RESTXQ

XQuery 3.0 Annotations for REST

Adam Retter

…Not a MarkLogic Employee or User!

Freelance Consultant

XQuery WebApp Evangelist

eXist Solutions

eXist-db

Almost worked for MarkLogic!

XML Guild Member

W3C XQuery WG

EXQuery

Software Engineer

A revision of my XML Prague 2012 presentation with ++

Agenda

1. The Problem!
2. The Solution?
3. Proof of Concept
4. How to Implement
5. Now and Then

The Problem!

…or, how I ended up here,

and why I care.

The Problem!

What am I trying to achieve?

- **Building Web Applications in XML tech.**

- **Why?**
  - **Good at producing (X)HTML, CSS, JSON, etc.**
  - **Data access is implicit (XPath, doc(), etc)**
  - **Same Type-Model for data and implementation**

    ∴ **Makes development, easy and fast.**

The Problem!

No Standard Web Capability

XPath, XQuery, XSLT and XProc all lack:

- **Standard Web Connectivity**
  - **How to associate a URI with…**
- **Standard Web Context operations**
  - **Cannot process HTTP Request**
  - **Cannot create HTTP Response**
- **Vendors!!!**

The Problem!

Vendors Compete

- **Implement W3C Standards**
- **Create/meet demand by adding value….**
  - **Innovate**
    - **Implementation features**
    - **Language Extensions…**

The Problem!

Vendors Innovate

- Extensions for accessing the Web Context
  - Extension functions e.g. Get HTTP Request Parameter
- Implementation for Web Connectivity
  - Invoking XML processing with a HTTP Request
- Extensions are non-standard
  - Code becomes Non-Portable
  - Vendor Lock-In
  - Contribution back into Standards?

The Problem!
XQuery*

- **Herein we solely focus on XQuery**
- **Lessons and Proposals could equally be applied to XSLT, XProc and others.**

(*not the problem itself!)

The Wheel Factory

…or, XQuery Web App Examples

The Problem!

The Wheel Factory:

- Create a new Wheel
- Store a Wheel in the Warehouse
- Inventory of Wheels in the Warehouse
- Retrieve a Wheel from the Warehouse

```
module namespace wheel = "http://wheelfactory.com";

import module namespace util = "http://exist-db.org/xquery/util";
import module namespace xmldb = "http://exist-db.org/xquery/xmldb";


declare function wheel:create-wheel($wheel-doc as document-node()) as xs:string? {
    let $id := util:uuid() return
        if(xmldb:store("/db/wheels", (),
            <wheel id="{$id}" created="{current-dateTime()}">
                { $wheel-doc/wheel/* }
            </wheel>
        ))then
            $id
        else()
};

declare function wheel:store-wheel($id, $wheel-doc as document-node()) as xs:boolean {
    let $existing-wheel := wheel:retrieve-wheel($id) return
        let $filename :=
            if(not(empty($existing-wheel)))then
              document-uri(root($existing-wheel))
            else()
        return
            not(empty(
                xmldb:store("/db/wheels", $filename,
                    <wheel id="{$id}" created="{current-dateTime()}">
                        { $wheel-doc/wheel/* }
                    </wheel>
                )
            ))
};
```

wheel.xqm (1/2)

```
declare function wheel:inventory() as element(inventory) {
    <inventory>
    {
        for $wheel in collection("/db/wheels")/wheel return
            <wheel id="{$wheel/@id}" created="{$wheel/@created}"/>
    }
    </inventory>
};

declare function wheel:retrieve-wheel($id) as element(wheel)? {
    collection("/db/wheels")/wheel[@id eq $id]
};
```

wheel.xqm (2/2)

The Problem!

The Wheel Factory as a Service

- **Assuming:**
  - **We want REST Services**
  - **We want logical URIs**
- **Require vendor processor support for Web Context**
- **Require function extensions for XQuery**

The Problem!

