角色和权限

注意：从这个功能开始，不一定实现全部功能，相似功能省略，请同学们自行完成

概念

每个用户都有自己的权力，能干什么，不能干什么？这就是权限。

```
User1有权限A、B、C
User2有权限A、B、C
User3有权限A、B、C、D
User4有权限D、F
```

有多个用户，可能拥有同样的权限，可以认为他们是一组

```
P1组有权限A、B、C
P2组有权限D、F

User1属于P1组，就直接拥有了A、B、C权限
User2属于P1组，也就直接拥有了A、B、C权限
User3属于P1组，也就直接拥有了A、B、C权限，单独再为User3赋予D权限即可
User4属于P2组，就直接拥有了D、F权限
```

从上面的分析可知，有用户、组、权限，它们形成了关系。组就是角色，很多用户都可以赋予该角色。

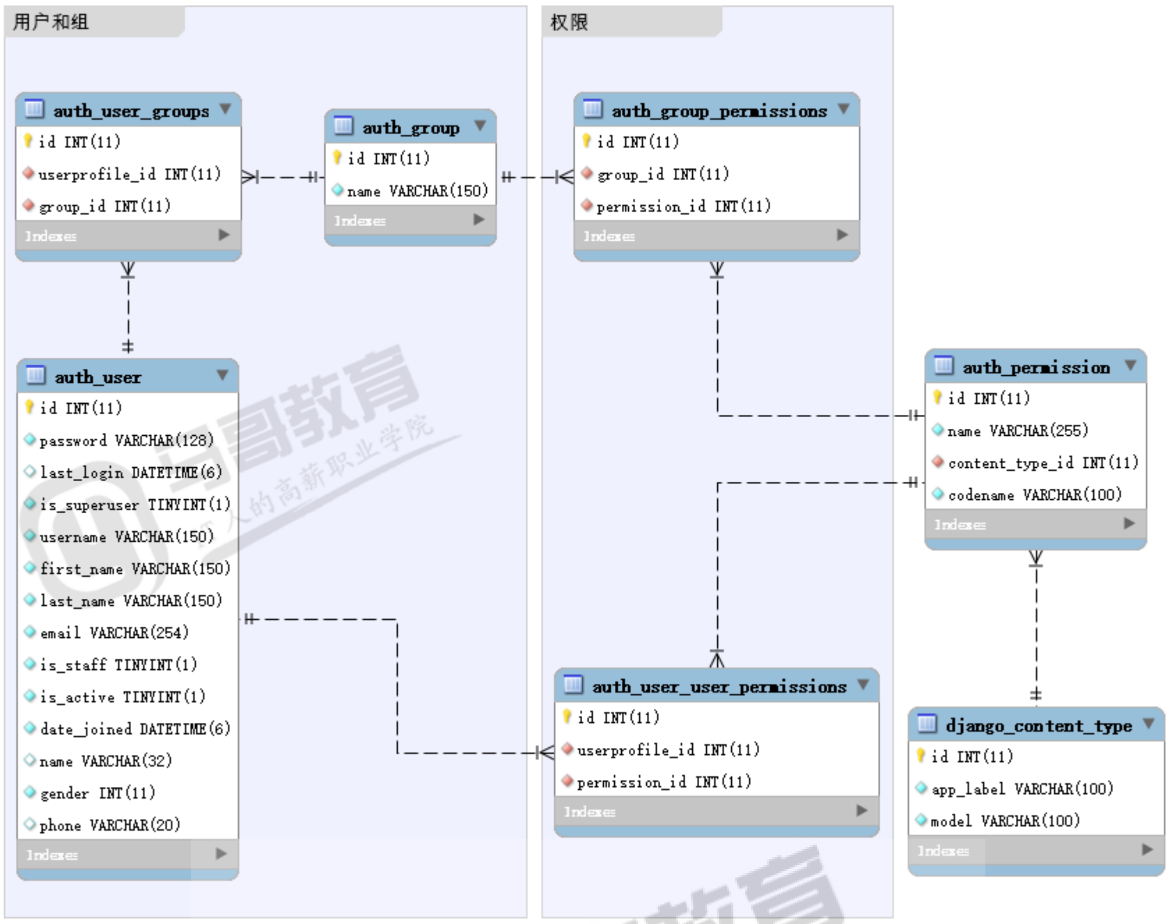
这种基于角色的权限设计称为RBAC（Role-Based Access Control）。

