XForms

Jussi Saarinen
jussi.p.saarinen@tut.fi
Institute of Software Systems
Tampere University of Technology
XForms in General

- Defined by World Wide Web Consortium (W3C)
- Splits traditional XHTML forms to three parts
  - model, instance data, and user interface
- Intended to be integrated into other markup languages such as XHMTL or SVG
  - adopted first in XHTML 2
- Mainstream browser support still quite limited
HTML Forms

• In HTML forms the collected data is a set of name/value pairs determined by controls in the form.

• Form navigation can be guided through use of accesskey and tabindex attributes

• Textual controls can be set “readonly” or “disabled”

• HTML Form controls include:
  - single-line text input
  - multi-line text input
  - password text input
  - submit and reset
  - buttons (script enabled)
  - radio buttons
  - checkboxes
  - single select menu
  - multiple-select menu
  - file select
  - hidden controls
  - object controls
HTML Forms

- Initial values can be given to different controls
- Submission can be done in urlencoded or multipart/form-data (MIME) formats
  - GET or POST methods used to submit data

```
<form action="http://example.com/cgi-bin/submit-here"
name="shake-poll">
  <p>Poll: to be or not to be?</p>
  <input type="radio" name="thequestion" id="radio1" value="b"/>
  <label for="radio1">To Be</label><br/>
  <input type="radio" name="thequestion" id="radio2" value="n"/>
  <label for="radio2">Not To Be</label><br/>
  <input type="radio" name="thequestion" id="radio3"/>
  <label for="radio3">Other (please specify)</label><br/>
  <input type="text" name="othersel"/>
</form>
```
Issues With HTML Forms

- Dependent of scripting languages
  - marking required information
  - performing validations and calculations
  - error messages
  - managing dynamic layout
- Form data initialization based on existing information
- “Flat” data representation
  - name/value pairs
- Assumption of one step process from a client to a server
  - reinterpreting of data in later steps
Solutions provided by XForms

- Reduces need for scripting
  - XPath based calculations and validations
  - provides dynamic features like repeating tables and optional selections
  - XForms actions (e.g. set focus or change data value)
- Form data can be initialized directly from an XML file
- Use of XML provides a richer way of representing data
- XForms enables the gathered data to be circulated among necessary parties without need of reinterpretation
Benefits of XForms

- Strong typing
  - ready tools for type checking
- Existing schema re-use
- External schema augmentation
- Internationalization
  - use of XML 1.0
- Enhanced accessibility for underlying applications
  - content and presentation separation
- Multiple device support
  - high level user interface
XPath

- A W3C Recommendation
- Used by XForms to address elements in XML structures
  - Enables selection of a single node or a node-set
  - Syntax for location step: axisname::nodetest[predicate]
  - Example 1: /html/head/title
  - Example 2: purchaseOrder/items/item[3]
  - Example 3: /html/head/*[@id][2]
- Provides also support for basic calculations
  - mathematics, rounding, string manipulation etc.
  - Example 4: string-length('hello world')
  - Example 5: purchaseOrder/subtotal * instance('taxtable')/tax
XPath

- Axes are used as navigation instructions
  - define a node-set relative to the current node
  - default axis is 'child'
  - Example: abbreviation .. equals parent::node( )
  - Axis names include:
    - ancestor
    - ancestor-or-self
    - descendant
    - attribute
    - following
    - preceding
    - etc. ...
XML Schema

• A W3C recommendation
• XForms uses datatypes defined in XML Schema
• XML Schema is written in XML
• Defines
  – elements and attributes that can appear in a document
  – possible child elements, their order and number
  – whether element is empty or contains text
  – data types for elements and attributes
  – default and fixed values for elements and attributes
XML Schema

- Enables
  - definition of allowed document content
  - validation of data correctness
  - definition of restrictions on data
  - definition of data patterns

<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
  <xs:element name="to" type="xs:string">
  ...
  </xs:element>
</xs:schema>

<?xml version="1.0"?>
<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

<xs:element name="xxx" type="yyy"/>
XML Schema

- Common types:
  - `xs:string`
  - `xs:decimal`
  - `xs:integer`
  - `xs:boolean`
  - `xs:date`
  - `xs:time`

- Both elements and attributes may have default or fixed values

- Attributes can be defined to be either “required” or “optional”
XML Schema

- Possible data value restrictions:

<table>
<thead>
<tr>
<th>enumeration</th>
<th>maxExclusive</th>
<th>minExclusive</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>fractionDigits</td>
<td>maxInclusive</td>
<td>minInclusive</td>
<td>totalDigits</td>
</tr>
<tr>
<td>length</td>
<td>maxLength</td>
<td>minLength</td>
<td>whiteSpace</td>
</tr>
</tbody>
</table>

- Complex types can be defined for complex elements
  - empty elements
  - contain other elements and/or attributes

```xml
<xs:element name="letter" type="lettertype"/>
<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
    <xs:element name="name" type="xs:string" maxOccurs="10" minOccurs="1"/>
    <xs:element name="orderid" type="xs:positiveInteger"/>
    <xs:element name="shipdate" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```
XML Schema

- Data value restriction examples:

