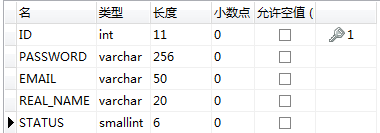
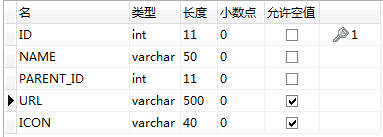
# 数据库设计

用户表ADMIN\_USER：



菜单表MENU：



角色表ROLE：



角色权限表ROLE\_PERMISSION：

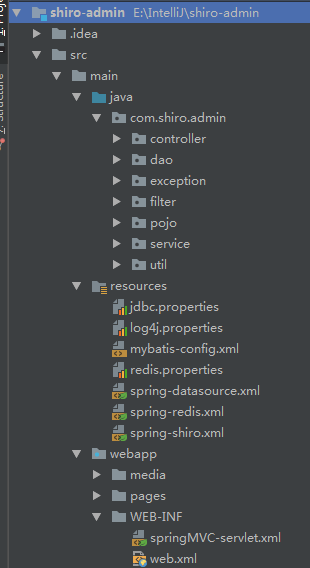


用户角色表USER\_ROLE：



# 工程搭建

完整的工程结构目录：



## 导入jar包

pom.xml文件配置

<dependencies>  
 <!--shiro包-->  
 <dependency>  
 <groupId>org.apache.shiro</groupId>  
 <artifactId>shiro-core</artifactId>  
 <version>1.2.3</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.shiro</groupId>  
 <artifactId>shiro-spring</artifactId>  
 <version>1.4.0</version>  
 </dependency>  
 <!--日志包-->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.6.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.6.1</version>  
 </dependency>  
 <!--servlet-->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>3.1.0</version>  
 </dependency>  
 <dependency>  
 <groupId>jstl</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <!-- Mybatis -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.2.8</version>  
 </dependency>  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.2.2</version>  
 </dependency>  
 <!--mysql驱动-->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.32</version>  
 </dependency>  
 <!--spring相关包-->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>4.3.11.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>4.3.11.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>4.3.11.RELEASE</version>  
 </dependency>  
 <!-- Jackson Json处理工具包 -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.7.4</version>  
 </dependency>  
 <!--redis包-->  
 <dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>2.9.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>1.8.7.RELEASE</version>  
 </dependency>

<dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>4.2.1</version>  
 </dependency>  
</dependencies>  
<!--加载配置文件-->  
<build>  
 <finalName>shiro-admin</finalName>  
 <resources>  
 <resource>  
 <directory>src/main/java</directory>  
 <includes>  
 <include>\*\*/\*.xml</include>  
 </includes>  
 </resource>  
 <resource>  
 <directory>src/main/resources</directory>  
 <includes>  
 <include>\*\*/\*.xml</include>  
 <include>\*\*/\*.properties</include>  
 </includes>  
 </resource>  
 </resources>  
</build>

## web.xml配置

*<?*xml version="1.0" encoding="UTF-8"*?>*<web-app xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 version="3.0">  
 <display-name>Archetype Created Web Application</display-name>  
 <!--加载spring配置文件-->  
 <context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-\*.xml</param-value>  
 </context-param>  
 <listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
 <!--编码过滤器，防止中文乱码-->  
 <filter>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 <init-param>  
 <param-name>forceEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
 </filter>  
 <filter-mapping>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
 <!--配置DispatcherServlet-->  
 <servlet>  
 <servlet-name>springMVC</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>springMVC</servlet-name>  
 <url-pattern>\*.html</url-pattern>  
 </servlet-mapping>  
  
 <!-- 配置shiro的核心拦截器 -->  
 <filter>  
 <filter-name>shiroFilter</filter-name>  
 <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  
 </filter>  
 <filter-mapping>  
 <filter-name>shiroFilter</filter-name>  
 <url-pattern>\*.html</url-pattern>  
 </filter-mapping>  
  
 <!--欢迎页面-->  
 <welcome-file-list>  
 <welcome-file>/login.html</welcome-file>  
 </welcome-file-list>  
</web-app>

## 编写配置文件

数据库配置文件jdbc.properties：

jdbc.driver=com.mysql.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/数据库?characterEncoding=utf-8  
jdbc.username=用户名  
jdbc.password=密码

日志文件log4j.properties：

log4j.rootLogger=INFO, stdout,   
  
log4j.appender.stdout=org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n  
  
log4j.logger.com.shiro.admin=DEBUG

redis配置文件redis.properties：

redis.host=127.0.0.1  
redis.port=6379   
redis.maxIdle=300   
redis.maxActive=600   
redis.maxWait=1000   
redis.testOnBorrow=true

springMVC-servlet.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd  
 http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd">  
 <!-- 配置资源映射 -->  
 <mvc:resources location="/media/" mapping="/\*\*"/>  
  
 <context:annotation-config/>  
 <!-- 启动自动扫描 -->  
 <context:component-scan base-package="com.shiro.admin.controller"

use-default-filters="false">  
 <context:include-filter type="annotation"

expression="org.springframework.stereotype.Controller"/>  
 </context:component-scan>  
  
 <!-- 异常处理 -->  
 <bean id="adminExceptionHandler"

class="com.shiro.admin.exception.AdminExceptionHandler"/>  
  
 <!--配置返回值转换器-->  
 <bean id="contentNegotiationManagerFactoryBean"  
 class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">  
 <!--是否支持后缀匹配-->  
 <property name="favorPathExtension" value="false"/>  
 <!--是否支持参数匹配-->  
 <property name="favorParameter" value="false"/>  
 <!--是否accept-header匹配-->  
 <property name="ignoreAcceptHeader" value="false"/>  
 <property name="mediaTypes">  
 <map>  
 <entry key="json" value="application/json"/>  
 </map>  
 </property>  
 </bean>  
 <mvc:annotation-driven

content-negotiation-manager="contentNegotiationManagerFactoryBean">  
 </mvc:annotation-driven>  
 <!--请求支持类型-->  
 <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">  
 <property name="messageConverters">  
 <list>  
 <bean

class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>application/json;charset=utf-8</value>  
 <value>text/html;charset=UTF-8</value>  
 </list>  
 </property>  
 </bean>  
 </list>  
 </property>  
 </bean>  
</beans>

spring-datasource.xml配置数据库和DAO的相关信息

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  
 http://www.springframework.org/schema/aop  
 http://www.springframework.org/schema/aop/spring-aop-3.0.xsd   
 http://www.springframework.org/schema/tx  
 http://www.springframework.org/schema/tx/spring-tx-3.0.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context-4.2.xsd">  
 <!-- 加载 properties -->  
 <context:property-placeholder

location="classpath:\*.properties" ignore-unresolvable="true"/>  
 <!--配置数据源-->  
 <bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property name="driverClassName" value="${jdbc.driver}"/>  
 <property name="url" value="${jdbc.url}"/>  
 <property name="username" value="${jdbc.username}"/>  
 <property name="password" value="${jdbc.password}"/>  
 </bean>  
  
 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="dataSource" ref="dataSource"/>  
 <!--扫描Mybatis所有映射文件-->  
 <property name="mapperLocations"

value="classpath:com/shiro/admin/dao/mappers/\*.xml"/>  
 <!--Mybatis配置文件的路径，里面设置了类的别名-->  
 <property name="configLocation" value="classpath:mybatis-config.xml"/>  
 </bean>  
 <!--事务管理器-->  
 <bean id="txManager"  
 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource"/>  
 </bean>  
 <!--定义声明式事务(面向切面)-->  
 <tx:advice id="txAdvice" transaction-manager="txManager">  
 <tx:attributes>  
 <tx:method name="add\*"

isolation="REPEATABLE\_READ" rollback-for="Exception"/>  
 <tx:method name="update\*"

isolation="REPEATABLE\_READ" rollback-for="Exception"/>  
 <tx:method name="delete\*"

isolation="REPEATABLE\_READ" rollback-for="Exception"/>  
 <tx:method name="get\*"

isolation="REPEATABLE\_READ" read-only="true"/>  
 <tx:method name="do\*"

isolation="REPEATABLE\_READ" rollback-for="Exception"/>  
 <tx:method name="\*"

isolation="REPEATABLE\_READ" propagation="NOT\_SUPPORTED"/>  
 </tx:attributes>  
 </tx:advice>  
 <!--切入点-->  
 <aop:config>  
 <aop:pointcut

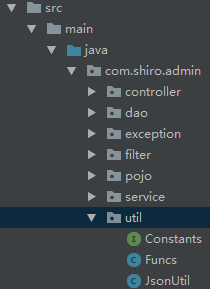
expression="execution(\* com.shiro.admin.service..\*(..))" id="point"/>  
 <aop:advisor advice-ref="txAdvice" pointcut-ref="point"/>  
 </aop:config>  
 <!--扫描所有的Mybatis的DAO接口，生成代理实现类-->  
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <property name="basePackage" value="com.shiro.admin.dao"/>  
 </bean>  
</beans>

spring声明式事务isolation的介绍，面试的时候可能问，参见博客：

<http://blog.csdn.net/u011731544/article/details/78340228>

## 通用工具类

工具类在util包下



**package com.shiro.admin.util**;  
  
**/\*\*  
 \* 定义字符串常量，整个工程中通用  
 \*/  
public interface** Constants **{  
 // md5(用户密码+PASSWORD\_SALT\_KEY)保存到数据库中。  
 String *PASSWORD\_SALT\_KEY* = "Shiro.admin.2017"**;  
 **//Shiro的session中存放用户的key  
 String *SESSION\_USER\_KEY* = "SESSION\_USER\_KEY"**;  
 **//redis中存放的用户权限菜单的key  
 String *SESSION\_USER\_MANU* = "SESSION\_USER\_MANU"**;  
 **//Shiro存放的角色信息  
 String *SESSION\_USER\_ROLE* = "SESSION\_USER\_ROLE"**;  
**}**

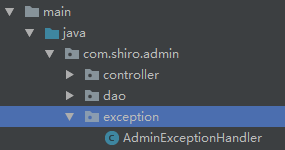
一些自定义的函数Funcs

**package com.shiro.admin.util**;  
  
**import javax.servlet.http.**HttpServletRequest;  
**import java.security.**MessageDigest;  
  
**public class Funcs {  
 /\*\*  
 \* 生成MD5加密字符串  
 \*/  
 public static String** MD5(**String pwd**) **{  
 char md5String**[] **= {'0'**, **'1'**, **'2'**, **'3'**, **'4'**, **'5'**, **'6'**, **'7'**, **'8'**, **'9'**,  
 **'A'**, **'B'**, **'C'**, **'D'**, **'E'**, **'F'}**;  
 **try {  
 byte**[] **btInput = pwd**.getBytes();  
 MessageDigest **mdInst =** MessageDigest.getInstance(**"MD5"**);  
 **mdInst**.update(**btInput**);  
 **byte**[] **md = mdInst**.digest();  
 **int j = md**.**length**;  
 **char str**[] **= new char**[**j \* 2**];  
 **int k = 0**;  
 **for** (**int i = 0**; **i < j**; **i++**) **{ // i = 0  
 byte byte0 = md**[**i**]; **//95  
 str**[**k++**] **= md5String**[**byte0 >>> 4 & 0xf**]; **// 5  
 str**[**k++**] **= md5String**[**byte0 & 0xf**]; **// F  
 }  
 return new** String(**str**);  
 **} catch** (**Exception e**) **{  
 return null**;  
 **}  
 }  
  
