Learn Django!

Yashraj Kakkad (LinkedIn / GitHub)

Web Development Series, Programming Club, Ahmedabad University

Source

- **Static Website:** "Prebuilt" pages
- **Dynamic Website:** Pages are built on runtime! (Often after a database interaction)
- **Database management system:** Software that handles the storage, retrieval, and updating of data in a computer system. (PostgreSQL, MySQL, SQLite etc.)

Why Django?

- Rapid application development.
- Easy to use
- Python based web framework.
- Secure, Scalable and Reliable

- .....and more reasons.

Top-down learning (vs Traditional Bottom-up)

Elon Musk:

> One bit of advice: it is important to view knowledge as sort of a semantic tree — make sure you understand the fundamental principles, ie the trunk and big branches, before you get into the leaves/details or there is nothing for them to hang on to.

Installation

- Install Python.
- Install pip. (if it didn't come with Python)
- Setup a virtual environment
- Install Django.

```
pip install django
```

Getting started

- Create a project.
  - Linux/OSX:

```
django-admin startproject mysite .
```

  - Windows:

```
django-admin.exe startproject mysite .
```

- A quick look at the `mysite/`:
  - `manage.py`
  - `settings.py`
  - `urls.py`
- Update `settings.py`:

```
ALLOWED_HOSTS = ['127.0.0.1', 'localhost']
```

- Python supports SQLite3 out of the box (and that's what we'll use for now).

First application

- To create an app:

```
python manage.py startapp foodie
```

- A quick look at `foodie/`:
  - `admin.py`
  - `apps.py`
  - `models.py`

    - `views.py`
- Model-View-Control architecture.
- Add it to `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = [
    ..
    ..
   'foodie.apps.FoodieConfig',
]
```

- `foodie/views.py`:

```
from django.shortcuts import render, HttpResponse

def index(request):
    return HttpResponse('Hello World!')
```

- `mysite/urls.py`:

```
from django.urls import include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('foodie.urls')),
]
```

- `foodie/urls.py` (Create it):

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

- Migrate your changes to the database:

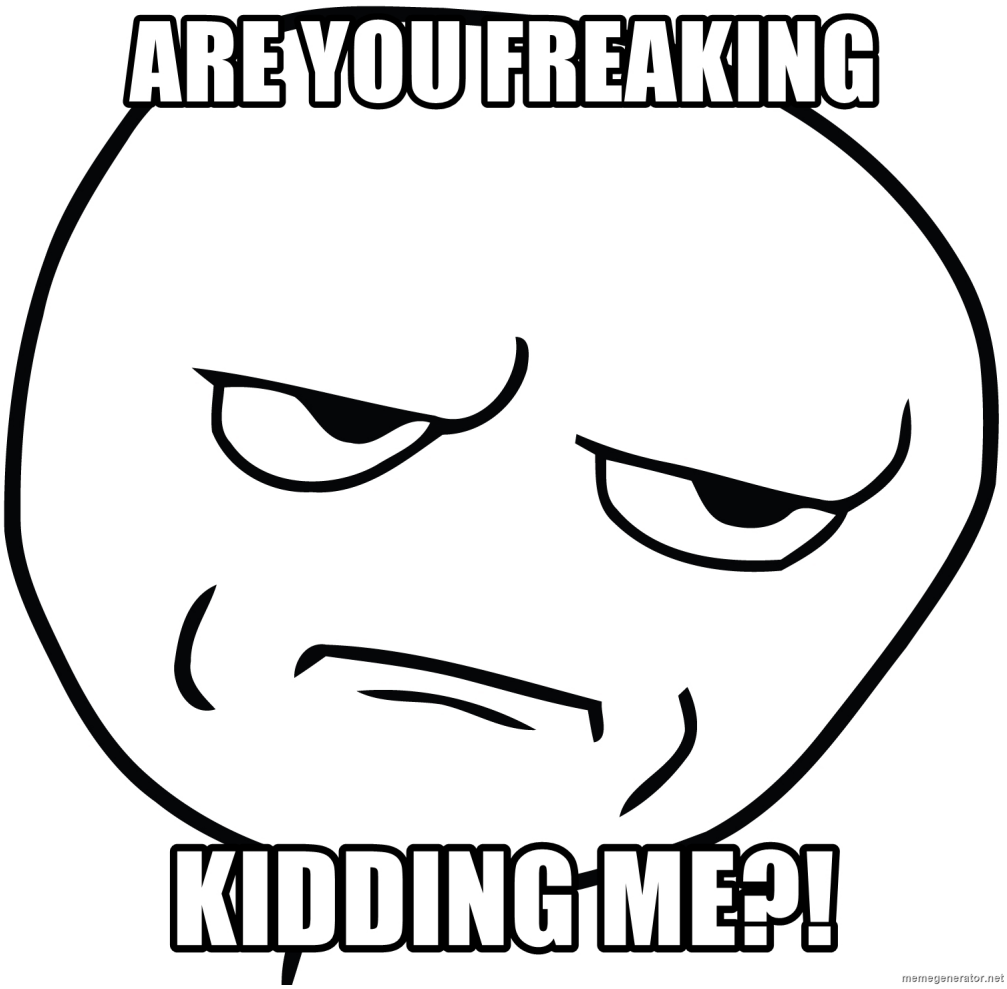
```
python manage.py migrate
```

- Launch Django development server!

