

JS Ajax 和 jQuery Ajax

今日内容介绍

- ◆ 案例：异步用户名校验
- ◆ 案例：异步自动填充
- ◆ 案例：省市联动

今日内容学习目标

- ◆ 使用 jQuery 可以发送 ajax 请求
- ◆ 将 Java 对象转换成 JSON 数据
- ◆ 使用 jQuery 处理 JSON 数据

第1章 案例：异步用户名校验

1.1 案例介绍

在实际开发中，完成注册功能前，如果用户填写用户名，准备填写其他信息时，将提示当前用户的用户名是否可用。效果图如下：



The figure consists of two screenshots of a "User Registration" form. Both screenshots show a "Username" input field containing "jack". In the top screenshot, a red box highlights the text "用户名不可用" (Username unavailable) next to the input field, indicating that the username is already taken. In the bottom screenshot, a green box highlights the text "用户名可用" (Username available) next to the input field, indicating that the username is free.



1.2 相关知识

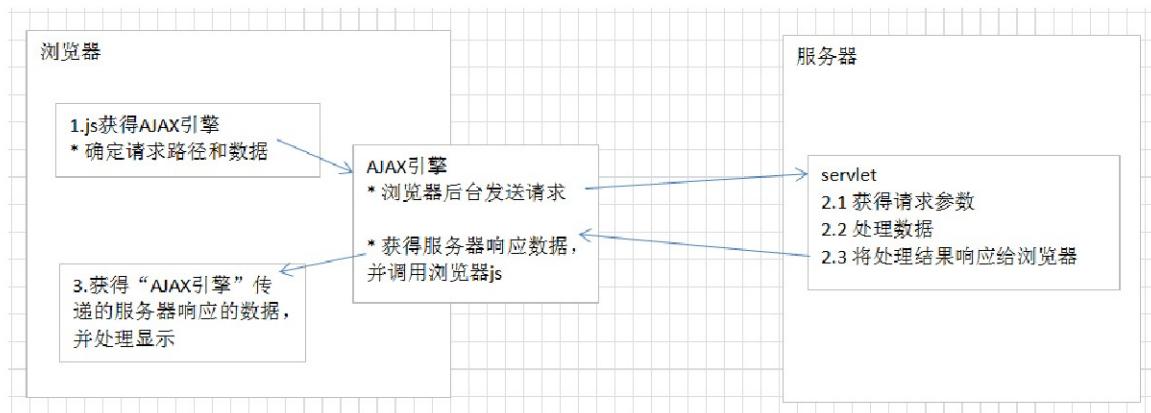
1.2.1 AJAX 原理

1.2.1.1 介绍

AJAX 即 “Asynchronous Javascript And XML”（异步 JavaScript 和 XML）可以使网页实现异步更新，就是不重新加载整个网页的情况下，对网页的某部分进行更新（局部刷新）。传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

AJAX = 异步 JavaScript 和 XML，是一种新的思想，整合之前的多种技术，用于创建快速交互式网页应用的网页开发技术。

1.2.1.2 AJAX 原理分析



- 1.1 使用 JavaScript 获得浏览器内置的 AJAX 引擎（`XMLHttpRequest` 对象）
- 1.2 通过 AJAX 引擎确定请求路径和请求参数
- 1.3 通知 AJAX 引擎发送请求
- AJAX 引擎会在不刷新浏览器地址栏的情况下，发送请求
- 2.1 服务器获得请求参数
- 2.2 服务器处理请求参数（添加、查询等操作）
- 2.3 服务器响应数据给浏览器
- AJAX 引擎获得服务器响应的数据，通过执行 JavaScript 的回调函数将数据传递给浏览器页面。
- 3.1 通过设置给 AJAX 引擎的回调函数获得服务器响应的数据
- 3.2 使用 JavaScript 在指定的位置，显示响应数据，从而局部修改页面的数据，达到局部刷新目的。



1.2.1.3 JavaScript AJAX 使用（了解）

- 原生态 JS 操作 ajax 步骤
 1. 获得 ajax 引擎
 2. 设置回调函数
 3. 确定请求路径
 4. 发送请求
- JavaScript ajax 处理 GET 和 POST 请求有细微差异，接下来简单介绍
- 提供处理程序 HelloServlet，并分别实现 doGet 和 doPost 两个方法

```
<servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>cn.itcast.demo01.HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/helloServlet</url-pattern>
</servlet-mapping>
```

