# 第18章

在本章中，你将使用Django来创建“学习笔记”，这是一个简单的Web应用，让你能够记录已学习的有关特定主题的内容。本节将帮助你搭建虚拟环境。

## 创建虚拟环境

要使用Django，最好先建立一个虚拟环境。虚拟环境是系统中的一个位置，你可在其中安装供特定项目使用的包，并将这些包与系统的其他部分隔离。通过使用这种方法，系统中可有很多项目，同时不用担心这些项目的包相互干扰。每个项目都可根据需要拥有特定版本的包，而你可更新任何项目的包，而不影响其他项目。

要为“学习笔记”搭建虚拟环境，务必先在系统中创建一个空文件夹，并将其命名为learning\_log。

如果你使用的是Python 3，可使用如下命令来建立虚拟环境：

learning\_log$ python -m venv ll\_env

learning\_log$

这个命令运行模块venv，并使用它创建一个名为ll\_env的虚拟环境。

如果这样做管用，请跳到“激活虚拟环境”一节；如果不管用，请安装virtualenv。

## 激活虚拟环境

建立虚拟环境后，需要激活它。在Linux和OS X系统中，请使用如下命令来激活虚拟环境：

learning\_log$ source ll\_env/bin/activate

(ll\_env)learning\_log$

在Windows系统中，请执行如下命令：

learning\_log> ll\_env\Scripts\activate

(ll\_env)learning\_log>

这个命令运行ll\_env/bin中的脚本activate。环境处于活动状态时，环境名将包含在括号内，这意味着你可在环境中安装包，还可使用环境中已安装的包。你在环境ll\_env中安装的包仅在该环境处于活动状态时才可用；如果发生错误，请核实终端提示符的开头是否有(ll\_env)，如果没有，请再次执行命令activate。

### 停止使用虚拟环境

要停止使用虚拟环境，可执行命令deactivate：

(ll\_env)learning\_log$ deactivate

learning\_log$

如果你关闭运行虚拟环境的终端，虚拟环境也将不再处于活动状态。

## 安装virtualenv

如果你使用的是较早的Python版本，或者系统没有正确地设置，导致无法使用模块venv，可安装virtualenv包。要安装virtualenv，请执行如下命令：

$ pip install --user virtualenv.

在你的系统中，pip的用法可能稍有不同；要获取有关pip用法的帮助，请参阅安装Python包。

如果你使用的是Linux系统，且上面的做法不管用，可使用系统的包管理器来安装virtualenv。例如，要在Ubuntu系统中安装virtualenv，可使用命令sudo apt-get install python-virtualenv。

### 使用virtualenv创建虚拟环境

在终端中切换到目录learning\_log，并像下面这样创建一个虚拟环境：

learning\_log$ virtualenv ll\_env

New python executable in ll\_env/bin/python

Installing setuptools, pip...done.

learning\_log$

如果这样做管用，请跳到“激活虚拟环境”一节。

如果你的系统中安装了多个Python版本，需要指定virtualenv要使用的版本。例如，命令virtualenv ll\_env --python=python3创建一个使用Python 3的虚拟环境。

## 安装Django

在虚拟环境中，可使用pip来安装Django：

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

Installing collected packages: Django

Successfully installed Django

Cleaning up...

(ll\_env)learning\_log$

由于是在虚拟环境中，因此在所有的系统中，安装Django的命令都相同：不需要指定标志--user，也无需使用python -m pip install django这样较长的命令。

别忘了，Django仅在虚拟环境处于活动状态时才可用。

### 安装特定的Django版本

Django 2.0做了一些修改，这些修改会影响项目“学习笔记”。如果你使用最新的Django版本，就需按后面的介绍修订这个项目；但你也可使用前一个版本，这样就无需做任何修订。

要安装特定的版本，可在使用pip时进行指定：

(ll\_env)learning\_log$ pip install Django==1.11

Collecting Django==1.11

Installing collected packages: pytz, Django

Successfully installed Django-1.11 pytz-2017.3

即便你之前安装了其他版本，这个命令也管用，因为当你命令pip安装特定的Django版本时，它将首先卸载以前安装的版本。

## 修订

不久前发布了Django 2.0，其中很多变化都对项目“学习笔记”有影响。下面简要地介绍一下总体变化，再列出对本章代码的修订。

### 概述

对Django最新的变化概述如下：

* 定义外键关系时，需要显式地将参数on\_delete设置为models.CASCADE；在以前的版本中，会默认地设置。用户删除其学习笔记中的主题时，参数on\_delete告诉Django如何处理该主题中的条目；如果该参数的值为CASCADE，Django将在删除指定主题的同时删除所有与之相关联的条目。
* 提供了一种为应用的URL定义命名空间的新方法，即不在项目的urls.py文件中设置命名空间，而在每个应用的urls.py中设置命名空间。
* 提供了一种更简单的URL定义方法。在以前的版本中，对于每个包含标识符的URL（如http://localhost:8000/topics/1/），都必须为其定义一个正则表达式。并非必须使用这种新方法，但你可能喜欢这种新方法，进而想了解它。
* 将函数reverse()从模块django.core.urlresolvers移到了模块django.urls。
* Django 2.0只支持Python 3，因此如果你使用的是Python 2.7，必须安装Django 1.11。

