SMS权限设计说明

一．**RBAC概念**

[RBAC](http://www.sojson.com/tag_rbac.html)是基于角色的访问控制（Role-Based Access Control ）在[RBAC](http://www.sojson.com/tag_rbac.html)中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限。这就极大地简化了权限的管理。这样管理都是层级相互依赖的，权限赋予给角色，而把角色又赋予用户，这样的权限设计很清楚，管理起来很方便。

二．**权限模型**

如小图为权限模型



模型中有 用户的角色，角色，资源，菜单，操作

用户的角色：将角色分配给用户，用户就具备了角色的权限

角色：拥有权限的抽象对象，谁拥有角色，谁就拥有角色的权限，角色的权限可以随时变更，无需与用户进行关联

资源（权限）：资源是将可以被访问的操作，页面元素等的统称，最终将分配给角色拥有；

菜单：菜单属于资源的细化和界面的菜单进行对应，控制菜单的状态以及分配菜单的到具体的角色上；

操作：操作提现在页面元素中，同时又有uri与后端进行关联，实现权限拦截，操作属于菜单下的某个页面拥有，且菜单拥有某个页面，如果某个菜单被停用则操作也被停用；

以上模型中考虑开发成本，暂时把区域，部门，角色组等等拿掉，采用rbac的基础模型具备了自由扩展的基础

权限控制：

1. 为用户分配权限，其实是为用户选择合适的角色进行分配；分配后，用户的编码与角色的编码将会建立关联关系，这样在登录后就可以通过获取用户的角色来判定用户的权限了；
2. 删除用户的权限，其实是对用户的角色进行删除，或者对某个角色的资源进行屏蔽导致用户登陆后无权限可用，当然权限系统一般都会采用缓存模式进行加速访问，也就是缓存也会被删除，实现立马生效的效果；
3. 创建角色，其实就是给角色起个名字，然后将其他的资源记性关联即可，形成角色编码与菜单编码，操作编码在资源中形成关联；这样查询一个角色有哪些权限其实就有有哪些资源数据，就可以判断页面上是否要显示某些菜单或者按钮了；同时请求时拦截器或者网关层进行token拦截判定是否具备某个接口的访问权限；
4. 创建菜单或者操作，其实是将前端可以看到的ui界面中的菜单进行编码对应，写入到前端代码中，然后配置到菜单中，操作中；菜单，操作中的uri 是控制访问权限拦截使用，具体需要后端的接口uri配合，同时也要配置进去才可以被拦截；

