

## 一.前后端分离:

前端: Web, 移动端(iOS, Android), 微信公众号, 微信小程序, 百度小程序, 支付宝小程序, PC端软件

后台: Java, PHP, C#, Python

封装一个result类: (有的有返回值, 有的无返回值)

```
public class Result {  
    private int code;  
    private String msg;  
    private Object data;  
  
    public Result() {  
    }  
  
    //没有返回结果  
    public Result(int code, String msg) {  
        this.code = code;  
        this.msg = msg;  
    }  
  
    //有返回结果  
    public Result(int code, String msg, Object data) {  
        this.code = code;  
        this.msg = msg;  
        this.data = data;  
    }  
}
```

service业务逻辑层, 写判断参数是否为空, 状态码, 信息,

```

public class UserServiceImpl implements UserService {
    private UserDao userDao = new UserDaoImpl();
    @Override
    public Result login(String username, String password) {
        //判断参数是否为空
        if (StringUtils.isEmpty(username, password)) {
            return new Result( code: 1001, msg: "账号或密码为空");
        }
        //数据库查找后仍然为空,说明没有这个用户
        User user = userDao.selectByUsernameAndPassword(username, password);
        if (user == null) {
            return new Result( code: 1002, msg: "账号或密码有误!");
        }
        //成功
        return new Result( code: 200, msg: "登陆成功!", user);
    }
}

```

## Servlet中返回数据格式,

```

UserService userService = new UserServiceImpl();
Result result = userService.login(username, password);

//返回数据的格式, json占主流
response.setContentType("application/json;charset=utf-8");
PrintWriter writer = response.getWriter();
writer.write(JSON.toJSONString(result));

```

## 二. 前端向后台发起请求的方式:

1. a: 超链接

2. form: 表单

3. ajax: 异步javascript和xml, 是一种网页发起请求的常用方式(js的技术), 配合js可以实现网页局部刷新

## 阻止表单向系统提交的默认行为

```

<script src="/static/js/jquery.min.js"></script>
<script>
    $(function () {
        // 表单提交事件
        $("#login-form").submit(function () {
            console.log("发起表单请求");

            // 阻止系统的默认行为(bi)
            return false;

        });
    });
</script>

```

## jq版对ajax做了一部分简化

```

1 <script src="/static/js/jquery.min.js">
2 </script>
3
4 <script>
5     $(function () {
6         // 表单的提交时间
7         $("#login-form").submit(function () {
8             console.log("发起表单请求");
9             // 发起ajax请求(jq版)
10            $.ajax({
11                // 请求地址
12                url: "http://localhost:8080/user/login",
13                // 请求方式
14                type: "get",
15                // 请求参数
16                // 快速获取输入框的值
17                data: {
18                    username: $("#username").val(),
19                    password: $("#password").val()
20                },
21                // 请求成功的回调
22                success: function (resp) {
23                    // resp: 服务器端响应数据
24                    console.log(resp);
25                    if (resp.code === 200) {
26                        location.href = "list.html"

