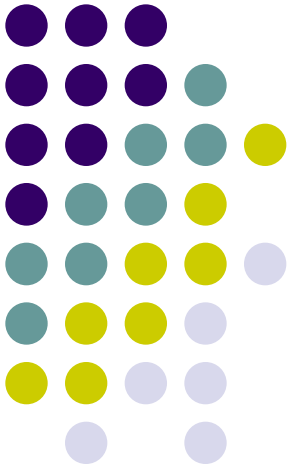
NPRG039

Advanced Aspects and New Trends in XML (and Related) Technologies

RNDr. Irena Holubová, Ph.D.

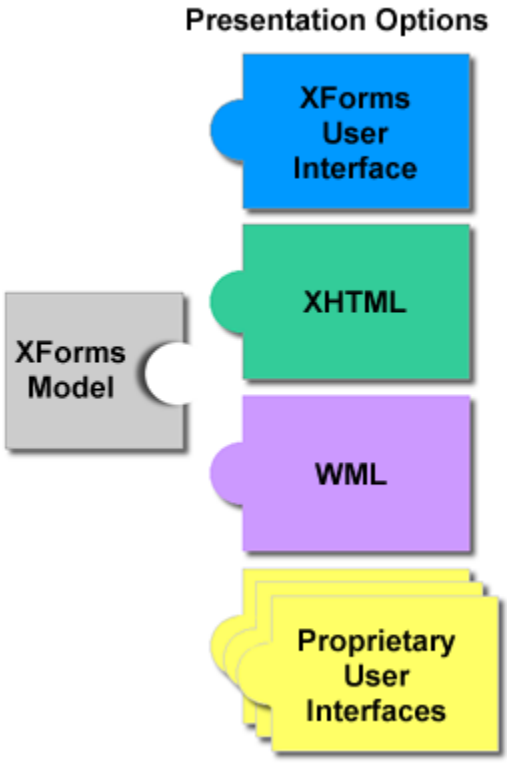
holubova@ksi.mff.cuni.cz



Lecture 8. XForms

http://www.ksi.mff.cuni.cz/~svoboda/courses/171-NPRG039/

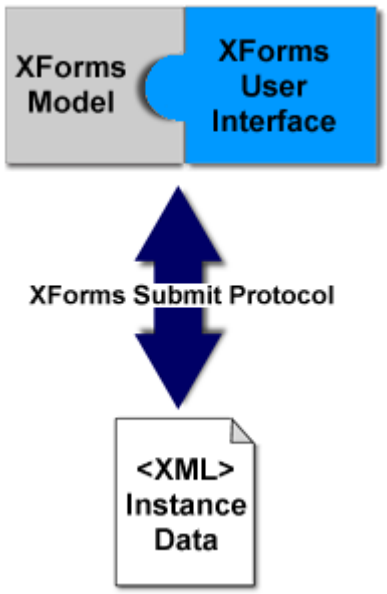
XForms



- W3C Specification
- XML markup for a new generation of forms and form-like applications on the Web
- Motivation: HTML Web forms do not separate purpose from presentation of a form
  - XForms separate what the form does and how it looks
- XForms Model can work with various interfaces
- XForms User Interface provides a standard set of visual controls
  - Aim: replace XHTML forms
    - And possibly other
  - Directly usable inside XHTML and other XML documents

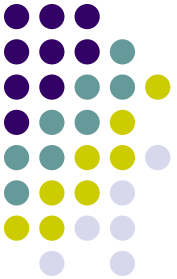
http://www.w3.org/TR/xforms20/

XForms



- Forms collect data, which is expressed as XML instance data
  - XForms model describes their structure
- XForms Submit Protocol defines how XForms send and receive data
  - A channel for instance data to flow to and from the XForms processor
  - Ability to suspend and resume the completion of a form

XForms – Example



```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xf="http://www.w3.org/2002/xforms">
  <head>
    <title>Hello World in XForms</title>
    <xf:model>
      <xf:instance>
        <f:data xmlns:f="http://foo.com">
          <f:PersonGivenName/>
        </f:data>
      </xf:instance>
    </xf:model>
  </head>
  ...
```

XForms – Example

```
...
<body>
  <p>Type your first name in the input box. If you are running XForms,
     the output should be displayed in the output area.</p>
  <p>
    <xf:input ref="f:PersonGivenName" incremental="true">
      <xf:label>Please enter your first name: </xf:label>
    </xf:input>
  </p>
  <p>
    <xf:output value="concat('Hello ',
                             f:PersonGivenName,
                             '. We hope you like XForms!')">
      <xf:label>Output: </xf:label>
    </xf:output>
  </p>
  <div id="console" style="display: block"/>
</body>
</html>
```

XForms – Example

Type your first name in the input box.
If you are running XForms, the output should be displayed in the output area.

