

课程回顾

- 1 1. 文件上传
- 2 1-1 文件上传的基本流程: jsp如何配置, 如果提交Servlet, form属性配置 servlet实现上传: 获取文件路径、验证文件格式、大小、重命名
- 3
- 4 2. jsp解析过程: work目录生成jsp对应java代码
- 5 2-1 配置isErrorPage errorPage错误页
- 6 2-2 jsp在web.xml进行配置
- 7 2-3 `<%@include page=""%>`

课程目标

1 ajax的作用 ===== 理解

2 ajax的XMLHttpRequest核心对象 ===== 理解

3.ajax实现===== 使用jquery完成

用户名唯一验证

三级联动

4 JSON数据的实现

课程实施

1 ajax

1-1 AJAX概念

ajax:异步的 JavaScript And XML技术。

XML格式不好解析, 所以主流的ajax配合使用数据格式: JSON格式

1-2 Ajax意义

提升用户体验度

- 1 1. 同步请求
- 2 缺点:
- 3
- 4 2. 异步请求: Ajax


用户名: 1.输入一个用户名

登录密码:

确认密码:

Email:

验证码:

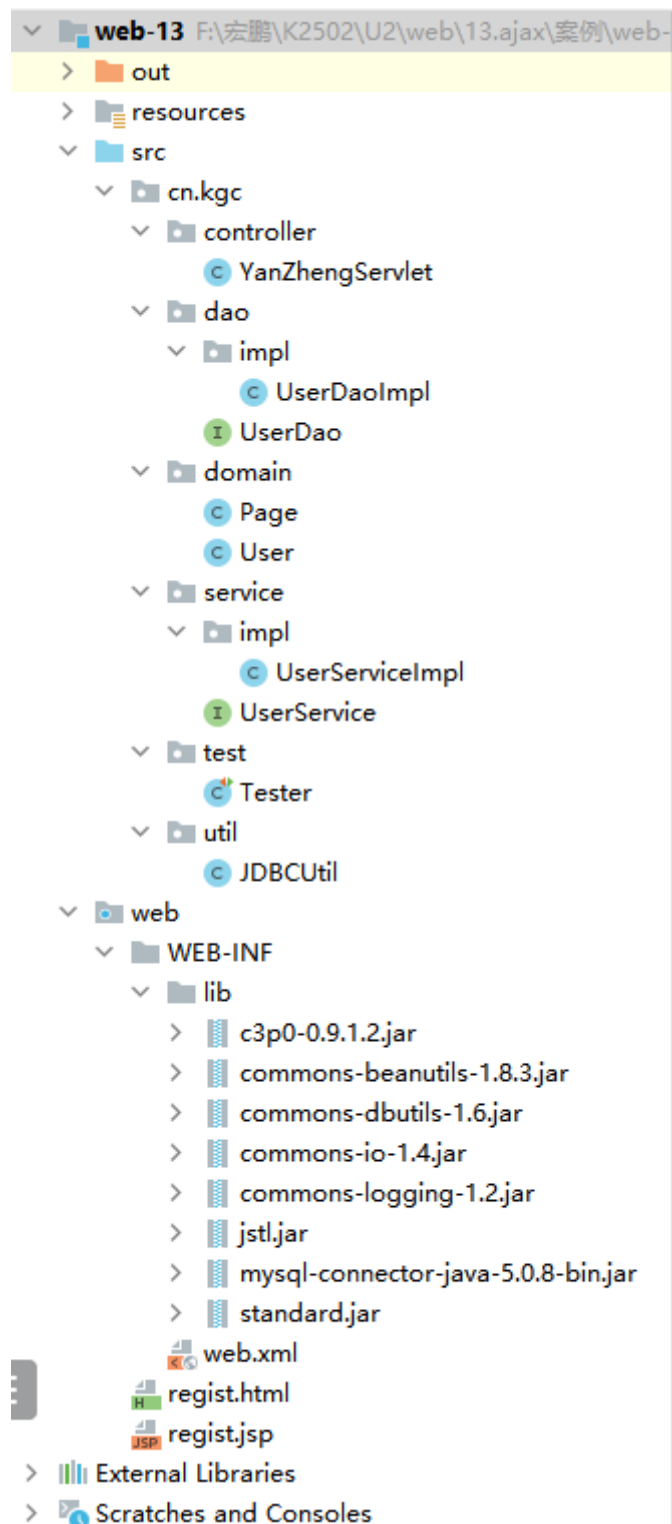
 [换一张](#)

