**Django 起步文档**

# Django 初探

因为Django是在快节奏的新闻编辑室环境下开发的，它旨在使Web开发任务变得简单快捷。 这是一个关于如何用Django编写数据库驱动的Web应用程序的非正式概述。 本文档的目的是为您提供足够的技术细节来了解Django的工作原理，但这不是一个教程或参考！ 当您准备开始一个项目时，您可以从教程开始，或者直接查看更详细的文档。

## 设置模型

虽然您可以使用没有数据库的Django，但它附带一个对象关系映射中间件，您可以在其中以Python代码描述数据库的布局。

数据模型语法提供了许多丰富的代表模型的方法 - 到目前为止，它一直在解决多年的数据库模式问题。以下是一个简单的例子：

**from** django.db **import** models

**class Reporter**(models**.**Model):

full\_name **=** models**.**CharField(max\_length**=**70)

**def \_\_str\_\_**(self): # \_\_unicode\_\_ on Python 2

**return** self**.**full\_name

**class Article**(models**.**Model):

pub\_date **=** models**.**DateField()

headline **=** models**.**CharField(max\_length**=**200)

content **=** models**.**TextField()

reporter **=** models**.**ForeignKey(Reporter, on\_delete**=**models**.**CASCADE)

**def \_\_str\_\_**(self): # \_\_unicode\_\_ on Python 2

**return** self**.**headline

## 安装

接下来，运行Django命令行实用程序自动创建数据库表：

python manage.py migrate

migrate命令用于查看所有可用的模型，并在数据库中创建不存在的表以及可选地提供更丰富的模式控制的表。

## 使用方便的API

接着，你就可以使用一个便捷且功能丰富的[Python API](http://python.usyiyi.cn/documents/django_182/topics/db/queries.html)来访问你的数据。这些API是即时创建的，不需要代码生成：

Import the models we created from our "news" app

**>>> from** news.models **import** Reporter, Article

# No reporters are in the system yet.

**>>>** Reporter**.**objects**.**all()

[]

# Create a new Reporter.

**>>>** r **=** Reporter(full\_name**=**'John Smith')

# Save the object into the database. You have to call save() explicitly.

**>>>** r**.**save()

# Now it has an ID.

**>>>** r**.**id

1

## 动态的管理界面：它不只是一个框架 ，而是一所完整的房子

一旦你的模型定义完毕之后，Django能自动创建一个专业的、可以用于生产环境的 [管理界面](http://python.usyiyi.cn/documents/django_182/ref/contrib/admin/index.html) – 可以让认证的用户添加、修改和删除对象的一个站点。只需简单地在admin site中注册你的模型即可：

**#** models.py

**from** django.db **import** models

**class Article**(models**.**Model):

pub\_date **=** models**.**DateField()

headline **=** models**.**CharField(max\_length**=**200)

content **=** models**.**TextField()

reporter **=** models**.**ForeignKey(Reporter, on\_delete**=**models**.**CASCADE)

# admin.py

**from** django.contrib **import** admin

**from** . **import** models

admin**.**site**.**register(models**.**Article)

这里的设计理念是你的网站由一个员工、客户或者可能是你自己去编辑 —— 而你不想仅仅为了管理内容而去创建后台界面。

创建Django应用的一个典型工作流程是创建模型然后尽快地让admin sites启动和运行起来， 这样您的员工（或客户）能够开始录入数据。 然后，才开发展现数据给公众的方式。

## 设计你的URLs

对于高质量的Web 应用来说，使用简洁、优雅的URL 模式是一个非常值得重视的细节。Django鼓励使用漂亮的URL设计且不会像.php或.asp一样把乱七八糟的东西放到URLs里面，.

为了给一个应用设计URLs，你需要创建一个叫做[URLconf](http://python.usyiyi.cn/documents/django_182/topics/http/urls.html)的Python模块。这其实是你应用的目录，它包含URL模式与Python回调函数间的一个简单映射。 URLconfs 还用作从Python代码中解耦URLs。

下面是针对上面Reporter/Article例子URLconf 可能的样子：

**#** urls.py

**from** django.conf.urls **import** url

**from** . **import** views

urlpatterns **=** [

url(r'^articles/([0-9]{4})/$', views**.**year\_archive),

url(r'^articles/([0-9]{4})/([0-9]{2})/$', views**.**month\_archive),

url(r'^articles/([0-9]{4})/([0-9]{2})/([0-9]+)/$', views**.**article\_detail),

]

