A practical introduction to EXSLT 2.0

XML Prague, March 22nd, 2009

Florent Georges
fgeorges@fgeorges.org

fgeorges.org

XSLT 1.0

- Released as a W3C recommendation on November 11th, 1999
- Great tool to transform XML
- Hard to make complex transform because of Result Tree Fragments
- Missing features: regex, date and time manipulation, dynamic evaluation, user-written functions...

EXSLT 1.0

- Community project launched in the beginning of 2001
- Centralized repository of extensions to XSLT 1.0 for various processors
- Modules: dates and times, dynamic, common, functions, math, random, regular expressions, sets, and strings
- The common module defines the famous exsl:node-set and exsl:document extensions

XSLT 2.0

- Released as a W3C recommendation on January 23rd, 2007
- Includes several features provided by EXSLT 1.0
- Has a much complete function library
- Introduces its own issues and limitations: no first-class functions, no dynamic evaluation, no parsing nor serializing facility, no nested sequences, no ZIP file handling for ODF...

EXSLT 2.0

- In the same spirit of EXSLT 1.0, tries to improve XSLT 2.0
- There is a demand for *“standardized”* extensions to enrich XSLT 2.0 features
- This is a great place to test changes to XSLT 2.0 on a large scale before XSLT Next Generation

And XQuery?

- Most extensions are XPath functions
- XSLT 2.0 and XQuery both build on XPath 2.0
- XQuery processors have interesting extension function libraries, but each different
- EXSLT provides a unique function on all processors
- Ability to write more complex library modules in a processor-independent way

fgeorges.org

Challenges

- Multi-processor implementations
- Extensions must be defined to work in several, different environments
- **How to deliver?**
  - Vital point for a successful EXSLT 2.0
  - I hope XQuery implementers won't reproduce errors from the SQL world, and will understand interoperability benefits

Delivery

- eXist

```
import module namespace ns = "..."
   at "xmldb:exist:///db/.../module.xq";
```

- MarkLogic

```
import module namespace ns = "..."
   at "/on/db/module.xq";
```

- Saxon

```
declare namespace ns = "java:com.sample.JavaClass";
```

fgeorges.org

Stop theory!

...and show me some cool code

HTTP Client

- Send HTTP requests and provide responses
- Based on XProc step p:http-request
- Implementation for Saxon, partial implementations for eXist and MarkLogic
- Enable to query REST services, Google services, Web services, or simply to retrieve resources on the Web

http:send-request()

```
http:send-request($request as element(http:request)) as item()+
```

```
<http:request href="http://www.example.com/..." method="post">
  <http:header name="X-Header" value="some value"/>
  <http:body content-type="application/xml">
    <hello>World!</hello>
  </http:body>
</http:request>
```

```
<http:response status="200" message="Ok">
  <http:header name="..." value="..."/>
  ...
  <http:body content-type="application/xml"/>
</http:request>
```

fgeorges.org

http:send-request()

```
http:send-request(
   <http:request href="http://www.xmlprague.cz/" method="get"/>)

⇒

(
 <http:response status="200" message="OK">
    <http:header name="Server" value="Apache/1.3.37 (Unix)"/>
    ...
    <http:body content-type="text/html"/>
 </http:response>
,
 <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
       <title>XML Prague 2009</title>
       ...
)
```

fgeorges.org

HTTP Client samples

- XQuery samples (Saxon, MarkLogic & eXist)
- GData samples
- WSDL Compiler

ZIP file handling

- Extract entries
- Update entries
- Create new ZIP files
- Well suited for OpenDocument (aka ODF, from OASIS) and Office Open XML (from Microsoft)

