1.وب‌سرور چیست؟ (مثل نقش Nginx)

وب‌سرور یک برنامه است که کارش اینه:

- گوش می‌کنه ببینه کسی (کلاینت) درخواست داده یا نه.
- درخواست رو می‌گیره، بررسی می‌کنه، و پاسخ مناسب می‌ده.

مثلاً **Nginx** می‌تونه:

- فایل‌های ثابت (HTML، CSS، عکس‌ها) رو مستقیماً بده.
- درخواست‌های پیچیده رو بفرسته به برنامه‌های پشت صحنه (مثل جنگو).
- ترافیک رو بین چند سرور تقسیم کنه.(Load Balancing)

تشبیه: مثل منشی یک شرکت که یا خودش جواب مشتری رو می‌ده (فایل‌های ساده)، یا اگه موضوع پیچیده بود، می‌فرسته پیش کارمند متخصص (جنگو).

2.ارتباط بین کلاینت (مرورگر) و سرور (بک‌اند) چطور است؟

- مرورگر (کلاینت) یک **درخواست** (Request) می‌سازد.
- این درخواست را از طریق اینترنت با پروتکل HTTP یا HTTPS به سرور می‌فرستد.
- سرور درخواست را می‌خواند و یک **پاسخ** (Response) برمی‌گرداند.

این درخواست‌ها و پاسخ‌ها مثل نامه‌نگاری دیجیتال هستند:

- کلاینت نامه می‌نویسد: سلام، صفحه اصلی سایت رو می‌خواهم.
- سرور نامه می‌دهد: بفرما، این HTML صفحه اصلی.

3. پروتکل HTTPچیست؟ و وقتی URL را می‌زنی چه می‌شود؟

- **HTTP** یک زبان و قانون است که کلاینت و سرور با آن حرف می‌زنند.
- وقتی یک URL می‌زنی، این اتفاق می‌افتد:

1. مرورگر اول می‌فهمد که آدرس سایت به چه IP تعلق دارد (با کمکDNS).
2. به آن IP درخواست HTTP یا HTTPS می‌فرستد.
3. سرور پاسخ را می‌دهد (مثلاً یک صفحه HTML یا یک خطا).
4. مرورگر پاسخ را پردازش و نمایش می‌دهد (و اگر لازم باشدCSS ، JS و عکس‌ها را هم درخواست می‌کند).

4.جایگاه یک برنامه Django در معماری شبکه کجاست؟

- مرورگر به وب‌سرور (مثلNginx) وصل می‌شود.
- Nginxدرخواست‌ها را که نیاز به پردازش دارند، می‌فرستد به **برنامهDjango** .
- Djangoمنطق کار (Business Logic) را اجرا می‌کند، با دیتابیس حرف می‌زند و داده را برمی‌گرداند.
- خروجی Django دوباره از طریق وب‌سرور به مرورگر می‌رسد.

یعنی Django مغز سیستم است که تصمیم می‌گیرد چه چیزی نمایش داده شود.

5.جریان زیر را مرحله به مرحله توضیح میدهم:

Browser → DNS → Network → Web Server → Django App → Database

1. **Browser**
   - کاربر آدرس سایت را وارد می‌کند. مرورگر باید آن را پیدا کند.
2. **DNS**
   - یک سرویس است که اسم سایت (مثلexample.com) را به IP تبدیل می‌کند (مثل 203.0.113.5).

3. **Network**
   - مرورگر درخواست را با IP پیدا شده به سرور ارسال می‌کند (مثل ارسال یک نامه به یک آدرس پستی دیجیتال).
4. **Web Server**
   - درخواست را می‌گیرد.
   - اگر فایل ساده باشد، مستقیم می‌دهد.
   - اگر نیاز به پردازش باشد، می‌فرستد به.Django
5. **Django App**
   - درخواست را بررسی می‌کند.
   - اگر لازم باشد، به دیتابیس وصل می‌شود تا اطلاعات را بگیرد یا ذخیره کند.
   - نتیجه را به وب‌سرور می‌دهد.
6. **Database**
   - اطلاعات خام را ذخیره و مدیریت می‌کند (مثل انبار داده).
   - Djangoداده را از آن می‌گیرد یا تغییر می‌دهد.

در نهایت پاسخ از دیتابیس به جنگو و به وب‌سرور و به مرورگر برمی‌گردد.

6.معماری Django

Djangoیک **فریم‌ورک وب** است که معماری‌اش بر پایه **MVT** (Model–View–Template)است، که مشابه MVC است ولی کمی تفاوت دارد.

— MVC (Model–View–Controller)معماری کلی وب‌اپ‌ها

- **Model** : داده‌ها و منطق مربوط به ذخیره و بازیابی آنها (مثلاً جدول‌های دیتابیس).
- **View** : مسئول نمایش به کاربر.(UI)

- **Controller** : پل ارتباطی بین Model وView ؛ درخواست را دریافت می‌کند، داده را از Model می‌گیرد و به View می‌دهد.

MVTدرDjango

در Django به جای "Controller" ما چیزی به نام **View**داریم (که نقش Controller را بازی می‌کند) و بخش View واقعی در MVC را در **Template** Django می‌نامیم.

