第一部分：

Windows 命令 ：django-admin startproject mysite(创建项目)

mysite/

manage.py

mysite/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

这些目录和文件的用处是：

* 最外层的:file: *mysite/* 根目录只是你项目的容器， Django 不关心它的名字，你可以将它重命名为任何你喜欢的名字。
* **manage.py**: 一个让你用各种方式管理 Django 项目的命令行工具。你可以阅读 [django-admin and manage.py](https://docs.djangoproject.com/zh-hans/2.1/ref/django-admin/) 获取所有 **manage.py** 的细节。
* 里面一层的 **mysite/** 目录包含你的项目，它是一个纯 Python 包。它的名字就是当你引用它内部任何东西时需要用到的 Python 包名。 (比如 **mysite.urls**).
* **mysite/\_\_init\_\_.py**：一个空文件，告诉 Python 这个目录应该被认为是一个 Python 包。如果你是 Python 初学者，阅读官方文档中的 [更多关于包的知识](https://docs.python.org/3/tutorial/modules.html#tut-packages)。
* **mysite/settings.py**：Django 项目的配置文件。如果你想知道这个文件是如何工作的，请查看 [Django settings](https://docs.djangoproject.com/zh-hans/2.1/topics/settings/) 了解细节。
* **mysite/urls.py**：Django 项目的 URL 声明，就像你网站的“目录”。阅读 [URL调度器](https://docs.djangoproject.com/zh-hans/2.1/topics/http/urls/) 文档来获取更多关于 URL 的内容。
* **mysite/wsgi.py**：作为你的项目的运行在 WSGI 兼容的Web服务器上的入口。阅读 [如何使用 WSGI 进行部署](https://docs.djangoproject.com/zh-hans/2.1/howto/deployment/wsgi/) 了解更多细节。

创建投票应用**[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "creating-the-polls-app" \o "永久链接至标题)**

现在你的开发环境——这个“项目” ——已经配置好了，你可以开始干活了。

在 Django 中，每一个应用都是一个 Python 包，并且遵循着相同的约定。Django 自带一个工具，可以帮你生成应用的基础目录结构，这样你就能专心写代码，而不是创建目录了。

**项目 VS 应用**

项目和应用有啥区别？应用是一个专门做某件事的网络应用程序——比如博客系统，或者公共记录的数据库，或者简单的投票程序。项目则是一个网站使用的配置和应用的集合。项目可以包含很多个应用。应用可以被很多个项目使用。

你的应用可以存放在任何 [Python path](https://docs.python.org/3/tutorial/modules.html#tut-searchpath) 中定义的路径。在这个教程中，我们将在你的 **manage.py** 同级目录下创建投票应用。这样它就可以作为顶级模块导入，而不是 **mysite** 的子模块。

请确定你现在处于 **manage.py** 所在的目录下，然后运行这行命令来创建一个应用：

python manage.py startapp polls

这将会创建一个 **polls** 目录，它的目录结构大致如下：

polls/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py

tests.py

views.py

这个目录结构包括了投票应用的全部内容。

编写第一个视图**[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "write-your-first-view" \o "永久链接至标题)**

让我们开始编写第一个视图吧。打开 **polls/views.py**，把下面这些 Python 代码输入进去：

polls/views.py**[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "id1" \o "永久链接至代码)**

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

**def** index(request):

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

这是 Django 中最简单的视图。如果想看见效果，我们需要将一个 URL 映射到它——这就是我们需要 URLconf 的原因了。

为了创建 URLconf ( 路由路径 )，请在 polls 目录里新建一个 **urls.py** 文件。你的应用目录现在看起来应该是这样：

polls/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py

tests.py

urls.py

views.py

在 **polls/urls.py** 中，输入如下代码：

polls/urls.py**[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "id2" \o "永久链接至代码)**

**from** **django.urls** **import** path

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

urlpatterns = [

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

]

下一步是要在根 URLconf 文件中指定我们创建的 **polls.urls** 模块。在 **mysite/urls.py** 文件的 **urlpatterns** 列表里插入一个 **include()**， 如下：

mysite/urls.py**[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "id3" \o "永久链接至代码)**

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

**from** **django.urls** **import** include, path

urlpatterns = [

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

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

]

函数 [**include()**](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.include) 允许引用其它 URLconfs。每当 Django 遇到 :func：*~django.urls.include* 时，它会截断与此项匹配的 URL 的部分，并将剩余的字符串发送到 URLconf 以供进一步处理。

我们设计 [**include()**](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.include) 的理念是使其可以即插即用。因为投票应用有它自己的 URLconf( **polls/urls.py** )，他们能够被放在 "/polls/" ， "/fun\_polls/" ，"/content/polls/"，或者其他任何路径下，这个应用都能够正常工作。

**何时使用**[**include()**](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.include)

当包括其它 URL 模式时你应该总是使用 **include()** ， **admin.site.urls** 是唯一例外。

你现在把 **index** 视图添加进了 URLconf。可以验证是否正常工作，运行下面的命令:

/ 

**$** python manage.py runserver

用你的浏览器访问 <http://localhost:8000/polls/>，你应该能够看见 "*Hello, world. You're at the polls index.*" ，这是你在 **index** 视图中定义的。

函数 [**path()**](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.path) 具有四个参数，两个必须参数：**route** 和 **view**，两个可选参数：**kwargs** 和 **name**。

### [path()](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.path) 参数： route[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/#path-argument-route)

**route** 是一个匹配 URL 的准则（类似正则表达式）。当 Django 响应一个请求时，它会从 **urlpatterns**的第一项开始，按顺序依次匹配列表中的项，直到找到匹配的项。

这些准则不会匹配 GET 和 POST 参数或域名。例如，URLconf 在处理请求 **https://www.example.com/myapp/** 时，它会尝试匹配 **myapp/** 。处理请求 **https://www.example.com/myapp/?page=3** 时，也只会尝试匹配 **myapp/**。

### [path()](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.path) 参数： view[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/#path-argument-view)

当 Django 找到了一个匹配的准则，就会调用这个特定的视图函数，并传入一个 **[HttpRequest](https://docs.djangoproject.com/zh-hans/2.1/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** 对象作为第一个参数，被“捕获”的参数以关键字参数的形式传入。稍后，我们会给出一个例子。

### [path()](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.path) 参数： kwargs[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/" \l "path-argument-kwargs" \o "永久链接至标题)

任意个关键字参数可以作为一个字典传递给目标视图函数。本教程中不会使用这一特性。

### [path()](https://docs.djangoproject.com/zh-hans/2.1/ref/urls/#django.urls.path) 参数： name[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial01/#path-argument-name)

为你的 URL 取名能使你在 Django 的任意地方唯一地引用它，尤其是在模板中。这个有用的特性允许你只改一个文件就能全局地修改某个 URL 模式。

当你了解了基本的请求和响应流程后，请阅读 [教程的第 2 部分](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial02/) 开始使用数据库.