# RBAC Manual

> 医院信息系统——基于角色的访问控制（RBAC）说明书。
>
> Author: zyc

医院信息系统需要对用户权限进行管理，以保证数据安全性。根据需求与实现的复杂度，HIS使用基于URL和资源对象的 RBAC 权限管理系统。由于 Django 仅支持数据库层面的大粒度权限与权限组，因此基于 `Django.contrib.auth` 模块开发 `HIS.rbac` 模块。

## 基本说明

`HIS.rbac` 包含两个权限系统，分别控制**URL访问**和**数据库特定数据行**的权限。

### 对象资源权限 Object Permission

包括三个方面：数据库对象、行级资源对象和操作类型。即可以对特定数据库的某个数据行做出何种操作，例如能够对内科1号患者数据进行访问（患者数据库 + row: 1 + 访问）。

### URL 访问权限 URL Permission

控制系统用户能够访问的页面。即能否访问特定的URL。

### 用户 User

包括医院职工、大型设备与机器等。由于C端用户（患者）不应该深入使用医院内部的信息系统，因此患者使用另外一套及其简单的权限管理方式。

### 角色 Role

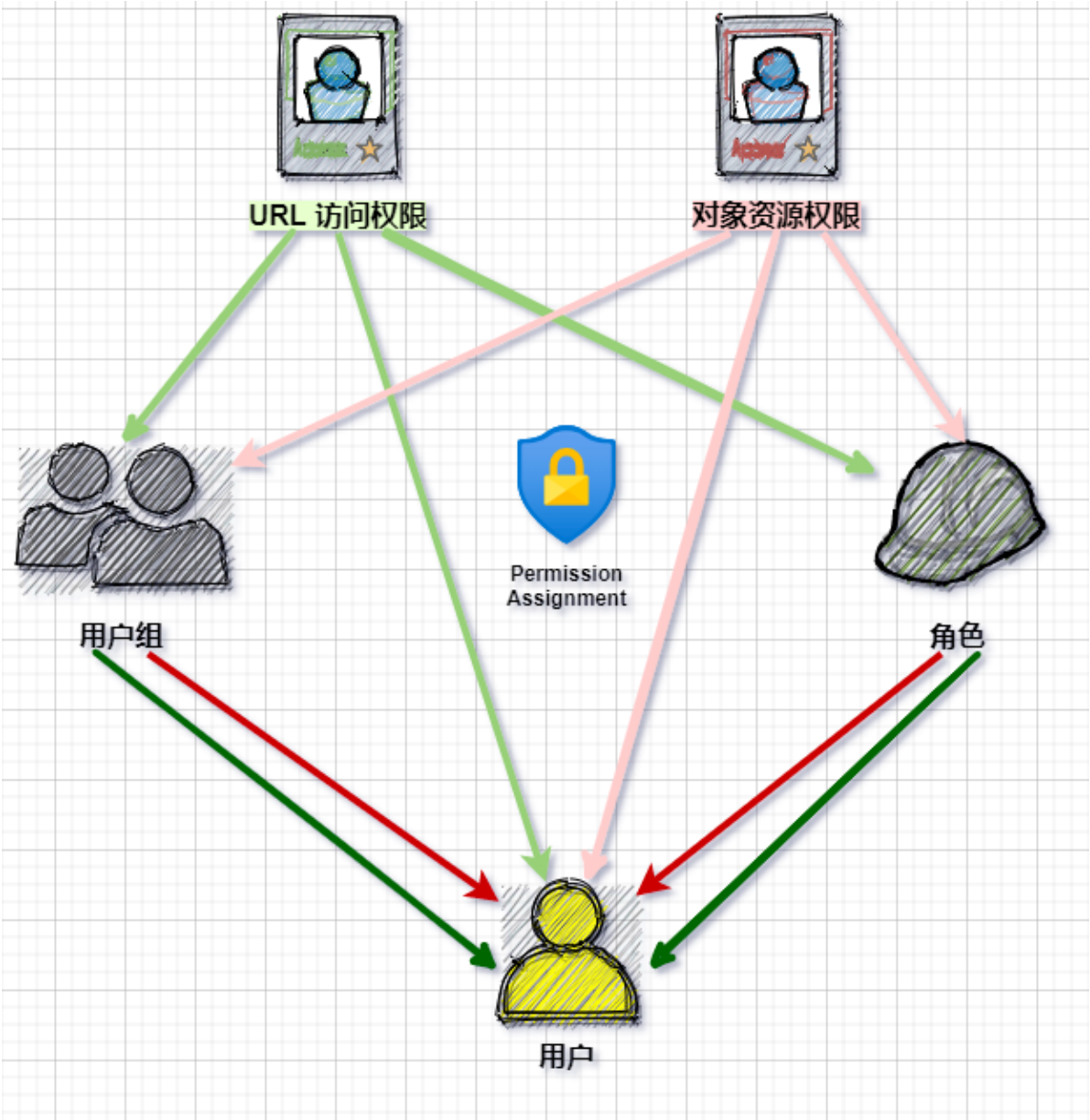
角色作为多个相似权限的集合，用于简化向用户分配权限的过程。一个角色可以拥有多个权限，一个用户可以拥有多个角色。向用户授权时，关联相关角色即可。

