Validation is done in the client using XML schema and inter-data constraints
- Constraints and calculations are defined declaratively in the markup. XForms processor implements them rather than program them in JavaScript
- XForms capable client receives and sends XML directly

XForms Design Goals

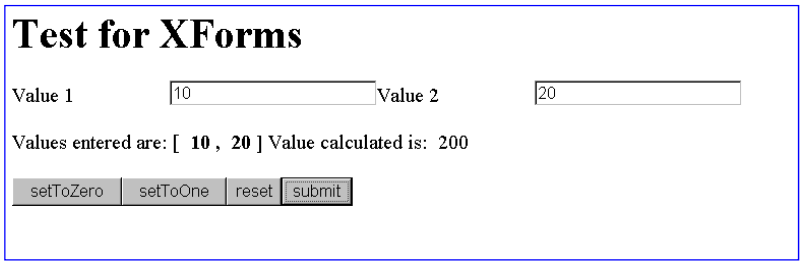
- Be a good XML citizen
  - Submit well-formed XML
  - Build on other XML vocabularies
- Anywhere, anyone, any time, any device
  - Support for desktop browsers, handheld, phones, ATMs, iTV, etc…
- Minimize need for scripting
- From simple client/server to n-tier
  - Decoupled data, logic and presentation
- Accessible
- Not a standalone document type
  - Re-usable module
- XForms is not designed to work alone. There is no such thing as an XForms document.
  - Hosted by other document types: It could run inside XHTML 1.0, or XHTML 2.0.

XForms Architecture

- Model Definition
  - Default data
  - XML Schema references
  - Business rules
- View Definition
  - UI characteristics (HTML/CSS)
- Controller Definition
  - View to Model bindings
  - Submission declaration



XForms Form Example



XForms Form Example (cont.)

<head>

```
<xforms:model id="MyDocument" schemas="local.xsd"
                                xmlns="http://verifone.com/isd/NAXML/XFormsTest">
        <xforms:instance>
                <MyDoc>
                        <Val1>10</Val1>
                        <Val2>20</Val2>
                        <Res/>
                </MyDoc>
        </xforms:instance>
        <xforms:submission ref="MyDocument" id="s00"/>
        <xforms:bind type="xs:integer" nodeset="/MyDoc/*"/>
        <xforms:bind ref="/MyDoc/Val2" constraint="/MyDoc/Val2 &gt; /MyDoc/Val1" />
        <xforms:bind nodeset="/MyDoc/Res" calculate="/MyDoc/Val1 * /MyDoc/Val2"/>
        <xforms:action ev:event="naxml-setToOne">
                <xforms:setvalue ref="/MyDoc/Val1">1</xforms:setvalue>
                <xforms:setvalue ref="/MyDoc/Val2">1</xforms:setvalue>
        </xforms:action>
</xforms:model>
```

XForms Form Example (cont.)

<body>

```
<p>
    <xforms:input ref="/MyDoc/Val1">
        <xforms:label style="width:150px;">Value 1</xforms:label>
        <xforms:hint>Enter the first value, which must be less than the second</xforms:hint>
    </xforms:input>
    <xforms:input ref="/MyDoc/Val2">
        <xforms:label style="width:150px;">Value 2</xforms:label>
        <xforms:hint>Enter the second value, which must be greater than the first</xforms:hint>
    </xforms:input>
</p>
<p>
    Values entered are: [ <b><xforms:output ref="/MyDoc/Val1" />&amp;nbsp;,&amp;nbsp;
                    <xforms:output  ref="/MyDoc/Val2" /> ]</b>
    Value calculated is: <xforms:output ref="/MyDoc/Res"/>
</p>
```

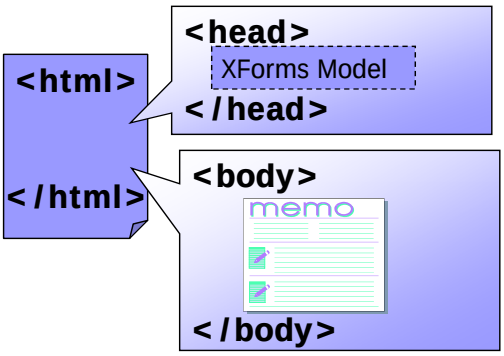

XForms Form Example (cont.)