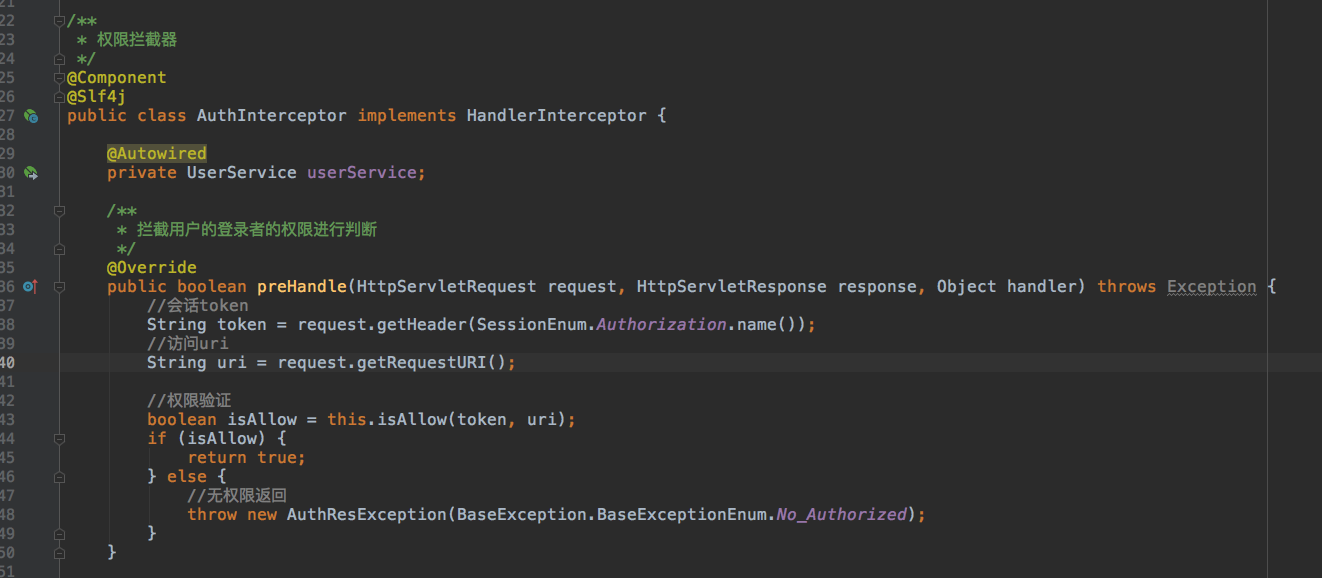
三．**权限与框架结合**

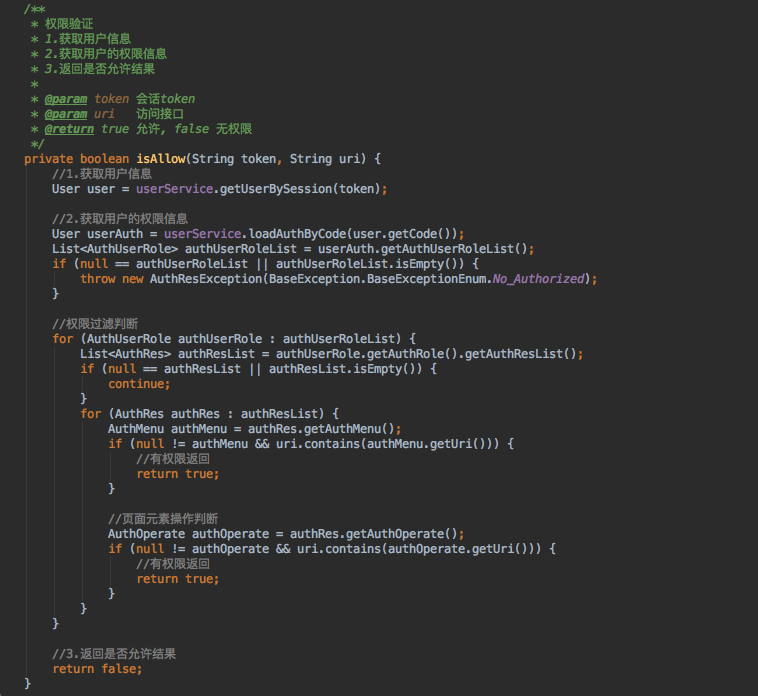
1. **权限实现**

大项目分布式的具备一个网关层就是gateway层，或者单体项目中有拦截器 如springboot 中的interceptor，通过gateway，或者拦截器拦截所有的请求，提取token或者会话信息，宗旨就是找到 访问者是谁，是哪个用户，再根据用户的如：userCode 查询到用户的所有角色，角色对应的资源等等关系，进行判断，是放行，还是拦截驳回；比如访问的是 /order/list 接口，只要用户的角色中有一个操作，且这个操作中有这个uri /order/list，而且是可用状态，则放行访问后端的控制器 OrderController 的 /order/list 接口；否则就直接中断访问，并返回错误提示，如：{msg:无权访问,errorCode:9999} 等；

代码片段：

**以下为权限的全部实现代码**





1. **接口权限控制**

接口权限的控制是和操作，菜单等进行结合使用；主要是采用网关，拦截器层对 uri 如：/order/list 这种接口的拦截判断；首先需要配置菜单，操作 中的uri，将后端需要配置的接口统统配置进去，然后对这些资源建立角色，将角色分配个具体的用户即可；由于请求时有host+uri的方式构成的，拦截了uri就是拦截了权限，放行还是中断访问在于 token或者会话的用户是否具备此权限和此资源；

拦截接口的做法的优势在于，权限和业务代码无关，可以将权限放到网关层，或者拦截器中进行实现，代码无需侵入到任何业务代码中，权限系统亦可以分离出来作为独立的auth系统存在，为所有需要权限的系统微服务，提供权限的支持，一次开发将不再需要开发，只需要配置即可；

代码片段：



**四．前端页面与权限的结合**

权限分为，页面权限，和api接口权限；接口权限就是上面的说明；关于页面权限很多公司都不曾实现，都是靠api接口方式判断实现的一种伪权限；

页面权限是指将页面拆分为，菜单和页面两部分；通过菜单导航到页面，页面中的每一个按钮，组件均可以作为配置化的权限控制；菜单权限分为两种：

1. 菜单全部写死在静态页面上，通过判断控制菜单的显示或者隐藏（静态）；
2. 菜单全部写在后端的数据库中，通过用户的登录查询具体的菜单列表再循环显示到组件中进行显示（动态）；

另种均可以实现，菜单的控制，动态的就是数据库出来的较为简单和权限天然结合；静态的需要做一些改造，将所有的菜单进行编码 任意编码都可以 如：1001，1002， nav\_user,nav\_order 都可以；目的是将菜单通过编码区分开，录入到权限系统中的菜单中，进行保存；如果用户具备 某个菜单的编码，就显示这个菜单，否则就隐藏这个菜单，对于前端可以通过前端的拦截器或者数组集合的方式进行配置将会更灵活方便些；页面的配置类似，需要对每个组件进行编码，通过组件组合成页面，需要在页面上判断，是否有某些组件的权限码，没有就不显示，有了就显示等控制；实现权限的局部控制与全页面控制模式；

结合这样的方式就可以将前端页面的权限和接口权限统一；实现有权限就显示，可以操作，没权限就不显示，即便接口请求也一样被拒绝；

总结：

以上是权限的模型设计以及技术实现的细节说明，这种模式我之前用在其他项目中，非常成熟[www.ponddy.ai](http://www.ponddy.ai) 以及 [www.pi-top.com](http://www.pi-top.com)； 将权限和业务分离，再通过配置的方式进行整合，通过拦截的技术完成权限的过滤，实现权限系统的分离与通用；再多系统也可以配置下即可使用（目前模型需要增加域名和系统的设计即可支持）；对于后续团队中再增加新的系统，只需要增加业务代码即可，无需关心权限的开发；目前是在sms中进行单个项目实例使用，后续需要需要改造后提炼出来，实现一个账户登录所有的系统无需，每个系统一个账户的麻烦；