上面的代码将URLs映射作为简单的[正则表达式](https://docs.python.org/howto/regex.html)映射到Python的回调函数（视图）。正则表达式通过圆括号来“捕获”URLs中的值。当一个用户请求一个页面时，Django将按照顺序去匹配每一个模式，并停在第一个匹配请求的URL上。（如果没有匹配到，Django将调用一个特殊的404视图。）整个过程是极快的，因为正则表达式在加载时就已经编译好了。

一旦有一个正则表达式匹配上了，Django 将导入和调用对应的视图，它其实就是一个简单的Python函数。每个视图将得到一个request对象 —— 它包含了request 的metadata(元数据) —— 和正则表达式所捕获到的值。

例如，如果一个用户请求了URL “/articles/2005/05/39323/”，Django将调用函数news.views.article\_detail(request, '2005', '05', '39323')。

## 编写视图

每个视图只负责两件事中的一件：返回一个包含请求的页面内容的[HttpResponse](http://python.usyiyi.cn/documents/django_182/ref/request-response.html#django.http.HttpResponse)对象， 或抛出一个异常如[Http404](http://python.usyiyi.cn/documents/django_182/topics/http/views.html#django.http.Http404)。剩下的就靠你了。

通常，一个视图会根据参数来检索数据、加载一个模板然后使用检索出来的数据渲染模板。下面是上文year\_archive的一个视图例子：

**#** views.py

**from** django.shortcuts **import** render

**from** .models **import** Article

**def year\_archive**(request, year):

a\_list **=** Article**.**objects**.**filter(pub\_date\_\_year**=**year)

context **=** {'year': year, 'article\_list': a\_list}

**return** render(request, 'news/year\_archive.html', context)

这个例子使用了Django的[模板系统](http://python.usyiyi.cn/documents/django_182/topics/templates.html)，它具有几个强大的功能，但是仍然努力做到了即使对于非编程人员也能保持足够的简单。

## 设计模板

上面的代码加载news/year\_archive.html模板。

Django有一个模板搜索路径，它允许您最大限度地减少模板之间的冗余。在你的Django设置中，你可以通过[DIRS](http://python.usyiyi.cn/documents/django_182/ref/settings.html#std:setting-TEMPLATES-DIRS)指定一个查找模板的目录列表。如果这个模板没有在第一个目录中，那么它会去查找第二个，以此类推。

让我们假设news/year\_archive.html模板已经找到。它看起来可能是下面这个样子：

<!-- year\_archive.html -->

**{% extends** "base.html" **%}**

**{% block** title **%}**Articles for **{{** year **}}{% endblock %}**

**{% block** content **%}**

<h1>Articles for **{{** year **}}**</h1>

**{% for** article **in** article\_list **%}**

<p>**{{** article.headline **}}**</p>

<p>By **{{** article.reporter.full\_name **}}**</p>

<p>Published **{{** article.pub\_date**|date**:"F j, Y" **}}**</p>

**{% endfor %}**

**{% endblock %}**

变量使用两对大括号包围。{{ article.headline }}表示“输出 article的headline属性”。但是点符号不仅用于属性查找。它们还用于字典的键值查找、索引查找和函数调用。

注意{{ article.pub\_date|date:"F j, Y" }}使用Unix风格的“管道”（“|”字符）。这叫做模板过滤器，它是过滤变量值的一种方式。在本例中，date过滤器格式化Python的datetime对象成给定的格式（正如在PHP中日期函数）。

你可以无限制地串联使用多个过滤器。你可以编写[自定义的目标过滤器](http://python.usyiyi.cn/documents/django_182/howto/custom-template-tags.html#howto-writing-custom-template-filters)。你可以编写[自定义的模板标签](http://python.usyiyi.cn/documents/django_182/howto/custom-template-tags.html)，在幕后运行自定义的Python代码。

最后，Django使用“模板继承”的概念。这就是{% extends "base.html" %}所做的事。它表示“首先载入‘base’ 模板，该模板中定义了一系列block，然后使用接下来的（存在于继承模板）blocks填充这些blocks”。简而言之，模板继承让你大大减少模板间的冗余内容：每个模板只需要定义它独特的部分。

下面是“base.html”模板可能的样子，它使用了[静态文件](http://python.usyiyi.cn/documents/django_182/howto/static-files/index.html)：