1.2.1.3.1 GET 请求

- 步骤 1：编写 servlet，doGet()方法获得数据，并发送响应数据

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //1 获得数据
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    //2 处理
    System.out.println(username + " : " + password);
    //3 响应数据
    response.getWriter().print("get success");
}
```

- 步骤 2：在 demo01.jsp 提供按钮，点击发送 get 请求。（注意：只能谷歌或火狐浏览器使用）

```
<script type="text/javascript">
    function sendGet(){
        //1 获得 ajax 引擎
        var xmlhttp = new XMLHttpRequest();
        //2 设置回调函数
        xmlhttp.onreadystatechange = function(){
            if(xmlhttp.readyState == 4){
                if(xmlhttp.status == 200){
```

```
        alert("响应数据" + xmlhttp.responseText);
    }
}
};

//3 确定请求方式、路径及参数
xmlhttp.open("GET","/day15/helloServlet?username=jack&password=1234");
//4 发送请求
xmlhttp.send(null);
}

</script>
<input type="button" onclick="sendGet()" value="get 请求"/> <br/>
```

1.2.1.3.2 POST 请求

- 步骤 1：编写 servlet, doPost()方法，用于接收数据和发送响应数据

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                     throws ServletException, IOException {
    //0 处理乱码
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    //1 获得数据
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    //2 处理
    System.out.println(username + " : " + password);
    //3 响应数据
    response.getWriter().print("post 成功");
}
```

- 步骤 2：在 demo01.jsp 提供按钮，点击发送 post 请求。（注意：只能谷歌或火狐浏览器使用）

```
<script type="text/javascript">
    function sendPost(){
        //1 获得 ajax 引擎
        var xmlhttp = new XMLHttpRequest();
        //2 设置回调函数
        xmlhttp.onreadystatechange = function(){
            if(xmlhttp.readyState == 4){
                if(xmlhttp.status == 200){
                    alert("响应数据" + xmlhttp.responseText);
                }
            }
        };
        //3 确定请求方式、路径及参数
        xmlhttp.open("POST","/day15/helloServlet");
    }
</script>
```



```
//4 设置请求编码
xmlhttp.setRequestHeader
    ("content-type", "application/x-www-form-urlencoded");
//5 发送请求
xmlhttp.send("username=杰克&password=1234");
}
</script>
<input type="button" onclick="sendPost()" value="post 请求"/> <br/>
```

1.2.1.3.3 XMLHttpRequest 对象浏览器兼容

```
function getXMLhttp(){
    var xmlhttp=null;
    // 谷歌、火狐、IE 最新浏览器
    if (window.XMLHttpRequest) {
        xmlhttp=new XMLHttpRequest();
    }else if (window.ActiveXObject){
        //IE 老版本浏览器 (IE6、7、8 等)
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

1.2.2 \$.post() 和\$.get()

- **\$.post()** 以 post 请求方式发送 ajax

格式: `jQuery.post(url, [data], [callback], [type])`

参数 1: url, 请求路径

参数 2: data, 请求参数

参数 3: callback, 回调函数

参数 4: type, 返回内容格式, xml, html, script, json, text, _default.

服务器响应编码为: application/json; charset=UTF-8, 回调函数 data 类型是 json 对象

服务器响应编码为: text/html; charset=UTF-8, 回调函数 data 类型是字符串。

- **\$.get()** 以 get 请求方式发送 ajax

除了请求方式不同, 使用方式与\$.post()完全一致。

- 测试:

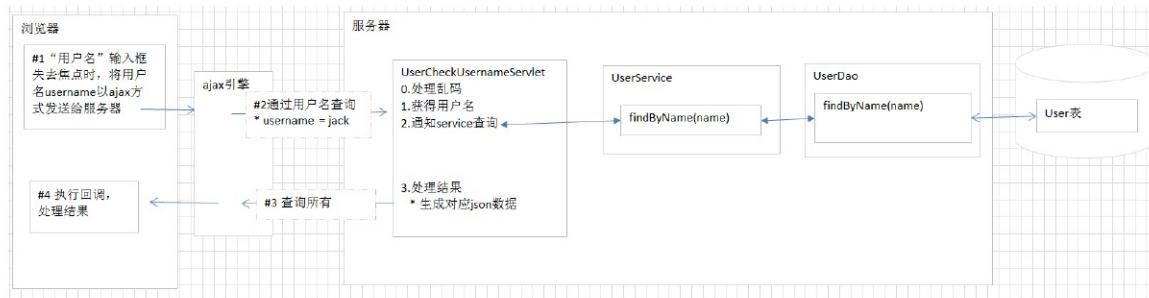
```
<script type="text/javascript">
```



