Tutorial
on Windows

Create a Django Project in Anaconda

Anna Christiane Kolandjian

How To Create a Django Project in Anaconda?

1. CREATE A NEW ENVIRONMENT

2. ACTIVATE THE NEW ENVIRONMENT

3. INSTALL DJANGO IN THE NEW ENVIRONMENT

4. CREATE A NEW DJANGO PROJECT

5. RUN THE DJANGO SERVER

1. Create a new environment

First of all, you need to create a separate environment for Django development.

By default, anaconda comes up with the base environment. But creating a separate environment will be helpful to manage the projects without any package dependency confliction.

So, let’s create a separate environment specifically for web development projects using the Django framework.

Open your CMD or Anaconda Prompt and create a separate environment called *djangoenv* by typing in the following command.

```
conda create -n djangoenv python=3.6 anaconda
```

- The previous command may take some time to finish its execution. Once it is done, let’s go to the next step.
- I just mentioned the version of Python that I’m using as *Python 3.6*. If you want to use other versions of Python, you can do so.

2. Activate the new environment

```
conda activate djangoenv
```

Now, we are inside the newly created environment. But our environment does not have Django in it. So, let’s install it.

3. Install Django in the new environment

```
conda install -c anaconda django
```

Let’s try to create a sample project to check whether everything is ready. Let’s do this using the following commands.

Wait till the installation completes. It may take some time to finish. That’s it. Now you can create Django projects in this separate environment.

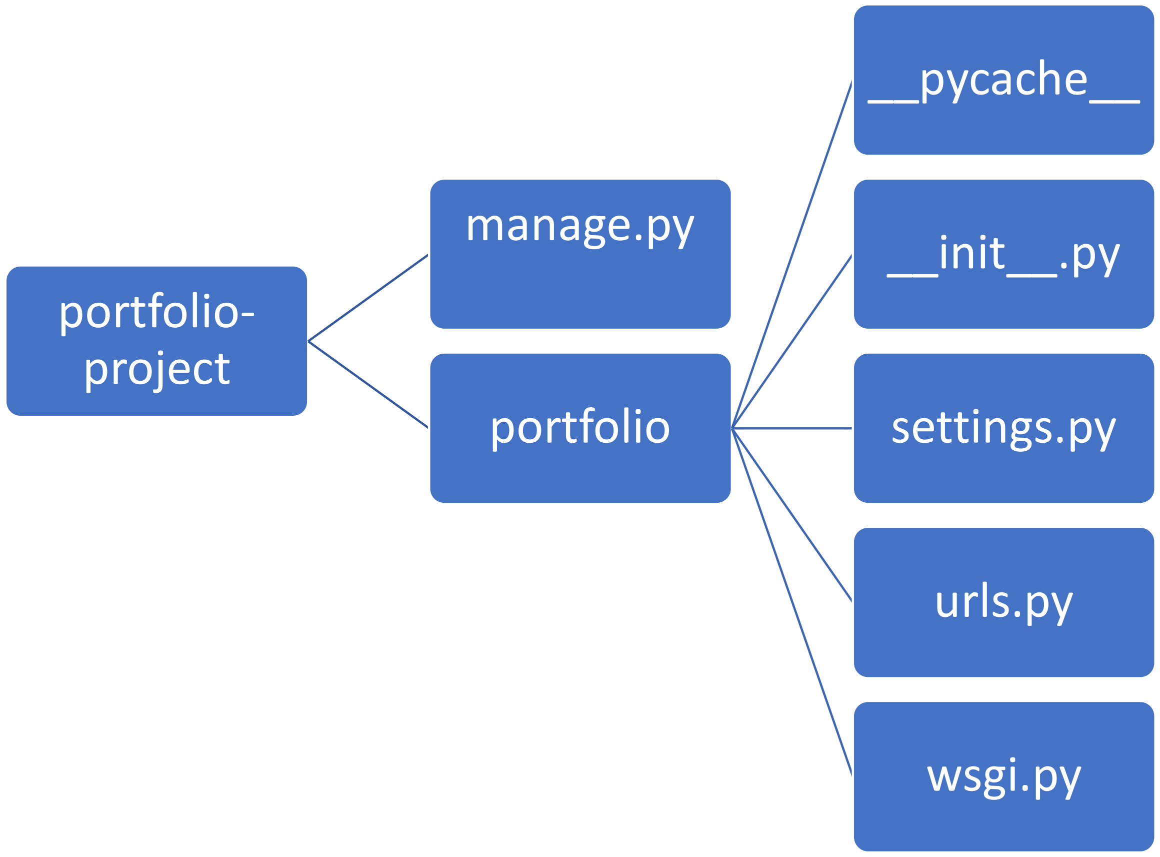
4. Create a new Django project

```
django-admin startproject portfolio
```