<!-- base.html -->

**{% load** staticfiles **%}**

<html>

<head>

<title>**{% block** title **%}{% endblock %}**</title>

</head>

<body>

<img src**=**"**{% static** "images/sitelogo.png" **%}**" alt**=**"Logo" />

**{% block** content **%}{% endblock %}**

</body>

</html>

简单地说，它定义网站的外观（使用网站的logo ），并提供“空洞”让子模板填充。这使站点的重构变得非常容易，只需改变一个文件 —— base模板。

它还可以让你利用不同的基础模板并重用子模板创建一个网站的多个版本。Django 的创建者已经利用这一技术来创造了显著不同的手机版本的网站 —— 只需创建一个新的基础模板。

请注意，如果你喜欢其它模板系统，你可以不使用Django的模板系统。  虽然Django的模板系统与Django的模型层集成得特别好，但并没有强制你使用它。同理，你也可以不使用Django的数据库API。 你可以使用任何你想要的方法去操作数据，包括其它数据库抽象层，读取 XML 文件或者直接从磁盘中读取文件。Django的每个组成部分 —— 模型、视图和模板都可以解耦。

## 编写你的第一个 Django 程序

让我们通过例子来学习。

在本教程中，我们将引导您创建一个基本的投票应用。

它将包含两部分：

* 一个公共网站，可让人们查看投票的结果和让他们进行投票。
* 一个管理网站，可让你添加、修改和删除投票项目。

我们假设你已经 安装了 Django 。你可以运行以下命令来验证是否已经安装了 Django 和运行着的版本号：

python -m django --version

如果Django已经安装，你应该看到安装的版本。 如果还没有安装，你会看到一个“No module named django”的错误。

本教程是针对Django 1.8和Python 3.2或更高版本编写的。如果Django版本不符，可以通过当前页面右下角的版本转换器查看适用于你所使用的版本的Django教程，或者把Django升级到最新的版本。 如果你还在使用2.7版本的Python，你将需要按照注释中的内容稍微调整一下示例代码。

关于如何删除旧版本的Django并安装一个新的，请参见[如何安装 Django](http://python.usyiyi.cn/documents/django_182/topics/install.html)中的建议。

你可以在下列地址寻求帮助：

如果您在学习本教程的过程中遇到问题，请向[django-users](http://python.usyiyi.cn/documents/django_182/internals/mailing-lists.html#django-users-mailing-list)发送邮件或者访问[#django on irc.freenode.net](irc://irc.freenode.net/django)与其他可能会帮助你的Django用户交流。

## 创建一个项目

如果这是你第一次使用Django，你需要完成一些初始化设置。 你需要自己用代码来创建一个Django[项目](http://python.usyiyi.cn/documents/django_182/glossary.html#term-project) ——一个Django框架开发的网站，创建项目后我们需要的配置的东西，包括数据库的配置、针对Django的配置选项和app的配置选项。

在命令行（终端）中，cd（例如cd exam)到你想要用来保存代码的目录，然后运行如下命令：

django-admin startproject mysite

这将会在你的当前目录下生成一个 mysite 目录。如果它不能正常工作，请查看[运行django-admin遇到的问题](http://python.usyiyi.cn/documents/django_182/faq/troubleshooting.html#troubleshooting-django-admin)。

**注意**

你给项目命名时，项目名称不能和Python或Django的内部组件名称同名。特别地，你应该避免使用类似django（与Django自身冲突）或 test（与Python内建的包冲突）这样的名称。

### 代码应该存在哪里？

如果你曾经学过普通的旧式的PHP（没有使用过现代的框架），你可能习惯于将代码放在Web服务器的文档根目录下（例如/var/www）。但是对于Django，你不该这么做。将Python代码放在你的Web服务器的根目录不是个好主意，因为它可能会产生让人们在网上看到你的代码的风险。 这样不安全。

将你的代码放置在Web服务器根目录**以外**的地方，例如/home/mycode。

让我们看一下[startproject](http://python.usyiyi.cn/documents/django_182/ref/django-admin.html#django-admin-startproject)生成了什么：

