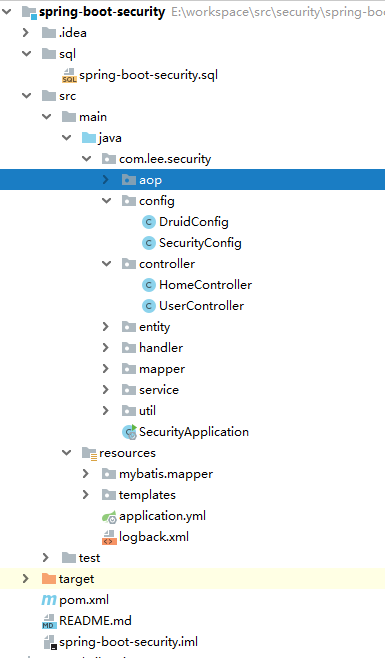
## **springboot整合security实现登录认证与授权**

### 项目结构



### 基本表结构

用户表（tbs\_user）

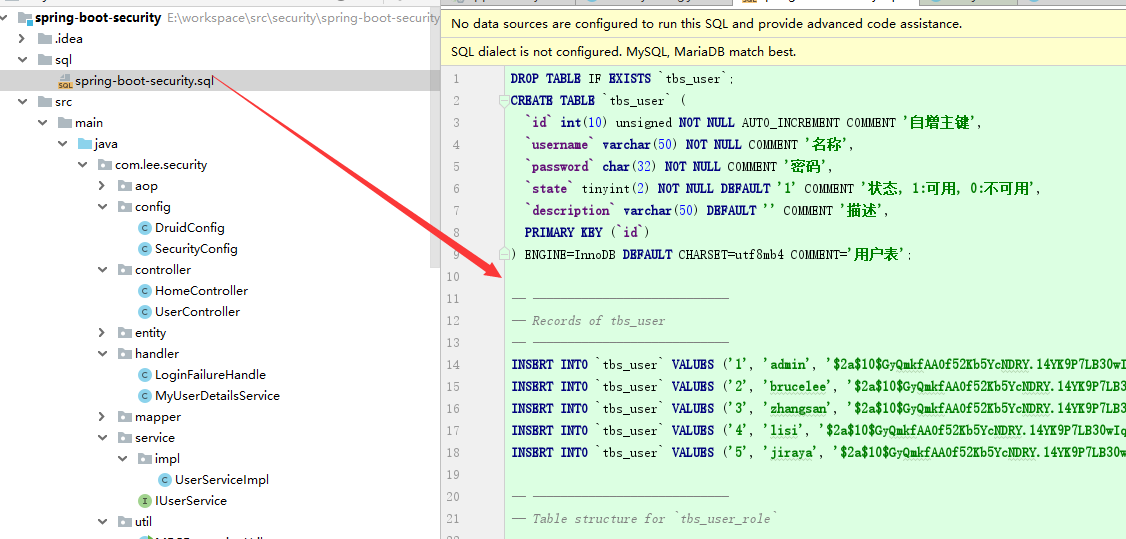
角色表（tbs\_role），

权限表（tbs\_permission）

用户角色表（tbs\_user\_role）

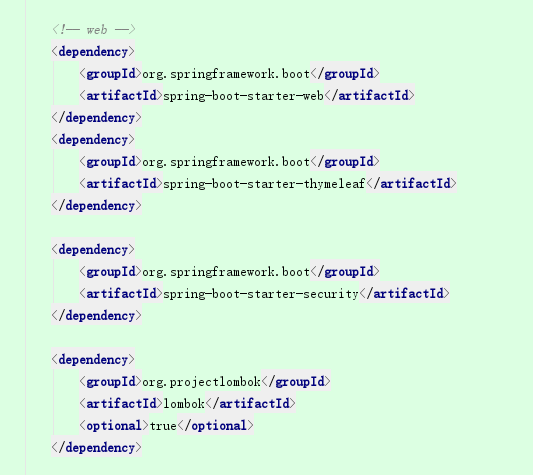
角色权限表（tbs\_role\_permission）

表生成可以根据执行项目中的spring-boot-security.sql脚本生成，如图所示：



### ****代码说明****

pom.xml主要依赖配置如下：



@EnableWebMvcSecurity 注解开启Spring Security的功能

@EnableGlobalMethodSecurity(prePostEnabled = true,securedEnabled=true) 启用方法级别的权限认证