### 用户组 UserGroup

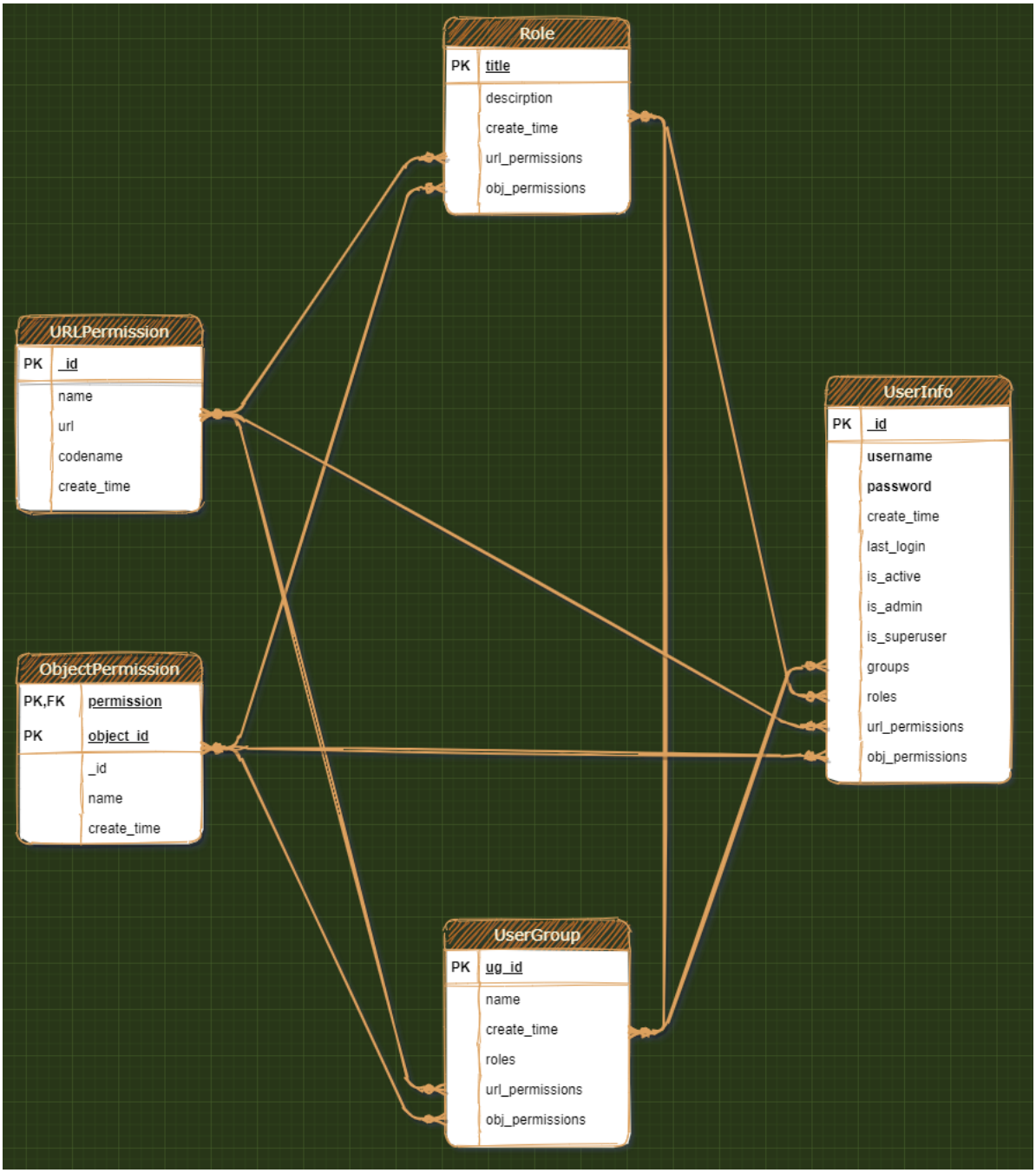
在用户基数很大时，通过将相关角色关联到一个个用户来进行授权会变得十分麻烦。这时使用用户组来汇集具有相同权限和角色特点的用户。