 /\*\*  
 \* 判断请求是不是ajax请求  
 \*/  
 public static boolean** isAjax(HttpServletRequest **request**) **{  
 return "XMLHttpRequest"**.equalsIgnoreCase(**request**.**getHeader**(**"X-Requested-With"**)) **|| request**.**getParameter**(**"ajax"**) **!= null**;  
 **}  
}**

JSON转换工具类JsonUtil

**package com.shiro.admin.util**;  
  
**import com.fasterxml.jackson.core.type.**TypeReference;  
**import com.fasterxml.jackson.databind.ObjectMapper**;  
  
**import java.io.IOException**;  
**import java.text.SimpleDateFormat**;  
**import java.util.HashMap**;  
**import java.util.**Map;  
  
  
**/\*\*  
 \* JSON工具类  
 \*/  
public class JsonUtil {  
  
 /\*\*  
 \* 将Java对象转化为JSON字符串  
 \*/  
 public static String** getJSON(**Object obj**) **throws IOException {  
 if** (**null == obj**) **{  
 return ""**;  
 **}  
 ObjectMapper mapper = new** ObjectMapper();  
 **//转换date类型的时候，时间戳  
 mapper**.getSerializationConfig().with(**new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**));  
 **String jsonStr = mapper**.writeValueAsString(**obj**);  
 **return jsonStr**;  
 **}  
  
 /\*\*  
 \* 将JSON字符串转化为Java对象，集合  
 \*/  
 public static <T> T** getObj(**String json**, TypeReference**<T> ref**)  
 **throws IOException {  
 if** (**null == json || json**.length() **== 0**) **{  
 return null**;  
 **}  
 ObjectMapper mapper = new** ObjectMapper();  
 **mapper**.getDeserializationConfig().with(**new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**));  
 **return** (**T**) **mapper**.readValue(**json**, **ref**);  
 **}  
 /\*\*  
 \* 将JSON字符串转化为Java对象，一个对象  
 \*/  
 public static Object** getObj(**String json**, **Class pojoClass**) **throws Exception {  
 ObjectMapper mapper = new** ObjectMapper();  
 **return mapper**.readValue(**json**, **pojoClass**);  
 **}  
  
 /\*\*  
 \* 本工程中页面用ajax请求成功时，统一使用这个方法封装  
 \* 页面根据statusCode判断请求是否成功  
 \*/  
 public static** Map**<String**, **String>** getOkStatusMsg(**String msg**) **{** Map**<String**, **String> res = new** HashMap**<String**, **String>**();  
 **res**.**put**(**"statusCode"**, **"200"**);  
 **if** (**null != msg && msg**.length() **> 0**) **{  
 res**.**put**(**"message"**, **msg**);  
 **}  
 return res**;  
 **}  
 /\*\*  
 \* 本工程中页面用ajax请求失败时，统一使用这个方法封装  
 \* 页面根据statusCode判断请求是否成功  
 \*/  
 public static** Map**<String**, **String>** getErrorStatusMsg(**String msg**) **{** Map**<String**, **String> res = new** HashMap**<String**, **String>**();  
 **res**.**put**(**"statusCode"**, **"300"**);  
 **res**.**put**(**"message"**, **msg**);  
 **return res**;  
 **}  
}**

## 自定义异常



**package com.shiro.admin.exception**;  
  
**import org.slf4j.**Logger;  
**import org.slf4j.LoggerFactory**;  
**import org.springframework.web.servlet.**HandlerExceptionResolver;  
**import org.springframework.web.servlet.ModelAndView**;  
**import org.springframework.web.servlet.view.json.MappingJackson2JsonView**;  
**import com.shiro.admin.util.Funcs**;  
**import com.shiro.admin.util.JsonUtil**;  
  
**import javax.servlet.http.**HttpServletRequest;  
**import javax.servlet.http.**HttpServletResponse;  
  
  
**/\*\*  
 \* 自定义异常  
 \*/  
public class AdminExceptionHandler implements** HandlerExceptionResolver **{  
  
 private static** Logger logger **= LoggerFactory**.getLogger(**AdminExceptionHandler**.**class**);  
  
 **public ModelAndView** resolveException(HttpServletRequest **request**, HttpServletResponse **response**, **Object handler**, **Exception ex**) **{  
 String msg = ex**.getMessage();  
 logger.**error**(**msg**, **ex**);  
 **return** handlerError(**request**, **response**, **msg**);  
 **}  
  
 public static ModelAndView** handlerError(HttpServletRequest **request**, HttpServletResponse **response**, **String msg**) **{  
 if** (**response**.**isCommitted**()) **{  
 return null**;  
 **}  
 if** (**Funcs**.isAjax(**request**)) **{  
 MappingJackson2JsonView JSON\_VIEW = new** MappingJackson2JsonView();  
 **return new** ModelAndView(**JSON\_VIEW**, **JsonUtil**.getErrorStatusMsg(**msg**));  
 **} else {  
 request**.**setAttribute**(**"errorMsg"**, **msg**);  
 **return new** ModelAndView(**"/pages/error.jsp"**);  
 **}  
 }  
  
}**

# 登录功能

## 实体类

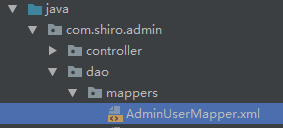
**package com.shiro.admin.pojo**;  
  
**public class AdminUser {  
 private Integer id**;**//主键  
 private String email**;**//邮箱，登录的用户名  
 private String password**;**//密码  
 private String realName**;**//真实姓名  
 private Integer status**;**//状态，0禁用，1有效**

**//getter/setter方法略**

**}**

mybatis-conf.xml中配置别名。略。

## AdminUserMapper.xml



*<?*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="com.shiro.admin.dao.AdminUserDAO">  
  
 <resultMap id="adminUserMap" type="AdminUser">  
 <id property="id" column="ID"/>  
 <result property="email" column="EMAIL"/>  
 <result property="password" column="PASSWORD"/>  
 <result property="realName" column="REAL\_NAME"/>  
 <result property="status" column="STATUS"/>  
 </resultMap>  
  
 <sql id="select-from-adminuser">  
 select ID, EMAIL, PASSWORD, REAL\_NAME, STATUS from ADMIN\_USER  
 </sql>  
  
 <sql id="where-from-adminuser">  
 <where>  
 <if test="id != null">and ID = #{id}</if>  
 <if test="email != null and email != ''">and EMAIL = #{email}</if>  
 <if test="realName != null and realName != ''">and REAL\_NAME like CONCAT('%', #{realName}, '%')</if>  
 <if test="status != null">and STATUS = #{status}</if>  
 </where>  
 </sql>  
  
 <select id="getUserByEmail" parameterType="java.lang.String" resultMap="adminUserMap">  
 <include refid="select-from-adminuser"/>  
 where EMAIL = #{email}  
 </select>  
  
</mapper>

## AdminUserDAO

**package com.shiro.admin.dao**;  
  
**import com.shiro.admin.pojo.AdminUser**;  
  
**public interface** AdminUserDAO **{  
  
 AdminUser** getUserByEmail(**String email**);

}

## AdminUserService接口及实现类

**package com.shiro.admin.service**;  
  
**public interface** AdminUserService **{  
  
 Integer** doLogin(**String email**, **String password**) **throws Exception**;

**}**

实现类

**package com.shiro.admin.service.impl**;  
  
**import com.shiro.admin.dao.**AdminUserDAO;  
**import com.shiro.admin.pojo.AdminUser**;  
**import com.shiro.admin.service.**AdminUserService;  
**import org.apache.shiro.**SecurityUtils;  
**import org.apache.shiro.authc.\***;  
**import org.apache.shiro.session.**Session;  
**import org.apache.shiro.subject.**Subject;  
**import org.springframework.beans.factory.annotation.Autowired**;  
**import org.springframework.stereotype.Service**;  
**import com.shiro.admin.util.**Constants;  
**import com.shiro.admin.util.Funcs**;  
  
  
**import java.util.**List;  
 **@Service  
public class AdminUserServiceImpl implements** AdminUserService **{  
  
 @Autowired  
 private** AdminUserDAO **adminUserDAO**;  
  
 **/\*\*  
 \* 登录  
 \*/  
 @Override  
 public Integer** doLogin(**String email**, **String password**) **throws Exception {  
 //数据库中存的是密文，所以要先将用户的输入加密再比较  
 password = Funcs**.MD5(**password +** Constants.***PASSWORD\_SALT\_KEY***);  
 Subject **currentUser =** SecurityUtils.getSubject();  
 **//根据用户名查询出用户信息  
 AdminUser user = adminUserDAO**.**getUserByEmail**(**email**);  
 **//如果user为null表示用户不存在，user.getStatus() == 0表示用户被禁用  
 if** (**user == null || user**.getStatus() **== 0 || !password**.equals(**user**.getPassword())) **{  
 throw new** Exception(**"用户名或密码错误"**);  
 **}  
 try {  
 //使用Shiro进行登录  
 UsernamePasswordToken token =  
 new** UsernamePasswordToken(**email**, **password**);  
 **currentUser**.**login**(**token**);**//执行登录** Session **session = currentUser**.**getSession**();  
 **//保存用户的信息  
 session**.**setAttribute**(Constants.***SESSION\_USER\_KEY***, **user**);  
 **return user**.getId();  
 **} catch** (**Exception ex**) **{  
 ex**.printStackTrace();  
 **throw new** Exception(**ex**);  
 **}  
 }**

**}**

## 自定义Shiro的Ream

**package com.shiro.admin.filter**;  
  
**import org.apache.shiro.authc.\***;  
**import org.apache.shiro.authz.**AuthorizationInfo;  
**import org.apache.shiro.realm.**AuthorizingRealm;  
**import org.apache.shiro.subject.**PrincipalCollection;  
  
**import java.util.ArrayList**;  
**import java.util.**List;  
  
**public class MyShiroRealm extends** AuthorizingRealm **{**  
  
 **/\*\*  
 \* 每次请求url都会调用这个方法  
 \*/  
 @Override  
 protected** AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection **principalCollection**) **{  
 return null**;  
 **}  
  
 @Override  
 protected** AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken **authenticationToken**) **throws AuthenticationException {  
 UsernamePasswordToken token =** (**UsernamePasswordToken**) **authenticationToken**;  
 **// 因为service里已经验证过用户名密码了，所以这里不做任何验证了  
 String email = token**.getUsername();  
 **if** (**email != null && !""**.equals(**email**)) **{  
 return new** SimpleAuthenticationInfo(**email**, **token**.getPassword(), getName());  
 **}  
 return null**;  
 **}  
}**