First, let’s start a new project called *portfolio*.

You’ll find a new folder called portfolio that contains built-in files

Change the name of the original folder from portfolio to portfolio-project.



5. Run the Django server

- Now, navigate to the project folder by using the cd command. Let’s run the project and see whether everything is working fine or not.

```
cd portfolio-project
```

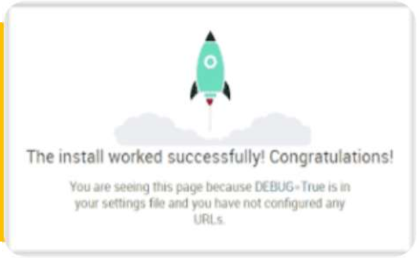
```
python manage.py runserver
```

Congrats !

Once you do that, a URL will be generated on the command line. Now, copy the URL that is generated and paste it on a new tab in your browser. (Chrome)

http://127.0.0.1:8000/ (URL similar to this)

That's it. Everything is set up nicely if you can see this rocket launching up towards the sky.



The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Now, your Django environment is set up nicely. After this, you can open Command Prompt or Anaconda Prompt whenever needed and start doing the project just by activating the environment for Django.

The following command is used whenever we need to activate the Django environment.

```
conda activate djangoenv
```

Create a Django app in a project

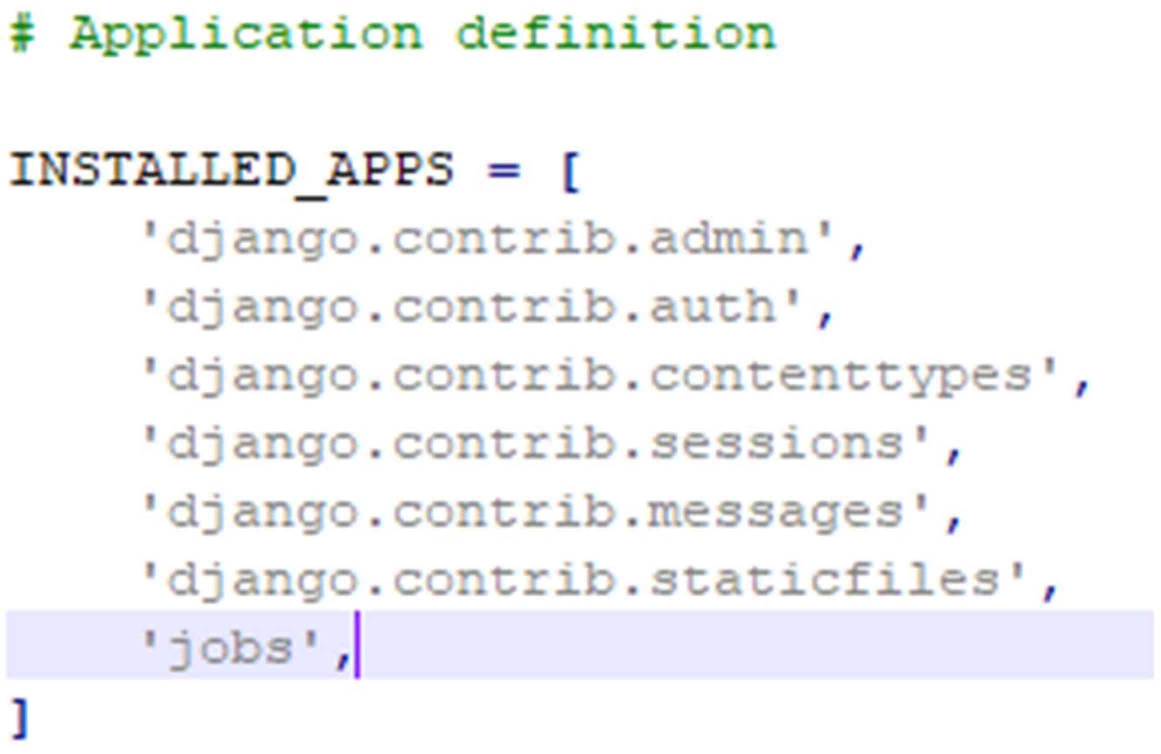
Tutorial on Windows

```
django-admin startapp jobs
```