2.用户光标离开用户名文本框的时候，向服务器发送一个请求：验证输入的用户名是否被占用

3.如果服务器响应的是占用，提示错误消息
如果服务器响应的没有占用，提示：用户名可以使用

× 用户名已被注册！

1-3 课堂案例的实现代码



domain\User

```
1 package cn.kgc.domain;
2
3 /**
4  * @Author: lc
5  * @Date: 2022/5/27
6  * @Description: cn.kgc.domain
7  * @Version: 1.0
8  */
9 public class User {
10     private String uid;
11     private String username;
12     private String password;
```

```

13
14     public String getUid() {
15         return uid;
16     }
17
18     public void setUid(String uid) {
19         this.uid = uid;
20     }
21
22     public String getUsername() {
23         return username;
24     }
25
26     public void setUsername(String username) {
27         this.username = username;
28     }
29
30     public String getPassword() {
31         return password;
32     }
33
34     public void setPassword(String password) {
35         this.password = password;
36     }
37
38     @Override
39     public String toString() {
40         final StringBuilder sb = new StringBuilder("User{");
41         sb.append("uid=").append(uid).append('\n');
42         sb.append(", username=").append(username).append('\n');
43         sb.append(", password=").append(password).append('\n');
44         sb.append('}');
45         return sb.toString();
46     }
47 }

```

daoのUserDaoImpl

```

1  package cn.kgc.dao.impl;
2
3  import cn.kgc.dao.UserDao;
4  import cn.kgc.domain.User;
5  import cn.kgc.util.JDBCUtil;
6  import org.apache.commons.dbutils.QueryRunner;
7  import org.apache.commons.dbutils.handlers.BeanHandler;
8
9  import java.sql.SQLException;
10
11  public class UserDaoImpl implements UserDao {
12      private QueryRunner qr=new QueryRunner(JDBCUtil.datasource);
13      @Override
14      public User selectBy(String username) {
15          try {
16              return qr.query("SELECT * FROM USER WHERE username=?",
17                  new BeanHandler<>(User.class),
18                  username);
19          } catch (SQLException e) {

```

```

20         throw new RuntimeException(e);
21     }
22 }
23 }

```

serviceのUserServiceImpl

```

1  package cn.kgc.service.impl;
2
3  import cn.kgc.dao.UserDao;
4  import cn.kgc.dao.impl.UserDaoImpl;
5  import cn.kgc.domain.User;
6  import cn.kgc.service.UserService;
7
8  public class UserServiceImpl implements UserService {
9      //1.创建依赖的dao层对象
10     UserDao dao=new UserDaoImpl();
11     /**
12      * 验证用户名是否被占用
13      * @param username 用户要验证的用户名
14      * @return true-用户名未占用 false-用户名已占用
15      */
16     @Override
17     public boolean validateUserName(String username) {
18         //2.调用dao层方法
19         User user = dao.selectBy(username);
20         return user==null;
21     }
22 }

```

同步请求实现用户名的验证

jspのregist.jsp

```

1  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2  <html>
3  <head>
4      <title>用户注册页面</title>
5  </head>
6  <script type="text/javascript">
7      /**
8       * 同步请求：只要没有看到老师写ajax，都是同步请求
9       */
10     function yanzheng(){
11         //js常用给服务器发请求：
12         //思考思路：是否需要给/YanZhengServlet带参数，就看Servlet依赖的service的方法有没有参数
13         //service的方法有参数，请求带参数
14
15         location.href="${pageContext.servletContext.contextPath}/YanZhengServlet?username="+document.getElementById("txtUserName").value;
16     }
17 </script>
18 <body>
19 <form>
20     <p>