<body> (cont.)

```
<xforms:trigger>
        <xforms:label>setToZero</xforms:label>
        <xforms:action ev:event="xforms-activate">
            <xforms:setvalue ref="/MyDoc/Val1">0</xforms:setvalue>
            <xforms:setvalue ref="/MyDoc/Val2">0</xforms:setvalue>
        </xforms:action>
</xforms:trigger>
<xforms:trigger>
        <xforms:label>setToOne</xforms:label>
        <xforms:action ev:event="xforms-activate">
            <xforms:dispatch target="MyDocument" name="naxml-setToOne"/>
        </xforms:action>
</xforms:trigger>
...
<xforms:submit submission="s00" class="submit">
        <xforms:label>submit</xforms:label>
</xforms:submit>
```

How It Works?

- The purpose of an HTML form is to collect data. XForms has the same purpose.
- With XForms, input data is described in two different parts:
  - The XForm model (to describe the data and the logic)
  - The XForm user interface (to display and input the data)



Purpose
(non-visible)

Presentation
Form controls

XForms Model

- Defines the 'Purpose' of the form: defines what the form is, what data it contains, and what it should do.
  - presentation independent
- Example

define the XML template for data to be collected,

describe how to submit the data

```
<model>
    <instance>
        <person>
            <fname/>
            <lname/>
        </person>
    </instance>
    <submission id="form1" action="submit.asp" method="get"/>
</model>
```

XForms Model (cont.)

- Includes:
  - <instance> element
    - define the data to be collected, i.e. defines the XML document to be created for the data collected
  - Submit Information
    - describe how to submit the data.
    - Uses <submission> element
      - Attributes: id="form1" identifies the form, action="submit.asp" defines the URL to where the form should be submitted, and method="get" defines the method to use when submitting the data.
  - XML Schema that constrains instance data
  - XForms (dynamic) constraints
  - Privacy information (P3P)

XForms Instance

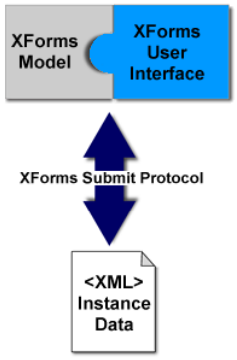
- Provides a template (i.e. an instance of XML document) for the data to be collected in a form
  - default or partially submitted data
  - can be inline or referenced externally
  - used to create an "instance DOM"
- "instance DOM" continually updated
- A subset of the "instance DOM" is serialized and submitted

XForms Without Scripting

- XML Schema defines static constraints
  - Datatypes
  - minimum/maximum occurrences
- XForms extends these with 'computed expressions' evaluated at runtime
  - Is something relevant or required?
  - Calculated fields
  - XPath expressions

XForms Submission

- <submition> specifies
  - What – a portion of the "instance DOM"
  - Where – target URI
  - How – serialization format & transmission protocol
  - Response – life after submit
- Default is 'post' encoded in XML



XForms Model - illustration

```
<model id="p1">
  <schema src=". . ." />
  <instance xmlns="">
  <my:age/>
  </instance>
  <bind ref="age" type="xsd:integer" . . . />
  <submission action=". . ." />
  <privacy src="policyref . . ." />
</model>
```


XForms UI

- Create a user interface
  - used to display and input the data.
  - The user interface elements of XForms are called controls (or input controls)
- "connect" the user interface elements to the appropriate data in the model
- Create interactive "views" of a model
- Example

```
<input ref="fname"> <label>First Name</label></input>
<input ref="lname"><label>Last Name</label></input> <submit
  submission="form1"><label>Submit</label></submit>
```

