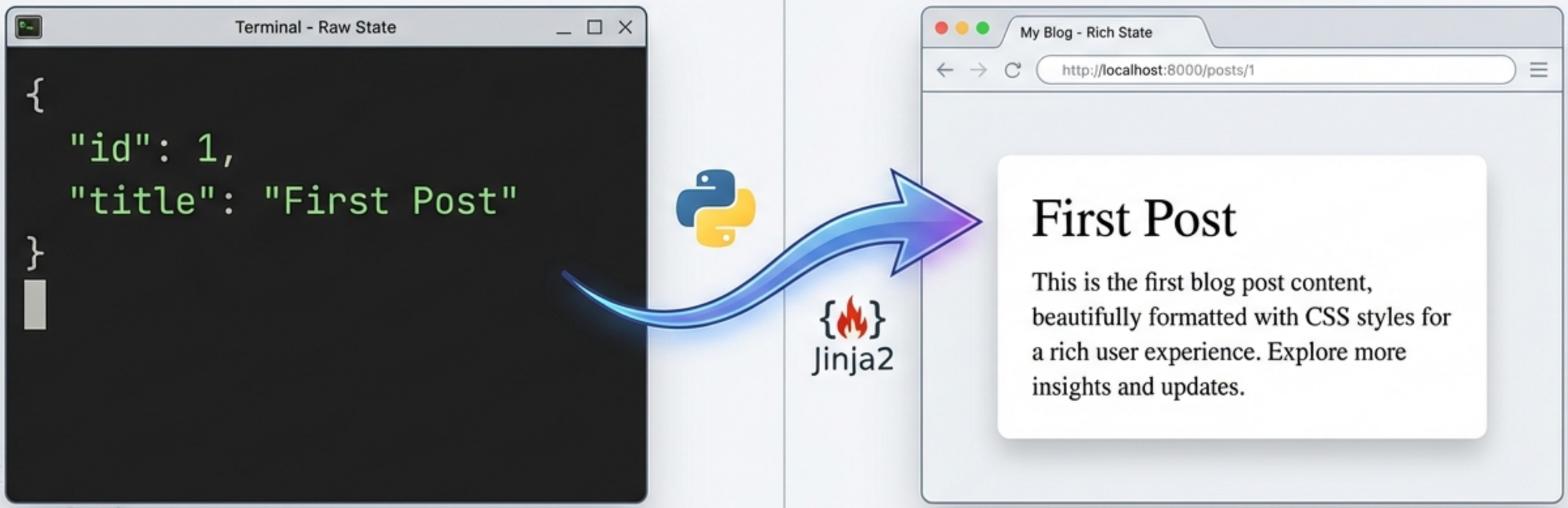
Building an HTML Frontend with FastAPI & Jinja2

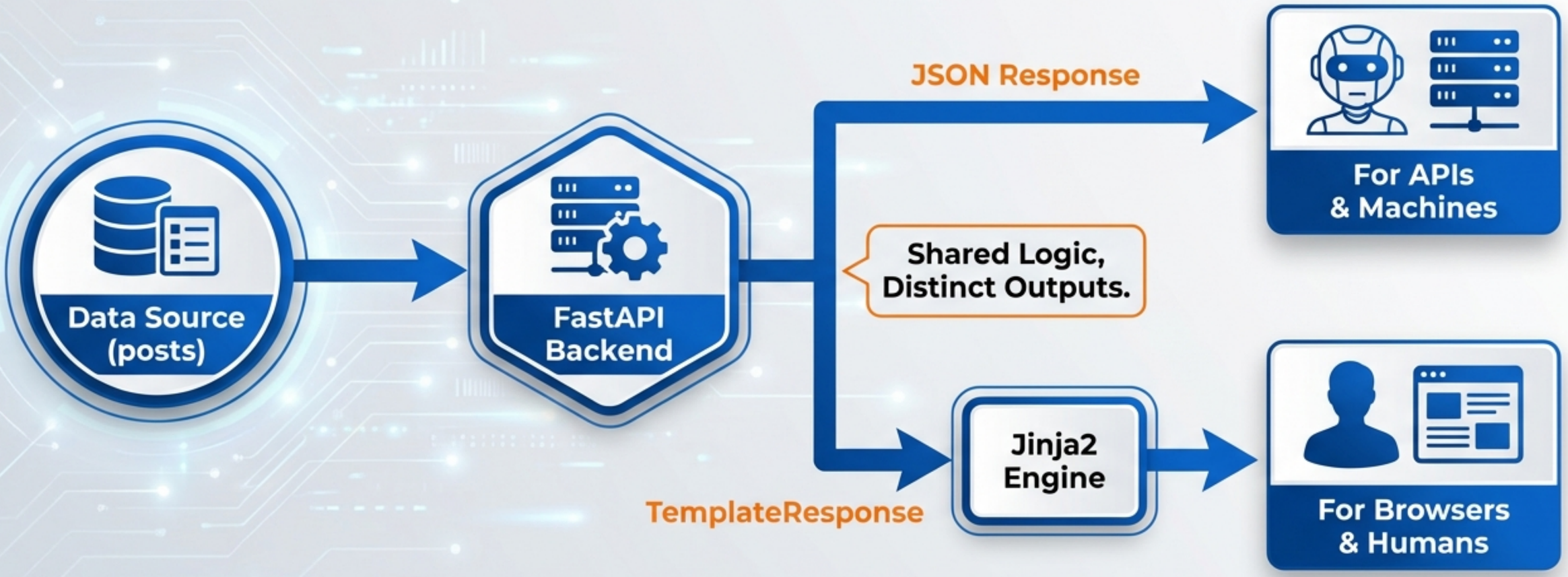
Transforming Raw JSON into Rich User Interfaces



