SpringBoot整合权限框架Shiro

一、简介

1.1 权限框架

Apache Shiro

Spring Security

Shiro比Spring更容易使用，实现和最重要的理解

Spring Security更加知名的唯一原因是因为品牌名称

“Spring”以简单而闻名，但讽刺的是很多人发现安装Spring Security很难

然而，Spring Security却有更好的社区支持

Apache Shiro在Spring Security处理密码学方面有一个额外的模块

Spring-security 对spring 结合较好，如果项目用的springmvc ，使用起来很方便。但是如果项目中没有用到spring，那就不要考虑它了。

Shiro 功能强大、且 简单、灵活。是Apache 下的项目比较可靠，且不跟任何的框架或者容器绑定，可以独立运行

1.2 Shiro是什么

Apache Shiro 是 Java 的一个安全（权限）框架。

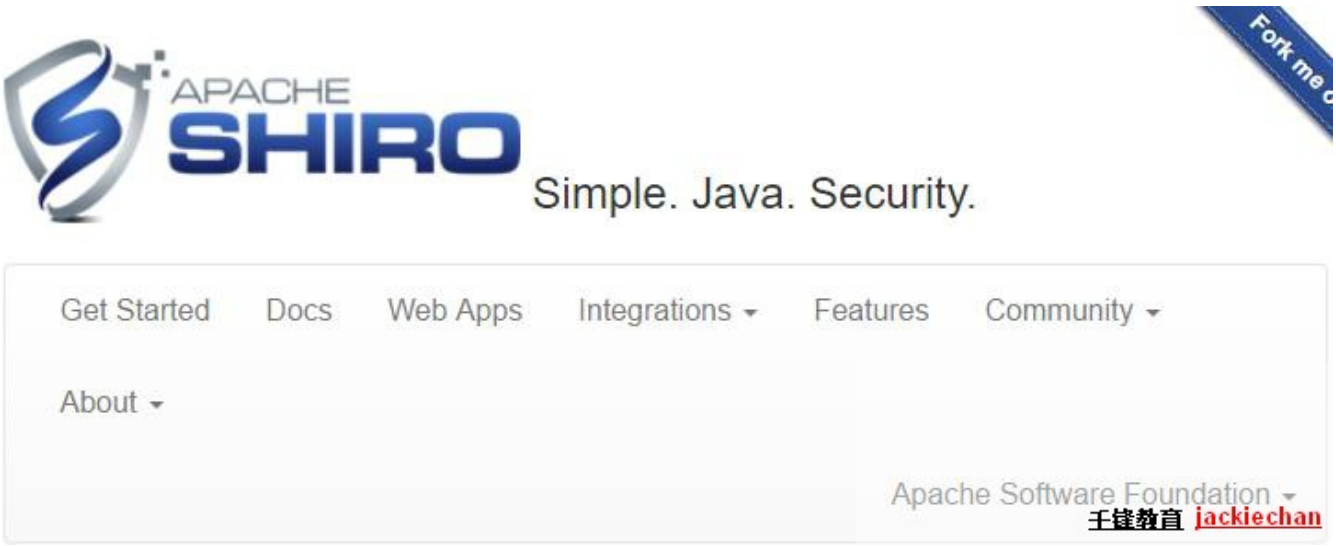
• Shiro 可以非常容易的开发出足够好的应用，其不仅可以用在

JavaSE 环境，也可以用在 JavaEE 环境。

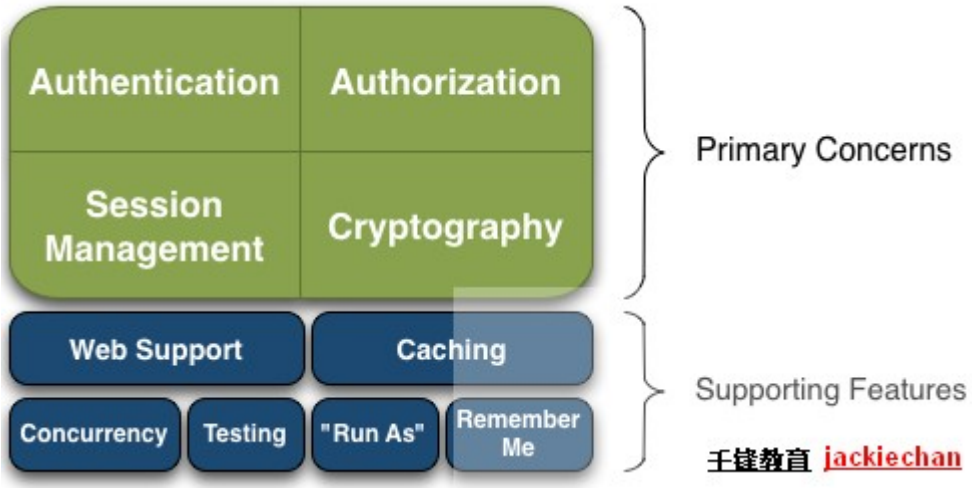
• Shiro 可以完成：认证、授权、加密、会话管理、与Web 集成、缓存

等。

• 下载：http://shiro.apache.org/



1.3 Shiro的功能



Authentication：身份认证/登录，验证用户是不是拥有相应的身份；

• Authorization：授权，即权限验证，验证某个已认证的用户是否拥有某个权限；即判断用户是否能进行什么操作，如：验证某个用户是否拥有某个角色。或者细粒度的验证某个用户对某个资源是否具有某个权限；

• Session Manager：会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中；会话可以是普通 JavaSE 环境，也可以是 Web 环境的；

