Table of Contents

1. What is Django?

- Django is a Python-based web framework that follows the Model-View-Template (MVT) architecture.
- Why Use Django?
    - Fast development
    - Secure by default
    - Scalable and maintainable

---

2. Installation

```
pip install django
```

Verify Installation:

```
django-admin --version
```

---

3. Create a Django Project

```
django-admin startproject projectname
cd projectname
python manage.py runserver
```

- Access at: http://127.0.0.1:8000/

---

4. Project Structure

```
projectname/
│
├── manage.py                # Command-line utility
├── projectname/             # Project directory
│   ├── __init__.py
│   ├── asgi.py              # ASGI config
│   ├── settings.py          # Project settings
│   ├── urls.py              # URL routing
│   └── wsgi.py              # WSGI config
└── db.sqlite3               # Default SQLite database
```

---

5. Create an App

```
python manage.py startapp appname
```

Register the App in settings.py:

```
INSTALLED_APPS = [
    'appname',
]
```

---

6. Models (Database Design)

Define Models (in models.py):

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
```

Apply Migrations:

```
python manage.py makemigrations
python manage.py migrate
```

---

7. Django Admin

```
python manage.py createsuperuser
python manage.py runserver
```

- Access Admin Panel: http://127.0.0.1:8000/admin/

Register Model for Admin (in admin.py):

```
from .models import Post
admin.site.register(Post)
```

---

8. Views (Business Logic)

Define Views (in views.py):

```
from django.shortcuts import render

def home(request):
    return render(request, 'home.html', {'name': 'Django'})
```

---

9. URLs (Routing)

Map URLs (in urls.py):

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

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

---

10. Templates (Frontend Design)

Create HTML File (in templates/home.html):

```
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h1>Welcome to {{ name }}!</h1>
</body>
```

```
</html>
```

---

11. Static Files (CSS, JS, Images)

Configure in settings.py:

```
STATIC_URL = '/static/'
```

Usage in Template:

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

---

12. Forms (User Input)

Create Form (in forms.py):

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(max_length=100)
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

---

13. Query Data (ORM Queries)

```
# Get all objects
Post.objects.all()

# Filter objects
Post.objects.filter(title__contains='Django')
```

```
# Get single object
Post.objects.get(id=1)

# Create new entry
Post.objects.create(title='New Post', content='Hello World!')

# Update entry
post = Post.objects.get(id=1)
post.title = 'Updated Title'
post.save()

# Delete entry
post.delete()
```

---

14. Middleware

Custom Middleware (in middleware.py):

```
from django.utils.timezone import now

class TimingMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        print(f"Request at {now()}")
        response = self.get_response(request)
        return response
```

Add to settings.py:

```
MIDDLEWARE = [
    'appname.middleware.TimingMiddleware',
]
```

---

15. User Authentication

Login View (in views.py):

```
from django.contrib.auth import authenticate, login

def user_login(request):
    user = authenticate(request, username='john', password='secret')
    if user:
        login(request, user)
```

---

16. Deployment

```
python manage.py collectstatic
```

- Use Gunicorn and Nginx for production deployment.

---

17. Common Django Commands

```
python manage.py runserver          # Start server
python manage.py makemigrations     # Create migration files
python manage.py migrate            # Apply migrations
python manage.py createsuperuser    # Create admin user
python manage.py collectstatic      # Collect static files
python manage.py shell              # Django shell
```

---

18. Useful Shortcuts

- Django Shell:

  ```
  python manage.py shell
  ```

- Check for Errors:

```
    python manage.py check
```

Example: Simple Blog App

```
# models.py
class Blog(models.Model):
    title = models.CharField(max_length=100)
    body = models.TextField()

# views.py
def blog_list(request):
    blogs = Blog.objects.all()
    return render(request, 'blog_list.html', {'blogs': blogs})

# urls.py
path('blogs/', views.blog_list, name='blog_list')
```