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。JSON

采用完全独立于语言的文本格式，而且很多语言都提供了对 json 的支持（包括 C, C++, C#, Java, JavaScript, Perl, Python

等）。 这样就使得 JSON 成为理想的数据交换格式。

json 是一种轻量级的数据交换格式。

轻量级指的是跟 xml 做比较。

数据交换指的是客户端和服务器之间业务数据的传递格式。

1.1、JSON 在 JavaScript 中的使用

1.1.1、json 的定义

json 是由键值对组成，并且由花括号（大括号）包围。每个键由引号引起来，键和值之间使用冒号进行分隔，

多组键值对之间使用逗号进行分隔。

json 定义示例

```
<script type="text/javascript">
  // json的定义
  var jsonObj = {
    "key1":11,
    "key2":"value2",
    "key3":false,
    "key4":[11,"22",true],
    "key5":{
      "key5_1":511,
      "key5_2":"key_5_2_value"
    },
    "key6":[{
      "key6_1_1":6111,
      "key6_1_2":"key6_1_2_value"
```

```
    },{
      "key6_2_1":6211,
      "key6_2_2":"key6_2_2_value"
    }]
  };
</script>
```

1.1.2、json 的访问

json 本身是一个对象。

json 中的 key 我们可以理解为是对象中的一个属性。

json 中的 key 访问就跟访问对象的属性一样： json 对象.key

json 访问示例：

```
<script type="text/javascript">
  // json的定义
  var jsonObj = {
    "key1":11,
    "key2":"value2",
    "key3":false,
    "key4":[11,"22",true],
    "key5":{
      "key5_1":511,
      "key5_2":"key_5_2_value"
    },
    "key6":[{
      "key6_1_1":6111,
      "key6_1_2":"key6_1_2_value"
    },{
      "key6_2_1":6211,
      "key6_2_2":"key6_2_2_value"
    }]
  };
  // json的访问
  alert(jsonObj.key1);
  alert(jsonObj.key2);
  alert(jsonObj.key3);


  // alert(jsonObj.key4);
  for (var i=0; i < jsonObj.key4.length; i++){
    alert(jsonObj.key4[i]);
  }

  alert(jsonObj.key5.key5_1);
```

```
    alert(jsonObj.key5.key5_2);

    var jsonItem = jsonObj.key6[0];
    alert(jsonItem.key6_1_1);
    alert(jsonItem.key6_1_2);

    // json对象转字符串
    // json字符串转json对象
</script>
```

1.1.3、json 的两个常用方法

json 的存在有两种形式:

- 一种是：对象的形式存在，我们叫它 json 对象。
- 一种是：字符串的形式存在，我们叫它 json 字符串。

一般我们要操作 json 中的数据的时候，需要 json 对象的格式。

一般我们要在客户端和服务器之间进行数据交换的时候，使用 json 字符串。

- JSON.stringify()　把 json 对象转换成为 json 字符串
- JSON.parse()　　把 json 字符串转换成为 json 对象

```
<script type="text/javascript">
    // json的定义
    var jsonObj = {
        "key1":11,
        "key2":"value2",
        "key3":false,
        "key4":[11,"22",true],
        "key5":{
            "key5_1":511,
            "key5_2":"key_5_2_value"
        },
        "key6":[{
            "key6_1_1":6111,
            "key6_1_2":"key6_1_2_value"
        },{
            "key6_2_1":6211,
            "key6_2_2":"key6_2_2_value"
        }]
    };
```

```
        // json对象转字符串
        var jsonObjString = JSON.stringify(jsonObj);
        alert(jsonObjString)
        // json字符串转json对象
        var jsonObj2 = JSON.parse(jsonObjString);
        alert(jsonObj2)
    </script>
```

1.2、JSON 在 java 中的使用

1.2.1、javaBean 和 json 的互转

1.导入谷歌的 gson 包:gson-2.2.4.jar

2.新建 JavaBean

```
package com.admiral.pojo;

/**
 * @author 白世鑫
 * @title: Person
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/15  1:55 上午
 */
public class Person {

    private Integer id;
    private String name;
    private String address;

    public Person() {
    }

    public Person(Integer id, String name, String address) {
        this.id = id;
        this.name = name;
        this.address = address;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
```

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

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

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return "Person{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

3.测试类:

```
package com.admiral.json;

import com.admiral.pojo.Person;
import com.google.gson.Gson;
import org.junit.Test;

/**
 * @author 白世鑫
 * @title: JsonTest
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/15  1:57 上午
 */
public class JsonTest {

    @Test
    public void testJavaBeanToJson(){
        Person p = new Person(1,"小花花","beijing");
        //将 JavaBean 转换为 JSON 字符串
```

```
        Gson gson = new Gson();
        String personObjString = gson.toJson(p);
        System.out.println(personObjString);

        //将 JSON 字符串转换为 JavaBean
        Person person = gson.fromJson(personObjString, Person.class);
        System.out.println(person);
    }
}
```

