AJAX

Author：Amy

Version：9.0.2

一、引言

1.1 什么是JSON

> JSON(JavaScript Object Notation, JS 对象标记) 是一种轻量级的数据交换格式。它基于 ECMAScript (W3C制定的JS规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

1.2 JSON语法

> - [] 表示数组
> - {} 表示对象
> - "" 表示是属性名或字符串类型的值
> - : 表示属性和值之间的间隔符
> - , 表示多个属性的间隔符或者是多个元素的间隔符

二、JSON解析【 重点 】

> 要解析的字符串：将字符串解析为Java对象

```
//对象嵌套数组嵌套对象
String json1="{'id':1,'name':'JAVAEE-1703','stus':[{'id':101,'name':'刘一','age':16}]}";
//数组
String json2="['北京','天津','杭州']";
```

> - 初始的类：
>   - Student.java
>   - Grade.java

```java
public class Student {
    private int id;
    private String name;
    private int age;
    //此处省略get和set方法
}
```

```java
public class Grade {
    private int id;
    private String name;
    private ArrayList<Student> stus;
    //此处省略get和set方法
}
```

2.1 FastJSON解析

> - Fastjson 是一个 Java 库，可以将 Java 对象转换为 JSON 格式，当然它也可以将 JSON 字符串转换为 Java 对象
> - 提供了 toJSONString() 和 parseObject() 方法来将 Java 对象与 JSON 相互转换：
>   - 调用toJSONString方 法即可将对象转换成 JSON 字符串
>   - parseObject 方法则反过来将 JSON 字符串转换成对象。

> parseObject方法：字符串转换成对象

```java
public class FASTJson {
    //解析
    @Test
    public void test1() {
        // 对象嵌套数组嵌套对象
        String json1 = "{'id':1,'name':'JAVAEE-1703','stus':[{'id':101,'name':'刘铭','age':16}]}";
        // 数组
        String json2 = "['北京','天津','杭州']";
        //1、
        //静态方法
        Grade grade=JSON.parseObject(json1, Grade.class);
        System.out.println(grade);
        //2、
        List<String> list=JSON.parseArray(json2, String.class);
        System.out.println(list);
    }
}
```

> toJSONString方法：对象转换成 JSON 字符串

```java
public class FASTJson {
//生成
    @Test
    public void test2(){
        ArrayList<Student> list=new ArrayList<>();
        for(int i=1;i<3;i++){
            list.add(new Student(101+i, "码子", 20+i));
        }
        Grade grade=new Grade(100001,"张三", list);
        String json=JSON.toJSONString(grade);
        System.out.println(json);
    }
}
```

2.2 Jackson解析

> - Jackson 是一个能够将Java对象序列化为JSON字符串，也能够将JSON字符串反序列化为Java对象的框架；
> - 通过方法readValue和writeValue实现；

```java
public class JackSonTest {
    //解析
    @Test
    public void test1() throws Exception{
        // 对象嵌套数组嵌套对象
        String json1 = "{\"id\":1,\"name\":\"JAVAEE-1703\",\"stus\":[{\"id\":101,\"name\":\"刘一\",\"age\":16}]}";
        // 数组
        String json2 = "[\"北京\",\"天津\",\"杭州\"]";
        //1、
        ObjectMapper mapper=new ObjectMapper();
        Grade grade=mapper.readValue(json1, Grade.class);
        System.out.println(grade);
        //2、
        ArrayList<String> list=mapper.readValue(json2,
                new TypeReference<ArrayList<String>>() {
        });
        System.out.println(list);
    }
    //生成
    @Test
    public void test2() throws JsonProcessingException{
```

```
        ArrayList<Student> list=new ArrayList<>();
        for(int i=1;i<3;i++){
            list.add(new Student(101+i, "码子", 20+i));
        }
        Grade grade=new Grade(100001,"张三", list);
        ObjectMapper mapper=new ObjectMapper();
        //将对象转换为JSON格式字符串
        String json=mapper.writeValueAsString(grade);
        System.out.println(json);
    }
}
```

2.3 浏览器处理JSON字符串

> - JSON.stringify()

```
var json={name:'zs',age:34};
var str=JSON.stringify(json);
alert(str);
```

