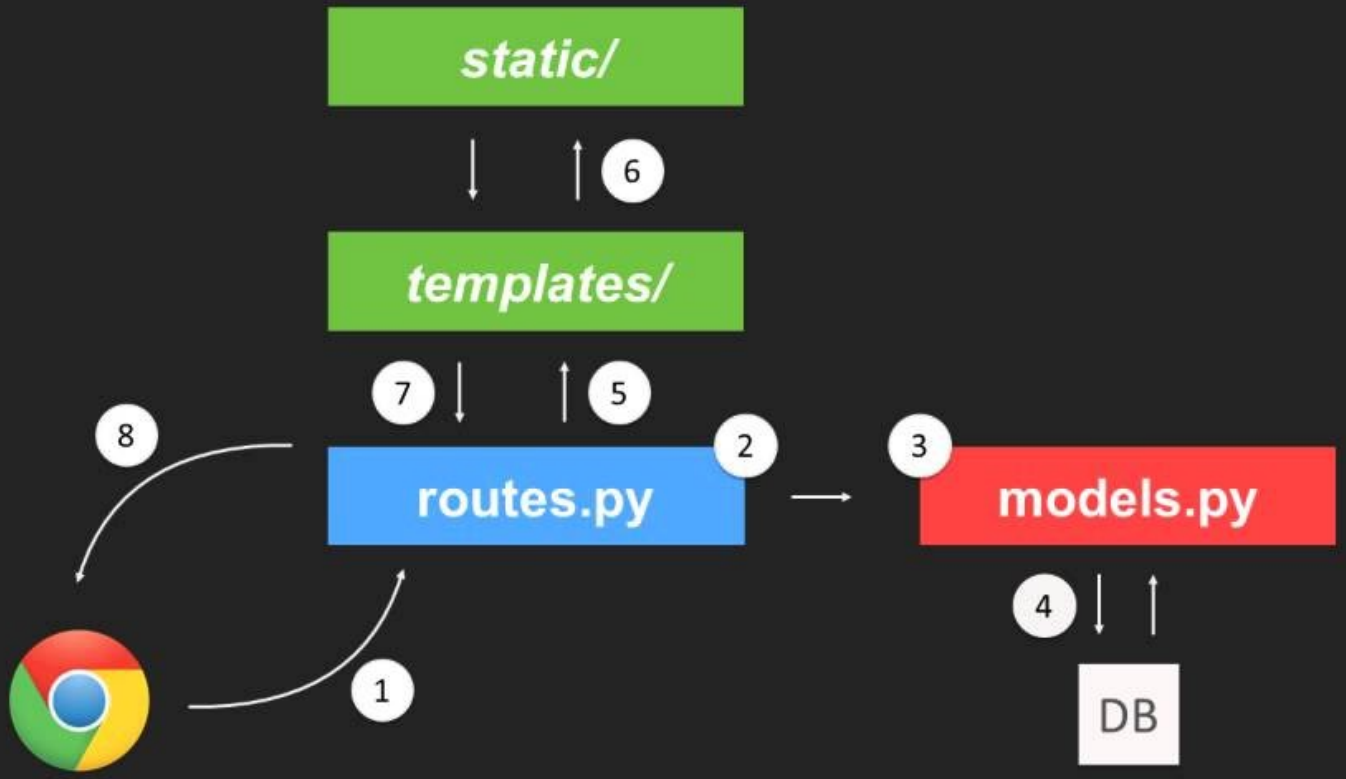
BUILDING A PYTHON APP IN

FLASK

Реализация Web сайта на Flask

Web приложения на Flask могут быть реализованы как REST API сервисы а могут отдавать контент в виде web страниц. Рассмотрим подробно как происходит отдача html страниц.

The Request-Response Cycle



Из предыдущего слайда видно что HTTP запрос (request) приходит на сервер (web server) разбирается под капотом URL а затем вызывается конкретный роутер который этот запрос обрабатывает. Далее идет процесс извлечения данных из БД с помощью модели, это может быть (SQLAlchemy) или простые SQL запросы через библиотеку psycopg2. Как только мы получили данные идет вызов шаблонизатора файлы которого находятся в папке templates. Файлы как правило имеют расширение .html Данные файлы включают в себя код разметки, стили , ссылки на медия ресурсы. Все что относится к медия ресурсам хранится в папке static.

