今日内容

1. AJAX：
2. JSON

AJAX：

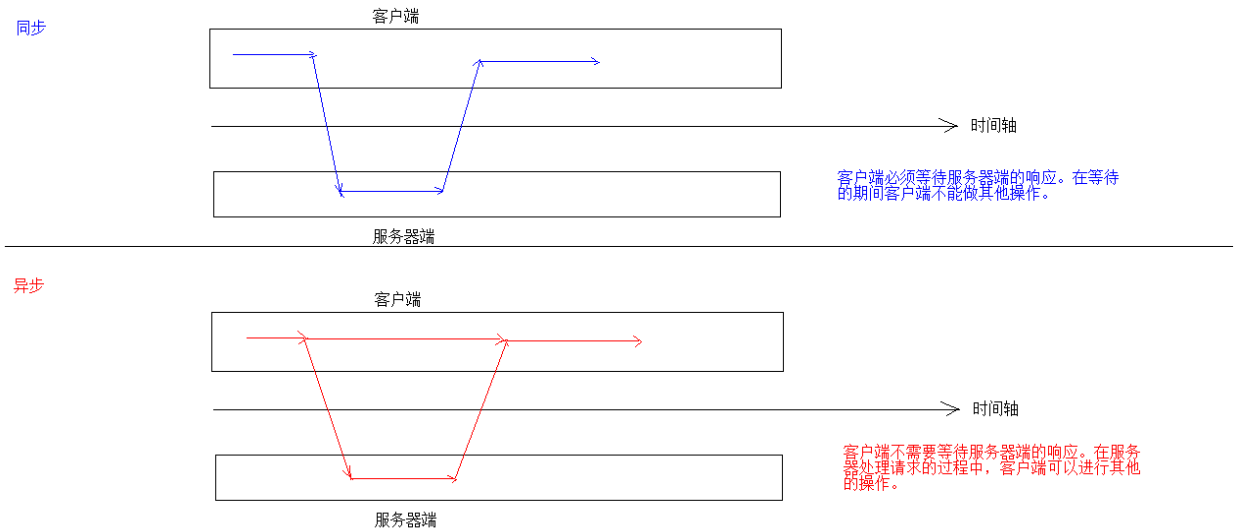
概念

> *ASynchronous JavaScript And XML 异步的JavaScript 和XML*

异步和同步：客户端和服务器端相互通信的基础上

- 客户端必须等待服务器端的响应。在等待的期间客户端不能做其他操作。
- 客户端不需要等待服务器端的响应。在服务器处理请求的过程中，客户端可以进行其他的操作。Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。[1] 通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。 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。

提升用户的体验



实现方式：

原生的JS实现方式（了解，开发不会用）

```
//1.创建核心对象
    var xmlhttp;
    if (window.XMLHttpRequest)
    {
        // code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }else
    {
// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }

//2. 建立连接
    /*
        参数:
        1. 请求方式:GET、POST
            - get方式,请求参数在URL后边拼接。send方法为空参
    - post方式,请求参数在send方法中定义
        2. 请求的URL:
        3. 同步或异步请求:true(异步)或 false(同步)
*/
    xmlhttp.open("GET","ajaxServlet?username=tom",true);


//3.发送请求
        xmlhttp.send();


//4.接受并处理来自服务器的响应结果

        //获取方式 : xmlhttp.responseText
        //什么时候获取?当服务器响应成功后再获取
        //当xmlhttp对象的就绪状态改变时,触发事件onreadystatechange。
        xmlhttp.onreadystatechange=function()
        {
            //判断readyState就绪状态是否为4,判断status响应状态码是否为200
            if (xmlhttp.readyState==4 && xmlhttp.status==200)
            {
                //获取服务器的响应结果
                var responseText = xmlhttp.responseText;
            alert(responseText);
            }
        }
```

JQeury实现方式

$.ajax()

- 语法:$.ajax({键值对});

```
//使用$.ajax()发送异步请求
$.ajax({
    url:"ajaxServlet1111" , // 请求路径

    type:"POST" , //请求方式

    //data: "username=jack&age=23",//请求参数
    data:{"username":"jack","age":23},

    success:function (data) {
        alert(data);
    },//响应成功后的回调函数

    error:function () {
        alert("出错啦...")
    },//表示如果请求响应出现错误，会执行的回调函数

    dataType:"text"//设置接受到的响应数据的格式
});
```

- *常用：*
    1. *url：路径*
    2. *type：请求数据的类型*
        - *get*
        - *post*
    3. *data：发送的数据*

    data：{"username":"zhangsan"}

    4. *success:函数：成功之后的回调函数*
    5. *error:函数：失败之后的回调函数*
    6. *dataType：接受到的相应数据的格式，如果不指定那么会自动进行响应头进行判断*
        - *xml*
        - *html*
        - *script*
        - *json*
        - *text*