```

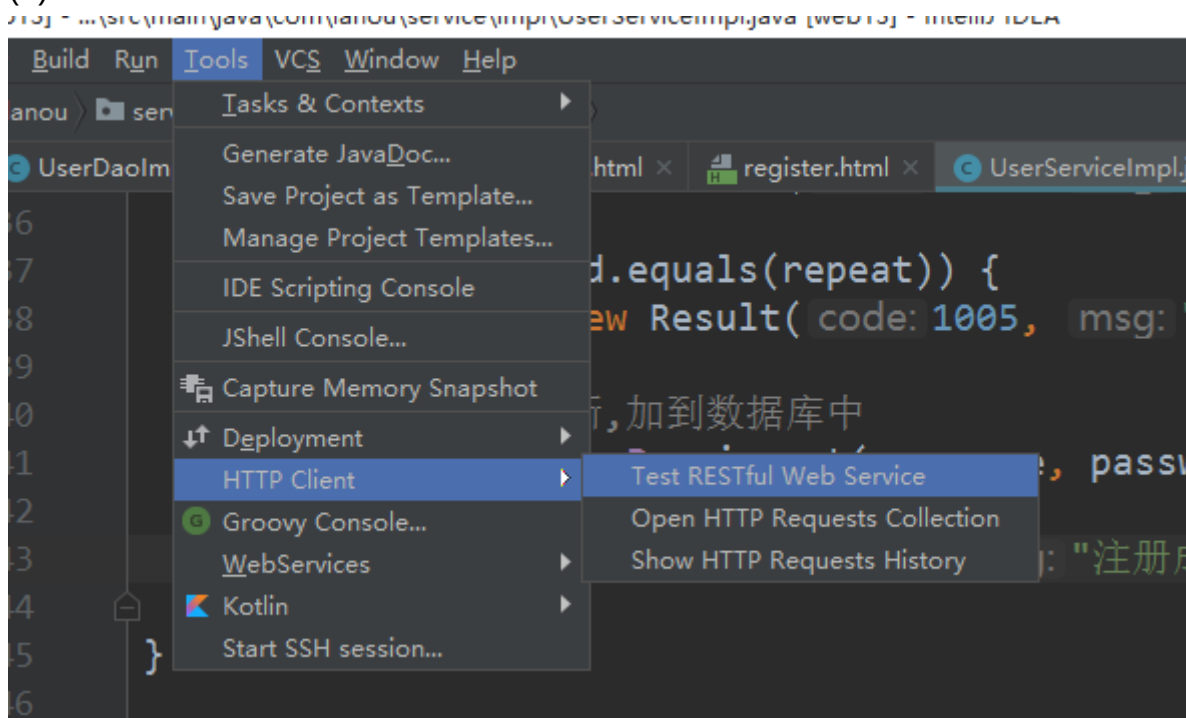
```

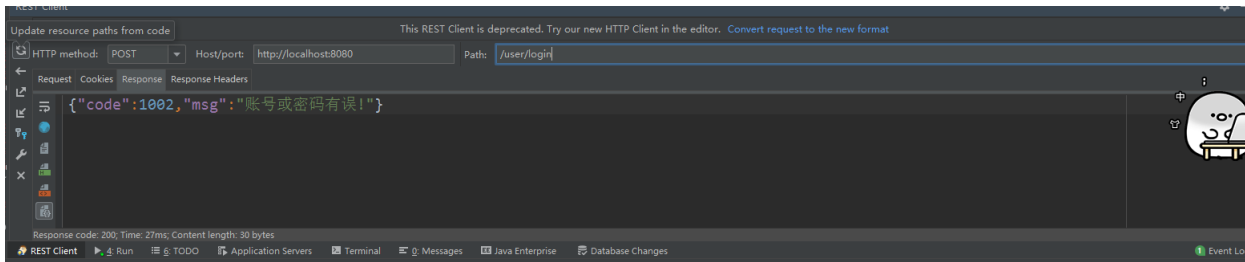
27 } else {
28     //显示弹框,失败
29     alert(resp.msg);
30 }
31 },
32 //请求失败的回调
33 error: function (resp) {
34     console.log(resp);
35 },
36 //请求参数的格式,默认:
37 //application/x-www-form-urlencoded
38 contentType: "application/x-www-form-urlencoded",
39 //返回数据的格式eg:json,xml,text
40 dataType: "json"
41 });
42 //阻止系统的默认行为(表单请求时,跳转页面)
43 return false;
44 });
45 });
46 </script>

```

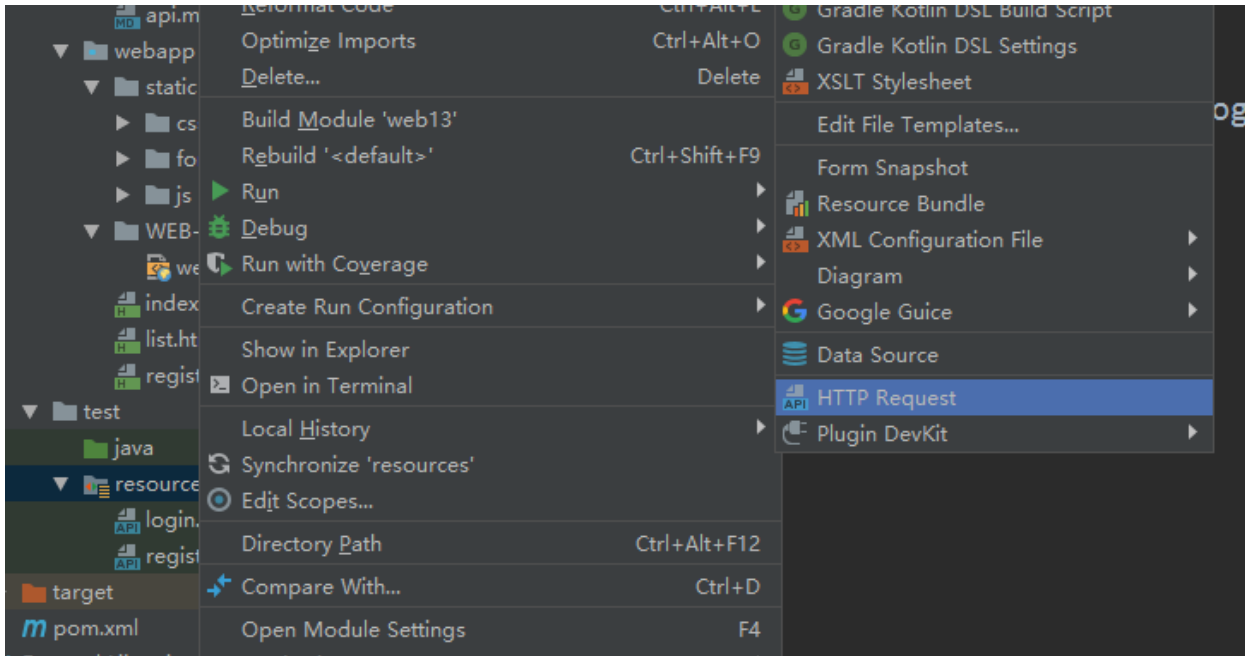
两种测试方法:

