

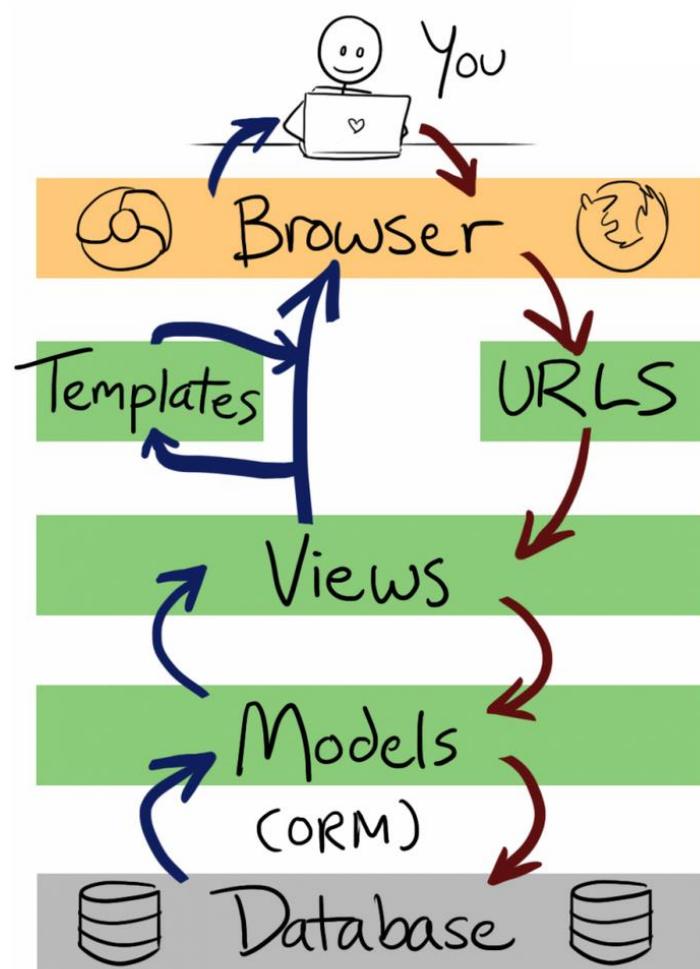
What Is Django?

Django is a **high-level Python web framework** used to build websites and web applications quickly and securely. It follows a design philosophy called **MVT (Model–View–Template)**.

It's used by companies like Instagram and Mozilla because it is scalable, secure, and fast to develop with.

□ Django Architecture Explained for Beginners

Django follows the **MVT Architecture Pattern**:



Let's break this down step by step in simple terms.

1 Model (M)

What is a Model?

A **Model** defines the structure of your database.

Think of it like:

"How should my data look?"

For example, if you're building a blog website, you might have a model like:

- Title
- Content
- Author
- Date

What Model Does:

- Defines database tables
- Handles database operations (Create, Read, Update, Delete)
- Talks directly to the database

Example (Conceptual)

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

Here Django automatically creates a database table for you.

2 View (V)

What is a View?

A **View** contains the business logic of your application.

Think of it like:

"What should happen when someone visits this page?"

It:

- Receives user requests
- Gets data from the model
- Sends data to a template
- Returns a response

Example Flow:

User visits:

www.site.com/posts

View:

- Fetches all blog posts from database
- Sends them to the template
- Returns an HTML page

3 Template (T)

What is a Template?

A **Template** controls how your website looks (UI).

Think of it like:

"How should the data be displayed?"

Templates:

- Use HTML

- Can include Django template language ({{ }})
- Display dynamic data

Example:

```
<h1>{{ post.title }}</h1>
<p>{{ post.content }}</p>
```

□ How Everything Works Together (Full Request Cycle)

Let's understand what happens when a user visits your website:

Step 1: User Sends Request

User types:

www.site.com/posts

Step 2: URL Dispatcher

Django checks `urls.py` to see which view should handle this URL.

Step 3: View Executes

The view:

- Talks to the model
- Fetches data from database

Step 4: Model Interacts with Database

Django ORM converts Python code into SQL queries.

Step 5: Template Renders HTML

Template combines:

- Static HTML
- Dynamic data

Step 6: Response Sent Back

User sees the final web page.

Important Components of Django Architecture

□ 1. URL Dispatcher (urls.py)

- Maps URLs to views
- Acts like a traffic controller

Example:

```
path('posts/', views.post_list)
```

□ 2. Django ORM (Object Relational Mapper)

Django has a built-in ORM.

Instead of writing SQL like:

```
SELECT * FROM posts;
```

You write:

```
Post.objects.all()
```

Django converts it into SQL automatically.

□ 3. Settings File (settings.py)

Controls:

- Database configuration
 - Installed apps
 - Middleware
 - Static files
 - Security settings
-

□ 4. Middleware

Middleware sits between:

- Request
- Response

It handles:

- Authentication
- Security
- Sessions
- Logging