Tutorial 1:

1. Creating a project

$ django-admin startproject mysite

startproject creates:

mysite/

manage.py

mysite/

\_\_init\_\_.py

settings.py

urls.py

asgi.py

wsgi.py

These files are:

* + The outer mysite/ root directory is a container for your project. Its name doesn’t matter to Django; you can rename it to anything you like.
  + manage.py: A command-line utility that lets you interact with this Django project in various ways. You can read all the details about manage.py in django-admin and manage.py.
  + The inner mysite/ directory is the actual Python package for your project. Its name is the Python package name you’ll need to use to import anything inside it (e.g. mysite.urls).
  + mysite/\_\_init\_\_.py: An empty file that tells Python that this directory should be considered a Python package. If you’re a Python beginner, read more about packages in the official Python docs.
  + mysite/settings.py: Settings/configuration for this Django project. Django settings will tell you all about how settings work.
  + mysite/urls.py: The URL declarations for this Django project; a “table of contents” of your Django-powered site. You can read more about URLs in URL dispatcher.
  + mysite/asgi.py: An entry-point for ASGI-compatible web servers to serve your project. See How to deploy with ASGI for more details.
  + mysite/wsgi.py: An entry-point for WSGI-compatible web servers to serve your project. See How to deploy with WSGI for more details.

1. Changing the port

By default, the runserver command starts the development server on the internal IP at port 8000.

If you want to change the server’s port, pass it as a command-line argument. For instance, this command starts the server on port 8080:

python manage.py runserver 8080

If you want to change the server’s IP, pass it along with the port. For example, to listen on all available public IPs (which is useful if you are running Vagrant or want to show off your work on other computers on the network), use:

python manage.py runserver 0:8000

1. Creating our App

To create your app, make sure you’re in the same directory as **manage.py** and type this command:

python manage.py startapp polls

That’ll create a directory **polls**, which is laid out like this:

polls/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py // interacts with the db

tests.py

views.py

1. Using the App

* Write a View (In views.py):

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

**def** index(request):

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

* To call the view, we need to map it to a URL

To create a URLconf in the polls directory, create a file called **urls.py**

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

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

urlpatterns = [

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

]

Tutorial 2:

* Using shortcut Render():

It’s a very common idiom to load a template, fill a context and return an **[HttpResponse](https://docs.djangoproject.com/en/4.0/ref/request-response/" \l "django.http.HttpResponse" \o "django.http.HttpResponse)** object with the result of the rendered template. Django provides a shortcut. Here’s the full **index()** view, rewritten:

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

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

**def** index(request):

latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

context = {'latest\_question\_list': latest\_question\_list}

**return** render(request, 'polls/index.html', context)