Server-Side Templates

Web Dev, Spring 2021

Let's go back....

We talked about having a web application server to hold data

- persistence, sharing

Our architecture for using data inform the server in the frontend:

- server sends an HTML document that depends on some server data
- use JavaScript to fetch data from the server via a GET or a POST
- use JavaScript update the document with the data
  - we've done it in raw Javascript with some MVC patterns
  - we've done it in ReactJS
- difference in degree, not in kind

An alternative

When the browser asks for an HTML document that depends on server data:

- dynamically build the document using the data on the server
- send the document to the browser
- the browser doesn't know it it's using a dynamically-generated document

Mostly useful for pages that depend on data directly pulled from a DB

Very popular in the early days of e-commerce and of blogs

- one page per product, each page the same but uses different product info + reviews
- one page per blog post, each page the same but uses different text + comments

Dynamic document creation in web application servers

ASP.NET — "Active Server Pages"

JSP — "Java Server Pages"

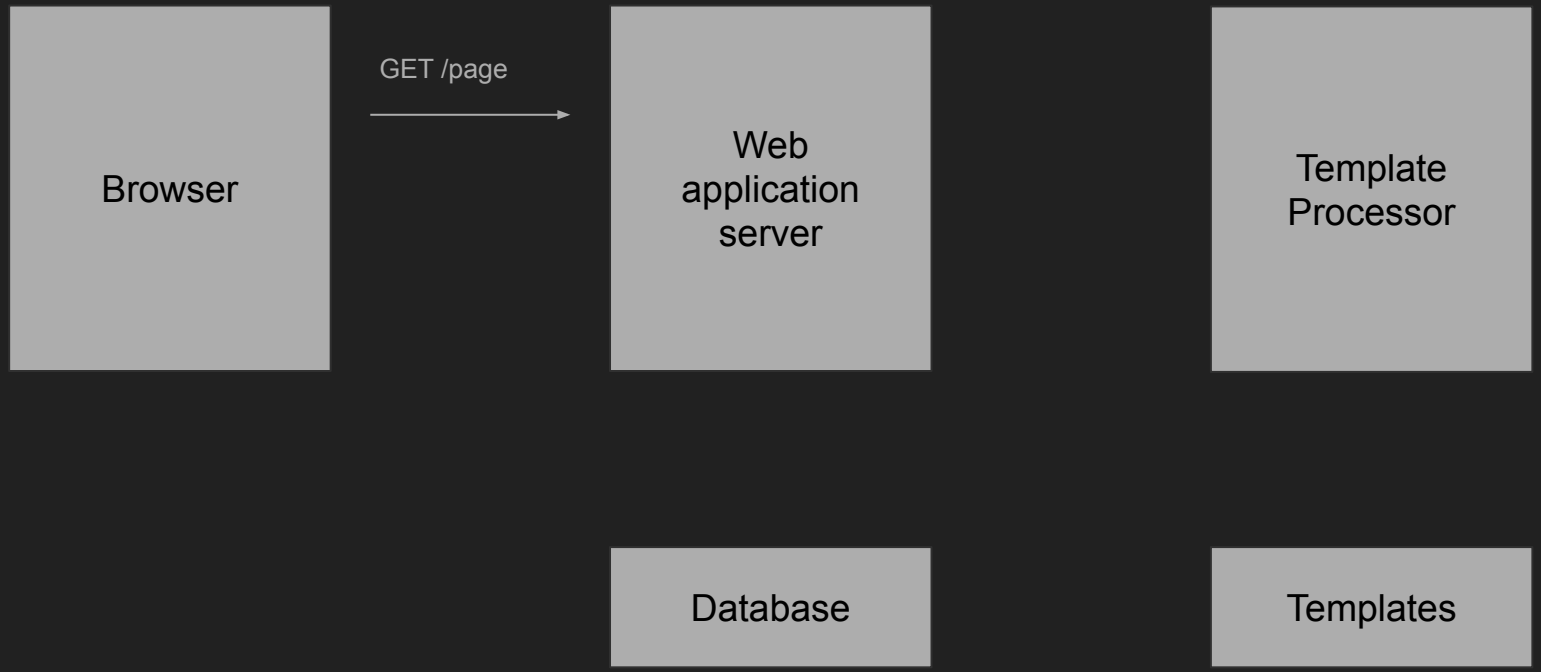
Both heavy duty web application servers with facilities to build HTML documents dynamically using C# (or Visual Basic) and Java, respectively