```

```

20      用户名: <input id="txtUserName" type="text" name="user"
onblur="yanzheng()"><span style="color: red;">${msg}</span>
21      </p>
22      <p>
23          密码: <input type="text" name="pass">
24      </p>
25      <p>
26          确认密码: <input type="text" name="repass">
27      </p>
28      <p>
29          <input type="submit" value="注册">
30          <input type="reset" value="清空">
31      </p>
32  </form>
33  </body>
34  </html>

```

controllerのYanZhengServlet

```

1  package cn.kgc.controller; /**
2      * @Author: lc
3      * @Date: 2022/5/27
4      * @Description: ${PACKAGE_NAME}
5      * @Version: 1.0
6      */
7
8  import cn.kgc.service.UserService;
9  import cn.kgc.service.impl.UserServiceImpl;
10
11  import javax.servlet.*;
12  import javax.servlet.http.*;
13  import javax.servlet.annotation.*;
14  import java.io.IOException;
15
16  @WebServlet("/YanZhengServlet")
17  public class YanZhengServlet extends HttpServlet {
18      @Override
19      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
20          //1.取
21          String username = request.getParameter("username");
22          //2.调用
23          UserService userService=new UserServiceImpl();
24          boolean bool = userService.validateUserName(username);
25          //3.存
26          request.setAttribute("msg",bool?"用户名可用":"用户名已被占用，不可用");
27          //4.转
28
29          request.getRequestDispatcher("/regist.jsp").forward(request, response);
30      }
31
32      @Override
33      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
34          doGet(request, response);
35      }
36  }

```

异步请求实现用户名的验证

html的regist.html

```
1  <!DOCTYPE html>
2  <html lang="zh">
3  <head>
4      <meta charset="utf-8"/>
5      <title>ajax实现用户名唯一性验证</title>
6  <!-- 导入js的依赖类库文件-->
7      <script type="text/javascript" src="/js/jquery-3.6.0.min.js"></script>
8  </head>
9
10 <script type="text/javascript">
11     /**
12      * 异步请求: XMLHttpRequest对象, 代表使用AJAX
13      */
14     function yanzheng2(){//JQUERY实现的
15         //get(url,发给Servlet的数据, 响应成功后, 你想处理的页面代码
16         $.get("/YanzhengServlet", { username:
document.getElementById("txtUserName").value },
17             function(data){
18                 //设置到span进行显示
19                 //xmlHttpRequest.responseText封装到当前函数的形参里面去了
20                 document.getElementById("sperror").innerText=data;
21             });
22     }
23 </script>
24 <body>
25 <!--
26 JSP需求: EL+JSTL
27 html不支持EL+JSTL
28 -->
29 <form>
30     <p>
31         用户名: <input id="txtUserName" type="text" name="user"
onblur="yanzheng2()">
32         <span id="sperror" style="color: red;">? ? ? </span>
33     </p>
34     <p>
35         密码: <input type="text" name="pass">
36     </p>
37     <p>
38         确认密码: <input type="text" name="repass">
39     </p>
40     <p>
41         <input type="submit" value="注册">
42         <input type="reset" value="清空">
43     </p>
44 </form>
45 </body>
46 </html>
```

controllerのYanZhengServlet

```
1 package cn.kgc.controller; /**
2  * @Author: lc
3  * @Date: 2022/5/27
4  * @Description: ${PACKAGE_NAME}
5  * @Version: 1.0
6  */
7
8 import cn.kgc.service.UserService;
9 import cn.kgc.service.impl.UserServiceImpl;
10
11 import javax.servlet.*;
12 import javax.servlet.http.*;
13 import javax.servlet.annotation.*;
14 import java.io.IOException;
15
16 @WebServlet("/YanZhengServlet")
17 public class YanZhengServlet extends HttpServlet {
18     @Override
19     protected void doGet(HttpServletRequest request, HttpServletResponse
20 response) throws ServletException, IOException {
21         //1.取
22         String username = request.getParameter("username");
23         //2.调用
24         UserService userService=new UserServiceImpl();
25         boolean bool = userService.validateUserName(username);
26         //响应ajax请求，就要使用响应体响应
27         response.setContentType("text/html;charset=utf-8");
28         response.getWriter().print(bool?"用户名可用":"用户名已被占用，不可用");
29     }
30
31     @Override
32     protected void doPost(HttpServletRequest request, HttpServletResponse
33 response) throws ServletException, IOException {
34         doGet(request, response);
35     }
36 }
```