(1)

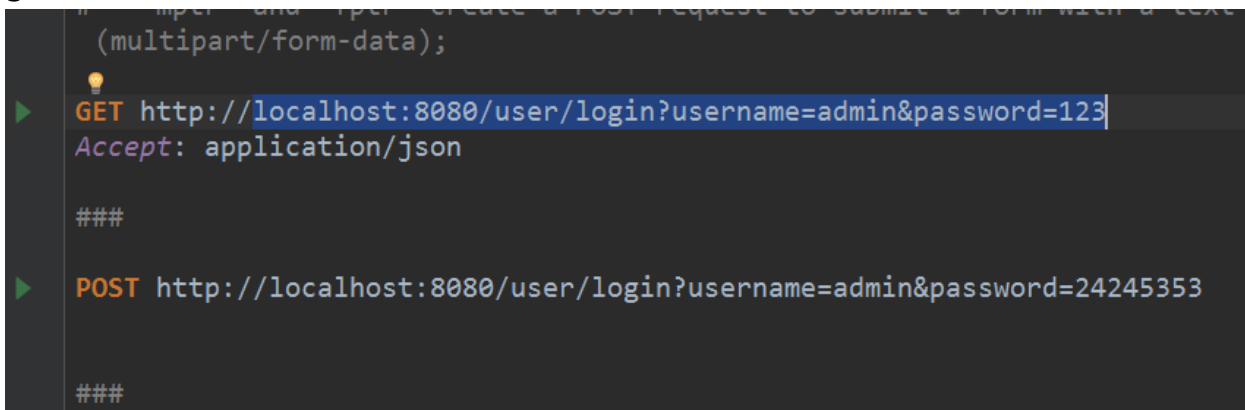




(2)



gtr



接口文档:新建file,后缀为.md

## 分页查询:

封装一个私有的pageBean,设置页数和条数,总条数和总页数无法设置,需要从Dao里写查找的总条数

的接口,/查询总条数 int selectCount();实现这个接口,注意返回是一个map(key,value)

dao层:

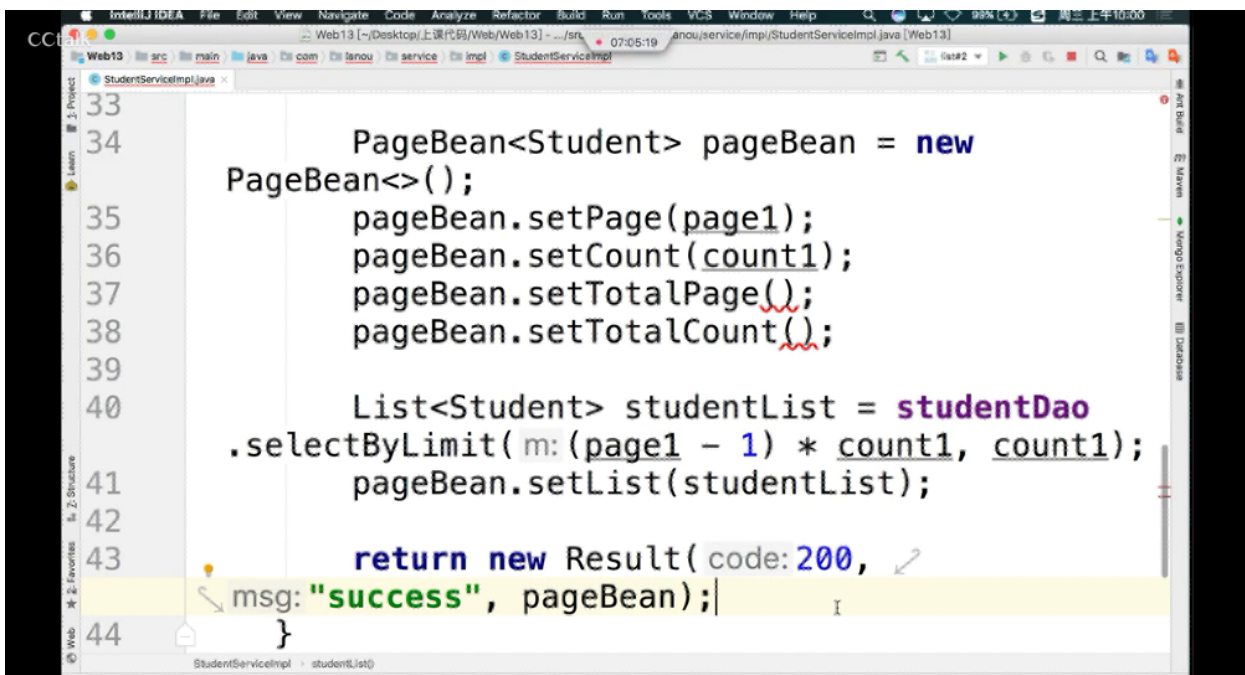
```

public class StudentDaoImpl implements StudentDao {
    private QueryRunner queryRunner = JDBCUtils.getQueryRunner();
    @Override
    public List<Student> selectAll(int m, int n) {
        List<Student> studentList=null;
        try {
            studentList = queryRunner.query( sql: "select * from student limit ?,?", new
BeanListHandler<Student>(Student.class), m, n);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return studentList;
    }

    @Override
    public int selectCount() {
        Map<String, Object> map=null;
        try {
            map = queryRunner.query( sql: "select count(*) count from student", new MapHandler());
            //count,别名,相当于map的key,获取用get.count=value.
        } catch (SQLException e) {
            e.printStackTrace();
        }
        //转成字符串,转数字
        return Integer.parseInt( s: map.get("count")+"" );
    }
}

```

service层:



```

33
34     PageBean<Student> pageBean = new
PageBean<>();
35     pageBean.setPage(page1);
36     pageBean.setCount(count1);
37     pageBean.setTotalPage();
38     pageBean.setTotalCount();
39
40     List<Student> studentList = studentDao
.selectByLimit(m: (page1 - 1) * count1, count1);
41     pageBean.setList(studentList);
42
43     return new Result( code: 200,
msg: "success", pageBean);
44 }

```

```

1 public class StudentServiceImpl implements StudentService {
2     private StudentDao studentDao = new StudentDaoImpl();
3
4     @Override
5     public Result studentList(String page, String count) {
6         //默认值
7         int page1 = 1, count1 = 3;
8         if (StringUtils.isEmpty(page)) {
9             page1 = Integer.parseInt(page);

```

```

10  }
11  if (StringUtils.isEmpty(count)) {
12      count1 = Integer.parseInt(count);
13  }
14
15  if (page1 < 0 || count1 <= 0) {
16      return new Result(1003, "参数有误");
17  }
18  PageBean<Student> pageBean=new PageBean<>();
19  pageBean.setCount(count1);
20  pageBean.setPage(page1);
21  // pageBean.setTotalCount();
22  // pageBean.setTotalPage();
23
24  int = studentDao.selecint totalCoutCount();
25  pageBean.setTotalCount(totalCount);
26
27  int totalPage = totalCount % count1 == 0 ? totalCount / count1 : totalC
  ount / count1 + 1;
28  pageBean.setTotalPage(totalPage);
29
30
31  List<Student> studentList = studentDao.selectAll((page1 - 1) * count1,
  count1);
32  pageBean.setList(studentList);
33
34  return new Result(200, "success", pageBean);
35
36  }

```

前端页面:

```

1  <h1>学生管理系统</h1>
2  <table border="1" width="500" cellspacing="0">
3
4  <thead>
5  <tr>
6  <th>id</th>