• Cryptography：加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储；

• Web Support：Web 支持，可以非常容易的集成到Web 环境；

• Caching：缓存，比如用户登录后，其用户信息、拥有的角色/权限不必每次去查，这样可以提高效率；

• Concurrency：Shiro 支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能

• 把权限自动传播过去；

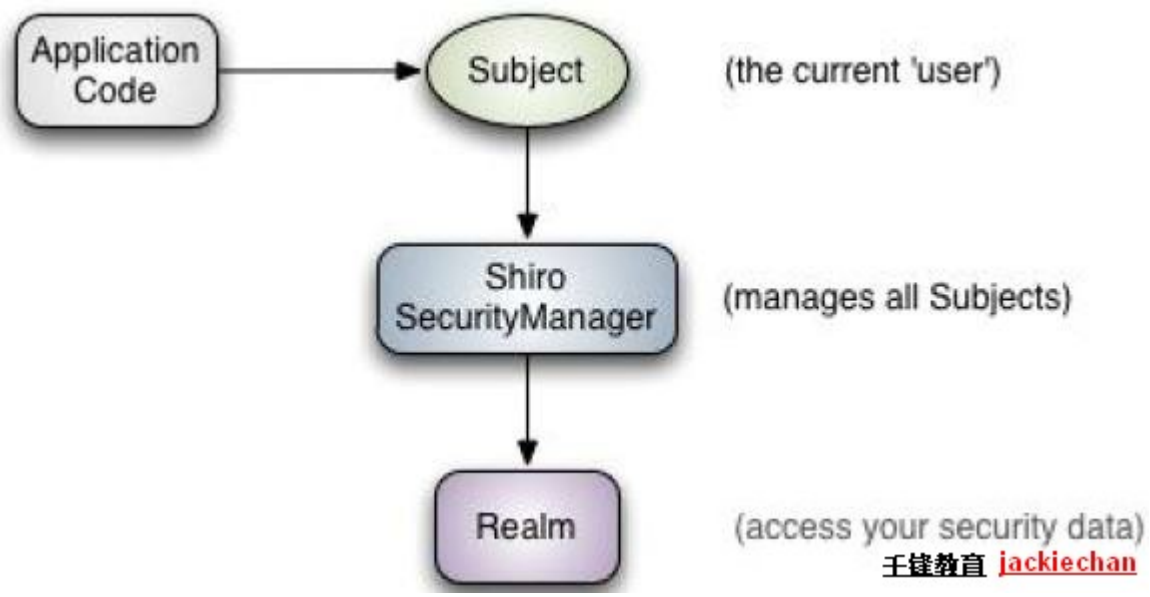
• Testing：提供测试支持；

• Run As：允许一个用户假装为另一个用户（如果他们允许）的身份进行访问；

• Remember Me：记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登

录了

1.5 Shiro的架构组成



Subject：应用代码直接交互的对象是 Subject，也就是说 Shiro 的对外

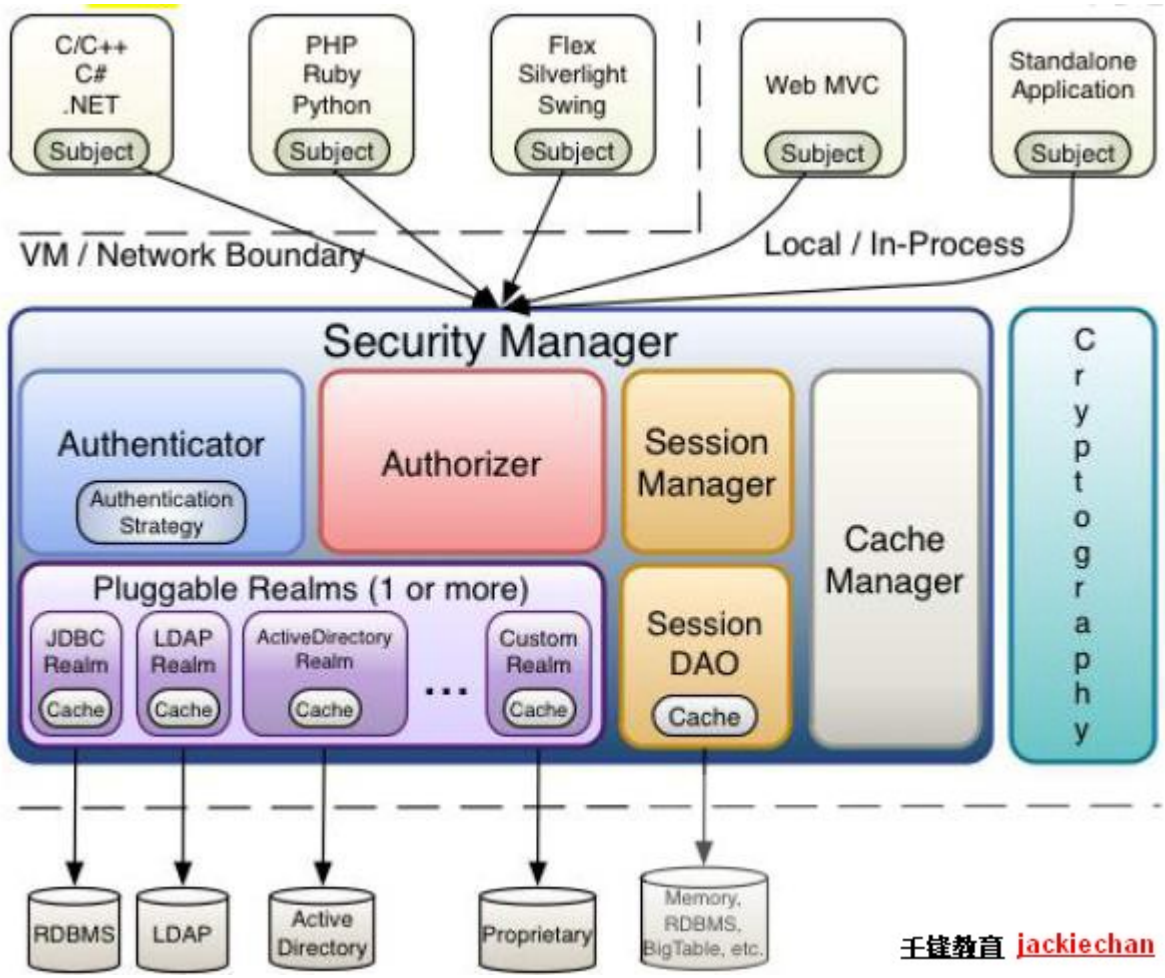
API 核心就是 Subject。Subject 代表了当前“用户”，与 Subject 的所有交互都会委托给 SecurityManager；