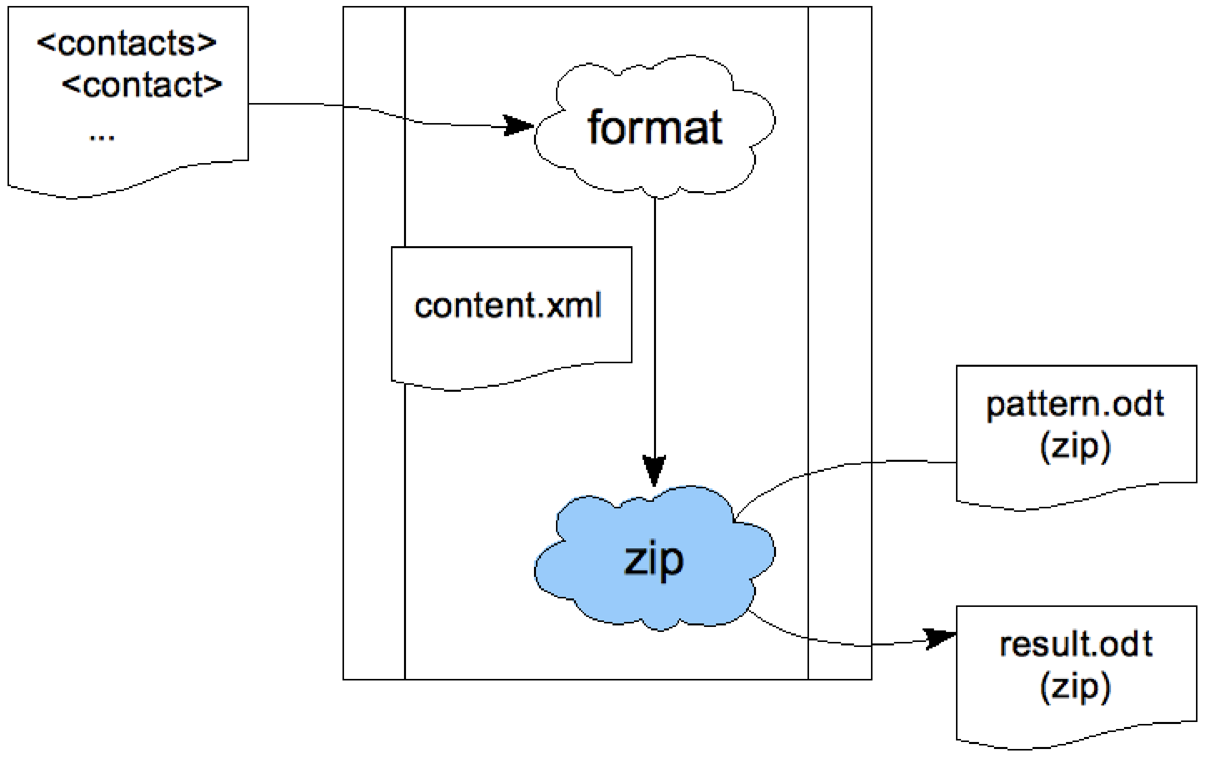
ZIP functions

- zip:xml-entry($href, $path) as document-node()
- zip:html-entry($href, $path) as document-node()
- zip:text-entry($href, $path) as xs:string
- zip:binary-entry($href, $path) as xs:base64Binary
- zip:entries($href) as element(zip:file)
- zip:zip-file($zip) as empty()
- zip:update-entries($zip, $output) as empty()

<zip:file>

```
<zip:file href="some.zip">
  <zip:entry name="file.xml" output="xml">
    <hello>World!</hello>
  </zip:entry>
  <zip:entry name="index.html" output="html" href="/some/file.html"/>
  <zip:entry name="dir">
    <zip:entry name="file.txt" output="text">
      Hello, world!
    </zip:entry>
  </zip:entry>
</zip:file>
```