```

```
7  <th>姓名</th>
8  <th>性别</th>
9  <th>年龄</th>
10 </tr>
11 </thead>
12
13 <tbody id="list">
14
15 </tbody>
16 </table>
17
18 <input type="button" value="上一页" id="pre">
19 <input type="button" value="下一页" id="next">
20
21 //////////////////////////////////////
22 <script>
23 $(function () {
24   //时间戳转data
25   function timestampTOAge(timestamp) {
26     var date1 = new Date(timestamp);
27     var date2 = new Date();
28     return date2.getFullYear() - date1.getFullYear();
29   }
30
31   function network(page, count) {
32     $.ajax({
33       url: "http://localhost:8080/student/list",
34       type: "get",
35       data: {
36         page: page,
37         count: count
38       },
39       dataType: "json",
40       success: function (resp) {
41         //得到数据
42         console.log(resp);
43         if (resp.code === 200) {
44           $("#list").empty();
```



```
45  totalPage = resp.data.totalPage;
46  for (var i = 0; i < resp.data.list.length; i++) {
47    var student = resp.data.list[i];
48    var html = "<tr>";
49    html += "<td>" + student.id + "</td>";
50    html += "<td>" + student.name + "</td>";
51    html += "<td>" + student.gender + "</td>";
52    html += "<td>" + timestmpTOAge(student.birthday) + "</td>";
53    html += "</tr>";
54    $("#list").append(html);
55  }
56  } else {
57    alert(resp.msg);
58  }
59  }
60  })
61  }
62
63  var page = 1, count = 3, totalPage = 0;
64  network(page, count);
65  //将第一页的上一页,隐藏
66  $("#pre").hide();
67  方法1:/*
68  $("#next").click(function () {
69    if (page<totalpage){
70      page++;
71      network(page, count);
72    }
73
74  });
75  $("#pre").click(function () {
76    if (page>1){
77      page--;
78      network(page, count);
79    }
80
81  });
82  })
```

```

83  */
84
85  方法2:当有下一页时隐藏上一页
86  $("#next").click(function () {
87      page++;
88      network(page, count);
89      if (page === totalPage) {
90          $("#next").hide();
91      }
92      $("#pre").show();
93  });
94  $("#pre").click(function () {
95      page--;
96      network(page, count);
97      if (page === 1) {
98          $("#pre").hide();
99      }
100     $("#next").show();
101  });
102  })
103
104  </script>

```

登录验证码:

1.新建sevlet接口:

```

1  @WebServlet(name = "ImageServlet", urlPatterns = "/user/image")
2  public class ImageServlet extends HttpServlet {
3
4
5      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
6          //创建图片
7          int width = 100;
8          int height = 40;
9          //创建一个画布
10         BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
11
12         //获取画布的画笔

```

```
13 Graphics graphics = image.getGraphics();
14
15 //设置画笔颜色
16 Color color = new Color(122, 122, 0);
17 graphics.setColor(color);
18 //画长方形
19 graphics.fillRect(0, 0, width, height);
20
21 //设置画笔颜色
22 graphics.setColor(Color.blue);
23
24 //设置字体
25 Font font = new Font("斜体", Font.BOLD, 30);
26 graphics.setFont(font);
27
28 //将空字符串拼接上去
29 String codeString = "";
30 String randomString = "abcdefghijklmnopqrstuvwxyz123456789ABCDEFGHIJKLM
NOPQRSTUVWXYZ";
31 Random random = new Random();
32 for (int i = 0; i < 4; i++) {
33     int index = random.nextInt(randomString.length());
34     String temp = randomString.charAt(index) + "";
35     //保证每次显示都是一样的4个
36     codeString += temp;
37     //写字
38     graphics.drawString(temp, width / 4 * i, 30);
39
40 }
41
42 // 把验证码存入session
43 HttpSession session = request.getSession();
44 session.setAttribute("codeString", codeString);
45
46
47 //把创建的图片传个前端
48 ImageIO.write(image, "jpg", response.getOutputStream());
49
```

```
50 }  
51 }
```

2.修改登录界面的sevlet,获取服务器存到验证码,让他和code比较,相等,说明验证码正确,不相等,写状态码.

```
1  @WebServlet(name = "LoginServlet", urlPatterns = "/user/login")  
2  public class LoginServlet extends HttpServlet {  
3      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
4          request.setCharacterEncoding("utf-8");  
5          String username = request.getParameter("username");  
6          String password = request.getParameter("password");  
7          System.out.println("username = " + username);  
8          System.out.println("password = " + password);  
9  
10         String code = request.getParameter("code");  
11         System.out.println("code = " + code);  
12  
13         //获取服务存的验证码  
14         HttpSession session = request.getSession();  
15         String codeString = String.valueOf(session.getAttribute("codeString"));  
16  
17         UserService userService = new UserServiceImpl();  
18         Result result = userService.login(username, password, code, codeString);  
19  
20         //返回数据的格式, json占主流  
21         response.setContentType("application/json;charset=utf-8");  
22         PrintWriter writer = response.getWriter();  
23         writer.write(JSON.toJSONString(result));  
24  
25     }  
26  
27     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
28         doPost(request, response);  
29     }
```