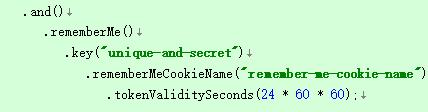
继承 WebSecurityConfigurerAdapter ，并重写它的方法来设置一些web安全的细节

configure(HttpSecurity http) 方法

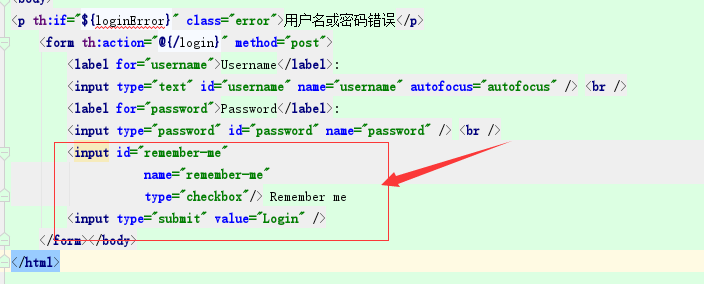
* + 通过 authorizeRequests() 定义哪些URL需要被保护、哪些不需要被保护。例如以上代码指定了 / 和 /home 不需要任何认证就可以访问，其他的路径都必须通过身份验证。
  + 通过 formLogin() 定义当需要用户登录时候，转到的登录页面。

@Configuration  
@EnableWebSecurity  
@EnableGlobalMethodSecurity(prePostEnabled = **true**,securedEnabled=**true**) *// 启用方法级别的权限认证***public class** SecurityConfig **extends** WebSecurityConfigurerAdapter {  
  
 @Autowired  
 **private** MyUserDetailsService **myUserDetailsService**;  
  
 @Autowired  
 **private** LoginFailureHandle **loginFailureHandle**;  
  
 */\*\*  
 \* 说明：  
 \* 除了“/”,”/home”(首页),”/login”(登录),”/logout”(注销),之外，其他路径都需要认证。  
 \* 指定“/login”该路径为登录页面，当未认证的用户尝试访问任何受保护的资源时，都会跳转到“/login”。  
 \* 默认指定“/logout”为注销页面  
 \** ***@param http*** *\** ***@throws*** *Exception  
 \*/* @Override  
 **protected void** configure(HttpSecurity http) **throws** Exception {  
 *// 允许所有用户访问"/"和"/index.html"* http.authorizeRequests()  
 .antMatchers(**"/"**, **"/index"**).permitAll()  
 .anyRequest().authenticated() *// 其他地址的访问均需验证权限* .and()  
 .formLogin()  
 .loginPage(**"/login"**) *// 登录页  
 //.failureHandler(loginFailureHandle)* .failureUrl(**"/login-error"**)  
 .permitAll()  
 .and()  
 .logout()  
 *//.logoutSuccessUrl("/login")* .and()  
 .rememberMe()  
 .key(**"unique-and-secret"**)  
 .rememberMeCookieName(**"remember-me-cookie-name"**)  
 .tokenValiditySeconds(24 \* 60 \* 60);  
 }

Spring Security记住我哈希认证配置，要想使记住我散列认证配置可用，我们需要在Spring中注册，通过扩展我们的Spring配置类WebSecurityConfigurerAdapter，我们可以简单地配置记住我的身份验证configure(HttpSecurity http)方法。我们需要配置一个安全和唯一的密钥。这个密钥通常是一个强大而独特的密码。我们可以随意配置“记住我”cookie名称并设置令牌有效期。默认为2周。configure(HttpSecurity http)添加如下配置



登陆页面添加如下配置



自定义UserDetailsService，从数据库获取用户认证信息

@Service  
@Slf4j  
**public class** MyUserDetailsService **implements** UserDetailsService {  
  
 @Autowired  
 **private** UserMapper **userMapper**;  
  
 @Autowired  
 **private** RoleMapper **roleMapper**;  
  
 @Autowired  
 **private** PermissionMapper **permissionMapper**;  
  
 @Override  
 **public** UserDetails loadUserByUsername(String s) **throws** UsernameNotFoundException {  
 User user = **userMapper**.findUserByUsername(s);  
 **if** (**null** == user) {  
 **throw new** UsernameNotFoundException(s);  
 }  
 List<Role> roles = **roleMapper**.findRoleByUsername(user.getUsername());  
 ***log***.info(**"用户：{}, 角色有{}个"**, user.getUsername(), roles.size());  
 List<SimpleGrantedAuthority> authorities = **new** ArrayList<>();  
 roles.stream().forEach(  
 role -> {  
 **permissionMapper**.findPermissionByRoleId(role.getId()).stream().forEach( *// 添加角色拥有的权限* permission -> authorities.add(**new** SimpleGrantedAuthority(permission.getPermission()))  
 );  
 }  
 );  
 **return new** org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), authorities);  
 }  
  