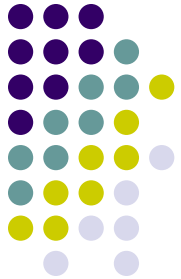
Please enter your first name: Irena

Output: Hello Irena. We hope you like XForms!

0 -> Dispatching event xforms-recalculate on <SPAN class="xforms-model" id="xsltforms-mainform-model-default"/>
1 -> Dispatching event xforms-revalidate on <SPAN class="xforms-model" id="xsltforms-mainform-model-default"/>
2 -> Dispatching event xforms-refresh on <SPAN class="xforms-model" id="xsltforms-mainform-model-default"/>
1 -> Dispatching event xforms-optional on <SPAN class=" xforms-control xforms-input xforms-appearance xforms-optional xforms-enabled xforms-readwrite xforms-valid xforms-focus" id="xsltforms-mainform-input-4_4_3_"/>
1 -> Dispatching event xforms-enabled on <SPAN class=" xforms-control xforms-input xforms-appearance xforms-optional xforms-enabled xforms-readwrite xforms-valid xforms-focus" id="xsltforms-mainform-input-4_4_3_"/>
1 -> Dispatching event xforms-readwrite on <SPAN class=" xforms-control xforms-input xforms-appearance xforms-optional xforms-enabled xforms-readwrite xforms-valid xforms-focus" id="xsltforms-mainform-input-4_4_3_"/>
1 -> Dispatching event xforms-valid on <SPAN class=" xforms-control xforms-input xforms-appearance xforms-optional xforms-enabled xforms-readwrite xforms-valid xforms-focus" id="xsltforms-mainform-input-4_4_3_"/>
1 -> Dispatching event xforms-value-changed on <SPAN class=" xforms-control xforms-input xforms-appearance xforms-optional xforms-enabled xforms-readwrite xforms-valid xforms-focus" id="xsltforms-mainform-input-4_4_3_"/>

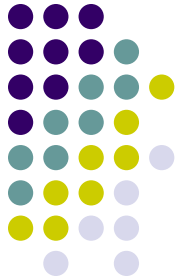
http://www.agencexml.com/xsltforms/hello.xml

Key Goals of XForms



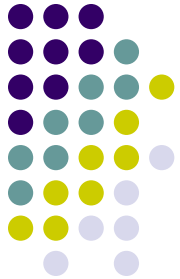
1. Support for structured form data
2. Advanced forms logic without server round-tripping
3. Dynamic access to server data sources during form execution
4. Decoupled data, logic and presentation
5. Seamless integration with other XML tag sets
6. Richer user interface to meet the needs of business, consumer and device control applications
7. Support for handheld, television, and desktop browsers, plus printers and scanners
8. Improved internationalization and accessibility
9. Multiple forms per page, and pages per form
10. Suspend and Resume capabilities

XForms Key Advantages



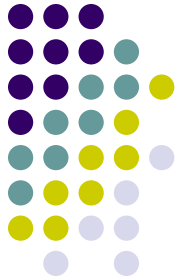
1. Model-View-Controller (MVC) architecture
2. Declarative programming
   - Easier to learn, maintain and debug
3. Rich set of user interface controls for handling complex data
4. Compatibility with XML standards
   - Cascading Style Sheets (CSS), XML Schema and XPath
5. Extensibility
6. Less JavaScript

XForms Key Disadvantages



1. Web browser limitations ⇒ difficult for XForms tools vendors to support all XForms tags in all browsers
2. Slow performance on very-large forms
   - With over 200 fields
3. Lack of understanding of XML
   - Traditional HTML/JavaScript forms developers are not familiar with XML standards
4. Few complete working XForms examples and applications exist
   - Gets better
5. Few high-quality and low-cost GUI forms builder tools
   - Exception: IBM's Workplace forms, OpenOffice Forms, Onyx Forms, …

XForms Form



- Parts:
  1. XForms model = specification of what it should do
  2. XForms interface = specification of how it should look
     - Form controls
       - e.g., **`input`**, **`submit`**, **`label`**, …
- Binding = connection between the two parts
  - Based on XPath
  - Instance data – empty (blank form) or having initial data (pre-populated form)
    - Place for entered data to reside
  - Attribute **`ref`** on each form control = XPath expression that points to a location in the instance data
    - Or other types of attributes with similar purpose – see later

XForms Instance Data

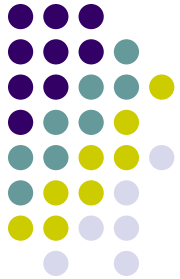
```
<xf:model id="m1">
  <xf:instance>
    <Invoice>
      <InvoiceLine>
        <InvoicedQuantity unitCode="PKG">5</InvoicedQuantity>
        <Item>
          <Description>Box of Protractors; 500 count<Description>
        </Item>
      </InvoiceLine>
    </inv:Invoice>
  <xf:instance>
  <xf:bind nodeset="my:InvoiceLine/my:Item/my:Description"
           required="1"/>
  <xf:submission id="s" method="put" action="po.xml"/>
<xf:model>
```