```
src="\${pageContext.request.contextPath}/js/jquery-1.11.3.min.js"></script>
<script type="text/javascript">
$(function(){
    //get 请求
    $("#getId").click(function(){
        var url = "/day15/helloServlet";
        var params = {
            "username":"jack",
            "password":"1234"
        };
        $.get(url,params,function(data){
            alert(data);
        });
    });

    //post 请求
    $("#postId").click(function(){
        var url = "/day15/helloServlet";
        var params = {
            "username":"杰克",
            "password":"1234"
        };
        $.post(url,params,function(data){
            alert(data);
        });
    });
});
</script>
```

1.3 案例分析



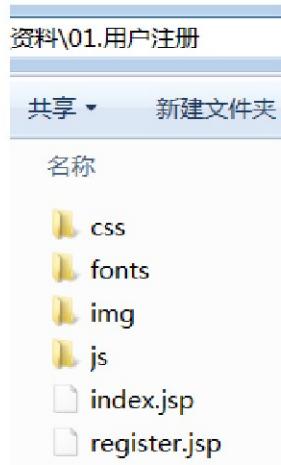
1. <input name="username">失去焦点时, 使用\$.post() 将用户名 username 以 ajax 方式发送给服务器
2. 服务器获得用户名, 并通过用户名查询用户
 - a) 如果用户名存在, 返回不可用提示

- b) 如果用户名可用，返回可用提示
3. 根据服务器响应的 json 数据，控制提示信息的显示和提交的按钮是否可用。

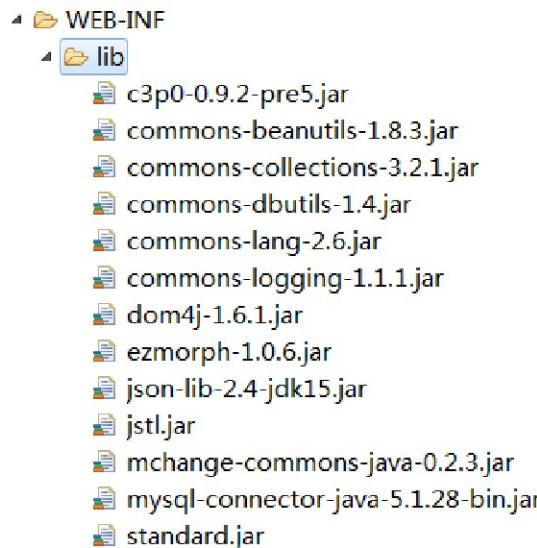
1.4 案例实现

1.4.1 搭建环境

- 步骤 1：创建项目，导入页面



- 步骤 2：导入 jar 包



- 步骤 3：复制工具类



```
<property name="driverClass">com.mysql.jdbc.Driver</property>
<property name="jdbcUrl">jdbc:mysql://127.0.0.1:3306/day15_db</property>
```



- 步骤 4：数据库及表

```
create database day22_db;
use day22_db;
create table user(
    uid varchar(32) primary key,
    username varchar(50),
    password varchar(32)
);
insert into user(uid,username,password) values('u001','jack','1234');
insert into user(uid,username,password) values('u002','rose','1234');
```

- 步骤 5：编写 JavaBean

- cn.itcast.domain

- User.java

```
public class User {

    private String uid;
    private String username;
    private String password;
```

1.4.2 服务器端程序

- 步骤 1：编写 servlet，获得用户名查询是否存在，并返回 json 数据。

```
public class UserCheckServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //0 编码
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        //1 获得用户名
        String username = request.getParameter("username");

        //2 查询是否存在
        UserService userService = new UserService();
        User user = userService.findByName(username);

