УДК 004.4: 004.9: 528.9: 912.43

UDC 004.4: 004.9: 528.9: 912.43

ОБРАБОТКА XML-ДОКУМЕНТОВ В JAVA EE

PROCESSING XML-DOCUMENTS INTO JAVA EE

Суханов Владимир Иванович
д.т.н., доцент

Уральский федеральный университет, Екатеринбург, Россия

Sukhanov Vladimir Ivanovich
Ph.D.(Tech.), docent.

Ural Federal University, Yekaterinburg, Russia

Приводятся рецепты программирования обработки XML-документов в веб-приложениях Java EE на сервере Glassfish v3

The article presents programming ways for XML processing by Java EE 6 and Glassfish v3 server

Ключевые слова: XML, БАЗА ДАННЫХ, ВЕБ-ПРИЛОЖЕНИЕ, СЕРВЕР

Keywords: XML, DATA BASE, WEB-APPLICATION, SERVER

Введение

XML-документы получили широкое применение в различных областях информационных технологий, стали для многих применений индустриальным стандартом хранения, обмена и обработки данных нереляционного типа [1]. Разработчиков информационных технологий привлекает их высокая гибкость, независимость от платформы и хорошая интерпретируемость кода как программно, так и человеком, что придает им черты универсального представления информации.

Разработка приложений для обслуживания манипуляций XML-документов, их схем [3] и запросов [2] в рамках интегрированного веб приложения на платформе Java EE 6 требует привлечения различных инструментов программирования в рамках одного проекта. Мировое сообщество программистов, поддерживающих идеологию открытого кода, накопило большое разнообразие инструментальных средств работы с XML. Однако будучи законченными продуктами для решения локальных задач эти инструменты требуется интегрировать в сложные проекты, необходимые конечным пользователям с использованием веб-интерфейсов. Ниже описываются средства, достаточные для разработки простого

репозитория XML-документов, поддерживающего операции хранения и манипулирования документами, основанные на современной технологии Java EE 6.

Репозиторий XML-документов

Простейший репозиторий может выполнен как реляционная таблица с полями, описывающими метаданные документа, и полем типа XML или Text для хранения самого документа. При этом следует выбирать решение в рамках которого удастся организовать исполнение запросов на языке XQuery [2]. Таким решением с открытым кодом может служить СУБД Sedna XML Database [4], имеющая встроенный процессор исполнения запросов на языке XQuery и программный интерфейс для его использования на языке Java. Хранилище СУБД Sedna является одноуровневой коллекцией документов, именуемых простым именем, и к сожалению, не позволяет в удобной для обработки форме хранить метаданные XML-документов, поэтому целесообразно их хранить в реляционной таблице, например, в СУБД PostgreSQL, а сами документы в Sedna, имея ссылку на них в таблице метаданных PostgreSQL как обычную строку символов – имя записи в Sedna.

Дистрибутив Sedna находится по адресу *http://modis.ispras.ru/sedna/download.html*. Установка выполняется файлом *sedna-3.5.161-bin-linux-x64.sh*. Запуск и подготовка к работе выполняется командами с консоли:

se_gov

– запуск сервера Sedna XML Database;

se_cdb test

– создание базы данных с именем *test* (однократно);

se_sm test

– старт базы данных с именем *test*.

Загрузка документа производится командой консоли

se_term -query "LOAD 'Путь/имя_файла.xml' 'имя_в_БД.xml'" Имя_БД

или из байтового массива *bytesd* в программе с использованием API Java:

```
con = DatabaseManager.getConnection(url, dbname, user, password);
con.begin();
SednaStatement st = con.createStatement();
st.loadDocument(new String(bytesd), docname);
con.commit();
```

Веб-интерфейс [5] для загрузки файла от пользователя с использованием свободной библиотеки Tomahawk for JSF 2.0 из архива *http://www.apache.org/dyn/closer.cgi/myfaces/binaries/tomahawk20-1.1.14-bin.tar.gz* (все файлы *jar* поместить в папку */WEB-INF/lib* проекта) выглядит так. В файл конфигурации `web.xml` добавляются описания фильтров:

```
<filter>
 <filter-name>MyFacesExtensionsFilter</filter-name>
 <filter-class> org.apache.myfaces.webapp.filter.ExtensionsFilter
 </filter-class>
</filter>
<filter-mapping>
 <filter-name>MyFacesExtensionsFilter</filter-name>
 <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
```

