登录功能后端接口

目录

目标任务		2
思路分析		2
1.后台接口开发思路分析		2
编码步骤		4
1.后端接口 Controller 页面		4
2.SXBUserServiceImpl	X	6
3.Dao		6
4.Model		8
源码		8
	(F) (O)	

目标任务

● 以接口的形式接收并判断前台数据,之后将结果返回给前台

思路分析

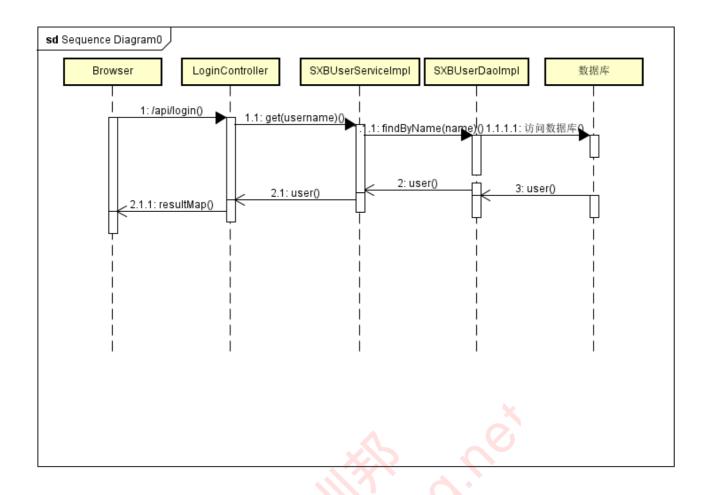
```
1.后台接口开发思路分析

If username && password 不为空
sxbUserService.get(username)

If 用户存在

If 密码正确
return {"code":"200","message":"成功","data":null};
else 密码错误
return {"code":"500201","message":"用户名或密码错误","data":null};

else 用户不存在
return {"code":"500201","message":"用户名或密码错误","data":null};
用户名或密码为空
return {"code":"500","message":"出错","data":null};;
```



url:/api/login

method: post

逻辑:根据 user 查找相对应 username 和 password,如果前后 username 和 password 相同,则跳转到课

程页面

参数:

参数名	类型	是否必须	说明
username	String	Υ	用户名
password	String	Υ	密码

返回值:

参数名	类型	是否必须	说明
code	Int	Υ	200 为成功
			500***为失败

message	String	Υ	成功或失败的消息内容
data	object	N	

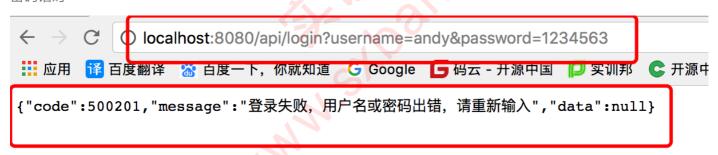
编码步骤

1. 运行效果截图

成功时



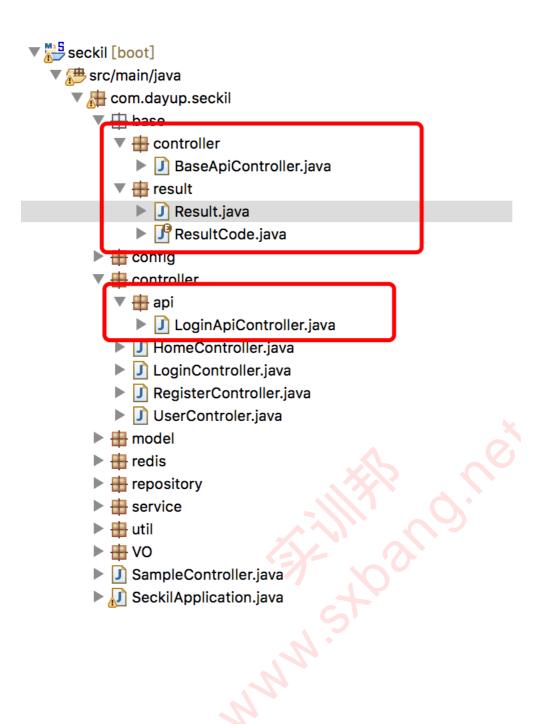
密码错时



用户名或密码为空



2. 文件目录结构



------ 正题分隔线 ------

3. BaseApiController.java

- 新建 BaseApiController 类,定义所有 api 接口的 URL 前缀,并开放跨域访问权限
- @CrossOrigin 是用来处理跨域请求让你能访问不同域的文件的注解, controller 要以接口形式为前端调用,
 必须继承 BaseApiController

4. Result.java

```
🚺 Result.java 💢
    package com.dayup.seckil.base.result;
  3 public class Result<T> {
  4
  5
         private Integer code;
  6
  7
         private String message;
  8
  9
         private T data;
 10
         public static <T> Result<T> failure() {
 11⊜
             Result<T> result = new Result<T>();
 12
 13
             result.setResultCode(ResultCode.FAIL);
 14
             return result;
 15
         }
 16
         public static <T> Result<T> failure(T data) {
 17⊜
 18
             Result<T> result = new Result<T>();
 19
             result.setResultCode(ResultCode.FAIL);
 20
             result.setData(data);
 21
             return result;
 22
         }
 23
 240
         public static <T> Result<T> failure(ResultCode resultCode) {
 25
             Result<T> result = new Result<T>();
             result.setResultCode(resultCode);
 26
 27
             return result;
 28
         }
 29
 30⊜
         public static <T> Result<T> failure(ResultCode resultCode, T data) {
 31
             Result<T> result = new Result<T>();
 32
             result.setResultCode(resultCode);
 33
             result.setData(data);
 34
             return result;
 35
         }
 36
 37⊜
         public static <T> Result<T> success() {
 38
             Result<T> result = new Result<T>();
 39
             result.setResultCode(ResultCode.SUCCESS);
 40
             return result;
         }
 41
 42
 43
         public static <T> Result<T> success(T data) {
 44
             Result<T> result = new Result<T>();
 45
             result.setResultCode(ResultCode.SUCCESS);
 46
             result.setData(data);
             return result;
 47
```

```
49
50⊜
        public void setResultCode(ResultCode resultCode) {
            this.code = resultCode.getCode();
51
            this.message = resultCode.getMessage();
52
53
        }
54
55⊜
        public Integer getCode() {
56
            return code;
57
        }
58
        public void setCode(Integer code) {
59⊜
            this.code = code;
60
        }
61
62
        public String getMessage() {
63⊜
64
            return message;
65
        }
66
67⊜
        public void setMessage(String message) {
            this.message = message;
68
69
70
71⊜
        public T getData() {
72
            return data;
73
        }
74
        public void setData(T data) {
75⊜
            this.data = data;
76
        }
77
78
79
   }
80
```