}

密码加密采用BCrypt强哈希方法 每次加密的结果都不一样

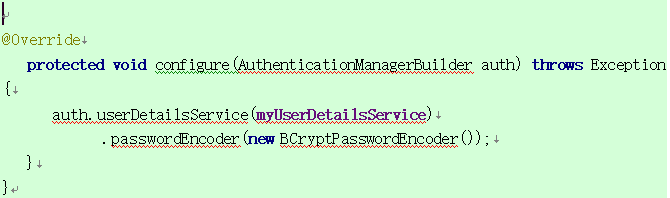
BCryptPasswordEncoder相关知识：

用户表的密码通常使用MD5等不可逆算法加密后存储，为防止彩虹表破解更会先使用一个特定的字符串（如域名）加密，然后再使用一个随机的salt（盐值）加密。

特定字符串是程序代码中固定的，salt是每个密码单独随机，一般给用户表加一个字段单独存储，比较麻烦。

BCrypt算法将salt随机并混入最终加密后的密码，验证时也无需单独提供之前的salt，从而无需单独处理salt问题。

添加 UserDetailsService， 实现自定义登录校验，在 SecurityConfig 中添加了如下代码：



也可以使用configureGlobal(AuthenticationManagerBuilder auth) 方法，在内存中创建了一个用户，这边使用的是从数据库加载，看具体业务而定

应用 BCryptPasswordEncoder 之后，明文密码是无法被识别的，就会校验失败，只有存入密文密码才能被正常识别。所以，应该在注册时对用户密码进行加密。初始密码加密的测试方法

@RunWith(SpringRunner.**class**)  
**public class** Test {  
  
 @org.junit.Test  
 **public void** testPasswordEncode() {  
 BCryptPasswordEncoder bCryptPasswordEncoder = **new** BCryptPasswordEncoder();  
 System.***out***.println(bCryptPasswordEncoder.encode(**"123456"**));  
 }  
}

创建了一些简单的导航控制器

@Controller  
**public class** HomeController {  
  
 @RequestMapping({**"/"**,**"/index"**})  
 **public** String index(){  
 **return"index"**;  
 }  
  
 @RequestMapping(**"/login"**)  
 **public** String login(HttpServletRequest request, Map<String, Object> map) **throws** Exception {  
  
 *// 此方法不处理登录成功,由shiro进行处理.\*/* **return "login"**;  
 }  
  
 @RequestMapping(**"/login-error"**)  
 **public** String loginError(Model model) {  
 model.addAttribute(**"loginError"**, **true**);  
 **return "login.html"**;  
 }  
  
}

@Controller  
@RequestMapping(**"/user"**)  
**public class** UserController {  
  
 **private static final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(UserController.**class**);  
  
 @Autowired  
 **private** IUserService **userService**;  
  
 */\*\*  
 \* 获取用户信息  
 \** ***@param user*** *\** ***@return*** *\*/* @RequestMapping(**"/me"**)  
 @ResponseBody  
 **public** Principal user(Principal user) {  
 **return** user;  
 }  
  
 @RequestMapping(**"/getUserName"**)  
 @ResponseBody  
 **public** String getUserName() {  
 String userName = **null**;  
 Object principal = SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  
  
 **if** (principal **instanceof** UserDetails) {  
 userName = ((UserDetails)principal).getUsername();  
 } **else** {  
 userName = principal.toString();  
 }  
 **return** userName;  
 }  
  
 @RequestMapping(**"/findUserByUsername"**)  
 @ResponseBody  
 **public** User findUserByUsername(String username) {  
 ***LOGGER***.info(**"username = {}"**, username);  
 User user = **userService**.findUserByUsername(username);  
 **return** user;  
 }  
  
 @RequestMapping(**"/userList"**)  
 @PreAuthorize(**"hasAuthority('user:view')"**)  
 **public** String userList(**int** pageNum, **int** pageSize, Model model) {  
 ***LOGGER***.info(**"pageNum = {}, pageSize = {}"**, pageNum, pageSize);  
 PageInfo pageInfo = **userService**.listUser(pageNum, pageSize);  
 model.addAttribute(**"pageInfo"**, pageInfo);  
  
