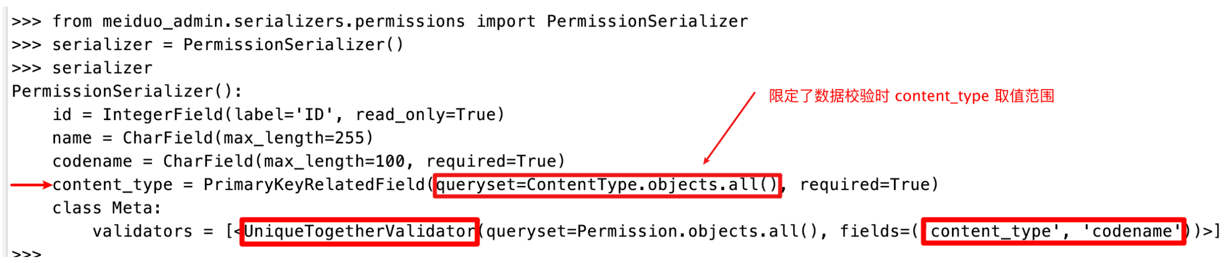
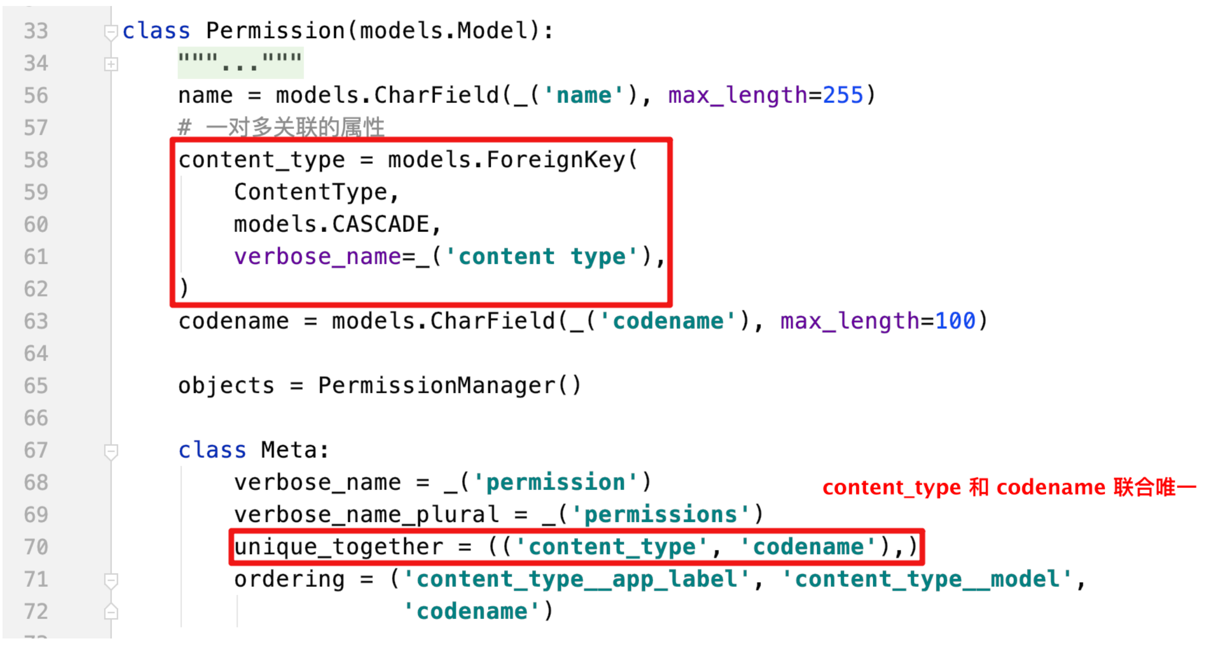
上午课程核心内容