Based on the tutorial by Corey Schafer

THE HYBRID ARCHITECTURE

Serving Machines and Humans from One Source



PROJECT STRUCTURE & DEPENDENCIES

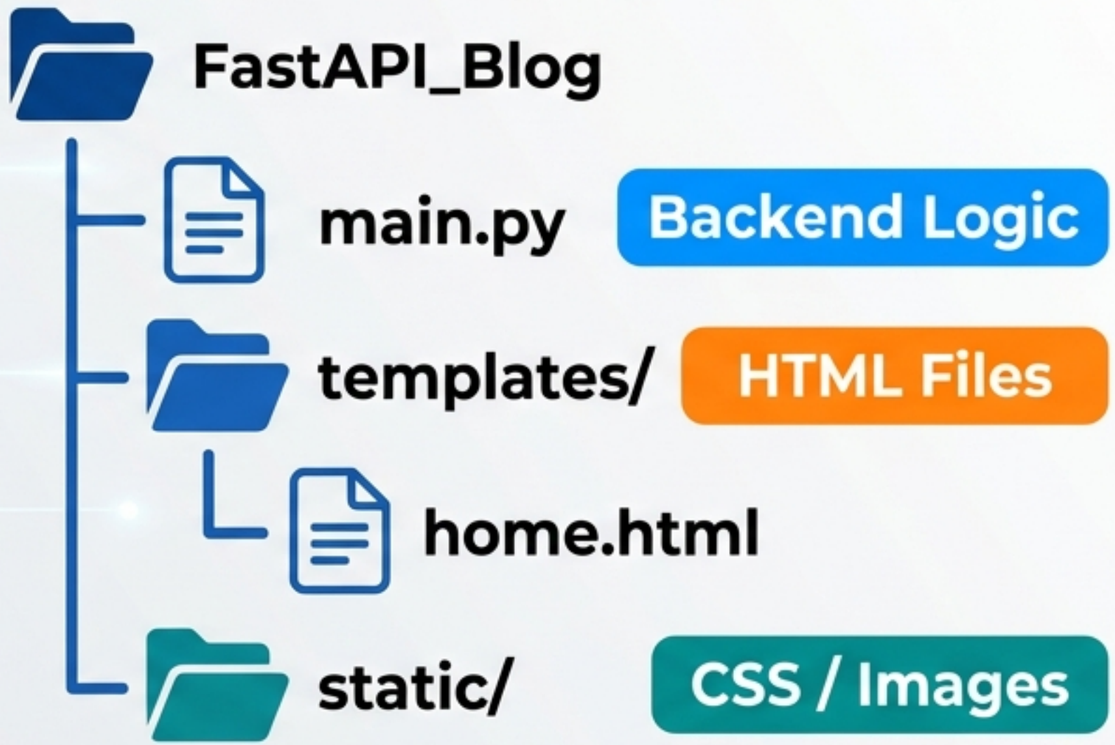
Serving Machines and Humans from One Source

Installation

- If using `**fastapi[standard]**`, Jinja2 is included.
- Otherwise: `**pip install jinja2**`

The Convention

FastAPI requires specific directory names to auto-detect files.