mysite**/**

manage**.**py

mysite**/**

**\_\_init\_\_.**py

settings**.**py

urls**.**py

wsgi**.**py

这些文件是：

* 外层的mysite/根目录仅仅是项目的一个容器。它的命名对Django无关紧要；你可以把它重新命名为任何你喜欢的名字。
* manage.py：一个命令行工具，可以使你用多种方式对Django项目进行交互。 你可以在[django-admin和manage.py](http://python.usyiyi.cn/documents/django_182/ref/django-admin.html)中读到关于manage.py的所有细节。
* 内层的mysite/目录是你的项目的真正的Python包。它是你导入任何东西时将需要使用的Python包的名字（例如 mysite.urls）。
* mysite/\_\_init\_\_.py：一个空文件，它告诉Python这个目录应该被看做一个Python包。 （如果你是一个Python初学者，[关于包的更多内容](https://docs.python.org/tutorial/modules.html" \l "packages)请阅读Python的官方文档）。
* mysite/settings.py：该Django 项目的设置/配置。[Django 设置](http://python.usyiyi.cn/documents/django_182/topics/settings.html) 将告诉你这些设置如何工作。
* mysite/urls.py：该Django项目的URL声明；你的Django站点的“目录”。 你可以在[URL 转发器](http://python.usyiyi.cn/documents/django_182/topics/http/urls.html) 中阅读到关于URL的更多内容。
* mysite/wsgi.py：用于你的项目的与WSGI兼容的Web服务器入口。 更多细节请参见[如何利用WSGI进行部署](http://python.usyiyi.cn/documents/django_182/howto/deployment/wsgi/index.html)。

### 开发服务器

让我们验证一下你的Django项目是否工作。 如果你在外层的mysite目录下，那么进入这个目录，然后运行以下命令：

python manage.py runserver

你将看到命令行下输出了以下内容：

Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.

Run 'python manage.py migrate' to apply them.

April 05, 2017 - 15:50:53

Django version 1.9, using settings 'mysite.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

这表明你已经启动了Django开发服务器，一个用纯Python写的轻量级Web服务器。 我们在Django中内置了它，这样你就可以在不配置用于生产环境的服务器 —— 例如Apache —— 的情况下快速开发出产品，直到你准备好上线。

请注意：**不要**在任何生产环境使用这个服务器。它仅仅是用于在开发中使用。（我们的重点是编写Web框架，非Web服务器。）

既然服务器已经运行，请用你的浏览器访问 <http://127.0.0.1:8000/>。在淡蓝色背景下，你将看到一个“Welcome to Django”的页面。 它运行成功了！

### 更改端口

默认情况下，[runserver](http://python.usyiyi.cn/documents/django_182/ref/django-admin.html#django-admin-runserver)命令在内部IP的8000端口启动开发服务器。

如果你需改变服务器的端口，把要使用的端口作为一个命令行参数传递给它。 例如，这个命令在8080端口启动服务器：

python manage.py runserver 8080

如果你需改变服务器的IP地址，把IP地址和端口号放到一起。 因此若要监听所有的外网IP，请使用（如果你想在另外一台电脑上展示你的工作，会非常有用）：

python manage.py runserver 0.0.0.0:8080

开发服务器的所有文档可以在[runserver](http://python.usyiyi.cn/documents/django_182/ref/django-admin.html#django-admin-runserver)的参考手册中找到。

### [runserver](http://python.usyiyi.cn/documents/django_182/ref/django-admin.html#django-admin-runserver)的自动重载

开发服务器会根据需要自动重新载入Python代码。 你不必为了使更改的代码生效而重启服务器。 然而，一些行为比如添加文件，不会触发服务器的重启，所以在这种情况下你需要手动重启服务器。

## 创建Polls应用

现在，你的开发环境 —— 一个“项目” —— 已经建立起来，你将开始在上面做一些东西。

你编写的每个Django应用都是遵循特定约定且包含一个Python包。 Django自带一个工具，它可以自动生成应用的基本目录结构，这样你就能专心于书写代码而不是创建目录。

### 项目 vs. 应用

项目和应用之间有什么不同？ 应用是一个Web应用程序，它完成具体的事项 —— 比如一个博客系统、一个存储公共档案的数据库或者一个简单的投票应用。 项目是一个特定网站中相关配置和应用的集合。一个项目可以包含多个应用。一个应用可以运用到多个项目中去。

你的应用可以放在[Python path](https://docs.python.org/tutorial/modules.html#the-module-search-path)上的任何位置。在本教程中，我们将在你的manage.py文件同级目录创建我们的投票应用，以便可以将它作为顶层模块导入，而不是mysite的子模块。