2 AJAX的核心对象

2-1 概念

AJAX的核心对象是XMLHttpRequest，在原生的JavaScript中，只有获取了XMLHttpRequest对象，才能完成异步请求的发送，以及跟踪服务器端的响应和响应结果。

2-2 XMLHttpRequest使用方式

```
1 原生态ajax实现步骤：
2 1. 获取Ajax的核心对象：XMLHttpRequest
3 (1)new XMLHttpRequest()，只有火狐和google浏览器支持，IE各个版本使用对象不尽相同
4 必须解决IE浏览器核心对象获取
5 var xmlhttp;
6
7 try
```



```

8      {
9          // Firefox, Opera 8.0+, Safari
10         xmlhttp=new XMLHttpRequest();
11     }
12     catch (e)
13     {
14
15         // Internet Explorer
16         try
17         {
18             xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
19         }
20         catch (e)
21         {
22
23             try
24             {
25                 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
26             }
27             catch (e)
28             {
29                 alert("您的浏览器不支持AJAX! ");
30                 return false;
31             }
32         }
33     }
34
35
36     2.发送请求: get、post请求
37     xmlhttp.open(请求提交的方式GET/POST,Servlet的url地址,是否使用异步发送true-异步
38     false-同步);
39     xmlhttp.send(null);//send(请求体), post请求有请求体 send("key=value"), get没有请
40     求体, send(null)
41
42     3.监听服务器的影响状态码: 局部使用dom解析完成html、jsp页面的刷新
43     xmlhttp.onreadystatechange=function()
44     {
45         //4表示请求已经处理完成, 且 status==200指服务器成功处理
46         if(xmlhttp.readyState==4&&xmlhttp.status==200)
47         {
48             //dom解析代码, 完成html、jsp最终的显示效果
49             //xmlhttp.responseText:服务器响应体
50             document.myForm.time.value=xmlhttp.responseText;
51         }
52     }

```

2-3 ajax实现的原生态代码

```

1     function yanzheng(){
2         //1.获取xmlHttpRequest对象
3         var xmlhttpRequest;//定义变量名
4         try{
5             //Firefox Chrome
6             xmlhttpRequest=new XMLHttpRequest();
7         }catch(e){
8             //IE7+
9             try{

```

```

10         xmlHttpRequest=new ActiveXObject("Msxml2.XMLHTTP")
11     } catch (e) {
12         //IE7-
13         try{
14             xmlHttpRequest=new ActiveXObject("Microsoft.XMLHTTP")
15         }catch (e) {
16             alert("您的浏览器不支持AJAX! ");
17             return false;
18         }
19     }
20 }
21 //2.发送异步请求
22 //建立与服务器的连接
23 xmlHttpRequest.open("get",
24     "/YanZhengServlet?
username="+document.getElementById("txtUserName").value,
25     true);//true-异步
26
27 xmlHttpRequest.send(null);//send() 发送，设置请求的请求体
28
29 //3.在span标签里面设置响应的结果
30 xmlHttpRequest.onreadystatechange=function(){
31     //4 表示请求已经完成，200 请求处理成功的
32     if(xmlHttpRequest.readyState==4 && xmlHttpRequest.status==200){
33         //设置到span进行显示
34
35         document.getElementById("sperror").innerText+xmlHttpRequest.responseText;
36     }
37 }