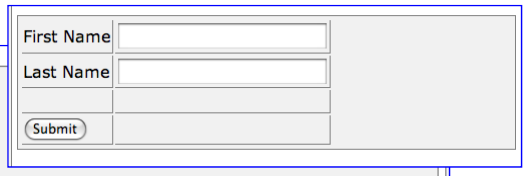
Put it all together

```
<xforms>

<model>
<instance>
  <person>
    <fname/>
    <lname/>
  </person>
</instance>
<submission id="form1"
action="submit.asp"
method="get"/>
</model>

<input ref="fname"><label>First Name</label></input>
<input ref="lname"><label>Last Name</label></input>
<submit submission="form1"><label>Submit</label></submit>

</xforms>
```



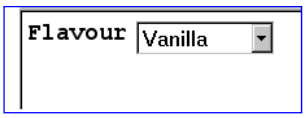
XForms Form Controls

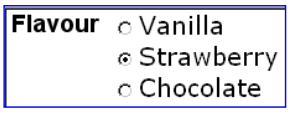
- Bind to the model
- Metadata for the user
- Shortcuts & navigation hints
- Presentation hints
- CSS styling

<input>	<output>	<secret>
<textarea>	<range>	<upload>
<select>	<select1>	<submit>
	<button>	

XForms UI Controls

```
<select1 ref="my:icecream/my:flavor">
  <caption>Flavour</caption>
  <item><caption>Vanilla</caption><value>v</value></item>
  <item><caption>Strawberry</caption><value>s</value></item>
  <item><caption>Chocolate</caption><value>c</value></item>
</select1>
```





XForms UI

- Aggregations create sophisticated user interfaces
  - Obviating common uses of scripting

Construct	Purpose
<group>	Group related controls
<switch>	Conditional for dynamic UI
<itemset>	Dynamic selections
<repeat>	Repeating structures

XForms repeat - illustration

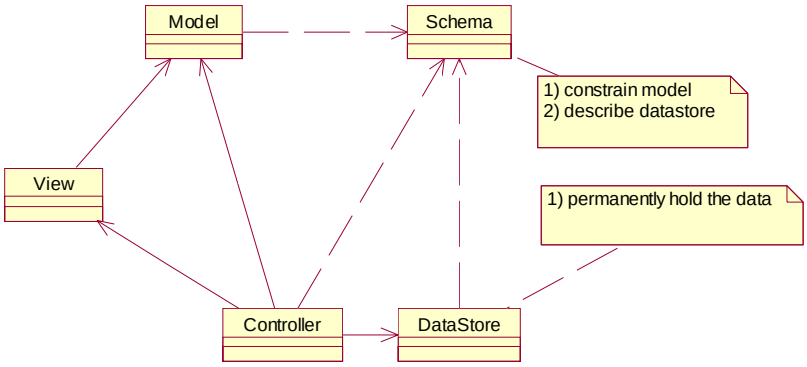
```
<repeat nodeset="/cart/item">
 <h:tr>
  <h:td>
   <input ref="product" />
  </h:td>
  <h:td>
   <input ref="description" />
  </h:td>
 </h:tr>
</repeat>
```


XForms Actions

- Declarative markup for event handlers
  - Uses XML Events
  - Reduces need for scripting

dispatch	refresh	recalculate	revalidate
setfocus	load	setvalue	send
reset	setindex	insert	delete
toggle	script	message	action

Classical Model View Controller (MVC)



XForms Deployment

- XForms can be implemented
  - In a client
  - In a server
    - Can deliver legacy markup to clients which lack native support
- Enables a front end to Web Services
- An XForms processor can be at multiple points in the network

Q & A

References

- Some useful links with examples and other resources:
  - W3School XForms Tutorial
    - http://www.w3schools.com/xforms/default.asp
  - Interactive XForms School
    - http://xformsinstitute.com/
  - W3C XForms page
    - http://www.w3.org/MarkUp/Forms/
  - formsPlayer -- a full-featured XForms processor
    - http://www.formsplayer.com