# 项目1：springboot+mybatis＋SpringSecurity 实现用户角色数据库管理

---zhb

<http://blog.csdn.net/u012373815/article/details/54632176>

以及 <https://github.com/527515025/springBoot>

数据库设置：

DROP TABLE IF EXISTS `SYS\_USER`;

CREATE TABLE `SYS\_USER` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `SYS\_ROLE`;

CREATE TABLE `SYS\_ROLE` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `SYS\_ROLE\_USER`;

CREATE TABLE `SYS\_ROLE\_USER` (

`SYS\_USER\_ID` int(11) NOT NULL,

`ROLES\_ID` int(11) NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

insert into SYS\_USER (id,username, password) values (1,'admin', 'admin');

insert into SYS\_USER (id,username, password) values (2,'abel', 'abel');

insert into SYS\_ROLE(id,name) values(1,'ROLE\_ADMIN');

insert into SYS\_ROLE(id,name) values(2,'ROLE\_USER');

insert into SYS\_ROLE\_USER(SYS\_USER\_ID,ROLES\_ID) values(1,1);

insert into SYS\_ROLE\_USER(SYS\_USER\_ID,ROLES\_ID) values(2,2);

遇到问题点： java.sql.SQLException: The server time zone value...

Caused by:

com.mysql.cj.core.exceptions.InvalidConnectionAttributeException:

解决方法：使用的数据库是MySQL，驱动是6.0.3，这是由于数据库和系统时区差异所造成的，在jdbc连接的url后面加上serverTimezone=GMT即可解决问题，如果需要使用gmt+8时区，需要写成GMT%2B8，否则会被解析为空。再一个解决办法就是使用低版本的MySQL jdbc驱动，5.1.28不会存在时区的问题。

第二个就是要屏蔽掉 user Details Service验证 如下

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

/\* auth.userDetailsService(customUserService()).passwordEncoder(new PasswordEncoder(){

@Override

public String encode(CharSequence rawPassword) {

return MD5Util.encode((String)rawPassword);

}

@Override

public boolean matches(CharSequence rawPassword, String encodedPassword) {

return encodedPassword.equals(MD5Util.encode((String)rawPassword));

}}); //user Details Service验证

\*/

auth.userDetailsService(customUserService());

}

第三就是登陆账号访问的时候会出现 加载 bookstrap.min.css源码 的页面，这时候只需要后退，刷新，重新跳转才行

启动项目 在浏览器输入 http://localhost:8080/ 出来的界面，没有加载上去css样式， 然后输入账号密码，跳转不到主页，而是出来bookstrap.min.css源码，很奇怪 然后点击网页后退，重又跳转到了登陆首页，这时css样式才出来，，，，，，再输账号密码，一切正常

解决上面展示的问题的方法为：

加一段过滤的代码：

/\*\* https://blog.csdn.net/bryBLOG/article/details/77529619

\* 配置.忽略的静态文件，不加的话，登录之前页面的css,js不能正常使用，得登录之后才能正常.

