第十四章：Ajax

第1节：Ajax简介

> 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 是一种用于创建快速动态网页的技术。
> 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。
> 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。

第2节：同步与异步

> 1. 同步：
> 发送一个请求，需要等待响应返回，然后才能够发送下一个请求，如果该请求没有响应，不能发送下一个请求，客户端会处于一直等待过程中。

> 2. 异步：
> 发送一个请求，不需要等待响应返回，随时可以再发送下一个请求，即不需要等待。

第3节：Ajax的应用场景

> 1.在线视频、直播平台等…评论实时更新、点赞、小礼物、…
> 2.会员注册时的信息验证，手机号、账号唯一
> 3.百度关键搜索补全功能

第4节：实现

4.1 原生实现

> 实现步骤：
> 1. 定义一个XMLHttpRequest核心对象xhr;
> 2. 通过xhr.open方法给当前对象提供访问方式、URL等。
> 3. 发送当前的请求至指定的URL
> 4. 接收返回值并处理

> 案例需求：
> 前台页面通过一个按钮向后台发送一个Ajax请求，后台做完处理后向前台页面响应一段文本信息显示在页面上。

> 页面代码：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
```

```
<head>
<title>Title</title>
<script type="text/javascript">
function testJsAjax(){
  //1. 创建核心对象
  var xmlhttp = new XMLHttpRequest();
  //2.通过核心对象方法给当前的对象提供访问方式和URL路径
  xmlhttp.open("GET","jsAjax?name=liuyan",true);
  //3.发送当前的请求至指定的URL
  xmlhttp.send();
  //4.接收返回值并处理
  xmlhttp.onreadystatechange=function(){
      //xmlhttp.readyState==4代表XMLHttpRequest对象读取服务器的响应结束
      //xmlhttp.status==200响应成功
      if (xmlhttp.readyState==4 && xmlhttp.status==200){
          var msg = xmlhttp.responseText;
          document.getElementById("msg").innerHTML=msg;
      }
  }
}
</script>
</head>
<body>
<div id="msg"></div>
<input type="button" name="btn" value="JS原生方式实现异步" onclick="testJsAjax()">
</body>
</html>
```

Servlet代码:

```
package com.ujiuye.js;

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;

@WebServlet("/jsAjax")
public class JsAjaxServlet extends HttpServlet {

@Override
protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=utf-8");
//获取名称
String name = request.getParameter("name");
System.out.println(name);
//响应
```

```
response.getWriter().write("响应成功");
}
}
```

4.2 JQuery实现

JS版的Ajax仅做为了解，我们重点学习jQuery版的Ajax，jQuery是一个优秀的js框架，自然对JS原生的Ajax进行了封装，封装后的Ajax的操作方法更简洁，功能更强大，这也是程序员最普遍使用，语法结构简单，代码可读性好。
与Ajax操作相关的jQuery方法经常使用的有三种：

请求方式	语法
Ajax请求	$.ajax([settings])
GET请求	$.get(url, *[data]*, *[callback]*, *[type]*)
POST请求	$.post(url, *[data]*, *[callback]*, *[type]*)

4.2.1 Ajax请求

语法格式：

```
$.ajax({
  url:"",
  data:{},
  type:"post/get",
  async:true,
  dataType:"text",
  success:function(obj){

  },
  error:function(){

  }
})
```

参数名称	描述
url	请求的服务器端url地址, 与form表单中的action一样，都是代表请求的路径
data	前台需要向后台传递的数据（键值对形式）
type	和form表单中method对应，代表请求类型 get/post
async	取值是布尔类型true/false，分别表示异步和同步，默认为true(异步)，一般不建议写
dataType	回传的数据类型。text、xml、html、json...
success	成功的回调函数,参数obj表示返回值
error	失败的回调函数，一般不写

注意事项：
1、每个属性后都要跟随一个英文逗号，最后一个不用。
2、每一个属性都是键值对的形式存在，中间用英文冒号:隔开
3、data:{} 是一个特殊的写法，值是一个{}，里面使用键值对存储
例如：data:{“键1”:值1, “键2”:值2, “键3”:值3}
4、以上属性没有先后顺序要求

4.2.2 GET请求

语法格式：