• SecurityManager：安全管理器；即所有与安全有关的操作都会与SecurityManager 交互；且其管理着所有Subject；可以看出它是 Shiro的核心，

它负责与 Shiro 的其他组件进行交互，它相当于 SpringMVC 中DispatcherServlet 的角色

• Realm：Shiro 从 Realm 获取安全数据（如用户、角色、权限），就是说SecurityManager 要验证用户身份，那么它需要从 Realm 获取相应的用户进行比较以确定用户身份是否合法；

也需要从 Realm 得到用户相应的角色/权限进行验证用户是否能进行操作；可以把 Realm 看成 DataSource



• Subject：任何可以与应用交互的“用户”；

• SecurityManager ： 相当于SpringMVC 中的 DispatcherServlet；是 Shiro 的心脏；所有具体的交互都通过 SecurityManager 进行控制；它管理着所有 Subject、且负责进行认证、授权、会话及缓存的管理。

• Authenticator：负责 Subject 认证，是一个扩展点，可以自定义实现；可以使用认证策略（Authentication Strategy），即什么情况下算用户认证通过了；

• Authorizer：授权器、即访问控制器，用来决定主体是否有权限进行相应的操作；即控制着用户能访问应用中的哪些功能；

• Realm：可以有 1 个或多个 Realm，可以认为是安全实体数据源，即用于获取安全实体的；可以是JDBC 实现，也可以是内存实现等等；由用户提供；所以一般在应用中都需要实现自己的 Realm；