## MyShiroFilterFactoryBean

这个类后面做权限的时候用到

**package com.shiro.admin.filter**;  
  
**import org.apache.shiro.config.Ini**;  
**import org.apache.shiro.spring.web.ShiroFilterFactoryBean**;  
**import org.apache.shiro.util.CollectionUtils**;  
**import org.apache.shiro.web.config.IniFilterChainResolverFactory**;  
  
**public class MyShiroFilterFactoryBean extends ShiroFilterFactoryBean {  
 public static final String *ROLE\_STRING* = "roles[{0}]"**;  
 **public static String** filterChainDefinitions **= ""**;  
  
  
 **@Override  
 public void** setFilterChainDefinitions(**String definitions**) **{  
 try {** filterChainDefinitions **= definitions**;**// 记录配置的静态过滤链  
  
 Ini ini = new** Ini();  
 **ini**.load(filterChainDefinitions);  
 **Ini**.**Section section = ini**.getSection(**IniFilterChainResolverFactory**.***URLS***);  
 **if** (**CollectionUtils**.isEmpty(**section**)) **{  
 section = ini**.getSection(**Ini**.***DEFAULT\_SECTION\_NAME***);  
 **}  
 section**.put(**"/\*\*"**, **"authc"**);  
 setFilterChainDefinitionMap(**section**);  
 **} catch** (**Exception e**) **{  
 e**.printStackTrace();  
 **}  
 }  
}**

## SystemController

**package com.shiro.admin.controller**;  
  
**import com.shiro.admin.service.**AdminUserService;  
**import com.shiro.admin.util.**Constants;  
**import org.apache.shiro.**SecurityUtils;  
**import org.apache.shiro.subject.**Subject;  
**import org.springframework.beans.factory.annotation.Autowired**;  
**import org.springframework.stereotype.Controller**;  
**import org.springframework.ui.**Model;  
**import org.springframework.web.bind.annotation.RequestMapping**;  
**import org.springframework.web.servlet.ModelAndView**;

**@Controller  
public class SystemController implements** Constants **{**  
 **@Autowired  
 private** AdminUserService **adminUserService**;  
  
 **@RequestMapping**(**"/index.html"**)  
 **public String** index() **{//去首页  
 return "/pages/welcome.jsp"**;  
 **}  
  
 @RequestMapping**(**"/login.html"**)  
 **public String** login() **{//去登录页  
 return "/pages/login.jsp"**;  
 **}  
  
 @RequestMapping**(**"/dologin.html"**)**//执行登录  
 public ModelAndView** login(**String email**, **String password**) **throws Exception {  
 try {  
 Integer userId = adminUserService**.**doLogin**(**email**, **password**);  
 **} catch** (**Exception ex**) **{  
 return new** ModelAndView(**"/pages/login.jsp"**, **"loginErrInfo"**, **ex**.getMessage());  
 **}  
 return new** ModelAndView(**"redirect:/index.html"**);  
 **}  
  
 @RequestMapping**(**"/logout.html"**)  
 **public ModelAndView** logout() **{//退出登录** Subject **currentUser =** SecurityUtils.getSubject();  
 **currentUser**.**logout**();**//shiro退出登录，清空权限信息  
 return new** ModelAndView(**"redirect:/login.html"**);  
 **}  
  
 @RequestMapping**(**"/auth\_error.html"**)  
 **public String** authError(Model **model**) **{  
 model**.**addAttribute**(**"errorMsg"**, **"权限不足"**);  
 **return "/pages/error.jsp"**;  
 **}**

**}**

## spring-shiro.xml配置文件

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd">  
 <!--扫描除Controller以外的所有组件，主要是Service-->  
 <context:component-scan base-package="com.shiro.admin.\*">  
 <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller" />  
 </context:component-scan>  
 <!--自定义的Shiro数据源-->  
 <bean id="myShiroRealm" class="com.shiro.admin.filter.MyShiroRealm">  
 </bean>  
 <bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">  
 <property name="realm" ref="myShiroRealm"/>  
 </bean>  
 <!--Shiro过滤器-->  
 <bean id="shiroFilter" class="com.shiro.admin.filter.MyShiroFilterFactoryBean">  
 <property name="securityManager" ref="securityManager"/>  
 <property name="loginUrl" value="/login.html"/>  
 <!--登录成功的跳转地址-->  
 <property name="successUrl" value="/index.html"/>  
 <!--验证失败的跳转地址-->  
 <property name="unauthorizedUrl" value="/auth\_error.html"/>  
 <!--静态的条件,登录不验证-->  
 <property name="filterChainDefinitions">  
 <value>  
 /login.html = anon  
 /dologin.html = anon  
 </value>  
 </property>  
 </bean>  
</beans>

## 页面

使用easyUI进行布局，页面中引用文件：

<link rel="stylesheet" type="text/css" href="/media/easyUI/themes/default/easyui.css">  
<link rel="stylesheet" type="text/css" href="/media/easyUI/themes/icon.css">  
<script type="text/javascript" src="/media/js/jquery-1.10.2.min.js"></script>  
<script type="text/javascript" src="/media/js/jquery.easyui.min.js"></script>

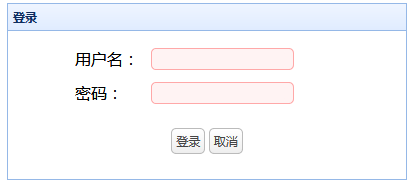
<div class="easyui-panel" title="登录" style="width:400px">  
 <div style="padding:10px 60px 20px 60px">  
 <form id="ff" action="/dologin.html" method="post">  
 <table cellpadding="5">  
 <tr>  
 <td>用户名：</td>  
 <td>

<input class="easyui-textbox" type="text"

name="email"  
 data-options="required:true,validType:'email'"/ >

</td>  
 </tr>  
 <tr>  
 <td>密码：</td>  
 <td><input class="easyui-textbox" type="password"

name="password"  
 data-options="required:true"/></td>  
 </tr>  
 </table>  
 </form>  
 <div style="text-align:center;padding:5px">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="*submitForm*()">登录</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick=" $('#ff').form('clear')">取消</a><span style="color:red">${loginErrInfo}</span>  
 </div>  
 </div>  
</div>  
<script>  
 function *submitForm*() {  
 $('#ff').form('submit', {  
 onSubmit: function () {  
 if ($(this).form('enableValidation').form('validate')) {  
 $('#ff').submit();//必须加，否则提交后页面不跳转  
 }  
 return false;  
 }  
 });  
 }  
</script>



登录成功进入首页

# 菜单权限管理模块

## 实体类

菜单节点属性类：

**package com.shiro.admin.pojo**;  
  
**public class TreeNodeAttribute {  
 private Integer parentId**; **//父菜单id**  
 **private String url**; **//链接地址**  
  
 **//getter/setter方法略**

**}**

菜单节点类：

**package com.shiro.admin.pojo**;  
  
**import java.util.ArrayList**;  
**import java.util.**List;  
  
**public class TreeNode {  
 private Integer id**;**//id  
 private String text**;**//对应tree的text属性  
 private TreeNodeAttribute attributes = new** TreeNodeAttribute();**//节点属性  
 private** List**<TreeNode> children = new** ArrayList**<>**();**//子节点集合  
  
 //gettet/setter方法略**

**}**

mybatis-config.xml配置别名，略

## PermissionMapper.xml

*<?*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="com.shiro.admin.dao.PermissionDAO">  
 <resultMap id="menuMap" type="TreeNode">  
 <id property="id" column="ID"/>  
 <result property="text" column="NAME"/>  
 <association property="attributes" javaType="TreeNodeAttribute">  
 <result property="parentId" column="PARENT\_ID"/>  
 <result property="url" column="URL"/>  
 </association>  
 </resultMap>  
  
 <select id="getUserMenuByUserId" parameterType="Integer" resultMap="menuMap">  
 select M.ID, M.NAME, M.PARENT\_ID, M.URL  
 from MENU M  
 inner join ROLE\_PERMISSION RP on M.ID=RP.MENU\_ID  
 inner join USER\_ROLE UR on UR.ROLE\_ID=RP.ROLE\_ID  
 where M.PARENT\_ID>=1 and UR.USER\_ID=#{userId}  
 </select>

</mapper>

## PermissionDAO

**package com.shiro.admin.dao**;  
  
**import com.shiro.admin.pojo.Menu**;  
  
**import java.util.**List;  
  
**public interface** PermissionDAO **{** List**<TreeNode>** getUserMenuByUserId(**Integer userId**);

**}**

## MenuService接口及实现类

**package com.shiro.admin.service**;  
  
**import com.shiro.admin.pojo.AdminUser**;  
**import com.shiro.admin.pojo.Menu**;  
  
**public interface** MenuService **{** List**<TreeNode>** getMenuTree(**AdminUser user**);

**}**

实现类MenuServiceImpl

**package com.shiro.admin.service.impl**;  
  
**import com.shiro.admin.dao.**MenuDAO;  
**import com.shiro.admin.dao.**PermissionDAO;  
**import com.shiro.admin.pojo.AdminUser**;  
**import com.shiro.admin.pojo.Menu**;  
**import com.shiro.admin.pojo.TreeNode**;  
**import com.shiro.admin.service.**MenuService;  
**import org.springframework.beans.factory.annotation.Autowired**;  
**import org.springframework.stereotype.Service**;  
  
**import java.util.ArrayList**;  
**import java.util.HashMap**;  
**import java.util.**List;  
**import java.util.**Map;  
  
**@Service  
public class MenuServiceImpl implements** MenuService **{  
 @Autowired  
 private** PermissionDAO **permissionDAO**;  
 **@Autowired  
 private** MenuDAO **menuDAO**;  
  
 **@Override  
 public** List**<TreeNode>** getUserMenu(**AdminUser user**) **{  
 //查询出用户所有的权限菜单** List**<TreeNode> menus = permissionDAO**.**getUserMenuByUserId**(**user**.getId());  
 **return** makeTree(**menus**);  
 **}  
  
 private** List**<TreeNode>** makeTree(List**<TreeNode> menus**) **{** Map**<Integer**, **TreeNode> map = new** HashMap**<>**();  
 **for** (**TreeNode menu : menus**) **{  
 map**.**put**(**menu**.getId(), **menu**);  
 **}  
 for** (**TreeNode menu : menus**) **{  
 //组装子菜单  
 //menu.getParentId() != null代表当前的菜单是子菜单  
 TreeNode parent = map**.**get**(**menu**.getAttributes().getParentId());  
 **if** (**parent != null**) **{  
 map**.**get**(**menu**.getAttributes().getParentId()).getChildren().**add**(**menu**);  
 **}  
 }  
 //1是根节点，返回根节点的子菜单就是一级菜单  
 return map**.**get**(**1**).getChildren();  
 **}**

