18-JSON & Ajax

今日任务

什么是 JSON
Json 在 JavaScript 中的使用。
    json 的定义
    json 的访问
    Json 的两个常用方法
        JSON.stringify()
        JSON.parse()
Json 在 java 中的使用
    javaBean 和 json 的互转
    List 和 json 的互转
    map 和 json 的互转
什么是 Ajax 请求？
原生 Ajax 请求的实现
JQuery 中的 Ajax 请求
    $.ajax 方法
        url
        type
        data
        success
        dataType
    $.get 方法和$.post 方法
        url
        data
        callback
        type
    $.getJSON 方法
        url
        data
        callback

    表单序列化 serialize()
项目第九阶段
    使用 Ajax 验证用户名是否可用
    使用 Ajax 请求修改购物车-----添加商品----修改数量

今日内容

先把笔记里，导入工程【day18】

1、JSON 学习

1.1、什么是 JSON

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于 JavaScript Programming Language， Standard ECMA-262 3rd Edition - December 1999 的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。 这些特性使 JSON 成为理想的数据交换语言。

1.2、JSON 对象定义和基本使用

在标准的 json 格式中，json 对象由在括号括起来，对象中的属性也就是 json 的 key 是一个字符串，所以一定要使用双引号引起来。每组 key 之间使用逗号进行分隔。

1.2.1、JSON 的定义

Json 定义格式：

```
var 变量名 = {
    “key” : value ,          // Number 类型
    “key2” : “value” ,       // 字符串类型
    “key3” : [] ,            // 数组类型
    “key4” : {},             // json 对象类型
    “key5” : [{},{}]         // json 数组
```

};

```
var jsons = {
        "key1":"abc", // 字符串类型
        "key2":1234,  // Number
        "key3":[1234,"21341","53"], // 数组
        "key4":{                    // json 类型
            "key4_1" : 12,
            "key4_2" : "kkk"
        },
        "key5":[{                   // json 数组
            "key5_1_1" : 12,
            "key5_1_2" : "abc"
        },{
            "key5_2_1" : 41,
            "key5_2_2" : "bbj"
        }]
};
```

1.2.2、JSON 对象的访问

json 对象，顾名思义，就知道它是一个对象。里面的 key 就是对象的属性。我们要访问一个对象的属性，只需要使用【对象名.属性名】的方式访问即可。

```
    <script type="text/javascript">
        // json 的定义
        var jsons = {
                "key1":"abc", // 字符串类型
                "key2":1234,  // Number
                "key3":[1234,"21341","53"], // 数组
                "key4":{                    // json 类型
                    "key4_1" : 12,
                    "key4_2" : "kkk"
                },
                "key5":[{                   // json 数组
                    "key5_1_1" : 12,
                    "key5_1_2" : "abc"
                },{
                    "key5_2_1" : 41,
                    "key5_2_2" : "bbj"
                }]
        };
        // 访问 json 的属性
```

```
        alert(jsons.key1); // "abc"
        // 访问 json 的数组属性
        alert(jsons.key3[1]); // "21341"
        // 访问 json 的 json 属性
        alert(jsons.key4.key4_1);//12
        // 访问 json 的 json 数组
        alert(jsons.key5[0].key5_1_2);//"abc"
    </script>
```

1.3、JSON 中两个常用的方法。

JSON 对象和字符串对象的互转

JSON.stringify(json); 此方法可以把一个 json 对象转换成为 json 字符串

JSON.parse(jsonString); 此方法可以把一个 json 字符串转换成为 json 对象

```
<script type="text/javascript">
    // 一个 json 对象
    var obj = {
        "a" : 12,
        "c" : "str"
    };
    // 把 json 对象转换成为字符串对象
    var objStr = JSON.stringify(obj);
    //
    alert(objStr);
    // 把 json 对象的字符串，转换成为 json 对象
    var jsonObj = JSON.parse(objStr);
    alert(jsonObj);
</script>
```

1.4、JSON 在 java 中的使用(****重点)