$.get()：发送get请求

- 语法：$.get(url, [data], [callback], [type])
- 参数：
- url：请求路径，必选
- data：请求参数
- callback：回调函数
- type：响应结果的类型

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="../js/jquery-3.3.1.min.js"></script>
</head>
<body>


<script>

    $.get("ajaxServlet",{username:"zhangsan"},function (data) {
        alert(data);
    },"text")

</script>
</body>
</html>
```

$.post()：发送post请求

- 语法：$.post(url, [data], [callback], [type])
- 参数：
- url：请求路径，必选
- data：请求参数
- callback：回调函数
- type：响应结果的类型

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="../js/jquery-3.3.1.min.js"></script>
</head>
<body>


<script>

    $.post("ajaxServlet",{username:"zhangsan"},function (data) {
        alert(data);
    },"text")

</script>
</body>
</html>
```

JSON：

概念

> *JavaScript Object Notation JavaScript对象表示法*

```
Person p = new Person();
p.setName("张三");
p.setAge(23);
p.setGender("男");
```

以上就是java为各种数据封装为了一个对象，非常方便，那么javascript也想搞一个对象来封装这些零散的数据，那么这个时候就需要json：

```
var p = {"name":"张三","age":23,"gender":"男"};
```

- json早起的确是用于封装数据的，但是现在多用于存储和交换文本信息的语法
  - 进行数据的传输
  - JSON 比 XML 更小、更快，更易解析。

语法：

基本规则

- 数据在名称/值对中：json数据是由键值对构成的
  - 键用引号(单双都行)引起来，也可以不使用引号
  - 值得取值类型：
    1. 数字（整数或浮点数）
    2. 字符串（在双引号中）
    3. 逻辑值（true 或 false）
    4. 数组（在方括号中），比如{"persons":[{},{}]}
    5. 对象（在花括号中），比如{"address":{"province": "陕西"....}}
    6. null
  - 数据由逗号分隔：多个键值对由逗号分隔
  - 花括号保存对象：使用{}定义json 格式
  - 方括号保存数组：[]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="../js/jquery-3.3.1.min.js"></script>
```

```
</head>
<body>


<script>
        /*Json基本格式*/
        var person = {"name":"zhangsan","age":18,"gender":true}

        /*嵌套格式，对象里面嵌套数组*/
        var persons = {
                        "persons":[
                            {"name":"zhangsan","age":23,"gender":true},
                            {"name":"lisi","age":24,"gender":true}
                        ]
                    }
        /*嵌套格式2，数组里面嵌套对象*/
        var persons2 = [
                            {"name":"zhangsan","age":23,"gender":true},
                            {"name":"lisi","age":24,"gender":true}
                        ]

</script>
</body>
</html>
```

获取数据:

1. json对象.键名
2. json对象["键名"]
3. 数组对象[索引]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="../js/jquery-3.3.1.min.js"></script>
</head>
<body>


<script>

        var person = {"name":"zhangsan","age":18,"gender":true};

        /*获取*/
        var name = person.name;
```

```
        var person1 = {
                        "persons":[
                            {"name":"zhangsan","age":23,"gender":true},
                            {"name":"lisi","age":24,"gender":true}
                        ]
                    };

        /*获取：变量名.角标名[索引].属性*/
        var name1 = person1.persons[0].name;


        var person2 = [
                            {"name":"zhangsan","age":23,"gender":true},
                            {"name":"lisi","age":24,"gender":true}
                        ];
        /*获取*/
        var name2 = person2[0].name;

</script>
</body>
</html>
```

4. 遍历

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="../js/jquery-3.3.1.min.js"></script>
</head>
<body>


<script>

        var person = {"name":"zhangsan","age":18,"gender":true};

        var person2 = [
                            {"name":"zhangsan","age":23,"gender":true},
                            {"name":"lisi","age":24,"gender":true}
                    ];