## 权限分配与数据库设计

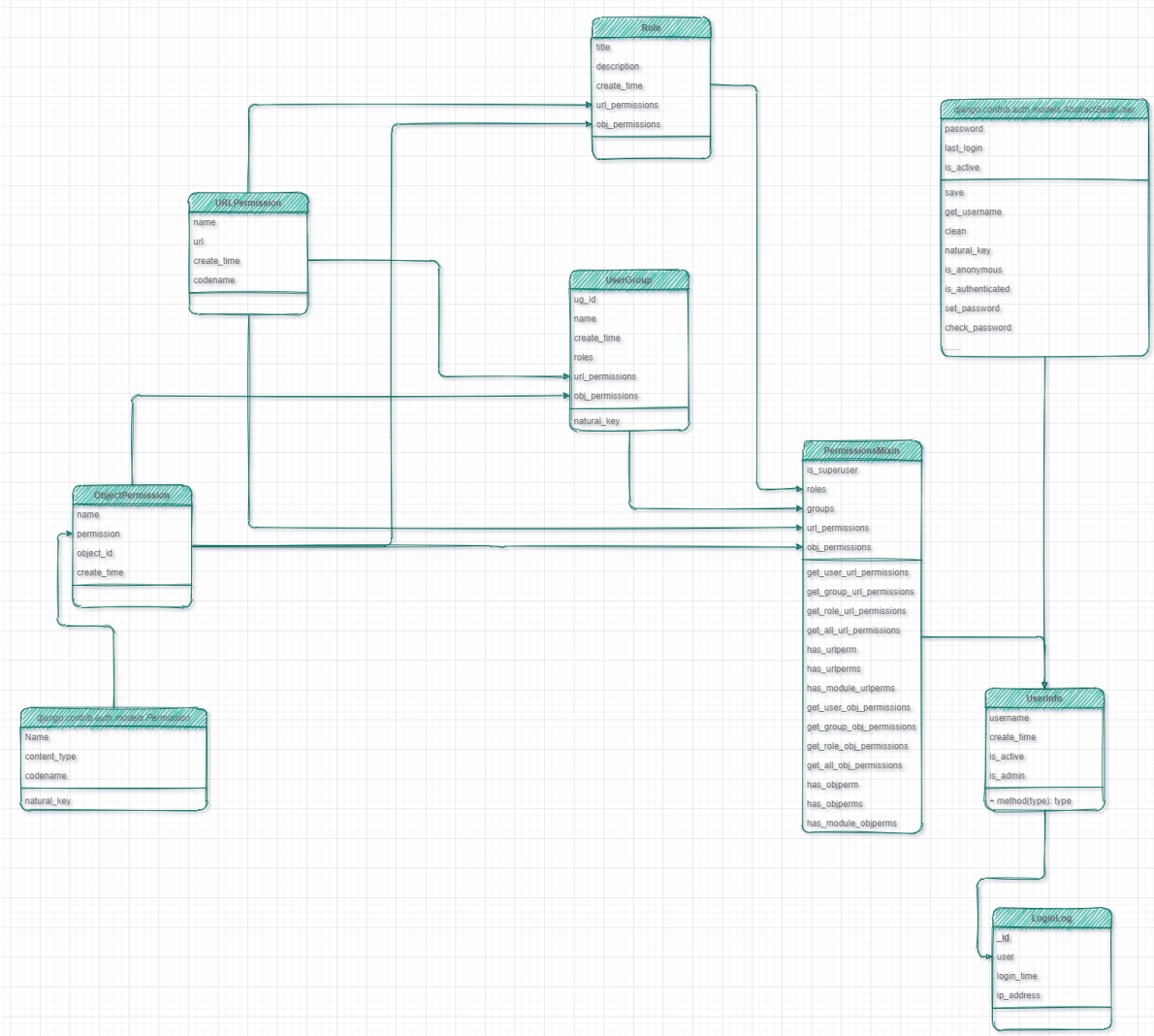
### 用户权限来源图




**ER图**


Role
PK
title
descirption
create_time
url_permissions
obj_permissions
URLPermission
PK
_id
name
url
codename
create_time
UserInfo
PK
_id
username
password
create_time
last_login
is_active
is_admin
is_superuser
groups
roles
url_permissions
obj_permissions
ObjectPermission
PK,FK
permission
PK
object_id
_id
name
create_time
UserGroup
PK
ug_id
name
create_time
roles
url_permissions
obj_permissions