1. 权限数据管理



```
>>> from meiduo_admin.serializers.permissions import PermissionSerializer
>>> serializer = PermissionSerializer()
>>> serializer
PermissionSerializer():
    id = IntegerField(label='ID', read_only=True)
    name = CharField(max_length=255)
    codename = CharField(max_length=100, required=True)
    content_type = PrimaryKeyRelatedField(queryset=ContentType.objects.all(), required=True)
    class Meta:
        validators = [<UniqueTogetherValidator(queryset=Permission.objects.all(), fields=('content_type', 'codename'))>]
>>>
```

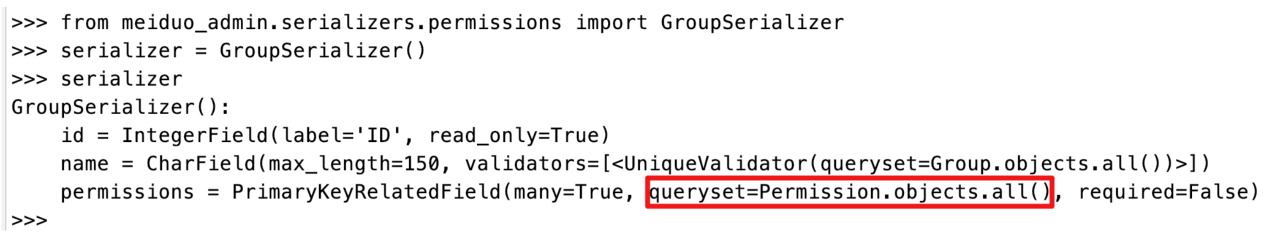
相关内容：



```
class Permission(models.Model):
    """..."""
    name = models.CharField(_('name'), max_length=255)
    # 一对多关联的属性
    content_type = models.ForeignKey(
        ContentType,
        models.CASCADE,
        verbose_name=_('content type'),
    )
    codename = models.CharField(_('codename'), max_length=100)

    objects = PermissionManager()

    class Meta:
        verbose_name = _('permission')
        verbose_name_plural = _('permissions')
        unique_together = (('content_type', 'codename'),)
        ordering = ('content_type__app_label', 'content_type__model',
                    'codename')
```

```python
from django.contrib.auth.models import Permssion
obj = Permssion.objects.get(id=1)
# 和权限对象关联的权限类型对象
obj.content_type
```

2. 用户组数据管理



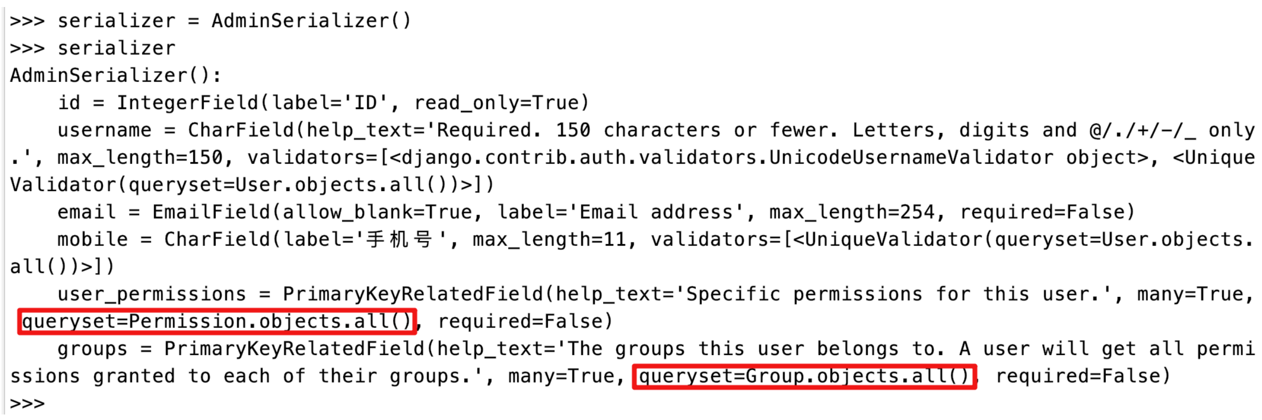
```
>>> from meiduo_admin.serializers.permissions import GroupSerializer
>>> serializer = GroupSerializer()
>>> serializer
GroupSerializer():
    id = IntegerField(label='ID', read_only=True)
    name = CharField(max_length=150, validators=[<UniqueValidator(queryset=Group.objects.all())>])
    permissions = PrimaryKeyRelatedField(many=True, queryset=Permission.objects.all(), required=False)
>>>
```