        /*获取person*/
        for(var key in person){
            /*注意了，因为key获取到的是字符串格式的，
            所以如果person.key==person."key",
            这样是获取不到的*/
```

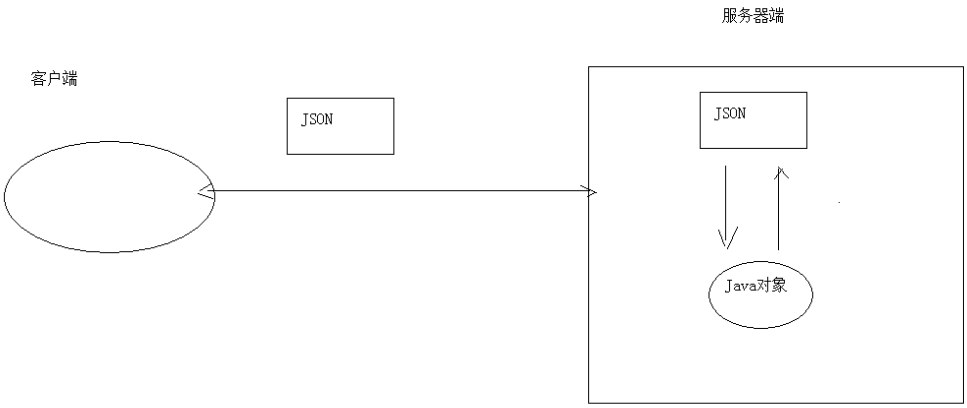
```
            alert(key+": "+person[key]);
        }

        /*获取嵌套数组中的值*/
        for(var i=0;i<person2.length;i++){
            for (var key in person2[i]){
                alert(key+": "+person2[i][key]);
            }
        }



</script>
</body>
</html>
```

JSON数据和Java对象的相互转换



- JSON解析器（封装好的工具类）：
  - 常见的解析器：Jsonlib(官方)，Gson(谷歌)，fastjson(阿里)，jackson(spring内置，学习)

JSON转为Java对象

1. 导入jackson的相关jar包
2. 创建Jackson核心对象 ObjectMapper
3. 调用ObjectMapper的相关方法进行转换
4. readValue(json字符串数据,Class)

```
package cn.itcast.domain;
```

```java
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Date;

public class Person {

    private String  name,
                    gender;

    private int     age;

    private Date birthday;

    public Person() {
    }

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

    public String getName() {
        return name;
    }

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

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getAge() {
        return age;
    }

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

    public Date getBirthday() {
        return birthday;
    }
```

```
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

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

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.*;

public class jacksonTest {

    /*json-->java*/
    @Test
    public void test1() throws IOException {

        /*假装获取到了json*/
        String json = "{\"gender\":\"男\",\"name\":\"张三\",\"age\":23}";

        /*创建ObjectMapper对象*/
        ObjectMapper mapper = new ObjectMapper();

        /*转换为java对象 Person对象*/
        Person person = mapper.readValue(json, Person.class);

        System.out.println(person);
        //Person{name='张三', gender='男', age=23, birthday=null}
    }
}
```

Java对象转换JSON

使用步骤：

导入jackson的相关jar包

- jackson-annotations-2.2.3.jar
- jackson-core-2.2.3.jar
- jackson-databind-2.2.3.jar

创建Jackson核心对象 ObjectMapper

调用ObjectMapper的相关方法进行转换

转换方法：

- writeValue(参数1，obj)：参数1：
  - File：将obj对象转换为JSON字符串，并保存到指定的文件中
  - Writer：将obj对象转换为JSON字符串，并将json数据填充到字符输出流中
  - OutputStream：将obj对象转换为JSON字符串，并将json数据填充到字节输出流中
- writeValueAsString(obj):将对象转为json字符串

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {
        /*创建Person对象*/
        Person person = new Person();

        person.setName("张三");
        person.setAge(23);
        person.setGender("男");


        /*创建Jackson和新对象 ObjectMapper*/
        ObjectMapper mapper = new ObjectMapper();

        /*转换
        * 转换方法:
        *   writeValue(参数1, obj)
        *       参数1:
        *           File: 将obj对象转换为Json字符串，并保存到指定的文件中
```

```
        *            Writer：将obj对象转换为Json字符串，并将Json数据填充到字符输出流中
        *            OutputStream：将obj对象转换为Json字符串，并将Json数据填充到字节输出流中
        *
        *   writeValueAsString(obj)：将对象转为Json字符串，然后使用response输出流输出出去
        * */

        //转为字符串
        String json = mapper.writeValueAsString(person);
        //这里因为没写Servlet就不输出了
        System.out.println(json);//{"name":"张三","gender":"男","age":23}


        /*将文件写到E://a.txt中*/
        mapper.writeValue(new File("e://a.txt"),person);


        /*将数据关联到Write中。一次性的做了：转Json+输出*/
        mapper.writeValue(response.getWriter(),person);
    }

}
```