**}**

## SystemController

**@Autowired  
private** MenuService **menuService**;  
  
  
**@RequestMapping**(**"/system/sidebar.html"**)  
**@ResponseBody  
public** List**<TreeNode>** getSidebar() **throws Exception {** Subject **currentUser =** SecurityUtils.getSubject();  
 Session **session = currentUser**.**getSession**();  
 **AdminUser user =** (**AdminUser**) **session**.**getAttribute**(Constants.***SESSION\_USER\_KEY***);  
 List**<TreeNode> userMenu =** (List**<TreeNode>** ) **session**.**getAttribute**(Constants.***SESSION\_USER\_MANU***);  
 **if** (**userMenu == null**) **{  
 userMenu = menuService**.getUserMenu(**user**);  
 **session**.**setAttribute**(Constants.***SESSION\_USER\_MANU***, **userMenu**);  
 **}  
 return userMenu**;  
**}**

## 页面

参考资料中pages/welcome.jsp

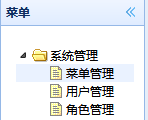
<head>  
 <title>Shiro管理后台</title>  
 <link rel="stylesheet" type="text/css" href="/media/easyUI/themes/default/easyui.css">  
 <link rel="stylesheet" type="text/css" href="/media/easyUI/themes/icon.css">  
 <script type="text/javascript" src="/media/js/jquery-1.10.2.min.js"></script>  
 <script type="text/javascript" src="/media/js/jquery.easyui.min.js"></script>  
 <script type="text/javascript">  
 $(function () {  
 $('#tree').tree({  
 url: "/system/sidebar.html", //远程加载tree数据的url路径  
 animate: true, //是否开启动画效果  
 checkbox: false, //是否显示复选框  
 onClick: function(node){  
 if($('#tree').tree("isLeaf",node.target)){  
 var tabs = $("#tabs");  
 var tab = tabs.tabs("getTab",node.text);  
 if(tab){  
 tabs.tabs("select",node.text);  
 }else{  
 tabs.tabs('add',{  
 title:node.text,  
 href: node.attributes.url,  
 closable:true,  
 bodyCls:"content"  
 });  
 }  
 }  
 }  
 });  
 });  
 </script>  
</head>  
<body class="easyui-layout">

<div data-options="region:'north',border:false" style="background:#000;">  
 <img src="/media/qf\_logo.jpg"/>  
</div>  
<div data-options="region:'west',split:true,title:'菜单'" style="width:150px; ">  
 <ul id="tree" class="easyui-tree" style="margin-top: 10px;margin-left: 5px;">

</ul>  
</div>  
<div data-options="region:'center',title:'Shiro管理后台'">  
 <div id="tabs" class="easyui-tabs">  
 </div>  
</div>

</body>

登录后可以看到菜单：



# Shiro权限验证

Shiro在启动的时候要把所有角色对应的权限加载进来，所以要查询每一个菜单有多少个角色可以访问它。

## 实体类

**package com.shiro.admin.pojo**;  
  
**import java.util.**List;  
 **public class Permission {  
 private Integer roleId**;**//角色id  
 private Integer menuId**;**//菜单id  
 private String url**;**//菜单url  
 private** List**<Integer> roleIds**;**//角色id集合，多个角色可以访问同一个url  
 //getter/setter方法略**

**}**

mybatis-config.xml中配置别名，略

## PermissionMapper.xml

在PermissionMapper.xml中加入如下代码：

<resultMap id="permissionMap" type="Permission">  
 <id property="menuId" column="MENU\_ID"/>  
 <result property="url" column="URL"/>  
 <!--当前菜单都有哪些角色可以访问-->  
 <collection property="roleIds" column="ROLE\_ID" ofType="Integer">  
 <constructor>  
 <arg column="ROLE\_ID"/>  
 </constructor>  
 </collection>  
</resultMap>  
  
<select id="getAllRolePermission" resultMap="permissionMap">  
 select M.ID MENU\_ID, M.URL, P.ROLE\_ID  
 from MENU M  
 left join ROLE\_PERMISSION P on M.ID=P.MENU\_ID  
</select>

## PermissionDAO

List**<Permission>** getAllRolePermission() **throws Exception**;

## RoleService接口和实现类

RoleService：

**package com.shiro.admin.service**;  
  
**import com.shiro.admin.pojo.Permission**;  
  
**import java.util.**List;  
 **public interface** RoleService **{** List**<Permission>** getAllRolePermission();

}

实现类RoleServiceImpl

**package com.shiro.admin.service.impl**;  
  
**import com.shiro.admin.dao.**PermissionDAO;  
**import com.shiro.admin.pojo.Permission**;  
**import com.shiro.admin.pojo.Role**;  
**import com.shiro.admin.service.**RoleService;  
**import org.springframework.beans.factory.annotation.Autowired**;  
**import org.springframework.stereotype.Service**;  
  
**import java.util.**List;  
 **@Service  
public class RoleServiceImpl implements** RoleService **{  
  
 @Autowired  
 private** PermissionDAO **permissionDAO**;  
 **@Override  
 public** List**<Permission>** getAllRolePermission()**{  
 return permissionDAO**.**getAllRolePermission**();  
 **}**

**}**

## 修改MyShiroFilterFactoryBean类

在3.6中创建的MyShiroFilterFactoryBean中，修改setFilterChainDefinitions方法，加入从数据库中动态读取的菜单和角色信息，完整的MyShiroFilterFactoryBean如下：

**package com.shiro.admin.filter**;  
  
**import com.shiro.admin.pojo.Permission**;  
**import com.shiro.admin.service.**RoleService;  
**import org.apache.shiro.config.Ini**;  
**import org.apache.shiro.spring.web.ShiroFilterFactoryBean**;  
**import org.apache.shiro.util.CollectionUtils**;  
**import org.apache.shiro.web.config.IniFilterChainResolverFactory**;  
**import org.springframework.beans.factory.annotation.Autowired**;  
  
**import java.text.MessageFormat**;  
**import java.util.**List;  
  
**public class MyShiroFilterFactoryBean extends ShiroFilterFactoryBean {  
 public static final String *ROLE\_STRING* = "roles[{0}]"**;  
 **public static String** filterChainDefinitions **= ""**;  
  
 **@Autowired  
 private** RoleService **roleService**;  
  
 **//这个方法在项目启动的时候执行  
 @Override  
 public void** setFilterChainDefinitions(**String definitions**) **{  
 try {** filterChainDefinitions **= definitions**;**// 记录配置的静态过滤链  
  
 Ini ini = new** Ini();  
 **ini**.load(filterChainDefinitions);  
 **Ini**.**Section section = ini**.getSection(**IniFilterChainResolverFactory**.***URLS***);  
 **if** (**CollectionUtils**.isEmpty(**section**)) **{  
 section = ini**.getSection(**Ini**.***DEFAULT\_SECTION\_NAME***);  
 **}  
 //读取所有角色对应的权限** List**<Permission> rolePermission = roleService**.**getAllRolePermission**();  
 **if** (**rolePermission != null**) **{  
 for** (**Permission permission : rolePermission**) **{  
 StringBuilder permFilters = new** StringBuilder();  
 List**<Integer> roleIds = permission**.getRoleIds();  
 **if** (**roleIds != null && roleIds.size()>0 && permission**.getUrl() **!= null && permission**.getUrl().trim().length() **> 0**) **{  
 for** (**Integer roleId : roleIds**) **{  
 //多个角色用逗号分隔  
 permFilters**.append(**roleId**).append(**","**);  
 **}**

**//去掉最后一个逗号  
 String str = permFilters**.substring(**0**, **permFilters**.length() **- 1**);  
 **//url里不能带参数，所以要从?开始截掉后面的参数  
 //最终的结果是等同于xx.html=roles[role1,role2,role3]  
 section**.put(**permission**.getUrl().substring(**0**, **permission**.getUrl().indexOf(**"?"**)),  
 **MessageFormat**.format(***ROLE\_STRING***, **str**));  
 **}  
 }  
 }  
 section**.put(**"/\*\*"**, **"authc"**);**//最后写这句，所有的路径都要验证** setFilterChainDefinitionMap(**section**);  
 **} catch** (**Exception e**) **{  
 e**.printStackTrace();  
 **}  
 }  
}**

## 重写Shiro数据源Ream

修改3.5中创建的MyShiroReam中doGetAuthorizationInfo方法。用户每次点击Url都会调用这个方法进行权限验证。url的访问权限通过角色控制，所以要查出当前用户拥有的所有角色。

RoleMapper.xml

*<?*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="com.shiro.admin.dao.RoleDAO">  
 <select id="getUserRoleIds" parameterType="Integer" resultType="Integer">  
 select ROLE\_ID from USER\_ROLE where USER\_ID=#{userId}  
 </select>

</mapper>

RoleDAO

List**<Integer>** getUserRoleIds(**Integer userId**);

RoleService

List**<Integer>** getUserRoleIds(**Integer userId**);

RoleServiceImpl

**@Autowired  
private** RoleDAO **roleDAO**;

**@Override  
public** List**<Integer>** getUserRoleIds(**Integer userId**) **{  
 return roleDAO**.**getUserRoleIds**(**userId**);  
**}**

MyShiroReam，修改doGetAuthorizationInfo方法：

**@Autowired  
private** RoleService **roleService**;  
**/\*\*  
 \* 每次请求url都会调用这个方法  
 \*/  
@Override  
protected** AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection **principalCollection**) **{  
 //登录之后用户的信息存放在了Shiro的session中，可以直接拿信息** Session **session =** SecurityUtils.getSubject().**getSession**();  
 **AdminUser user =** (**AdminUser**) **session**.**getAttribute**(Constants.***SESSION\_USER\_KEY***);  
 **if** (**user != null**) **{  
 try {  
 //从session中获取用户id，避免多次访问数据库  
 SimpleAuthorizationInfo info =** (**SimpleAuthorizationInfo**) **session**.**getAttribute**(Constants.***SESSION\_USER\_ROLE***);  
 **if** (**info == null**) **{//如果session中没有，直接从数据库中查  
 info = new** SimpleAuthorizationInfo();

List**<Integer> roleIds = roleService**.**getUserRoleIds**(**user**.getId());  
 **if** (**roleIds != null && !roleIds**.**isEmpty**()) **{  
 for** (**Integer roleId : roleIds**) **{**  
 **info**.addRole(**roleId**.toString());  
 **}  
 }  
 //将权限资源添加到用户信息中  
 session**.**setAttribute**(Constants.***SESSION\_USER\_ROLE***, **info**);  
 **}  
 return info**;  
 **} catch** (**Exception e**) **{  
 e**.printStackTrace();  
 **}  
 }  
 return null**;  
**}**

## 重写Shiro验证过滤器

Shiro的权限过滤器，比如xx.html=roles[role1,role2]，要求用户必须同时拥有role1和role2权限。但一般我们的需求是用户拥有role1或者role2其中任何一个权限就可以访问xx.html，要实现这个功能，需要重写一个shiro的过滤器：

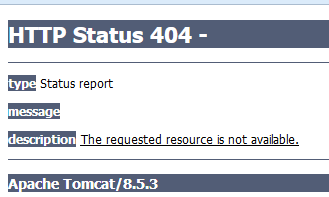
**package com.shiro.admin.filter**;  
  