Ruby on Rails emerged as a more palatable way of doing the same thing

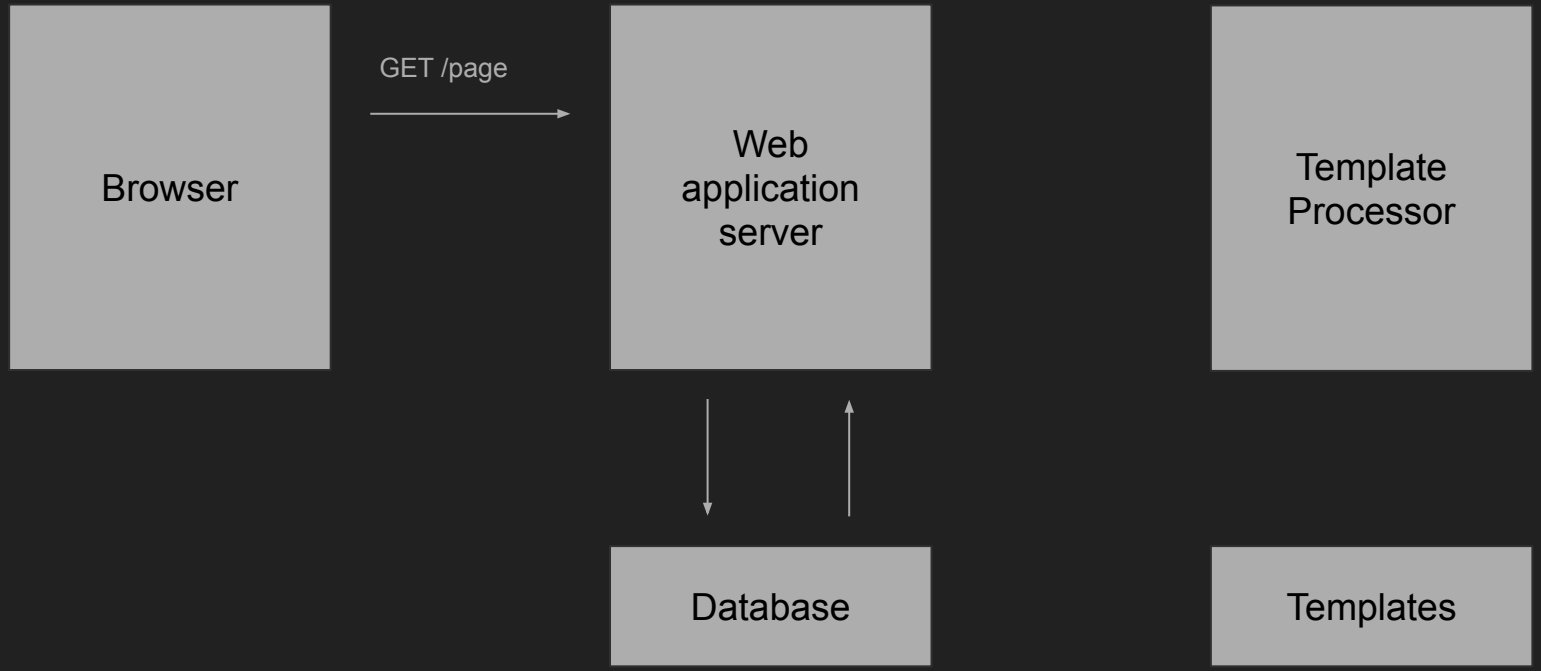
Most of these frameworks are based on filling HTML templates

