基于 Ajax+Servlet 的自动完成搜索框实现

唐振坤

2011-11-02

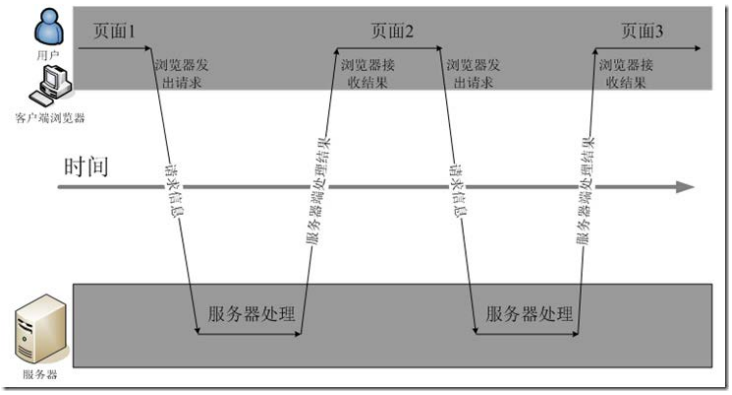
1. 实现目标



面对当前信息爆炸的时代，搜索已经成为了一项基本的生存技能。关键词搜索能够简化用户搜索流程，让用户获得方便快捷的搜索结果，已经被广泛地用于各类需要搜索的场合中。搜索框自动完成功能，在关键词搜索的基础上，进一步提供友好的用户体验，受到如今各类网站的格外青睐，已经成为了当今建搜索站标配功能。

2. 实现原理

实现后台使用 Java servlet，前台使用 Ajax 进行调用，有关 Ajax 的原理见下图：



个人觉得 Ajax（Asynchronous JavaScript）最大的创新之处是在异步这一特点，在传统的网页 HTTP 请求下需要一次请求、一次刷新，针对整个页面，而采用 Ajax 异步调用后，可以使用 Javascript 直接对网页的某些部分区域进行请求，而返回的结果再使用 Javascript 进行解析并处理。

在这个实现里面，自动完成搜索功能就是使用 Ajax 实现，其流程大致如下：

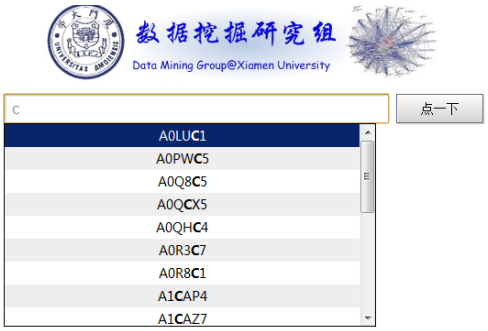
1、 用户在搜索框中输入关键字（部分）；

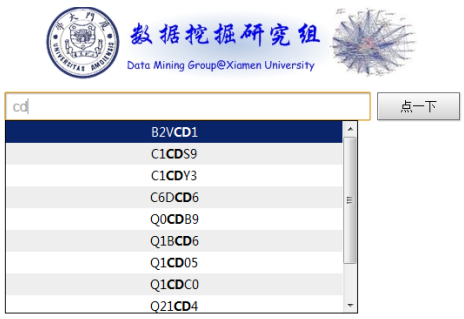
2、 虽然用户并没有点“搜索”按钮，但 Javascript 此时已经将用户输入的部分关键字传送至服务器；

3、 此时服务器针对搜索关键字返回查询结果；

4、 Javascript 接收到服务器返回结果，显示在搜索框下面的区域中作为提示。

这个过程一直重复，直到用户输入完成。





3. 具体内容

页面部分：

一般来说，由于 Ajax 受浏览器的不同而实现各有差异，因此这里采用已经封装良好的 jQuery 来实现 Ajax 调用，当然 jQuery.AutoComplete 也被直接拿来了，后者是基于前者实现的一个插件，有封装好的自动完成功能哦。

因此使用 jQuery.AutoComplete 之后，如下为 html 搜索框：

```
<input type="text" name="wd" id="kw" maxlength="100" />
```

而在页面的<script>段写入：

```
$("#kw").autocomplete("dataservice", {
        max : 20,
});
```

