электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.657

ПОДХОДЫ К ПОСТРОЕНИЮ RESTFUL-ПРИЛОЖЕНИЙ НА XQUERY

***Баранова Е.А.**, студент*
кафедры «Системы обработки информации и управления»,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: Балдин А.В., д. т. н., профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Введение. XQuery – язык запросов и функциональный язык программирования, разработанный XQuery Working Group и рекомендованный организацией W3C [2]. Изначально он задумывался лишь как язык запросов к XML данным с SQL-подобным синтаксисом, но на сегодняшний день представляет собой полноценный язык программирования, с помощью которого можно создавать целые Web-приложения.

Удобство использования XQuery на серверной стороне осознают и вендоры, и web-программисты, и группа разработчиков языка XQuery Working Group. Однако до сих пор не существует общепринятого стандарта, описывающего подход к построению RESTful-приложения на XQuery. Каждый из вендоров предлагает свою концепцию, разрабатывает дополнительные расширения для его реализации, а потому программы, написанные на XQuery, оказываются платформо-зависимыми и непереносимыми.

В данной статье будут рассмотрены несколько подходов к построению RESTful приложения на XQuery, предлагаемые тремя вендорами: eXist-db, MarkLogic ил EMC xDB.

В начале следует уточнить, что подразумевается под RESTful приложением.

Representational State Transfer (REST) – стиль построения архитектуры распределенного приложения, описанный и популяризованный в 2000 году Роем Филдингом. В своей докторской работе Рой Филдинг выделил четыре принципа REST: идентификация ресурсов; управление данными посредством их представлений; самодокументируемый формат сообщений; использование гипермедиа для взаимодействия с приложением [1]. Приложения, удовлетворяющие этим ограничениям, называются RESTful-приложениями. REST описывает только принципы построения приложения, абстрагируясь от конкретной реализации, что и привело к существованию разных подходов.

eXist-db.

eXist-db предоставляет три механизма, с помощью которых можно создавать RESTful приложения на XQuery. Первый из них – **REST Server**.

REST Server позволяет обратиться к любому ресурсу, хранящемуся в базе данных, по его URI. Например, GET запрос на URI http://localhost:8080/exist/rest/db/working/order.xml вернет XML документ order.xml, хранящийся в базе данных в коллекции working. С помощью HTTP запросов POST, PUT и DELETE можно манипулировать данными и файлами, например, создавать или удалять ресурсы в базе данных [4].

Варианты вызова XQuery с помощью REST Server:

1. При выполнении GET запроса на получение XML документа, в качестве параметра дописать необходимый XQuery. Например, http://localhost:8080/exist/rest/db/working/order.xml?_query=//suborder
2. Более сложный XQuery запрос можно указать в разделе CDATA простого XML, и отправить его на URL сервера с помощью POST.
3. XQuery скрипты, а так же различные модули, которые в них используются, могут храниться в базе данных, подобно ресурсам. Для вызова и выполнения таких скриптов необходимо обратиться к ним по их URI в GET или POST запросе. Например, http://localhost:8080/exist/rest/db/myQueries/order.xql.

REST Server - мощный механизм для построения RESTful приложений на XQuery. Однако используемые в данном подходе URI должны соответствовать внутреннему представлению данных, что является существенным ограничением и не передает особенности предметной области. Это ограничение ликвидировано во втором подходе, предлагаемом eXist-db – **XQuery URL Rewriting**.

XQuery URL Rewriting фильтрует все HTTP запросы, поступающие к eXist-db, и передает их контроллеру (XQuery-скрипт ‘controller.xql’) соответствующего приложения [5]. Дальнейшая судьба HTTP запроса полностью зависит от контроллера. По мере формирования ответа на запрос, он может вызывать другие XQuery-скрипты, java-сервлеты, накладывать XSLT-преобразования. Внутри скрипта ‘controller.xql’ можно использовать механизм конвеерной обработки: результаты выполнения одного скрипта отправлять сразу на вход другого. Например, можно последовательно накладывать несколько XSLT-преобразований, постепенно превращая XML-документ в красивую HTML страницу. В качестве результата контроллер может вернуть XML-документ, текст, HTML-страницу или даже код ошибки.

Ниже представлен небольшой кусочек кода controller.xql:

```
let $path-group-regexp := '^/?([^/]*)/?(.*)$'
let $params := subsequence(text:groups($exist:path, $path-group-regexp), 2)
return
switch($params[1])
    case("tasklist") return
        <dispatch xmlns="http://exist.sourceforge.net/NS/exist">
            <forward url="{$exist:controller}/modules/tasklist-controller.xql">
                <set-attribute name="tail" value="{$params[2]}"/>
            </forward>
            <view>
                <forward servlet="XSLTServlet">
                    <set-attribute name="xslt.stylesheet"
                     value="xmldb:exist://{$exist:controller}/resources/xsl/list.xsl" />
                </forward>
                <forward servlet="XSLTServlet">
                    <set-attribute name="xslt.stylesheet"
                     value="xmldb:exist://{$exist:controller}/resources/xsl/facade.xsl" />
                </forward>
            </view>
        </dispatch>
    case("interaction") return
        <dispatch xmlns="http://exist.sourceforge.net/NS/exist">
            <forward url="{$exist:controller}/modules/interaction-controller.xql">
                <set-attribute name="tail" value="{$params[2]}"/>
            </forward>
        </dispatch>
    default return
        <dispatch xmlns="http://exist.sourceforge.net/NS/exist">
            <cache-control cache="yes"/>
        </dispatch>
```