2.4 流览器转换为json对象

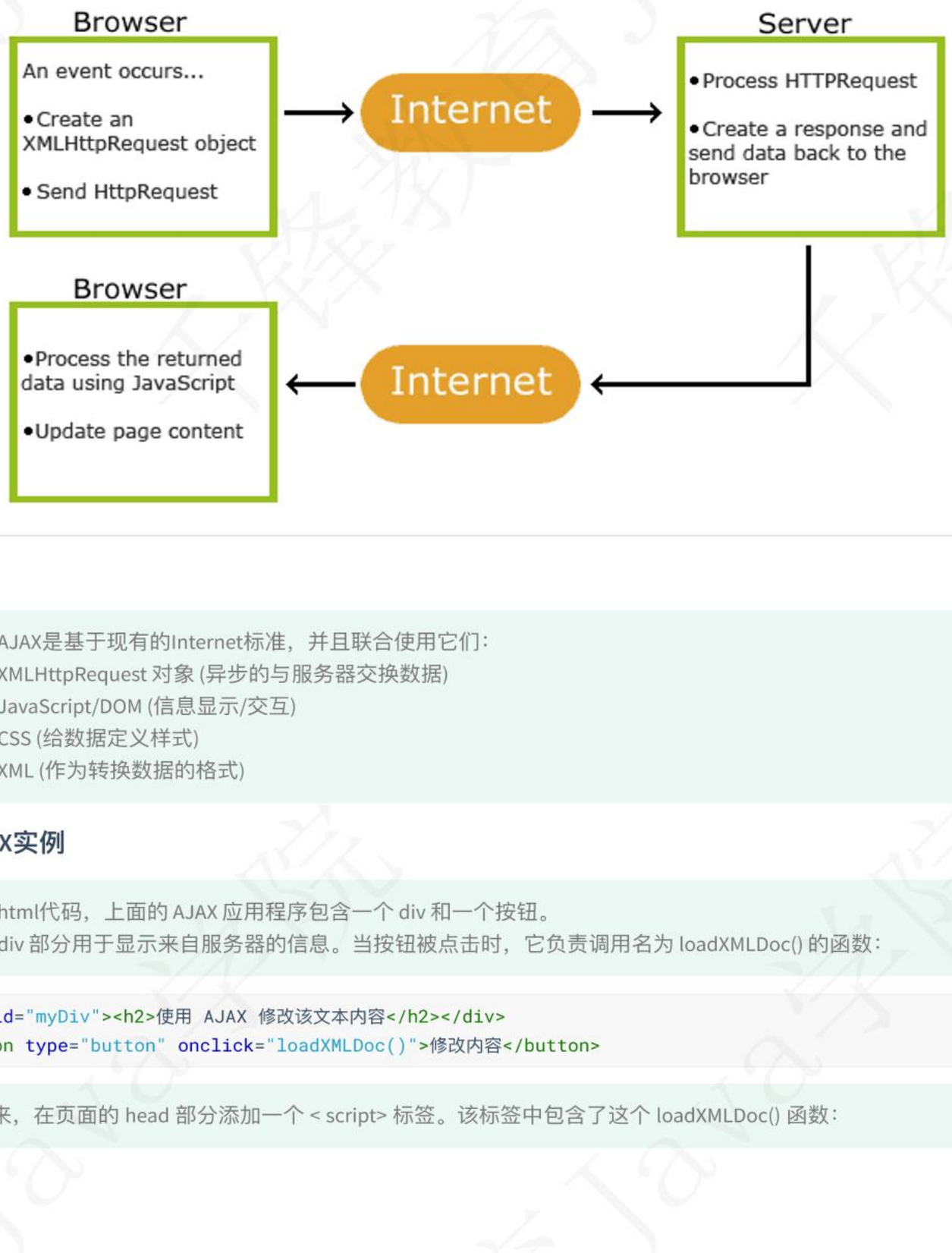
> JSON.parse()

三、Ajax使用【 重点 】

3.1 什么是AJAX?

> - 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。
> - AJAX = Asynchronous异步 JavaScript and XML。
> - AJAX 是一种用于创建快速动态网页的技术。
> - 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，必需重载整个网页。

3.2 AJAX工作原理

AJAX工作原理


> - AJAX是基于现有的Internet标准，并且联合使用它们：
> - XMLHttpRequest 对象 (异步的与服务器交换数据)
> - JavaScript/DOM (信息显示/交互)
> - CSS (给数据定义样式)
> - XML (作为转换数据的格式)

3.3 AJAX实例

> - html代码，上面的 AJAX 应用程序包含一个 div 和一个按钮。
> - div 部分用于显示来自服务器的信息。当按钮被点击时，它负责调用名为 loadXMLDoc() 的函数：

```
<div id="myDiv"><h2>使用 AJAX 修改该文本内容</h2></div>
<button type="button" onclick="loadXMLDoc()">修改内容</button>
```

> 接下来，在页面的 head 部分添加一个 < script> 标签。该标签中包含了这个 loadXMLDoc() 函数：

```
<head>
<script>
function loadXMLDoc()
{
    .... AJAX 脚本执行 ...
}
</script>
</head>
```