• SessionManager：管理 Session 生命周期的组件；而 Shiro 并不仅仅可以用在 Web 环境，也可以用在如普通的 JavaSE 环境

• CacheManager：缓存控制器，来管理如用户、角色、权限等的缓存的；因为这些数据基本上很少改变，放到缓存中后可以提高访问的性能

• Cryptography：密码模块，Shiro 提高了一些常见的加密组件用于如密码加密/解密。

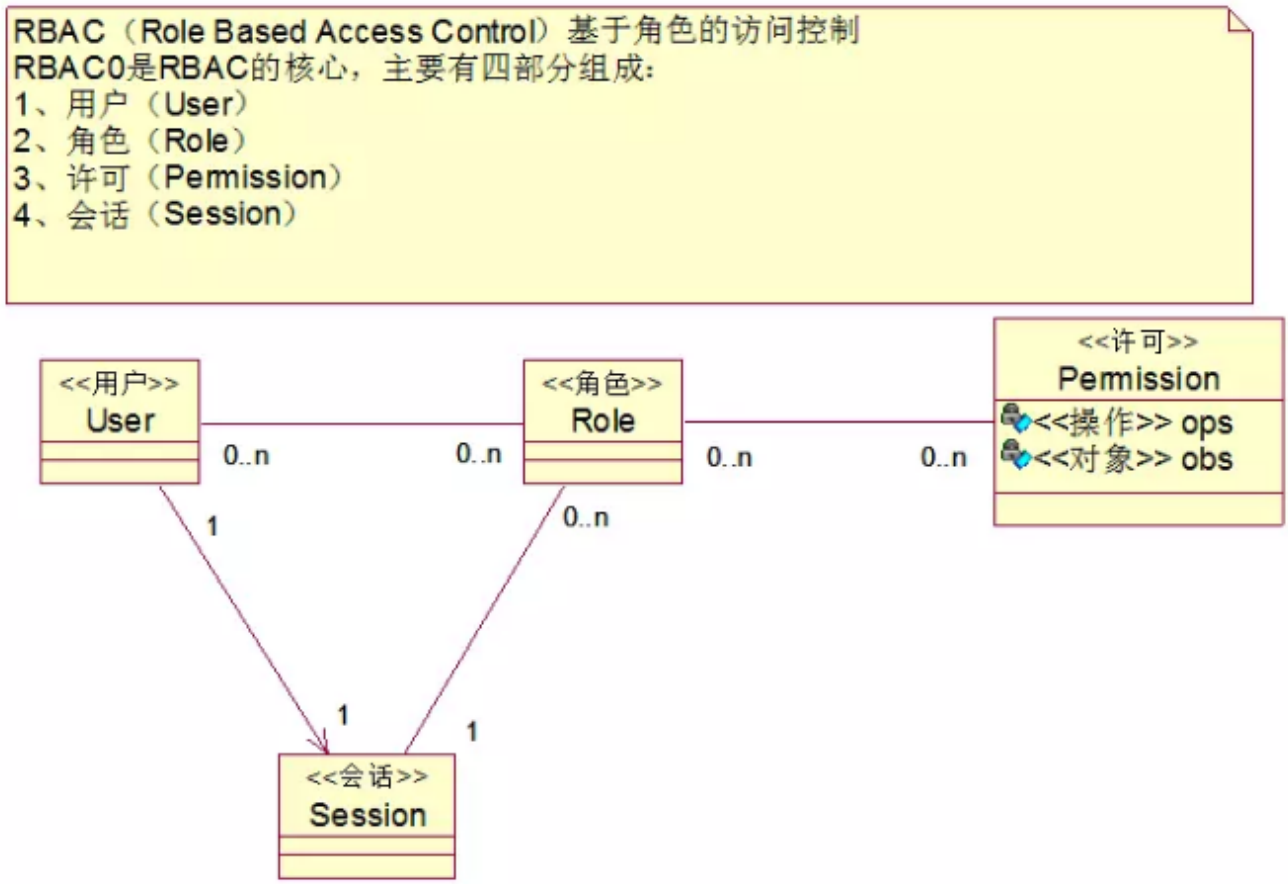
二、RBAC系统

2.1 RBAC简介

RBAC(Role-Based Access Control):基于角色的权限访问控制 Shiro

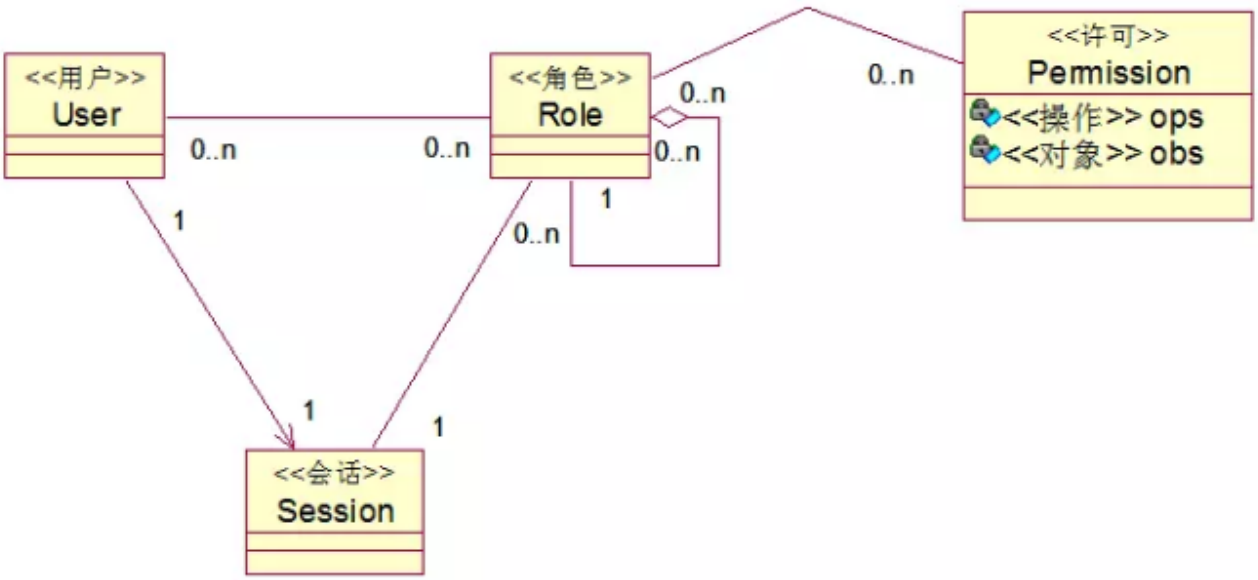
RBAC认为授权实际上是Who 、What 、How 三元组之间的关系，也就是Who 对What 进行How 的操作，也就是“主体”对“客体”的操作。 Who：是权限的拥有者或主体（如：User，Role）。 What：是操作或对象（operation，object）。 How：具体的权限（Privilege,正向授权与负向授权）。

2.2 RBAC核心



RBAC0

RBAC（Role Based Access Control）基于角色的访问控制
RBAC1是对RBAC0进行了扩展，是RBAC的角色分层模型，RBAC1引入了角色继承概念，有了继承就有了上下级的包含关系

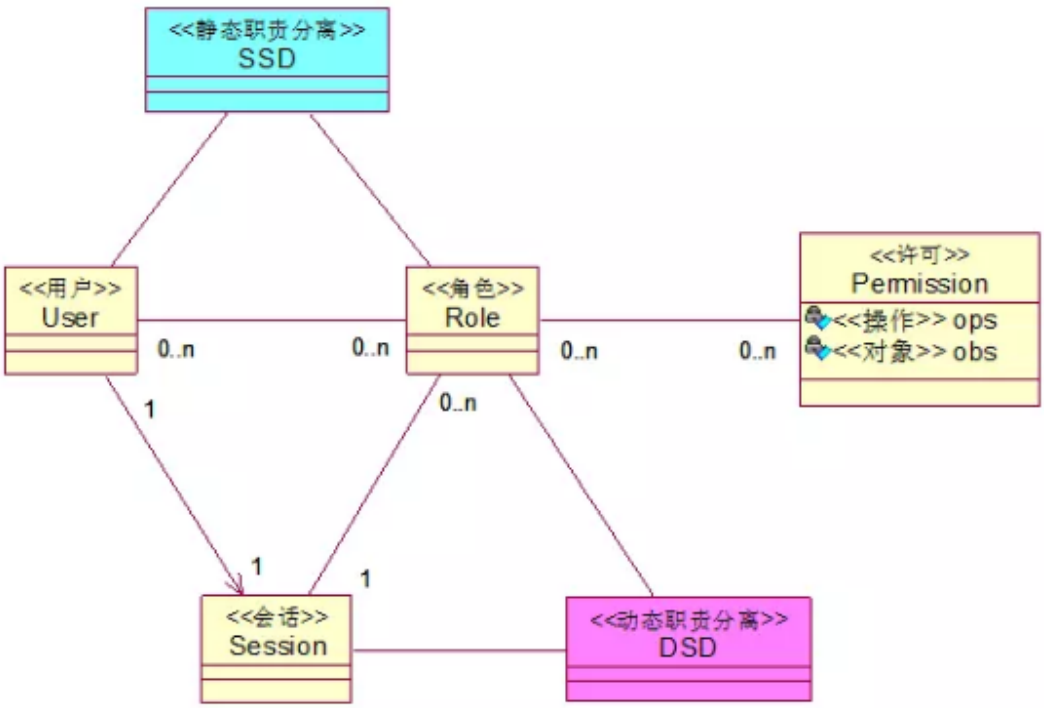


RBAC1

RBAC（Role Based Access Control）基于角色的访问控制
RBAC2是基于RBAC0扩展的，主要引入了SSD（静态职责分离）和DSD（动态职责分离）