我们要使用 json 和 java 中使用，我们需要使用到一个第三方的包。它就是 gson.jar。

Gson 是 Google 提供的用来在 Java 对象和 JSON 数据之间进行映射的 Java 类库。可以将一个 JSON 字符串转成一个 Java 对象，或者反过来。

json 在 java 中的操作。常见的有三种情况。

1、java 对象和 json 的转换
2、java 对象 list 集合和 json 的转换
3、map 对象和 json 的转换

```
package com.atguigu.gson;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

public class GsonTest {

    static class Person {
        private int age;
        private String name;

        public Person() {
            // TODO Auto-generated constructor stub
        }

        public Person(int age, String name) {
            this.age = age;
            this.name = name;
        }

        public int getAge() {
            return age;
        }

        public void setAge(int age) {
            this.age = age;
        }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Person [age=" + age + ", name=" + name + "]";
    }

}

// 要把复杂的 json 字符串转换成为 java 对象。需要继承 TypeToken 类。
// 并把返回的类型当成 TypeToken 的泛型注入
static class PersonType extends TypeToken<List<Person>> {
}

public static void main(String[] args) {
    // json 操作，一定要先 new 一个 gson 对象。
    Gson gson = new Gson();
    // java 对象--json
    Person person = new Person(12, "wzg168");
    // 把对象转成为 json 字符串
    String personjson = gson.toJson(person);

    System.out.println(personjson);
    // 把 json 字符串转换成为 java 对象
    Person p = gson.fromJson(personjson, Person.class);
    System.out.println(p);
    System.out.println("----------------------------------------");
    // 2、java 对象 list 集合和 json 的转换
    List<Person> list = new ArrayList<Person>();
    for (int i = 0; i < 3; i++) {
        list.add(new Person(10 * i, "name-" + i));
    }
    String jsonListString = gson.toJson(list);
    System.out.println(jsonListString);

    // 把 json 数组转换成为 List 对象
    // List<Person> ps = gson.fromJson(jsonListString, new PersonType().getType());
    // 我们也可以使用匿名内部类
    List<Person> ps = gson.fromJson(jsonListString, new TypeToken<List<Person>>() {
    }.getType());
```

```
        System.out.println(ps);
        System.out.println("----------------------------------------");

        // 3、map 对象和 json 的转换
        Map<String, Person> mapPerson = new HashMap<String, GsonTest.Person>();
        // 添加 person 到 map 中
        mapPerson.put("p1", new Person(1, "person-1"));
        mapPerson.put("p2", new Person(2, "person-2"));
        // 把 map 转换成为 json 对象
        String jsonMapString = gson.toJson(mapPerson);
        System.out.println(jsonMapString);
        // 通过使用匿名内部类的方式
        Map<String, Person> map = gson.fromJson(jsonMapString,
                new TypeToken<HashMap<String, Person>>() {}.getType());
        System.out.println(map);
    }
}
```

2、Ajax 学习

2.1、什么是 Ajax？

AJAX 即“**A***synchronous* **J***avascript* ***And*** **X***ML*”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。

ajax 是一种浏览器异步发起请求。局部更新页面的技术。

2.2、javaScript 原生 Ajax 请求

原生的 Ajax 请求，

1、我们首先要创建 XMLHttpRequest 对象
2、调用 open 方法设置请求参数
3、调用 send 方法发送请求
4、在 send 方法前绑定 onreadystatechange 事件，处理请求完成后的操作。