CONFIGURING THE ENGINE

Setup in main.py

```python
from fastapi import FastAPI, Request
from fastapi.templating import Jinja2Templates

app = FastAPI()

# Mount the templates directory
templates = Jinja2Templates(directory="templates")
```

Required by Jinja2 to build context.

Tells FastAPI where to look for HTML files.

SERVING THE FIRST TEMPLATE

Transitioning from Strings to Files

❌ The Old Way (Raw String)

```python
@app.get("/")
def home():
    return HTMLResponse("<h1>Hello World</h1>")
```

✅ The New Way (Template Response)

```python
@app.get("/", include_in_schema=False)
def home(request: Request):
    return templates.TemplateResponse(request, "home.html")
```

Key Takeaways

📝 **Argument:** Must accept `request: Request`.

📲 **Return:** Pass `request` back to `TemplateResponse`.

📝 **Docs:** `include_in_schema=False` hides HTML routes from Swagger UI.

INJECTING DATA: THE CONTEXT DICTIONARY

Bridging Python variables to HTML

```python
return templates.TemplateResponse(
    request,
    "home.html",
    {"request": request, "posts": posts_list}
)
```

The Context Dictionary makes backend data available to the frontend template.

JINJA2 SYNTAX ESSENTIALS

Cheat Sheet

Printing Variables

{{ variable }}

Outputs the value to the HTML.

```
<h2>{{ post.title }}</h2>
```

📝 **Supports dot notation (e.g., `post.title` instead of `post["title"]`).**

Control Flow

{% logic %}

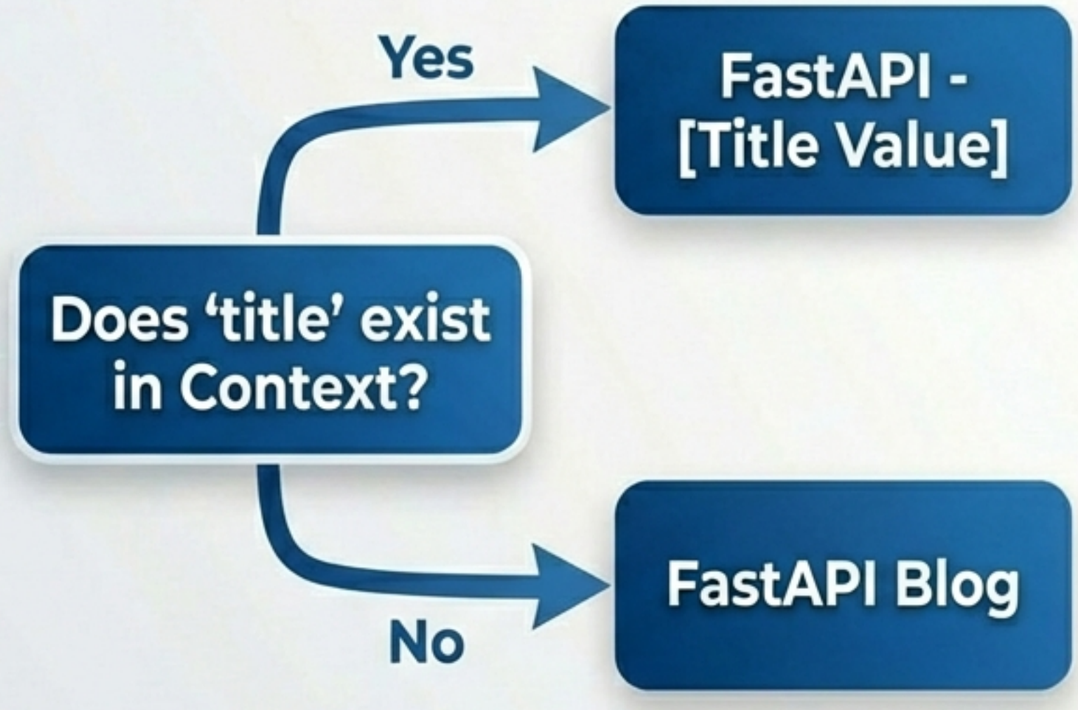
Executes loops and conditions.

```
{% for post in posts %}
  <p>{{ post.content }}</p>
{% endfor %}
```