PHP is a "lightweight" way of doing the same — not tied to a specific server

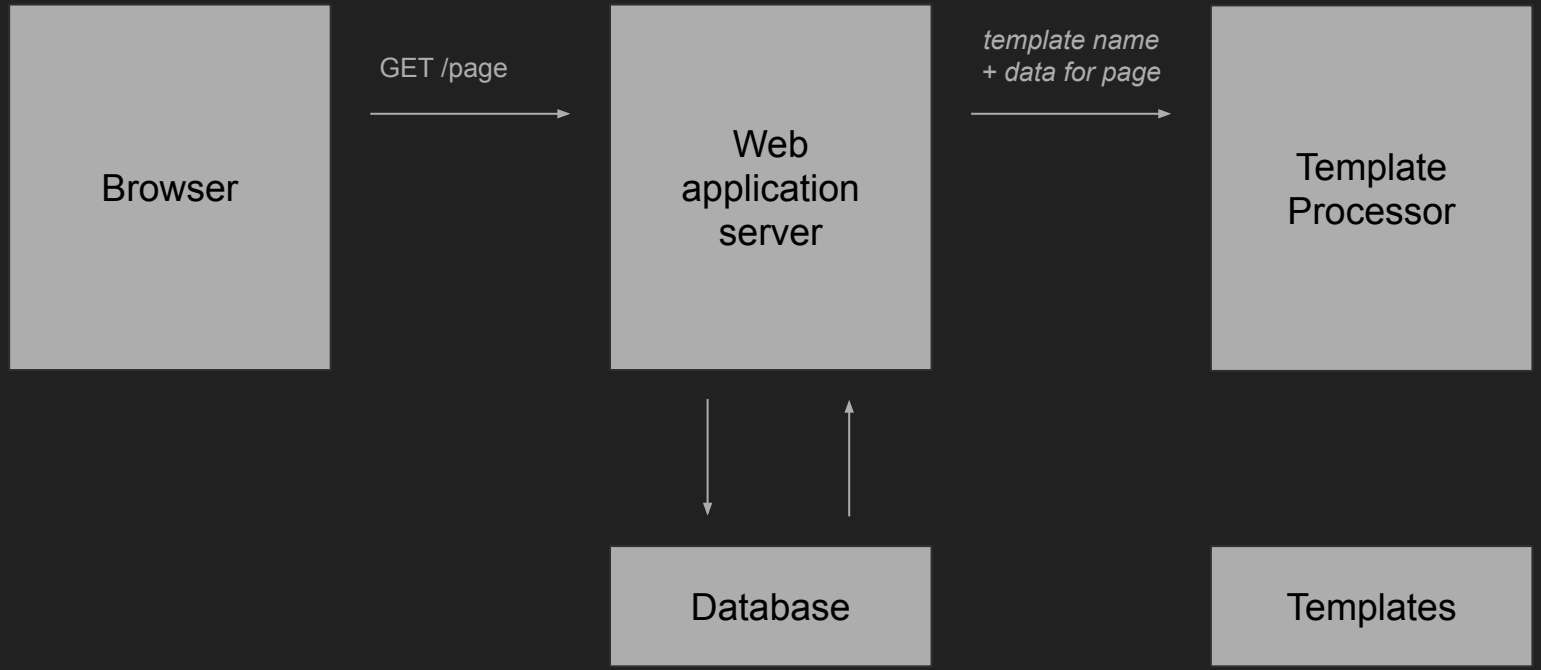
Basic architecture



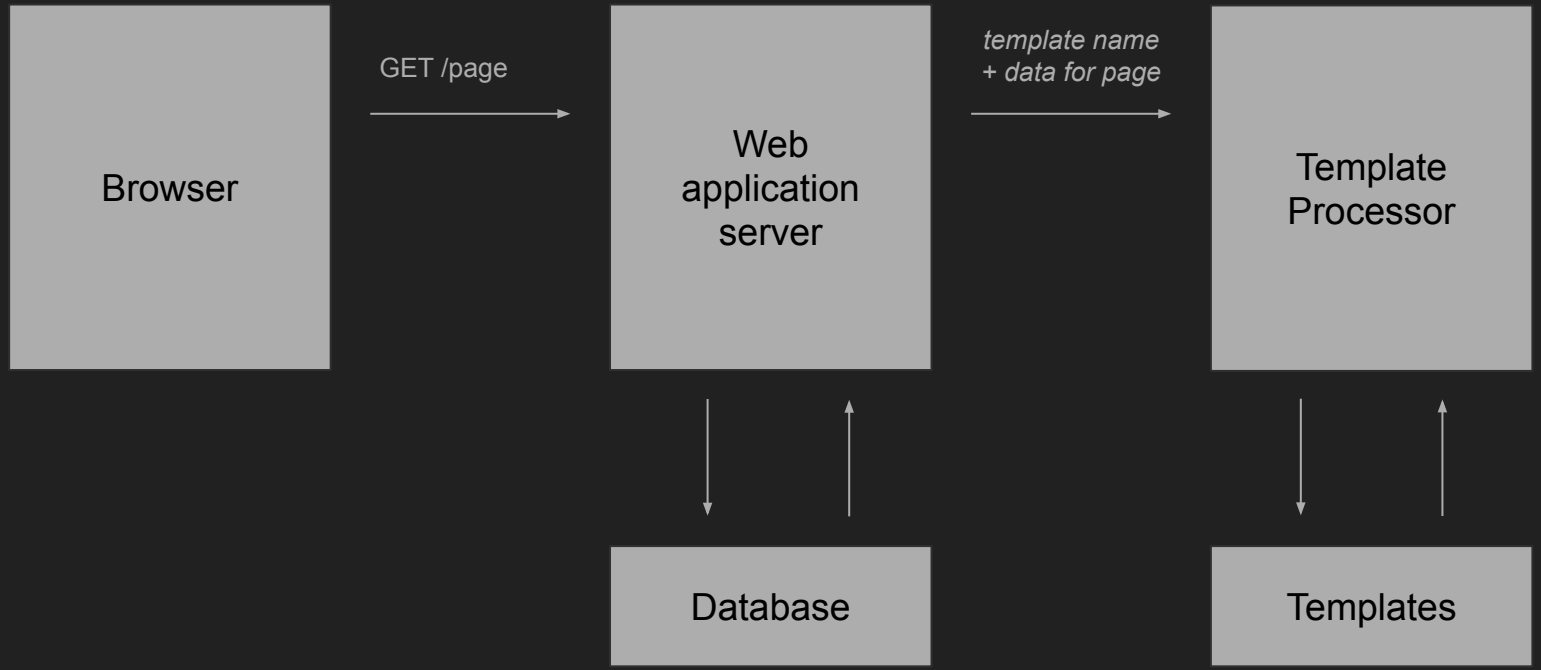