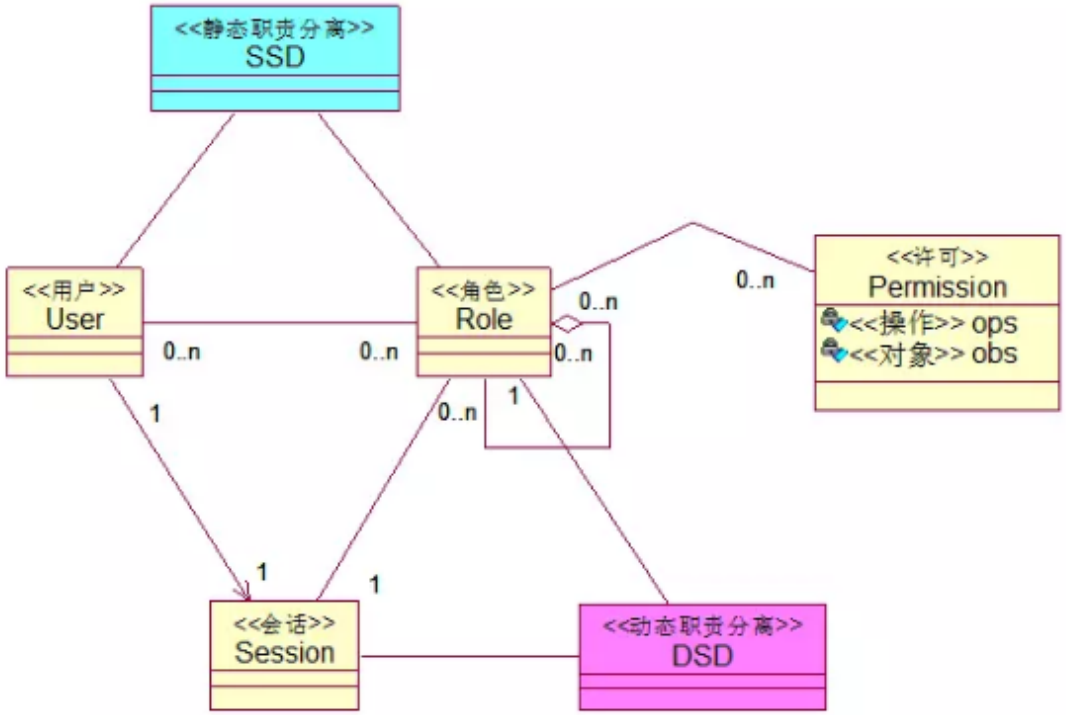
SSD主要应用在用户和角色之间（授权阶段），主要约束：
1、互斥角色，同一个用户不能授予互斥关系的角色，如：不能同时授予会计和出纳的角色
2、基数约束，一个用户拥有的角色是有限的，一个角色拥有的许可是有限的
3、先决条件约束，用户想得到高级权利，必须先拥有低级权利