要创建您的应用程序，请确保您与manage.py在同一目录中，并键入以下命令：

python manage.py startapp polls

这将创建一个目录polls，它的结构如下：

polls**/**

**\_\_init\_\_.**py

admin**.**py

apps**.**py

migrations**/**

**\_\_init\_\_.**py

models**.**py

tests**.**py

views**.**py

我们的投票应用将基于这个目录结构。

## 创建第一个视图文件

让我们来编写第一个视图。 打开**polls/views.py**文件并将以下Python代码写入：

views.py

**from** django.http **import** HttpResponse

**def index**(request):

**return** HttpResponse("Hello, world. You're at the polls index.")

这可能是Django中最简单的视图。 为了能够调用这个视图，我们需要将这个视图映射到URL上 —— 利用一个URLconf。

为了在投票应用目录内部创建URLconf，需要创建一个**urls.py**文件。你的应用的目录现在看起来应该像这样：

polls**/**

**\_\_init\_\_.**py

admin**.**py

apps**.**py

migrations**/**

**\_\_init\_\_.**py

models**.**py

tests**.**py

urls**.**py

views**.**py

在**polls/urls.py**文件中键入如下代码：

# polls/urls.py

**from** django.conf.urls **import** url

**from** . **import** views

urlpatterns **=** [

url(r'^$', views**.**index, name**=**'index'),

]

下一步，让主URLconf可以链接到**polls.urls**模块。在**mysite/urls.py**中插入一个**[include()](http://python.usyiyi.cn/documents/django_182/ref/urls.html" \l "django.conf.urls.include" \o "django.conf.urls.include)**：

# mysite/urls.py

**from** django.conf.urls **import** include, url

**from** django.contrib **import** admin

urlpatterns **=** [

url(r'^polls/', include('polls.urls')),

url(r'^admin/', admin**.**site**.**urls),

]

和你所看到的不一样？

如果你看到在**urlpatterns**的定义前有**admin.autodiscover()**，你可能使用的是不符合本教程的Django版本。你需要换一个老版本的教程或者换一个新版的Django。

现在我们已经将一个**index**视图关联到URLconf中。在你的浏览器中浏览 <http://localhost:8000/polls/> , 你会看到 “Hello, world. You’re at the polls index.”, 正如你在**index** 视图中定义的那样.

[**url()**](http://python.usyiyi.cn/documents/django_182/ref/urls.html#django.conf.urls.url)函数具有四个参数：两个必需的**regex**和 **view**，以及两个可选的**kwargs**和**name**。 到这里，就可以研究下这些参数的含义了：

### [url()](http://python.usyiyi.cn/documents/ref/urls/#django.conf.urls.url) 参数：regex[¶](http://python.usyiyi.cn/documents/django_182/intro/tutorial03.html" \l "url-argument-regex" \o "Permalink to this headline)

术语“regex”是“regular expression(正则表达式)”的常用的一个缩写，是一种用来匹配字符串中模式的语法，在这里是URL模式。 Django从第一个正则表达式开始，依次将请求的URL与每个正则表达式进行匹配，直到找到匹配的那个为止。

请注意，这些正则表达式不会检索URL中GET和POST的参数以及域名。 例如，对于**http://www.example.com/myapp/**请求，URLconf 将查找**myapp/**。对于**http://www.example.com/myapp/?page=3**请求，URLconf 也将查找**myapp/**。

如果你需要正则表达式方面的帮助，参见[Wikipedia 条目](http://en.wikipedia.org/wiki/Regular_expression) 和**[re](https://docs.python.org/3/library/re.html" \l "module-re" \o "(in Python v3.4))** 模块的文档。另外，O’Reilly出版Jeffrey Friedl编写的“Mastering Regular Expressions”是一本极好的书。 然而实际上，你不需要成为一个正则表达式的专家，只需要知道如何捕获简单的URL模式就好了。 事实上，复杂的正则表达式的性能并不是太好，所以你并不应该太依赖于正则表达式。

最后，性能方面的一个注意点：这些正则表达式会在URLconf模块第一次载入的时候被编译。 它们超级快（只要这些正则表达式不像上面提醒的那样过于复杂）。

### [url()](http://python.usyiyi.cn/documents/ref/urls/#django.conf.urls.url) 参数：view[¶](http://python.usyiyi.cn/documents/django_182/intro/tutorial03.html" \l "url-argument-regex" \o "Permalink to this headline)

当Django找到一个匹配的正则表达式时，它就会调用view参数指定的视图函数，并将[**HttpRequest**](http://python.usyiyi.cn/documents/django_182/ref/request-response.html#django.http.HttpRequest)对象作为第一个参数，从正则表达式中“捕获”的其他值作为其他参数，传入到该视图函数中。如果正则表达式使用简单的捕获方式，值将作为位置参数传递； 如果使用命名的捕获方式，值将作为关键字参数传递。一会儿，我们将给出一个例子。

