УДК 800.92

Б. СУРАНБАЕВ
E.mail. ksucta@elcat.kg

ОСОБЕННОСТИ СЕРВЕРА APACHE С ПОДДЕРЖКОЙ JAVA SERVLET В СОЗДАНИИ ПРОГРАММНОГО ИНТЕРФЕЙСА К БАЗЕ ДАННЫХ

Бул макалада JAVA жана APACHE технологияларын колдонуп, өзүнчө алынган ачкыч сөздөр эле эмес, документтин толук аты боюнча издөө жүргүзүүчү программалык интерфейс түзүү каралган.

В данной статье рассмотрена разработка программы с применением технологий JAVA и APACHE, позволяющей производить поиск интересующей информации в базе данных не только по отдельно взятым ключевым словам, но и полному названию документа.

In article considered the program with application of technologies Java and Apache, allowing to prospect the information in a database not only on separately taken keywords, also the full name of the document.

Главной задачей данной работы было создание программного интерфейса к существующей библиографической базе данных. Применение передовых технологий программирования позволили разработать программу, позволяющую производить поиск интересующей информации в базе данных не только по отдельно взятым ключевым словам, но и полному названию документа. Тестирование программы на массиве из 8366 записей показало, что поиск документа в конце массива занимает 2 минуты 16 секунд. Естественно, что при увеличении количества записей время обработки также будет увеличиваться. Массив данных, на котором проводилось тестирование, является реальной базой данных. Для того чтобы программа могла работать стабильно и с минимальными затратами времени на обработку запроса, нужно использовать ее на компьютере, обладающей большим быстродействием. Тестирование производилось на компьютере с такой конфигурацией: INTEL PENTIUM III, ОЗУ 0.99 Gb, под управлением операционной системы Windows XP 2010 с установленным SP6a. Развитие направления связанного с поиском информации в массивах данных библиотек очень эффективно, так как потребность в этой информации через сеть Интернет возрастает с каждым новым пользователем.

Практическая реализация поставленной задачи показала правильность выбранного подхода. Алгоритм был основан на методе перебора каждой записи всего массива и сравнении введенной строки запроса с полями записи прочитанной из массива. После того как запись и запрос совпали, запись выдается в нужном формате для отображения в браузере. Пока весь массив не будет прочитан последовательно, сессия с пользователем не будет закончена.

Первое что делает программа - это считывает файл настройки db.ini – который находится в папке c:\www\db. В данном файле находятся данные о месте нахождения интересующей базы данных, то есть локальный путь к базе данных. Определив интересующую базу данных и установив ее место нахождения, программа начинает процесс поиска всех удовлетворяющих запросу данных (библиографических описаний).

Программа считывает всю запись в массив, после чего начинается определение места нахождения полей и их длины.

public void dbFileRead(String dbNamePath, PrintStream out, String query) {

Сперва производится инициализация всех переменных используемых при работе процедуры.Первый блок. Переменные для занесения значений полей.

Второй блок. Файловые переменные для перемещения по файлу.

Третий блок. Переменные для работы с данными.

Начало выполнения поиска. Сперва проверяется, то что имеет ли запрос query значение неравное ‘пусто’, если условие выполняется и запрос имеет не нулевое значение устанавливается связь с файлом данных. Начальная позиция чтения равна нулю.

```
if (query != null){
try { RandomAccessFile dbfile = new RandomAccessFile(dbNamePath,"r");
// Цикл чтения файла по маркерам
  while (fPosMarker != dbfile.length()) {
    try { mC++;
        dbfile.seek(fPosMarker);
        dbfile.read(Jumper);
        String jBuf = new String(Jumper);
        JIndex = Integer.parseInt(jBuf,10);
         int b = 0;
```

Прочитав начальный блок из 5 символов говорящий о длине записи он преобразуется из символьного значения в числовое. Затем определяется длина словаря которая равна 12*n, где n – равно количеству заполненных полей в одной записи.

```
// Поиск конца словаря
            while ( b != MD){
             dbfile.seek(fPosMarker+24+MIndex);
              b = dbfile.read();
             MTemp++;
             MIndex = MTemp;
             }
             MTemp= MTemp - 1;
```