3.4 创建XMLHttpRequest对象

- XMLHttpRequest对象是AJAX的基础。
- 所有现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。
- XMLHttpRequest 用于在后台与服务器交换数据。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。
- 所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。创建 XMLHttpRequest 对象的语法：

```
var xmlhttp=new XMLHttpRequest();
```

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

```
var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
```

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject：

```
var xmlhttp;
if (window.XMLHttpRequest)
{
    //  IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码
    xmlhttp=new XMLHttpRequest();
}
else
{
    // IE6, IE5 浏览器执行代码
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
```

3.5 XMLHttpRequest请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

方法	描述
open(*method* , *url* , *async*)	规定请求的类型、URL 以及是否异步处理请求。 *method* ： 请求的类型；GET 或 POST； *url* ： 文件在服务器上的位置； *async* ： true（异步）或 false（同步），并且XMLHttpRequest 对象如果要用于 AJAX 的话，其 open() 方法的 async 参数必须设置为 true；
send(*string*)	将请求发送到服务器。 *string* ： 仅用于 POST 请求

- GET 还是 POST?
- 与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。
- 然而，在以下情况中，请使用 POST 请求：
    - 无法使用缓存文件（更新服务器上的文件或数据库）
    - 向服务器发送大量数据（POST 没有数据量限制）
    - 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

GET 请求

```
//示例一：一个简单的 GET 请求:
xmlhttp.open("GET","/try/ajax/demo_get.php",true);
xmlhttp.send();
//示例二：在上面的例子中，您可能得到的是缓存的结果，为了避免这种情况，请向 URL 添加一个唯一的 ID:
xmlhttp.open("GET","/try/ajax/demo_get.php?t=" + Math.random(),true);
xmlhttp.send();
//示例三：如果您希望通过 GET 方法发送信息，请向 URL 添加信息:
xmlhttp.open("GET","/try/ajax/demo_get2.php?fname=Henry&lname=Ford",true);
xmlhttp.send();
```

POST 请求

```
//示例一：一个简单 POST 请求
xmlhttp.open("POST","/try/ajax/demo_post.php",true);
xmlhttp.send();
//如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据：
xmlhttp.open("POST","/try/ajax/demo_post2.php",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

方法	描述
setRequestHeader(*header,value*)	向请求添加 HTTP 头。 *header* : 规定头的名称 *value* : 规定头的值

- 对于 web 开发人员来说，发送异步请求是一个巨大的进步。很多在服务器执行的任务都相当费时。AJAX 出现之前，这可能会引起应用程序挂起或停止。
- 通过 AJAX，JavaScript 无需等待服务器的响应，而是：
  - 在等待服务器响应时执行其他脚本
  - 当响应就绪后对响应进行处理
- 当使用Async=true时，请规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数：

```
//绑定执行函数：
xmlhttp.onreadystatechange=function()
{
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
        document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
}
xmlhttp.open("GET","/try/ajax/ajax_info.txt",true);
xmlhttp.send();
```

- 如需使用 async=false，请将 open() 方法中的第三个参数改为 false：
- 我们不推荐使用 async=false，但是对于一些小型的请求，也是可以的。
- 请记住，JavaScript 会等到服务器响应就绪才继续执行。如果服务器繁忙或缓慢，应用程序会挂起或停止。
- **注意：** 当您使用 async=false 时，请不要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

3.6 readyState

- 每当 readyState 改变时，就会触发 onreadystatechange 事件。
- 在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。
- readyState 属性存有 XMLHttpRequest 的状态信息。
- 当 readyState 等于 4 且状态为 200 时，表示响应已就绪：
- 下面是 XMLHttpRequest 对象的三个重要的属性：

属性	描述
onreadystatechange	存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。
readyState	存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。0: 请求未初始化1: 服务器连接已建立2: 请求已接收3: 请求处理中4: 请求已完成，且响应已就绪
status	例：200: "OK"； 404: 未找到页面