Basic architecture



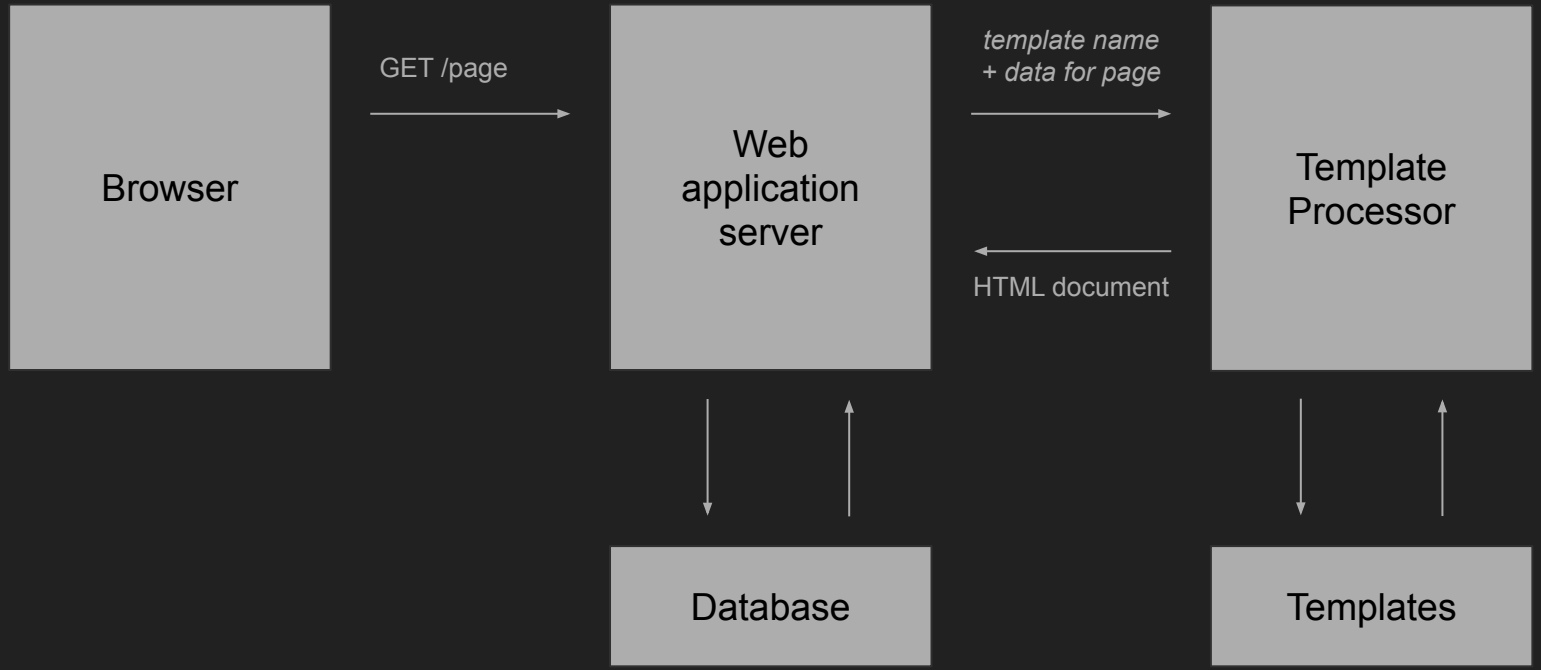
Basic architecture



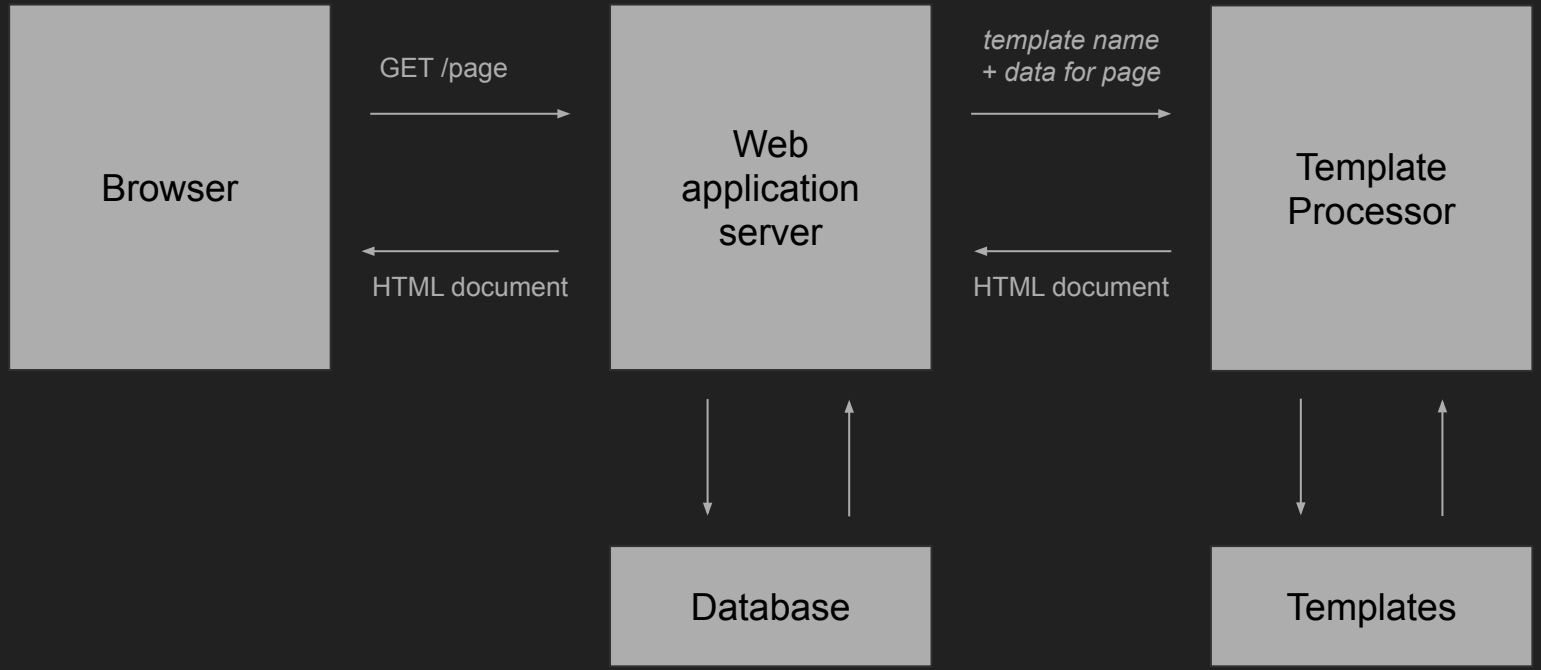
Basic architecture