\*/

@Override

**public** **void** configure(WebSecurity web) **throws** Exception {

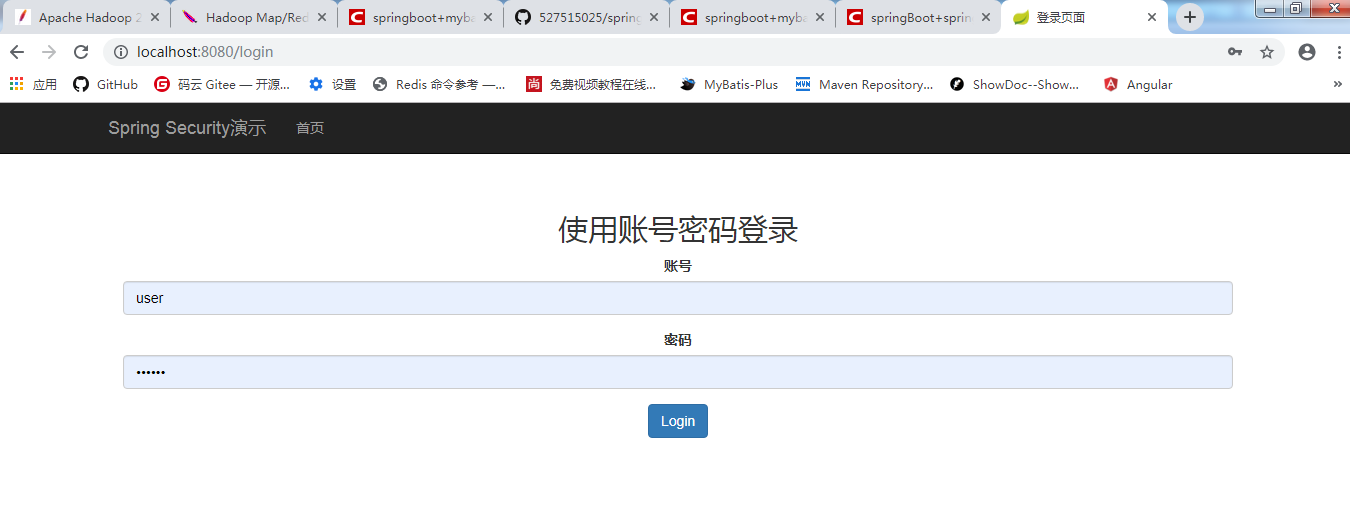
// 忽略URL

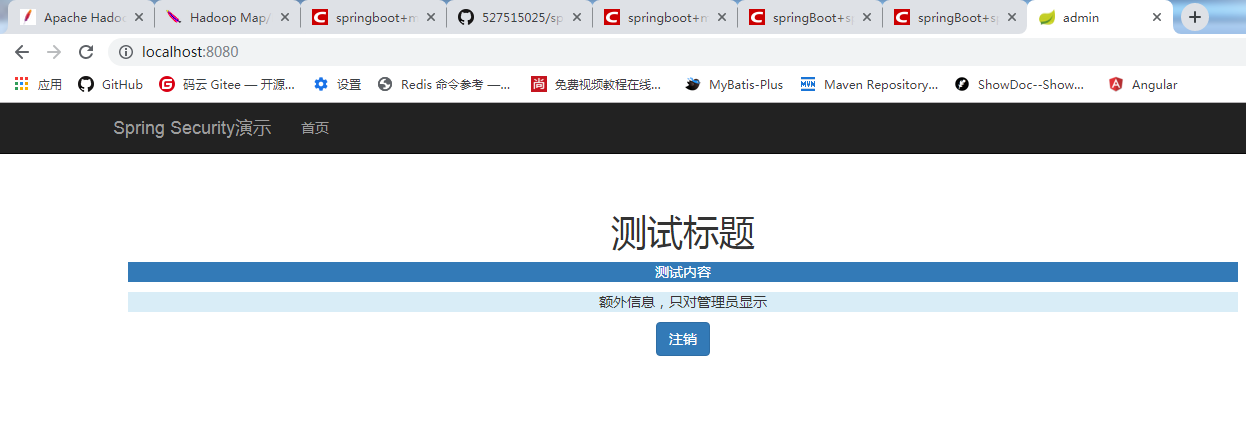
web.ignoring().antMatchers("/\*\*/\*.js", "/lang/\*.json", "/\*\*/\*.css", "/\*\*/\*.js", "/\*\*/\*.map", "/\*\*/\*.html",

"/\*\*/\*.png");

}

下图才是CSS真正加载成功的登陆页面展示





继续延伸的话 https://blog.csdn.net/u012373815/article/details/54633046

# 项目2：springBoot+springSecurity 数据库动态管理用户、角色、权限

spring security的简单原理：

使用众多的拦截器对url拦截，以此来管理权限。但是这么多拦截器，笔者不可能对其一一来讲，主要讲里面核心流程的两个。

首先，权限管理离不开登陆验证的，所以登陆验证拦截器AuthenticationProcessingFilter要讲；

还有就是对访问的资源管理吧，所以资源管理拦截器AbstractSecurityInterceptor要讲；

但拦截器里面的实现需要一些组件来实现，所以就有了AuthenticationManager、accessDecisionManager等组件来支撑。

现在先大概过一遍整个流程，用户登陆，会被AuthenticationProcessingFilter拦截，调用AuthenticationManager的实现，而且AuthenticationManager会调用ProviderManager来获取用户验证信息（不同的Provider调用的服务不同，因为这些信息可以是在数据库上，可以是在LDAP服务器上，可以是xml配置文件上等），如果验证通过后会将用户的权限信息封装一个User放到spring的全局缓存SecurityContextHolder中，以备后面访问资源时使用。