这里面的意思调用请求 url 为 dataservice，最多显示 20 条。页面部分就这么简单了。

Servlet 部分：

接下来的是服务器代码部分，这里我创建了一个 servlet，注册在 dataservice 的 url 请求上。如下即为 web.xml 中该 servlet 的配置部分：

```
<servlet>
    <servlet-name>dataservice</servlet-name>
    <servlet-class>
        cn.xmu.edu.dm.tangzk.autocomplete.servlet.DataService
    </servlet-class>
```

```
</servlet>
<servlet-mapping>
    <servlet-name>dataservice</servlet-name>
    <url-pattern>/dataservice</url-pattern>
</servlet-mapping>
```

其中的 url-pattern 就是写在页面脚本中的请求 url 位置。为什么一个加斜杠，另一个不加，这可以去参考相关 Java Servelet 的内容。

再下来肯定就是 servlet 类的实现了。由于 AutoComplete 仅调用 Get 请求，并且传递参数“q”和“limit”，前者为查询关键字，后者为限制显示记录数。所以 servlet 类中的仅写了 doGet 方法，如下：

```
45    public void doGet(HttpServletRequest request, HttpServletResponse response)
46        throws ServletException, IOException {
47      response.setContentType("text/html");
48      String kw = request.getParameter("q").toUpperCase();
49      String strmax = (request.getParameter("limit"));
50      int max = 20;
51      if (strmax != null && strmax.trim().length() > 0) {
52        max = Integer.parseInt(strmax);
53      }
54      PrintWriter out = response.getWriter();
55      List<String> strArr = DataProvider.getData(kw, max);
56      for (String str : strArr) {
57        out.println(str);
58      }
59      out.flush();
60      out.close();
61    }
```

方法中的第 48-53 行是获取 AutoComplete 传递的参数 q 和 limit，第 55 行是获取数据来源，我这里使用 List 进行封装，然后在第 56-58 行则是输出所获取的数据结果，AutoComplete 对请求服务器的响应要求是每一行一个结果即可。

数据源部分：

再来看看我的数据源 DataProvider 类，有点丑陋，不过好处是可随意扩充。

```
9  public class DataProvider {
10
11    private static List<String> data = null;
12
13    /**
14     * 读取数据，来源可为数据库、文件或是其他内容
15     */
16    static {
17      data = new ArrayList<String>();
18      String path = DataProvider.class.getResource("/data.dat").getPath();
19      Scanner cin;
20      try {
21        cin = new Scanner(new File(path));
22        while (cin.hasNext()) {
23          data.add(cin.next());
24        }
25      } catch (FileNotFoundException e) {
26        // TODO Auto-generated catch block
27        e.printStackTrace();
28      }
29    }
```

这里将数据读取放在 static 块中，在类加载的时候就读取整个数据源，这里是从文件 data.dat 中读取，当进行扩充时可在这里指定使用其它方式，利用在这里读取的 data，就有下面的 getData 了。

```
45    public static List<String> getData(String kw, int max)
46        throws FileNotFoundException {
47      List<String> list = new ArrayList<String>();
48      for (String str : data) {
49        if (str.indexOf(kw) >= 0) {
50          list.add(str);
51        }
52      }
53      return list;
54    }
```

getData 是获取实际数据的方法，其指定参数 kw 和 max 来自于 servlet 中获取的请求，在这个地方进行过滤后，返回至 servlet。这里我使用的是 indexOf 子串查询方式，即出现关键字就认为匹配，加入到返回结果中。

这样，把三大部分结合起来就完成了 AutoComplete 功能，页面 Ajax 在输入时进行异步请求，servlet 接收请求后获取相应的数据，数据源部分向 servlet 提供匹配的数据，最后在 servlet 返回后由 Ajax 显示在页面上。

4. 总结

本文基于 jQuery AutoComplete 和 Java Servlet 实现了一个简单的并且易于扩充的自动完成搜索框，此实现基本能满足中小数据规模的查询请求。当然问题也是存在的，比如直接将所有数据源读入至内存、页面请求频繁更新等，这都是需要根据应用的规模再进行细节调整的地方。