XQuery URL Rewriting предоставляет более гибкий подход к построению RESTful приложений, чем REST Server за счёт полного контроля жизненного цикла HTTP запроса, а так же разделения пространства URL приложений и пространства URI базы данных. URL в адресной строке становится более понятным простому пользователю, который

сможет переходить на уровень выше-ниже по приложению, стирая или дописывая части URL. Есть и недостаток у этого подхода: с расширением приложения значительно увеличивается количество строк кода в скрипте “controller.xql”, он становится менее понятным и нечитаемым. Такой скрипт сложно отлаживать, искать ошибки.

В последней версии eXist-db появился еще один механизм, позволяющий создавать RESTful приложения на XQuery [7]. Этот механизм связан с появлением в третьей версии языка XQuery поддержки аннотаций. Аннотации объявляют свойства функций и переменных, которые могут быть использованы при построении логики приложения. Они начинаются со знака «%» и состоят из имени и значения. В стандарте XQuery 3.0 определены только два вида аннотаций - %private и %public [3]. Однако при конкретной реализации XQuery процессора возможно объявление других аннотаций. На этом свойстве основан третий подход, недавно появившийся в eXist-db – **RESTful XQuery Annotations.**

Структурными элементами в данном подходе являются Resource Functions (функции-ресурсы). Это обычные XQuery функции, размеченные с помощью аннотаций. Аннотации указывают процессору, что при поступлении HTTP запроса, удовлетворяющего описанным ограничениям, должна выполниться данная функция, а результат ее выполнения вернуться в качестве ответа на запрос. Пример Resource Functions:

```
declare
%rest:GET
%rest:POST
%rest:path("/tasklist /{$list}")
function local:get-list($list as xs:string) {
    bpm:get-task-list($list) };
```

В качестве ограничений могут выступать аннотации пути “%rest:path(“{$path}”)” – URL, при обращении к которому должна быть выполнена функция, и HTTP методов “%rest:GET”, “%rest:POST”, “%rest:PUT”. Если не указать ни один метод конкретно, то ограничение по методу накладываться не будет, и при соответствующем URL функция выполнится независимо от HTTP метода.

Для получения параметров из GET запроса и передачи их в функцию используется аннотация “%rest:query-param”, в случае POST - “%rest:form-param”. Управлять ответом на запрос можно с помощью аннотации “%rest:output”

Данный подход явился результатом исследований Адама Риттера и группы разработчиков eXist-db, целью которых было предложить унифицированный подход к построению платформо-независимых, портативных XQuery-приложений. В RESTful XQuery Annotations они попытались соединить все плюсы существующих подходов

(предлагаемых не только eXist-db, но и другими вендорами), сделав его при этом простым, понятным и мощным одновременно. Кроме того данный подход может быть легко внедрен в любые другие СУБД, поддерживающие XQuery 3.0, а приложения, написанные подобным образом, станут платформо-независимыми.

MarkLogic.

MarkLogic так же предоставляет три подхода. Первый из них **HTTP App Server**. Основное отличие HTTP App Server от eXist-db's REST Server заключается в разделении пространства хранения данных на Modules и Content. Доступ к ресурсам, хранящимся в Content, возможен только из XQuery скриптов. Ресурсы, хранящиеся в Modules, доступны по уникальному URI, подобно файлам в eXist-db. Однако URI этих документов не отражает внутреннюю логическую структуру коллекций базы данных, а назначаются согласно внутреннему алгоритму.

Для вызова и выполнения XQuery скрипта достаточно выполнить POST или GET запрос на соответствующий URI, например, http://localhost:8060/some-script.xql.

В отличие от eXist-db's REST Server, в HTTP App Server нельзя напрямую взаимодействовать с данными посредством HTTP запросов.