3.修改service层的,忽略大小写的判断,判断两个是否相等,

```

1 public class UserServiceImpl implements UserService {
2     private UserDao userDao = new UserDaoImpl();
3
4     @Override
5     public Result login(String username, String password,String code,String
        codeString) {
6         //判断参数是否为空
7         if (StringUtils.isEmpty(username, password,code)) {
8             return new Result(1001, "账号或密码或验证码为空");
9         }
10
11         if (!code.equalsIgnoreCase(codeString)){
12             return new Result(1007, "验证码输入有误!");
13         }
14         //数据库查找后仍然为空,说明没有这个用户
15         User user = userDao.selectByUsernameAndPassword(username, password);
16         if (user == null) {
17             return new Result(1002, "账号或密码有误!");
18         }
19         //成功
20         return new Result(200, "登陆成功!", user);
21     }

```

#### 4.回到登录页面,写ajax.

```

1 <h1>登录</h1>
2
3 <form action="/user/login" id="login-form">
4     <input type="text" name="username" placeholder="账号:" id="username">
5     <input type="password" name="password" placeholder="密码:" id="password":
6     <input type="text" name="code" id="code" placeholder="验证码">
7     <input type="submit">
8 </form>
9 <a href="register.html">注册</a>
10
11
12 <!--//前端向后台发起请求的方式:-->
13 <!--//1.a:超链接-->
14 <!--//2.form:表单-->

```

```
15 <!--//3.ajax:异步javascript和xml,是一种网页发起请求的常用方式-->
16 <!--//(js的技术),配合js可以实现网页局部刷新-->
17 <script src="/static/js/jquery.min.js">
18 </script>
19
20 <script>
21 $(function () {
22     //表单的提交时间
23     $("#login-form").submit(function () {
24         console.log("发起表单请求");
25         //发起ajax请求(jq版)
26         $.ajax({
27             //请求地址
28             url: "http://localhost:8080/user/login",
29             //请求方式
30             type: "get",
31             //请求参数
32             //快速获取输入框的值
33             /*
34             data: {
35                 username: $("#username").val(),
36                 password: $("#password").val(),
37                 code:$("#code").val()
38             },
39             */
40             //获取表单id
41             data:$("#login-form").serialize(),
42
43             //请求成功的回调
44             success: function (resp) {
45                 //resp:服务器端响应数据
46                 console.log(resp);
47                 if (resp.code === 200) {
48                     location.href = "list.html"
49                 } else {
50                     //显示弹框,失败
51                     alert(resp.msg);
52                 }
53             }
54         });
55     });
56 }
```

```

53  },
54  //请求失败的回调
55  error: function (resp) {
56  console.log(resp);
57  },
58  //请求参数的格式,默认:
59  //application/x-www-form-urlencoded
60  contentType: "application/x-www-form-urlencoded",
61  //返回数据的格式eg:json,xml,text
62  dataType: "json"
63  });
64  //阻止系统的默认行为(表单请求时,跳转页面)
65  return false;
66  });
67
68  //刷新验证码(事件刷新验证码,使之前的不一样,不缓存,数学的随机数)
69  $("#img").click(function () {
70  //图片路径
71  var url="/user/image?a="+Math.random();
72  $("#img" ).attr("src",url);
73  })
74  });
75  </script>

```

项目:

**前后端不分离：**

- 1.后台：数据库(Mysql，驱动，数据库连接池，DBUtils，JDBCUtils，sql语句)，MVC模式(Controller,Service,Dao,Pojo),Servlet,Linstener,Filter,验证码
- 2.前端：JSP,JSTL,el,html,css,js

**前后端分离：**

- 1.后台：数据库(Mysql，驱动，数据库连接池，DBUtils，JDBCUtils，sql语句)，MVC模式(Controller,Service,Dao,Pojo),Servlet,Linstener,Filter，Fastjson,API，验证码
- 2.前端：html，css，js，jQuery，Ajax