1）创建一个 html 页面，发起请求。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="cache-control" content="no-cache" />
        <meta http-equiv="Expires" content="0" />
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
        <script type="text/javascript">
            function ajaxRequest() {
//              1、我们首先要创建 XMLHttpRequest
                var xhr = new XMLHttpRequest();
//              2、调用 open 方法设置请求参数
                xhr.open("GET","ajaxServlet?action=javaScriptAjax&a="+new Date(),true);
//              4、在 send 方法前绑定 onreadystatechange 事件，处理请求完成后的操作。
                xhr.onreadystatechange = function() {
                    // 判断请求完成，并且成功
                    if (xhr.readyState == 4 && xhr.status == 200) {
                        document.getElementById("div01").innerHTML = xhr.responseText;
                    }
                }
//              3、调用 send 方法发送请求
                xhr.send();
            }
        </script>
    </head>
    <body>
        <button onclick="ajaxRequest()">ajax request</button>
        <div id="div01">
        </div>
    </body>
</html>
```

2）创建一个 AjaxServlet 程序接收请求

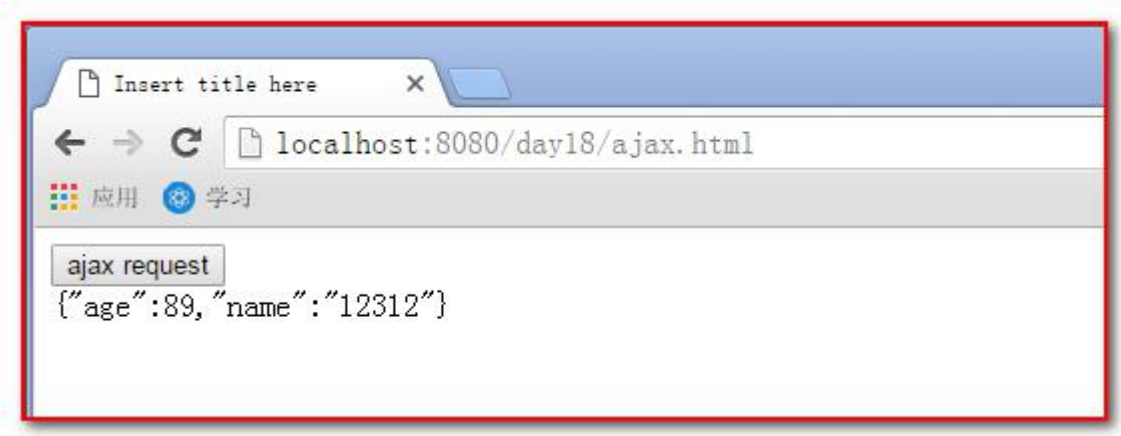
```
package com.atguigu.servlet;

import java.io.IOException;
import java.util.Random;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.atguigu.gson.GsonTest;
```

```
import com.google.gson.Gson;

public class AjaxServlet extends BaseServlet {
    private static final long serialVersionUID = 1L;

    protected void javaScriptAjax(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        System.out.println("ajax 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }

}
```

3）在 web.xml 文件中的配置：

```
<servlet>
  <servlet-name>AjaxServlet</servlet-name>
  <servlet-class>com.atguigu.servlet.AjaxServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>AjaxServlet</servlet-name>
  <url-pattern>/ajaxServlet</url-pattern>
</servlet-mapping>
```

3）测试效果



通过上面的代码我们发现。编写原生的 JavaScript 我们自己要写很多的代码。而且还要考虑浏览器兼容问题。所以

使用起来非常的不方便。那我们工作之后。怎么处理 Ajax 请求呢。我们一般会使用 JavaScript 的框架来解决这个问题，比如说我们前面学到的 Jquery 框架。它就有很好的 Ajax 解决方案。

2.3、JQuery 的 Ajax 请求(重点****)

四个 Ajax 请求方法

$.ajax 方法
$.get 方法
$.post 方法
$.getJSON 方法

一个表单序列化方法：serialize()表单序列化方法

如何使用上面的五个方法：

在 JQuery 中和 Ajax 请求有关的方法有四个

$.ajax 请求参数

url:	请求的地址	
type :	请求的方式	get 或 post
data :	请求的参数	string 或 json
success:	成功的回调函数	
dataType:	返回的数据类型	常用 json 或 text

下面的方法必须遵守参数的顺序

$.get 请求和$.post 请求

url:请求的 URL 地址

data:待发送 Key/value 参数。

callback:载入成功时回调函数。

type:返回内容格式，xml, html, script, json, text。

Jquery 的$.getJSON

url:待载入页面的 URL 地址

data:待发送 Key/value 参数。

callback:载入成功时回调函数。

表单的序列化

serialize() 方法可以把一个 form 表单中所有的表单项。都以字符串 name=value&name=value 的形式进行拼接，省去我们很多不必要的工作。

由于$.get、$.post 和 getJSON 这三个方法的底层都是直接或者间接地使用$.ajax()方法来实现的异步请求的调用。所以我们以$.ajax()方法的使用为示例进行展示：

1）Jquery_Ajax_request.html 的代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="cache-control" content="no-cache" />
        <meta http-equiv="Expires" content="0" />
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
        <script type="text/javascript" src="script/jquery-1.7.2.js"></script>
        <script type="text/javascript">
            $(function(){
                // ajax 请求
                $("#ajaxBtn").click(function(){
                    $.ajax({
                        url : "ajaxServlet", // 请求地址
                        error:function(){   // 请求失败回调
                            alert("请求失败");
                        },
                        success:function(data){ // 请求成功回调
                            alert( data );
                        },
                        type:"POST",              // 请求的方式
                        dataType:"json",          // 返回的数据类型为 json 对象
                        data:{                    // 请求的参数
                            action:"jqueryAjax",
                            a:12,
                            date: new Date()
                        }
                    });
                });
```

```
// ajax--get 请求
$("#getBtn").click(function(){
    $.get(
        "ajaxServlet",{
            action:"jqueryGet",
            a:12,
            date:new Date()
        },function(data){alert(data);},"json"
    );
});

// ajax--post 请求
$("#postBtn").click(function(){
    // post 请求
    $.post(
        "ajaxServlet", // 请求路径
        {              // 请求参数
            action:"jqueryPost",
            a:12,
            date:new Date()
        },
        function(data){ alert( data ) },  // 成功的回调函数
        "text"                        // 返回的数据类型
    );
});

// ajax--getJson 请求
$("#getJsonBtn").click(function(){
    // 调用
    $.getJSON(
        "ajaxServlet",        // 请求路径
        {                  // 请求参数
            action:"jqueryGetJSON",
            a:12,
            date:new Date()
        },
        function(data){ alert( data ) }  // 成功的回调函数
    );
});

// ajax 请求
$("#submit").click(function(){
    // 把参数序列化
    var data = $("#form01").serialize();
    alert(data);
```

```
            });

        });
    </script>
</head>
<body>
    <div>
        <button id="ajaxBtn">$.ajax 请求</button>
        <button id="getBtn">$.get 请求</button>
        <button id="postBtn">$.post 请求</button>
        <button id="getJsonBtn">$.getJSON 请求</button>
    </div>
    <br/><br/>
    <form id="form01" >
        用户名：<input name="username" type="text" /><br/>
        密码：<input name="password" type="password" /><br/>
        下拉单选：<select name="single">
            <option value="Single">Single</option>
            <option value="Single2">Single2</option>
        </select><br/>
        下拉多选：
        <select name="multiple" multiple="multiple">
            <option selected="selected" value="Multiple">Multiple</option>
            <option value="Multiple2">Multiple2</option>
            <option selected="selected" value="Multiple3">Multiple3</option>
        </select><br/>
        复选：
        <input type="checkbox" name="check" value="check1"/> check1
        <input type="checkbox" name="check" value="check2" checked="checked"/>
check2<br/>
        单选：
        <input type="radio" name="radio" value="radio1" checked="checked"/> radio1
        <input type="radio" name="radio" value="radio2"/> radio2<br/>
        <input id="submit" type="submit" />
    </form>
</body>
</html>
```

2)AjaxServlet 的代码如下：

```
package com.atguigu.servlet;

import java.io.IOException;
import java.util.Random;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.atguigu.gson.GsonTest;
import com.google.gson.Gson;

public class AjaxServlet extends BaseServlet {
    private static final long serialVersionUID = 1L;

    protected void javaScriptAjax(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        System.out.println("ajax 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }

    protected void jqueryAjax(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("jqueryAjax 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }

    protected void jqueryGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("jqueryGet 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }

    protected void jqueryPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("jqueryPost 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }
```

```
    protected void jqueryGetJSON(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        System.out.println("jqueryGetJSON 请求过来了 a--" + request.getParameter("a"));
        Random random = new Random(System.currentTimeMillis());
        // 使用随机数，可以让客户端看到变化
        response.getWriter().write(
                new Gson().toJson(new GsonTest.Person(random.nextInt(100), "12312")));
    }

}
```