```
$.get(url, [data], [callback], [type]);
```

参数名称	描述
url	请求的服务器端url地址, 与form表单中的action一样，都是代表请求的路径
data	前台需要向后台传递的数据（键值对形式）
callback	当请求成功后的回掉函数，可以在函数体中编写我们的逻辑代码
type	预期的返回数据的类型，取值可以是 text、xml、html、json...

注意事项：
这种写法功能和$.ajax是一样的，但是严格要求属性顺序。

4.2.3 POST请求

语法格式：

```
$.post(url, [data], [callback], [type]);
```

参数名称	描述
url	请求的服务器端url地址, 与form表单中的action一样，都是代表请求的路径
data	前台需要向后台传递的数据（键值对形式）
callback	当请求成功后的回掉函数，可以在函数体中编写我们的逻辑代码
type	预期的返回数据的类型，取值可以是 text、xml、html、json...

注意事项：

这种写法功能和$.ajax是一样的，但是严格要求属性顺序。

4.2.4 异同

相同点：

都是jQuery封装的方法实现异步交互。

不同点：

$.ajax()是jQuery的第一次封装，使用时稍显麻烦，但是功能强大，覆盖了get和post请求，有错误调试能力，写法顺序可以改变。

$.post()和$.get()是jQuery Ajax的第二次封装，由于$.Ajax()写法过于臃肿，简化为$.post()和$.get()，功能是相同的没有区别。但是写法要求更高，顺序不能改变。

第5节：案例

案例需求：校验用户名唯一

在用户注册页面，输入用户名，当用户名输入框失去焦点时，发送异步请求，将输入框的用户名传递给服务器端进行是否存在的校验。

页面代码：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>Title</title>
<script src="./js/jquery-3.3.1.js"></script>
<script>
$(function(){
  $("#uname").blur(function(){
      var uname = $("#uname").val();
      //判断用户名不为空
      var req = /^\s*$/;
```

```
        if(req.test(uname)){
            $("#msg").html("用户名不能为空").css("color","red");
            return;
        }
        //发送ajax请求
        $.ajax({
            url:"${pageContext.request.contextPath}/ckeckUsername",
            data:{"uname":uname},
            type:"post",
            dataType:"text",
            success:function(obj){
                //判断
                if(obj){
                    $("#msg").html("该用户名已被占用").css("color","red");
                }else{
                    $("#msg").html("该用户名可用").css("color","green");
                }
            }
        })
    })
})
</script>
</head>
<body>
<p>
用户名: <input type="text" id="uname" name="uname">
<span id="msg"></span>
</p>
<p>
<input type="button" value="注册">
</p>
</body>
</html>
```

Servlet代码:

```
package com.ujiuye.ajax;

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;

@WebServlet("/ckeckUsername")
public class CheckUsernameServlet extends HttpServlet {

@Override
```

```
public void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
request.setCharacterEncoding("utf-8");
response.setContentType("text/html;charset=utf-8");
String uname = request.getParameter("uname");
Boolean flag;
//判断
if("liuyan".equals(uname)){
  flag = true;
}else{
  flag = false;
}
//响应
response.getWriter().print(flag);
}
}
```

第6节：返回值类型

6.1 介绍

Ajax支持多种返回值类型：
  XML：太麻烦，解析太费劲，已经不使用
  HTML：网页，其实质和返回文本一样，没区别，一般使用文本代替
  Script： 直接返回脚本
  Text（文本）：（默认返回类型）字符串类型，返回直接接收字符串
  Json：返回是一个js对象，脚本中可以直接操作这个对象，非常方便
  Jsonp：和Json一样，只不过这个支持跨域访问。

6.2 JSON

6.2.1 简介

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。

6.2.2 语法

JSON对象的语法有三种：对象、数组、混合模式

类型	语法	解释
对象类型	{name:value,name:value...}	其中name是字符串类型，而value是任意类型
数组类型	[value,value,value...]	其中value是任意类型
混合类型	[{},{}... ...] 或 {name:[]... ...}	合理包裹嵌套对象类型和数组类型