访问资源（即授权管理），访问url时，会通过AbstractSecurityInterceptor拦截器拦截，其中会调用FilterInvocationSecurityMetadataSource的方法来获取被拦截url所需的全部权限，在调用授权管理器AccessDecisionManager，这个授权管理器会通过spring的全局缓存SecurityContextHolder获取用户的权限信息，还会获取被拦截的url和被拦截url所需的全部权限，然后根据所配的策略（有：一票决定，一票否定，少数服从多数等），如果权限足够，则返回，权限不够则报错并调用权限不足页面。

---------------------

作者：双斜杠少年

来源：CSDN

原文：https://blog.csdn.net/u012373815/article/details/54633046

版权声明：本文为博主原创文章，转载请附上博文链接！

延伸记录：

<https://blog.csdn.net/u012373815/article/details/54633046>

数据库补充：

CREATE TABLE `sys\_permission\_role` ( `id` int(11) NOT NULL AUTO\_INCREMENT, `role\_id` int(11) DEFAULT NULL, `permission\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`id`) ) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

CREATE TABLE `sys\_permission` ( `id` int(11) NOT NULL AUTO\_INCREMENT, `name` varchar(255) DEFAULT NULL, `descritpion` varchar(255) DEFAULT NULL, `url` varchar(255) DEFAULT NULL, `pid` int(11) DEFAULT NULL, PRIMARY KEY (`id`) ) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

INSERT INTO `Sys\_permission` VALUES ('1', 'ROLE\_HOME', 'home', '/', null), ('2', 'ROLE\_ADMIN', 'ABel', '/admin', null);

INSERT INTO `Sys\_permission\_role` VALUES ('1', '1', '1'), ('2', '1', '2'), ('3', '2', '1');

其他改动跟上面是一致的

# 项目3：博客 springBoot+springSecurity 数据库动态管理用户、角色、权限（二）

地址：<http://blog.csdn.net/u012373815/article/details/54633046>

数据库更改为： 先清空 SYS\_USER 表，然后再新增加密的上去

insert into SYS\_USER (id,username, password) values (1,'admin', '$2a$10$YwbP2rm18bOWOrkJHybp5uTRHCpn5Rk8rGT6fogf0KdtNY7jzmebu');

insert into SYS\_USER (id,username, password) values (2,'abel', '$2a$10$/h0hVDo3A78lEHhsIckGz.nfXGgUFx2yB4bfy6o15RZi8VlZqt.PK');

注意：由于采用了 使用了httpBasic 认证 ,因此 用页面访问登录的方式已经不好操作了，因为要在 header 中添加内容，所以 本项目的所有试验 均在 postman 上面验证

http Basic 实际上就是将我门的用户名和密码连接起来然后 使用base64进行加密，将加密后的密文放在http 的header 中进行验证

帐号密码加密如下 (假设账号密码都为admin)