**import org.apache.shiro.subject.**Subject;  
**import org.apache.shiro.web.filter.authz.**AuthorizationFilter;  
  
**import javax.servlet.**ServletRequest;  
**import javax.servlet.**ServletResponse;  
  
**public class MyShiroRoleFilter extends** AuthorizationFilter **{  
 @Override  
 protected boolean** isAccessAllowed  
 (ServletRequest **req**, ServletResponse **resp**, **Object mappedValue**) **throws Exception {** Subject **subject =** getSubject(**req**, **resp**);  
 **//MyShiroFilterFactoryBean的setFilterChainDefinitions方法里，  
 //把每个菜单对应的角色都加载了  
 //所以这里的roleArray就是当前可以访问这个url的所有角色  
 String**[] **roleArray =** (**String**[]) **mappedValue**;  
 **if** (**roleArray == null || roleArray**.**length == 0**) **{  
 return true**;  
 **}  
 for** (**int i = 0**; **i < roleArray**.**length**; **i++**) **{  
 //只要用户由任意一个角色，就可以访问成功  
 if** (**subject**.**hasRole**(**roleArray**[**i**])) **{  
 return true**;  
 **}  
 }  
 return false**;  
 **}  
}**

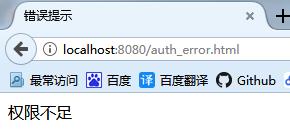
修改3.8中spring-shiro.xml里的id=”shiroFilter”的bean，追加一个filters属性：

<property name="filters">  
 <map>  
 <entry key="roles">  
 <bean class="com.shiro.admin.filter.MyShiroRoleFilter"/>  
 </entry>  
 </map>  
</property>

## 测试

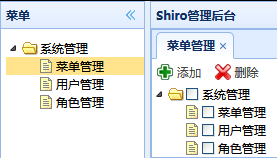
因为菜单中对应的用户管理、角色管理、菜单管理等列表功能没有实现，所以现在有权限的用户访问时，会看到404页面，而没有权限的用户访问时，会看到权限不足的错误提示页：





# 菜单管理功能

## 菜单列表



编辑菜单之前要显示菜单列表，如上图所示。

### MenuMapper.xml

*<?*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="com.shiro.admin.dao.MenuDAO">  
 <resultMap id="menuMap" type="TreeNode" extends="com.shiro.admin.dao.PermissionDAO.menuMap">  
 </resultMap>  
  
 <select id="getAllMenu" resultMap="menuMap">  
 select ID, NAME, PARENT\_ID, URL   
 from MENU  
 </select>

</mapper>

### MenuDAO

List**<TreeNode>** getAllMenu();

### MenuService接口和实现类

MenuService

List**<TreeNode>** getMenuTree();

MenuServiceImpl

**@Autowired  
private** MenuDAO **menuDAO**;

**@Override  
public** List**<TreeNode>** getAllMenu() **{** List**<TreeNode> allMenu = new** ArrayList**<>**();  
 **//编辑菜单的时候，如果添加一级菜单，需要将父菜单选成根节点  
 TreeNode root = new** TreeNode();  
 **root**.setId(**1**);  
 **root**.setText(**"根节点"**);  
 **allMenu**.**add**(**root**);  
 List**<TreeNode> menus = menuDAO**.**getAllMenu**();  
 **allMenu**.**addAll**(makeTree(**menus**));  
 **return allMenu**;  
**}**

### MenuController

**@Controller  
@RequestMapping**(**"/system/menu.html"**)  
**public class MenuController implements** Constants **{  
  
 @Autowired  
 private** MenuService **menuService**;  
  
 **//去菜单列表页  
 @RequestMapping**(**params = "act=list"**)  
 **public ModelAndView** list()**{  
 return new** ModelAndView(**"/pages/system/menu\_list.jsp"**);  
 **}  
 //加载菜单列表  
 @RequestMapping**(**params = "act=tree"**)  
 **@ResponseBody  
 public** List**<TreeNode>** getAllMenu()  
 **throws Exception {  
 return menuService**.**getMenuTree**();  
 **}**

**}**

### 页面

页面参见参考代码/pages/system/menu\_list.jsp

关键代码：

<ul id="menuList" class="easyui-tree"></ul>

<script type="text/javascript">  
 var ***menuTree***;  
 $(function () {  
 ***menuTree*** = $('#menuList').tree({  
 url: "/system/menu.html?act=tree", //远程加载tree数据的url路径  
 animate: true, //是否开启动画效果  
 checkbox: true, //是否显示复选框  
 cascadeCheck: false,  
 onClick: function (node) {  
 $("#menu\_id").val(node.id);  
 $("#menu\_name").textbox("setValue", node.text);  
 $("#url").textbox("setValue", node.attributes.url);  
 //赋值父级菜单  
 $("#parentId").combotree('setValue', node.attributes.parentId);  
 $("#editMenuDialog").dialog("open");  
 },  
 onLoadSuccess: function (node, data) {  
 //列表中不显示根节点  
 var root = $(this).tree("find", 1);  
 $(this).tree('remove', root.target);  
 }  
 });  
 });

</script>

## 添加和修改菜单

### 实体类

**package com.shiro.admin.pojo**;  
  
**public class Menu {  
 private Integer id**;**//菜单  
 private String name**;**//菜单名  
 private Integer parentId**;**//父菜单  
 private String url**;**//菜单的连接  
  
 //getter/setter方法略**

**}**

mybatis-config.xml中配置别名略。

### MenuMapper.xml

<insert id="addMenu" parameterType="Menu" useGeneratedKeys="true" keyProperty="id">  
 insert into MENU (NAME, PARENT\_ID, URL)  
 values (#{name}, #{parentId}, #{url})  
</insert>

<update id="updateMenu" parameterType="Menu">  
 update MENU  
 <set>  
 <if test="url != null">URL = #{url},</if>  
 <if test="name != null">NAME = #{name},</if>  
 <if test="parentId != null">PARENT\_ID = #{parentId},</if>  
 </set>  
 where ID = #{id}  
</update>

### MenuDAO

**void** addMenu(**Menu menu**);  
  
**void** updateMenu(**Menu menu**);

### MenuService接口和实现类

MenuService

**void** addMenu(**Menu menu**);

**void** updateMenu(**Menu menu**);

MenuServiceImpl

**@Override  
public void** addMenu(**Menu menu**) **{  
 menuDAO**.**addMenu**(**menu**);  
**}  
  
@Override  
public void** updateMenu(**Menu menu**) **{  
 menuDAO**.**updateMenu**(**menu**);  
**}**

### MenuController

**//去菜单列表页  
@RequestMapping**(**params = "act=list"**)  
**public ModelAndView** list() **{  
 return new** ModelAndView(**"/pages/system/menu\_list.jsp"**);  
**}**

**// 编辑菜单  
@RequestMapping**(**params = "act=edit"**)  
**@ResponseBody  
public** Map**<String**, **String>** add(**Menu menu**) **{  
 if** (**menu**.getId() **== null**) **{//如果没有id就添加  
 menuService**.**addMenu**(**menu**);  
 **} else {  
 menuService**.**updateMenu**(**menu**);  
 **}  
 return JsonUtil**.getOkStatusMsg(**"编辑成功"**);  
**}**

### 页面

/pager/system/menu\_list.jsp

<div id="tb" style="height:auto">  
 <a href="javascript:void(0)" class="easyui-linkbutton" data-options="iconCls:'icon-add',plain:true"  
 onclick="*goMenuAdd*()">添加</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" data-options="iconCls:'icon-cancel',plain:true"  
 onclick="*doMenuDel*()">删除</a>  
</div>  
<ul id="menuList" class="easyui-tree"></ul>  
<div id="editMenuDialog" class="easyui-dialog" modal="true" style="resizable:true;modal:true;" title="选择菜单"  
 closed="true">  
 <div style="padding:10px 60px 20px 60px">  
 <form id="menuForm" method="post">  
 <input type="hidden" name="id" id="menu\_id"/>  
 <table cellpadding="5">  
 <tr>  
 <td>菜单名:</td>  
 <td><input class="easyui-textbox" type="text" name="name" id="menu\_name" data-options="required:true"/>  
 </td>  
 </tr>  
 <tr>  
 <td>父节点:</td>  
 <td>  
 <!--使用easyui-combotree创建父菜单下拉列表-->  
 <input class="easyui-combotree" id="parentId" name="parentId"  
 data-options="url:'/system/menu.html?act=tree',method:'post'"  
 style="width:220px;">  
 </td>  
 </tr>  
 <tr>  
 <td>URL:</td>  
 <td><input class="easyui-textbox" type="text" name="url" id="url" size="40"/></td>  
 </tr>  
 </table>  
 </form>  
 <div style="text-align:center;padding:5px">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="*submitMenuForm*()">提交</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="$('#menuForm').form('clear')">重填</a>  
 </div>  
 </div>  
</div>

js脚本：

function *submitMenuForm*() {  
 $('#menuForm').form('submit', {  
 onSubmit: function () {  
 if ($(this).form('enableValidation').form('validate')) {  
 $.ajax({  
 url:"/system/menu.html?act=edit",  
 data:$('#menuForm').serialize(),  
 method:"post",  
 success:function(data){  
 if (data.statusCode != '200') {  
 alert(data.message);  
 }else{  
 $("#editMenuDialog").dialog("close");  
 ***menuTree***.tree('reload');  
 $("#parentId").combotree("tree").tree('reload');  
 }  
  
 }  
 })  
 }  
 return false;  
 }  
 });  
}  
function *goMenuAdd*(){  
 $('#menuForm').form('clear');  
 $("#editMenuDialog").dialog("open");  
}

## 删除菜单

删除的若是父节点，要将当前父节点中的子节点移动到根节点下。删除菜单的同时要删除角色中对应的权限。

### Mapper

MenuMapper.xml

<delete id="deleteMenu" parameterType="Integer">  
 delete from MENU WHERE ID = #{id}  
</delete>  
<!--将当前节点的子菜单全部移动到根节点下-->  
<update id="updateParentId" parameterType="Integer">  
 update MENU set PARENT\_ID=1 where PARENT\_ID=#{parentId}  
</update>

PermissionMapper.xml

<delete id="deleteRolePermissionByMenuId" parameterType="Integer">  
 delete from ROLE\_PERMISSION where MENU\_ID=#{menuId}  
</delete>

### DAO

MenuDAO

**void** deleteMenu(**Integer id**);  
  
**void** updateParentId(**Integer parentId**);

PermissionDAO

**void** deleteRolePermissionByMenuId(**Integer menuId**);

### MenuService接口及实现类

MenuService

实现批量删除：

**void** deleteMenu(**Integer**[] **ids**);

MenuServiceImpl

**@Override  
public void** deleteMenu(**Integer**[] **ids**) **{  
 for** (**Integer id : ids**) **{  
 if** (**id == 1**) **{// 根菜单不能删除  
 return**;  
 **}  
 menuDAO**.**deleteMenu**(**id**);  
 **menuDAO**.**updateParentId**(**id**);**// 当前菜单的子菜单全部转到根菜单下  
 permissionDAO**.**deleteRolePermissionByMenuId**(**id**);**// 删除角色的此项权限  
 }  
}**