```
xmlhttp.onreadystatechange=function()
{
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
        document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
}
```

响应码	描述
100	客户必须继续发出请求
101	客户要求服务器根据请求转换HTTP协议版本
200	交易成功
201	提示知道新文件的URL
202	接受和处理、但处理未完成
203	返回信息不确定或不完整
204	请求收到，但返回信息为空
205	服务器完成了请求，用户代理必须复位当前已经浏览过的文件
206	服务器已经完成了部分用户的GET请求
300	请求的资源可在多处得到
301	删除请求数据
302	在其他地址发现了请求数据
303	建议客户访问其他URL或访问方式
304	客户端已经执行了GET，但文件未变化
305	请求的资源必须从服务器指定的地址得到
306	前一版本HTTP中使用的代码，现行版本中不再使用
307	申明请求的资源临时性删除
400	错误请求，如语法错误
401	请求授权失败
402	保留有效ChargeTo头响应
403	请求不允许
404	没有发现文件、查询或URl
405	用户在Request-Line字段定义的方法不允许
406	根据用户发送的Accept拖，请求资源不可访问
407	类似401，用户必须首先在代理服务器上得到授权
408	客户端没有在用户指定的饿时间内完成请求
409	对当前资源状态，请求不能完成
410	服务器上不再有此资源且无进一步的参考地址
411	服务器拒绝用户定义的Content-Length属性请求
412	一个或多个请求头字段在当前请求中错误
413	请求的资源大于服务器允许的大小
414	请求的资源URL长于服务器允许的长度
415	请求资源不支持请求项目格式
416	请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段
417	服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求
500	服务器产生内部错误
501	服务器不支持请求的函数
502	服务器暂时不可用，有时是为了防止发生系统过载
503	服务器过载或暂停维修
504	关口过载，服务器使用另一个关口或服务来响应用户，等待时间设定值较长
505	服务器不支持或拒绝支请求头中指定的HTTP版本

3.7 XMLHttpRequest响应

> 如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

属性	描述
responseText	获得字符串形式的响应数据。
responseXML	获得 XML 形式的响应数据。

- responseText 属性
  - 如果来自服务器的响应并非 XML，请使用 responseText 属性。
  - responseText 属性返回字符串形式的响应，因此您可以这样使用：

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

- responseXML 属性
  - 如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 responseXML 属性：

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
{
    txt=txt + x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("myDiv").innerHTML=txt;
```

3.8 使用回调函数

- 回调函数是一种以参数形式传递给另一个函数的函数。
- 如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个 *标准* 的函数，并为每个 AJAX 任务调用该函数。
- 该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）：

```
function myFunction()
{
    loadXMLDoc("/try/ajax/ajax_info.txt",function()
    {
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
        }
    });
}
```

四、AJAX的使用

4.1 AJAX与服务器交互

- 模拟登陆验证
- 验证用户是否可以注册，利用AJAX技术!进行动态验证
  - 编写注册页面

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <base href="<%=basePath%>">

  <title>My JSP 'index.jsp' starting page</title>
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<script type="text/javascript" src="./ajax.js"></script>
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->
</head>

<body>
    <center>
       <font color="red" size="7">qq注册页面</font>
       <input type="text" name="username" onkeyup="kp(this)" /> <span id="sp"></span> <br/>
       <input type="password" name="password"/><br/>
```

```
    <input type="submit" value="注册"/>
</center>
<script type="text/javascript">

  //当用户名输入框输入内容就调用此方法
  function kp(ipt){

   //1.获取input输入框的value
     var value = ipt.value;
   //2.进行验证
     if(value != null && value !=""){
          //1-5
          //1.创建Ajax
          var xmlhttp = getAjax();

          //2.设置状态改变监听
          xmlhttp.onreadystatechange = function(){

            //5获取响应数据
            if(xmlhttp.readyState == 4 && xmlhttp.status ==200)
            {

  var result = xmlhttp.responseText; //获取结果

  //  1 行    2 不行

  //1.找到span标签

  var sp = document.getElementById("sp");

  if(result == "1"){
    //成功的 span 提示一句绿色的话
    sp.innerHTML=""; //清空
    var ft = document.createElement("font");
    var fttext = document.createTextNode("恭喜您!可以注册!!"); //文本标签
    ft.setAttribute("color", "green");
    ft.appendChild(fttext);
    sp.appendChild(ft);
  }else{
    //失败的 span 提示一句红色的话
    sp.innerText="用户已经被注册!";
    sp.style.color = "red";
  }
}
            }
            //3.设置ajax method  url
            xmlhttp.open("POST",
             "${pageContext.request.contextPath}/servlet/DealServlet");                    //4.发送请求
            //设置一个请求头
            xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
            xmlhttp.send("value="+value);

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

4.2 编写AJAX处理servlet

```
public class DealServlet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //0.设置编码格式
    request.setCharacterEncoding("UTF-8");
      response.setContentType("text/html;charset=UTF-8");
    //1.
      String value = request.getParameter("value");
    //2.
      String  result = null;
      if (value.equals("root") | value.equals("admin")) {
       result = "2";
       //代表已经存在
    }else{
      //可以注册
      result = "1";
    }
    //3.
    response.getWriter().write(result);
    }
}
```