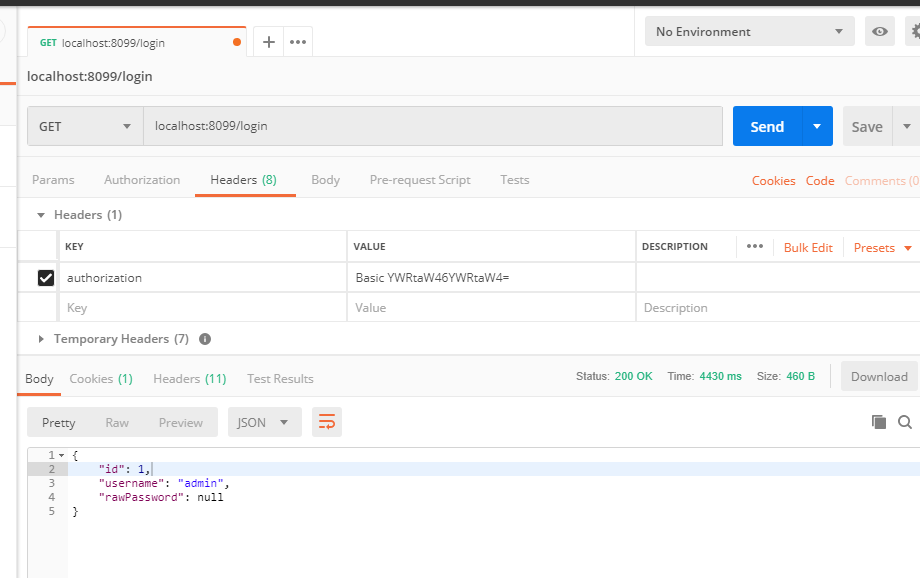
admin:admin base64 加密后为 YWRtaW46YWRtaW4=

加密后的串放入 header 时应在拼结为

Basic YWRtaW46YWRtaW4=

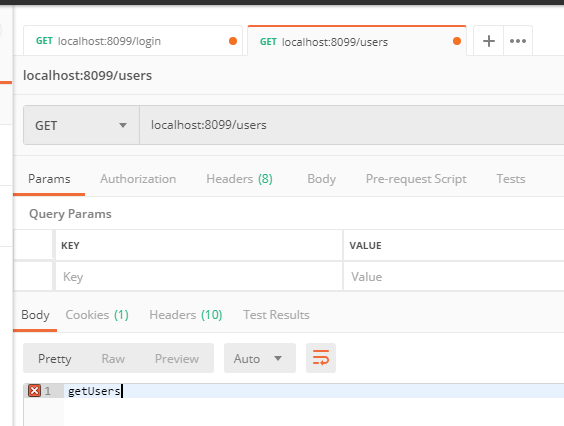
注意：Basic 与密码串之间为一个空格

postMan 请求如下：

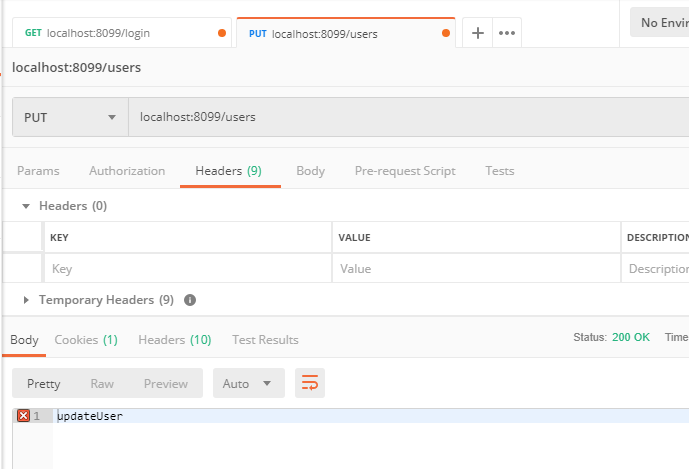


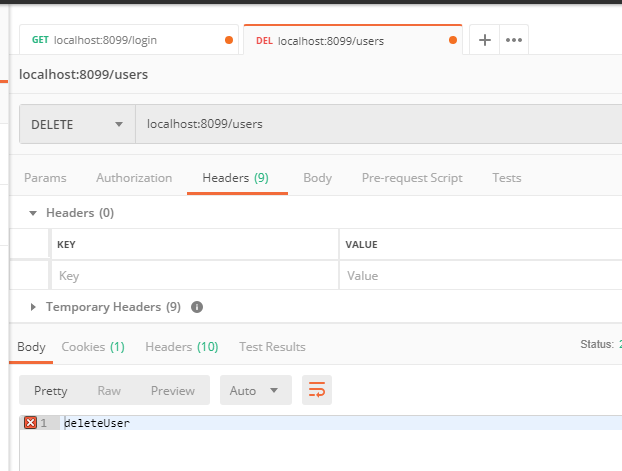
上图说明 admin 登录成功

验证 get 方法，postman直接选择 get方法 调用 localhost:8099/users 返回成功



验证 put方法，postman直接选择 put方法 调用 localhost:8099/users 返回成功





其他方法类似测试

测试结果发现：

用admin登录的话 也就是 YWRtaW46YWRtaW4= 可以访问 /users 路径下的get、post、put、delete 四个请求方法

使用abel 登录

帐号密码加密如下 (假设账号密码都为admin)

