PYTHON WEB • DJANGO FULL COURSE

Python Django Full Course

Projects, apps, models, admin, views, templates, forms and deployment

FREE FULL COURSE • MyWebUniversity.com

Learn it all

@MyWebUniversity

MyWebUniversity.com

DJANGO

What You Will Learn

Projects and apps
How a Django site is organized.

Models and migrations
Define your data in Python; Django builds the database.

The admin
A complete management interface, for free.

Views, URLs, templates
Turn data into web pages.

Forms and users
Safe input, login and permissions.

Deployment
Settings, static files and production servers.

Python Django • Full Course

PART 1 · SETUP AND MODELS

DJANGO

Create a Project and an App

Terminal — Python 3.13

```
$ python -m venv .venv && source .venv/bin/activate
$ pip install django
$ django-admin startproject mysite .

$ python manage.py startapp blog
# add "blog" to INSTALLED_APPS in mysite/settings.py

$ python manage.py runserver
Starting development server at http://127.0.0.1:8000/
```

Project

The whole website: settings, main URLs and configuration.

Apps

Reusable pieces with one job each: blog, shop, accounts.

manage.py

Your command center: runserver, startapp, migrate, createsuperuser, shell.

DJANGO

Models: Describe Your Data

blog/models.py — Python 3.13

```python
from django.db import models
from django.contrib.auth.models import User

class Post(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    body = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    created = models.DateTimeField(auto_now_add=True)
    published = models.BooleanField(default=False)


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

One class = one table

Each field becomes a column with a type and rules.

Relationships

ForeignKey links each post to its author (many posts, one user).

__str__

How a post appears in the admin and the shell.

DJANGO

Migrations and the ORM

Terminal — Python 3.13

```
$ python manage.py makemigrations blog
$ python manage.py migrate

$ python manage.py shell
>>> from blog.models import Post
>>> Post.objects.filter(published=True).order_by("-created")[:5]
>>> Post.objects.get(slug="hello-django")
>>> Post.objects.filter(title__icontains="python").count()
3
```

Migrations

makemigrations writes the change; migrate applies it to the database.

QuerySets

filter, exclude, order_by, get, count, first, with slicing for limits.

Field lookups

Double underscores: __icontains, __gte, __year, author__username.

DJANGO

The Django Admin

blog/admin.py — Python 3.13

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

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
   list_display = ["title", "author", "published", "created"]
   list_filter = ["published", "created"]
   search_fields = ["title", "body"]
   prepopulated_fields = {"slug": ["title"]}


$ python manage.py createsuperuser
# then visit http://127.0.0.1:8000/admin/
```

Free management UI

Add, edit, search and delete data from day one.

Customize it

list_display, list_filter, search_fields, prepopulated_fields.

For staff

The admin is for trusted staff, not the public site.

Python Django • Full Course

PART 2 · VIEWS, URLS AND TEMPLATES

DJANGO

Views and URLs

blog/views.py & urls.py — Python 3.13

```
from django.shortcuts import render, get_object_or_404
def post_list(request):
   posts = Post.objects.filter(published=True).order_by("-created")
   return render(request, "blog/post_list.html", {"posts": posts})



def post_detail(request, slug):
   post = get_object_or_404(Post, slug=slug, published=True)
   return render(request, "blog/post_detail.html", {"post": post})


urlpatterns = [path("", views.post_list, name="post_list"),
               path("<slug:slug>/", views.post_detail, name="post_detail")]
# mysite/urls.py: path("blog/", include("blog.urls"))
```

A view

A function that receives a request and returns a response.

get_object_or_404

Returns the object, or a proper 404 page if it does not exist.

Named URLs

name="post_detail" lets templates build links without hard-coding addresses.

DJANGO

Templates

blog/templates/blog/ — Python 3.13

```
{% extends "base.html" %}
{% block content %}
<h1>Latest posts</h1>
{% for post in posts %}
 <h2><a href="{% url 'post_detail' post.slug %}">{{ post.title }}</a></h2>
 <p>{{ post.created|date:"M d, Y" }} by {{ post.author }}</p>
 <p>{{ post.body|truncatewords:30 }}</p>
{% empty %}<p>No posts yet.</p>
{% endfor %}
{% endblock %}
```

Same ideas as Jinja2

{{ }} for values, {% %} for logic, extends and block for layouts.

Filters

date, truncatewords, upper, default, linebreaks... format values with |.

{% url %}

Builds links from URL names. {% empty %} handles empty lists.

Python Django • Full Course

PART 3 · FORMS, USERS AND DEPLOYMENT

DJANGO

ModelForms and Login

blog/forms.py & views.py — Python 3.13

```python
class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ["title", "slug", "body"]


@login_required
def post_create(request):
    form = PostForm(request.POST or None)
    if form.is_valid():
        post = form.save(commit=False)
        post.author = request.user
        post.save()
        return redirect("post_detail", slug=post.slug)
    return render(request, "blog/post_form.html", {"form": form})
```

ModelForm

Builds a form from a model, with validation, automatically.

is_valid()

Checks every field; errors are shown in the template with {{ form }}.

CSRF and login

Add {% csrf_token %} in every POST form; @login_required protects views.

DJANGO

Deployment Checklist

Terminal — Python 3.13

```
$ python manage.py check --deploy


# settings.py for production
DEBUG = False
SECRET_KEY = os.environ["DJANGO_SECRET_KEY"]
ALLOWED_HOSTS = ["www.example.com"]
STATIC_ROOT = BASE_DIR / "staticfiles"


$ python manage.py collectstatic
$ pip install gunicorn && gunicorn mysite.wsgi -w 4
```

check --deploy

Django lists the security settings you still need to fix.

Production settings

DEBUG off, secret key from env, ALLOWED_HOSTS, HTTPS settings, PostgreSQL.

Serve it

collectstatic, gunicorn or uvicorn, nginx in front, or a cloud platform.

Recap and Practice

Remember

Project → apps → models → migrations → admin → views, URLs, templates → forms.

Try it yourself

Add comments to the blog: a Comment model, admin, and a form under each post.

Flask or Django?

Flask: small and flexible. Django: batteries included for big sites.