OpenDocument Pattern

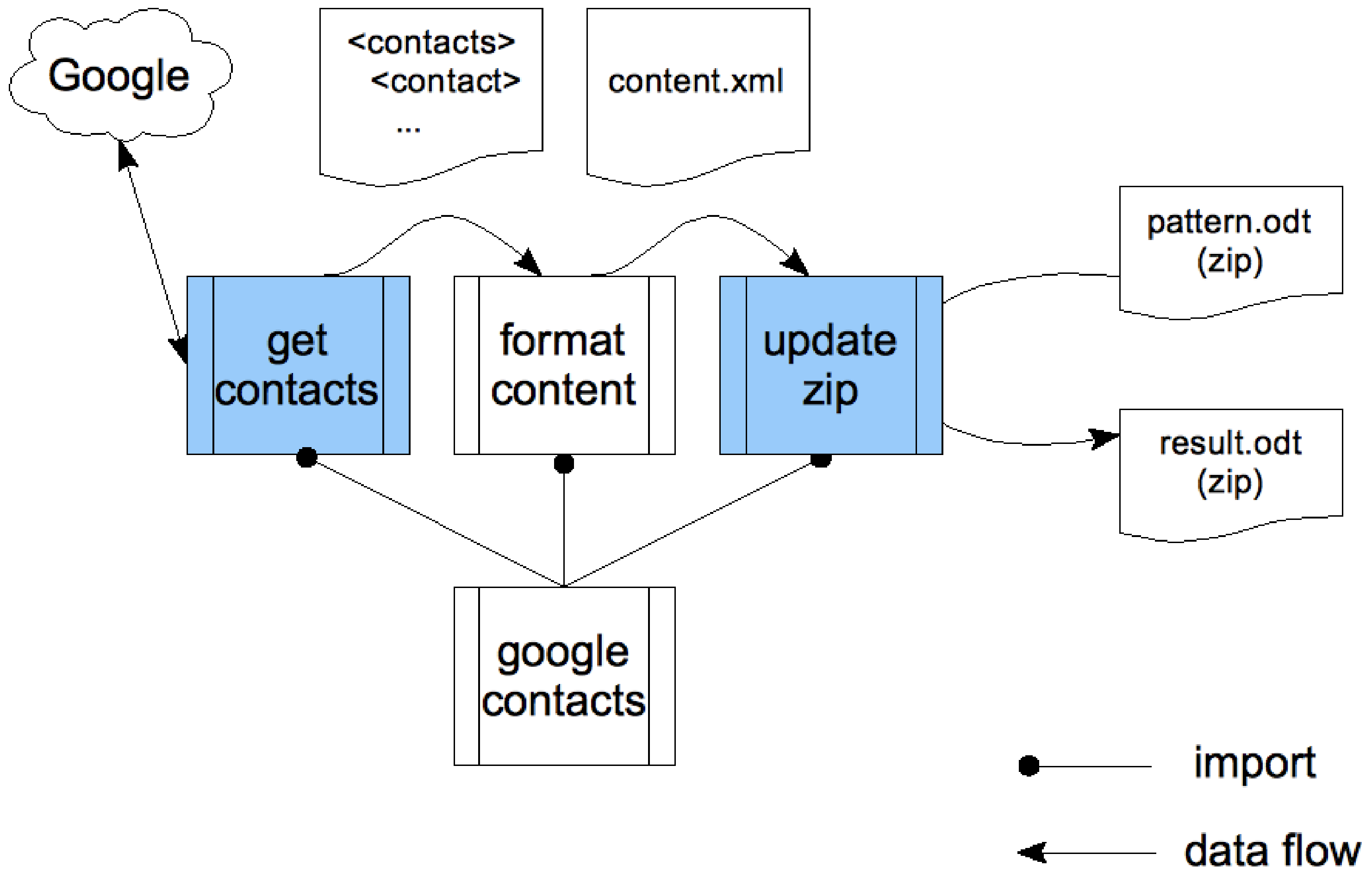


fgeorges.org

Putting it all together

- Google Contacts
  - Retrieve contacts from Google Contacts
  - Retrieve their pictures and maps
  - Format them based on a pattern content.xml
  - Create an ODT file based on a pattern

Google Contacts



fgeorges.org

That's all Folks!

- Plenty of other potential extensions
- More low-level, general purpose: nested sequences and first-class function objects

Join the community!

- http://www.exslt.org/list/
- http://community.zepheira.com/wiki/exslt/
- http://www.fgeorges.org/exslt2/

fgeorges.org