### [url()](http://python.usyiyi.cn/documents/ref/urls/#django.conf.urls.url) 参数：kwargs[¶](http://python.usyiyi.cn/documents/django_182/intro/tutorial03.html" \l "url-argument-regex" \o "Permalink to this headline)

任何关键字参数都可以以字典形式传递给目标视图。 我们在这个教程里不用Django的这个功能。

### [url()](http://python.usyiyi.cn/documents/ref/urls/#django.conf.urls.url) 参数：name[¶](http://python.usyiyi.cn/documents/django_182/intro/tutorial03.html" \l "url-argument-regex" \o "Permalink to this headline)

命名你的URL。 这样就可以在Django的其它地方尤其是模板中，通过名称来明确地引用这个URL。 这个强大的特性可以使你仅仅修改一个文件就可以改变全局的URL模式。

## 数据库设置

现在，编辑 mysite/settings.py 。 这是一个普通的 Python 模块，包含了代表 Django 设置的模块级变量。 更改 DATABASES 中 'default' 下的以下键的值，以匹配您的数据库连接设置。

* ENGINE – 从 'django.db.backends.postgresql\_psycopg2', 'django.db.backends.mysql', 'django.db.backends.sqlite3', 'django.db.backends.oracle' 中选一个， 至于其他请查看 also available.
* NAME – 你的数据库名。如果你使用 SQLite，该数据库将是你计算机上的一个文件；在这种情况下，NAME 将是一个完整的绝对路径，而且还包含该文件的名称。如果该文件不存在，它会在第一次同步数据库时自动创建（见下文）。

当指定路径时，总是使用正斜杠，即使是在 Windows 下(例如：C:/homes/user/mysite/sqlite3.db) 。

* USER – 你的数据库用户名 ( SQLite 下不需要) 。
* PASSWORD – 你的数据库密码 ( SQLite 下不需要) 。
* HOST – 你的数据库主机地址。如果和你的数据库服务器是同一台物理机器，请将此处保留为空 (或者设置为 127.0.0.1) ( SQLite 下不需要) 。查看 HOST 了解详细信息。

如果你是新建数据库，我们建议只使用 SQLite ，将 ENGINE 改为 'django.db.backends.sqlite3' 并且将 NAME 设置为你想存放数据库的地方。 SQLite 是内置在 Python 中的，因此你不需要安装任何东西来支持你的数据库。

Note

如果你使用 PostgreSQL 或者 MySQL，确保你已经创建了一个数据库。还是通过你的数据库交互接口中的 “CREATE DATABASE database\_name;” 命令做到这一点的。  
如果你使用 SQLite ，你不需要事先创建任何东西 - 在需要的时候，将会自动创建数据库文件。  
当你编辑 settings.py 时，将 TIME\_ZONE 修改为你所在的时区。默认值是美国中央时区（芝加哥）。

同时，注意文件底部的 INSTALLED\_APPS 设置。它保存了当前 Django 实例已激活的所有 Django 应用。每个应用可以被多个项目使用，而且你可以打包和分发给其他人在他们的项目中使用。

默认情况下，INSTALLED\_APPS 包含以下应用，这些都是由 Django 提供的：

* django.contrib.auth – 身份验证系统。
* django.contrib.contenttypes – 内容类型框架。
* django.contrib.sessions – session 框架。
* django.contrib.sites – 网站管理框架。
* django.contrib.messages – 消息框架。
* django.contrib.staticfiles – 静态文件管理框架。

这些应用在一般情况下是默认包含的。

所有这些应用中每个应用至少使用一个数据库表，所以在使用它们之前我们需要创建数据库中的表。要做到这一点，请运行以下命令：

python manage.py syncdb

syncdb 命令参照 INSTALLED\_APPS 设置，并在你的 settings.py 文件所配置的数据库中创建必要的数据库表。每创建一个数据库表你都会看到一条消息，接着你会看到一个提示询问你是否想要在身份验证系统内创建个超级用户。按提示输入后结束。

如果你感兴趣，可以在你的数据库命令行下输入：dt (PostgreSQL), SHOW TABLES; (MySQL), 或 .schema (SQLite) 来列出 Django 所创建的表。

极简主义者

就像我们上面所说的，一般情况下以上应用都默认包含在内，但不是每个人都需要它们。如果不需要某些或全部应用，在运行 syncdb 命令前可从 INSTALLED\_APPS 内随意注释或删除相应的行。syncdb 命令只会为 INSTALLED\_APPS 内的应用创建表。