```

3 JSON数据

3-1 概念

- 1 JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。理解类似html文档格式、xml文档、doc文档、ppt、json、JavaScript
- 2 JSON是用字符串来表示Javascript对象，例如可以在Servlet中发送一个JSON格式的字符串给客户端Javascript，Javascript可以执行这个字符串，得到一个Javascript对象。
- 3 XML也可以用来数据交换，前面已经学习过在Servlet中发送XML给Javascript，然后Javascript再去解析XML。

3-2 json格式

```

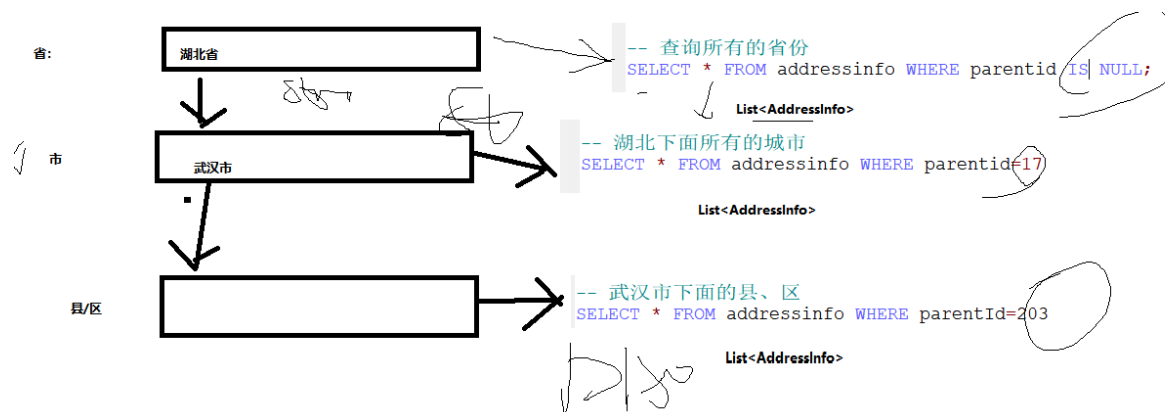
1 //单个json对象
2 var person = {"name":"zhangSan", "age":18, "sex":"male"};//java单个对象==> 单个
   json对象
3 alert(person.name + ", " + person.age + ", " + person.sex);
4
5
6 //json对象数组 list==>json数组
7 var people=[{"name":"zhangSan", "age":18, "sex":"male"}, {"name":"zhangSan",
   "age":18, "sex":"male"}, {"name":"zhangSan", "age":18, "sex":"male"}];

```

3-3 案例演示

```
1 <!DOCTYPE html>
2 <html lang="zh">
3 <head>
4     <meta charset="utf-8"/>
5     <title>JSON解析方式</title>
6 </head>
7 <script type="text/javascript">
8     //1. 定义json对象
9     var objJson={ "name": "张三", "age": 24, "sayHello": function()
10 {alert("Hello")}};
11 //2. 获取对象属性值，并调用方法
12 document.write(objJson.name+" , 年龄 "+objJson.age);
13 objJson.sayHello();
14 var arrJson=[{"name": "张三", "age": 24, "sayHello": function()
15 {alert("Hello")}},
16 {"name": "李四", "age": 26, "sayHello": function(){alert("你好")}}];
17
18 for(var i=0; i<arrJson.length; i++){
19     objJs=arrJson[i];
20     //2. 获取对象属性值，并调用方法
21     document.write(objJs.name+" , 年龄 "+objJs.age);
22     objJs.sayHello();
23 }
24 </script>
25 <body>
26 </body>
27 </html>
```