        //3 处理
        String jsonData = "";
        if(user != null){
            jsonData = "{\"message\":\"用户名不可用\",\"flag\":false}";
        } else {
            jsonData = "{\"message\":\"用户名可用\",\"flag\":true}";
        }
    }
}
```



```
        response.getWriter().print(jsonData);

    }
```

● 步骤 2：编写 service

```
public class UserService {

    /**
     * 通过用户名查询
     * @param username
     * @return
     */
    public User findByName(String username) {
        UserDao userDao = new UserDao();
        return userDao.findByName(username);
    }

}
```

● 步骤 3：编写 dao，提供 findByName()方法

```
public class UserDao {

    /**
     * 通过用户名查询用户
     * @param username
     * @return
     */
    public User findByName(String username) {
        try {
            QueryRunner queryRunner = new QueryRunner(C3P0Utils.getDataSource());
            String sql = "select * from user where username = ?";
            Object[] params = {username};
            return queryRunner.query(sql, new BeanHandler<User>(User.class),
params);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

1.4.3 浏览器 JS

- 步骤 1：修改提示信息显示区域

```
register.jsp
140<div class="col-sm-6">
141    <input type="text" class="form-control" id="username" name="username" placeholder="请输入用户名">
142</div>
143<div class="col-sm-4">
144    <span id="showUsernameSpan"></span>
145<!--
146    <span id="showUsernameSpan" class="label label-success">用户名可用</span>
147    <span id="showUsernameSpan" class="label label-danger">用户名不可用</span>
148-->
149...</div>
```

- 步骤 2：给按钮添加 id 属性，并设置禁用，使用 bootstrap 的样式显示禁用效果。

```
register.jsp
200<div class="form-group">
201    <div class="col-sm-offset-2 col-sm-10">
202        <input type="submit" id="registButton" disabled="disabled" width="100" value="注册" border="0"
203            style="background: url('./img/register.gif') no-repeat scroll 0 0 rgba(0, 0, 0, 0);
204            height:35px;width:100px;color:white;" class="form-control">
205    </div>
206</div>
```

- 步骤 3：编写 js，当 input 失去焦点，发送 ajax，并控制提示信息显示和按钮是否可用。

```
<script type="text/javascript">
$(function() {
    $("input[name='username']").blur(function(){
        //1 确定请求路径
        var url = "${pageContext.request.contextPath}/userCheckServlet";
        //2 确定请求参数
        var params = {"username":$(this).val()};
        //3 发送 ajax 请求
        $.post(url,params,function(data){
            //4.1 给“提示显示”区域添加 label 样式
            $("#showUsernameSpan").addClass("label");
            if(data.flag){
                //4.2 可用，添加 success 样式（bootstrap 提供），并移除按钮禁用
                $("#showUsernameSpan").addClass("label-success");
                $("#showUsernameSpan").removeClass("label-danger");
                $("#registButton").removeProp("disabled");
            } else {
                //4.3 不可用，添加 danger 样式（bootstrap 提供），并禁用按钮
                $("#showUsernameSpan").addClass("label-danger");
                $("#showUsernameSpan").removeClass("label-success");
                $("#registButton").prop("disabled","disabled");
            }
            //4.4 设置提示信息
            $("#showUsernameSpan").text(data.message);
        },"json");
    });
});
```



```
});  
</script>
```

第2章 案例：异步自动填充

2.1 案例介绍

在开发中，通常情况下，搜索功能是非常常见的，类似百度，当我们输入搜索条件时，将自动填充我们需要的数据，并提供选择，我们将此类功能称为：自动填充（autocomplete）。如下图：



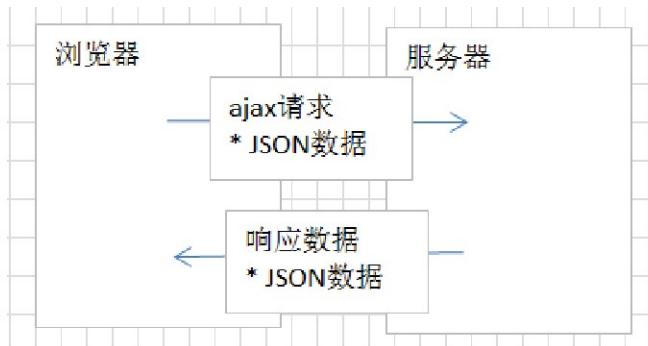
2.2 案例相关技术

2.2.1 JSON 数据

- 什么是 JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。JSON 采用完全独立于语言的文本格式，就是说不同的编程语言 JSON 数据是一致的。