复杂设计：用户自身有权力，用户还可以同时属于组，获得组的权力

简单设计：用户必须属于组，获得组的权力

Django实现了复杂的设计，但是我们可以简化它，按照简单的方式使用。

Django权限设计



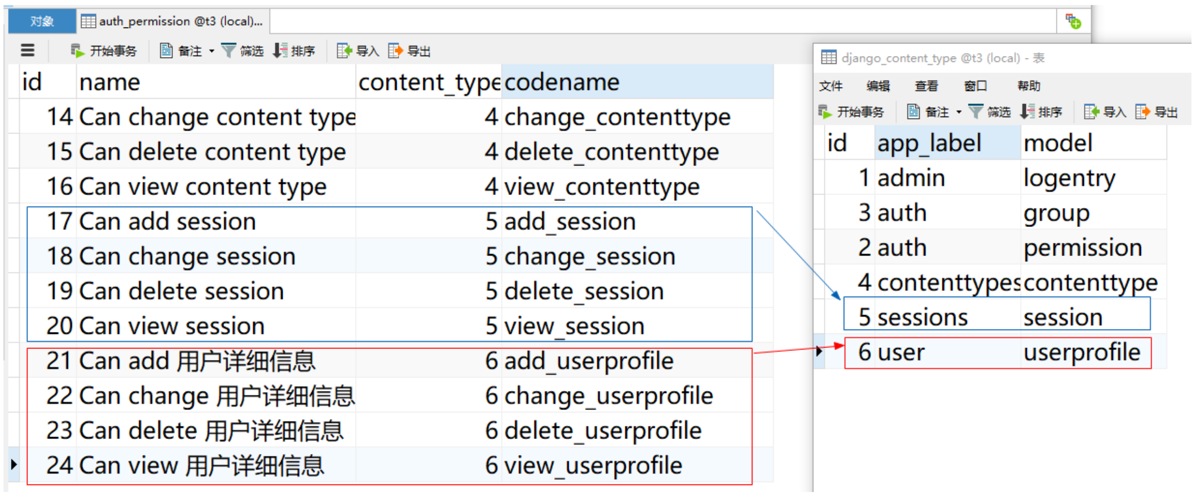
Django的权限表

- 用户和组，多对多。一个用户可以属于多个组，一个组可以有多个用户
- 组和**组权限**，多对多。一个组可以赋予多种权限，一个权限可以赋给多个组
- 用户和**用户权限**，多对多。一个用户可以有多个权限，一个权限可以赋给多个用户

权限设置的2条路径

- 用户设置所属组，通过组间接获得权限
- 用户直接设置权限，auth_user_user_permissions

下图是auth_permission和django_content_type表关系，一对多关系。一个Model对应4个默认权限。


对象
auth_permission @t3 (local)...
开始事务
备注
筛选
排序
导入
导出
django_content_type @t3 (local) - 表
文件
编辑
查看
窗口
帮助

id	name	content_type	codename
14	Can change content type	4	change_contenttype
15	Can delete content type	4	delete_contenttype
16	Can view content type	4	view_contenttype
17	Can add session	5	add_session
18	Can change session	5	change_session
19	Can delete session	5	delete_session
20	Can view session	5	view_session
21	Can add 用户详细信息	6	add_userprofile
22	Can change 用户详细信息	6	change_userprofile
23	Can delete 用户详细信息	6	delete_userprofile
24	Can view 用户详细信息	6	view_userprofile

id	app_label	model
1	admin	logentry
3	auth	group
2	auth	permission
4	contenttypes	contenttype
5	sessions	session
6	user	userprofile

userprofile就是user应用下的Model类UserProfile，一个模型类默认就有了4种权限

- add 增
- delete 删

- change 改
- view 查

也就是某个用户应该可以使用某个Model的某些权限。

那么用户是怎样拥有了权限?

```
class AbstractUser(AbstractBaseUser, PermissionsMixin):
    pass

class PermissionsMixin(models.Model): # 增加对组、权限的支持
    is_superuser = models.BooleanField()
    groups = models.ManyToManyField(Group, verbose_name=_('groups'))
    user_permissions = models.ManyToManyField(Permission,
verbose_name=_('user permissions'))

    def get_user_permissions(self, obj=None): # 获取用户直接的权限
        pass
    def get_group_permissions(self, obj=None): # 获取用户通过组获取的权限
        pass
    def get_all_permissions(self, obj=None): # 获取用户所有权限
        pass
    def has_perm(self, perm, obj=None):
        pass # 激活的超级用户永远返回True
```