You created a new app and called it jobs

Check your folder portfolio-project, you'll find a new folder called jobs

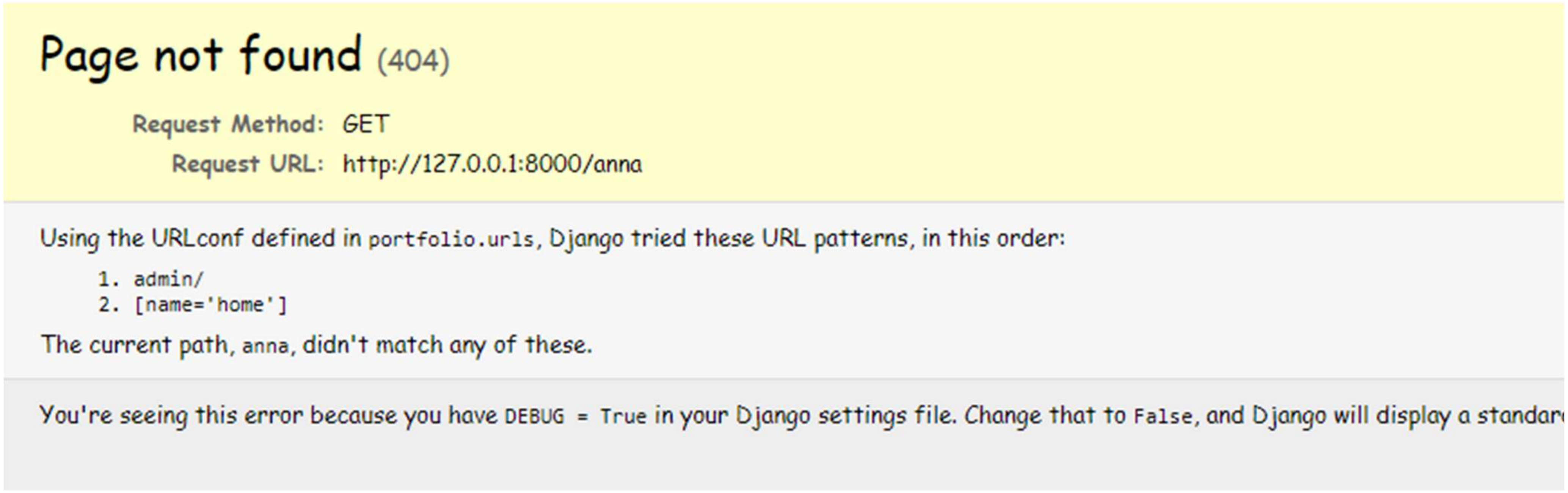
- We still should let our Django project know that we have a new app !
- Open your folder portfolio-project/portfolio
- Open the file settings.py and add 'jobs', in the INSTALLED_APPS.



```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'jobs',
]
```

Let’s try to add a ‘/anna’ on our URL

- http://127.0.0.1:8000/anna
- Oups! Page not found!