### 检查你当前使用的是哪个版本的Django

命令pip freeze列出你在虚拟环境中安装的每个包的版本。要获悉你使用的是哪个版本的Django，确保你处于活动的虚拟环境中，再执行如下命令：

(ll\_env)learning\_log$ pip freeze

Django==2.0

pytz==2017.3

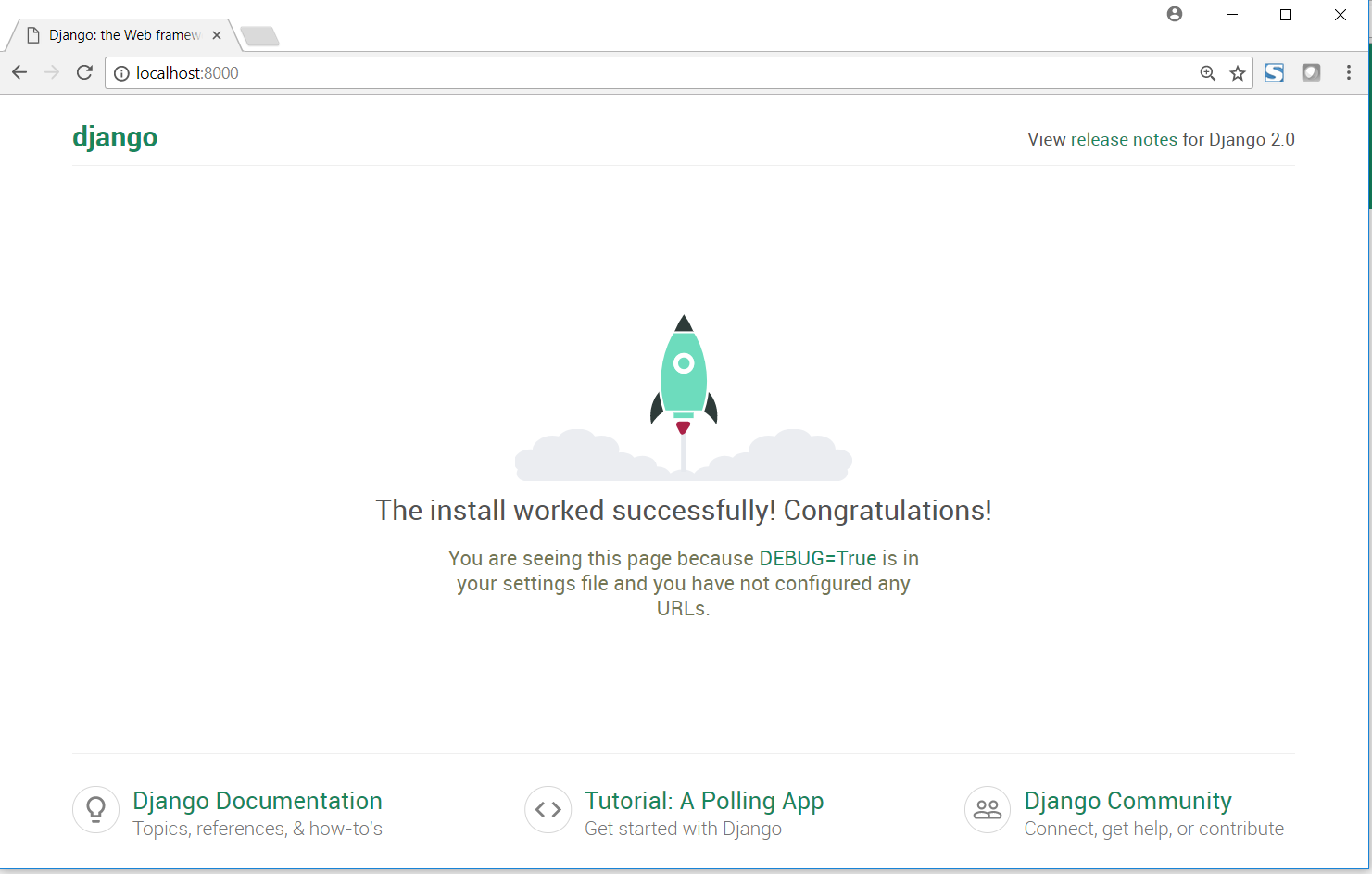
如果你要使用前一个Django版本，以便能够忽略后面列出的修订，可按前面介绍的方式安装Django 1.11。

### 具体修订

下面列出了对第18章所做的具体修订。所做的修改好像很多，但着手完成这个项目时，你将发现很多修改都遵循相同的模式。如果你遇到无法解决的问题，请向作者需求帮助（ehmatthes@gmail.com或@ehmatthes）。

#### 图18-1

Django项目的默认主页有变，现在的默认主页非常漂亮！



#### 18.2.4节定义模型Entry

将代码行topic = models.ForeignKey(Topic)替换为topic = models.ForeignKey(Topic, on\_delete=models.CASCADE)。