اجزای MVT در:Django

1. **Model**
   - کلاس‌های پایتونی که ساختار دیتابیس را تعریف می‌کنند.
   - مدیریت ارتباط با دیتابیس را بر عهده دارند.(ORM)
2. **View**
   - تابع یا کلاس پایتونی که درخواست کاربر را می‌گیرد.
   - داده را از Model می‌گیرد و آن را به Template یا به صورت JSON برمی‌گرداند.
   - در واقع **Controller**در MVC همین View است.
3. **Template**
   - فایل‌های HTML (به همراه تگ‌هایTemplate Language) که داده را به کاربر نمایش می‌دهند.

اجزای اصلی درDjango

- **URL** : مسیر درخواست را مشخص می‌کند و تصمیم می‌گیرد کدام View اجرا شود.
- **View** : کد پردازشی که داده را آماده می‌کند و تصمیم می‌گیرد چه پاسخی بدهد.
- **Model** : تعریف ساختار و منطق داده، ذخیره و بازیابی از دیتابیس.

- **Template** : فایل HTML که نتیجه را به کاربر نشان می‌دهد.

جریان درخواست:

User → URL → View → Model (optional) → Template → Response

7. Python Virtual Environment (virtualenv)

تعریف :یک محیط جداگانه و ایزوله برای نصب پکیج‌های پایتونی است که پروژه شما از آن استفاده می‌کند.

چرا مهم است؟

- هر پروژه نسخه خاصی از کتابخانه‌ها را نیاز دارد.
- باvirtualenv ، پکیج‌های پروژه‌های مختلف با هم قاطی نمی‌شوند.
- مثلاً یک پروژه ممکن است Django 3.2 بخواهد و دیگری — Django 5.0 بدون virtualenv این‌ها تداخل پیدا می‌کنند.

تشبیه :مثل داشتن یک آشپزخانه جدا برای هر رستوران، با مواد اولیه مخصوص خودش.

8.تفاوت Django REST Framework وDjango Templates

ویژگی	Django Templates	Django REST Framework
خروجی	HTMLبرای مرورگر	JSON / XMLبرایAPI
هدف	ساخت صفحات وب	ساختAPI
کاربرد	وقتی بخوای سمت سرور HTML بدهی	وقتی بخوای داده خام بدی به کلاینت
مثال	وبلاگHTML	APIاینستاگرام

9.ساخت یک پروژه سادهDjango

دستورات اصلی:

#ساخت پروژه

```
django-admin startproject myproject
```

#ورود به پوشه پروژه

```
cd myproject
```

#اجرای سرور محلی

```
python manage.py runserver
```

ساختار پوشه پروژه:

```
myproject/
|
├ manage.py          # فایل اجرایی اصلی
├ myproject/         # پوشه تنظیمات پروژه
|      ├── settings.py     # تنظیمات کلی پروژه
|      ├── urls.py         # مسیرهای URL اصلی
|      ├── wsgi.py / asgi.py# برای سرویس‌دهی
|      └── __init__.py
```

10.ساخت یک App داخلDjango

در Django یک پروژه می‌تواند چندین **App**داشته باشد. هر App یک بخش مستقل از سیستم است (مثل ماژول).

ساخت:App

```
python manage.py startapp blog
```

ساختار:App

```
blog/
|
├── admin.py      # ثبت مدل‌ها در پنل ادمین
├── apps.py       # تنظیمات App
├── models.py     # مدل‌های دیتابیس
├── views.py      # ویوها
├── urls.py       # مسیرهای مربوط به این App (باید خودمان بسازیم)
```

```
├── templates/   # قالب‌های HTML
└── tests.py     # تست‌ها
```

اتصال App به پروژه:

- در فایل settings.py، در INSTALLED_APPS نام اپ را اضافه می‌کنیم:

```
INSTALLED_APPS = [
    ...,
    'blog',
]
```

- در urls.py اصلی، مسیرهای app را اضافه می‌کنیم:

```
from django.urls import path, include

urlpatterns = [
    path('blog/', include('blog.urls')),
]
```

بخش قابل تحویل

توضیح اتفاقات هنگام باز کردنURL

الف)Network

- **مرورگر به آدرس 127.0.0.1:8000درخواست HTTP ارسال می‌کند (پورت 8000 یعنی سرور محلیDjango).**
- **درخواست از طریق پروتکل TCP/HTTP به سرور Django می‌رسد.**

ب)Request Path

- **مسیر (Path) درخواست بررسی می‌شود (در اینجا /چون صفحه اصلی است).**

ج)URL Dispatcher

- **Djangoاز فایل urls.pyپروژه شروع می‌کند.**
- **مسیر / را پیدا می‌کند و آن را به myapp.urls می‌فرستد.**
- **myapp.urls هم مسیر / را به تابع helloدر views.py متصل می‌کند.**

د)View Handling

- **تابع hello اجرا می‌شود.**
- **پاسخی از نوع HttpResponse ایجاد می‌کند.**
- **این پاسخ حاوی متن "Hello World <Your Name>"است.**

هـ) Template Rendering (در صورت استفاده)

- **در این مثال از Template استفاده نکردیم (مستقیماً متن برگرداندیم).**
- **اگر استفاده می‌کردیم، Viewداده را به Template می‌داد تا HTML ساخته شود و سپس پاسخ داده شود.**