# 第20章

## 20.1 将项目部署到Heroku

在本章中，你将给项目“学习笔记”设置样式，再将其部署到Heroku的服务器。部署这个应用后，你就能够从任何设备访问它，还可将工作成果与他人分享。本节将帮助你设置系统，为将项目部署到Heroku做好准备。

### 20.1.1 注册Heroku账户

要注册账户，可访问https://heroku.com/，并单击按钮Sign up。账户注册是免费的，Heroku提供了免费套餐，让你能够对在线部署的项目进行测试。

### 20.1.2 安装Heroku Toolbelt

要将项目部署到Heroku的服务器并对其进行管理，需要用到Heroku Toolbelt提供的工具。要安装最新的Heroku Toolbelt版本，请访问https://toolbelt.heroku.com/，并根据你使用的操作系统按相应的说明做。这需要执行一个终端命令或下载并运行安装程序。

### 20.1.3 安装必要的包

你还需安装很多包，以支持部署到在线服务器上的Django项目。为此，在一个活动的虚拟环境中执行下面这些命令：

(ll\_env)learning\_log$ pip install dj-database-url

(ll\_env)learning\_log$ pip install dj-static

(ll\_env)learning\_log$ pip install static3

(ll\_env)learning\_log$ pip install gunicorn

请每次执行一个命令，这样如果有包未能成功安装，你就能知道这一点（在Windows系统中，可能不会安装其中的一些包，因此如果安装某些包时出现错误消息，也不用担心，因为这不影响你将项目部署到Heroku）。

## 20.2 修订

Heroku项目存储在一个名为/app的目录中，在文件settings.py中，我们基于这一点来区分本地设置和Heroku设置。当你将Django项目推送到Heroku时，一开始将从临时目录构建它。如果只检查目录/app，项目将构建失败。

为修复这个问题，有一个简单的办法：修改settings.py中的if语句，使其除检查目录/app外，还检查目录/tmp。

### 20.2.1 检查目录/tmp

在本书中，settings.py中的if语句类似于下面这样：

*# Heroku设置*

**if** os**.**getcwd() **==** '/app':

import dj\_database\_url

下面是修订后的版本，它还检查目录/tmp：

*# Heroku设置*

cwd **=** os**.**getcwd()

**if** cwd **==** '/app' **or** cwd[:4] **==** '/tmp':

import dj\_database\_url

具体的临时目录随项目而异，它类似于下面这样：/tmp/build\_07bcdf3ae946245cce5541e622a3799a。上述代码检查当前目录的开头4个字符，并在这些字符为/tmp时使用Heroku设置。

这样修订后，部署项目时应该不会出现任何问题。如果你要使用WhiteNoise来管理静态文件，请阅读下一小节。

### 20.2.2 使用WhiteNoise来管理静态文件

WhiteNoise是一种给Django项目提供静态文件的更佳方式，就项目“学习笔记”而言，使用WhiteNoise带来的影响不大，但如果项目的访问流量很大，使用WhiteNoise将是明知的选择。

如果你还没有完成第20章的第二部分（即部署项目），可使用WhiteNoise来构建项目。为此，在20.2.3节中，不安装书中指定的包，而安装下面这些包：

(ll\_env)learning\_log$ pip install dj-database-url

(ll\_env)learning\_log$ pip install whitenoise

(ll\_env)learning\_log$ pip install gunicorn

这样做后，文件requirements.txt将与原来稍有不同：其中列出的是whitenoise，而不是dj-static和static3。

在文件settings.py的Heroku设置部分，需要再添加一行，即STATICFILES\_STORAGE设置，如下面的最后一行所示：

*# Heroku设置*

cwd **=** os**.**getcwd()

**print**("--- CWD ---\n", cwd, "\n---\n")

**if** cwd **==** '/app' **or** cwd[:4] **==** '/tmp':

import dj\_database\_url

DATABASES **=** {

'default': dj\_database\_url**.**config(default**=**'postgres://localhost')

}

*# 让request.is\_secure()承认X-Forwarded-Proto头*

SECURE\_PROXY\_SSL\_HEADER **=** ('HTTP\_X\_FORWARDED\_PROTO', 'https')

*# 只允许heroku托管项目*

ALLOWED\_HOSTS **=** ['\*']

DEBUG **=** True

*# 静态资产配置*

BASE\_DIR **=** os**.**path**.**dirname(os**.**path**.**abspath(\_\_file\_\_))

STATIC\_ROOT **=** 'staticfiles'

STATICFILES\_DIRS **=** (

os**.**path**.**join(BASE\_DIR, 'static'),

)

STATICFILES\_STORAGE **=** 'whitenoise.django.GzipManifestStaticFilesStorage'

最后，在文件wsgi.py中，需要使用WhiteNoise（而不是Cling）：

"""

项目learning\_log的WSGI配置

它将WSGI可调用对象暴露为模块级变量application

有关这个文件的更详细信息，请参阅https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

from whitenoise.django import DjangoWhiteNoise

os**.**environ**.**setdefault("DJANGO\_SETTINGS\_MODULE", "learning\_log.settings")

application **=** get\_wsgi\_application()

application **=** DjangoWhiteNoise(application)

完成这些修改后，项目便能在Heroku服务器上很好的运行，即便访问流量很大。

如果你使用dj-static和static3构建了项目，可卸载这些库，并安装whitenoise：

(ll\_env)learning\_log$ pip uninstall dj-static

(ll\_env)learning\_log$ pip uninstall static3

(ll\_env)learning\_log$ pip install whitenoise

然后，再次执行命令pip freeze > requirements.txt，并更新settings.py和wsgi.py；再使用Git提交所做的修改，并将项目部署到Heroku做好准备：

(ll\_env)learning\_log$ pip freeze > requirements.txt

(ll\_env)learning\_log$ git commit -am "Updated project to use WhiteNoise."

(ll\_env)learning\_log$ git push heroku master

## 20.3 资源

如果你想更深入地了解部署过程，可参阅如下资源：

* Getting Started on Heroku with Python。
* Django and Static Assets。
* Configuring Django Apps for Heroku。
* The WhiteNoise project。

## 20.4 问题

在本书中，这个项目是最复杂的，但看到自己的工作成果被部署到互联网上，真的很有成就感。真心希望你能成功地完成部署，但如果你在部署过程中遇到无法解决的问题，可通过如下方式与作者联系：

* Email：ehmatthes@gmail.com。
* Twitter：@ehmatthes。