Определив конечную позицию словаря производится считывание в массив блока состоящего из данных - метка поля; начальная позиция поля, относительно конца словаря; длинная поля и символах.

```
// чтение Словаря из файла в отдельный массив
            byte Dic[] = new byte[MTemp];
             dbfile.seek(fPosMarker+24);
              dbfile.read(Dic);
// чтение полей данных из файла в массив
          fPosData = fPosMarker+24+MTemp;
           String sDic = new String(Dic);
            int DI2 = 0,
               DI3 = 0,
               DI4 = 0,
               DI5 = 0,
            PNum    = 0, // Номер поля числовой
            PLength = 0, // Длинна поля числовая
            PStart  = 0; // Начальная позиция поля чиловая
```

Получив данные в результате преобразований, это строка, начинает последовательное вычитание метки поля, начальной позиции, размера поля.

```
// сканирование номеров полей  while ( DI2 != MTemp){
             DI3=DI2+3;
             String DStr  = sDic.substring(DI2,DI3);// Номер поля
             DI4=DI3+5;
             String DStr2 = sDic.substring(DI3,DI4);// Начальная позиция
             DI5=DI4+4;
             String DStr3 = sDic.substring(DI4,DI5);// Длинна поля
             DI2=DI2+12;
```

```
PLength = Integer.parseInt(DStr3,10);// Узнаем длинну поля
PStart = Integer.parseInt(DStr2,10);// Узнаем начало поля
PNum = Integer.parseInt(DStr,10);// Код
byte Pole[] = new byte[PLength];
```

Как только первая запись о первом поле разобрана на составляющие проверяется его метка, которая относится или нет к тому списку полей который нас интересует. Если да, то производится считывание его из файла. После того как поле считано над значением поля производится ряд преобразований, таких как, вычитание из поля служебной информации.

Получив все данные которые нас интересовали, создается запись состоящая из нескольких полей.

```
if ( mAvtor == null)  mAvtor=" ";
if ( msAvtor == null) msAvtor=" ";
if ( mName == null)  mName=" ";
if ( mPrinter == null) mPrinter=" ";
if ( mSize == null) mSize=" ";
if ( mKey == null) mKey=" ";
if ( mKaIndex == null) mKaIndex=" ";
if ( mBBK == null) mBBK=" ";
if ( mSeria == null) mSeria=" ";
```

Это собственно самая запись

```
MarcRecord Rec = new MarcRecord( mAvtor,
                                 msAvtor,
                                 mName,
                                 mPrinter,
                                 mSize,
                                 mKey,
                                 mSeria,
                                 mBBK,
                                 mKaIndex);
```

Над данными, которые занесены в запись, осуществляется сравнивание с запросом. Сравнение осуществляется после преобразования к одному регистру, это нужно для того чтобы найти полный список всех имеющихся записей относящихся к введенному запросу.

Затем распознанные данные сравниваются с запросом, сравнивание производится только с несколькими полями. Список полей приведен ниже:

100 – Автор
700 – Второй автор
245 – Название произведения
653 – Ключевые слова

```
if ( p01.indexOf(q) != -1 ||
     p02.indexOf(q) != -1 ||
     p03.indexOf(q) != -1 ||
     p04.indexOf(q) != -1)
     { mE++;
```

При совпадении запись сразу же отправляется браузеру для отображения в читабельной для пользователя форме.

```
out.println("<table width=\"461\" border=\"0\" cellpadding=\"0\" cellspacing=\"0\">"+
"<tr bgcolor=\"#3399FF\">"+
"<td colspan=\"3\" class=\"text\">   Автор: "+
"<font color=\"#000000\">"+
Rec.rAvtor+" "+
Rec.rsAvtor+
"</font></td></tr><tr>"+
```

```
"<td colspan=\"3\" valign=\"top\" class=\"bodytext\">"+mE+". "+mC+
" <b>Название:</b> "+
Rec.rName+"<br>"+
Rec.rPrinter+" "+
Rec.rSize+"<br>"+
Rec.rBBK+" "+
Rec.rKaIndex+" "+
Rec.rSeria+
"</td></tr></table>");
}
```

В конце обработки одной записи независимо соответствовала она запросу или нет производится переход к следующей записи.

```
fPosMarker = fPosMarker+JIndex;
MTemp = 0;
MIndex = 0;
}
```

В случае ошибки (исключительной ситуации) цикла обработки записи, прерывается и выдается сообщение об ошибке.

```
catch (IOException e) {
        out.println("Ошибка!!!"+"<br>");
        done=true; }
  }
 }
```