## UML 类图



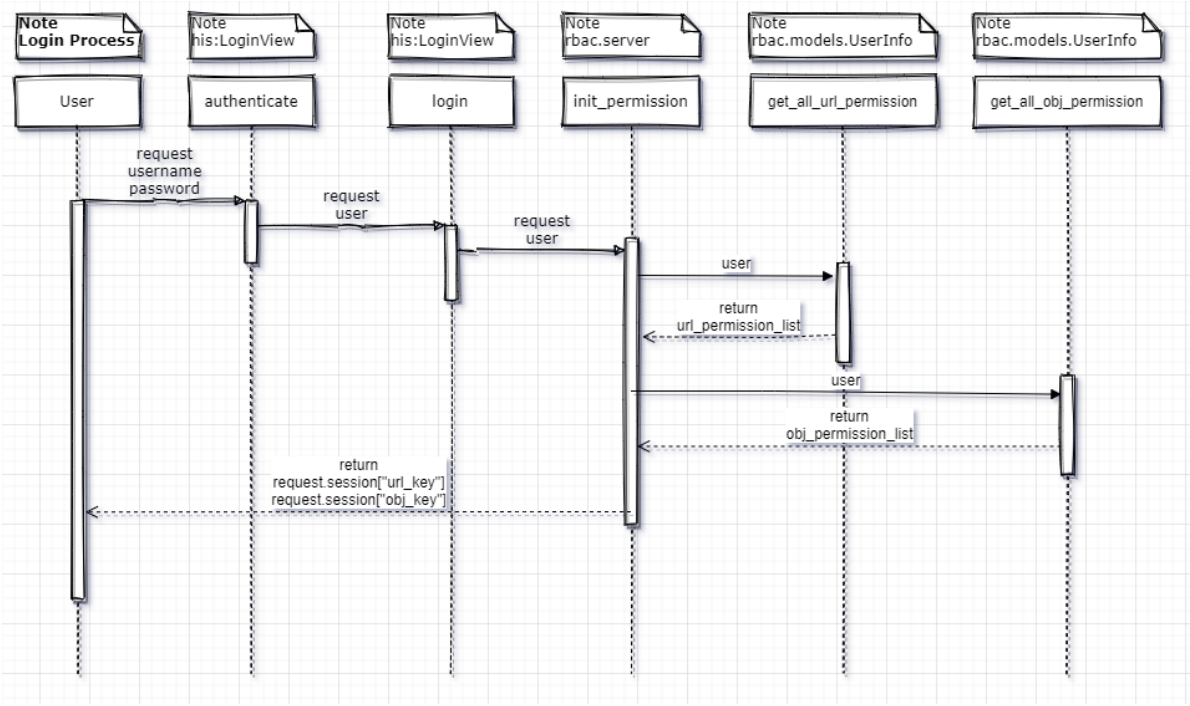

# 原理与使用说明

要使用 `HIS.rbac`，请按照以下步骤进行设置：

`settings.py`

- `INSTALLED_APPS` 中加入 `'rbac'`，启用 RBAC 应用；
- `MIDDLEWARE` 中加入 `"rbac.middleware.rbac.RBACMiddleware"`，启用 RBAC 中间件，用于鉴别是否拥有 URL 访问权限；
- 加入设置：`AUTH_USER_MODEL = "rbac.UserInfo"`，指定自定义的认证用户模型；
- 加入设置：`AUTHENTICATION_BACKENDS = (rbac.backends.CustomBackends", )`，设置认证后端，实现上述两个权限模型的获取与处理；
- 加入设置：`PERMISSION_URL_KEY = "url_key"`，指定 session 中用户全部 URL 访问权限的键；
- 加入设置：`PERMISSION_OBJ_KEY = "obj_key"`，指定 session 中用户全部对象资源权限的键；
- 加入设置：`SAFE_URL = [...]`，指定无需登录也可访问的 URL。

在用户登录时，需要使用 `HIS.rbac.server.init_permission` 中的 `init_permission` 函数获取用户所有的URL访问权限和对象资源权限。登录流程及所用函数如下所示：