```
python manage.py runserver
```

- Open http://127.0.0.1:8000 in your browser.

So much effort for so little results?



Source

Trust me, it'll all be worth it soon!

Models

- `foodie/models.py`:

```
from django.db import models


class Ingredient(models.Model):
    name = models.CharField(max_length=50)


class FoodItem(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()
    price = models.IntegerField()
    ingredients = models.ManyToManyField(Ingredient)
```

Migrations

- Django Documentation:

> Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema.

- Create migrations based on the changes made:

```
python manage.py makemigrations
```

- Apply migrations:

```
python manage.py migrate
```

- A quick look at `foodie/migrations`.

Django Admin

- `mysite/admin.py`:

```
from django.contrib import admin
from .models import *

admin.site.register(Ingredient)
admin.site.register(FoodItem)
```

- Re-run your server if not already.
- Open http://127.0.0.1:8000/admin
- Where's the username?

```
python manage.py createsuperuser
```

- Enter your created username and password. Add some instances now.
- Displaying names of objects in Django admin: `__str__()` method.
  `foodie/models.py`:

```
from django.db import models


class Ingredient(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name


class FoodItem(models.Model):
    name = models.CharField(max_length=50)
    description = models.TextField()
    price = models.IntegerField()
    ingredients = models.ManyToManyField(Ingredient)

    def __str__(self):
        return self.name
```

Templating

- `foodie/templates/foodie/index.html`:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Welcome to Foodie!</title>
  </head>
  <body>
    <h1>Welcome people!</h1>
  </body>
</html>
```

- Change the `index` method in `views.py`:

```
def index(request):
    return render(request, 'foodie/index.html')
```

- Check out the homepage now. Let's make it dynamic now!
  `foodie/templates/foodie/index.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Welcome to Foodie!</title>
  </head>
  <body>
    <h1>Welcome food lovers!</h1>
    {% for food in food_items %}
        <h2>{{ food.name }}</h2>
        <h4>Rs: {{ food.price }}</h4>
        <hr />
    {% endfor %}
  </body>
</html>
```

- Update `views.py` to provide the `food_items` variable:

```
from django.shortcuts import render
from .models import FoodItem


def index(request):
    food_items = FoodItem.objects.all()
    context = {'food_items': food_items}
    return render(request, 'foodie/index.html', context=context)
```

Adding One More View

- One more view:

```
from django.shortcuts import Http404

def detail(request, pk):
    try:
        food_item = FoodItem.objects.get(pk=pk)
    except FoodItem.DoesNotExist:
        raise Http404
    context = {'food_item': food_item}
    return render(request, 'foodie/detail.html', context=context)
```

- Add the corresponding url:
  `foodie/urls.py`

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:pk>/', views.detail),
]
```

- `foodie/templates/foodie/detail.html`:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Food Details - {{food_item}}</title>
  </head>
  <body>
    <h1>{{food_item}}</h1>
    <h2>Rs: {{food_item.price}}</h2>
    <h3>{{ food_item.description }}</h3>
    <p>Ingredients:</p>
    <ul>
      {% for ingredient in food_item.ingredients.all %}
      <li>{{ ingredient }}</li>
      {% endfor %}
    </ul>
  </body>
</html>
```

Static files

Static files: Images, CSS, JS etc.

Download and save an image (in this case `cafe.jpg`) in `foodie/static/img`

- In the first line of `index.html` add:

```
{% load static %}
```

- Wherever you want to place the image:

```
<img src="{% static 'img/cafe.jpg' %}" />
```

Hardly a drop in the ocean

Refer to the resources document and get started!