### MenuController

**// 删除菜单  
@RequestMapping**(**params = "act=delete"**)  
**@ResponseBody  
public** Map**<String**, **String>** delete(**Integer**[] **ids**) **{  
 menuService**.**deleteMenu**(**ids**);  
 **return JsonUtil**.getOkStatusMsg(**"删除成功"**);  
**}**

### 页面

function *doMenuDel*() {  
 if (!confirm("确定要删除吗？")) {  
 return;  
 }  
 var menuIds = new Array();  
 //将已选中的节点id封装到数组中  
 $(***menuTree***.tree("getChecked")).each(function (i, node) {  
 menuIds[i] = node["id"];  
 });  
 $.ajax({  
 url: '/system/menu.html?act=delete',  
 type: 'POST',  
 data: {ids: menuIds.join(",")},  
 cache: false,  
 success: function (json) {  
 if (json.statusCode != '200') {  
 alert(json.message);  
 } else {  
 ***menuTree***.tree('reload');//刷新列表  
 $("#parentId").combotree("tree").tree('reload');  
 }  
 }  
 });  
 return false;  
}

# 角色管理功能

## 角色列表

### 实体类

Role

**package com.shiro.admin.pojo**;  
 **public class Role {  
 private Integer id**;  
 **private String name**;**// 角色名称  
 private String remark**;**// 备注  
 //getter/setter方法略**

**}**

mybatis-config.xml中配置别名 略。

### RoleMapper.xml

<resultMap type="Role" id="roleResultMap">  
 <id property="id" column="ID"/>  
 <result property="name" column="NAME"/>  
 <result property="remark" column="REMARK"/>  
</resultMap>  
  
<sql id="select-from-role">  
 select ID, NAME, REMARK from ROLE  
</sql>  
  
<sql id="where-from-role">  
 <trim prefix="WHERE" prefixOverrides="AND|OR">  
 <if test="id != null">and ID = #{id}</if>  
 <if test="name != null and name != ''">and NAME like CONCAT('%', #{name},'%')</if>  
 </trim>  
</sql>  
  
<select id="getRoleList" parameterType="Role" resultMap="roleResultMap">  
 <include refid="select-from-role"/>  
 <include refid="where-from-role"/>  
</select>

mybatis-config.xml中配置分页插件：

<plugins>  
 <plugin interceptor="com.github.pagehelper.PageHelper">  
 <property name="dialect" value="mysql"/>  
 <property name="reasonable" value="false"/>  
 </plugin>  
</plugins>

### RoleDAO

List**<Role>** getRoleList(**Role criteria**);

### RoleService接口及实现类

RoleService:

**PageInfo<Role>** getRolePagerModel(**Role criteria**, **Integer page**, **Integer rows**);

RoleServiceImpl:

**@Override  
public PageInfo<Role>** getRolePagerModel(**Role criteria**, **Integer page**, **Integer rows**) **{  
 PageHelper**.startPage(**page**, **rows**, **"id desc"**);  
 List**<Role> roleList = roleDAO**.**getRoleList**(**criteria**);  
 **PageInfo<Role> pager = new** PageInfo**<>**(**roleList**);  
 **return pager**;  
**}**

### RoleController

RoleController

**@Controller  
@RequestMapping**(**"/system/role.html"**)  
**public class RoleController {  
 @Autowired  
 private** RoleService **roleService**;  
  
 **@RequestMapping**(**params = "act=list"**)  
 **public ModelAndView** roleList() **{  
 return new** ModelAndView(**"/pages/system/role\_list.jsp"**);  
 **}  
  
 @RequestMapping**(**params = "act=table"**)  
 **@ResponseBody  
 public** Map**<String**, **Object>** userTable(**Role role**, **Integer page**, **Integer rows**) **{  
 PageInfo<Role> pageInfo = roleService**.**getRolePagerModel**(**role**, **page**, **rows**);  
 **//key不可以改，页面上分页插件需要** Map**<String**, **Object> result = new** HashMap**<>**();  
 **result**.**put**(**"rows"**, **pageInfo**.getList());  
 **result**.**put**(**"total"**, **pageInfo**.getTotal());  
 **return result**;  
 **}  
  
}**

### 页面

/pages/system/role\_list.jsp

<table id="dg\_role" data-options="toolbar: '#tb\_role'">  
 <thead>  
 <tr>  
 <th data-options="field:'id',width:130">ID</th>  
 <th data-options="field:'name',width:130">角色名</th>  
 <th data-options="field:'remark',width:130">备注</th>  
 </tr>  
 </thead>  
</table>  
<div id="tb\_role" style="height:auto">  
 <div style="padding:2px 5px;">  
 <form id="searchForm">  
 id: <input class="easyui-textbox" name="id" id="search-role-id"/>  
 名称: <input class="easyui-textbox" name="name" id="search-role-name">  
 <a href="#" class="easyui-linkbutton" iconCls="icon-search" onclick="*doRoleSearch*()">搜索</a>  
 </form>  
 </div>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-add' "  
 onclick="*doRoleAdd*()">添加</a>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-save' "  
 onclick="*doRoleUpdate*()">编辑</a>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-edit' "  
 onclick="*doRoleAssign*()">分配权限</a>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-cancel' "  
 onclick="*doRoleDel*()">删除</a>  
</div>

<script type="text/javascript">  
 $(function () {  
 $("#dg\_role").datagrid({  
 url: "/system/role.html?act=table",  
 singleSelect: true,  
 pagination: true,  
 pageSize: 5,  
 pageList: [5, 10, 15, 20],  
 queryParams: *formToJson*()  
 })  
 });  
 function *formToJson*(){  
 var param={};  
 param.id=$("#search-role-id").val();  
 param.name=$("#search-role-name").val();  
 return param;  
 }  
 function *doRoleSearch*() {  
 $('#dg\_role').datagrid({ queryParams: *formToJson*() });  
 }  
</script>

运行效果：



## 添加和修改角色

### RoleMapper.xml

<update id="updateRole" parameterType="Role">  
 update ROLE  
 <set>  
 <if test="name != null">NAME = #{name},</if>  
 <if test="remark != null">REMARK = #{remark},</if>  
 </set>  
 where ID = #{id}  
</update>  
  
<delete id="deleteRole" parameterType="Integer">  
 delete from ROLE WHERE ID = #{id}  
</delete>

### RoleDAO

**void** addRole(**Role role**);  
  
**void** updateRole(**Role role**);

### RoleService接口及实现类

RoleService:

**void** addRole(**Role role**);  
  
**void** updateRole(**Role role**);

RoleServiceImpl:

**@Override  
public void** addRole(**Role role**) **{  
 roleDAO**.**addRole**(**role**);  
**}  
  
@Override  
public void** updateRole(**Role role**) **{  
 roleDAO**.**updateRole**(**role**);  
**}**

### RoleController

**@RequestMapping**(**params = "act=edit"**)  
**@ResponseBody  
public** Map**<String**, **String>** edit(**Role role**) **{  
 if** (**role**.getId() **== null**) **{  
 roleService**.**addRole**(**role**);  
 **} else {  
 roleService**.**updateRole**(**role**);  
 **}  
 return JsonUtil**.getOkStatusMsg(**"编辑成功"**);  
**}**

### 页面

/pages/system/role\_list.jsp

<div id="editRoleDialog" class="easyui-dialog" modal="true" title="编辑角色"  
 closed="true">  
 <div style="padding:10px 60px 20px 60px">  
 <form id="roleForm" method="post">  
 <input type="hidden" name="id" id="id"/>  
 <table cellpadding="5">  
 <tr>  
 <td>角色名:</td>  
 <td><input class="easyui-textbox" type="text" name="name" id="name" data-options="required:true"/>  
 </td>  
 </tr>  
 <tr>  
 <td>备注:</td>  
 <td><input class="easyui-textbox" type="text" name="remark" id="remark"  
 data-options="required:true"/></td>  
 </tr>  
 </table>  
 </form>  
 <div style="text-align:center;padding:5px">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="*submitRoleForm*()">提交</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="$('#roleForm').form('clear')">重填</a>  
 </div>  
 </div>  
</div>

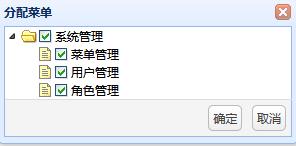
脚本：

function *submitRoleForm*() {  
 $('#roleForm').form('submit', {  
 onSubmit: function () {  
 if ($(this).form('enableValidation').form('validate')) {  
 $.ajax({  
 url: "/system/role.html?act=edit",  
 data: $('#roleForm').serialize(),  
 method: "post",  
 success: function (data) {  
 if (data.statusCode != '200') {  
 alert(data.message);  
 } else {  
 $("#editRoleDialog").dialog("close");  
 $('#dg\_role').datagrid({ queryParams: *formToJson*() });  
 }  
  
 }  
 })  
 }  
 return false;  
 }  
 });  
}  
function *doRoleAdd*() {  
 $('#roleForm').form('clear');  
 $("#editRoleDialog").dialog("open");  
}  
function *doRoleUpdate*() {  
 var row = $('#dg\_role').datagrid('getSelected');  
 $('#roleForm').form('clear');  
 $("#id").val(row.id);  
 $("#name").textbox("setValue", row.name);  
 $("#remark").textbox("setValue", row.remark);  
 $("#editRoleDialog").dialog("open");  
}

## 分配权限

### 获取菜单列表

分配菜单需要加载所有的菜单列表，和当前角色拥有的菜单。



PermissionMapper.xml

<select id="getRolePermissionByRoleId" parameterType="Integer" resultType="Integer">  
 select MENU\_ID from ROLE\_PERMISSION where ROLE\_ID=#{roleId}  
</select>

PermissionDAO:

List**<Integer>** getRolePermissionByRoleId(**Integer roleId**);

RoleService

List**<Integer>** getRoleMenuIds(**Integer roleId**);

RoleServiceImpl

**@Override  
public** List**<Integer>** getRoleMenuIds(**Integer roleId**) **{  
 return permissionDAO**.**getRolePermissionByRoleId**(**roleId**);  
**}**

RoleController:

**@Autowired  
private** MenuService **menuService**;  
  
**//加载所有菜单列表  
@RequestMapping**(**params = "act=menu\_list"**)  
**@ResponseBody  
public** List**<TreeNode>** getAllMenu() **{  
 return menuService**.**getAllMenu**();  
**}  
  
//加载角色拥有的菜单  
@RequestMapping**(**params = "act=role\_menu"**)  
**@ResponseBody  
public** List**<Integer>** getRoleMenu(**Integer roleId**)**{  
 return roleService**.**getRoleMenuIds**(**roleId**);  
**}**

页面：

<div id="menuDialog" class="easyui-dialog" modal="true" title="分配菜单" style="width: 300px;"  
 closed="true" data-options="buttons: '#dlg-buttons'">  
 <ul id="menuListRole" class="easyui-tree"></ul>  
</div>  
<div id="dlg-buttons">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="javascript:*assignMenu*()">确定</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="javascript:$('#menuDialog').dialog('close')">取消</a>  
</div>