Если же файл отсутствует то программа выдаст сообщение о том что файл базы данных отсутствует на сервере.

```
 catch (IOException e) { out.println("Ошибка доступа к "+dbNamePath); }
  }
 if (mE == 0) {
  out.println("Запос: "+query+" не найден");
} // end If
}
```

После того как проведено сравнение запроса и данных имеющихся в поле, при совпадении запись преобразуются в формат HTML. Преобразуются только несколько полей. Список полей которые выдаются по запросу в случае совпадений приведен ниже:

100 – Автор
700 – Второй автор
245 – Название произведения
490 – Серия
91 – Индекс ББК
90 – Каталожный индекс
260 - Издательство
300 – Объем, размер
653 – Ключевые слова

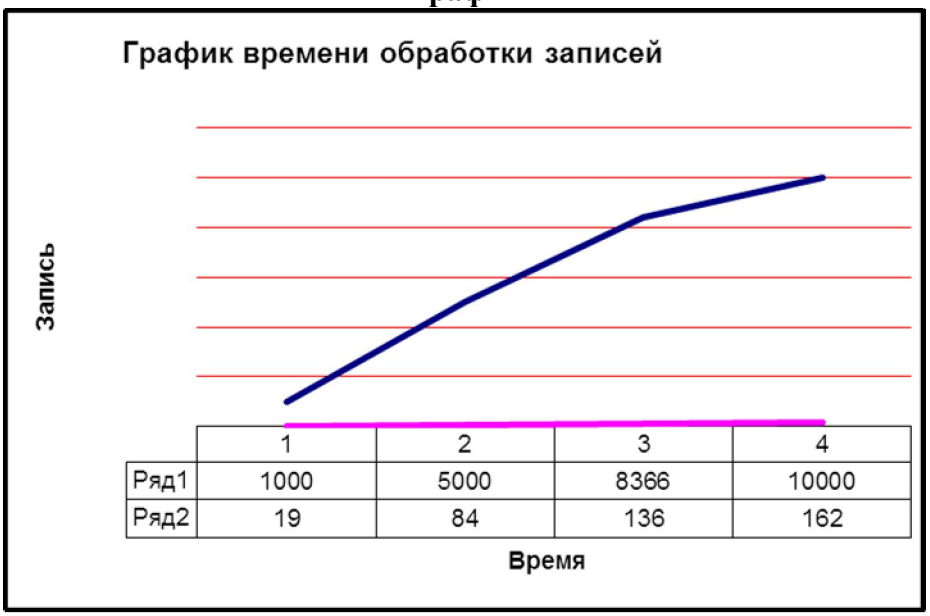
Код вывода в HTML формате выглядит так:

```
out.println("<table width=\"461\" border=\"0\" cellpadding=\"0\" cellspacing=\"0\">"+
"<tr bgcolor=\"#3399FF\">"+
"<td colspan=\"3\" class=\"text\">   Автор: "+
"<font color=\"#000000\">"+
Rec.rAvtor+" "+
Rec.rsAvtor+
"</font></td></tr><tr>"+
"<td colspan=\"3\" valign=\"top\" class=\"bodytext\">"+mE+". "+mC+
" <b>Название:</b> "+
```

```
Rec.rName+"<br>"+
Rec.rPrinter+" "+
Rec.rSize+"<br>"+
Rec.rBBK+" "+
Rec.rKaIndex+" "+
Rec.rSeria+
"</td></tr></table>");
```

После чего программа производит считывание и обработку следующей записи. Тестирование программы проводилось на 4 массивах данных размером 1000, 5000, 8366, 10000 записей. Метод тестирования заключался в поиске последней записи массива по уникальному названию, чтобы определить время поиска по всему массиву. Полученные данные приведены в таблице и на основе данных построен график.

График



	1	2	3	4
Ряд1	1000	5000	8366	10000
Ряд2	19	84	136	162

Список литературы

1. Джейсон Мейнджер. Java: основы программирования: Пер. с англ. [Текст] /Джейсон Мейнджер. - К.: Издательская группа BHV,1997. - 320с.
2. http://www.java.sun.com
3. http://httpd.apache.org/dist/httpd/binaries/win32/old/apache_1_3_14_win32_r2.exe
4. http://java.apache.org/jserv/dist/ApacheJServ-1.1.2-2.exe
5. http://www.apache.org