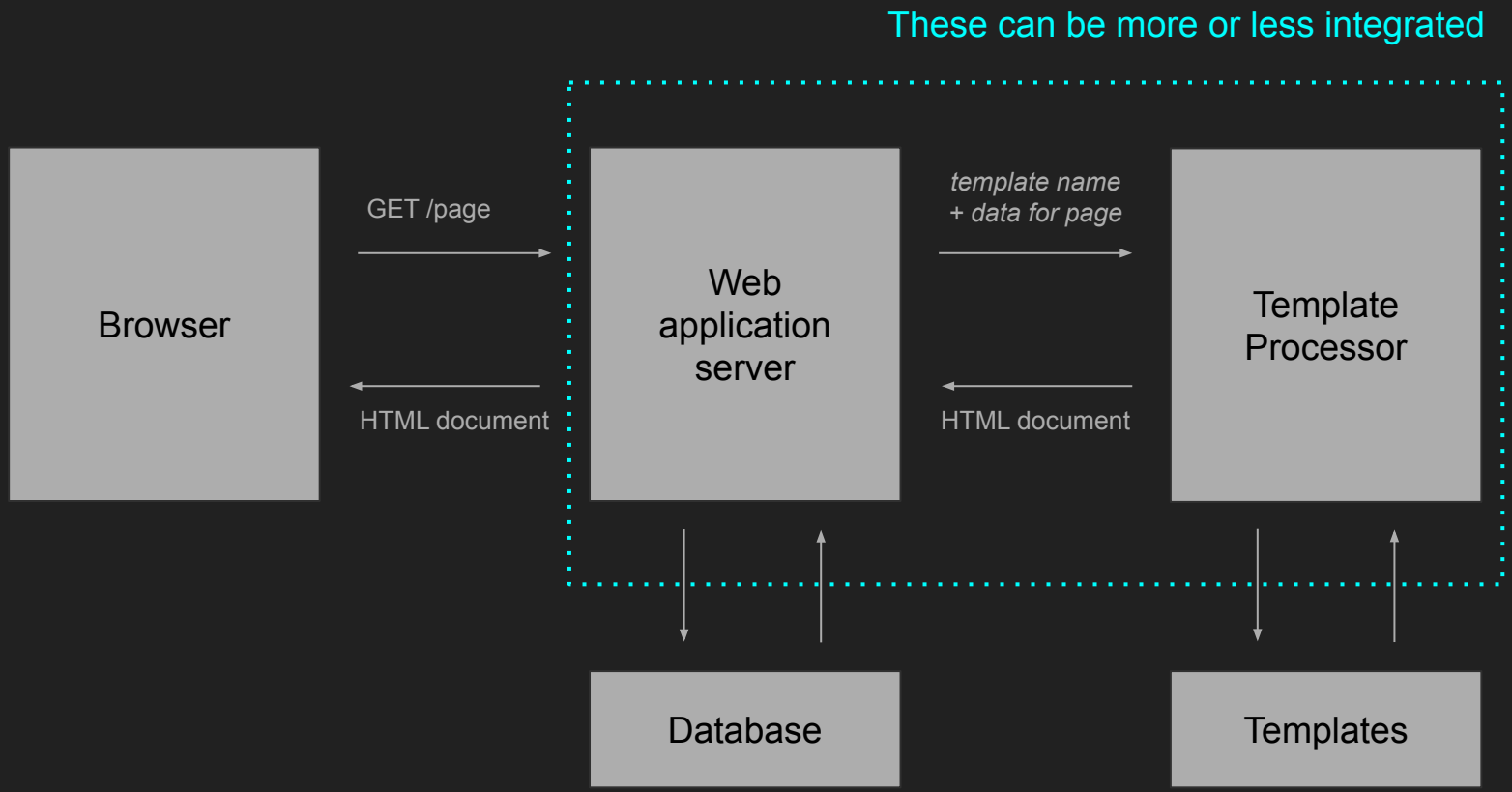
Basic architecture



Basic architecture



Basic architecture



Flask + Jinja2

Jinja2 is a template processor with a Python-like syntax

Integrated with Flask

- you can ask flask to render a template, passing it values it can use
- the Jinja2 processor has access to those values passed by Flask