注解：

```
package cn.itcast.domain;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Date;

public class Person {

    private String  name,
                    gender;

    private int     age;

    private Date birthday;

    public Person() {
    }

    public Person(String name, String gender, int age, Date birthday) {
        this.name = name;
        this.gender = gender;
        this.age = age;
```

```
        this.birthday = birthday;
    }

    public String getName() {
        return name;
    }

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

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getAge() {
        return age;
    }

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

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

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

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.Date;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {

        Person person = new Person();

        person.setName("张三");
        person.setAge(23);
        person.setGender("男");

        /*生成new Date()*/
        person.setBirthday(new Date());


        ObjectMapper mapper = new ObjectMapper();

        String json = mapper.writeValueAsString(person);

        System.out.println(json);
        /*
        * {"name":"张三","gender":"男","age":23,"birthday":1558572554110}
        * 这样的birthday实在是太难看，所以有两种解决方案，一种是直接不让其显示，另一种是转换一下格式，推荐转换格式
        * */


    }
}
```

1. @JsonIgnore：排除属性，就是说以后

```
package cn.itcast.domain;

import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Date;

public class Person {
```

```
    private String  name,
                    gender;

    private int     age;


    @JsonIgnore //使用这个主角来忽略显示
    private Date birthday;

    public Person() {
    }

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

    public String getName() {
        return name;
    }

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

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getAge() {
        return age;
    }

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

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }
```

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

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.Date;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {

        Person person = new Person();

        person.setName("张三");
        person.setAge(23);
        person.setGender("男");

        /*生成new Date()*/
        person.setBirthday(new Date());


        ObjectMapper mapper = new ObjectMapper();

        String json = mapper.writeValueAsString(person);

        System.out.println(json);
        /*
        * {"name":"张三","gender":"男","age":23}
        * */


    }
}
```

2. @JsonFormat：属性值的格式化
    - @JsonFormat(pattern = "yyyy-MM-dd")

```
package cn.itcast.domain;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Date;

public class Person {

    private String  name,
                    gender;

    private int     age;


    @JsonFormat(pattern = "yyyy-MM-dd") //使用这个主角来格式化
    private Date birthday;

    public Person() {
    }

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

    public String getName() {
        return name;
    }

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

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getAge() {
        return age;
```

```
    }

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

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

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

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.Date;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {

        Person person = new Person();

        person.setName("张三");
        person.setAge(23);
        person.setGender("男");

        /*生成new Date()*/
        person.setBirthday(new Date());
```

```
        ObjectMapper mapper = new ObjectMapper();

        String json = mapper.writeValueAsString(person);

        System.out.println(json);
        /*
        * {"name":"张三","gender":"男","age":23}
        * */

    }
}
```

复杂java对象转换

1. List：数组

```
package cn.itcast.domain;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Date;

public class Person {

    private String  name,
                    gender;

    private int     age;


    @JsonFormat(pattern = "yyyy-MM-dd") //使用这个主角来格式化
    private Date birthday;

    public Person() {
    }

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

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

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

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getAge() {
        return age;
    }

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

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

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

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Date;
import java.util.List;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {

        Person person = new Person();
        person.setName("张三");
        person.setAge(23);
        person.setGender("男");
        person.setBirthday(new Date());

        Person person1 = new Person();
        person.setName("张三");
        person.setAge(23);
        person.setGender("男");
        person.setBirthday(new Date());

        Person person2 = new Person();
        person.setName("张三");
        person.setAge(23);
        person.setGender("男");
        person.setBirthday(new Date());



        List<Person> list = new ArrayList<Person>();

        list.add(person);
        list.add(person1);
        list.add(person2);

        ObjectMapper mapper = new ObjectMapper();

        String json = mapper.writeValueAsString(list);

        System.out.println(json);
        /*
        * [
        * {"name":"张三","gender":"男","age":23,"birthday":"2019-05-23"},
        * {"name":null,"gender":null,"age":0,"birthday":null},
        * {"name":null,"gender":null,"age":0,"birthday":null}
        * ]
        * */

    }
}
```

2. Map：对象格式一致

```
package cn.itcast.test;

import cn.itcast.domain.Person;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.*;

public class jacksonTest {

    /*java对象-->json*/
    @Test
    public void test1() throws IOException {

        Map<String,Object> map = new HashMap<String,Object>();

        map.put("name","张三");
        map.put("age",23);
        map.put("gender","男");


        ObjectMapper mapper = new ObjectMapper();

        String json = mapper.writeValueAsString(map);