Inlined XML data, or document pointed to by a **src** attribute

XForms Key Form Controls

Control	Intent	Examples
`input`	Entry of free-form value	edit box
`textarea`	Entry of large amounts of free-form text	email body, weblog entry, …
`secret`	Entry of sensitive information	password prompt
`select1`	Choice of one-and-only-one item from a list	radio buttons, drop-list
`select`	Choice of one or more items from a list	checkbox group, listbox
`range`	Selecting a value from a range	slider, volume control
`upload`	Selecting a data source	file picker, digital camera interface
`trigger`	Activating a defined process	button, hyperlink
`submit`	Activating submission of the form	submit button
`output`	Display-only of form data	inline text

Form Controls

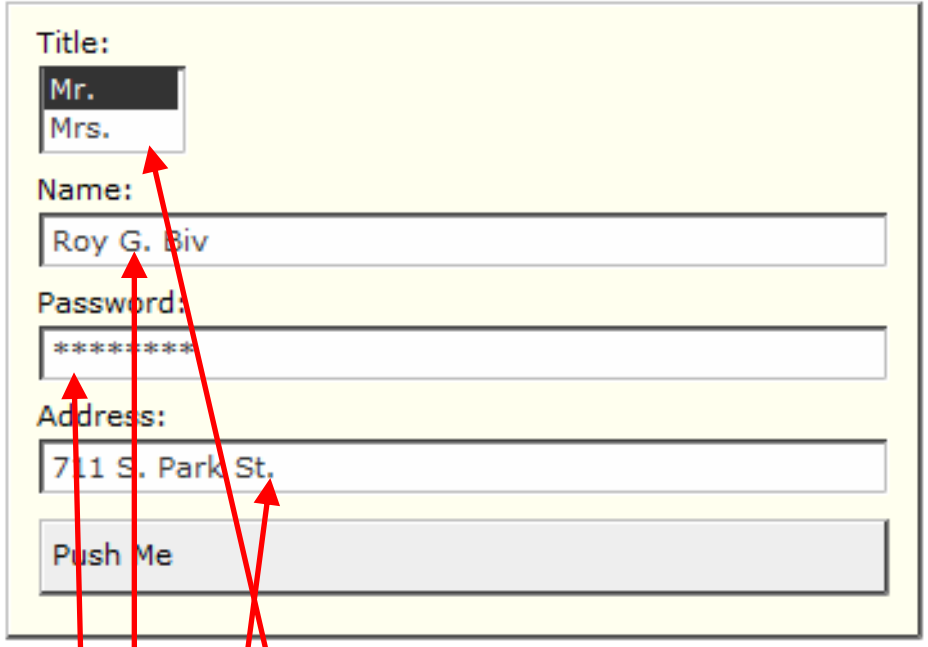


- Every form control has a required label child
  - Except for **`output`**, where it is optional
  - Enforces the good design habit of always associating a label with a form control
- Common child elements:
  - **`help`** – message at the user's request
  - **`hint`** – message at the user agent's request
  - **`alert`** – error message

Form Controls

Example

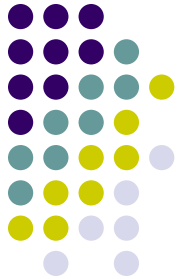
```
<xf:select1
    ref="employee/name/@title">
  <xf:label>Title:</xf:label>
</xf:select1>
<xf:input ref="employee/name">
  <xf:label>Name:</xf:label>
</xf:input>
<xf:secret ref="employee/password">
  <xf:label>Password:</xf:label>
</xf:secret>
<xf:textarea ref="employee/address">
  <xf:label>Address:</xf:label>
</xf:textarea>
<xf:trigger>
  <xf:label>Push Me</xf:label>
</xf:trigger>
```




```
<xf:model>
  <xf:instance>
    <employee xmlns="http://example.info">
      <name title="Mr.">Roy G. Biv</name>
      <email>rgb@example.info</email>
      <password>Password</password>
      <address>711 S. Park St.</address>
      <whuffie>32</whuffie>
      <picture></picture>
    </employee>
  </xf:instance>
</xf:model>
```

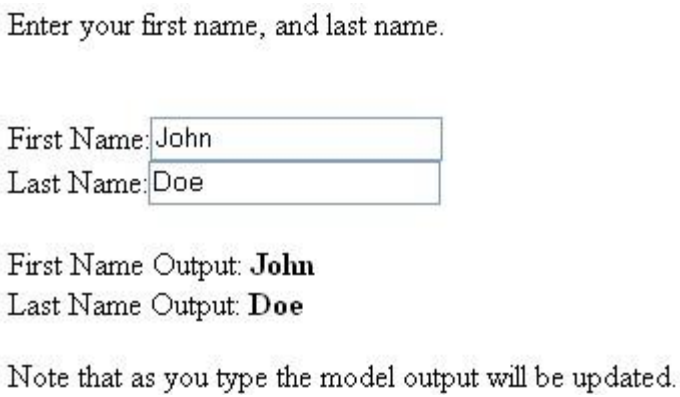
Form Controls

Adjusting Appearance



- Attribute appearance
  - For each form control
  - Finer control of the appearance
    - **`full`**, **`compact`**, **`minimal`**
- Example for **`select1`**:
  - **`full`** = shows every possible choice
    - A large radio button group
  - **`minimal`** = presents only a tiny one-line list that would have to be expanded to see the choices
  - **`compact`** = presents a middle ground
- Only suggestions to the XForms engine