Второй подход - **URL Rewriting**. URL Rewriting настраивается независимо для каждого HTTP App Server. Его можно разрешить или запретить при необходимости. В случае, когда URL Rewriting разрешен, все запросы к приложению обрабатываются XQuery-скриптом “url_rewrite.xqy”. Идея URL Rewriting в MarkLogic гораздо проще, чем URL Rewriting в eXist-db. Так, главная задача “url_rewrite.xqy” - вернуть строковое значение нового URL, на который будет перенаправлен клиент. Однако, как и “controller.xql”, скрипт “url_rewrite.xqy” получается довольно большим, состоящим из множества if-else и регулярных выражений:

```
let $url := xdmp:get-request-url() return
   (: homepage :)
   if(fn:matches($url, "^/$") or fn:matches($url, "^/home.xml$")) then
      "/home.xqy"
   (: login page :)
   else if(fn:matches($url, "^/login$")) then
      "/login.xqy"
      (: user sign-up page :)
   else if(fn:matches($url, "^/register$")) then
         if(xdmp:get-request-method() eq "GET")then
            "/registration-form.xqy"
```

```
        else if(xdmp:get-request-method() eq "POST")then
            "/sign-up.xqy"
        else
            "/nowehere.html"
    else   "/nowehere.html"
```

Справиться с этим и упростить разработку приложения можно, подключив вместо стандартного URL Rewriting другие библиотеки (**XQuery Libraries**), например MarkLogic REST Library или Corona [6]. Как правило, в таких библиотеках используются конфигурационные XML файлы, описывающие отображение URL на XQuery функцияи. Использование подобных библиотек составляет сущность третьего подхода, предлагаемого MarkLogic.

EMC xDB.

EMC xDB предлагает единственный способ выполнения XQuery скриптов в RESTful манере - **xDB REST API** [8]. Подобно REST Server eXist-db, xDB REST API позволяет обращаться к любому ресурсу, хранящемуся в базе данных, по его URI. Однако возможности по выполнению XQuery скриптов менее широкие. Так, в xDB нет возможности заранее создавать XQuery-скрипты, модули и вызвать их в последствии по их URI. Для выполнения XQuery его нужно будет передать в качестве параметра в GET запросе, либо в разделе CDATA XML-документа в POST запросе.

Сравнение подходов. Выводы.

REST Server eXist-db и REST Api EMC xDB – реализуют основные принципы REST архитектуры. У каждого хранящегося в базе файла есть уникальный URI. Данными можно манипулировать с помощью стандартных HTTP запросов. В качестве ответа на запрос могут выступать XML, XHTML, HTML, текстовые документы. Использование гипермедиа в приложениях зависит от разработчика, а не ограничено функционалом предлагаемого API. Однако сравнивая REST Server eXist-db и REST Api EMC xDB предпочтение следует отдать первому подходу, так как возможность выполнять сохраненные в базе или на сервере XQuery-скрипты, а так же подключать модули – значительно упрощает разработку приложения.

URL Rewriting, предлагаемые eXist-db и MarkLogic, выглядят более предпочтительно, чем подходы, рассмотренные выше. Во-первых, из-за их способности разделять множество URI ресурсов и URL приложений. URL адреса при таком подходе оказываются более понятными для пользователей, да и для самих разработчиков. Во вторых, эти подходы позволяют полностью контролировать поступающие к базе данных запросы, управлять реакцией приложений на них. Минусом URL Rewriting у обоих

вендоров является наличие одного «главного» исполняемого модуля – «controller.xql» или “url-rewriting.xqy”. Эти скрипты при построении больших приложений становятся узким местом – сложная логика делает их непонятными и неуправляемыми. MarkLogic решает эту проблему, подключая дополнительные библиотеки. Однако использование подобных библиотек предполагает написание вспомогательных файлов, описывающих отображение URL адресов на XQuery-функции. Таким образом увеличивается число источников возможных ошибок.

Сильно выделяется среди других, подход с использованием аннотаций. Кроме того, что данный подход отвечает всем требованиям REST архитектуры, скрипты получаются лаконичными, понятными, а приложения (в перспективе) платформо-независимыми.

Список литературы

1. Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures, 2000. [электронный ресурс] – Режим доступа: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm – Загл. с экрана.
2. XQuery 1.0: An XML Query Language (Second Edition). [электронный ресурс]. Режим доступа: http://www.w3.org/TR/xquery/ – Загл. с экрана.
3. XQuery 3.0: An XML Query Language. [электронный ресурс]. Режим доступа: http://www.w3.org/TR/2013/CR-xquery-30-20130108/ – Загл. с экрана.
4. REST-Style Web API. [электронный ресурс]. Режим доступа: http://www.exist-db.org/exist/apps/doc/devguide_rest.xm– Загл. с экрана.
5. URL Rewriting and MVC Framework. [электронный ресурс]. Режим доступа: http://exist-db.org/exist/apps/doc/urlrewrite.xml – Загл. с экрана.
6. MarkLogic 6 Product Documentation. [электронный ресурс]. Режим доступа: http://docs.marklogic.com/ – Загл. с экрана.
7. Adam Retter. RESTful XQuery, 2012. [электронный ресурс]. Режим доступа: http://www.adamretter.org.uk/papers/restful-xquery_january-2012.xhtml – Загл. с экрана.
8. EMC xDB 10.2.0 manual. Web client. [электронный ресурс]. Режим доступа: http://developer.emc.com/docs/documentum/xdb/manual/index.html#doc:topic/web_client.html – Загл. с экрана.