RESTful HTTP Wheel Factory:

- **Create a new Wheel**
  POST /factory/warehouse/wheel

- **Store a Wheel in the Warehouse**
  PUT /factory/warehouse/wheel/***some-id***

- **Inventory of Wheels in the Warehouse**
  GET /factory/warehouse/wheel

- **Retrieve a Wheel from the Warehouse**
  GET /factory/warehouse/wheel/***some-id***

The Wheel Factory Service

implemented in

eXist-db URL Rewrite

```
xquery version "1.0";

declare namespace exist = "http://exist.sourceforge.net/NS/exist";
import module namespace request = "http://exist-db.org/xquery/request";
import module namespace response = "http://exist-db.org/xquery/response";

import module namespace wheel = "http://wheelfactory.com" at "wheel.xqm";

declare variable $exist:path external;

if(matches($exist:path, "/factory/warehouse/wheel/[0-9A-Za-z\\-]+"))then

    let $wheel-id :=
        replace($exist:path, "/factory/warehouse/wheel/([0-9A-Za-z\\-]+)", "$1")
        return

        if(request:get-method() eq "GET")then
            let $wheel := wheel:retrieve-wheel($wheel-id) return
                if(empty($wheel))then
                    (response:set-status-code(404),
                        <error>wheel not found {$wheel-id}</error>)
                else
                    $wheel

        else if(request:get-method() eq "PUT")then
            if(wheel:store-wheel($wheel-id, request:get-data()))then
                (response:set-status-code(201),
                    response:set-header("Location", concat("/factory/warehouse/wheel/", $wheel-id)))
            else
                (response:set-status-code(500),
                    <error>Could not store wheel. Warehouse is full?</error>)
        else
            (response:set-status-code(405),
                <error>Unknown Method</error>)
```

controller.xql (1/2)

```
else if(matches($exist:path, "/factory/warehouse/wheel"))then

    if(request:get-method() eq "GET")then
        wheel:inventory()

    else if(request:get-method eq "POST") then
        let $wheel-id := wheel:create-wheel(request:get-data()) return
            if(not(empty($wheel-id)))then
                (response:set-status-code(201),
                    response:set-header("Location", concat("/factory/warehouse/wheel/", $wheel-id)))
            else
                (response:set-status-code(500),
                    <error>Could not create wheel. Factory has broken down?</error>)
    else
       (response:set-status-code(405),
            <error>Unknown Method</error>)

else
       (response:set-status-code(400),
            <error>Bad Request</error>)
```

controller.xql (2/2)

The Wheel Factory Service

implemented in

MarkLogic REST Library

```
import module namespace rest="http://marklogic.com/appservices/rest"
    at "/MarkLogic/appservices/utils/rest.xqy";

import module namespace endpoints = "http://wheelfactory.com/rest/endpoints"
    at "endpoints.xqy";

    let $rewrite := rest:rewrite(endpoints:options())
return
  if (empty($rewrite))
  then
    <hello>{xdmp:get-request-path()}</hello>
  else
    $rewrite
```

rewriter.xqy

```
module namespace endpoints = "http://wheelfactory.com/rest/endpoints";

import module namespace rest="http://marklogic.com/appservices/rest"
    at "/MarkLogic/appservices/utils/rest.xqy";

declare function endpoints:options() {
    <options xmlns="http://marklogic.com/appservices/rest">

            <request uri="^/factory/warehouse/wheel/([0-9A-Za-z\-]+)$"
            endpoint="/wheel-proxy.xqy">
                <http method="GET"/>
                <uri-param name="id">$1</uri-param>
                <uri-param name="action">get-wheel</uri-param>
            </request>

            <request uri="^/factory/warehouse/wheel/([0-9A-Za-z\-]+)$"
            endpoint="/wheel-proxy.xqy">
                <http method="PUT"/>
                <uri-param name="id">$1</uri-param>
                <uri-param name="action">put-wheel</uri-param>
            </request>
```

endpoints.xqm (1/2)

```
            <request uri="^/factory/warehouse/wheel$" endpoint="/wheel-proxy.xqy">
                <http method="GET"/>
                <http method="POST"/>
                <uri-param name="action">get-wheel-list</uri-param>
            </request>

            <request uri="^/factory/warehouse/wheel$"
            endpoint="/wheel-proxy.xqy">
                <http method="GET"/>
                <http method="POST"/>
                <uri-param name="action">post-wheel</uri-param>
            </request>
    </options>
};

declare function endpoints:request($module as xs:string) as element(rest:request)?
{
  (endpoints:options()/rest:request[@endpoint = $module])[1]
};
```