1.2.2、List 和 json 的互转

```
package com.admiral.pojo;

import com.google.gson.reflect.TypeToken;

import java.util.ArrayList;

/**
 * @author 白世鑫
 * @title: PersonListType
 * @projectName JavaWeb
 * @description:
 * @date 2020/9/15  2:08 上午
 */
public class PersonListType extends TypeToken<ArrayList<Person>> {
}
```

```
    @Test
    public void test2(){
        List<Person> personList = new ArrayList<>();
        Person p1 = new Person(1,"小花花","beijing");
        Person p2 = new Person(2,"小红红","shanghai");

        personList.add(p1);
        personList.add(p2);
```

```
        Gson gson = new Gson();
        String personListString = gson.toJson(personList);
        System.out.println(personListString);

        List<Person> persons = gson.fromJson(personListString, new
PersonListType().getType());
        System.out.println(persons);
    }
```

1.2.3、map 和 json 的互转

```
    @Test
    public void test3(){
        Map<Integer,Person> personMap = new HashMap<>();
        Person p1 = new Person(1,"小花花","beijing");
        Person p2 = new Person(2,"小红红","shanghai");

        personMap.put(1,p1);
        personMap.put(2,p2);

        Gson gson = new Gson();
        String personMapString = gson.toJson(personMap);
        System.out.println(personMapString);

        Map<Integer,Person> personMap1 = gson.fromJson(personMapString,new
TypeToken<HashMap<Integer,Person>>(){}.getType());
        System.out.println(personMap1);
    }
```

2、AJAX 请求

2.1、什么是 AJAX 请求

AJAX 即“Asynchronous Javascript And XML”（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 是一种浏览器通过 js 异步发起请求，局部更新页面的技术。

Ajax 请求的局部更新，浏览器地址栏不会发生变化

局部更新不会舍弃原来页面的内容

2.2、原生 AJAX 请求的示例

```
<!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 xmlHttpRequest = new XMLHttpRequest();
//        2、调用open方法设置请求参数
        xmlHttpRequest.open("GET","http://localhost:8080/10_JSON/ajaxServlet?
action=jsAjaxServlet",true);
//        3、调用send方法发送请求
        xmlHttpRequest.send();
//        4、在send方法前绑定onreadystatechange事件，处理请求完成后的操作。
        xmlHttpRequest.onreadystatechange = function(){
          if (xmlHttpRequest.readyState == 4 && xmlHttpRequest.status == 200){
            document.getElementById("div01").innerText =
xmlHttpRequest.responseText;
          }
        }
      }
    </script>
  </head>
  <body>
    <button onclick="ajaxRequest()">ajax request</button>
    <div id="div01">
    </div>
  </body>
</html>
```

2.3、jQuery 中的 AJAX 请求

$.ajax 方法

url　　表示请求的地址

type　　表示请求的类型 GET 或 POST 请求

data　　表示发送给服务器的数据

格式有两种：

一：name=value&name=value

二：{key:value}

success	请求成功，响应的回调函数
dataType	响应的数据类型 常用的数据类型有： text 表示纯文本 xml 表示 xml 数据 json 表示 json 对象

```
        // ajax请求
        $("#ajaxBtn").click(function(){
          $.ajax({
            url:"http://localhost:8080/10_JSON/ajaxServlet?",
            data:"action=jQueryServlet",
            type:"GET",
            success:function (data) {
              // var jsonObj = JSON.parse(data);
              // $("#msg").html("编号:"+jsonObj.id + ",姓名:"+jsonObj.name+",地
址:" + jsonObj.address);
              // alert(data)
              $("#msg").html("编号:"+data.id + ",姓名:"+data.name+",地址:" +
data.address);
            },
            dataType:"json"
          })
          // alert("ajax btn");
        });
```

$.get 方法和$.post 方法

url	请求的 url 地址
data	发送的数据
callback	成功的回调函数
type	返回的数据类型

```
        // ajax--get请求
        $("#getBtn").click(function(){


$.get("http://localhost:8080/10_JSON/ajaxServlet?","action=jQueryGet",function
(data) {
```

```
            $("#msg").html("Get 编号:"+data.id + ",姓名:"+data.name+",地址:" +
data.address);
          },"json")

        });

        // ajax--post请求
        $("#postBtn").click(function(){
          // post请求

$.get("http://localhost:8080/10_JSON/ajaxServlet?","action=jQueryGet",function
(data) {
            $("#msg").html("Post 编号:"+data.id + ",姓名:"+data.name+",地址:" +
data.address);
          },"json")

        });
```