相关内容：

```python
from django.contrib.auth.models import Group
obj = Group.objects.get(id=1)
# 和用户组对象关联的权限对象数据
obj.permissions.all()
```

3. 管理员数据管理

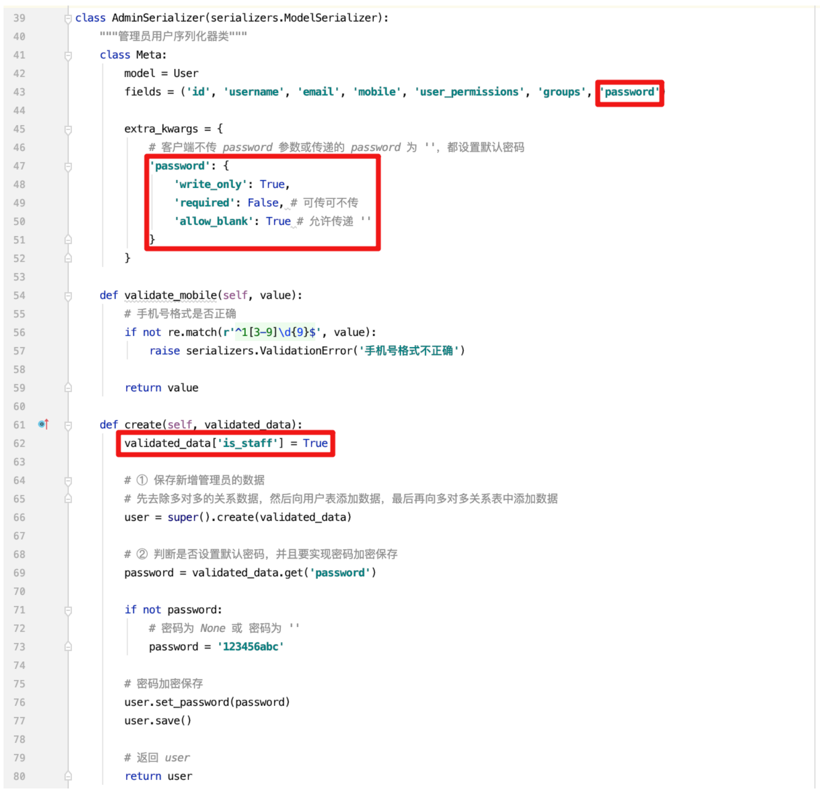
3.1 管理员数据获取



```
>>> serializer = AdminSerializer()
>>> serializer
AdminSerializer():
    id = IntegerField(label='ID', read_only=True)
    username = CharField(help_text='Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only
.', max_length=150, validators=[<django.contrib.auth.validators.UnicodeUsernameValidator object>, <Unique
Validator(queryset=User.objects.all())>])
    email = EmailField(allow_blank=True, label='Email address', max_length=254, required=False)
    mobile = CharField(label='手机号', max_length=11, validators=[<UniqueValidator(queryset=User.objects.
all())>])
    user_permissions = PrimaryKeyRelatedField(help_text='Specific permissions for this user.', many=True,
 queryset=Permission.objects.all(), required=False)
    groups = PrimaryKeyRelatedField(help_text='The groups this user belongs to. A user will get all permi
ssions granted to each of their groups.', many=True, queryset=Group.objects.all(), required=False)
>>>
```