endpoints.xqm (2/2)

```
import module namespace rest="http://marklogic.com/appservices/rest"
    at "/MarkLogic/appservices/utils/rest.xqy";

import module namespace endpoints = "http://wheelfactory.com/rest/endpoints" at "endpoir

import module namespace wheel = "http://wheelfactory.com" at "wheel.xqy";

declare option xdmp:mapping "false";

try {
  let $params := rest:process-request(endpoints:request('/wheel-proxy.xqy'))
  return
    let $action := map:get($params, "action") return
        if($action eq "get-wheel")then
            wheel:retrieve-wheel(map:get($params, "id"))
        else if($action eq "put-wheel")then
            wheel:store-wheel(map:get($params, "id"), xdmp:get-request-body())
        else if($action eq "get-wheel-list")then
            wheel:inventory()
        else if($action eq "post-wheel")then
            wheel:create-wheel(xdmp:get-request-body())
        else
            <unknown/>
} catch ($e) {
```

wheel-proxy.xqm

XQuery Web Apps are **beautiful**
…*but* each uses different shaped wheels

Flickr CC-BY: a bike that can only run on special roads by vrogy

The Problem!

There is no Standard

- **Why no W3C Standards?**
  - **XQuery/XSLT/XProc was not envisaged for this task**
  - **Out-of-Scope? *Good* -> focus on Core enablers**
  - **Must see requirement and support…**
- **Standard Proposal**
  - **Initial impetus, individual/organisation research**
  - **Peer Review and grow in the community**
    - **EXQuery/EXPath/W3C Community Group**

The Solution?

…or, an approach that I came up with.

The Solution?

Goals

Build Web Applications in XML technologies:

- **Portable Code**
- **Vendor Agnostic**
  - **Implementer Friendly**
- **Simple for XQuery Developers**
- **Easily Web-enable existing code**
- **Win!**

The Solution?

RESTful Annotations

- **XQuery 3.0 adds Annotations**
  - **Specifies *%public* and *%private***

  "Implementations MAY define further annotations, whose behaviour is implementation-defined"

- **JSR-311: Java Annotations for REST**
  - **We apply these ideas to XQuery 3.0**

The Solution?

RESTful Annotations

- **Functions => Resource Functions**
  - **Constraint Annotations**
  - **Parameter Annotations**

```
xquery version "3.0";

declare namespace rest="http://exquery.org/ns/rest/annotation/";

declare
    %rest:path("/say/hello/{$name}")
function local:say-hello($name) {
    <hello>{$name}</hello>
};

()
```

The Solution?

RESTful Annotations

Path Annotation (Constraint)

- Matching
- Templating
- Auto-Type Conversion (xs:anyAtomicType)

```
%rest:path("/product/{$prod-id}/part/{$part-id}")
```

?

```
my:part-lookup($prod-id as xs:string, $part-id as xs:integer, $sid)
```

The Solution?

RESTful Annotations

HTTP Method Annotation (Constraint)

- Resource Function may have more than one

- Simple Method Annotations

  %rest:GET

  %rest:HEAD

  %rest:DELETE

- Content Method Annotations (optional body)

  %rest:POST %rest:POST("{$body}")

  %rest:PUT %rest:PUT("{$body}")

The Solution?

RESTful Annotations

Media Type Annotations (Constraint)

- HTTP Content Type
  - One or more Internet Media Types

```
%rest:consumes("text/xml", "application/xml")
```

- HTTP Accept

```
%rest:produces("application/atom+xml")
```

- Constraint, so if omitted, then default is */*

The Solution?

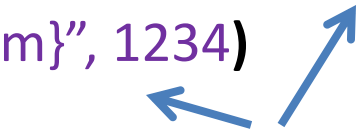
RESTful Annotations

Query String Annotations (Parameter)

```
%rest:query-param("my-field", "{$my-param}")
```

- Optional default value

```
%rest:query-param("my-field", "{$my-param}", "default value")
%rest:query-param("my-field", "{$my-param}", 1234)
```



- Parameter, so if omitted, then default value or ignore

The Solution?

RESTful Annotations

HTML Form field Annotations (Parameter)

```
%rest:form-param("my-field", "{$my-param}")
```

- Optional default value

```
%rest:form-param("my-field", "{$my-param}", "default value")
%rest:form-param("my-field", "{$my-param}", 1234)
```

Auto-Type Conversion!

- Seamlessly extracted from POST or GET
- Parameter, so if omitted, then default value or ignore

The Solution?

RESTful Annotations

HTTP Header Annotations (Parameter)

```
%rest:header-param("my-header", "{$my-hdr}")
```

- **Optional default value**

```
%rest:header-param("my-header", "{$my-hdr}", "default")
%rest:header-param("my-header", "{$my-hdr}", 1234)
```

Auto-Type Conversion!