Input, Output




```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <PersonGivenName/>
      <PersonSurName/>
    </data>
  </xf:instance>
</xf:model>
```

```
<p>Enter your first name, and last name.</p>
<xf:input ref="PersonGivenName" incremental="true">
  <xf:label>Input First-Name:</xf:label>
  <xf:hint>Also known as given name.</xf:hint>
</xf:input>
<br/>
<xf:input ref="PersonSurName" incremental="true">
  <xf:label>Input Last Name:</xf:label>
  <xf:hint>Also known as sur name or family name.</xf:hint>
</xf:input>
<br/><br/>
Output First Name: <b><xf:output ref="PersonGivenName"/></b>
<br/>
Output Last Name: <b><xf:output ref="PersonSurName"/></b>
<p>Note that as you type the model output will be updated.</p>
```

Output is immediately updated

XForms, HTML, CSS

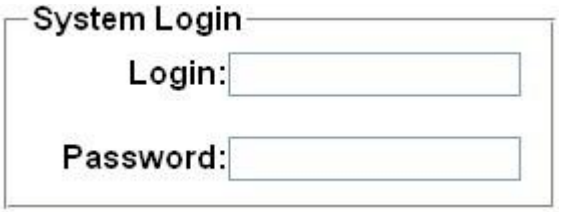

```
<style type="text/css"><![CDATA[
@namespace xf
    url("http://www.w3.org/2002/xforms");

label, legend {
    font-family: Arial, Helvetica, sans-
    serif;
    font-weight: bold;
}
fieldset {
    padding: 5px;
    width: 260px;
}
xf|label {
    width: 150px;
    margin: 3px;
    text-align: right;
}
xf|value {
    text-align: left;
}
]]></style>
```

```
<xf:group>
  <fieldset>
    <legend>Mailing Address</legend>
    <xf:input
    ref="LocationStreetFullText">
      <xf:label>Street: </xf:label>
    </xf:input>
    <xf:input
    ref="LocationStreetFullText2">
      <xf:label />
    </xf:input>
    <xf:input ref="LocationCityName">
      <xf:label>City:</xf:label>
    </xf:input>
    <xf:input ref="LocationStateName">
      <xf:label>State:</xf:label>
    </xf:input>
    <xf:input ref="LocationPostalID">
      <xf:label>Postal Code:</xf:label>
    </xf:input>
  </fieldset>
</xf:group>
```

Secret

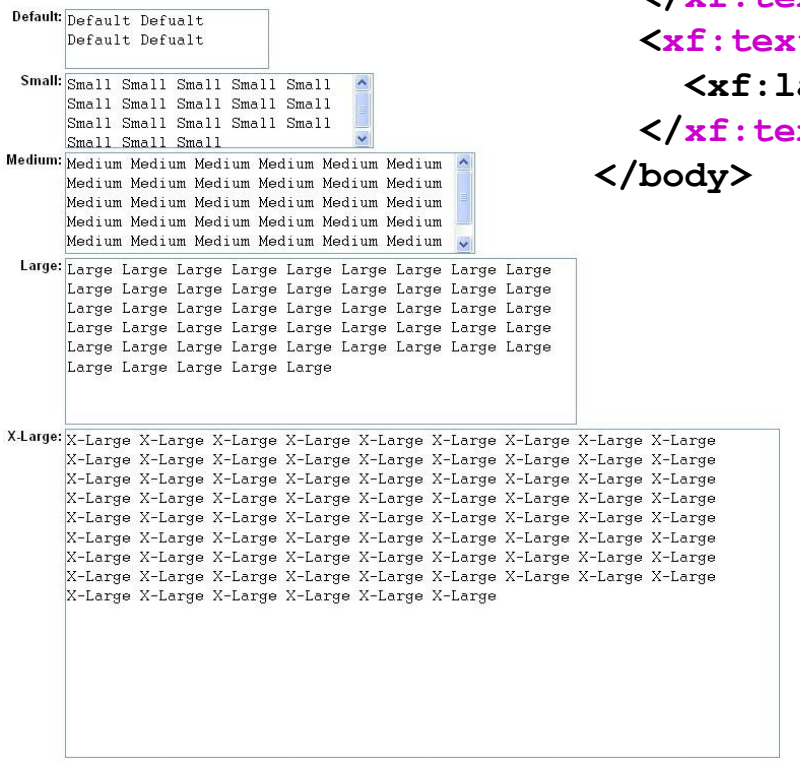
```
<xf:model>
   <xf:instance xmlns="">
      <Login>
         <LoginID />
         <Password />
      </Login>
   </xf:instance>
</xf:model>
```

```
<fieldset>
  <legend>System Login</legend>
  <xf:input ref="LoginID">
    <xf:label>Login: </xf:label>
  </xf:input>
  <br />
  <xf:secret ref="Password">
    <xf:label>Password: </xf:label>
  </xf:secret>
</fieldset>
```



