

Developing a REST interface to a database for OMDoc

<http://kwarc.info/students/projects/OMBase.html>

Andrei Ioniță

Advisors: Prof. Dr. Michael Kohlhase, Christoph Lange, Normen Müller

September 15, 2007

Contents

1	Original project description	1
2	Internship Stages	2
2.1	1st stage : Getting acquainted with the technology	2
2.2	2nd stage : Research on REST	2
2.3	3rd stage : First servlets	3
2.4	4th stage : Implementing XPointer path	3
2.5	5th stage : Intro to eXist	4
2.6	6th stage : Changing to XML documents via XUpdate	4
2.7	7th stage : A proxy servlet that RESTfully processes resources	5
3	Implementation details	6
4	Future work	6
4.1	OMPath	6
4.2	Versioning	7

1 Original project description

The OMBase project is about creating a database for OMDoc documents that is accessible on the web via a RESTful interface. OMDoc [1] is a structured XML-based format for semantic markup of mathematical documents. REST (Representational State Transfer) is a stateless client/server protocol that uses HTTP to perform actions (e.g. view or upload) on resources on a server. Any resource is available through an URL, and actions on resources are performed via the HTTP requests GET/POST/PUT/DELETE.

Originally, it was intended to realize OMBase as a RESTful interface on top of the database that is currently used by SWiM (<http://kwarc.info/projects/swim>), the OMDoc-based semantic wiki that Christoph Lange is developing.

Beforehand it was thought that the RESTful interface will be developed on top of a thin Java layer that wraps OMDoc documents and handles versioning. This layer would directly accesses the database.

The programming language used is Java, as SWiM is a Java project (this compatibility is foremost needed for easy integration). The web interface that handles the above-mentioned URL's is implemented as servlets. As OMDoc is an XML-based document format and parts of these documents should be accessible through the interface, XPath (<http://www.w3.org/TR/xpath>) will be used for querying.

A previous system that achieved a similar goal as above following a different approach and not aiming for being RESTfulness is [2].

2 Internship Stages

2.1 1st stage : Getting acquainted with the technology

I read an introduction to the servlet technology [3] and Java servlet specification [4]. I installed the required software: Java Development Kit v1.6.0 (<http://java.sun.com/javase/>), Eclipse IDE v3.2 (<http://www.eclipse.org/>), Apache Tomcat v5.5 (<http://tomcat.apache.org/>), Apache Ant v1.7.0 (<http://ant.apache.org/>) and read corresponding tutorials, as well as integrated them into the my framework. To access the elements of the document that will be processed, a parser had to be run beforehand. I installed Xerces v1.4.4 (<http://xerces.apache.org/xerces-j/>).

2.2 2nd stage : Research on REST

I gathered information on REST (Representational State Transfer). I wrote a presentation (<http://kwarc.info/students/projects/OMBBase.html>) that would inform the members of the group about the concept. A discussion followed that helped understanding the difference to the SOAP/XML-RPC technologies and go over advantages and disadvantages of using REST. A point was made on what URL syntax is RESTful: is having a parameter after the resource address unrecommended practice, e.g. *resource address?key=value&...*? The following article [5] presents an analysis on the subject. Also, another question was whether the order of the elements that are uploaded/updated matters. Christoph proposed an XPointer-like syntax for the URL's. Using plain XPointer syntax does not work, for the fragment IDs (i.e. sequence after the hash mark) cannot be sent to the server.

In short, a REST system supposes the use of the simple HTTP operations on resources. A resource in the case of our OMDoc database is a self-contained XML fragment (subtree of the original document). The operations are:

- GET: Accessing a resource for viewing
- DELETE: Removing a resource from the database
- PUT: Uploading a new resource. If there is already a resource at the designated position, this will be overwritten
- POST: Modifying a resource (e.g. change the name of an element/attribute; change the content of an element; etc.)

Notice that according to the above, GET and PUT should have a idempotent behavior. The syntax of the URL when using a method should address the location of a resource, e.g. when PUTting an element with the following URL

```
/omdoc[@xml:id="doc"]/theory[@xml:id="thy"]/omtext[@xml:id="text4"]
```

GETting it should be possible using this exact URL.

For further information about REST, see the presentation attached and the bibliography (especially [6]).

2.3 3rd stage : First servlets

I wrote a first servlet interface based on a tutorial [7], in which data was introduced in a form and then the servlet class was called (see MyFirstServlet project). Was given access to the SWiM source code, and kept it as a project in Eclipse.