CONDITIONAL LOGIC

Dynamic Page Titles

```
<title>
    {% if title %}
        FastAPI - {{ title }}
    {% else %}
        FastAPI Blog
    {% endif %}
</title>
```

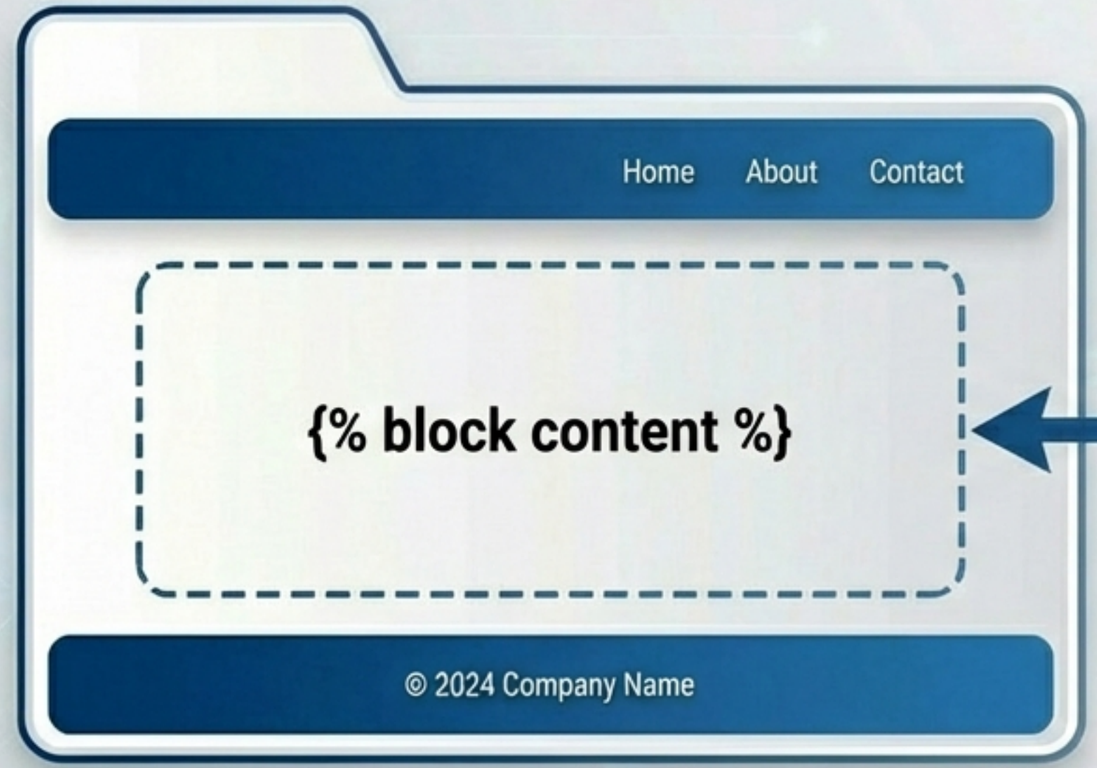


Requires passing `title` in the Python context dictionary.

JINJA2 SYNTAX ESSENTIALS

The DRY Principle: Template Inheritance

Don’t Repeat Yourself



Parent Template: layout.html
(“frame and insert”)

```
<h1>Welcome to My Blog</h1>
<p>This is my first post using
template inheritance.</p>
{% endblock %}
```

Child Template: home.html

In Child:

```
{% extends "layout.html" %}

{% block content %}
  <h1>Welcome to My Blog</h1>
  <p>This is my first post using template inheritance.</p>
{% endblock %}
```

MANAGING STATIC ASSETS

CSS, Images, and JavaScript

```python
from fastapi.staticfiles import StaticFiles

# Mount the static directory
app.mount("/static", StaticFiles(directory=
    directory="static"),
    name="static")
```

- "/static": "URL Path (e.g., example.com/static/style.css)"
- directory="static": "Internal Folder Name"
- name="static": "Internal Reference Name for Reverse Lookups"

DYNAMIC URL GENERATION

Why we use `url_for`

The Problem: Hardcoding

```
<link href="/static/css/main.css">
```

Brittle. Breaks if folder structure or mount path changes.

The Solution: Dynamic Generation

```
{{ url_for('static', path='/css/main.css') }}
```

Resilient. Automatically resolves the correct path.