6.2.3 书写规范

1.使用{}和[]书写，{}表示对象类型，[]表示数组类型
2.对象类型中每组数据之间用逗号隔开，每组中的关键字与值之间用冒号隔开
3.数组类型中每个值之间用逗号隔开，最后一个值后面不要加逗号

6.2.4 工具介绍

1. JSON在线解析工具：http://www.bejson.com/
作用：
   1.校验JSON数据的语法是否正确
   2.将不规范的JSON格式数据进行格式化

2. 常见的json转换工具
   json的转换工具是通过java封装好的一些jar工具包，直接将java对象或集合转换成json格式的字符串。

工具名称	介绍
Jsonlib	Java 类库，需要导入的jar包较多
Gson	google提供的一个简单的json转换工具
Fastjson	alibaba技术团队提供的一个高性能的json转换工具
Jackson	开源免费的json转换工具，springmvc转换默认使用jackson

6.2.5 应用

1. Jackson使用：
   1）导入json相关jar包
   2）创建java对象或集合
   3）使用jackson的ObjectMapper对象的writeValueAsString方法进行转换

```
package com.ujiuye;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.ujiuye.pojo.User;

import javax.servlet.ServletException;
```

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

@WebServlet("/jackson")
public class jackson extends HttpServlet {

@Override
protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  //创建User对象
  User user = new User();
  user.setUid(1001);
  user.setUsername("柳岩");
  user.setPassword("123");
  //创建List集合
  List<String> list = new ArrayList<>();
  list.add("Java研发工程师");
  list.add("web前端工程师");
  list.add("UI设计师");
  //创建Map集合
  Map<String, User> map = new HashMap<>();
  map.put("user", user);
  //转换json
  ObjectMapper mapper = new ObjectMapper();
  String userJson = mapper.writeValueAsString(user);
  String listJson = mapper.writeValueAsString(list);
  String mapJson = mapper.writeValueAsString(map);
  System.out.println(userJson);
  System.out.println(listJson);
  System.out.println(mapJson);
}
}
```

6.2.6 案例

案例需求：搜索框自动补全
    在输入框输入关键字，下拉框中异步显示与该关键字相关的数据

jsp代码：

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

```
<html>
<head>
<title>$Title$</title>
<style type="text/css">
          * {
              margin: 0px;
              padding: 0px;
          }

          #box {
              display: inline-block;
          }

          #search {
              width: 545px;
              height: 42px;
              border-color: #4E6EF2;
              border-top-left-radius: 10px;
              border-bottom-left-radius: 10px;
              float: left;
          }

          #btn {
              width: 108px;
              height: 42px;
              background-color: #4e6ef2;
              color: #fff;
              border: none;
              font-size: 17px;
              font-weight: 400;
              border-top-right-radius: 10px;
              border-bottom-right-radius: 10px;
              float: left;
          }

          #show {
              width: 545px;
              border: 1px solid #4e6ef2;
              position: relative;
              left: -55px;
              text-align: left;
          }
      </style>
      <script src="./js/jquery-3.3.1.js"></script>
      <script>
          $(function(){
              //绑定键盘弹起事件
              $("#search").bind('input porpertychange',function(){
                  //获取输入框的值
```

```
                var word = $(this).val();
                //判断不为空
                if(word != ""){
                    console.log(word)
                    //发送ajax请求
                    $.ajax({
                        url:"searchWord",
                        data:{"word":word},
                        type:"post",
                        dataType:"json",
                        success:function(obj){
                            var htmlStr = "";
                            for(var i = 0; i < obj.length; i++){
                                //console.log(obj[i].sname)
                                htmlStr += "<div>"+obj[i].sname+"</div>";
                            }
                            $("#show").html(htmlStr).show();
                        }
                    })
                }else{
                    $("#show").hide();
                }
            })
        })
    </script>
  </head>
  <body>
    <center>
        <img alt="" width="310" height="150"
src="${pageContext.request.contextPath }/img/baidu.png"><br/>
        <div id="box">
            <input id="search" type="text" name="search"/>
            <button id="btn">百度一下</button>
        </div>
        <div id="show">