靠的是PermissionsMixin，同时提供了访问用户权限、组权限的方法。

- 超级用户拥有所有权限，或者说无视权限限制
- 普通用户，创建完，没有设置任何组、任何权限，就是什么权限都没有

AbstractUser继承了PermissionsMixin，因此用户就拥有了权限，从代码中可以看出

- 和用户权限、组都是多对多关系
- 用户对象可以获取该用户直接权限get_user_permissions
- 用户对象可以获得该用户组的权限get_group_permissions
- 用户对象可以获得该用户所有权限get_all_permissions
- 用户是否具有该权限has_perm

Django官方文档

```
Permissions can be set not only per type of object, but also per specific
object instance. By using the has_view_permission(), has_add_permission(),
has_change_permission() and has_delete_permission() methods provided by the
ModelAdmin class, it is possible to customize permissions for different
object instances of the same type.

User objects have two many-to-many fields: groups and user_permissions. User
objects can access their related objects in the same way as any other Django
model:

1 组权限设置
group.permissions.set([permission_list])
group.permissions.add(permission, permission, ...)
group.permissions.remove(permission, permission, ...)
group.permissions.clear()
2 关注用户和组（角色）的方法，项目中我们不直接给用户赋权，而是通过角色
myuser.groups.set([group_list])
myuser.groups.add(group, group, ...)
myuser.groups.remove(group, group, ...)
myuser.groups.clear()

下面是用户权限设置，我们不用
myuser.user_permissions.set([permission_list])
myuser.user_permissions.add(permission, permission, ...)
myuser.user_permissions.remove(permission, permission, ...)
myuser.user_permissions.clear()
```

创建分支

前后台代码中，都创建perm分支，开始新的开发

```
$ git checkout -b perm
```

权限管理

功能：

- 权限列表、分页
- 权限搜索

因为添加权限是Django对模块自动增加的，所以不需要提供 `增加权限` 功能，编辑、删除也不提供了，是只读的。

前台代码

src/router/index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../components/Login.vue'
import Home from '../components/Home.vue'
import Welcome from '../components/Welcome.vue'
import User from '../components/user/Users.vue'
import Perm from '../components/user/Perms.vue'

Vue.use(VueRouter)

const routes = [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome',
    children: [
      { path: '/welcome', component: Welcome },
      { path: '/users', component: User },
      { path: '/users/perms', component: Perm }
    ]
  }
]
```

src/components/user/Perms.vue

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>权限列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <div class="grid-content bg-purple-dark">
            <el-input placeholder="请输入内容" v-model="search">
              <el-button slot="append" icon="el-icon-search" @click="getList(1)"></el-button>
            </el-input>
          </div>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="id" label="id"> </el-table-column>
        <el-table-column prop="name" label="权限名称"> </el-table-column>
        <el-table-column prop="codename" label="codename"> </el-table-column>
        <el-table-column prop="content_type.app_label" label="应用"> </el-table-column>
        <el-table-column prop="content_type.model" label="模型"> </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
  },
  data() {
    return {
      search: '',
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 }
    }
  },
  methods: {
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/perms/', {
        params: {
          page,
          search: this.search
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    }
  }
}
</script>

<style>
</style>
```

后台代码

user/urls.py

```python
from django.urls import path, reverse
from .views import menulist_view, UserViewSet, PermViewSet
from rest_framework.routers import SimpleRouter

router = SimpleRouter()
router.register('', UserViewSet)
# router.register('perms', PermViewSet) # list有问题, retrieve没有问题

urlpatterns = [
    path('menulist/', menulist_view),
    path('perms/', PermViewSet.as_view({'get':'list'})),
    path('perms/<int:pk>/', PermViewSet.as_view({'get':'retrieve'})),
] + router.urls

print('~' * 30)
print(urlpatterns)
print('~' * 30)
```

```python
router.register('perms', PermViewSet) # list有问题, retrieve没有问题
```

解决方案，给UserViewSet增加前缀，否则会影响其他的注册的ViewSet的list。

```
router.register('mgr', UserViewSet)
```