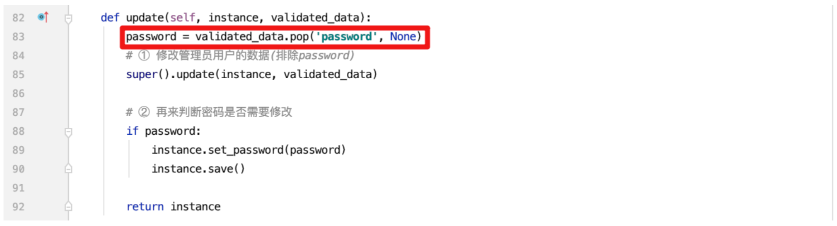
3.2 管理员数据新增

需要在 AdminSerializer 序列化器类中增加一个 password 字段：



```python
class AdminSerializer(serializers.ModelSerializer):
    """管理员用户序列化器类"""
    class Meta:
        model = User
        fields = ('id', 'username', 'email', 'mobile', 'user_permissions', 'groups', 'password')

        extra_kwargs = {
            # 客户端不传 password 参数或传递的 password 为 ''，都设置默认密码
            'password': {
                'write_only': True,
                'required': False, # 可传可不传
                'allow_blank': True # 允许传递 ''
            }
        }

    def validate_mobile(self, value):
        # 手机号格式是否正确
        if not re.match(r'^1[3-9]\d{9}$', value):
            raise serializers.ValidationError('手机号格式不正确')

        return value

    def create(self, validated_data):
        validated_data['is_staff'] = True

        # ① 保存新增管理员的数据
        # 先去除多对多的关系数据，然后向用户表添加数据，最后再向多对多关系表中添加数据
        user = super().create(validated_data)

        # ② 判断是否设置默认密码，并且要实现密码加密保存
        password = validated_data.get('password')

        if not password:
            # 密码为 None 或 密码为 ''
            password = '123456abc'

        # 密码加密保存
        user.set_password(password)
        user.save()

        # 返回 user
        return user
```

3.3 指定管理员数据修改

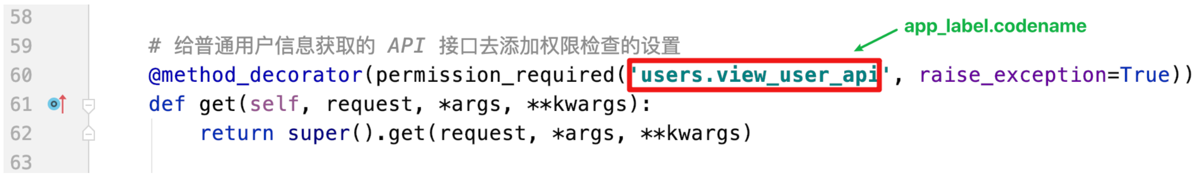


```python
    def update(self, instance, validated_data):
        password = validated_data.pop('password', None)
        # ① 修改管理员用户的数据(排除password)
        super().update(instance, validated_data)

        # ② 再来判断密码是否需要修改
        if password:
            instance.set_password(password)
            instance.save()

        return instance
```

下午课程核心内容

1. 权限检查设置 - permission_required 装饰器

在 Django 中，可以通过 permission_required 给每个 API 接口添加指定的权限检查设置。某个 API 使用 permssion_required 指定了对应的权限检查之后，当用户来访问该 API 时，permission_required 方法内部会先检查访问用户是否有对应的权限，如果有权限则 API 会被正常调用，如果没有权限则会抛出 PermissionDenied 异常。



```python
    # 给普通用户信息获取的 API 接口去添加权限检查的设置
    @method_decorator(permission_required('users.view_user_api', raise_exception=True))
    def get(self, request, *args, **kwargs):
        return super().get(request, *args, **kwargs)
```

2. 后台整体总结 - DRF 框架核心背记知识点

参考 `DRF 框架核心背记知识点.pdf` 文档.