3、第九阶段：

1、Ajax 验证用户名是否可用。
2、Ajax 修改购物车模块---添加商品---修改数量

3.1、Ajax 验证用户名是否可用。

使用 Ajax 验证用户名是否可用。我们需要在页面端，给用户名输入框添加一个失去焦点事件。当用户名输入框失去焦点的时候，触发事件。获取输入的用户名。然后发送 Ajax 请求到服务器端去验证。
然后服务器通过 json 数据，返回是否存在，result 为 0 表示 不存在，result 为 1 表示存在。当然我们还要做一个用户名不为空的简单验证。才能让请求发送到服务器端。

1）修改 pages/user/regist.jsp 页面。给用户名输入框添加失去焦点事件。

```
        //用户名是否存在验证
        $("#username").blur(function(){
            // 获取用户名
            var usernameValue = this.value;
            //判断用户名不能为空
            if (usernameValue == "") {
                $("#errorSpan").html("用户名不能为空！ ");
```

```
            return;
        }

        // 发送 ajax 请求验证

    $.getJSON("userServlet?action=existsUsername&username="+usernameValue,function(data)
{
            // result 等于 0，说明用户名不存在
            if (data.result == 0) {
                $("#errorSpan").html("");
            }
            // result 等于 1，说明用户名存在
            else if (data.result == 1) {
                $("#errorSpan").html("用户名已存在");
            }
        });

    });
```

2）修改 UserServlet 类，添加检查用户名是否存在的方法：

```
    public void existsUsername(HttpServletRequest request, HttpServletResponse response)
            throws Exception, IOException {
        // 获取用户名
        String username = request.getParameter("username");
        // 判断用户名是否存在
        boolean existsUsername = userService.existsUsername(username);
        // 返回用户是否存在
        Map<String, Integer> result = new HashMap<String, Integer>();
        // 如果用户存在，返回 result 为 1,如果用户不存在。result 返回 0
        if (existsUsername) {
            result.put("result", 1);
        } else {
            result.put("result", 0);
        }
        // 生成 Gson 对象，用于把 map 转换成为 json 字符串返回
        Gson gson = new Gson();
        String responseStr = gson.toJson(result);
        response.getWriter().write(responseStr);
    }
```

3.2、Ajax 修改购物车模块---添加商品---修改数量

以 Ajax 请求的方式修改购物车的模块。我们修改的功能有，添加到购物车，修改数量，以及删除商品，和清空购物车。我们以添加购物车和修改商品数量为例给大家演示。

3.2.1、添加商品

添加商品到购物车。首先我们要把商品的编号，以 Ajax 的方式传到服务器。然后器添加成功后把购物车的数量，最后一本书的名字返回，给用户显示。

1）修改 pages/client/index.jsp 页面，添加购物车的 a 标签代码：

```
        <div class="book_add">
            <a idv="${ item.id }" class="addCart"
href="cartServlet?action=addItem&id=${ item.id }">加入购物车</a>
        </div>
```

2）添加 Ajax 请求的 js 代码：

```
        // 添加购物车 Ajax 请求
        $(".addCart").click(function(){
            // 通过添加 idv 属性保存商品 id 信息
            var idv = $(this).attr("idv");
            $.getJSON("cartServlet?action=ajaxAddItem&id=" + idv,function(data) {
                if (data.result == 0) {
                    $("#cart_totalCount").html("您的购物车中有" + data.totalCount + "件商品
");
                    $("#last_product").html("您刚刚将<span style='color: red'>" +
data.last_product + "</span>加入到了购物车中");
                }
            });

            return false;
        });
```

3）修改添加购物车后。搜索下方的购物车显示：

```
            <div style="text-align: center">
                <c:choose>
                    <%-- 购物车为空 --%>
                    <c:when test="${ empty cart.items }">
                        <span id="cart_totalCount">您的购物车为空！</span>
                        <div id="last_product">   </div>
```

```
                </c:when>
                <c:otherwise>
                    <span id="cart_totalCount">您的购物车中有${ cart.totalCount }件商品
</span>
                    <div id="last_product">
                        您刚刚将<span style="color:
red">${ sessionScope.last_product }</span>加入到了购物车中
                    </div>
                </c:otherwise>
            </c:choose>
        </div>
```