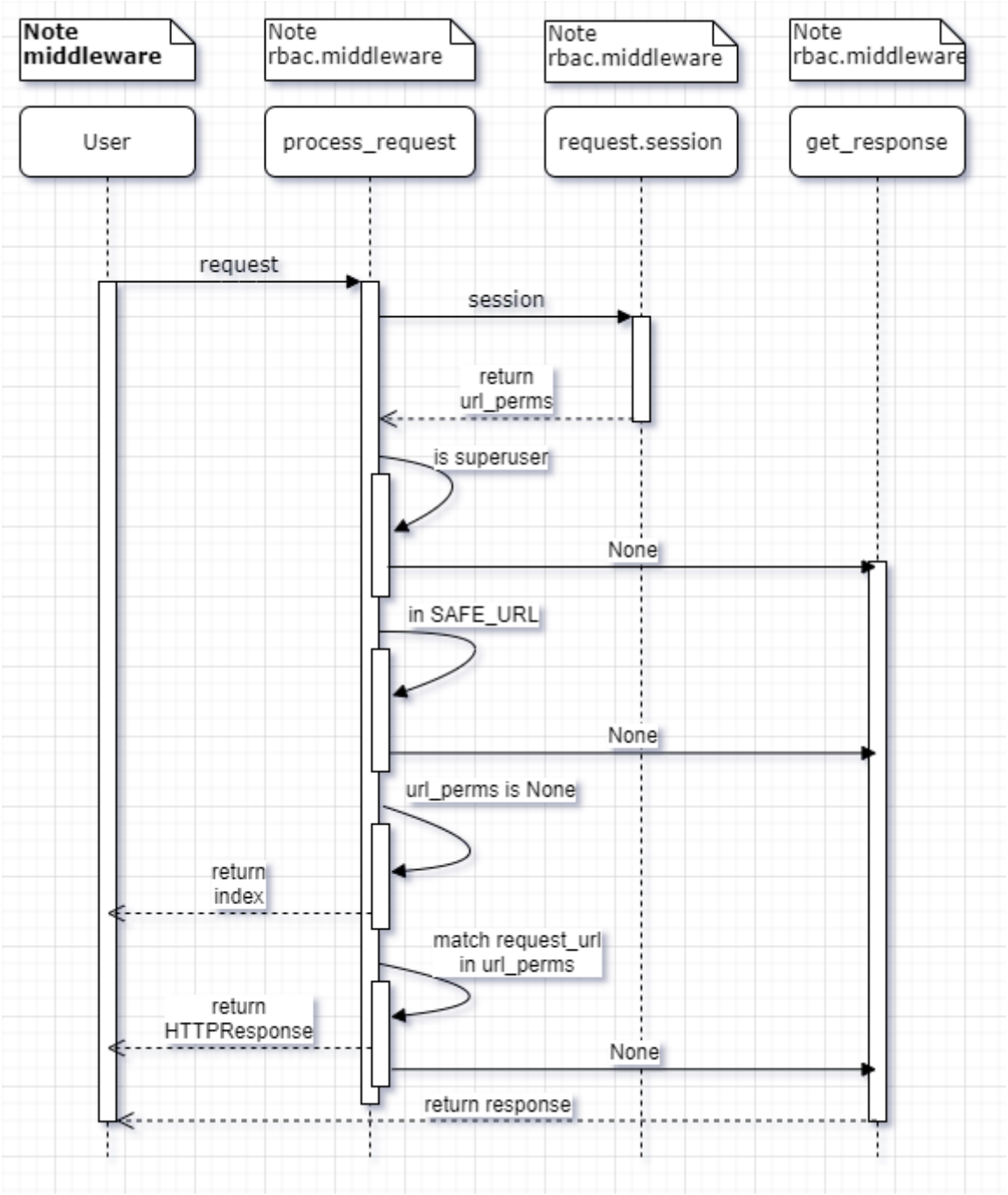



权限以字符串嵌套列表的形式储存在 `request.session` 中，如下例所示：

```
url permissions: [['test_1', '/1/'], ['test_8', '/8/']]
obj permissions: [['staff', '7', 'change_staff'], ['department', '1',
'change_department']]
```

> URL Permission 的格式为 `<codename>.<url>`，经 `init_permission()` 处理为 `(<codename>, <url>)` 的格式；
>
> Object Permission 的格式为 `<app_label>.<model>.<object_id>.<codename>`，经 `init_permission()` 处理为 `(<model>, <object_id>, <codename>)` 的格式；

每次访问 URL 时，通过 RBAC 中间件鉴权：




对象资源权限相关函数，属于 `UserInfo` 类的实例方法：

- `user_obj.get_user_url_permissions()`
  - 获取该用户直接拥有的 URL 访问权限。
  - `@return` 形如 `<codename>.<url>` 的字符串组成的集合。
- `user_obj.get_group_url_permissions()`
  - 获取该用户所属用户组拥有的全部 URL 访问权限，包括用户组直接拥有的权限与用户组所属角色拥有的全部权限。
  - `@return` 形如 `<codename>.<url>` 的字符串组成的集合。
- `user_obj.get_role_url_permissions()`
  - 获取该用户所属角色拥有的 URL 访问权限。
  - `@return` 形如 `<codename>.<url>` 的字符串组成的集合。
- `user_obj.get_all_url_permissions()`
  - 获取该用户直接或间接拥有的全部 URL 访问权限。
  - `@return` 形如 `<codename>.<url>` 的字符串组成的集合。
- `user_obj.has_urlperm(urlperm)`

    - 判断用户是否具有给定的 `urlperm`
    - `@urlperm` 形如 `<codename>.<url>` 的字符串。
    - `@return` True or False。
- `user_obj.has_urlperms(urlperm_list)`
    - 判断用户是否具有给定的 `urlperm_list` 中的所有URL访问权限。
    - `@urlperm_list` 形如 `<codename>.<url>` 的字符串列表。