易于人阅读和编写，同时也易于机器解析和生成(一般用于提升网络传输速率)。



- JSON 格式：

JSON 对象格式

```
{ "key": "value", "key": "value", ... }
```



键和值使用冒号分隔。

标准规范要求 key 必须使用双引号, value 如果没有使用双引号表示变量。

JSON 数组

[obj , obj , obj ,]

表示一组值, 多个值使用逗号分隔

● 例如:

```
<script type="text/javascript">

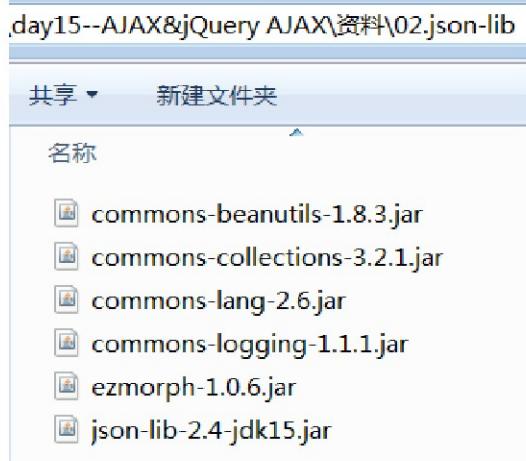
    //1 json 对象
    var user = {
        "username":"jack",
        "password":"1234"
    };
    alert(user.username); //通过 key 获得 json 数据

    //2 json 数组
    var arr = ['jack','rose','tom'];
    alert(arr[1]);

    //3 综合案例
    var data = [
        {"id":"b001","title":"javaweb","price":"998"},
        {"id":"b002","title":"java 基础","price":"123"},
        {"id":"b003","title":"ssh","price":"250"},
    ];
    alert(data[1].title);
</script>
```

2.2.2 JSON-LIB 工具

- json-lib 是将 java 对象与 json 数据相互转换的工具。
- 第三方工具, 使用时需要导入 jar 包





● 常用对象：

- JSONObject, java 对象 (JavaBean、Map) 与 JSON 数据 转换工具类
- JSONArray, java 集合 (List、Array) 与 JSON 数据 转换工具类

● 常用方法：

- static fromObject(...), 静态方法, 用于将 java 对象或集合转换成 jsonlib 对象。
- toString() 将 jsonlib 对象 转换成 json 字符串。

● 例如：

```
// map 或 javabean
Map<String, String> map = new HashMap<String, String>();
map.put("username", "jack");
map.put("password", "1234");

String str = JSONObject.fromObject(map).toString();
System.out.println(str);
/* 输出结果:
{
    "username":"jack",
    "password":"1234"
}

*/

// list 或 array
List<Map<String, String>> list = new ArrayList<>();
list.add(map);
list.add(map);

String str2 = JSONArray.fromObject(list).toString();
System.out.println(str2);
/* 输出结果:
[
    {"username":"jack", "password":"1234"},
    {"username":"jack", "password":"1234"}
]
*/
```

2.3 案例分析

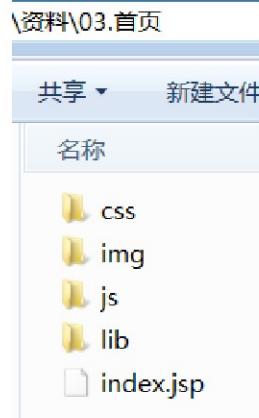


- 1. 用户输入搜索条件，键盘弹起时，发送 ajax 请求，将用户输入的内容发送给服务器
- 2.1 服务器获得用户输入的内容
- 2.2 根据要求拼凑查询条件，商品名称需要匹配，拼音也需要匹配，用户项可以不连续。
- 2.3 根据拼凑条件查询商品信息
- 3 将查询的商品信息使用 json-lib 转换成 json 数据。
- 4. 在 \$.post() 回调函数中处理查询结果，如果数据中有输入关键，需要“高亮”显示。