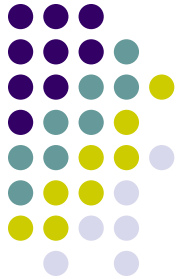
Textarea

```
<body>
  <xf:textarea ref="Default">
    <xf:label>Default: </xf:label>
  </xf:textarea>
  <xf:textarea class="small-textarea" ref="Small">
    <xf:label>Small: </xf:label>
  </xf:textarea>
  <xf:textarea class="medium-textarea" ref="Medium">
    <xf:label>Medium: </xf:label>
  </xf:textarea>
  <xf:textarea class="large-textarea" ref="Large">
    <xf:label>Large: </xf:label>
  </xf:textarea>
  <xf:textarea class="x-large-textarea" ref="XLarge">
    <xf:label>X-Large: </xf:label>
  </xf:textarea>
</body>
```




```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <Default>Default ...</Default>
      <Small>Small ... </Small>
      <Medium>Medium ...</Medium>
      <Large>Large ...</Large>
      <XLarge>X-Large ...</XLarge>
    </data>
  </xf:instance>
</xf:model>
```

Textarea



```
@namespace xf url("http://www.w3.org/2002/xforms");
...
.default-textarea
{
 font-style: regular;
}
.small-textarea textarea {
   height: 4.4em;
   width: 300px;
}
.medium-textarea textarea {
   height: 6em;
   width: 400px;
}
.large-textarea textarea {
   font-family: Courier, sans-serif;
   height: 10em;
   width: 500px;
}
...
```

Checkbox

```
<body>
   <h1>XForms Checkbox Demo</h1>
   <xf:input ref="bool1">
      <xf:label>Bool 1: </xf:label>
   </xf:input>
   <br />
   <xf:input bind="bool2">
      <xf:label>Bool 2: </xf:label>
   </xf:input>
   <br />
   <xf:output ref="bool1">
      <xf:label>Bool 1: </xf:label>
   </xf:output>
   <br />
   <xf:output bind="bool2">
      <xf:label>Bool 2: </xf:label>
   </xf:output>
</body>
```

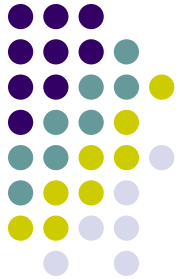
two options: bind / ref

```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <bool1>true</bool1>
      <bool2>false</bool2>
    </data>
  </xf:instance>
  <xf:bind nodeset="bool1"
           type="xs:boolean" />
  <xf:bind id="bool2"
           nodeset="bool2"
           type="xs:boolean" />
</xf:model>
```

two options: nodeset / ref

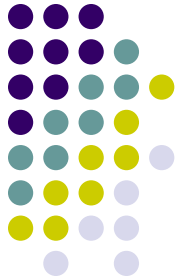
Bool 1: ☑
Bool 2: ☐
Bool 1: true
Bool 2: false

Binding



- Core concepts:
  1. Model = tree of data elements
  2. Presentation = tree of presentation elements
  3. Two trees need to be wired together (binding)
- Element **`bind`**
  - Can have an attribute **`id`**
    - Used for binding
  - Can have other attributes to specify further features of the bound data
- Single input can be bound to many outputs
- Single output can depend on many inputs
  - e.g., using **`calculate`** attribute of **`bind`** statement

Bind Attributes



- **`type`** = associates XML Schema datatype with the attached instance data nodes
- **`relevant`** = controls whether the attached instance data is relevant with regard to given expression
  - Non-relevant items are usually not rendered
    - Can be fine-tuned
- **`required`** = attached instance data must not be blank
  - Prevents submission until the condition is met
- **`readonly`** = the attached instance data ca not be user-modified
- **`calculate`** = XPath expression which can include mathematical operators and function calls to automatically provide a value to the attached instance data nodes
  - Multiple calculations can interact
- **`constraint`** = XPath expression which must evaluate to true() in order for the attached instance data to be considered valid (before submission can succeed)