DSD会话和角色之间的约束，主要动态决定怎么样计划角色，如：一个用户拥有5个角色，只激活2个



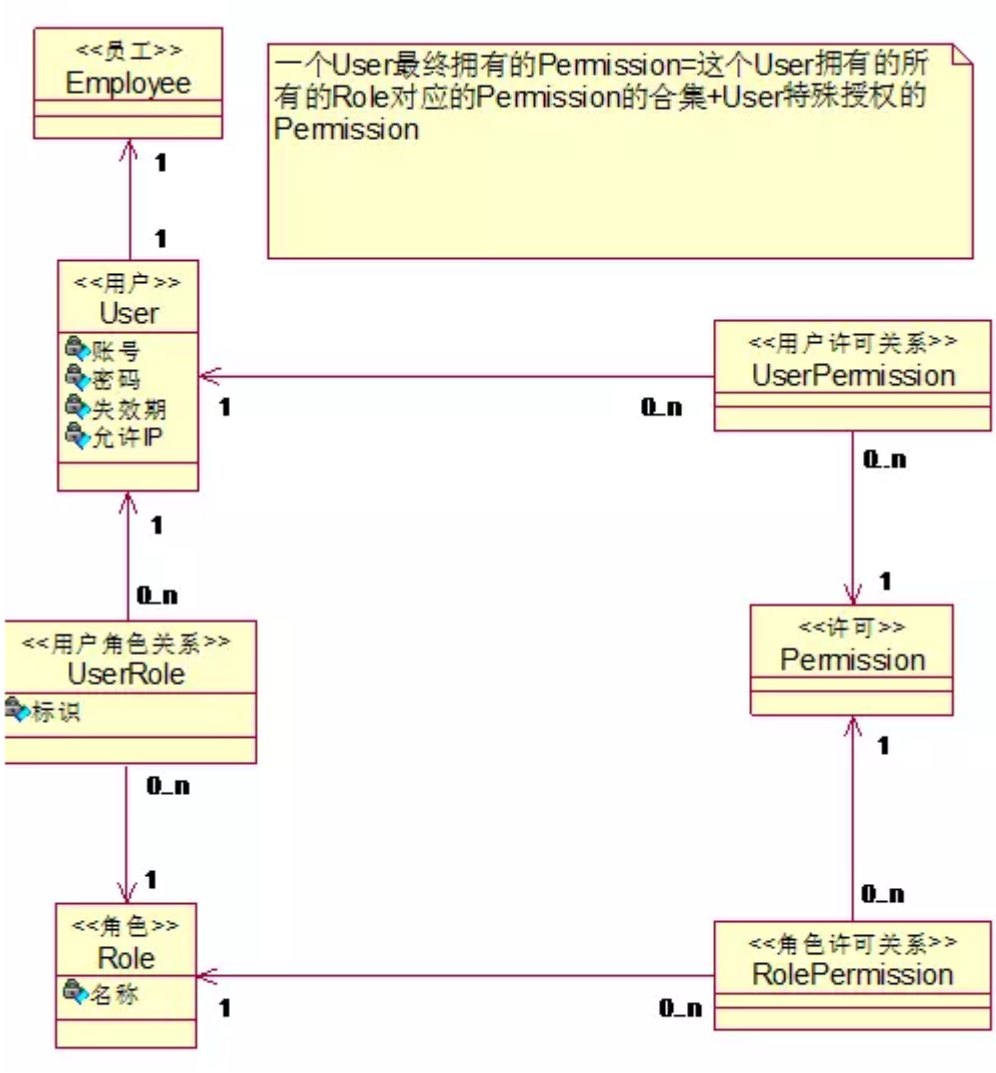
RBAC2

RBAC（Role Based Access Control）基于角色的访问控制
RBAC3=RBAC1+RBAC2



RBAC3

用户表 关联 **用户角色表**，**用户角色表**关联 **角色权限表**



关系类图

2.3 RBAC系统分析

实现权限管理模块，不同的人拥有不同的权限

权限分为2种：

1、粗粒度 动态菜单 不同人看到菜单不一样

2、细粒度 页面中某些操作 比如：增删改查

2.4 数据库设计

1、用户表

id

username

password

flag

ctime

2、角色表

id

name

info

3、用户角色表

id

uid

rid

4、权限表

id

name

icon

parentid

prms 如果是菜单 表示：路径 如果是权限：表示的权限名称

type 类型 1菜单 2权限

level 级别

5、角色权限表

id

rid

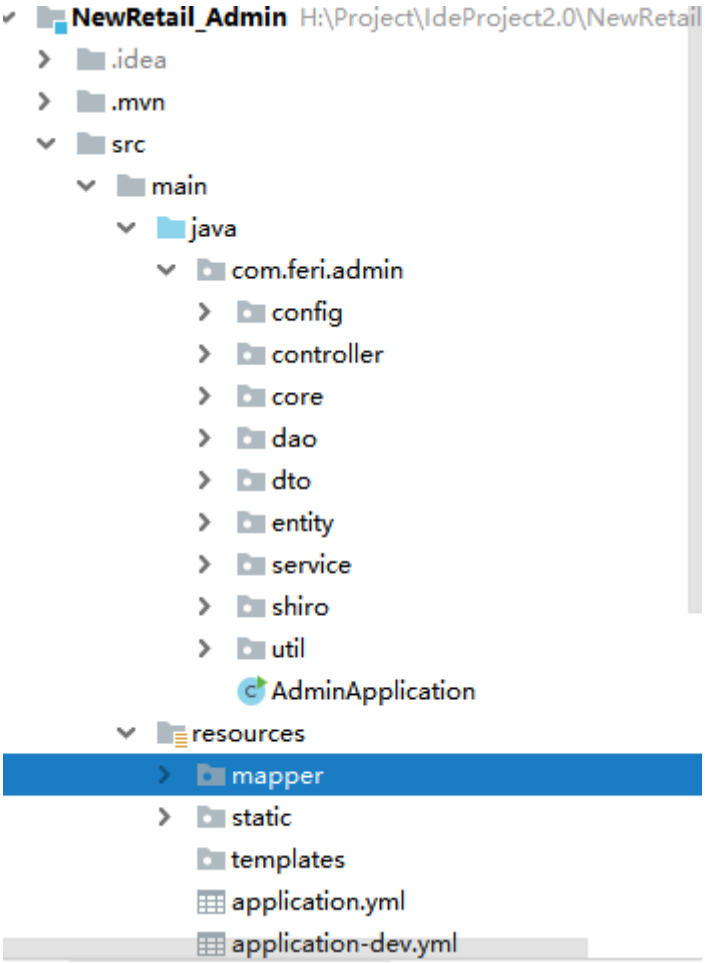
pid

```
#RBAC5表
#1、用户表
create table sys_member(id int primary key AUTO_INCREMENT,mname varchar(20),password varchar(50),ctime
datetime,flag int);
#2、角色表
create table sys_role(id int primary key AUTO_INCREMENT,name varchar(20),info varchar(50));
#3、角色表
create table sys_userrole(id int primary key AUTO_INCREMENT,uid int,rid int);
#4、权限表
create table sys_permission(id int primary key AUTO_INCREMENT,name varchar(20),icon
varchar(30),parentid int
,prms varchar(50),type int,level int);
#5、角色权限表
create table sys_rolepermission(id int primary key AUTO_INCREMENT,rid int,pid int);
```

2.5 项目实现

1、创建项目

SpringBoot类型，项目结构如下：



2、依赖jar

shiro

数据库

druid

swagger

mybatis-plus

pom文件内容如下：

``

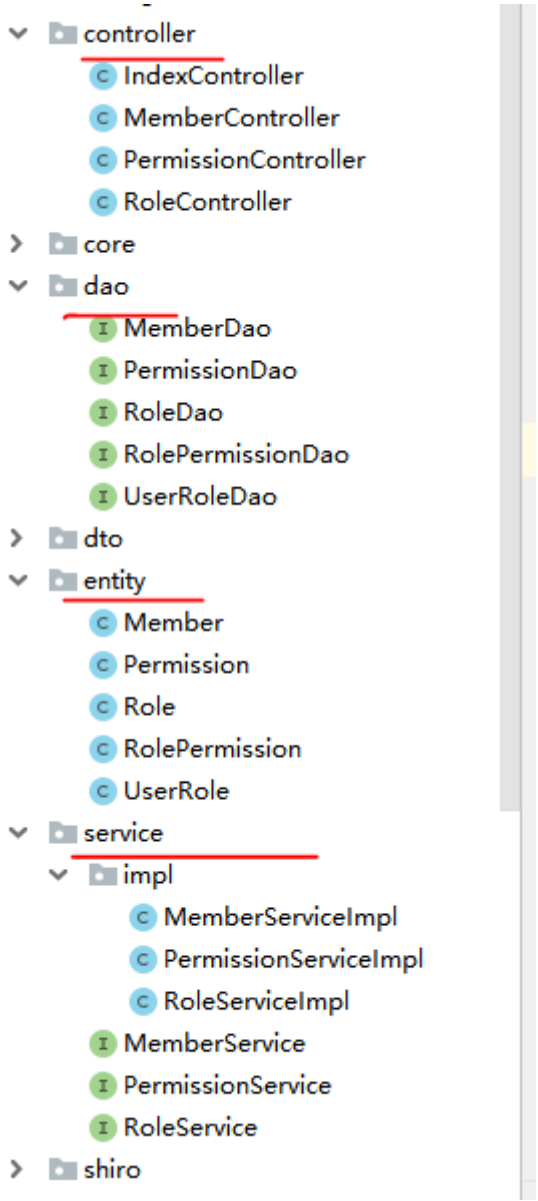
```
<!--1、声明版本号-->
<properties>
   <java.version>1.8</java.version>
   <shirospring.version>1.4.1</shirospring.version>
```

```
    <swagger.version>2.9.2</swagger.version>
    <lombox.version>1.18.8</lombox.version>
    <mybatisplus.version>3.1.2</mybatisplus.version>
    <druid.version>1.1.17</druid.version>
</properties>
<!--2、限定版本-->
<dependencyManagement>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.shiro/shiro-spring -->
        <dependency>
            <groupId>org.apache.shiro</groupId>
            <artifactId>shiro-spring</artifactId>
            <version>${shirospring.version}</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>${druid.version}</version>
        </dependency>
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>${mybatisplus.version}</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>${lombok.version}</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>${swagger.version}</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>${swagger.version}</version>
        </dependency>
    </dependencies>
</dependencyManagement>
<!--3、依赖jar-->
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
```

```
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-test</artifactId>
       <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.shiro/shiro-spring -->
    <dependency>
       <groupId>org.apache.shiro</groupId>
       <artifactId>shiro-spring</artifactId>
    </dependency>
    <dependency>
       <groupId>com.alibaba</groupId>
       <artifactId>druid</artifactId>
    </dependency>
    <dependency>
       <groupId>com.baomidou</groupId>
       <artifactId>mybatis-plus-boot-starter</artifactId>
    </dependency>
    <dependency>
       <groupId>org.projectlombok</groupId>
       <artifactId>lombok</artifactId>
    </dependency>
    <dependency>
       <groupId>io.springfox</groupId>
       <artifactId>springfox-swagger2</artifactId>
    </dependency>
        <!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
        </dependency>
</dependencies>
```