        </div>
    </center>
  </body>
</html>
```

Servlet代码:

```
package com.ujiuye.web;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.ujiuye.pojo.Student;
import com.ujiuye.service.SearchService;

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.List;

@WebServlet("/searchWord")
public class SearchWordServlet extends HttpServlet {

@Override
protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
request.setCharacterEncoding("utf-8");
response.setContentType("text/html;charset=utf-8");
//获取word值
String word = request.getParameter("word");
//业务处理
SearchService ss = new SearchService();
List<Student> sList = ss.searchWord(word);
//判断
if(sList != null && sList.size() > 0){
  //将集合转换成json
  ObjectMapper mapper = new ObjectMapper();
  String strJson = mapper.writeValueAsString(sList);
  System.out.println(strJson);
  //响应页面
  response.getWriter().write(strJson);
}
}
}
```

Service代码:

```
package com.ujiuye.service;

import com.ujiuye.dao.SearchDao;
import com.ujiuye.pojo.Student;

import java.util.List;

public class SearchService {
public List<Student> searchWord(String word) {
SearchDao sd = new SearchDao();
return sd.searchWord(word);
}
}
```

Dao代码:

```
package com.ujiuye.dao;

import com.ujiuye.pojo.Student;
import com.ujiuye.utils.C3P0Utils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanListHandler;

import java.sql.SQLException;
import java.util.List;

public class SearchDao {
public List<Student> searchWord(String word) {
try {
  QueryRunner qr = new QueryRunner(C3P0Utils.getDataSource());
  String sql = "select * from student where sname like ?";
  List<Student> sList = qr.query(sql, new BeanListHandler<Student>(Student.class),
"%"+word+"%");
  return sList;
} catch (SQLException e) {
  e.printStackTrace();
}
return null;
}
}
```

实现效果：

6.3 XML

6.3.1 简介

> XML 指可扩展标记语言（EXtensible Markup Language），是一种标记语言，很类似 HTML，设计宗旨是传输数据，而非显示数据。标签没有被预定义。您需要自行定义标签。

6.3.2 语法

1. 文档声明：

```
<?xml version="1.0" encoding="UTF-8"?>
```

2. 自定义标签：

```
<根标签>
    <子标签 属性="值">内容</子标签>
    <子标签 属性="值">内容</子标签>
    ......
</根标签>
```

6.3.3 约束

1. DTD（文档类型定义）的作用是定义 XML 文档的合法构建模块。

```
<?xml version="1.0"?>
<!DOCTYPE note [
        <!ELEMENT note (to+,from,heading,body)>
        <!ELEMENT to      (#PCDATA)>
        <!ELEMENT from    (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body    (#PCDATA)>
        ]>
<note>
    <to>xx</to>
    <to></to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting!</body>

</note>
```

2. XML Schema 是基于 XML 的 DTD 替代者，XML Schema 描述 XML 文档的结构。

student.xsd文件：

```
<?xml version="1.0"?>
<xsd:schema xmlns="http://www.ujiuye.com"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ujiuye.com" elementFormDefault="qualified">
    <xsd:element name="students" type="studentsType"/>
    <xsd:complexType name="studentsType">
        <xsd:sequence>
            <xsd:element name="student" type="studentType" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="studentType">
```

```
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="age" type="ageType" />
            <xsd:element name="sex" type="sexType" />
        </xsd:sequence>
        <xsd:attribute name="number" type="numberType" use="required"/>
    </xsd:complexType>
    <xsd:simpleType name="sexType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="male"/>
            <xsd:enumeration value="female"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ageType">
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="0"/>
            <xsd:maxInclusive value="256"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="numberType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{4}"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

```
student.xml:
<?xml version="1.0" encoding="UTF-8" ?>
<!--
  1、编写根标签
  2、引入实例名称空间 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  3、引入名称空间 xsi:schemaLocation="http://www.ujiuye.cn/xml student.xsd"
  4、引入默认的名称空间
 -->

<students
  xmlns="http://www.ujiuye.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ujiuye.com student.xsd"
>
  <student number="1111">
    <name>tom</name>
    <age>111</age>
    <sex>male</sex>
  </student>
</students>
```