DISTRIBUTING XSLT PROCESSING BETWEEN CLIENT AND SERVER

XML London, 10 June 2017

O'Neil Delpratt
oneil@saxonica.com

Debbie Lockett
debbie@saxonica.com

SAXONICA.com
XSLT AND XQUERY PROCESSING

SETTING THE SCENE

XSLT PROCESSING: SERVER V CLIENT

SERVER SIDE XSLT

CLIENT SIDE XSLT

Native support, not so much...

- Browsers only support XSLT 1.0
- Many mobile browsers not even that

However...

SAXON-JS

- XSLT 3.0 runtime processor, in pure JavaScript; runs in browser's JavaScript engine
- Executes compiled stylesheet export files (SEFs)
- Distributed processing becomes a viable option

Warning: To disable advertisements look away now. Other XSLT processors are available.

INTERACTIVE XSLT

Saxon-JS not only provides XSLT 3.0 in the browser, but also 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

WHY MOVE PROCESSING CLIENT SIDE?

- Improve speed
- Simplify application architecture
- Remove translations
  between different third-party tools and languages
- Minimise possible failures and incompatibilities
  (e.g. encoding issues)
- XML end-to-end
- Retain some server side processing
  to maintain security of sensitive data and keep data centralised

EXAMPLE WEB APPLICATION

Task: Redesign in-house **License Tool** webapp making use of client side interactive XSLT 3.0

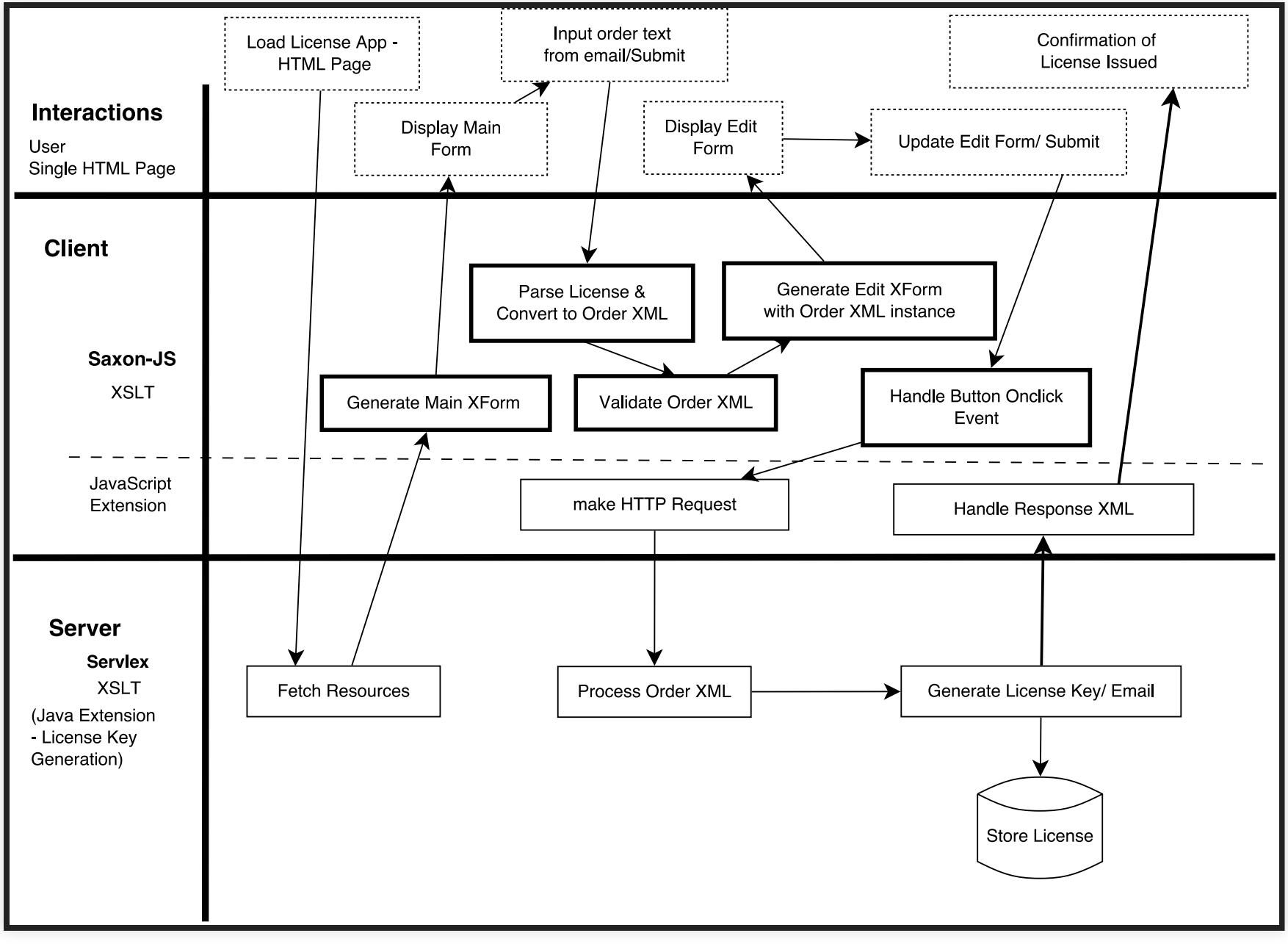
- Generates licenses for Saxon commercial products
- XForms in the front end
- Servlex server side