3、搬砖

逆向工程生成代码，然后CV到本项目中



4、整合Shiro

1、依赖jar

shiro-spring

2、自定义Realm

Realm:数据源 提供用户认证和授权的方法

自定义类 并继承AuthorizingRealm

重写2大方法

doGetAuthorizationInfo：

实现当前登录用户的授权

查询当前用户的角色或权限并设置到系统中

doGetAuthenticationInfo：

实现当前用户的登录认证

``

```
import com.feri.admin.entity.Member;
import com.feri.admin.service.PermissionService;
import org.apache.shiro.SecurityUtils;
import org.apache.shiro.authc.*;
import org.apache.shiro.authz.AuthorizationInfo;
import org.apache.shiro.authz.SimpleAuthorizationInfo;
import org.apache.shiro.realm.AuthorizingRealm;
import org.apache.shiro.subject.PrincipalCollection;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

/**
 * @program: NewRetail_Admin
 * @description: 基于Shiro 实现自定义Realm
 * @author: Feri
 * @create: 2019-08-16 11:29
 */
@Service
public class ShiroRealm extends AuthorizingRealm {

    @Autowired
    private PermissionService permissionService;

    //授权 查询用户的权限并设置
    @Override
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection
principalCollection) {
        //1、获取登录信息
        Member member= (Member)
SecurityUtils.getSubject().getSession().getAttribute("user");
        //2、验证用户信息
        if(member!=null){
            //3、查询当前用户的所有的权限 只要权限名称
            List<String> prms=permissionService.queryUidPrms(member.getId());
            //4、设置权限到Shiro中
            SimpleAuthorizationInfo authorizationInfo=new SimpleAuthorizationInfo();
            authorizationInfo.addStringPermissions(prms);
            return authorizationInfo;
        }
        return null;
    }
    //认证 校验用户是否登录成功
    @Override
    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken
authenticationToken) throws AuthenticationException {
        //1、获取传递的令牌
        UsernamePasswordToken token=(UsernamePasswordToken)authenticationToken;
        //2、校验令牌的内容
```

```
        if(token!=null && token.getUsername()!=null){
            //3、生成认证信息
            SimpleAuthenticationInfo authenticationInfo=new
SimpleAuthenticationInfo(token.getUsername(),token.getPassword(),getName());
            return authenticationInfo;
        }
        return null;
    }
}
```

3、实现Shiro的配置

编写配置类 @Configuration

``

```
import com.feri.admin.shiro.ShiroRealm;
import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.apache.shiro.mgt.SecurityManager;
import java.util.LinkedHashMap;

/**
 * @program: NewRetail_Admin
 * @description:
 * @author: Feri
 * @create: 2019-08-16 11:44
 */
@Configuration
public class ShiroConfig {

    //1、创建Shiro管理器对象
    @Bean
    public SecurityManager createSM(ShiroRealm shiroRealm){
        DefaultWebSecurityManager securityManager=new DefaultWebSecurityManager();
        securityManager.setRealm(shiroRealm);
        return securityManager;
    }
    //2、配置Shiro工厂对象
    @Bean
    public ShiroFilterFactoryBean createSFFB(SecurityManager securityManager){
        ShiroFilterFactoryBean shiroFilterFactoryBean=new ShiroFilterFactoryBean();
        shiroFilterFactoryBean.setSecurityManager(securityManager);
        //设置3个页面 登录页 首页 错误页
        shiroFilterFactoryBean.setLoginUrl("/login.html");
        shiroFilterFactoryBean.setSuccessUrl("/index.html");
        shiroFilterFactoryBean.setUnauthorizedUrl("/error.html");
        //设置拦截的接口 哪些需要放行 哪些需要拦截
        //什么样Map集合 可以保证添加顺序
        LinkedHashMap<String,String> hashMap=new LinkedHashMap<>();
        //放行资源
```

```
        hashMap.put("/login.html","anon");
        hashMap.put("/admin/user/login.do","anon");
        hashMap.put("/css/**","anon");
        hashMap.put("/assets/**","anon");
        hashMap.put("/font/**","anon");
        hashMap.put("/upload/**","anon");
        hashMap.put("/js/**","anon");
        hashMap.put("/Widget/**","anon");
        hashMap.put("/images/**","anon");
        hashMap.put("/products/**","anon");
        //拦截资源
        hashMap.put("/**","authc");//全部拦截
        //设置拦截规则
        shiroFilterFactoryBean.setFilterChainDefinitionMap(hashMap);
        return shiroFilterFactoryBean;
    }
}
```

5、实现接口

1、登录接口

完成初始用户

``

```
public R login(String name, String pass) {
    //1、校验账号是否存在
    Member member=getBaseMapper().selectByName(name);
    if(member!=null){
        //2、校验密码是否正确
        if(Objects.equals(member.getPassword(), ShiroPassUtil.md5Pass(pass))){
            //3、Shiro相关操作
            //3、创建主题
            Subject subject= SecurityUtils.getSubject();
            //4、创建令牌
            UsernamePasswordToken token=new UsernamePasswordToken(name,pass);
            //5、发起登录
            subject.login(token);
            subject.getSession().setAttribute("user",member);
            return R.ok("成功");
        }
    }
    return R.failed("用户名或密码不正确");
}
```

