XQuery Server Pages

XSP documentation

Getting started

Download the WAR file from the Web site.

You need a tomcat installation (we recommend tomcat 6).

Launch tomcat. Then just drop the WAR file in tomcat's webapps folder, wait for a few seconds until an XSP folder is created. Then try the following URLs with your browser:

http://www.localhost:8080/XSP/admin.xquery

(the administration interface)

http://www.localhost:8080/XSP

(the registration interface - it will not work until you create the database with the former link)

http://www.localhost:8080/XSP/rssfeed.xquery

(an example of RSS feed)

Basic principles

XSP allows you to produce XHTML with XQuery server-side code, just like you would do with PHP or Java EE.

The XQuery engine behind XQIB is MXQuery (http://www.mxquery.org). Hence, most of the MXQuery documentation applies for XSP as well.

Building a Hello World application

Create an XQuery file in webapps/XSP0.6.0, for example helloworld.xquery

```
<html>
  <head>
    <title>Hello, World</title>
  </head>
  <body>
    <h1>{concat("Hello", " World"}</h1>
  </body>
```

```
</html>
```

Then try the following URL in your browser:
http://www.localhost:8080/XSP/helloworld.xquery

Query local (server-side) file

You can use the doc function, for example `doc("db.xml")` when the database of the demo app has been created. You can use the `put("filename.xml")` function to create a new XML file on the server.

Navigating other Web sources

To query other sources from the Web, you can also use the `doc(URL)` function for XML, but the EXPath HTTP library provides much better features. This API is documented at
http://expath.org/modules/http-client/

It returns a sequence of two items:

1) The request metadata, such as header data
2) The payload. Depending on the data type, the payload is presented as XML document, String or Base64 Binary.

For example:

```
import module namespace http = "http://expath.org/ns/http-client";

http:send-request(<http:request       method="get"       href="http://www.tf.uni-freiburg.de/kollektion_test/RSS" />)[2]
```

will return the XML file contains the department's RSS feed.

HTML Scraping

Often, Web content is not in a pure XHTML format. If you direct retrieve content via the EXPath HTTP library, the necessary cleanup is automatically performed.

If HTML is embedded in other content (such inside RSS or JSON), XSP/MXQuery provides an extension function which takes this content and interprets as HTML
http://www.zorba-xquery.com/doc/zorba-latest/zorba/html/util.html

For example:

```
import module namespace zorba-util = "http://www.zorba-xquery.com/zorba/util-
functions";

zorba-util:tidy('<html><head></head><body><b>text<i>ital</b>123</body>'),
```

Likewise, there is actually a standard function fn:parse-xml which takes a string and transforms it into an XDM/XML document node:

```
fn:parse-xml("<root>123<elem>456</elem></root>")
```

JSON

Much relevant content on the web is now being transferred as JSON. There is no standard mapping of JSON to XDM (given the semantic mismatches), but several useful de-facto standards are being used.
MXQuery/XSP (likes Zorba/Sausalito) uses the mapping proposed by John Snelson: http://john.snelson.org.uk/parsing-json-into-xquery.

For example:

```
import module namespace
  json = "http://www.zorba-xquery.com/modules/converters/json";

json:parse('{"type"  : "home", "number": "212 555-1234"}')
```

When JSON is retrieved over the web, EXPath might not properly recognize it as text, as the MIME type is application/json. So sometimes it might be necessary to perform decoding from the Base64 format.

For example

```
import module namespace base64 = "http://www.zorba-xquery.com/modules/base64";
import module namespace json = "http://www.zorba-
xquery.com/modules/converters/json";
import module namespace http = "http://expath.org/ns/http-client";

(: without base64 handling :)
json:parse(http:send-request(<http:request method="get"
href="http://api.twitter.com/1/statuses/public_timeline.json" />)[2])
```

```
(: with base64 handling :)
json:parse(base64:decode(xs:base64Binary(http:send-request(<http:request
method="get"         href="http://api.twitter.com/1/statuses/public_timeline.json"
/>)[2])))
```

Updates, Full text

You can switch updates or fulltext on and off in the WEB-INF/web.xml file, if this becomes necessary. Within the same query, Update and Fulltext operations cannot be performed on the same data, as some internal operations are not fully compatible.

Request Parameters

If you submit a form, input parameters will be sent to the action page. In the action page, you can access them with external variables:

```
declare variable $myparam external;
```

They will be automatically bound, so you can then just use $myparam to access the myparam parameter.

Performing updates and returning a result

As you know, the XQuery Update Facility does not allow you to return a result. In XSP, you can put an XQuery updating file as the action of a form, and redirect the form to a (non-update) XQuery page after the update has been performed with a hidden parameter. For example:

```
<form name="postForm" action="performupdate.xquery" method="post">
  <input type="hidden" name="redirect" value="displayresult.xquery"/>
  …
</form>
```

Alternatively, you can use the XQuery Scripting Extension, which is basically supported by MXQuery.

Samples

The WAR file contains an exercise group registration program as well as an RSS feed example, which are provided as samples to get started.