- **Parameter, so if omitted, then default value or ignore**

The Solution?

RESTful Annotations

Cookie Annotations (Parameter)

```
%rest:cookie-param("my-cookie", "{$my-ckie}")
```

- Optional default value

```
%rest:header-param("my-cookie", "{$my-ckie}", "default")
%rest:header-param("my-cookie", "{$my-ckie}", 1234)
```

Auto-Type Conversion!

- Parameter, so if omitted, then default value or ignore

The Solution?

RESTful Annotations

Serialization; Resource Function returns either:

1. **Resource**

   The result of the XQuery function

2. **rest:response element**

   Serialization settings, HTTP Headers, Status Code and Reason

3. **rest:response and Resource**

   (<rest:response >…</rest:response>, $resource)

The Solution?

RESTful Annotations

Controlling Serialization:

1. **Annotations based on W3C XSLT and XQuery Serialization 3.0**

   %output:method(“xhtml”) ← **static!**

2. **W3C XSLT and XQuery Serialization 3.0 Serialization Parameters in rest:response**

```
<rest:response>
        <output:serialization-parameters>
                <output:method value=“html”/>
        </output:serialization-parameters>
        …
</rest:response>
```

← **dynamic, can override annotation in (1)!**

Proof of Concept

…demo of an implementation

Re-inventing

The Wheel Factory Service

RESTful Annotations

```
module namespace wheel = "http://wheelfactory.com";

import module namespace util = "http://exist-db.org/xquery/util";
import module namespace xmldb = "http://exist-db.org/xquery/xmldb";

declare namespace rest="http://exquery.org/ns/rest/annotation/";

declare
    %rest:POST("{$wheel-doc}")
    %rest:path("/factory/warehouse/wheel")
function wheel:create-wheel($wheel-doc as document-node()) as xs:string? {
    let $id := util:uuid() return
        if(xmldb:store("/db/wheels", (),
            <wheel id="{$id}" created="{current-dateTime()}">
                { $wheel-doc/wheel/* }
            </wheel>
        ))then
            $id
        else()
};

declare
    %rest:PUT("{$wheel-doc}")
    %rest:path("/factory/warehouse/wheel/{$id}")
function wheel:store-wheel($id, $wheel-doc as document-node()) as xs:boolean {
    let $existing-wheel := wheel:retrieve-wheel($id) return
        let $filename :=
            if(not(empty($existing-wheel)))then
              document-uri(root($existing-wheel))
            else()
        return
            not(empty(
                xmldb:store("/db/wheels", $filename,
                    <wheel id="{$id}" created="{current-dateTime()}">
```

RESTful wheel.xqm (1/2)

```
                    { $wheel-doc/wheel/* }
                </wheel>
            )
        ))
};

declare
    %rest:GET
    %rest:path("/factory/warehouse/wheel")
function wheel:inventory() as element(inventory) {
    <inventory>
    {
        for $wheel in collection("/db/wheels")/wheel return
            <wheel id="{$wheel/@id}" created="{$wheel/@created}"/>
    }
    </inventory>
};

declare
    %rest:GET
    %rest:path("/factory/warehouse/wheel/{$id}")
function wheel:retrieve-wheel($id) as element(wheel)? {
    collection("/db/wheels")/wheel[@id eq $id]
};
```

RESTful
wheel.xqm (2/2)

Proof of Concept

Satisfies our Goals:

- **Portable Code**
  - No longer needs vendor extensions!
  - No RESTful Annotation support? Code is still valid!
- **Vendor Agnostic**
  - RESTful Annotations can be implemented by any and all
- **Simple for XQuery Developers**
  - Declarative annotations sit beside your functions
  - Mappings are obvious
- **Easily Web-enable existing code**
  - RESTful Annotations can be added to existing functions