4 综合练习：三级联动



4-1 fastJson转换的结果

```
1 [
2   {
3     "addName": "北京",
4     "id": 1
5   },
6   {
7     "addName": "天津",
8     "id": 2
9   }
10 ]
```

```
9      },
10     {
11         "addName": "河北",
12         "id": 3
13     },
14     {
15         "addName": "山西",
16         "id": 4
17     },
18     {
19         "addName": "内蒙古",
20         "id": 5
21     },
22     {
23         "addName": "辽宁",
24         "id": 6
25     },
26     {
27         "addName": "吉林",
28         "id": 7
29     },
30     {
31         "addName": "黑龙江",
32         "id": 8
33     },
34     {
35         "addName": "上海",
36         "id": 9
37     },
38     {
39         "addName": "江苏",
40         "id": 10
41     },
42     {
43         "addName": "浙江",
44         "id": 11
45     },
46     {
47         "addName": "安徽",
48         "id": 12
49     },
50     {
51         "addName": "福建",
52         "id": 13
53     },
54     {
55         "addName": "江西",
56         "id": 14
57     },
58     {
59         "addName": "山东",
60         "id": 15
61     },
62     {
63         "addName": "河南",
64         "id": 16
65     },
66     {
```

```
67     "addName": "湖北",
68     "id": 17
69 },
70 {
71     "addName": "湖南",
72     "id": 18
73 },
74 {
75     "addName": "广东",
76     "id": 19
77 },
78 {
79     "addName": "广西",
80     "id": 20
81 },
82 {
83     "addName": "海南",
84     "id": 21
85 },
86 {
87     "addName": "重庆",
88     "id": 22
89 },
90 {
91     "addName": "四川",
92     "id": 23
93 },
94 {
95     "addName": "贵州",
96     "id": 24
97 },
98 {
99     "addName": "云南",
100    "id": 25
101 },
102 {
103     "addName": "西藏",
104     "id": 26
105 },
106 {
107     "addName": "陕西",
108     "id": 27
109 },
110 {
111     "addName": "甘肃",
112     "id": 28
113 },
114 {
115     "addName": "青海",
116     "id": 29
117 },
118 {
119     "addName": "宁夏",
120     "id": 30
121 },
122 {
123     "addName": "新疆",
124     "id": 31
```



```

32         $("#city").append($("#<option
value='"+objJson.id+"'>"+objJson.addName+"</option>"));
33     });
34     }, "json");
35 });
36 });
37 </script>
38 <body>
39 省: <select id="prov">
40     <option>===请选择===</option>
41 </select>
42 <br>
43 市: <select id="city">
44     <option>===请选择===</option>
45 </select>
46 <br>
47 县/区: <select id="region">
48     <option>===请选择===</option>
49 </select>
50 <br>
51 </body>
52 </html>

```

Servletの获取省市区

```

1  package cn.kgc.controller; /**
2      * @Author: lc
3      * @Date: 2022/5/27
4      * @Description: ${PACKAGE_NAME}
5      * @Version: 1.0
6      */
7
8  import cn.kgc.domain.AddressInfo;
9  import cn.kgc.service.impl.AddressInfoServiceImpl;
10 import com.alibaba.fastjson.JSON;
11
12 import javax.servlet.ServletException;
13 import javax.servlet.annotation.WebServlet;
14 import javax.servlet.http.HttpServlet;
15 import javax.servlet.http.HttpServletRequest;
16 import javax.servlet.http.HttpServletResponse;
17 import java.io.IOException;
18 import java.util.ArrayList;
19 import java.util.List;
20
21 @WebServlet("/GetProvinceServlet")
22 public class GetProvinceServlet extends HttpServlet {
23     @Override
24     protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
25         //取所有的省份
26         String parentId = request.getParameter("parentId");
27         List<AddressInfo> allProvince=new ArrayList<>();
28         if(parentId==null) {
29             allProvince = new AddressInfoServiceImpl().getAllProvince();
30         }else{

```

```

31         allProvince=new
AddressInfoServiceImpl().getAllCities(Integer.parseInt(parentId));
32     }
33     //关键步骤: java中为了迎合ajax数据处理的需求, 之间交换数据格式使用: json格式
34     //一般简化json的格式转换, 耗费无用工具, 使用第三方的插件: 阿里巴巴 fastjson
35     String jsonStr = JSON.toJSONString(allProvince, true); //true: json格式
    阅读性更好
36     System.out.println(jsonStr);
37     //响应体回送ajax进行处理
38     response.setContentType("text/html;charset=utf-8");
39     response.getWriter().print(jsonStr);
40 }
41
42 @Override
43 protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
44     doGet(request, response);
45 }
46 }

