**项目配置说明**

[1、 数据库设置 1](#_Toc421369427)

[2、 访问路径匹配 1](#_Toc421369428)

[3、 访问权限配置 1](#_Toc421369429)

[4、 前端页面与后端的传参方式 2](#_Toc421369430)

[5、 依赖注入配置 3](#_Toc421369431)

[6、 MyBatis传参方式 4](#_Toc421369432)

[7、 开启事务功能 5](#_Toc421369433)

1. **数据库设置**

在resources/jdbc.properties文件中配置。

如：

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/3dprint3

username=root

password=123456

1. **访问路径匹配**

使用以下标签来配置Controller的路径为/view/admin。

@RequestMapping("/view/admin")

使用以下标签来配置Controller中某个方法的路径为/mainpage

@RequestMapping(value = "/mainpage")

更详细的路径匹配内容可以参阅：[http://docs.spring.io/spring/docs/4.1.0.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/#mvc-ann-requestmapping](http://docs.spring.io/spring/docs/4.1.0.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/%23mvc-ann-requestmapping)

1. **访问权限配置**

在resources/spring-security.xml文件中配置。

以下代码就配置了/view/login路径所有人都可以访问，/view/user/\*\*路径只有权限为USER的用户能够访问。

同时，还配置了如果访问了没有权限的页面会重定向到/view/accessdenied路径。登录成功、登录失败、注销成功需要重定向的页面。

如某个用户访问了/view/admin/mainpage，如果该用户是MANAGER就能够正常访问，否则会重定向到/view/accessdenied。

<http auto-config=*"true"* use-expressions=*"true"*>

<intercept-url pattern=*"/view/login"* access=*"permitAll"*></intercept-url>

<intercept-url pattern=*"/view/common/\*\*"* access=*"permitAll"*></intercept-url>

<intercept-url pattern=*"/view/user/\*\*"* access=*"hasRole('ROLE\_USER')"*></intercept-url>

<intercept-url pattern=*"/view/admin/\*\*"* access=*"hasRole('ROLE\_ADMIN')"*></intercept-url>

<intercept-url pattern=*"/view/manager/\*\*"* access=*"hasRole('ROLE\_MANAGER')"*></intercept-url>

<access-denied-handler error-page=*"/view/accessdenied"* />

<form-login login-page=*"/view/login"* default-target-url=*"/view/common/index"*

authentication-failure-url=*"/view/loginfailed"*></form-login>

<logout logout-success-url=*"/view/logout"*></logout>

</http>

1. **前端页面与后端的传参方式**

使用JSON.stringify()方法来把一个Object转成JSON字符串，传到后台会被解析成Map。注意不要忘了设置contentType。

$.ajax({

url : '../../admin/addPlainUser',

type : 'post',

dataType : 'json',

contentType : 'application/json',

data : JSON.stringify({

name : 'Mike',

password : '123456'

}),

success : **function**(res) {

**if** (res && res.success) {

alert('新增用户成功！');

$('#usernameText').val('');

$('#passwordText').val('');

} **else** **if** (res) {

alert(res.message);

} **else** {

alert('新增用户失败！');

}

},

error : **function**(err) {

alert('未知错误');

}

});

后端接受参数。不要忘记设置consumes。只需要使用param.get(“”)方法就能够取出参数了。

@RequestMapping(value = "/addPlainUser", method = RequestMethod.***POST***, consumes = "application/json")

@ResponseBody

**public** Map addPlainUser(@RequestBody Map param) {

String name = (String) param.get("name");

String password = (String) param.get("password");

**return** initResult(**true**, userService.addPlainUser(param));

}

从后端返回结果到前端。如果需要返回的是一个JSON对象，如上面两段代码所示，ajax请求的时候不要忘了dataType的设置。后台也需要在方法前面加上@ResponseBody标签，返回参数也必须是Map。