4）CartServlet 中添加 Ajax 版的添加购物车代码：

```
    /**
     * Ajax 版--添加到购物车
     */
    protected void ajaxAddItem(HttpServletRequest request, HttpServletResponse response)
            throws Exception, IOException {
        response.setContentType("text/html;charset=UTF-8");
        // 获取购物车
        Cart cart = (Cart) request.getSession().getAttribute("cart");
        if (cart == null) {
            // 生成一个新的购物车，放到 Session 对象中
            cart = new Cart();
            request.getSession().setAttribute("cart", cart);
        }
        // 获取图书 的 id
        int id = Utils.parseInt(request.getParameter("id"), 0);
        // 查找图书
        Book book = bookService.findBookById(id);
        // 添加到购物车中
        cart.addItem(book);
        // 添加商品名到 Session 对象中。
        request.getSession().setAttribute("last_product", book.getName());
        // 打印测试
        System.out.println(cart);
        // 创建一个 map 用于返回结果
        Map<String, Object>result = new HashMap<String, Object>();
        result.put("result", 0);
        result.put("totalCount", cart.getTotalCount());
        result.put("last_product", book.getName());

        Gson gson = new Gson();
```

```
        response.getWriter().write(gson.toJson(result));
    }
```

3.2.2、修改数量

修改购物车数量，我们要把修改的商品编号和数量发送到服务器。然后服务器把修改后商品的总价 item_totalMoney，购物车的总数量 cart_totalCount，以及购物车的总金额 cart_totalMoney 返回用于前端的修改。

1）修改原来购物车更新数量的方法，返回修改后商品的总金额

```
    public void updateItem(int id, int count) {
        // 先从购物车中取出商品
        CartItem item = items.get(id);
        // 如果为 null，说明之前没有此商品
        if (item != null) {
            // 修改商品数量和总金额
            item.setCount(count);
            item.setTotalMoney(item.getPrice() * item.getCount());
        }
    }
```

修改为：

```
    public double updateItem(int id, int count) {
        // 先从购物车中取出商品
        CartItem item = items.get(id);
        // 如果为 null，说明之前没有此商品
        if (item != null) {
            // 修改商品数量和总金额
            item.setCount(count);
            item.setTotalMoney(item.getPrice() * item.getCount());
            return item.getTotalMoney();
        }
        return 0;
    }
```

2）修改 pages/cart/cart.jsp 页面中的内容：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>购物车</title>

<!-- 头部共享信息的引入。包含 jquery，base 标签，以及 css 样式 -->
<%@ include file="/pages/common/header.jsp" %>
<script type="text/javascript">
    $(function(){

        // 修改商品数量的事件。
        $("a.deleteItem").click(function(){
            // 提示用户是否修改
            var name = $(this).parent().parent().children("td:first").html();
            // 询问用户是否要删除
            return confirm("你确定要删除【" + name + "】吗?");
        });


        // 修改商品数量的事件。
        $(".updateItem").change(function(){
            // 提示用户是否修改
            var name = $(this).parent().parent().children("td:first").html();
            if ( confirm("你确定修改【" + name + "】个数为：" + this.value) ){
//              // 发起请求
//              location.href="cartServlet?action=updateItem&id=" + $(this).attr("data") +
"&count="+this.value;

                var id = $(this).attr("data");
                // 修改请求为 Ajax
                $.getJSON("cartServlet?action=ajaxUpdateItem&id=" + id +
"&count="+this.value,
                        function(data){
//                  alert( JSON.stringify(data) );
                    $("#item_totalMoney_" + id).html(data.item_totalMoney);
                    $("#cart_totalCount").html(data.cart_totalCount);
                    $("#cart_totalMoney").html(data.cart_totalMoney);
                });