How to Implement

…for vendors

How to Implement

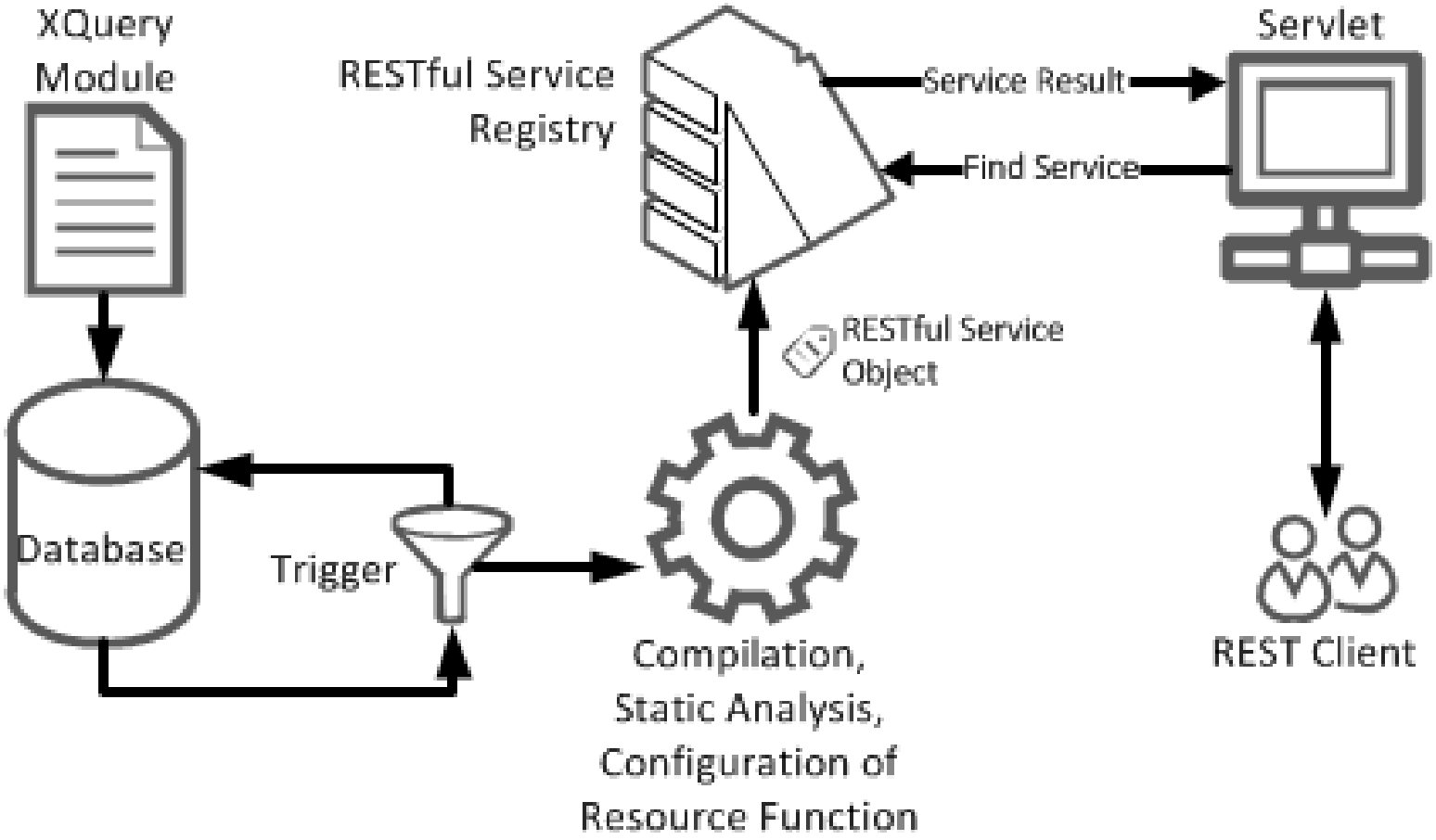
Implementation Options:

Requires Pre-Knowledge of Annotations

- **Adapt XQuery Grammar**
  - Its not part of the W3C XQuery spec (yet)
  - Adapting Grammars and Parsers is hard!
- **Compilation Hook**
  - Compile XQuery on Save/Store, Extract anns. from AST
  - Can still throw Static Analysis Errors
  - W3C XQuery and other Annotations can evolve independently
- **Translate XQuery to Vendor Specific XQuery**

How to Implement

Compilation Hook in eXist-db:



How to Implement

Translate XQuery in MarkLogic:

“going to investigate adding it to the ml-rest-lib project”

- NDW

Now and Then

…status and thoughts

Now and Then

Current Status

- GET/POST/PUT/DELETE/HEAD
- Params: form/query/header/cookie
- Content Negotiation: Media Type
- Serialization, static and dynamic
- Supporting REST XQuery Module

^^ Implemented in eXist-db and BaseX

- %output:method("binary")
- %output:method("json")
- %output:method("html5")

Now and Then

Roadmap

- Matrix Parameters
- Multipart request and response

- Security Annotations

- SOAP Annotations

Questions?

…and maybe answers