脚本：

var ***menuTree***;  
  
function *doRoleAssign*() {  
 var row = $('#dg\_role').datagrid('getSelected');  
 if (row != null && row != "") {  
 //加载所有菜单  
 var resultData = new Array();  
 $.ajax({  
 url: '/system/role.html?act=role\_menu',  
 type: 'POST',  
 data: {roleId: row.id},  
 dataType: "json",  
 cache: false,  
 success: function (json) {  
 resultData = json;  
 //加载菜单树列表  
 ***menuTree*** = $("#menuListRole").tree({  
 url: '/system/role.html?act=menu\_list',  
 method: "post",  
 checkbox: true,  
 onLoadSuccess: function (node, data) {  
 //点击选中用户已有菜单  
 for (var i = 0; i < resultData.length; i++) {  
 var nodeMenu = ***menuTree***.tree("getChildren", ***menuTree***.tree("find", resultData[i]).target);  
 if (nodeMenu.length < 1) {  
 ***menuTree***.tree("check", ***menuTree***.tree("find", resultData[i]).target);  
 }  
 }  
 var defNode = ***menuTree***.tree("find", 1);  
 ***menuTree***.tree('remove', defNode.target);  
 $("#menuDialog").dialog("open");  
 }  
 });  
 }  
 });  
 }  
}

### 分配菜单

分配菜单操作的是角色权限表RPLE\_PERMISSION，这张表是中间表，添加权限的时候要删除角色原有的权限：

PermissionMapper.xml

<delete id="deleteRolePermissionByRoleId" parameterType="Integer">  
 delete from ROLE\_PERMISSION where ROLE\_ID=#{roleId}  
</delete>  
  
<insert id="addRolePermission" parameterType="map">  
 insert into ROLE\_PERMISSION (ROLE\_ID,MENU\_ID)  
 values (#{roleId},#{menuId})  
</insert>

PermissionDAO：

**void** deleteRolePermissionByRoleId(**Integer roleId**);  
  
**void** addRolePermission(Map**<String**, **Integer> param**);

RoleService:

**void** addRolePermission(**Integer roleId**, **Integer**[] **menuIds**);

RoleServiceImpl：

**@Autowired  
private MyShiroFilterFactoryBean shiroFilterFactoryBean**;  
  
**@Override  
public void** addRolePermission(**Integer roleId**, **Integer**[] **menuIds**) **{  
 permissionDAO**.**deleteRolePermissionByRoleId**(**roleId**);  
 Map**<String**, **Integer> param = new** HashMap**<String**, **Integer>**();  
 **param**.**put**(**"roleId"**, **roleId**);  
 **for** (**Integer id : menuIds**) **{  
 param**.**put**(**"menuId"**, **id**);  
 **permissionDAO**.**addRolePermission**(**param**);  
 **}  
 shiroFilterFactoryBean**.update();**//更新整个shiro中所有的url与role的对应信息  
}**

因为shiro是在tomcat启动的时候加载所有角色对应的权限，所以当角色权限变动的时候，需要清除原有的权限，加载新权限。否则用户登录后权限不会更新，必须重启tomcat。

在MyShiroFilterFactoryBean中添加update方法：

**public void** update()**{  
 synchronized** (**this**) **{ //强制同步，控制线程安全  
 try {** AbstractShiroFilter **shiroFilter =** (AbstractShiroFilter) **this**.getObject();  
 **PathMatchingFilterChainResolver resolver =** (**PathMatchingFilterChainResolver**) **shiroFilter** .getFilterChainResolver();  
 **// 过滤管理器  
 DefaultFilterChainManager manager =** (**DefaultFilterChainManager**) **resolver**.getFilterChainManager();  
 **// 清除权限配置  
 manager**.getFilterChains().**clear**();  
 **this**.getFilterChainDefinitionMap().**clear**();  
 **// 重新设置权限  
 this**.setFilterChainDefinitions(**MyShiroFilterFactoryBean**.filterChainDefinitions);**//传入配置中的filterchains** Map**<String**, **String> chains = this**.getFilterChainDefinitionMap();  
 **//重新生成过滤链  
 if** (**!CollectionUtils**.isEmpty(**chains**)) **{  
 for** (Map.Entry**<String**, **String> chain : chains**.**entrySet**()) **{  
 manager**.createChain(**chain**.**getKey**(), **chain**.**getValue**().replace(**" "**, **""**));  
 **}  
 }  
 } catch** (**Exception e**) **{  
 e**.printStackTrace();  
 **}  
 }  
}**

RoleController

**@RequestMapping**(**params = "act=assign\_menu"**)  
**@ResponseBody  
public** Map**<String**, **String>** assignRoleMenu(**Integer roleId**, **Integer**[] **menuIds**) **throws Exception {  
 roleService**.**addRolePermission**(**roleId**, **menuIds**);  
 **return JsonUtil**.getOkStatusMsg(**"分配权限成功"**);  
**}**

页面：

function *assignMenu*(){  
 var row = $('#dg\_role').datagrid('getSelected');  
 var userIds = new Array();  
 $(***menuTree***.tree("getChecked")).each(function (i, node) {  
 userIds[i] = node["id"];  
 });  
 var index = userIds.length;  
 $(***menuTree***.tree('getChecked', 'indeterminate')).each(function (i, node) {//半选中状态  
 userIds[i + index] = node["id"];  
 });  
 $.ajax({  
 url: '/system/role.html?act=assign\_menu',  
 type: 'POST',  
 data: {menuIds: userIds.join(","), roleId: row["id"]},  
 cache: false,  
 success: function (data) {  
 if (data.statusCode != '200') {  
 alert(data.message);  
 } else {  
 $("#menuDialog").dialog("close");  
 }  
 }  
 });  
 return false;  
}

## 删除角色

### RoleMapper.xml

RoleMapper.xml

<delete id="deleteRole" parameterType="Integer">  
 delete from ROLE WHERE ID = #{id}  
</delete>  
  
<delete id="deleteUserRoleByRoleId" parameterType="Integer">  
 delete from USER\_ROLE where ROLE\_ID=#{roleId}  
</delete>

### RoleDAO

**void** deleteRole(**Integer id**);  
  
**void** deleteUserRoleByRoleId(**Integer roleId**);

### RoleService接口和实现类

RoleService：

**void** deleteRole(**Integer id**);

RoleServiceImpl：

**@Override  
public void** deleteRole(**Integer id**) **{  
 roleDAO**.**deleteRole**(**id**);**//删除角色  
 roleDAO**.**deleteUserRoleByRoleId**(**id**);**//删除用户角色  
 permissionDAO**.**deleteRolePermissionByRoleId**(**id**);**//删除角色权限  
}**

### RoleController

**@RequestMapping**(**params = "act=delete"**)  
**@ResponseBody  
public** Map**<String**, **String>** delete(**Integer id**) **throws Exception {  
 roleService**.**deleteRole**(**id**);  
 **return JsonUtil**.getOkStatusMsg(**"删除成功"**);  
**}**

### 页面

function *doRoleDel*() {  
 var row = $('#dg\_role').datagrid('getSelected');  
 if (row != null && row != "") {  
 if (!confirm("确定要删除吗？")) {  
 return;  
 }  
 $.ajax({  
 url: '/system/role.html?act=delete',  
 type: 'POST',  
 data: {id: row.id},  
 success: function (data) {  
 if (data.statusCode != '200') {  
 alert(data.message);  
 } else {  
 $('#dg\_role').datagrid({queryParams: *formToJson*()});  
 }  
 }  
 });  
 }  
 return false;  
}

# 用户管理功能

## 用户列表

### AdminUserMapper.xml

<select id="getAdminUsers" parameterType="AdminUser" resultMap="adminUserMap">  
 <include refid="select-from-adminuser"/>  
 <include refid="where-from-adminuser"/>  
</select>

### AdminUserDAO

List**<AdminUser>** getAdminUsers(**AdminUser criteria**) ;

### AdminUserService接口及实现类

AdminUserService:

**PageInfo<AdminUser>** getAdminUsers(**AdminUser criteria**, **int page**, **int pageSize**);

AdminUserServiceImpl:

**@Override  
public PageInfo<AdminUser>** getAdminUsers(**AdminUser criteria**, **int page**, **int pageSize**) **{  
 PageHelper**.startPage(**page**, **pageSize**,**"id desc"**);  
 List**<AdminUser> userList = adminUserDAO**.**getAdminUsers**(**criteria**);  
 **PageInfo<AdminUser> pager = new** PageInfo**<>**(**userList**);  
 **return pager**;  
**}**

### AdminUserController

**package com.shiro.admin.controller**;  
  
**import com.github.pagehelper.PageInfo**;  
**import com.shiro.admin.pojo.AdminUser**;  
**import com.shiro.admin.service.**AdminUserService;  
**import org.springframework.beans.factory.annotation.Autowired**;  
**import org.springframework.stereotype.Controller**;  
**import org.springframework.web.bind.annotation.RequestMapping**;  
**import org.springframework.web.bind.annotation.ResponseBody**;  
  
**import java.util.HashMap**;  
**import java.util.**Map;  
  
**@Controller  
@RequestMapping**(**"/system/adminuser.html"**)  
**public class AdminUserController {  
 @Autowired  
 private** AdminUserService **adminUserService**;  
  
 **@RequestMapping**(**params = "act=list"**)  
 **public String** list() **{  
 return "/pages/system/admin\_user\_list.jsp"**;  
 **}  
  
 @RequestMapping**(**params = "act=table"**)  
 **@ResponseBody  
 public** Map**<String**, **Object>** userTable(**AdminUser user**, **Integer page**, **Integer rows**) **{  
 PageInfo<AdminUser> pageInfo = adminUserService**.**getAdminUsers**(**user**, **page**, **rows**);  
 Map**<String**, **Object> result = new** HashMap**<>**();  
 **result**.**put**(**"rows"**, **pageInfo**.getList());  
 **result**.**put**(**"total"**, **pageInfo**.getTotal());  
 **return result**;  
 **}  
}**

### 页面

/pages/system/admin\_user\_list.jsp

<table id="dg\_user" data-options="toolbar: '#tb\_user'">  
 <thead>  
 <tr>  
 <th data-options="field:'id',width:100">ID</th>  
 <th data-options="field:'email',width:200">用户名</th>  
 <th data-options="field:'realName',width:130">真实姓名</th>  
 <th data-options="field:'status',width:130,formatter:formatStatus">状态</th>  
 </tr>  
 </thead>  
</table>  
<div id="tb\_user" style="height:auto">  
 <div style="padding:2px 5px;">  
 <form id="searchForm">  
 id: <input class="easyui-textbox" id="search-id"/>  
 用户名: <input class="easyui-textbox" id="search-email">  
 真实姓名：<input class="easyui-textbox" id="search-realName">  
 状态：  
 <select style="width:90px" id="search-status" >  
 <option value="">状态</option>  
 <option value="1">有效</option>  
 <option value="0">无效</option>  
 </select>  
 <a href="#" class="easyui-linkbutton" iconCls="icon-search" onclick="*doSearch*()">搜索</a>  
 </form>  
 </div>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-add'"  
 onclick="*doUserAdd*()">添加</a>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-save'"  
 onclick="*doUserUpdate*()">编辑</a>  
 <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-edit'"  
 onclick="*doUserAssign*()">分配角色</a>  
</div>  
  