2、实现校验权限的接口

``

```
public R checkPerms(String permission) {
    if(SecurityUtils.getSubject().isPermitted(permission)){
        return R.ok("拥有权限");
    }else {
        return R.failed("暂无权限");
    }
}
```

3、注销

```
public R loginout() {
    Subject subject=SecurityUtils.getSubject();
    subject.getSession().removeAttribute("user");
    subject.logout();
    return R.ok("注销成功");
}
```

2.6 前端开发

修改登录页面，进行登录接口请求

```
$.ajax({
                url:"/admin/user/login.do",
                data:$("#fm1").serialize(),
                method:"post",
                success:function(obj){
                    if(obj.code==1){
                         layer.alert('登陆成功！',{
                               title: '登录',
                               icon:1,
                          });
                           location.href="index.html";
                          layer.close(index);
                    }else{
                     layer.alert('登陆失败，'+obj.msg,{
                               title: '登录',
                               icon:1,
                          });
                    }
                }
                })
```

2.7 运行测试


登陆
localhost:9898/login.html
欢迎后台管理界面平台
返回首页
帮助
关于
New Retail
管理员登陆
用户名
密码
验证码
M D w H
保存密码
登陆
通知
本网站系统不再对IE8以下浏览器支持，请见谅。
版权所有 2016-2019 河南小禅院科技有限责任公司


登陆
localhost:9898/index.html
欢迎后台管理界面平台
返回首页
帮助
关于
New Retail
管理员登陆
用户名
密码
验证码
g o 9 6
保存密码
登陆
通知
本网站系统不再对IE8以下浏览器支持，请见谅。
版权所有 2016-2019 河南小禅院科技有限责任公司

三、Shiro总结

3.1 常用过滤器

authc：所有已登陆用户可访问 登录才可以访问

roles：有指定角色的用户可访问，通过[ ]指定具体角色，这里的角色名称与数据库中配置一致

perms：有指定权限的用户可访问，通过[ ]指定具体权限，这里的权限名称与数据库中配置一致

anon：所有用户可访问，通常作为指定页面的静态资源时使用

四、动态菜单

系统的菜单因人而异

4.1 初始数据

```
#初始数据
#1、用户表 管理员 admin 密码：admin
insert into sys_member(mname,password,ctime,flag) values('admin','v/Q7WDs24G/oneuI3VjfMQ==',now(),1);
#2、初始角色
insert into sys_role(name,info) values('管理员','拥有一切权限，严禁泄露');
#3、用户角色表
insert into sys_userrole(uid,rid) values(1,1);
#4、初始菜单数据
#一级菜单
insert into sys_permission(name,icon,parentid,level,type,prms)values
('系统首页','icon-home',-1,1,1,''),
('产品管理','icon-desktop',-1,1,1,''),
('图片管理','icon-picture',-1,1,1,''),
('交易管理','icon-list',-1,1,1,''),
('支付管理','icon-credit-card',-1,1,1,''),
('会员管理','icon-user',-1,1,1,''),
('消息管理','icon-home',-1,1,1,''),
('文章管理','icon-home',-1,1,1,''),
('财务管理','icon-home',-1,1,1,''),
('系统管理','icon-home',-1,1,1,''),
('管理员管理','icon-home',-1,1,1,'');
#初始化二级菜单
insert into sys_permission(name,icon,parentid,level,type,prms)values
(12,'产品列表','',2,'Products_List.html',1,2),(13,'品牌管理','',2,'Brand_Manage.html',1,2),
(14,'分类管理','',2,'Category_Manage.html',1,2),
(15,'广告管理','',3,'advertising.html',1,2),(16,'分类管理','',3,'Sort_ads.html',1,2),
(17,'交易信息','',4,'transaction.html',1,2),(18,'交易订单','',4,'Order_Chart.html',1,2),
(19,'订单管理','',4,'Orderform.html',1,2);

#5、角色权限初始内容
insert into sys_rolepermission(rid,pid) values
(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(1,7),
(1,8),(1,9),(1,10),(1,11),(1,12),(1,13),(1,14),
(1,15),(1,16),(1,17),(1,18),(1,19);
```

4.2 编写接口

实现登录用户的菜单数据的查询

1、编写SQL语句

实现当前用户的菜单数据查询 查询出所有的菜单

```
#查询用户的所有菜单
select p.* from sys_permission p inner join sys_rolepermission rp on p.id=rp.pid
inner join sys_userrole ur on ur.rid=rp.rid where p.type=1 and ur.uid=1 order by p.level,p.id asc;
```

2、实现业务逻辑

``

```
@Data
public class Menu {
    private Permission permission;
    private List<Permission> childs;
    public Menu() {
    }
    public Menu(Permission permission, List<Permission> childs) {
        this.permission = permission;
        this.childs = childs;
    }
}
```

``

```
public R queryMenu() {
    //1、获取登录用户
    Member member= (Member) SecurityUtils.getSubject().getSession().getAttribute("user");
    //2、获取所有菜单
    List<Permission> list=getBaseMapper().selectMenu(member.getId());
    //3、获取一级菜单菜单
    //4、获取一级对应的二级菜单
    List<Menu> menus=new ArrayList<>();
    for(Permission p:list){
        if(p.getLevel()==1){
            //一级菜单
            menus.add(new Menu(p,new ArrayList<>()));
        }else {
            //二级菜单
            //将当前二级菜单添加到一级菜单的集合中
            int index=getPid(menus,p.getParentid());
            if(index!=-1){
                menus.get(index).getChilds().add(p);
            }
        }
    }
    return R.ok(menus);
}
//根据权限的id，找到权限对象
private int getPid( List<Menu> menus,int  pid){
```

```
    for(int i=0;i<menus.size();i++){
        if(menus.get(i).getPermission().getId()==pid){
            return i;
        }
    }
    return -1;
}
```

4.3 页面数据绑定

源码地址：https://github.com/xingpenghui/NewRetail_Admin