        System.out.println(json);
        //{"gender":"男","name":"张三","age":23}
    }
}
```

案例

- 校验用户名是否存在
    1. 服务器响应的数据，在客户端使用时，要想当做json数据格式使用。有两种解决方案：
        1. $.get()/$.post()/$.ajax():将参数type指定为"json"

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>注册页面</title>
    <script src="js/jquery-3.3.1.min.js"></script>


    <script>
        //在页面加载完成后
        $(function () {
            //给username绑定blur事件
            $("#username").blur(function () {
                //获取username文本输入框的值
                var username = $(this).val();
                //发送ajax请求
                //期望服务器响应回的数据格式:
{"userExsit":true,"msg":"此用户名太受欢迎,请更换一个"}
                //
{"userExsit":false,"msg":"用户名可用"}
                $.get("findUserServlet",
{username:username},function (data) {
                    //判断userExsit键的值是否是true

                    // alert(data);
                    var span = $("#s_username");
                    if(data.userExsit){
                        //用户名存在
                        span.css("color","red");
                        span.html(data.msg);
                    }else{
                        //用户名不存在
                        span.css("color","green");
                        span.html(data.msg);
                    }
                },"ajax");

            });
        });

    </script>
</head>
<body>


    <form>

        <input type="text" id="username" name="username"
placeholder="请输入用户名">
        <span id="s_username"></span>
        <br>
        <input type="password" name="password" placeholder="请输
入密码"><br>
```

```
        <input type="submit" value="注册"><br>

    </form>

</body>
</html>
```

```
package cn.itcast.web.servlet;

import com.fasterxml.jackson.databind.ObjectMapper;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

@WebServlet("/findUserServlet")
public class FindUserServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //1.获取用户名
        String username = request.getParameter("username");

        //2.调用service层判断用户名是否存在

        //期望服务器响应回的数据格式：{"userExsit":true,"msg":"此用户
名太受欢迎,请更换一个"}
        //                          {"userExsit":false,"msg":"用户
名可用"}

        Map<String,Object> map = new HashMap<String,Object>();

        if("tom".equals(username)){
            //存在
            map.put("userExsit",true);
            map.put("msg","此用户名太受欢迎,请更换一个");
        }else{
            //不存在
            map.put("userExsit",false);
            map.put("msg","用户名可用");
        }

        //将map转为json, 并且传递给客户端
        //将map转为json
        ObjectMapper mapper = new ObjectMapper();
```

```
            //并且传递给客户端
            mapper.writeValue(response.getWriter(),map);


        }

        protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
            this.doPost(request, response);
        }
}
```

2. 在服务器端设置MIME类型
response.setContentType("application/json;charset=utf-8");

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>注册页面</title>
    <script src="js/jquery-3.3.1.min.js"></script>


    <script>
        //在页面加载完成后
        $(function () {
            //给username绑定blur事件
            $("#username").blur(function () {
                //获取username文本输入框的值
                var username = $(this).val();
                //发送ajax请求
                //期望服务器响应回的数据格式:
{"userExsit":true,"msg":"此用户名太受欢迎,请更换一个"}
                //
{"userExsit":false,"msg":"用户名可用"}
                $.get("findUserServlet",
{username:username},function (data) {
                    //判断userExsit键的值是否是true

                    // alert(data);
                    var span = $("#s_username");
                    if(data.userExsit){
                        //用户名存在
                        span.css("color","red");
                        span.html(data.msg);
                    }else{
                        //用户名不存在
                        span.css("color","green");
```

```
                        span.html(data.msg);
                    }
                });

            });
        });

    </script>
</head>
<body>


    <form>

        <input type="text" id="username" name="username"
placeholder="请输入用户名">
        <span id="s_username"></span>
        <br>
        <input type="password" name="password" placeholder="请输
入密码"><br>
        <input type="submit" value="注册"><br>

    </form>

</body>
</html>
```

```
package cn.itcast.web.servlet;

import com.fasterxml.jackson.databind.ObjectMapper;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

@WebServlet("/findUserServlet")
public class FindUserServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        //1.获取用户名
        String username = request.getParameter("username");

        //2.调用service层判断用户名是否存在
```

```
        //期望服务器响应回的数据格式：{"userExsit":true,"msg":"此用户名太受欢迎,请更换一个"}
        //                          {"userExsit":false,"msg":"用户名可用"}

        //设置响应的数据格式为json
        response.setContentType("application/json;charset=utf-8");


        Map<String,Object> map = new HashMap<String,Object>();

        if("tom".equals(username)){
            //存在
            map.put("userExsit",true);
            map.put("msg","此用户名太受欢迎,请更换一个");
        }else{
            //不存在
            map.put("userExsit",false);
            map.put("msg","用户名可用");
        }

        //将map转为json，并且传递给客户端
        //将map转为json
        ObjectMapper mapper = new ObjectMapper();
        //并且传递给客户端
        mapper.writeValue(response.getWriter(),map);


    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doPost(request, response);
    }
}
```