abel :abel base64 加密后为 YWJlbCA6YWJlbA==

加密后的串放入 header 时应在拼结为

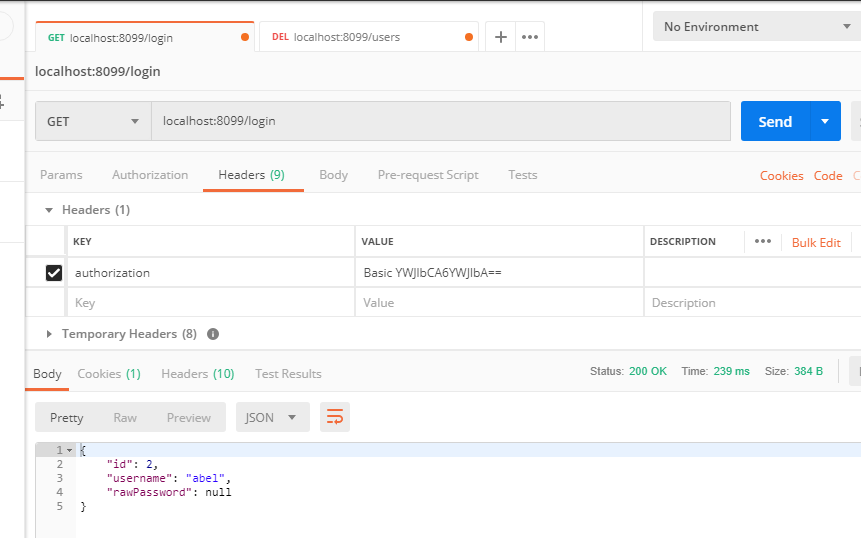
Basic YWJlbCA6YWJlbA==

注意：Basic 与密码串之间为一个空格

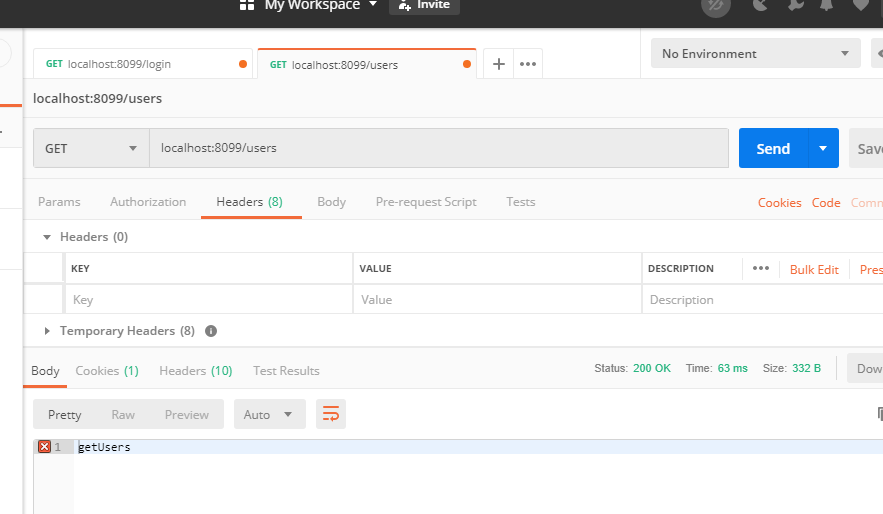
可以访问 /users 路径下的get、post 两个请求方法   
访问 put、delete 方法会抛403 没有权限异常

注意： base64 加解密 操作方式： <https://1024tools.com/base64>

例如，登录为：



访问 get 是有权限，可以访问的



访问 delete 是没有权限，会抛403 没有权限异常