2.4 案例实现

2.4.1 搭建环境

- 步骤 1：创建项目，导入页面



- 步骤 2：导入 jar 包：



WEB-INF

lib

```
c3p0-0.9.2-pre5.jar  
commons-beanutils-1.8.3.jar  
commons-collections-3.2.1.jar  
commons-dbutils-1.4.jar  
commons-lang-2.6.jar  
commons-logging-1.1.1.jar  
dom4j-1.6.1.jar  
ezmorph-1.0.6.jar  
json-lib-2.4-jdk15.jar  
jstl.jar  
mchange-commons-java-0.2.3.jar  
mysql-connector-java-5.1.28-bin.jar  
standard.jar
```

资料\02.json-lib

共享 新建文件夹

名称

```
commons-beanutils-1.8.3.jar 已有  
commons-collections-3.2.1.jar  
commons-lang-2.6.jar  
commons-logging-1.1.1.jar 已有  
ezmorph-1.0.6.jar  
json-lib-2.4-jdk15.jar
```

步骤 3: 复制工具类和 c3p0 配置文件

src

```
cn.itcast.utils  
C3P0Utils.java  
c3p0-config.xml
```

步骤 4: 创建表

```
create table product(  
    pid varchar(32) primary key,  
    pname varchar(100),  
    pinyin varchar(200)  
);  
  
insert into product(pid,pname,pinyin) values('p001','服装','fuzhuang');  
insert into product(pid,pname,pinyin) values('p002','男装','nanzhuang');  
insert into product(pid,pname,pinyin) values('p003','女装','nvzhuang');  
insert into product(pid,pname,pinyin) values('p004','电脑','diannao');  
insert into product(pid,pname,pinyin) values('p005','奢侈品','shechipin');  
insert into product(pid,pname,pinyin) values('p006','图书','tushu');  
insert into product(pid,pname,pinyin) values('p007','食品','shipin');
```

步骤 5: 创建 Javabean

```
public class Product {  
    private String pid;  
    private String pname;  
    private String pinyin;
```

2.4.2 服务器程序

步骤 1: 编写 servlet, 获得关键字, 然后查询, 将查询结果转换成 json 数据。

```
public class ProductFindByWordServlet extends HttpServlet {
```



```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //0 编码
    request.setCharacterEncoding("UTF-8");
    response.setContentType("application/json; charset=UTF-8");
    //1 关键字
    String word = request.getParameter("word");
    //2 查询
    ProductService productService = new ProductService();
    List<Product> allProduct = productService.findAllByWord(word);
    //3 转换成 json
    String jsonData = JSONArray.fromObject(allProduct).toString();
    response.getWriter().println(jsonData);

}
```

- 步骤 2：编写 service，允许用户可以输入“汉字”、“拼音”、“若干字母”等进行查询。

```
public class ProductService {
    /**
     * 通过关键字查询
     * @param word
     * @return
     */
    public List<Product> findAllByWord(String word) {
        StringBuilder builder = new StringBuilder();
        List<Object> paramsList = new ArrayList<Object>();

        if(word != null){
            // 拼凑关键字的属性，hao 拼凑成 "%h%a%o%"
            StringBuilder wordBuilder = new StringBuilder();
            wordBuilder.append("%");
            for(int i = 0 ; i < word.length() ; i ++ ){
                wordBuilder.append(word.charAt(i)).append("%");
            }

            //1 汉字匹配
            builder.append(" and pname like ?");
            paramsList.add(wordBuilder.toString());

            //2 拼音匹配
            builder.append(" or pinyin like ?");
            paramsList.add(wordBuilder.toString());
        }
    }
}
```

```
        }

        //转换成需要的条件
        String condition = builder.toString();
        Object[] params = paramsList.toArray();

        ProductDao productDao = new ProductDao();
        return productDao.findAll(condition,params);
    }
}
```

- 步骤 3：编写 dao，查询带有条件商品信息

```
public class ProductDao {

    /**
     * 条件查询商品
     * @param condition
     * @param params
     * @return
     */
    public List<Product> findAll(String condition, Object[] params) {
        try {
            QueryRunner queryRunner = new QueryRunner(C3P0Utils.getDataSource());
            String sql = "select * from product where 1=1 " + condition;
            return queryRunner.query(sql,
                    new BeanListHandler<Product>(Product.class), params);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

2.4.3 浏览器 JS

- 步骤 1：添加 div 和 css，用于显示自动填充数据的 div