    - `@return` True or False。
- `user_obj.has_module_urlperms(url_regex)`
    - 判断用户在指定页面正则表达式上是否具有访问权限。
    - `@url_regex` 形如 `^/index$` 的 URL 匹配正则表达式。
    - `@return` True or False。
- `user_obj.get_user_obj_permissions()`
    - 获取该用户直接拥有的对象资源权限。
    - `@return` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串组成的集合。
- `user_obj.get_group_obj_permissions()`
    - 获取该用户所属用户组拥有的全部对象资源权限，包括用户组直接拥有的权限与用户组所属角色拥有的全部权限。
    - `@return` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串组成的集合。
- `user_obj.get_role_url_permissions()`
    - 获取该用户所属角色拥有的对象资源权限。
    - `@return` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串组成的集合。
- `user_obj.get_all_url_permissions()`
    - 获取该用户直接或间接拥有的全部对象资源权限。
    - `@return` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串组成的集合。
- `user_obj.has_objperm(objperm)`
    - 判断用户是否具有给定的 `objperm`
    - `@objperm` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串。
    - `@return` True or False。
- `user_obj.has_urlperms(urlperm_list)`
    - 判断用户是否具有给定的 `urlperm_list` 中的所有URL访问权限。
    - `@objperm_list` 形如 `<app_label>.<model>.<object_id>.<codename>` 的字符串列表。
    - `@return` True or False。
- `user_obj.has_module_objperms(app_label)`
    - 判断用户在指定应用上是否具有访问权限。
    - `@app_label` 应用名称字符串。
    - `@return` True or False。