<script type="text/javascript">  
 function *formatStatus*(val,row){  
 if (val ==0){  
 return '<span style="color:#ff0200;">无效</span>';  
 } else {  
 return '<span style="color:#00cf4d;">有效</span>';  
 }  
 }  
 $(function () {  
 $("#dg\_user").datagrid({  
 url: "/system/adminuser.html?act=table",  
 singleSelect: true,  
 pagination: true,  
 pageSize: 5,  
 pageList: [5, 10, 15, 20],  
 queryParams: *formToJson*()  
 })  
 });  
  
 function *formToJson*() {  
 var param = {};  
 param.id = $("#search-id").val();  
 param.email = $("#search-email").val();  
 param.realName = $("#search-realName").val();  
 param.status = $("#search-status").val();  
 return param;  
 }  
  
 function *doSearch*() {  
 $('#dg\_user').datagrid({queryParams: *formToJson*()});  
 }  
</script>

## 添加和修改用户

### AdminUserMapper.xml

AdminUserMapper.xml

<insert id="addAdminUser" parameterType="AdminUser" useGeneratedKeys="true" keyProperty="id">  
 insert into ADMIN\_USER (PASSWORD,EMAIL,REAL\_NAME,STATUS)  
 values(#{password}, #{email}, #{realName},#{status});  
</insert>  
  
<update id="updateAdminUser" parameterType="AdminUser">  
 update ADMIN\_USER  
 <set>  
 <if test="password != null and password!=''">  
 PASSWORD = #{password},  
 </if>  
 <if test="realName != null and realName!=''">  
 REAL\_NAME = #{realName},  
 </if>  
 <if test="status != null">  
 STATUS = #{status},  
 </if>  
 </set>  
 where ID = #{id}  
</update>

### AdminUserDAO

AdminUserDAO:

**void** addAdminUser(**AdminUser user**);  
  
**void** updateAdminUser(**AdminUser user**);

### AdminUserService接口及实现类

AdminUserService:

**void** addAdminUser(**AdminUser user**) **throws Exception**;  
  
**void** updateAdminUser(**AdminUser user**);

AdminUserServiceImpl:

**@Override  
public void** addAdminUser(**AdminUser user**) **throws Exception {  
 AdminUser count = adminUserDAO**.**getUserByEmail**(**user**.getEmail());  
 **if** (**count != null**) **{  
 throw new** Exception(**"该邮箱已注册"**);  
 **}  
 adminUserDAO**.**addAdminUser**(**user**);  
**}  
  
@Override  
public void** updateAdminUser(**AdminUser user**) **{  
 adminUserDAO**.**updateAdminUser**(**user**);  
**}**

### AdminUserController

**@RequestMapping**(**params = "act=edit"**)  
**@ResponseBody  
public** Map**<String**, **String>** editUser(**AdminUser adminUser**) **throws Exception {  
 if**(**!**StringUtils.isEmpty(**adminUser**.getPassword()))**{  
 adminUser**.setPassword(**Funcs**.MD5(**adminUser**.getPassword() **+** Constants.***PASSWORD\_SALT\_KEY***));  
 **}  
 if** (**adminUser**.getId() **== null**) **{  
 adminUserService**.**addAdminUser**(**adminUser**);  
 **}else{  
 adminUserService**.**updateAdminUser**(**adminUser**);  
 **}  
 return JsonUtil**.getOkStatusMsg(**"编辑成功"**);  
**}**

### 页面

<div id="editDialog" class="easyui-dialog" modal="true" title="编辑角色" closed="true">  
 <div style="padding:10px 60px 20px 60px">  
 <form id="userForm" method="post">  
 <input type="hidden" name="id" id="id"/>  
 <table cellpadding="5">  
 <tr>  
 <td>邮箱:</td>  
 <td><input class="easyui-textbox" type="text" name="email" id="email" data-options="required:true,validType:'email'"/>  
 </td>  
 </tr>  
 <tr>  
 <td>真实姓名:</td>  
 <td><input class="easyui-textbox" type="text" name="realName" id="realName" data-options="required:true"/></td>  
 </tr>  
 <tr>  
 <td>密码</td>  
 <td><input class="easyui-textbox" type="password" name="password" id="password"/></td>  
 </tr>  
 <tr>  
 <td>状态</td>  
 <td>  
 <select style="width:90px" id="status" name="status">  
 <option value="1" selected="selected">有效</option>  
 <option value="0">无效</option>  
 </select>  
 </td>  
 </tr>  
 </table>  
 </form>  
 <div style="text-align:center;padding:5px">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="*submitForm*()">提交</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="$('#userForm').form('clear')">重填</a>  
 </div>  
 </div>  
</div>

脚本：

function *doUserAdd*() {  
 $('#userForm').form('clear');  
 $("#email").textbox('textbox').attr('readonly',false);  
 $("#editDialog").dialog("open");  
}  
  
function *doUserUpdate*() {  
 var row = $('#dg\_user').datagrid('getSelected');  
 $('#userForm').form('clear');  
 $("#id").val(row.id);  
 $("#email").textbox("setValue", row.email);  
 $("#email").textbox('textbox').attr('readonly',true);  
 $("#realName").textbox("setValue", row.realName);  
 $("#status").val(row.status);  
 $("#editDialog").dialog("open");  
}  
  
function *submitForm*() {  
 $('#userForm').form('submit', {  
 onSubmit: function () {  
 if ($(this).form('enableValidation').form('validate')) {  
 $.ajax({  
 url: "/system/adminuser.html?act=edit",  
 data: $('#userForm').serialize(),  
 method: "post",  
 success: function (data) {  
 if (data.statusCode != '200') {  
 alert(data.message);  
 } else {  
 $("#editDialog").dialog("close");  
 $('#dg\_user').datagrid({queryParams: *formToJson*()});  
 }  
 }  
 })  
 }  
 return false;  
 }  
 });  
}

## 分配角色

### 获取角色列表

RoleService:

List**<Role>** getRoleList(**Role criteria**);

RoleServiceImpl:

**@Override  
public** List**<Role>** getRoleList(**Role criteria**)**{  
 return roleDAO**.**getRoleList**(**criteria**);  
**}**

RoleController:

**@Autowired  
private** RoleService **roleService**;  
  
**//加载所有角色  
@RequestMapping**(**params = "act=role\_list"**)  
**@ResponseBody  
public** List**<**Map**<String**, **Object>>** getAllRoles() **{** List**<**Map**<String**, **Object>> list = new** ArrayList**<>**();  
 List**<Role> roles = roleService**.**getRoleList**(**null**);  
 **for** (**Role role : roles**) **{** Map**<String**, **Object> tep = new** HashMap**<>**();  
 **tep**.**put**(**"text"**, **role**.getName());  
 **tep**.**put**(**"id"**, **role**.getId());  
 **list**.**add**(**tep**);  
 **}  
 return list**;  
**}  
  
//加载被编辑的用户角色  
@RequestMapping**(**params = "act=user\_role"**)  
**@ResponseBody  
public** List**<Integer>** getUserRole(**Integer userId**) **{** List**<Integer> ids = roleService**.**getUserRoleIds**(**userId**);  
 **return ids**;  
**}**

页面：

<div id="roleDialog" class="easyui-dialog" modal="true" title="分配角色" style="width: 300px;"  
 closed="true" data-options="buttons: '#dlg-buttons'">  
 <ul id="roleList" class="easyui-tree"></ul>  
</div>  
<div id="dlg-buttons">  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="javascript:*assignRole*()">确定</a>  
 <a href="javascript:void(0)" class="easyui-linkbutton" onclick="javascript:$('#roleDialog').dialog('close')">取消</a>  
</div>

var ***roleTree***;  
  
function *doAssign*() {  
 var row = $('#dg\_user').datagrid('getSelected');  
 if (row != null && row != "") {  
 //加载所有菜单  
 var resultData = new Array();  
 $.ajax({  
 url: '/system/adminuser.html?act=user\_role',  
 type: 'POST',  
 data: {userId: row.id},  
 dataType: "json",  
 cache: false,  
 success: function (json) {  
 resultData = json;  
 //加载菜单树列表  
 ***roleTree*** = $("#roleList").tree({  
 url: '/system/adminuser.html?act=role\_list',  
 method: "post",  
 checkbox: true,  
 onLoadSuccess: function (node, data) {  
 //点击选中用户已有角色  
 for (var i = 0; i < resultData.length; i++) {  
 var nodeMenu = ***roleTree***.tree("getChildren", ***roleTree***.tree("find", resultData[i]).target);  
 if (nodeMenu.length < 1) {  
 ***roleTree***.tree("check", ***roleTree***.tree("find", resultData[i]).target);  
 }  
 }  
 $("#roleDialog").dialog("open");  
 }  
 });  
 }  
 });  
 }  
}

### 分配角色

RoleMapper.xml

<insert id="addUserRole" parameterType="map">  
 insert into USER\_ROLE (USER\_ID,ROLE\_ID)  
 values (#{userId},#{roleId})  
</insert>  
  
<delete id="deleteUserRoleByUserId" parameterType="Integer">  
 delete from USER\_ROLE where USER\_ID=#{userId}  
</delete>

RoleDAO:

**void** deleteUserRoleByUserId(**Integer userId**);  
  
**void** addUserRole(Map**<String**, **Integer> param**);

RoleService

**void** addUserRole(**Integer userId**, **Integer**[] **roleIds**);

RoleServiceImpl

**@Override  
public void** addUserRole(**Integer userId**, **Integer**[] **roleIds**)**{** Map**<String**, **Integer> param = new** HashMap**<>**();  
 **param**.**put**(**"userId"**, **userId**);  
 **roleDAO**.**deleteUserRoleByUserId**(**userId**);**//先删除用户原有的角色  
 if** (**roleIds != null**) **{  
 for** (**Integer roleId : roleIds**) **{  
 param**.**put**(**"roleId"**, **roleId**);  
 **roleDAO**.**addUserRole**(**param**);  
 **}  
 }  
}**

AdminUserController

**@RequestMapping**(**params = "act=assign\_role"**)  
**@ResponseBody  
public** Map**<String**, **String>** addUserRole(**Integer userId**, **Integer**[] **roleIds**) **throws Exception {  
 roleService**.**addUserRole**(**userId**, **roleIds**);  
 **return JsonUtil**.getOkStatusMsg(**"授权成功"**);  
**}**

页面：

function *assignRole*() {  
 var row = $('#dg\_user').datagrid('getSelected');  
 var roleIds = new Array();  
 $(***roleTree***.tree("getChecked")).each(function (i, node) {  
 roleIds[i] = node["id"];  
 });  
 $.ajax({  
 url: '/system/adminuser.html?act=assign\_role',  
 type: 'POST',  
 data: {roleIds: roleIds.join(","), userId: row["id"]},  
 cache: false,  
 success: function (data) {  
 if (data.statusCode != '200') {  
 alert(data.message);  
 } else {  
 $("#roleDialog").dialog("close");  
 }  
 }  
 });  
 return false;  
}