            } else {
                // 还原商品数量
                this.value = $(this).attr("ov");
            }
        });
```

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

    <div id="header">
            <img class="logo_img" alt="" src="static/img/logo.gif" >
            <span class="wel_word">购物车</span>

        <!-- 登录成功之后所有相同的菜单  -->
        <%@ include file="/pages/common/login_success_menu.jsp" %>

    </div>

    <div id="main">
        <table>
            <tr>
                <td>商品名称</td>
                <td>数量</td>
                <td>单价</td>
                <td>金额</td>
                <td>操作</td>
            </tr>
            <c:choose>
                <%-- 先判断，如果购物车有商品，则显示，没有则提示用户，购物车是空 --%>
                <c:when test="${  not empty sessionScope.cart.items }">
                    <!-- 遍历购物车中的内容 -->
                    <c:forEach items="${ sessionScope.cart.items }" var="item">
                        <%--
                            把购物车中的 CartItem 取到,存到 pageScope 域中
                         --%>
                        <c:set value="${ item.value }" var="cartItem" />
                        <%-- 输出 --%>
                        <tr>
                            <td>${ cartItem.name }</td>
                            <!-- ov 为原来的数量备份，data 属性保存商品编号，class 属性，方便我们
通过 class 选择器查找到输入框 -->
                            <td><input ov="${ cartItem.count }" style="width: 35px;"
class="updateItem" data="${ cartItem.id }" value="${ cartItem.count }" type="text" /> </td>
                            <td>${ cartItem.price }</td>
                            <td
id="item_totalMoney_${ cartItem.id }">${ cartItem.totalMoney }</td>
                            <!-- class 属性方便 jquery 查找所有删除的 a 标签 -->
                            <td><a class="deleteItem"
href="cartServlet?action=deleteItem&id=${ cartItem.id }">删除</a></td>
                        </tr>
```

```
                </c:forEach>
            </c:when>
            <%-- 没有商品，提示用户 --%>
            <c:otherwise>
                <tr>
                    <td colspan="5"><a href="${ pageContext.request.contextPath }">亲，
购物车是空的。快去买，买，买！！！</a></td>
                </tr>
            </c:otherwise>
        </c:choose>

    </table>
    <div class="cart_info">
        <%-- 先判断，如果购物车有商品，则显示 --%>
        <c:if test="${  not empty sessionScope.cart.items }">
            <span class="cart_span">购物车中共有<span id="cart_totalCount"
class="b_count">${ cart.totalCount }</span>件商品</span>
            <span class="cart_span">总金额<span id="cart_totalMoney" class="b_price">
${ cart.totalMoney } </span>元</span>
            <span class="cart_span"><a href="cartServlet?action=clear">清空购物车
</a></span>
            <span class="cart_span"><a href="client/orderServlet?action=createOrder">
去结账</a></span>
        </c:if>
    </div>
</div>
<!-- 这是页脚的引入 -->
<%@ include file="/pages/common/footer.jsp" %>
</body>
</html>
```

3）添加 CartServlet 中 ajaxUpdateItem 方法实现 Ajax 请求的修改购物车商品数量

```
    /**
     * ajax版--更新购物车商品数量
     */
    protected void ajaxUpdateItem(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // 获取删除的商品编号
        int id = Utils.parseInt(request.getParameter("id"), 0);
        // 获取商品数量
        int count = Utils.parseInt(request.getParameter("count"), 1);
        // 获取购物车
```

```
        Cart cart = (Cart) request.getSession().getAttribute("cart");
        if (cart == null) {
            // 生成一个新的购物车，放到 Session 对象中
            cart = new Cart();
            request.getSession().setAttribute("cart", cart);
        }
        // 删除商品
        double item_totalMoney = cart.updateItem(id, count);
        // 创建一个 Map 返回要显示的内容
        Map<String, Object> result = new HashMap<String, Object>();
        result.put("item_totalMoney", ""+item_totalMoney);
        result.put("cart_totalMoney", cart.getTotalMoney());
        result.put("cart_totalCount", cart.getTotalCount());
        Gson gson = new Gson();
        response.getWriter().write(gson.toJson(result));
    }
```