Bind Attributes

type, relevant

```
<xf:select1 ref="/var/first" >
  <xf:label>Should I show the second question?
   </xf:label>
  <xf:item select="yes">
    <xf:label>Yes Please!</xf:label>
    <xf:value>1</xf:value>
  </xf:item>
  <xf:item>
    <xf:label>No Thank You</xf:label>
    <xf:value>0</xf:value>
  </xf:item>
</xf:select1>
<xf:input ref="/var/second">
  <xf:label>Second value: </xf:label>
</xf:input>
```

```
<xf:model>
  <xf:instance>
    <var xmlns="">
      <first>1</first>
      <second>This is the second value</second>
    </var>
  </xf:instance>
  <xf:bind nodeset="/var/first"  type="xs:decimal" />
  <xf:bind nodeset="/var/second" relevant="/var/first &gt; 0" />
</xf:model>
```

Bind Attributes

calculate, constraint

```
<xf:model>
  <xf:instance>
    <order>
      <item>
        <amount />
        <discount />
      </item>
    </order>
  </xf:instance>
  <xf:bind ref="my:item/my:discount"
           calculate="../my:amount * 0.1"
           relevant="../my:amount > 1000"/>
</xf:model>
```

```
<xf:model>
  <xf:instance>
    <range>
      <from />
      <to />
    </range>
  </xf:instance>
  <xf:bind ref="my:to" constraint=". > ../my:from" />
</xf:model>
```

Bind Attributes

Multiple Predicates

```
<xf:model id="model">
  <xf:instance id="input">
    <DataIn xmlns="">
      <InputIndicator>false</InputIndicator>
    </DataIn>
  </xf:instance>
  <xf:bind id="input_bind" nodeset="/DataIn/InputIndicator"
           type="xs:boolean" />

  <xf:instance id="output">
    <DataOut xmlns="">
      <OutputValue>Hello World!</OutputValue>
    </DataOut>
  </xf:instance>

  <!-- if the input is true, then the output is relevant -->
  <xf:bind id="output_bind"
           nodeset="instance('output')/OutputValue"
           relevant="instance('input')/InputIndicator[.='true']"/>
</xf:model>
<p>
  <xf:input bind="input_bind">
    <xf:label>Check to see the value of output: </xf:label>
  </xf:input>
<br/>
  <xf:output bind="output_bind">
    <xf:label>Value of Output: </xf:label>
  </xf:output>
</p>
```

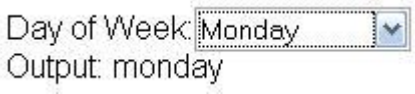
Select1

```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <ColorCode/>
    </data>
  </xf:instance>
</xf:model>
```


```
<body>
  <xf:select1 ref="ColorCode" appearance="full" >
    <xf:item>
      <xf:label>Red</xf:label>
      <xf:value>red</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Orange</xf:label>
      <xf:value>orange</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Yellow</xf:label>
      <xf:value>yellow</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Green</xf:label>
      <xf:value>green</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Blue</xf:label>
      <xf:value>blue</xf:value>
    </xf:item>
  </xf:select1>
  Output: <xf:output ref="ColorCode"/>
</body>
```

Select1

```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <DayOfWeekCode/>
    </data>
  </xf:instance>
</xf:model>
```

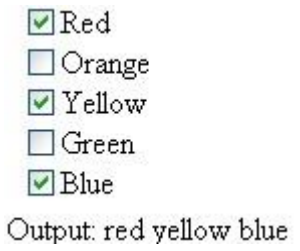


default appearance

```
<body>
  <xf:select1 ref="DayOfWeekCode">
    <xf:label>Day of Week:</xf:label>
    <xf:item>
      <xf:label>Monday</xf:label>
      <xf:value>monday</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Tuesday</xf:label>
      <xf:value>tuesday</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Wednesday</xf:label>
      <xf:value>wednesday</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Thursday</xf:label>
      <xf:value>thursday</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Friday</xf:label>
      <xf:value>friday</xf:value>
    </xf:item>
  </xf:select1>
Output: <xf:output ref="DayOfWeekCode"/>
</body>
```

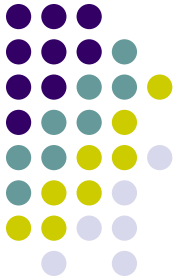
Select

```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <MyCode type="xs:string"/>
    </data>
  </xf:instance>
</xf:model>
```




```
<body>
  <xf:select ref="MyCode" appearance="full" >
    <xf:item>
      <xf:label>Red</xf:label>
      <xf:value>red</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Orange</xf:label>
      <xf:value>orange</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Yellow</xf:label>
      <xf:value>yellow</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Green</xf:label>
      <xf:value>green</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Blue</xf:label>
      <xf:value>blue</xf:value>
    </xf:item>
  </xf:select>
  Output: <xf:output ref="/data/MyCode"/>
</body>
```

Loading Data



- The instance data can be:
  - Explicitly written in the form
  - Loaded from an XML file
  - Loaded form a Web Service