相应的，需要修改前端代码，搜索this.$http中使用的`users/`，跟UserViewSet相关的，全部改为`users/mgr/`

user/serializers.py

```python
from rest_framework import serializers
from .models import UserProfile
from django.contrib.auth.models import make_password, Group, Permission, ContentType

class ContentTypeSerializer(serializers.ModelSerializer):
    class Meta:
        model = ContentType
        fields = '__all__'

class PermSerializer(serializers.ModelSerializer):
    class Meta:
        model = Permission
        fields = '__all__'
    content_type = ContentTypeSerializer(read_only=True) # 嵌套序列化器
```

user/views.py

特别注意：排除掉Django的内建应用

```python
from rest_framework.views import Response, Request
from rest_framework.decorators import api_view, permission_classes, action
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from rest_framework.viewsets import ModelViewSet, ReadOnlyModelViewSet
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer, PermSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from django.http.response import Http404
from utils.exceptions import InvalidPassword
from django.contrib.auth.models import Group, Permission, ContentType

_exclude_contenttypes = [c.id for c in ContentType.objects.filter(model__in=[
    'logentry', 'group', 'permission',
    'contenttype', 'session'
])] # 排除掉Django内建应用的权限
# print(_exclude_contenttypes, '++++++++++++++++++')

class PermViewSet(ReadOnlyModelViewSet):
    queryset = Permission.objects.exclude(content_type__in=_exclude_contenttypes)
    serializer_class = PermSerializer
    filter_backends = [filters.SearchFilter]
    search_fields = ['name', 'codename']
```

角色列表

功能：

- 角色列表、分页
- 添加角色
- 修改、删除角色（同用户功能，请自行实现）
- 用户赋权

前台代码

src/router/index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../components/Login.vue'
import Home from '../components/Home.vue'
import Welcome from '../components/Welcome.vue'
import User from '../components/user/Users.vue'
import Perm from '../components/user/Perms.vue'
import Role from '../components/user/Roles.vue'

Vue.use(VueRouter)

const routes = [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome',
    children: [
      { path: '/welcome', component: Welcome },
      { path: '/users', component: User },
      { path: '/users/perms', component: Perm },
      { path: '/users/roles', component: Role }
    ]
  }
]
```

src/components/user/Roles.vue

编辑、删除、搜索功能，请自行实现

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>权限列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
```

```
      <el-row :gutter="20">
        <el-col :span="12">
          <div class="grid-content bg-purple-dark">
            <el-input placeholder="请输入内容" v-model="search">
              <el-button slot="append" icon="el-icon-search" @click="getList(1)"></el-button>
            </el-input>
          </div>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addDialogVisible = true">增加角色</el-button>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="id" label="id"> </el-table-column>
        <el-table-column prop="name" label="名称"> </el-table-column>
        <el-table-column label="操作">
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 添加对话框 -->
    <el-dialog title="增加" :visible.sync="addDialogVisible" @close="resetForm('add')">
      <el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
        <el-form-item label="名称" prop="name">
          <el-input v-model="addForm.name"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="add">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
  },
  data() {
    return {
      search: '',
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 },
      addDialogVisible: false,
      addForm: {
        name: ''
      },
      addRules: {
        name: [
          { required: true, message: '请输入名称', trigger: 'blur' },
          { min: 1, max: 16, message: '长度在 1 到 16 个字符', trigger: 'blur' }
        ]
      }
    }
  },
  methods: {
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/roles/', {
        params: {
          page,
          search: this.search // 搜索自行实现
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    },
    resetForm(name) {
      this.$refs[name].resetFields()
    },
    add() {
      const name = 'add'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('users/roles/', this.addForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.addDialogVisible = false
          this.resetForm(name)
          this.getList() // 刷新用户列表
        }
      })
    }
  }
}
</script>

<style>
</style>
```

后台代码

user/urls.py

```
from django.urls import path
from .views import menulist_view, UserViewSet, PermViewSet, RoleViewSet
from rest_framework.routers import SimpleRouter


router = SimpleRouter()
router.register('mgr', UserViewSet)
router.register('perms', PermViewSet)
router.register('roles', RoleViewSet)

urlpatterns = [
    path('menulist/', menulist_view),
] + router.urls

print('~' * 30)
print(*urlpatterns, sep='\n')
print('~' * 30)
```

user/serializers.py

```
class GroupSerializer(serializers.ModelSerializer):
    class Meta:
        model = Group
        fields = '__all__'
```

user/views.py

```
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth.models import Permission, Group
from .serializers import UserSerializer, PermSerializer, GroupSerializer


class RoleViewSet(ModelViewSet):
    queryset = Group.objects.all()
    serializer_class = GroupSerializer
```