A Jinja2 template — product.jj2

```
<html>
  <body>
    <h1>Product: {{ info['name'] }}</h1>
    <img src="{{ info['image'] }}"></img>
    <h2>Reviews</h2>
    <div class="reviews">
      {% for r in info['reviews'] %}
      <p> {{ r }} </p>
      {% endfor %}
    </div>
  </body>
</html>
```

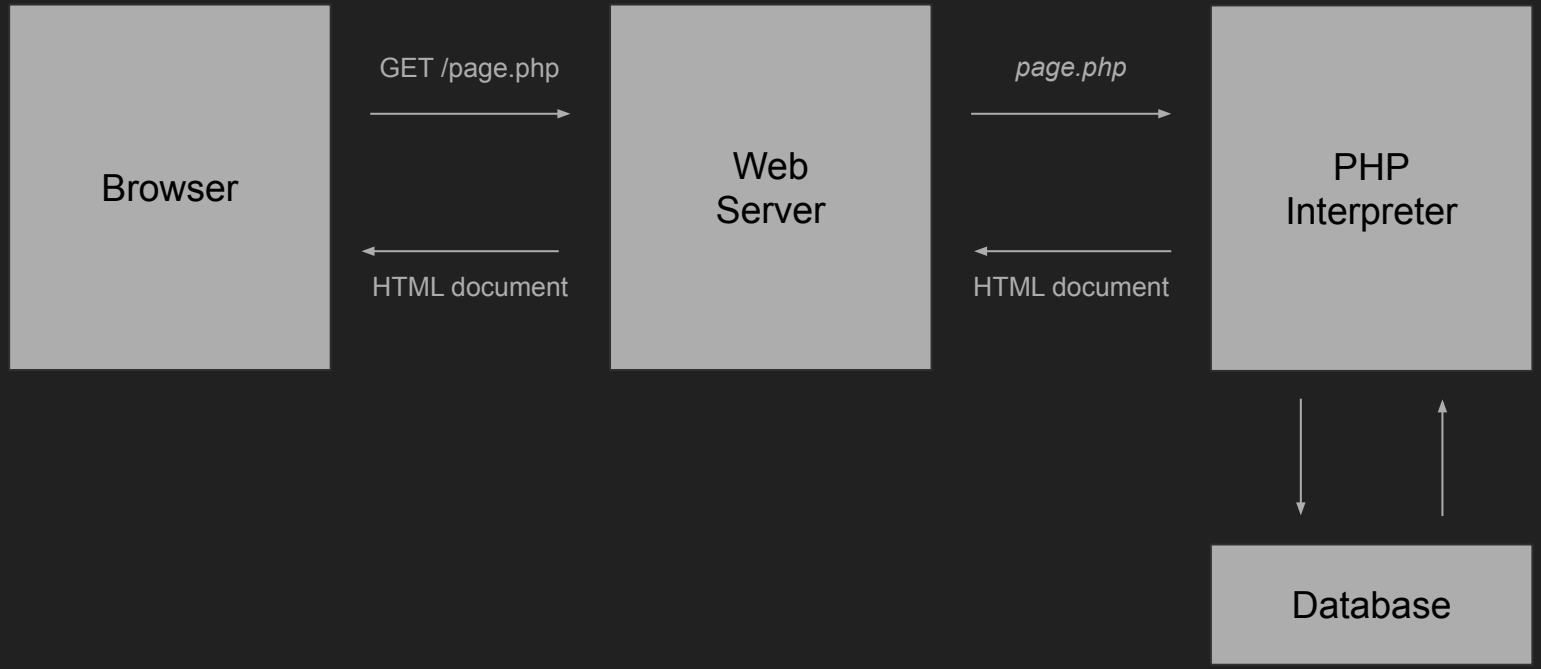
Rendering a template from Flask

```
@app.route('/product/<int:id>')
def product_route (id):
    info = get_product_info_from_db(id)
    return render_template('product.jj2', id=id, info=info)
```

Demo

Homework 4 revisited (again)

PHP


Browser
GET /page.php
HTML document
Web Server
page.php
HTML document
PHP Interpreter
Database

PHP

```
<html>
  <body>
    <h1>Blog Post</h1>
    <p>
      <?php
          $id=_GET['id'];
          $text = getBlogPostTextFromDatabase($id);
          echo $text;
       ?>
  </body>
</html>
```