Originally, using functions from the XOM library (<http://www.xom.nu/>) was recommended, as it focuses on corectness and clearness, as oposed to performance (JDOM). In getting familiar with the XOM, I modified the code provided in the tutorial for a parser [8]. Also tried a serialization [9] of XML documents. (see projects: ParseXML, XOM_OMDocReader). Tried a RPC (Remote Procedure Call) service (see the `xmlrpc_example` project) and identified some differences to REST. A list of these can be found on the presentation slides of REST.

2.4 4th stage : Implementing XPointer path

In a first attempt to construct REST URL's, we decided to use the XPointer element [10] addressing scheme. It essentially uses the index of the children of children for addressing, e.g. `element(group/2/1)` selects the first child of the second child of the element with the id "group".

After beginning to implement the XPointer element path for accessing elements. I did not succeed having URL's that separated the path info from the authority (i.e. resource address). Then we found out that in the `web.xml` file of each project (in the WEB-INF directory) there is a URL mapping element. I changed it from e.g.

```
<url-pattern>/OMDocServlet</url-pattern>
```

to

```
<url-pattern>/OMDocServlet/*</url-pattern>
```

Discovered an inconsistency in querying with XPointer: the XML fragment that is the result of a query does not inherit the namespace of its parent after extraction, and thus it could not be printed as a valid XML document. If the `ContentType` of the response is not set to "text/xml", the code is displayed as text in Mozilla Firefox 1.5.

2.5 5th stage : Intro to eXist

The idea to use the eXist XML database (<http://exist.sourceforge.net/>) as the base for OMBase emerged (Immanuel Normann sent an e-mail to the group list mentioning it, after the discussion from REST presentation). I installed and configured eXist and then read some documentation on its homepage. Used REST with Jetty (a servlet container that came with the eXist installation), and then tried to use it with Tomcat. After suggestions from Christoph I made it working with Tomcat.

I decided to use Apache Tomcat over Jetty, since I had it integrated in Eclipse, while Jetty came along with eXist and was configured to work alongside the latter.

Since the usual web browsers, e.g. Firefox, support only GET (by introducing the address in the designed field) and POST (via a form) the need for using a browser that handles PUT and DELETE was immediate. Christoph pointed me to cURL (<http://curl.haxx.se/>) a command-line HTTP client, which is also used in eXist.

2.6 6th stage : Changing to XML documents via XUpdate

I read and modified the scripts for the REST API of eXist (that are located in the *samples/http/ directory* of the eXist trunk). I understood the mechanism of REST servlet in eXist (the trace of java source code files). Later I found a small bug in the one of the aforementioned scripts (`client.sh`). I learned XUpdate - see link below - (since this is what eXist uses for POST) and went through almost all XUpdate use cases on simple OMDoc documents.

XUpdate is a standard of modifying XML documents. Although its latest update is not recent (<http://xmldb-org.sourceforge.net/xupdate/>) throughout the XML world it is still used. The changes in the eXist XML documents are also made via XUpdate.

Then I encountered problems when querying elements: since all were in the OMDoc namespace, I had to declare the namespace at the beginning of the query, and then write the query itself. For example, instead of

```
//theory/omtext
```

I used

```
declare default element namespace 'http://www.mathweb.org/omdoc';
//theory/omtext
```

I subscribed to the eXist mailing list (<https://lists.sourceforge.net/lists/listinfo/exist-open>) to be up-to-date to the latest issues and in the eventuality of asking questions. Shortly afterwards I discovered a bug related to namespaces: inserting an element with a namespace is possible only with a non-empty prefix. In the case of the OMDoc namespace, the element will appear different from the other elements. Reported this to the eXist mailing list, the issue did not get relevant answers (its relevance to the mainstream eXist development is marginal, however). After tracing the error in the source code (by repeatedly building the eXist.jar file and deploying it into Tomcat) I modified the code by taking the namespace and prefix of the element in which it was inserted (i.e. the parent element). I did not do this for multiple namespaces and prefixes, since only the OMDoc namespace was used, but it should be approached in the future.

2.7 7th stage : A proxy servlet that RESTfully processes resources

Michael and Christoph decided that I should implement a servlet that makes "pure" REST operations, i.e. a servlet having a REST interface, which forwards requests to eXist, which in turn is not completely RESTful I used the apache httpclient (<http://commons.apache.org/httpclient/>) and read the tutorials and API of the methods from its homepage.