- 静态方法可以帮助我们快速为 javaBean 赋值,得到我们想要的格式结果。
- 这个 javaBean 转成 json 将会是{"code": 500201", "message": "用户名或密码错误", "data": ...}
- 5. ResultCode.java

```
LoginApiController.java
                         BaseApiController.java
                                                  🚺 ResultCode.java 🔀
    package com.dayup.seckil.base.result;
  2
 3
    public enum ResultCode {
 4
  5
        // 全局成功
  6
        SUCCESS(200, "成功"),
  7
        // 全局失败
 8
        FAIL(500, "失败"),
 9
10<sub>0</sub>
11
         * 跟用户有关的错误
12
         * USER_ 开头
13
         * 错误码 500200 - 500299
         */
14
15
        USER_LOGIN_ERROR(500201, "登录失败, 用户名或密码出错, 请重新输入。"),
16
        USER_HAS_EXISTED(500202, "用户已存在,请试试其他用户名。"),
17
        USER_NOT_LOGIN(500203, "用户未登录或登录已失效, 请重新登录。");
18
19
        private Integer code;
 20
21
        private String message;
22
23 🖨
        ResultCode(Integer code, String message) {
24
            this.code = code;
25
            this.message = message;
26
        }
27
280
        public Integer getCode() {
29
            return code;
 30
        }
31
32⊜
        public void setCode(Integer code) {
33
            this.code = code;
 34
        }
35
36⊜
        public String getMessage() {
37
            return message;
 38
        }
39
        public void setMessage(String message) {
40⊜
41
            this.message = message;
42
        }
43
 44
    }
```

- 使用枚举类型定义静态错误信息,方便修改,查找,添加。
- 我们约定,200为成功,500为失败,因为失败的情况和业务较多,我们用500开头,如5002**表示用户相 关业务出错等。

2.LoginApiController

```
🔳 sxbang - seckil/src/main/java/com/dayup/seckil/controller/api/LoginApiController.java - Spring Tool Suite
          Quick Access
   Project Explo 🗀 🗓 LoginApiController.java 🕱
                                                                                                                                                                                   - -
                          E 🕏 👂 🔻
                                             package com.dayup.seckil.controller.api;
                                                                                                                                                                                          ▣
0
                                             ⊕ import javax.servlet.http.Cookie;[
    gitee

seckil [boot]
                                           23
24 @RestController
                                                                                                                                                                                          ▼ # src/main/java
▼ # com.dayup.seckil
▼ # base
▼ # controller
                                                                                                                                                                                          №
                                              public class LoginApiControlle extends BaseApiController
                                                  private static Logger log = LoggerFactory.getLogger(LoginApiController.class);
               A B B □ Ju
             ▼ # result
                                                  public UserService userService;
                                                 @ResponseBody
                                                 config
            # controlle
            ▼ 🖶 api
                                                                                                                                                                                          ***
                                                        bindingResult.hasErrors()){
return Result.failure();
// 500, "error"
           ► J<sup>S</sup> HomeController.java

► J<sup>S</sup> LoginController.java

► J<sup>S</sup> RegisterController.java
             UserVO dbUser = userService.getUser(user.getUsername());
                                                         ► # repository
                                                                       sword().equals(MD5Util.inputToDb(user.getPassword(), dbUser.getDbflag()))){
          ▶ ∰ service
          Cookie cookie = new Cookie("token", token);
cookie.setMaxAge(3600);
cookie.setPath("/");
response.addCookie(cookie);
      ▶ # src/main/resources
       ▶ Æ src/test/iava
       ➤ Mayen Dependencies
                                                          return Result.success(); // Result.success(); 200, "success"
      ▶ 🥞 src
        (A) target
                                                            return Result. failure(ResultCode. USER_LOGIN_ERROR);
         mvnw.cmc
    > seckill-back [boot] [seckill-back master
    Servers SO
     sxb
                                                                                                            Writable
                                                                                                                              Smart Insert 38 : 56
```

- 复制 LoginController 到 api 包中命名为 LoginApiController 并继承 BaseApiController 改写 login 方法。
- 返回值为 Result < Object >
- 调用 Result 中封装的静态成功与失败方法去返回。
- Controller 上的@Controller 改为@RestController, @RestController 相关于@Controller 和
 @ResponseBody的合并注解,如果类注解了@RestController,下面的方法就不用每个都写@ReponseBody
 了。

回马枪总结

- 1. 使用 Result 返回固定格式结果
- 2. 使用注解@CrossOrigin:处理跨域请求的注解
- 3. 使用注解@RestController:返回 json 数据