Page not found (404)

Request Method: GET
Request URL: http://127.0.0.1:8000/anna

Using the URLconf defined in `portfolio.urls`, Django tried these URL patterns, in this order:

```
1. admin/
2. [name='home']
```

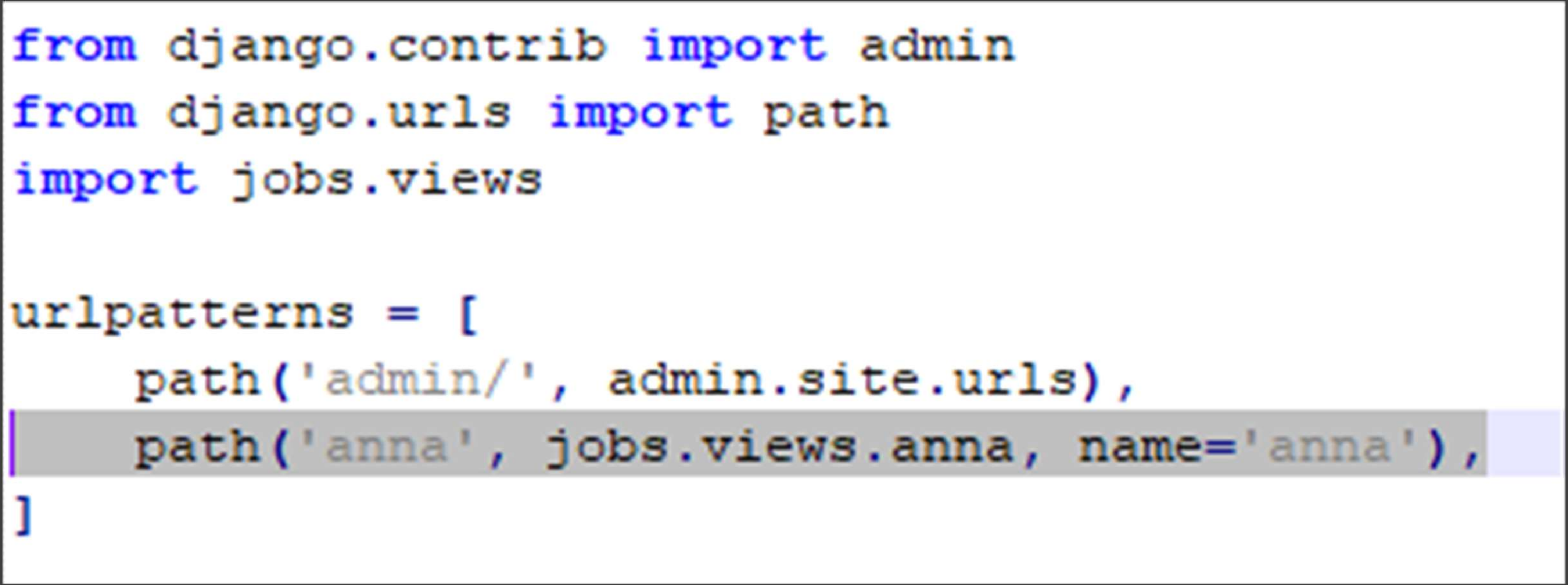
The current path, `anna`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standar