GET: in eXist this accesses documents, e.g

```
curl http://localhost:8080/eXistWrapper/Methods/sample.omdoc
```

as well as sets of elements, e.g.

```
curl http://localhost:8080/eXistWrapper/Methods/sample.omdoc/omdoc/theory
```

although according to REST principles (citation needed), individual elements should be possible to be accessed. So a forward of the request it apparently sufficient. However, the query for elements should contain the default namespace, as exemplified above.

DELETE: in eXist the method removes whole documents. In this case a forward was sufficient. For the deletion of elements, a XUpdate request for removal was constructed and passed to the eXist POST method.

POST: An XUpdate file was passed to eXist POST with the necessary modifications.

PUT: PUTting documents was forwarded, while PUTting elements was made via a XUpdate request to eXist POST.

In eXist, the request is forwarded to *eXist-repository-trunk/org.exist.http.servlet.EXistServlet.java* which makes the preparations for the real processing, which begins in *eXist-repository-trunk/org.exist.http.RESTServer.java*

This forwards the requests to be processed to specialized files present in the eXist source directory, according to the method (GET/DELETE/PUT/POST) that is called.

3 Implementation details

The interface is designed to work as follows.

Inside the eXistWrapper.jar archive, eXistWrapper.methods package, there is the Methods class that implements GET, DELETE, PUT, POST RESTfully.

Methods.doGET first extracts the path info, separates the filename of the document from the eventual query. Then it constructs the URL to eXist which contains the declaration of the default namespace for XQuery (as a query URL parameter). An instance of HttpClient is created, which is executed with an instance of the GetMethod class. Then a checking for the successful return to the client is made, and a processing of the response (which needs to be retrieved to clear the workflow client/server, as it is mentioned in httpClient API (citation needed)). Finally the connection is released to allow other requests. For now, we do not mind threading and performance issues.

Methods.doDelete is split into two cases. First, a document can be removed, which means that the eXist URL with the file name will be passed to the eXist DELETE. Secondly, when a resource is removed, a XUpdate removal request is passed to eXist's POST. Then the path info is extracted from the request, it is split into file name and query. In the case of document deletion, the client instance is created and it is executed with the DeleteMethod.

4 Future work

4.1 OMPath

In this version, queries are made by specifying the element name and its id attribute, e.g.

```
/omdoc[@xml:id='doc']/theory[@xml:id='thy']
```

as in OMDoc the id attribute is considered when addressing elements. For the future, other path syntaxes should be accepted, as for example addressing only by id, e.g.

```
/doc/thy
```

Here the point is that the identifiers are unique (can be called "names") and they are locally scoped. A translation between these two addressing syntaxes should be provided. Also a translation to XPointer element may be of help.

4.2 Versioning

A variant of versioning centered on resources would be a further step in establishing the RESTful interface. Already a Management of Change System locator is developed by Normen Müller, which is integrated in the KWARC framework (<http://kwarc.info/publications/baukasten.pdf>) However, the latter system operates on documents, and we have not decided yet whether a (modified) request from **locator** would be efficient to use in realising this task.

References

- [1] Michael Kohlhase *OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2)* Springer Verlag, 2006, available here
- [2] Michael Kohlhase, chapter 26 of [1]
- [3] Servlet tutorial available at <http://java.sun.com/j2ee/tutorial/1.3-fcs/doc/Servlets.html>
- [4] Servlet Specification available at <http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html>
- [5] Hao He, *Implementing REST Web Services: Best Practices and Guidelines* <http://www.xml.com/pub/a/2004/08/11/rest.html>
- [6] Leonard Richardson, Sam Ruby, *RESTful Web Services*, 2007, ISBN 10: 0-596-52926-0, ISBN 13: 9780596529260 <http://proquest.safaribooksonline.com/9780596529260> accessible in the Jacobs University network
- [7] Servlet tutorial for eclipse available at <http://www.java-tips.org/java-tutorials/tutorials/introduction-to-java-servlets-with-eclipse.html>
- [8] <http://www.ibiblio.org/xml/XOM/tutorial.xhtml#d0e417>
- [9] <http://www.ibiblio.org/xml/XOM/tutorial.xhtml#d0e155>
- [10] XPointer element scheme <http://www.w3.org/TR/xptr-element/>