3. 项目部署课程 - Nginx 服务器

Nginx 服务器在项目部署时的作用?：

1）作静态 Web 服务器：提供静态页面

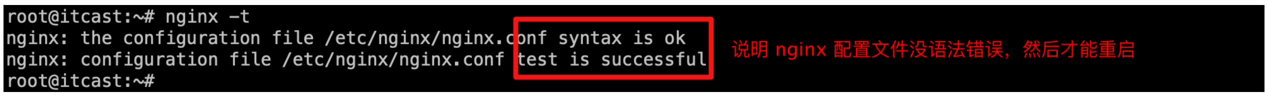
python -m http.server 8080

npm run dev

2）作反向代理服务器，实现负载均衡

检查 Nginx 配置文件命令：

```
nginx -t
```



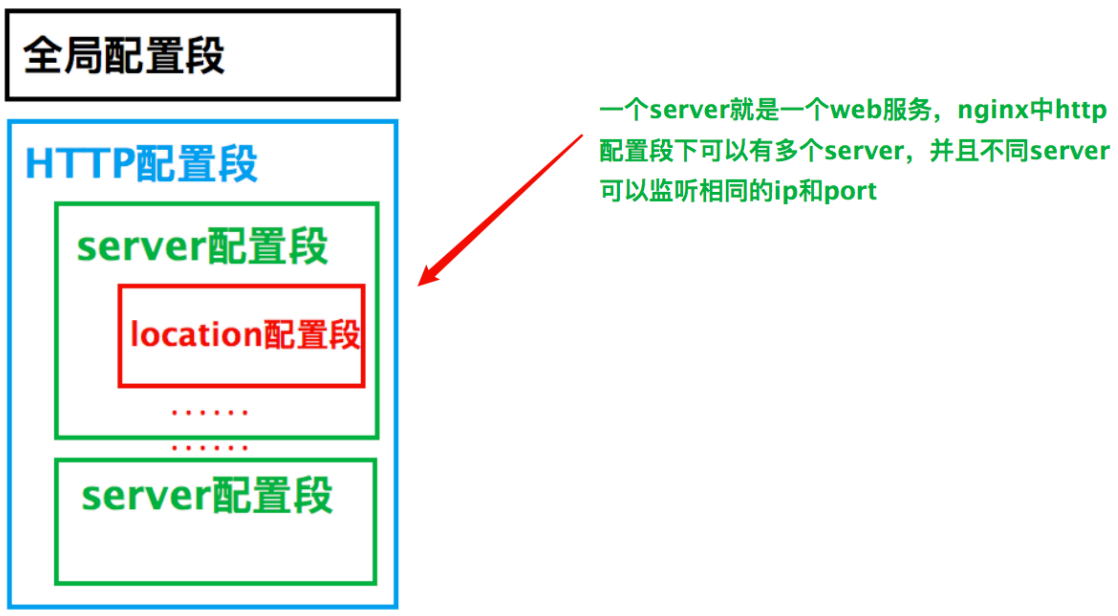
```
root@itcast:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@itcast:~#
```

说明 nginx 配置文件没语法错误，然后才能重启

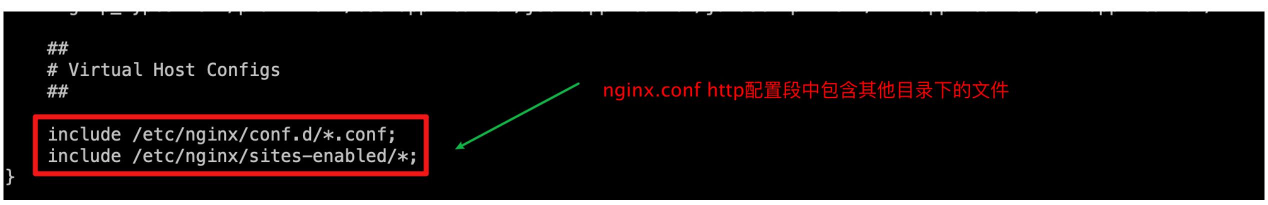
Nginx 配置核心目录：

```
配置目录：/etc/nginx
日志目录：/var/log/nginx
```

Nginx 配置文件结构：


全局配置段
HTTP配置段
server配置段
location配置段
......
......
server配置段
一个server就是一个web服务，nginx中http配置段下可以有多个server，并且不同server可以监听相同的ip和port