```xml
<xs:element name="age">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="100"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="car" type="carType"/>
    <xs:simpleType name="carType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Audi"/>
            <xs:enumeration value="Golf"/>
            <xs:enumeration value="BMW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="password">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z0-9]{8}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="gender">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="male|female"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```
XForms Model

• The Model of MVC

• Is used to describe the data separately from the UI (control elements)

• Marked with `<model>` element

• Contains an `<instance>` element
  – contains a template for the data to be collected

• Contains a `<submission>` element
  – defines how the instance data is submitted

• May contain a `<bind>` element
  – used to bind elements inside the model to control elements
XForms Model

• An example of a model:

```
<model>
  <instance>
    <person>
      <name>
        <fname/>
        <lname/>
      </name>
    </person>
  </instance>
  <bind nodeset="/person/name/fname" id="firstname"/>
  <bind nodeset="/person/name/lname" id="lastname"/>
  <submission id="form1" method="get" action="submit.asp"/>
</model>
```
XForms User Interface

- The View of MVC
- Uses XForms controls
  - device independent
- Each control element contains a `<label>` element
  - defines the visual guide (text) presented for the user
- `<select>` and `<select1>` elements contain multiple `<item>` elements that each contain their own `<label>` and `<value>` elements
- Control elements:

<table>
<thead>
<tr>
<th>input</th>
<th>submit</th>
<th>upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>secret</td>
<td>trigger</td>
<td>select(1)</td>
</tr>
<tr>
<td>textarea</td>
<td>output</td>
<td>range</td>
</tr>
</tbody>
</table>
XForms User Interface

- Each control element is bound to some element(s) in the model
- With IDREFs

```xml
<!-- in the XForms Model -->
<xforms:bind nodeset="email" id="mybind" required="true()"/>
...
<!-- later in the document -->
<xforms:input bind="mybind"...>
```

- With XPath

```xml
<!-- in the XForms Model -->
<xforms:bind nodeset="email" id="mybind" required="true()"/>
...
<!-- later in the document -->
<xforms:input ref="email"...>
```
XForms User Interface

- Dynamic forms can be made without scripting
- **With `<switch>` and `<case>` elements**

```xml
<model>
    <toggle ev:event="xforms-ready" case="go">
        ...
    </model>
<switch>
    <case id="default_message">You are using a browser that doesn't support XForms</case>
    <case id="go">...<case>
</switch>
```

- **With `<repeat>` element**

```xml
<!-- insert just after the index item -->
<trigger>
    <label>Insert</label>
    <insert nodeset="/items/item" at="index('r1')" position="after"/>
    <setvalue ref="/items/item[index('r1')]/@quantity">0</setvalue>
</trigger>

<!-- delete the index item -->
<trigger>
    <label>Delete</label>
    <delete ev:event="DOMActivate" nodeset="/items/item" at="index('r1')"/>
</trigger>
```
XForms Processing

Bootstrap

Create and Initiate XForms Model

Create and Initiate form control

User Interaction

Notification events

Interaction events

Error indications

Serialization and Submission

Shutdown

source: XForms Essentials
Actions and Events

- XForms actions are handling response to events

**Actions:**

<table>
<thead>
<tr>
<th>message</th>
<th>reset</th>
<th>delete</th>
<th>refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>setvalue</td>
<td>load</td>
<td>setindex</td>
<td>rebuild</td>
</tr>
<tr>
<td>setfocus</td>
<td>toggle</td>
<td>revalidate</td>
<td>dispatch</td>
</tr>
<tr>
<td>send</td>
<td>insert</td>
<td>recalculate</td>
<td>action</td>
</tr>
</tbody>
</table>

- Some useful events:

<table>
<thead>
<tr>
<th>DOMActivate</th>
<th>xforms-model-destruct</th>
<th>xforms-submit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMFocusIn/Out</td>
<td>xforms-help</td>
<td>xforms-value-changed</td>
</tr>
<tr>
<td>xforms-ready</td>
<td>xforms-hint</td>
<td>xforms-select</td>
</tr>
<tr>
<td>xforms-model-construct-done</td>
<td>xforms-reset</td>
<td>xforms-deselect</td>
</tr>
<tr>
<td>xforms-valid</td>
<td>xforms-invalid</td>
<td>xforms-readonly</td>
</tr>
<tr>
<td>xforms-readwrite</td>
<td>xforms-required</td>
<td>xforms-optional</td>
</tr>
<tr>
<td>xforms-enabled</td>
<td>xforms-disabled</td>
<td>xforms-out-of-range</td>
</tr>
<tr>
<td>xforms-in-range</td>
<td>xforms-submit-done</td>
<td></td>
</tr>
</tbody>
</table>
Actions and Events

- Example: Message action

```html
<input ref="fname">
<label>First Name</label>
<message level="ephemeral" event="DOMFocusIn">
    Input Your First Name
</message>
</input>
```

source: XForms Essentials
Used Sources

- XForms Essentials, Micah Dubinko, O'Reilly 2003
  - Available at: http://xformsinstitute.com/essentials/
- W3C tutorials (XML, XPath, XML Schema, XForms) 2006
  - Available at: http://www.w3schools.com/
- W3C XForms Recommendation 2006
  - Available at: http://www.w3.org/TR/xforms/
- X-Smiles Browser
  - Available at: http://www.x-smiles.org/