```
index.jsp
79      </li>
80  </ul>
81  <form class="navbar-form navbar-right" role="search">
82    <div class="form-group">
83      <input id="search" type="text" class="form-control" placeholder="Search">
84    </div>
85    <!--手机设备隐藏按钮，手机输入法“确定”按钮-->
86    <button type="submit" class="btn btn-default hidden-xs">Submit</button>
87  <div id="completeShow">
88    <ul id="itemul" class="list-group">
89      <%-- <li class="list-group-item">Cras justo odio</li>--%>
90    </ul>
91  </div>
92</form>

//HTML 页面已经提供
<div id="completeShow">
  <ul id="itemul" class="list-group">
    <%-- <li class="list-group-item">Cras justo odio</li>--%>
  </ul>
</div>

//CSS 样式，“./css/main.css” 文件中已提供
#completeShow{
  border: 1px solid #999;
  min-height: 200px;
  position: absolute;
  width: 196px;
  z-index: 1000;
  background-color: #fff;
  border-radius: 5px;
  display: none;
}
```

- 步骤 2：将查询结果显示到指定的区域，数据使用``包裹，使用 bootstrap 的`list-group-item` 涂染列表项。

```
//自动填充
$(function(){

  $("#search").keyup(function(){
    var url = "/day15_autocomplete/productFindByWordServlet";
    var word = $(this).val();
    if(word == "") {
      //如果没有输入关键字，隐藏提供区域
      $("#completeShow").slideUp(200);
      return false;
    }
    var params = {"word":word};
```

```
$.post(url,params,function(data){  
    $("#completeShow").html("<ul id='itemul' class='list-group'></ul>");  
    for(var i = 0 ; i < data.length ; i ++){  
        var product = data[i];  
        //处理关键字高亮  
        var str = ""+product.pname + "("+ product.pinyin +")";  
        str = highlight(word,str); //已提供算法  
  
        $("#itemul").append("<li class='list-group-item'><a href='#'>" + str + "</a></li>");  
        $("#completeShow").show();  
  
    }  
});  
}).focus(function(){  
    //获得焦点时，如果有搜索项显示  
    if($("#completeShow li").size() > 0){  
        $("#completeShow").show();  
    }  
}).click(function(){  
    //如果点击的是文本框，阻止点击事件，及不触发 document 的 click 事件  
    return false;  
});  
  
/*高亮，如果有输入的关键，使用红色标记  
 * 用户输入的关键字的个数，为替换内容的替换次数  
 * 例如：word = sp ; str = 食品(shipin)  
 * 第1次循环，使用 s 截取  
     食品(  
         <font>s</font>  
         hipin)  
     结果是：食品(<font>s</font>hipin)  
 * 第2次循环，使用 p 从后半段开始截取  
     hi  
     <font>p</font>  
     in)  
     结果是：食品(<font>s</font> + hi<font>p</font>in)  
 */  
  
function highlight(word,str){  
    var start = ""; //记录不需要或已经替换后的内容  
    var end = str; //记录需要被替换的内容，默认是整个内容。循环一次后，为后半段没有被替换的。  
  
    for(var i = 0 ; i < word.length ; i ++){  
        if(str.indexOf(word[i]) >= 0){  
            str = str.substring(0,i) + "<span style='color:red;font-weight:bold;">" + word[i] + "" + str.substring(i+1);  
        }  
    }  
    return str;  
}
```

```
// 1) 获得每一个关键字
var w = word.substring(i , i +1);

// 2) 从目标字符串中截取需要的内容，并进行替换，将字符串三分部：
var index = end.indexOf(w);

// * 前端：不需要或已经替换后的内容：shi
start += end.substring(0,index);

// * 需要使用<font>包裹内容：p --> <font ...>p</font>
start += "<font color='red'>" +w+ "</font>";

// * 后端：需要继续处理内容：in
end = end.substring(index + 1 ,end.length );

}

// 追加最后不用处理数据
start += end;

return start;
}

/**
 * 点击其他位置时，隐藏提示区域
 */
$(document).click(function(){
    $("#completeShow").slideUp(200);
});
});
```

第3章 总结