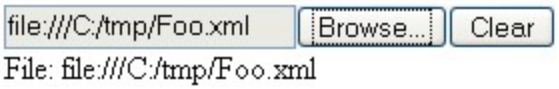
```
<xf:instance src="XMLSchemaTypeCode.xml" id="XMLSchemaTypeCode"/>

<xf:instance id="ApprovalCodes"
    src="/db/mdr/services/all-codes.xq?code=ApprovalCodes&group=editor"/>
```

- Note: large forms with many selection lists ⇒ many HTTP GETs to load the data
  - Solution: we can load all the data at one into a single instance

```
<xf:instance id="ApprovalCodes"
    src="/db/mdr/services/all-codes.xq?form=DataElementManager&group=admin"/>
```

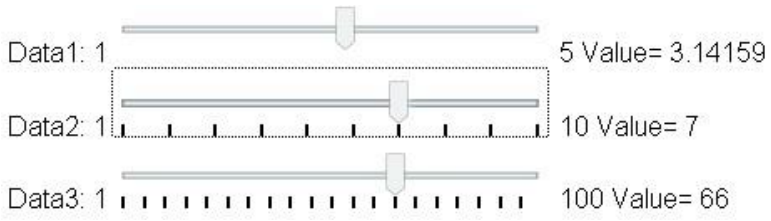
Upload

```
<xf:model>
  <xf:instance xmlns="">
    <Data>
      <File xsi:type="xs:anyURI"/>
    </Data>
  </xf:instance>
</xf:model>
```




```
<body>
  <xf:upload ref="/Data/File">
    <xf:filename>file:///C:/tmp/*.xml</xf:filename>
    <xf:mediatype>text/xml</xf:mediatype>
  </xf:upload>
  <br/>
  <xf:output ref="/Data/File">
    <xf:label>File: </xf:label>
  </xf:output>
</body>
```

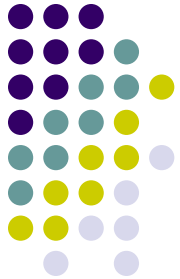
Range




```
<xf:range ref="data1" start="1" end="5"
   step="1" incremental="true">
  <xf:label>Data1: </xf:label>
</xf:range>
<xf:output ref="data1">
  <xf:label> Value= </xf:label>
</xf:output><br />
<xf:range ref="data2" start="1"
   end="10" step="1"
   incremental="true">
  <xf:label>Data2: </xf:label>
</xf:range>
<xf:output ref="data2">
  <xf:label> Value= </xf:label>
</xf:output><br />
<xf:range ref="data3" start="1"
   end="100" step="5"
   incremental="true">
  <xf:label>Data3: </xf:label>
</xf:range>
<xf:output ref="data3">
  <xf:label> Value= </xf:label>
</xf:output>
```

```
<xf:model>
  <xf:instance xmlns="">
    <data>
      <data1>3.14159</data1>
      <data2>6</data2>
      <data3>66</data3>
    </data>
  </xf:instance>
  <!-- we MUST bind all to the decimal type -->
  <xf:bind nodeset="/data/data1" type="xs:decimal" />
  <xf:bind nodeset="/data/data2" type="xs:decimal" />
  <xf:bind nodeset="/data/data3" type="xs:decimal" />
</xf:model>
```

Submit



- The gathered XML data can be:
  - Saved to a file
  - Send to a Web Service
  - Send to a database
- Submission is controlled through two main attributes of element **`submission`**:
  - **`method`** – takes predefined strings
    - e.g., **`get`**
  - **`action`** – takes a URL
    - The scheme of the URL influences how the submission will occur
      - e.g., **`http:`**

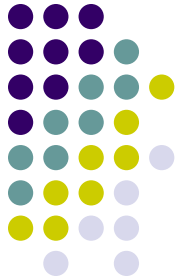
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xf="http://www.w3.org/2002/xforms"
      xmlns:ev="http://www.w3.org/2001/xml-events">
  <head>
    <title>XForms Submit Example</title>
    <xf:model>
      <xf:instance xmlns="">
        <MyData>
          <Data1>One</Data1>
          <Data2>Two</Data2>
          <Data3>Three</Data3>
        </MyData>
      </xf:instance>
      <xf:submission id="save" method="put"
                     action="myData.xml" ref="/MyData"/>
    </xf:model>
  </head>
  <body>
    <xf:submit submission="save">
      <xf:label>Save</xf:label>
    </xf:submit>
  </body>
</html>
```

creates a file containing the instance data

Scenario	Submission element
Small amount of data, no sensitive information and no obligations	`<submission method="get" action="http URI"/>`
Work with a server configured to accept GET requests from HTML forms	`<submission method="get" separator="&amp;" action="http or https URI"/>`
Create new XML data	`<submission method="put" inlcudenamespaceprefixes="" action="http, https, ftp, or file URI"/>`
Feeds XML data into a larger processing system	`<submission method="post" includenamespaceprefixes="" action="http, https, or mailto URI"/>`
Work with a server configured to accept standard POST requests from HTML forms	`<submission method="urlencoded-post" action="http or https URI"/>`
Work with a server configured to accept file uploads from HTML forms	`<submission method="form-data-post" action="http or https URI"/>`
Produces XML data that includes lots of embedded binary data	`<submission method="form-data-post" action="http, https, or mailto URI"/>`
Includes upload controls that accept anyURI data	`<submission method="multipart-post" action="http, https, or mailto URI"/>`

Trigger