DEMO

INTERNAL ARCHITECTURE

Flow diagram for redesigned License Tool

- Shows that much of the processing has been moved client side: parsing text input, converting to XML format, validation, generating XForms form


Interactions
User
Single HTML Page
Client
Saxon-JS
XSLT
JavaScript
Extension
Server
Servlex
XSLT
(Java Extension
- License Key
Generation)
Load License App -
HTML Page
Input order text
from email/Submit
Confirmation of
License Issued
Display Main
Form
Display Edit
Form
Update Edit Form/ Submit
Parse License &
Convert to Order XML
Generate Edit XForm
with Order XML instance
Generate Main XForm
Validate Order XML
Handle Button Onclick
Event
make HTTP Request
Handle Response XML
Fetch Resources
Process Order XML
Generate License Key/ Email
Store License

NEW DEVELOPMENTS

We'll now focus on two major areas of technical development, required for the License Tool redesign.

1. XForms implementation
2. HTTP

XFORMS

XFORMS IMPLEMENTATION USING INTERACTIVE XSLT 3.0

(PROTOTYPE)

Replaces previously used **XSLTForms**:

- Limiting XSLT 1.0 implementation in the browser
- Decoding issues in HTTP request

XFORMS IMPLEMENTATION USING INTERACTIVE XSLT 3.0

INTERNALS

- Implemented in XSLT 3.0, runs in browser using Saxon-JS
- XForms model, instance and form controls written in XML document
- Handle events using **interactive XSLT 3.0**
- Instance data held on page.
  XPath 3.1: `json-to-xml()`

BENEFITS

- Maintains XML to XML processing
- Better support on mobile devices
- XSLT 3.0
- Better integration into the whole license tool

HTTP

HTTP REQUESTS USING INTERACTIVE XSLT

- New mechanism - developed since writing the paper
- Uses `<ixsl:schedule-action>`
- New attribute `http-request` will be available with Saxon 9.8 and Saxon-JS 1.0.1 *(coming soon)*

Alternative: call custom global JavaScript function to make HTTP request

IXSL:SCHEDULE-ACTION

Instruction to make an asynchronous call to a named template.

- `wait` - after waiting a specified time
- `document` - after fetching a document
  using an internal, asynchronous "GET" HTTP request
- `http-request` - after receiving a response from a specified HTTP request

EXAMPLE

```
<ixsl:schedule-action http-request="$HTTPrequest">
   <xsl:call-template name="HTTPsubmit"/>
</ixsl:schedule-action>
```

- Specify HTTP request using an XPath 3.1 map
  (More convenient than using EXPath HTTP Client Module `http:request` element)
- Called template handles the response - which is also returned as an XPath 3.1 map

CONCLUSION

- Using interactive XSLT 3.0:
  - Redesign our License Tool webapp
  - Prototype XForms implementation
  - HTTP request instruction

  All client-side in the browser
- Some server-side XSLT processing still required
- XML data end-to-end

THANK YOU FOR LISTENING

QUESTIONS?

SAXONICA.COM
XSLT AND XQUERY PROCESSING