$.getJSON 方法

url	请求的 url 地址
data	发送给服务器的数据
callback	成功的回调函数

```
        // ajax--getJson请求
        $("#getJSONBtn").click(function(){
          // 调用
          // post请求

$.get("http://localhost:8080/10_JSON/ajaxServlet?","action=jQueryJson",functio
n (data) {
            $("#msg").html("JSON 编号:"+data.id + ",姓名:"+data.name+",地址:" +
data.address);
          },"json")

        });
```

表单序列化 serialize()

serialize()可以把表单中所有表单项的内容都获取到，并以 name=value&name=value 的形式进行拼接。

```
        // ajax请求
        $("#submit").click(function(){

          // alert($("#form01").serialize())

          // 把参数序列化

$.getJSON("http://localhost:8080/10_JSON/ajaxServlet?","action=jQuerySerialize
&"+$("#form01").serialize(),function (data) {
            $("#msg").html("jQuerySerialize 编号:"+data.id + ",姓
名:"+data.name+",地址:" + data.address);
          })
        });
```

3、书城项目第九阶段

3.1、使用 AJAX 验证用户名是否可用

UserServlet 程序中 ajaxExistsUsername 方法

```
    /**
     * 使用 Ajax 验证用户名是否可用
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    protected void ajaxExistsUsername(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        //先获取请求的参数
        String username = request.getParameter("username");
        boolean existsUsername = userService.existsUsername(username);
        //把返回的结果封装成 map 对象
        Map<String, Object> map = new HashMap<>();
        map.put("existsUsername", existsUsername);
        Gson gson = new Gson();
        String json = gson.toJson(map);
        response.getWriter().write(json);
    }
```

regist.jsp 页面中的代码：

```
            $("#username").blur(function () {
                //1.获取用户名
                var username = this.value;

 $.getJSON("http://localhost:8080/book/userServlet","action=ajaxExistsUsername
&username="+username,function (data) {
                    if(data.existsUsername){
                        $("span.errorMsg").text("用户名已存在!");
                    }else {
                        $("span.errorMsg").text("用户名可用!");
                    }
                    console.log(data);
                })
            });
```

3.2、使用 AJAX 修改把商品添加到购物车

CartServlet 程序

```
    /**
     * 使用 Ajax 添加购物车
     *
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    protected void ajaxAddItem(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        //获取请求的参数,图书编号
        Integer id = WebUtils.parseInt(req.getParameter("id"), 0);
        //根据编号调动 service 查询图书信息
        Book book = bookService.queryBookById(id);
        //根据图书信息封装cartItem
        CartItem cartItem = new CartItem(book.getId(), book.getName(), 1,
book.getPrice(), book.getPrice());
        //调用 cart.additem 添加到购物车
        Cart cart = (Cart) req.getSession().getAttribute("cart");
        if (cart == null) {
            cart = new Cart();
            req.getSession().setAttribute("cart", cart);
        }
        cart.addItems(cartItem);
        System.out.println(cartItem);

        //在添加商品的时候,保存最后一个添加商品的名称
```

```
        req.getSession().setAttribute("lastName", cartItem.getName());

        //返回购物车商品数量和最后一个添加的商品名称
        Map<String,Object> result = new HashMap<>();
        result.put("totalCount",cart.getTotalCount());
        result.put("lastName",cartItem.getName());

        Gson gson = new Gson();
        String resultJsonString = gson.toJson(result);
        resp.getWriter().write(resultJsonString);

        //重定向到原来的页面
//        resp.sendRedirect(req.getHeader("Referer"));

    }
```

pages/client/index.jsp 页面

```
    <div style="text-align: center">

        <c:if test="${not empty sessionScope.cart.items}">
            <span id="cartTotalCount">您的购物车中有
${sessionScope.cart.totalCount}件商品</span>
            <div>
                您刚刚将<span style="color: red"
id="cartLastName">${sessionScope.lastName}</span>加入到了购物车中
            </div>
        </c:if>
        <c:if test="${empty sessionScope.cart.items}">
            <span id="cartTotalCount"></span>
            <div>
                <span style="color: red" id="cartLastName">购物车为空</span>
            </div>
        </c:if>


    </div>
```

javaScript 代码

```
    <script type="text/javascript">
        $(function (){
            $("button.addToCart").click(function () {

                var bookId = $(this).attr("bookId");
```

```
                // location.href="http://localhost:8080/book/cartServlet?
action=addItem&id="+bookId;

 $.getJSON("http://localhost:8080/book/cartServlet","action=ajaxAddItem&id="+b
ookId,function (data) {
                    $("#cartTotalCount").text("您的购物车中有" + data.totalCount
+ "件商品")
                    $("#cartLastName").text(data.lastName)
                })
            });
        });
    </script>
```