IMPLEMENTING XFORMS USING INTERACTIVE XSLT 3.0

XML Prague 2018

O'Neil Delpratt  Debbie Lockett
oneil@saxonica.com  debbie@saxonica.com
INTRODUCTION
SAXON-FORMS

- New partial XForms implementation for browsers
- Implemented using Saxon-JS technologies: XSLT 3.0 and interactive XSLT
- More than just another XForms implementation: Expands the capabilities of XForms by integration with application logic
WHY ANOTHER XFORMS IMPLEMENTATION?

- Motivation: project to improve in-house form-based application, by using Saxon-JS
- Discover what enhancements are needed to improve capability of Saxon-JS for real-world applications
- Rather than using existing implementations, a new implementation which runs in Saxon-JS allows for better integration within application
IN THIS TALK...

• What makes Saxon-Forms interesting
• A look inside the implementation, and how it works
• How Saxon-Forms can be used to better integrate forms in a form-based application
USE CASE: LICENSE TOOL APPLICATION
DEMO
LICENSE TOOL

- In-house tool for generating and managing product licenses
- Form-based application
- 90% of the code written in client-side or server-side
- XSLT
- Saxon-Forms integrated in the client-side enriched with additional application logic
- XML data end-to-end
SAXON-JS TECHNOLOGIES
SAXON-JS

- XSLT 3.0 runtime processor, in pure JavaScript
- Runs in browser's JavaScript engine
-Executes compiled stylesheet export files (SEFs), generated using Saxon-EE
INTERACTIVE XSLT

Saxon-JS allows interactive web applications to be written directly in XSLT using *interactive XSLT*.

- Extension instructions, functions, modes
- Event handling templates
- Dynamic generation of HTML page content

First introduced with Saxon-CE a few years ago. Further developments with Saxon-JS.
IMPLEMENTING XFORMS
XFORMS INTRODUCTION

- XForms model: instance, bindings
- Section with form controls
- Validation
- Events (Interactive)
WHAT SAXON-FORMS DOES

1. Initialization:
   - Transform the section with form controls into HTML forms elements
   - Behind the scenes: set JavaScript global variables for XForms model and interactive properties (actions, model item properties)

2. Interaction/Event handling:
   - Form data changes, etc.
   - Submission
<html>
  <head>
    <script id="xforms-cache">
      var XFormsDoc;
      var initialInstanceDoc;
      var instanceDoc;
      var pendingUpdatesMap; /* XPath map*/
      var relevantMap; /* XPath map*/
      var actions;

      /*Getter/Setter Functions */

      var setInstance = function(doc) {
        instanceDoc = doc;
      }
    </script>
  </head>
</html>
XSLT CODE TO ADD ACTION TO JSON OBJECT IN JAVASCRIPT SPACE

```xml
<xsl:variable name='action-map' select='map{
    "@ref": "Document/Shipmen/Order/MaintenanceDays",
    "@event": "xforms-value-changed",
    "setvalue": [map{"@value": "if(xs:integer(.) > 0) then ...
                    "ref": "/..//../Options/MaintenanceDate"},
                  map{"value": "true",
                      "ref": "/..//../Options/Updated"}]
} />

<xsl:sequence select='js:addAction("d26aApDhDa", $action-map)'
```

Call JavaScript global function from interactive XSLT by using http://saxonica.com/ns/globalJS namespace
EVENT HANDLING
xsl:template match="input[exist(@data-action)]"
  mode="xsl:onChange"

Update Instance XML

get-Actions(@data-action)
  Execute actions
  (e.g. setvalue, delete, add)

Update associated form control
add to pending update list

Update Instance XML
  - fetch pending update list
  - fetch form changes

Execute XForms Relevant Tests
  - get relevant attributes map
  - filter relevant fields
EVENT HANDLING

<xsl:template match="input[exists(@data-action)]"
  mode="ixsl:onclick">
  <xsl:variable name="refi" select="@data-ref"/>
  <xsl:variable name="refElement" select="@data-element"/>
  ...

  <xsl:variable name="xforms-value-change"
    select="js:getAction(string(@data-action))"/>

  <xsl:variable name="updatedInstanceXML">
    ...
  </xsl:variable>
  <xsl:sequence
    select="js:setInstance($updatedInstanceXML)"/>
</xsl:template>
<xsl:for-each select="$xforms-value-change">
  <xsl:variable name="action-map" select="."/>
  <xsl:variable name="ref"
    select="map:get($action-map, '@ref')"/>

  <!-- if and while clause setup-->
  ...

  <xsl:variable name="instanceXML_Frag" as="node()">
    <xsl:evaluate xpath="$ref"
      context-item="$updatedInstanceXML"/>
  </xsl:variable>
  ...
  <xsl:sequence>
INTEGRATING XFORMS INTO APPLICATIONS
EXAMPLES OF HOW APPLICATION LOGIC CAN BE USED TO DO MORE WITH XFORMS

- Parsing structured text from a form input textarea to XML
- Overriding submission
- User defined functions
PARSING STRUCTURED TEXT TO XML

Email Text

Order #1000 has just been placed using payment method: Credit card
Email: smail@saxonic.com
Comments: GPM-0000

--- Items ---
item_name: Saxon-EE (Enterprise Edition), initial license (ref: EE001)
item_SKU: EE001
item_quantity: 1
item_price: £300.00

item_name: Saxon-EE (Enterprise Edition), additional licenses (ref: EE002)
item_SKU: EE002
item_quantity: 1
item_price: £100.00

--- Order Totals ---
Items: £400.00
Shipping: £0.00
Tax: £0.00
TOTAL: £400.00

--- Payment method ---
Credit card

--- Billing address ---
company: Saxonic
billing_name: O'MeiL Delpratt
billing_street: 1 xxx road
billing_city: Reading
billing_state: Berkshire
billing_postalCode: YYY 70D
billing_countryName: UK
billing_phone: 01180000000

Main Form

License Tool
Paste the license order text into the area below:

Order #1000 has just been placed using payment method: Credit card
Email: smail@saxonic.com
Comments: GPM-0000

--- Items ---
item_name: Saxon-EE (Enterprise Edition), initial license (ref: EE001)
item_SKU: EE001
item_quantity: 1
item_price: £300.00

item_name: Saxon-EE (Enterprise Edition), additional licenses (ref: EE002)
item_SKU: EE002
item_quantity: 1
item_price: £100.00

Select User: Default

Parse the license order text, and (select one of the following):
View in Edit Form
Submit order
SUBMISSION
USING HTTP CLIENT IN SAXON-JS
<xsl:template match="button[@id='submitXML']"
    mode="ixsl:onclick" priority="1">
  ...
  <xsl:variable name="requestBody" as="document-node(element(s)
      <xsl:document>
        <submit>
          <xsl:sequence select="$updatedInstanceXML/Document"/>
        </submit>
      </xsl:document>
    )"
    select="map{'body':$requestBody,
                'method':'POST',
                'href':$receive-orderXML,"}
CONCLUSION
ACHIEVEMENTS OF PROJECT

• Real-world use of Saxon-JS in Saxon-Forms
• Much improved Saxon License Tool Application
SAXON-FORMS ALLOWS YOU TO DO MORE WITH XFORMS

- More than just another XForms implementation
- XForms integrated into declarative client-side applications
- Do more beyond the capabilities of XForms
SAXON-FORMS

• Saxon-Forms is available at https://github.com/Saxonica/Saxon-Forms
• Future goal: Full implementation? (With help from the community)
THANK YOU FOR LISTENING

QUESTIONS?