Файл index.xhtml:

```
...
  xmlns:t="http://myfaces.apache.org/tomahawk"
. . .
<h:form enctype="multipart/form-data">
          <t:inputFileUpload value="#{ldr.uploadedFile}" id="file" />
          <h:message for="file" style="color: red;" />
```

```
          <h:commandButton action="#{ldr.doUpload}"
value="П е р е д а т ь"/>
        </h:form>
. . .
```

Фрагмент обработчика *doUpload* бина *ldr*:

```
private UploadedFile uploadedFile;
public UploadedFile getUploadedFile() {
        return uploadedFile;
}
public void setUploadedFile(UploadedFile uploadedFile) {
        this.uploadedFile = uploadedFile;
}
. . .
String fileName = uploadedFile.getName();
if (uploadedFile.getContentType().equals("text/xml")) {
   byte[] bytesd = uploadedFile.getBytes();
. . .
}
```

Обработка запросов на языке XQuery

При работе с XML-документами для извлечения нужной информации полезным инструментом является язык XQuery. XQuery является языком запросов, извлекающим требуемую информацию из документов XML [2]. Примером запроса является:

```
for $i in doc("items.xml")/*/item
 let $b := document("books.xml")
 /*/ book[itemno = $i/itemno]
where count ($b) > 10
```

```
return
  <popular-book>
    {
      $i/itemno,
      $i/description,
      <book-count> {count
      ($b)}</book-count>
    }
  </popular-book>
```

Операторы *for* и *let* порождают пару для каждой записи *item* в документе ***items.xml*** и ***books.xml***. В каждой паре переменная ***$i*** связана с книгой, а ***$b*** - с последовательностью, содержащей все продажи для этой книги. Оператор ***where*** извлекает только те кортежи, в которых ***$b*** содержит более десяти записей. Оператор ***return*** для каждой из этих пар генерирует выходной элемент, содержащий номер книги, описание и число продаж.

Для формирования пакета данных на основе запроса можно использовать свободное ПО процессоров XQuery.

Одним из таких процессоров является **Xqilla** version 2.3.0 фирмы Oracle (http://xqilla.sourceforge.net/). Программа позволяет исполнять выражения XQuery и XPath 2 из файла. Программа распространяется под лицензией Apache License, Version 2.0. Полный текст документации доступен по адресу http://xqilla.sourceforge.net/Documentation>. XQilla – библиотека с открытым исходным кодом (ASL2.0) для обработки XQuery с поддержкой новых функций XQuery Update Facility – расширенного языка XQuery. XQilla написана на C++ и включает в себя исполняемую в командной строке оболочку для исполнения запросов к XML-контенту в локальной файловой системе. Библиотека активно развивается и часть продукта поддерживает Oracle, Berkeley DB XML.

Другой консольный процессор [6] **Zorba** XQuery Engine, Version: 2.2.0 распространяется под лицензией Apache License Version 2.0. Устанавливается из репозитория операционной системы. Например, для ОС Linux Fedora 16 процессор **Zorba** устанавливается командой *yum install zorba*. Поддерживает XQuery версии 3.0.

Отличительной особенностью упомянутых процессоров является консольный режим их исполнения и обработка документов, хранимых в отдельных файлах, как и самих запросов. Эти обстоятельства затрудняют интеграцию процессоров в веб-приложения, исполняемые на стороне сервера. Причиной являются возможные коллизии в именах файлов, требуемых запросами, поступающими одновременно на сервер от различных клиентов приложения, что требует блокировок и вызывает замедление реакции сервера. В этом смысле более подходящим инструментом является встроенный в API Sedna процессор обработки запросов XQuery, который работает с документами, хранимыми в этой же СУБД. Для его работы необходима библиотека SR-000225 XQuery API for Java [7].

XQJ API for Sedna – *http://www.cfoster.net/sedna/xqj/* позволяет с использованием внешних переменных задавать исходный документ для анализа, что дает возможность создавать настраиваемые на входной документ запросы, которые в свою очередь можно хранить в отдельной таблице репозитория в СУБД PostgreSQL.

Пусть *xmlinstance.getNameDoc()* содержит имя документа в БД Sedna, который необходимо обработать запросом XQuery, а *xmlquery.getquery()* – текст запроса. Выбор экземпляров соответствующих сущностей на странице пользователя можно организовать с использованием компонент вида:

```
<h:selectOneMenu id="idQuery" required = "true"
  value = "#{ldr.id_Sel_Query}" >
```

```
   <f:selectItems value="#{ldr.querySelector}" />
  </h:selectOneMenu>
```

