# Django权限管理自定义以及权限校验

在大多数情况下 Django 默认的权限管理，不能满足开发者的实际业务需求，这时候就需要添加自定义权限，Django 给开发者提供了不止一种的方法来完成自定义权限，在本节我们将一起认识它们。实现了权限的自定义，我们就需要应用这些权限以及校验它们是否达到预期的功能，在本节我们将围绕这些知识点展看讲解。

## 1.实现添加自定义权限

### 1) Meta属性中创建权限

使用这种方式比较简单，我们可以在我们需要添加权限的 Model 的 Meta 属性中，创建添加相应的权限，比如我们在 BookStore 项目的 user 应用中 models.py 的 User Model 中添加以下代码：

```
class Meta:
    permission=((
        'can_view_index',' Can View Index '
                ))
```

可以看到，permissions 元选项中定义一个二元组，这个二元组的第一个元素指定了权限的 codename，第二个元素指定了权限的 name。那么如何将自定义的权限应用到系统中呢？这时就需要执行数据库迁移操作了，大家一定要记住，只要改动了 Model 就需要我们执行数据库迁移操作，在这里就不给大家进行演示了。同步之后，大家可查看 auth_permission 表，就可以看到自定义的权限了。

### 2) 程序的方式创建权限

我们还可以使用 ContentType 程序化创建 permissions，即用程序的形式创建自定义权限，这个过程需要按照以下步骤执行：首先获取某个应用的 ContentType 实例对象，然后给定 codename 和 name 以及 ContentType 创建好的实例对象。这里需要大家注意的是由于 codename 与 ContentType 之间具有 together_unique 限制即联合唯一性限制，所以不能与当前权限的 codename 存在重复。

我们打开 Django  shell 环境使用上述方法添加一个新的权限，程序代码如下所示：

```
In [1]: from index.models import UserInfo
   ...: from django.contrib.auth.models import Permission
   ...: from django.contrib.contenttypes.models import ContentType
   ...: content_type = ContentType.objects.get_for_model(UserInfo)
```

```
   ...: permission1 = Permission.objects.create(codename = 'publish_book',name = 'Can publish books',content_type = con
   ...: tent_type,
   ...: )
   ...: permission2 = Permission.objects.create(codename = 'comment_book',name = 'Can comment books',content_type = con
   ...: tent_type,
   ...: )
In [2]: permission2
Out[2]: <Permission: index | user info | Can comment books>
In [3]: permission1
Out[3]: <Permission: index | user info | Can publish books>
```

通过如下代码就给 UserInfo Model 成功添加了两个新的权限。

# 2. 授权的校验与验证

在 Django shell 环境中我们通过以下方式可以得知，在 auth_user 表中已经存在以下两个用户：

```
from django.contrib.auth.models import User
User.objects.all()
返回结果：<QuerySet [<User: admin>, <User: bookstore>]>
```

其中 admin 是超级用户不需要授予其他权限，所以我们可以通过操作 bookstore 用户的权限进行讲解。

### 1) 用户权限管理操作

给用户添加、删除权限的过程其实就是修改 auth_user_user_permissions 表数据记录的过程，它是 User 和 Permission 的多对多关联关系表。在操作用户权限的过程中，我们要给 User 的实例添加属性需要使用它的 user_permissions 属性，首先，我们来获取 User 对象和 Permission 对象实例，操作权限的代码分别如下所示：

```
from django.contrib.auth.models import User, Permission
user=User.objects.get(username='bookstore')
add_book=Permission.objects.get(codename='add_book')
change_book=Permission.objects.get(codename='change_book')
#查看实例对象所有权限若无任何返回值是空集合set
user.get_all_permission()
#将user的权限设置为当前权限值, 之前权限的会自动去掉
user.user_permission.set([add_book])
#在当前权限的基础新增权限
user.user_permission.add(change_book)
#同时也可接受多个权限值
user.user_permission.add(add_book,change_book)
#删除权限
user.user_permission.remove(change_book)
#清空所有权限
user.user_permission.clear()
```

上述代码其实很好理解，我们在这里只对其中的一项做简单的介绍，user_permission.ser() 可以将当前用户的权限设置为当前指定的值，这里的意思就是说，无论之前 user 实例拥有多少权限，使用完这个方法后只会设置成当前的值，所以说 user 再执行完这个方法后，只拥有 add_book 权限。

**2) 用户组权限管理操作**

理解了用户的权限管理操作，那么用户组的权限操作就变的更为简单。由于 Group 中也定义了与 Permission 的关联关系，所以，给用户组添加、删除权限的过程基本同 User 是类似的。在 Django 的源码中，Group 的关联 Permission 的字段是 permissions，所以我们可以通过下面的方式给用户组设置相应的权限：

```
from django.contrib.auth.models import User, Permission,Group
add_book=Permission.objects.get(codename='add_book')
change_book=Permission.objects.get(codename='change_book')
#创建用户组
group_book=Group.objects.get(name="library")
#添加用户组全权限
group_book.permissions.set([add_book,change_book])
#查当前用户权限
user.get_all_permission()
```

上述语句执行之后，Group 与 Permission 的关联表 auth_group_permissions 中会增加两条记录。之前在介绍 Group 的时候曾经说过，属于某个用户组的用户会自动拥有用该户组被授予的权限。

**3) 用户的权限校验**

通过上面的介绍。我知道了如何对用户与用户组的权限进行操作，接下来，我们还要明白权限授予后，我们还要对其进行校验，校验成功的用户方可执行某项权限规定的操作。用户的校验可以使用 User 实例的 has_perm 或 has_perms 方法，han_perm 判断当前用户是否有某一项权限，而后者则表示用户是否同时拥有多个权限。格式如下，has_perm 中传递的权限格式为：

has_perm('appname.codename(权限编码)')
而 has_perms 在校验多个权限时，需要将 n 个权限放入列表中，如下所示：

has_perms(["add_book","change_book"])

本节主要讲解了权限的自定义以及权限的基本操作与校验，在下一节中，我们将深入一步，讲解 Django 用户认证系统的实际业务应用。