```

Serviceの获取省市区

```

1 package cn.kgc.service.impl;
2
3 import cn.kgc.dao.AddressInfoDao;
4 import cn.kgc.dao.impl.AddressInfoDaoImpl;
5 import cn.kgc.domain.AddressInfo;
6 import cn.kgc.service.AddressInfoService;
7
8 import java.util.List;
9
10 public class AddressInfoServiceImpl implements AddressInfoService {
11     private AddressInfoDao dao=new AddressInfoDaoImpl();
12     @Override
13     public List<AddressInfo> getAllProvince() {
14         return dao.selectBy(null);
15     }
16
17     @Override
18     public List<AddressInfo> getAllCities(Integer parentId) {
19         return dao.selectBy(parentId);
20     }
21 }

```

Dao的实现

```

1 package cn.kgc.dao.impl;
2
3 import cn.kgc.dao.AddressInfoDao;
4 import cn.kgc.domain.AddressInfo;
5 import cn.kgc.util.JDBCUtil;
6 import org.apache.commons.dbutils.QueryRunner;
7 import org.apache.commons.dbutils.handlers.BeanListHandler;
8
9 import java.sql.SQLException;
10 import java.util.ArrayList;

```



```

11 import java.util.List;
12 import java.util.Objects;
13
14 public class AddressInfoDaoImpl implements AddressInfoDao {
15     private QueryRunner qr=new QueryRunner(JDBCUtil.datasource);
16     @Override
17     public List<AddressInfo> selectBy(Integer parentId) {
18         StringBuilder sb=new StringBuilder();
19         sb.append("SELECT * FROM addressinfo WHERE 1=1");
20         //定义保存? 对应参数的集合
21         List params=new ArrayList();
22         if(Objects.isNull(parentId)){
23             sb.append(" and parentid IS NULL;");
24         }else{
25             sb.append(" and parentid =?;");
26             params.add(parentId);
27         }
28         try {
29             return qr.query(sb.toString(),
30                 new BeanListHandler<>(AddressInfo.class),
31                 params.toArray());
32         } catch (SQLException e) {
33             throw new RuntimeException(e);
34         }
35     }
36 }
37 }

```

4-3 get和原始ajax的类比

jQuery实现

```

function yanZheng2(){//JQUERY实现的
//get(url, 发给Servlet的数据, 响应成功后, 你想处理的页面代码)
$.get("/YanZhengServlet", { username: document.getElementById("txtUser") },
function(data){ data就是一个自定义的形参名称, 形参名字能不能改? 必须可以!
//设置到span进行显示 data=xmlHttpRequest.responseText, 类型是String
//xmlHttpRequest.responseText封装到当前函数的形参里面去了
document.getElementById("sperror").innerText=data;
});
$.get()不会执行最后的function()函数体, 要基于服务器是否响应的200

```

JavaScript的原生方式

```

21 //2. 发送异步请求
22 //建立与服务器的连接
23 xmlHttpRequest.open("get",
24     "/YanZhengServlet?
25     username="+document.getElementById("txtUserName").value,
26     true); //true-异步
27
28 xmlHttpRequest.send(null); //send() 发送, 设置请求的请求体
29
30 //3. 在span标签里面设置响应的结果
31 xmlHttpRequest.onreadystatechange=function(){
32     //4 表示请求已经完成, 200 请求处理成功的
33     if(xmlHttpRequest.readyState==4 && xmlHttpRequest.status==200
34         //设置到span进行显示
35         document.getElementById("sperror").innerText=xmlHttpRequest.responseText
36     }
37 }

```

课程总结

用法: jquery的getJson() get() post()

面试导向: 原生态 (核心对象是什么? XMLHttpRequest、open() send() onreadystatechange, responseText)

Ajax: 完成课堂案例

过滤器和监听器

四大域对象

休息时间

周六休息

下周：周二休息

端午节：周五周六周日