# 测试用例

## 权限预定义与预分配

1. URL 访问权限：

| 权限名 | URL 2 | 权限代码 1 |
| --- | --- | --- |
| 测试URL访问权限1 | /1/ | test_1 |
| 测试URL访问权限10 | /10/ | test_10 |
| 测试URL访问权限2 | /2/ | test_2 |
| 测试URL访问权限3 | /3/ | test_3 |
| 测试URL访问权限4 | /4/ | test_4 |
| 测试URL访问权限5 | /5/ | test_5 |
| 测试URL访问权限6 | /6/ | test_6 |
| 测试URL访问权限7 | /7/ | test_7 |
| 测试URL访问权限8 | /8/ | test_8 |
| 测试URL访问权限9 | /9/ | test_9 |

10 URL访问权限

2. 对象访问权限：

| 对象权限名称 | 权限 | 对象ID |
| --- | --- | --- |
| 测试对象权限1 | his \| 科室部门 \| Can change 科室部门 | 1 |
| 测试对象权限2 | his \| 科室部门 \| Can delete 科室部门 | 2 |
| 测试对象权限3 | his \| 科室部门 \| Can view 科室部门 | 3 |
| 测试对象权限4 | his \| 职工 \| Can change 职工 | 4 |
| 测试对象权限5 | his \| 职工 \| Can delete 职工 | 5 |
| 测试对象权限6 | his \| 职工 \| Can view 职工 | 6 |
| 测试对象权限7 | his \| 职工 \| Can change 职工 | 7 |
| 测试对象权限8 | his \| 职工 \| Can delete 职工 | 8 |
| 测试对象权限9 | his \| 职工 \| Can view 职工 | 9 |

9 对象权限

3. 角色：

| ID | Role | URL Perm | OBJ Perm |
|---|---|---|---|
| 1 | HIS 开发人员 | All (1-10) | All (1-9) |
| 2 | HIS 测试工具人 | 1, 2 | 1, 2 |
| 3 | 内科住院部医生 | 3, 4 | 3, 4 |
| 4 | 外科门诊医生 | 5, 6 | 5, 6 |

4. 用户组：

| ID | UserGroup | URL Perm | OBJ Perm |
|---|---|---|---|
| 1 | HIS开发部 | All (1-10) | All (1-9) |
| 2 | HIS测试部 | 9, 10 | 8, 9 |
| 3 | 内科 | 7, 8 | 6, 7 |
| 4 | 外科 | 5, 6 | 4, 5 |

## 用户权限分配

用户：

| User | Total URL Perms | Total OBJ Perms | Roles | UserGroups | URL Perms | OBJ Perms |
|---|---|---|---|---|---|---|
| zyc | All (1-10) | All (1-9) | 1 | 1 | | |
| 000000 | 1, 2, 9, 10 | 1, 2, 8, 9 | 2 | 2 | | |
| 000001 | 1, 2, 7, 8 | 1, 2, 6, 7 | 2 | 3 | | |
| 000002 | 1, 2, 5, 6 | 1, 2, 4, 5 | 2 | 4 | | |
| 000003 | 1, 2, 7, 8, 9, 10 | 1, 2, 6, 7, 8, 9 | 2 | 2, 3 | | |
| 000004 | All (1-10) | All (1-9) | 2, 3 | 2, 3, 4 | | |
| 000005 | 1, 2, 5, 9, 10 | 1, 2, 5, 8, 9 | 2 | 2 | 5 | 5 |
| 000006 | 1, 2, 7, 8, 9 | 1, 2, 3, 6, 7 | 2 | 3 | 7, 8, 9 | 1, 2, 3 |
| 000007 | 1, 2, 3, 4 | 2, 3, 4 | 3 | | 1, 2 | 2, 3 |
| 000008 | 4, 5, 6, 7 | 3, 4, 5, 6 | | 4 | 4, 7 | 3, 6 |
| 000009 | 1, 5, 10 | 2, 6, 9 | | | 1, 5, 10 | 2, 6, 9 |