где свойства, например, кодируются:

```
 public int getId_Sel_Query() {
    return (xmlquery.getIdQuery() == null) ? 1
                 : xmlquery.getIdQuery();
 }
 public void setId_Sel_Query(int id) {
         xmlquery = xmlqueryFacade.find(id);
 }
 public ArrayList getQuerySelector() {
         ArrayList resourceSelector = new ArrayList();
         List<Xmlquery> allRes = xmlqueryFacade.findAll();
         for (int i = 0; i < (allRes.size()); i++) {
             Xmlquery inst = allRes.get(i);
             resourceSelector.add(new SelectItem(inst.getIdResource(),
 inst.toString(), inst.toString()));
         }
         return resourceSelector;
     }
```

С использованных приведенных выше приемов программирования обработчик кнопки выполнения запросов будет иметь следующий вид.

```
  public String doQuery() throws IOException {
      String result = "";
      String query = xmlquery.getquery();
      // Соединение с БД Sedna
      SednaConnection con = null;
      try {
        conn = DatabaseManager.getConnection("localhost",
      "test", "SYSTEM", "MANAGER");
        // Создание исполнительного объекта
```

```
      XQExpression xqe = conn.createExpression();
        xqe.bindString(new QName("docname"),
    xmlinstance.getNameDoc(), null);
        XQResultSequence rs = xqe.executeQuery(xquery);
        // Обработка результата запроса
        while (rs.next()) {
          result = result + "\n" + rs.getItemAsString(null);
        }
        conn.close();
        // Отправка результата клиенту
        ExternalContext context =
    FacesContext.getCurrentInstance().getExternalContext();
        HttpServletResponse response = (HttpServletResponse)
    context.getResponse();
        response.setContentType("application/octet-stream");
        response.setContentLength(result.length());
        String headerKey = "Content-Disposition";
        String headerValue = String.format("attachment;
    filename=\"%s\"", "resultname.txt");
        response.setHeader(headerKey, headerValue);
        // Получение выходного потока для ответа
        OutputStream outStream = response.getOutputStream();
        outStream.write(result.getBytes());
        // Запись ответа в поток
        outStream.flush();
        outStream.close();
        FacesContext.getCurrentInstance().responseComplete();
    } catch (DriverException e) {
            e.printStackTrace();
    }
    return "index";
  }
```

Текст запроса в базе данных в этом случае должен содержать объявление внешней переменной с именем `$docname` как показано в примере:

```
declare variable $docname external;
for $x in doc($docname)//book
  return $x/title/text()
```

Пользователь в ответ на запрос получит приглашение указать имя файла, где будут сохранены результаты выполнения запроса.

Заключение

Приведенные приемы программирования задач обработки XML-документов средствами технологии Java EE 6 с использованием стандартной среды разработки Netbeans и веб-сервера GlassFish, дополнительных пакетов Tamahawk, Sedna и XQJ позволяют разрабатывать репозитории XML-документов для манипулирвания документами в целом, так и выборками нужной массовым пользователям информации с использованием мощного языка запросов на языке XQuery, по своим возможностям значительно превышающим язык SQL для реляционных баз данных. Это обстоятельство делает перспективным использование XML-технологий для организации хранилищ слабо структурированных данных и обмена различной информацией между пользователями без предварительных соглашений о её структуре.

Список литературы

1 Основы использования XML-схем для определения элементов [Электронный ресурс]. - Режим доступа: http://www.ibm.com/developerworks/ru/library/xml-schema/, 18.11.2012

2 XQuery: язык запросов XML. [Электронный ресурс]. - Режим доступа: http://citforum.ru/internet/articles/xqlzxml.shtml, 18.11.2012

3 XML Schema (XML схема) — описание структуры XML-документов. [Электронное издание]. Режим доступа: http://dmitriydenisov.com/xml-xslt/xml-basics/xml-schema.html, 18.11.2012

4 Sedna. [Электронный ресурс]. Режим доступа: http://xqj.net/sedna/, 04.01.2013

5 The Java EE 6 Tutorial, Volume II Advanced Topics . [Электронный ресурс] Sun Microsystems, Inc. 4150 Network Circle Santa Clara, CA 95054 U.S.A. Part No: 820–7628–10 December 2009, 21.11.2012

6 Zorba 2.7.0 Documentation. [Электронный ресурс]. Режим доступа: http://www.zorba-xquery.com/html/documentation, 04.01.2013

7 JSR-000225 XQuery API for Java 1.0 Final Release. [Электронный ресурс]. Режим доступа: http://download.oracle.com/otndocs/jcp/xqj-1.0-fr-oth-JSpec/, 04.01.2013

Работа поддерживается Министерством образования и науки Российской Федерации, ГК № 14.514.11.4002 (шифр 2012-1.4-07-514-0061-007).