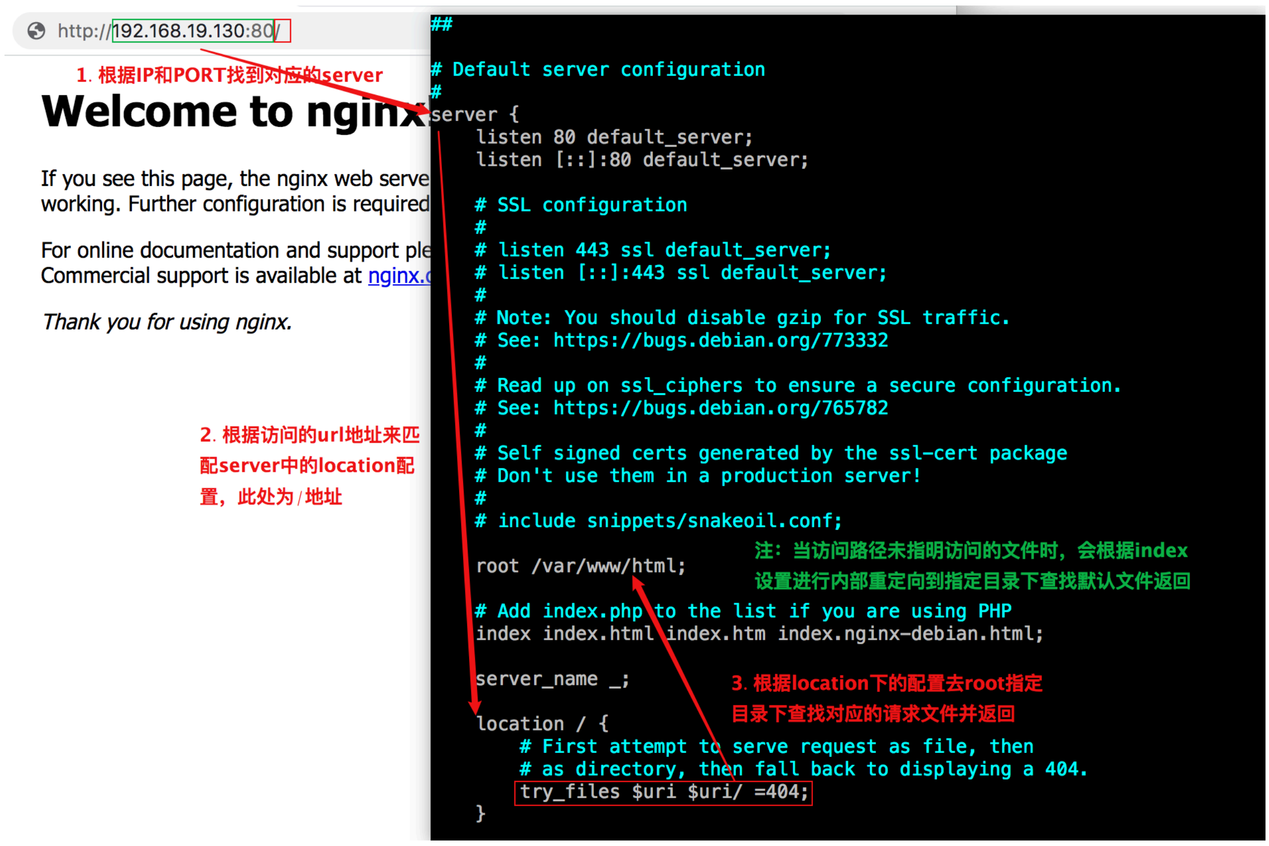
`nginx.conf 是 nginx 的主配置文件，nginx.conf 配置文件中只有 全局配置段 和 HTTP 配置段，` `server 配置段和 location 配置段是从其他文件中包含过来的`。



```
    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

nginx.conf http配置段中包含其他目录下的文件

Nginx 访问的基本原理：



http://192.168.19.130:80/

1. 根据IP和PORT找到对应的server

Welcome to nginx

If you see this page, the nginx web serve
working. Further configuration is required

For online documentation and support ple
Commercial support is available at nginx.c

Thank you for using nginx.

2. 根据访问的url地址来匹配server中的location配置，此处为/地址

```
##
# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
```

注：当访问路径未指明访问的文件时，会根据index设置进行内部重定向到指定目录下查找默认文件返回

3. 根据location下的配置去root指定目录下查找对应的请求文件并返回