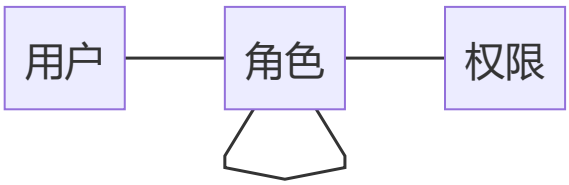
## 结果

测试用例全部通过。

# 未来开发方向

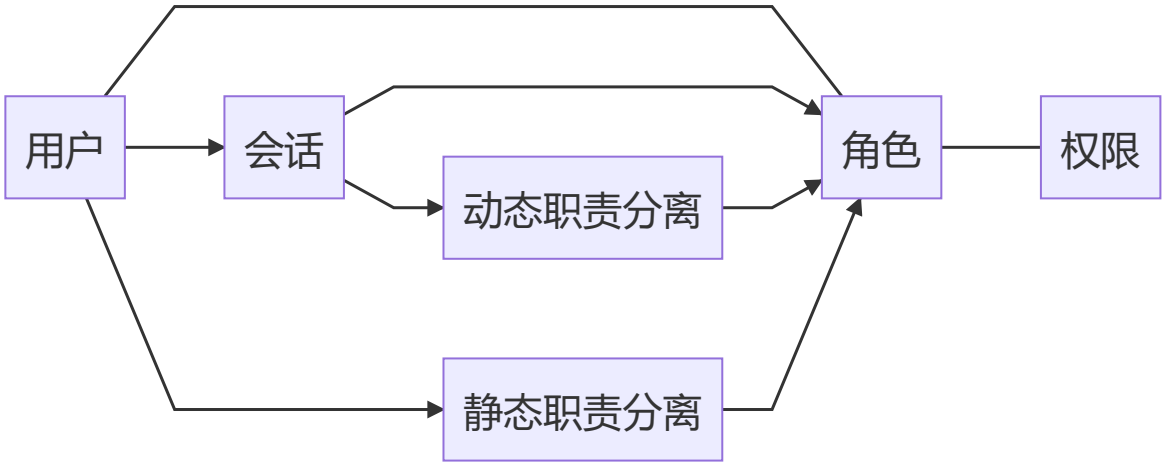
`HIS.rbac` 只完成了 RBAC0 级别的权限管理系统，如下图所示：


未来可以进一步完善 RBAC 系统，加入角色的继承机制，达到 RBAC1：

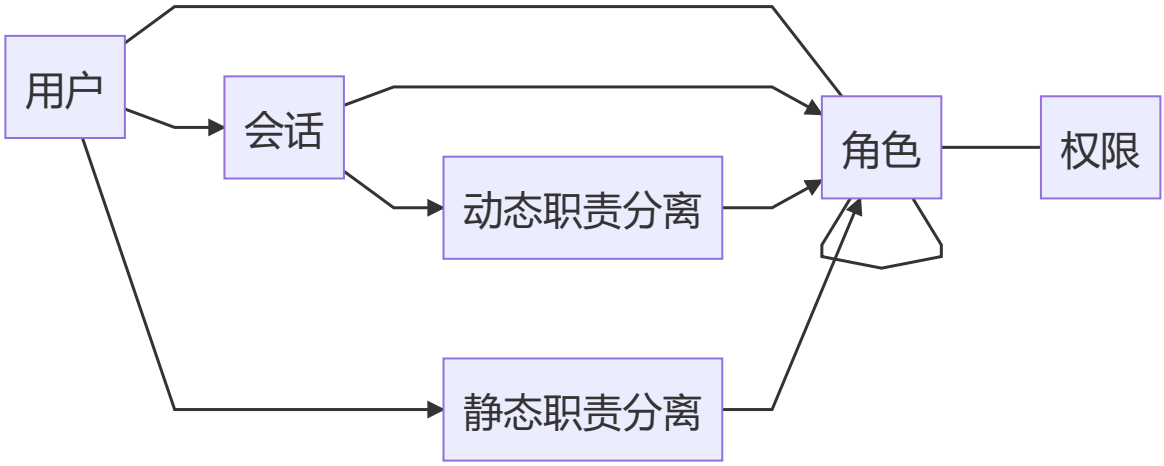



或者加入授权时应遵循的强制性规则，完成动态职责分离（DSD）、静态职责分离（SSD）：

> 例如，一个用户不应该同时具有出纳和会计两个角色的权限。




或者，使系统同时具有 RBAC1 和 RBAC2 的特点，达到 RBAC3：


用户
会话
角色
权限
动态职责分离
静态职责分离

##