Uses:

- CSS & JS imports
- Images (`path='/img/logo.png'`)
- Navigation Links (`url_for('home')`)

RESOLVING ROUTE CONFLICTS

Fixing ambiguous navigation links

The Bug

If multiple routes share a function, `url_for` might pick the wrong path.

```
@app.get("/")
@app.get("/posts")
def root(): ...
```

The Fix: Explicit Naming

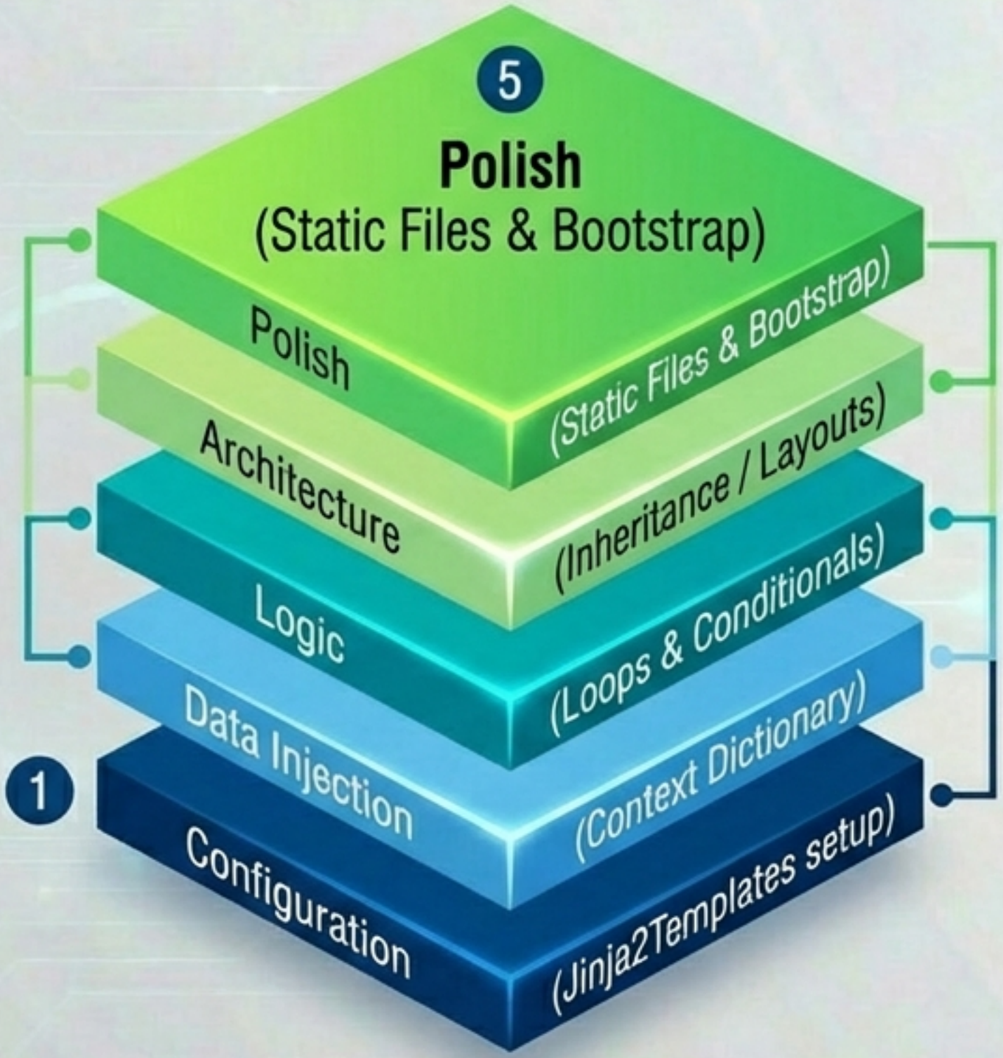
Assign unique names to each decorator.

```
@app.get("/", name="home")
def root(): ...

@app.get("/posts", name="posts")
def root(): ...
```

Result: `{{ url_for('home') }}` now explicitly targets the root path.

THE COMPLETE BUILD RECAP



Result: A **Hybrid FastAPI Application** serving both **JSON API** and **HTML Frontend**.

NEXT STEPS & RESOURCES

COMING UP NEXT

- URL Parameters & Dynamic Routes
- Viewing individual posts (`/post/{id}`)
- Error Handling

RESOURCES

- **Source Code:** Link in description
- Includes: `layout_finished.html` & `static_finished` assets
- **Credits**: Based on the FastAPI Series by Corey Schafer