 **return "user/userList"**;  
 }  
  
 */\*\*  
 \* 用户添加;  
 \** ***@return*** *\*/* @RequestMapping(**"/userAdd"**)  
 @PreAuthorize(**"hasAuthority('user:add')"**)  
 **public** String userAdd(){  
 **return "user/userAdd"**;  
 }  
  
 */\*\*  
 \* 用户删除;  
 \** ***@return*** *\*/* @RequestMapping(**"/userDel"**)  
 @PreAuthorize(**"hasAuthority('user:del')"**)  
 **public** String userDel(){  
 **return "user/userDel"**;  
 }

[@PreAuthorize](http://www.cnblogs.com/ranger2016/p/3914146.html)用于权限控制

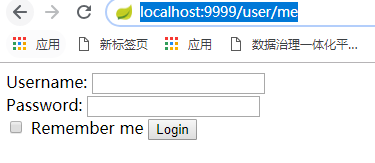
使用这个注解必须在SecurityConfig类中添加注解，@EnableGlobalMethodSecurity(prePostEnabled=true),prePostEnabled要设置为true，默认为false，如果不设置，[@PreAuthorize](http://www.cnblogs.com/ranger2016/p/3914146.html)拦截无效;

启动spring，用springboot来启动程序

@SpringBootApplication  
@EnableTransactionManagement *// 开启事务管理支持*@EnableAspectJAutoProxy  
**public class** SecurityApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(SecurityApplication.**class**, args);  
 }  
}

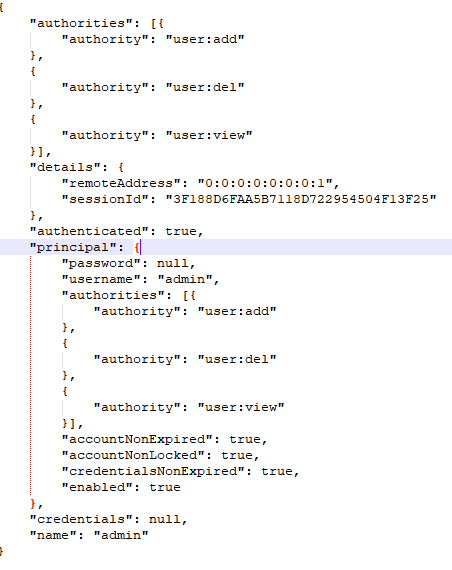
### 项目运行演示

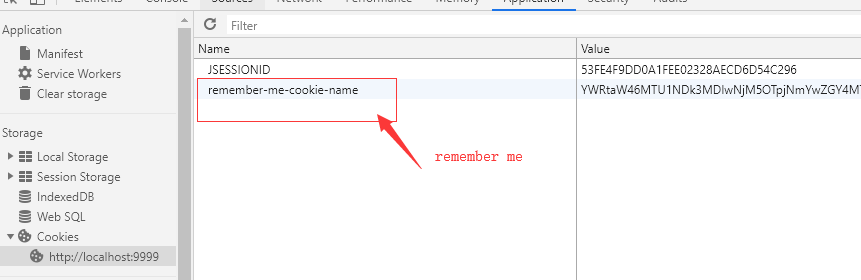
浏览器输入：<http://localhost:9999/user/me>，未登录过，则跳转到登录页面



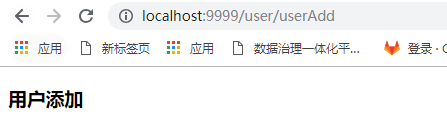
输入用户名/密码（admin/123456）,勾选remember me，点击login按钮

则浏览器输出用户具体信息：

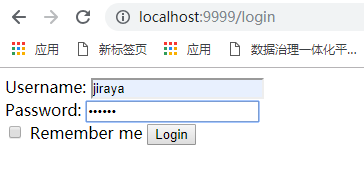




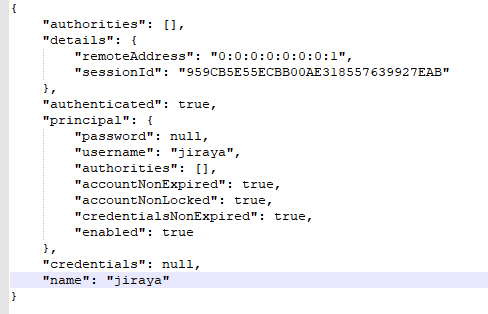
浏览器输入：<http://localhost:9999/user/userAdd>



更换用户登录：（jiraya/123456）



登录后请求：<http://localhost:9999/user/me>



由于该用户没有添加用户的权限，请求<http://localhost:9999/user/userAdd>