```
<xf:trigger>
  <xf:label>Submit</xf:label>
  <xf:action ev:event="DOMActivate">
    <xf:send submission="getTime"/>
    <xf:send submission="getTemperature"/>
  </xf:action>
</xf:trigger>
```

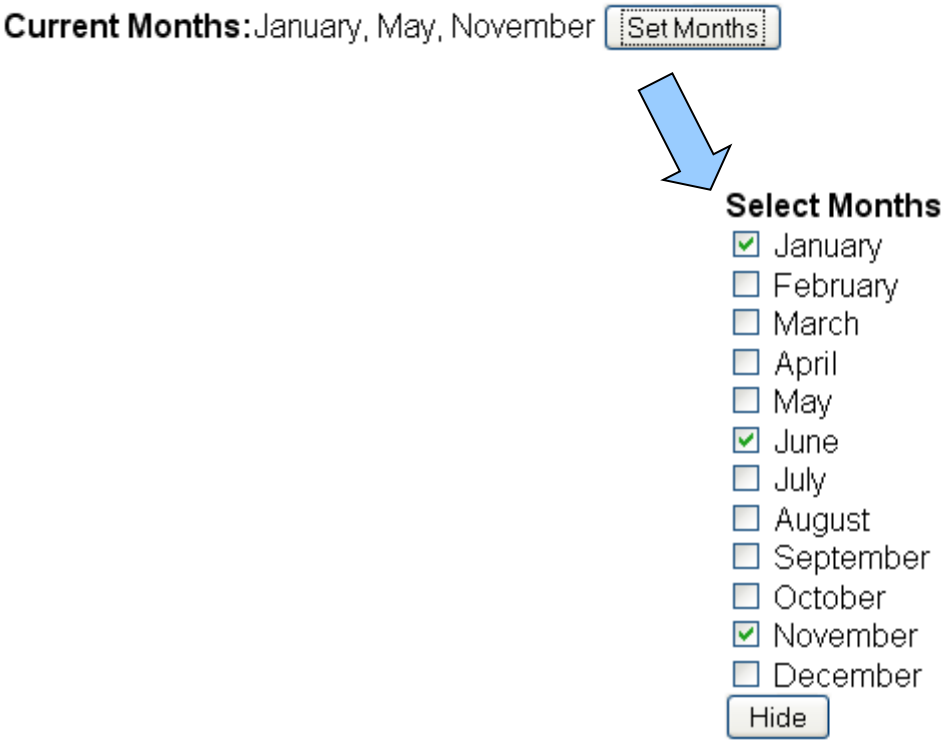


- Generalization of a button
  - Its appearance can be controlled
  - Its actions can be defined

```
<html>
  <head>
    <xf:model>
      <xf:instance xmlns="">
        <data/>
      </xf:instance>
    </xf:model>
  </head>
  <body>
    <xf:trigger>
      <xf:label>Button</xf:label>
      <xf:hint>If you press this you will get a hello world message.</xf:hint>
      <xf:message level="modal"
                  ev:event="DOMActivate">Hello World!</xf:message>
    </xf:trigger>
  </body>
</html>
```

Dynamic Forms

Show/Hide Controls



see `form01.xml`

Dynamic Forms

Dynamic Labels

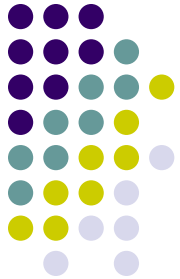
see `form02.xml`

Dynamic Forms

Incremental Model Loading

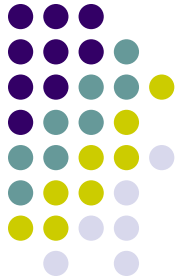
see `form03.xml`

Other Tools in XForms



- Composite controls
- Tables, menus, tabs, …
- Graphs, charts, maps, …
  - Exploitation of libraries
- Validation against XML Schema
- Combination with XSLT, SVG, …
- Exploitation of JavaScript
- Creating custom controls
  - Using existing controls, JavaScript, CSS, …
- …

References



- The XForms Working Group http://www.w3.org/MarkUp/Forms/
- XForms Specification http://www.w3.org/TR/xforms20/
- XForms Tutorial and CookBook http://en.wikibooks.org/wiki/XForms
- XForms Institute – Interactive XForms School http://xformsinstitute.com/
- XForms Implementations http://www.w3.org/community/xformsusers/wiki/XForms_Implementations
- XForms test Suite http://www.w3.org/MarkUp/Forms/Test/
- XForms Quick Reference http://www.w3.org/MarkUp/Forms/2006/xforms-qr