这告诉Django，当一个主题被删除时，应同时删除与该主题相关的所有条目。这被称为级联删除（cascading delete）。

#### 18.3.1节映射URL

在Django 2.0中新建项目时，该项目的主urls.py文件默认包含如下内容：

Django 2.0引入了函数path()，这简化了定义URL的方式。使用url()的方法依然管用，但接下来的修订演示了如何使用新函数path()。

下面演示了如何包含应用程序learning\_logs的URL：

from django.contrib import admin

from django.urls import path

urlpatterns = [

path('admin/', admin.site.urls),

]

在Django 2.0中，函数include()位于模块django.urls中。新的URL路由方法不需要使用正则表达式，因此无需在URL字符串开头包含脱字符。另外，调用函数include()时，不需要指定参数namespace，因为命名空间将在应用程序的urls.py中指定。

文件learning\_logs/urls.py应类似于下面这样：

"""定义learning\_logs的URL模式"""

from django.urls import path

from . import views

app\_name = 'learning\_logs'

urlpatterns = [

# 主页

path('', views.index, name='index'),

]

需要从模块django.urls中导入新增的函数path()。在Django 2.0中，应用程序的命名空间是在应用程序的urls.py中通过变量app\_name定义的。最后，主页的URL模式更简单，不需要定义正则表达式。

#### 18.4.2节显示所有主题的页面的URL模式

显示所有主题的页面的URL模式应类似于下面这样：

urlpatterns = [

--snip--

# 显示所有的主题

path('topics/', views.topics, name='topics'),

]

18.4.3节显示特定主题的页面的URL模式

显示特定主题的页面的URL模式应类似于下面这样：

urlpatterns = [

--snip--

# 特定主题的详细页面

path('topics/<int:topic\_id>/', views.topic, name='topic'),

]

使用这种基于path()的新方法时，你首先确定自己希望URL是什么样的：

http://localhost:8000/topics/5/

再编写一个表示URL的通用字符串，并将所有可变部分放在尖括号内。在尖括号内，你指定了用于存储URL中可变部分的变量的类型（这里是int），还有变量的名称。与通过编写正则表达式来捕获URL中的值相比，这要简单得多。

# 第19章修订

#### 19.1.1节的URL模式new\_topic

URL模式new\_topic应类似于下面这样：

urlpatterns = [

--snip--

# 用于添加新主题的页面

path('new\_topic/', views.new\_topic, name='new\_topic'),

]

#### 19.1.1节的视图函数new\_topic()

函数reverse()移到了了模块django.urls中，因此必须在这个视图函数中修改导入这个函数的语句：

from django.shortcuts import render

from django.http import HttpResponseRedirect

from django.urls import reverse

#### 19.1.2节的URL模式new\_entry

URL模式new\_entry应类似于下面这样：

urlpatterns = [

--snip--

# 用于添加新条目的页面

path('new\_entry/<int:topic\_id>/', views.new\_entry, name='new\_entry'),

]

#### 19.1.3节的URL模式edit\_entry

URL模式edit\_entry应类似于下面这样：

urlpatterns = [

--snip--

# 用于编辑条目的页面

path('edit\_entry/<int:entry\_id>/', views.edit\_entry, name='edit\_entry'),

]

#### 19.2.1节包含应用程序users的URL的代码

包含应用程序users的URL的代码行应类似于下面这样：

from django.urls import path, include

from django.contrib import admin

urlpatterns = [

path('admin/', admin.site.urls),

path('users/', include('users.urls')),

path('', include('learning\_logs.urls')),

]

与包含应用程序learning\_logs的URL时一样，指定了一个表示基础URL的简单字符串，这里为'users/'。同样，这里不需要指定参数namespace，因为命名空间将在文件users/urls.py中指定。

#### 19.2.2节的登录页面URL模式

文件users/urls.py应类似于下面这样：

"""为应用程序users定义URL模式"""

from django.urls import path

from django.contrib.auth.views import login

from . import views

app\_name = 'users'

urlpatterns = [

# 登录页面

path('login/', login, {'template\_name': 'users/login.html'},

name='login'),

]

这里定义了变量app\_name，给应用程序users的URL指定了命名空间；另外，还使用函数path()定义了登录页面的URL模式。

#### 19.2.3节的注销页面URL模式

注销页面的URL模式应类似于下面这样：

urlpatterns = [

--snip--

# 注销页面

path('logout/', views.logout\_view, name='logout'),

]

#### 19.2.3节的视图函数logout\_view()

在users/views.py中，需要从模块django.urls导入函数reverse()：

from django.urls import reverse

#### 19.2.4节的注册页面的URL模式

注册页面的URL模式应类似于下面这样：

urlpatterns = [

--snip--

# 注册页面

path('register/', views.register, name='register'),

#### 19.3.2节修改模型Topic

在learning\_logs/models.py中，在主题和用户之间定义外键关系的代码行应类似于下面这样：

owner = models.ForeignKey(User, on\_delete=models.CASCADE)

这告诉Django，用户被删除时，应同时删除与该用户相关联的所有主题。