在前台有个接受回调的函数success : **function**(res)，其中的res就对应着后台的传回的map。

1. **依赖注入配置**

使用标签来注入依赖。

**以下代码能够为AdminController注入一个UserService对象。**

在Controller类前面需要加上@Controller标签。

在其成员UserService的前面需要加上@Resource(name = "userServiceImpl")来说明这个接口的具体实现的名字。

需要定义UserService接口，和UserServiceImpl类。@Service("userServiceImpl")来命名这个具体类。

@Controller

**public** **class** AdminController **extends** BaseController {

@Resource(name = "userServiceImpl")

UserService userService;

}

**public** **interface** UserService {

}

@Service("userServiceImpl")

**public** **class** UserServiceImpl **implements** UserService {

}

**以下代码能够为UserServiceImpl注入一个UserMapper对象。**

注意到这里只有UserMapper接口，没有具体实现。具体的实现不是在java文件中，而是放在了xml文件中。

这是MyBatis的一种实现方式，如果接口文件名叫UserMapper.java，那么具体实现的文件名必须为UserMapper.xml。

@Service("userServiceImpl")

**public** **class** UserServiceImpl **implements** UserService {

@Resource(name = "userMapper")

UserMapper userMapper;

}

**public** **interface** UserMapper {

**public** Map findUserByName(Map map);

}

以下就是UserMapper.xml的内容。<mapper namespace=*"mapper.UserMapper"* >指明了它实现的是mapper.UserMapper接口。

支持的标签有<select><insert><update>等。<select id=*"findUserByName"* resultType=*"map"*>指明了它实现的是findUserByName方法，返回类型是Map。

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

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace=*"mapper.UserMapper"* >

<select id=*"findUserByName"* resultType=*"map"*>

select \* from user

where name=#{name}

<if test=*"status!=null"*>and status=#{status}</if>

</select>

</mapper>

1. **MyBatis传参方式**

MyBatis支持多种传入参数类型，如：int、String、Map、用户自定义的对象，这些都可以。在这里我只介绍一种方式，就是使用Map作为入参。

以下代码就把一个Map作为入参传入。在xml文件中用#{name}来获取传入的名字”Mike”。

@Service("userServiceImpl")

**public** **class** UserServiceImpl **implements** UserService {

@Resource(name = "userMapper")

UserMapper userMapper;

**public** Map queryUser() {

Map param = **new** HashMap();

param.put("name", "Mike");

Map user = userMapper.findUserByName(param);

**return** user;

}

}

返回类型有Map、List、Integer等。

一般来说<select>的返回类型写为map：<select id=*"findUserByName"* resultType=*"map"*>。

尽管xml文件中是写为map，对应的接口里面写Map和List都是可以的：**public** Map findUserByName(Map map)或者**public** List findUserByName(Map map)。

<insert><update>这两种标签不需要在xml中写明返回类型，它们返回的都是受影响行数，类型都为Integer。接口要这样：**public** Integer insertOneUser(Map map)。

附带说一句，如果需要insert一行，而且要返回这一行的id（因为id一般是自增的，不需要程序员指定），那么怎么实现？

在xml中这样写：

<select id=*"insertOneUser"* useGeneratedKeys=*"true"* keyProperty=*"user\_id"*>。

在Java中这样写：

Integer rows = userMapper.insertOneUser(param);

Integer userId = param.get(*"user\_id"*);

1. **开启事务功能**

所谓的开启事务功能，就是说，用户提交了一项请求，服务器在处理的时候发生了异常，那么需要把这次请求作废，不修改数据库。

只需要加上@Transactional标签，加上**throws** Exception抛出异常就可以了。

看下面的代码，如果其中if条件判断为真，那么抛出了异常，事务就不会提交。如果if条件为假，那么运行到**int** a = 1/0代码的时候系统会抛出一个/by zero的异常，事务同样不会提交。

@Transactional(propagation=Propagation.***REQUIRED***, rollbackFor=Exception.**class**)

**public** Boolean addPlainUser(Map param) **throws** Exception {

Integer row = userMapper.addPlainUser(param);

**if**(row == **null** || row != 1 || param.get("user\_id")==**null**){

**throw new** Exception("添加用户失败，该用户名已被占用。");

}

**int** a = 1/0;

**return true;**

}