# Django用户认证系统权限管理

在设计用户认证系统时用户的权限管理是是一项不可忽视的重要内容，权限管理可以限制用户是否能够拥有某些功能。Django 的用户认证系统对开发者同样提供了非常方便的权限管理，在《Django Auth用户与用户组详述》一节，我们了解了 User 与 Group 都有与权限相关联的表，它们之间存在着关联关系，可见用户与权限之间是密不可分的关系，在它们关联表中记录着当前用户或用户组拥有的权限。Django 默认为每一个模型表添加四个权限即查看、增加、更新、删除。那么这些又是怎么样实现的呢？结合前面的知识内容，我们就来一起分析一下吧。

## 1. 认识Django的权限管理

在《Django Auth用户与用户组详述》我们提到过 Permission 这张权限表，我们可以通过数据库查看它的数据表结构，如下所示：

```
mysql> desc auth_permission;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| name            | varchar(255)| NO   |     | NULL    |                |
| content_type_id | int(11)     | NO   | MUL | NULL    |                |
| codename        | varchar(100)| NO   |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

从上述表结构可以看出，Permission 权限表主要定义了 name、content_type_id、codename 三个字段，其中它的 content_type_id 为外键关联字段，它的各个字段的含义如下：name 表示权限名称，字符最大长度为 255；content_type 表示与 ContentType 是外键关联关系，这张表主要用于记录 App 与 model 的信息，最后一个字段 codename 代表权限的名称编码值，最多允许 100 个字符长度。它的表结构如下：

```
mysql> desc django_content_type;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int(11)     | NO   | PRI | NULL    | auto_increment |
| app_label | varchar(100)| NO   | MUL | NULL    |                |
| model     | varchar(100)| NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

### 1) Permission Model源码

Permission 的 Model 在 Django 中的源码定义如下所示：

```
class Permission(models.Model):
 name = models.CharField(_('name'), max_length=255)
    content_type = models.ForeignKey(
        ContentType,
        models.CASCADE,
        verbose_name=_('content type'),
    )
    codename = models.CharField(_('codename'), max_length=100)

    objects = PermissionManager()
    #元数据项
    class Meta:
        verbose_name = _('permission')
        verbose_name_plural = _('permissions')
        #联合唯一即联合约束
        unique_together = (('content_type', 'codename'),)
        ordering = ('content_type__app_label', 'content_type__model',
                    'codename')

    def __str__(self):
        return "%s | %s | %s" % (
            self.content_type.app_label,
            self.content_type,
            self.name,
        )
```

### 2) Options默认权限分类

权限系统提供了一种将权限分配给特定用户和用户组的方法主要通过 Options 类实现。Django 管理站点使用权限系统，但是在你自己的代码中也可以使用它。Django 管理站点使用的权限如下:

- “添加”权限限制用户“添加”表单的能力并添加一个 Model 实例对象。
- “更改”权限限制用户更改的能力，能够修改 Model 的实例对象。
- “删除”权限限制删除对象的能力，删除 Model 实例对象。
- “查看”权限限制查看对象的能力，查看 Model 实例对象

权限是针对对象类型的全局设置，而不是针对特定对象实例。定义了权限就可以将权限分配给用户或者用户组，给用户分配权限会关联到 auth_user_user_permission；用户组会关联到 auth_group_permisssion，如果要给对应的用户或用户组添加权限，那么这两张表就会相应的增加权限，Django 会默认添加四个权限，通过查看 auth_permission 表可以得知 user 权限如下：

```
mysql> select * from auth_permission where content_type_id=4;
+----+-----------------+-----------------+-------------+
| id | name            | content_type_id | codename    |
+----+-----------------+-----------------+-------------+
| 13 | Can add user    |               4 | add_user    |
| 14 | Can change user |               4 | change_user |
| 15 | Can delete user |               4 | delete_user |
| 16 | Can view user   |               4 | view_user   |
+----+-----------------+-----------------+-------------+
4 rows in set (0.01 sec)
```

Model 内置的权限被定义在 Django 如下模块中：

django.db.models.options.Options
在定义的 Options 类中包含了 default_permisssions 属性，它指定了这以上的四种权限，如下所示：

default_permissions = ('add', 'change', 'delete', 'view')
这几个权限在实际的开发业务中，可以根据需求进行相应的设置，同时还可以应用到 Django 的后台管理系统中，实现对 Model 的操作，当然这需要用户有访问管理后台的权限。

## 2. 查看用户或用户组的权限

权限名一般由权限动作和模型名组成。以 user 应用为例，Django 为 User 模型自动创建的 4 个可选权限名分别为:

- 查看用户(view)：user.view_article
- 创建用户(add)：user.add_article
- 更改用户(change)：user.change_article
- 删除用户(delete)：user.delete_article

那么如何查看用户拥有哪些权限呢？我们现在可以使用 user.has_perm() 方法来判断用户是否已经拥有相应权限。下例中应该返回 True。如下所示，其中 username 代表用户名字：

- user_username.has_perm('user.add_article')
- user_username.has_perm('user.change_article')

如果我们要查看某个用户所在用户组的权限或某个用户的所有权限(包括从用户组获得的权限)，我们可以使用 get_group_permissions() 和 get_all_permissions() 方法。

- user_username.get_group_permissions()
- user_username.get_all_permissions()

在本节我们主要认识了 Django 的权限的管理以及我们如何查看用户的权限，权限管理在 Django 的用户认证系统中有着极其重要的作用，在下一节中，我们将讲解如何添加自定义权限以及完成权限的授予与校验。