Open the file urls.py in portfolio

- Add import jobs.views
- Then add a path in urlpatterns in the code

  path('anna', jobs.views.anna, name='anna'),



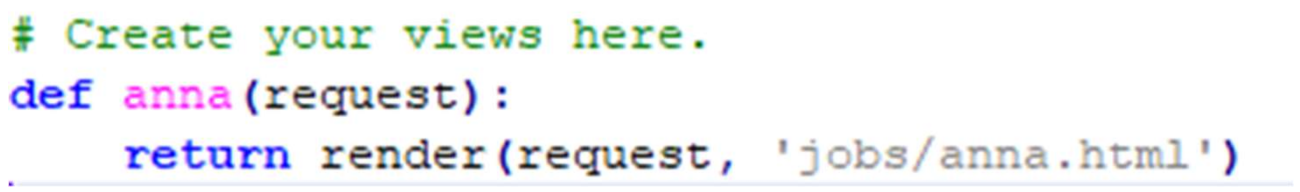
```
from django.contrib import admin
from django.urls import path
import jobs.views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('anna', jobs.views.anna, name='anna'),
]
```

There’s no function anna! Let’s create it.

- But first, bare with me in creating new folders in jobs.
- Create a new folder in jobs, call it templates.
- Then, create a new folder in templates, call it jobs.
- In this last folder jobs, create a new file, call it anna.html
- This is the HTML file we’re trying to send back to the user.
- Open anna.html and do the classic programming. Write Hello World! And save it.

Creating a new function anna in views.py

- Let’s get back to the original folder jobs in portfolio.
- Open views.py and create a function anna.



```
# Create your views here.
def anna(request):
    return render(request, 'jobs/anna.html')
```

To modify the home page (without a /anna)

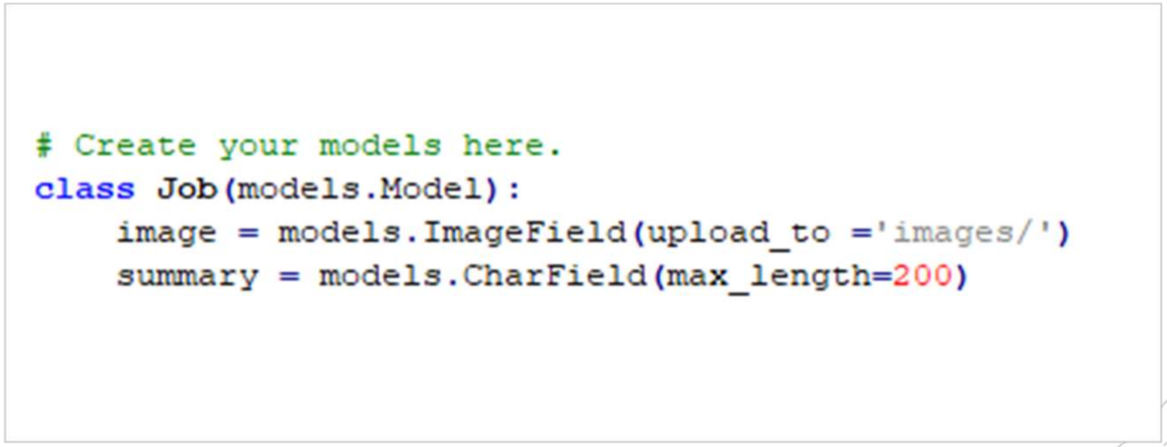
- Open urls.py in portfolio
  - Add `path ('', jobs.views.home, name='home’),` in urlpatterns.
- Open jobs/templates/jobs and create a new file called home.html
  - Modify the home.html by adding Hello World! For example
- Open views.py in the original jobs folder
  - Add a new function called home

```
def home(request):
return render(request, 'jobs/home.html’)
```

- Open the URL http://127.0.0.1:8000/ , you’ll find your modifications. For example, here we have Hello World!

Creating the models in Django

- Open models.py in the original folder jobs.
- Let’s create the model Job and give it two properties image and summary.



```
# Create your models here.
class Job(models.Model):
    image = models.ImageField(upload_to ='images/')
    summary = models.CharField(max_length=200)
```

Installing pillow

- Any time you work with ImageField in django, you have to have something from pip called pillow installed.
- It just allows Django to work with images much simpler.
- First control+C in the Command Prompt, to get out of the server.
- Then, type `pip install pillow`