После формирования страницы (шаблон + данные из БД) идет ответ сервера (request) из роутера браузеру.

Технологический стрек для Web 2.0

- HTML5/CSS3
- Препроцессоры CSS - SASS/LESS
- Язык JavaScript/TypeScript
- Ajax
- Web Socket

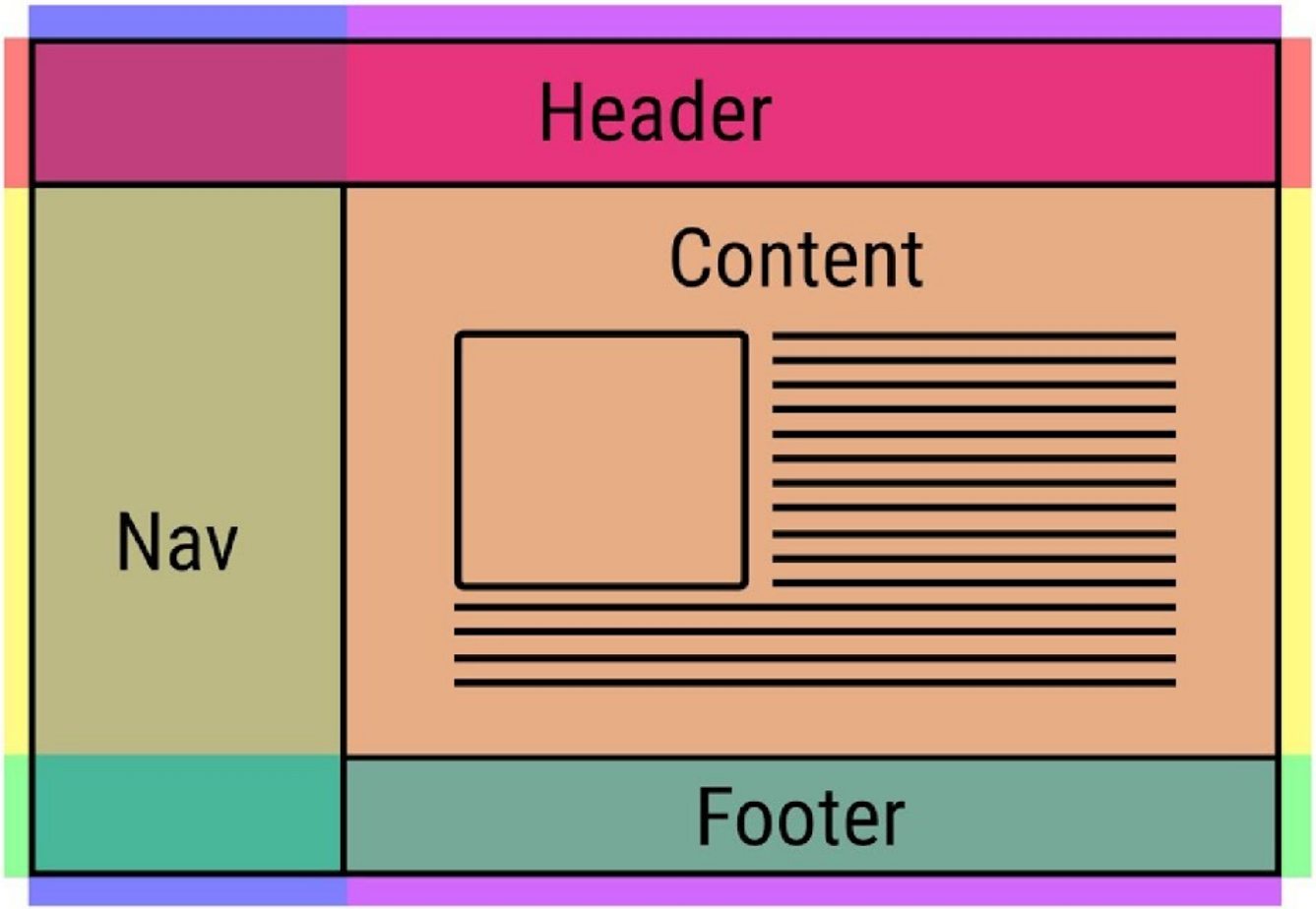
HTML5

HTML5 – стандарт языка гипертекстовой разметки. Служит для структурирования и представления материалов в сети WWW

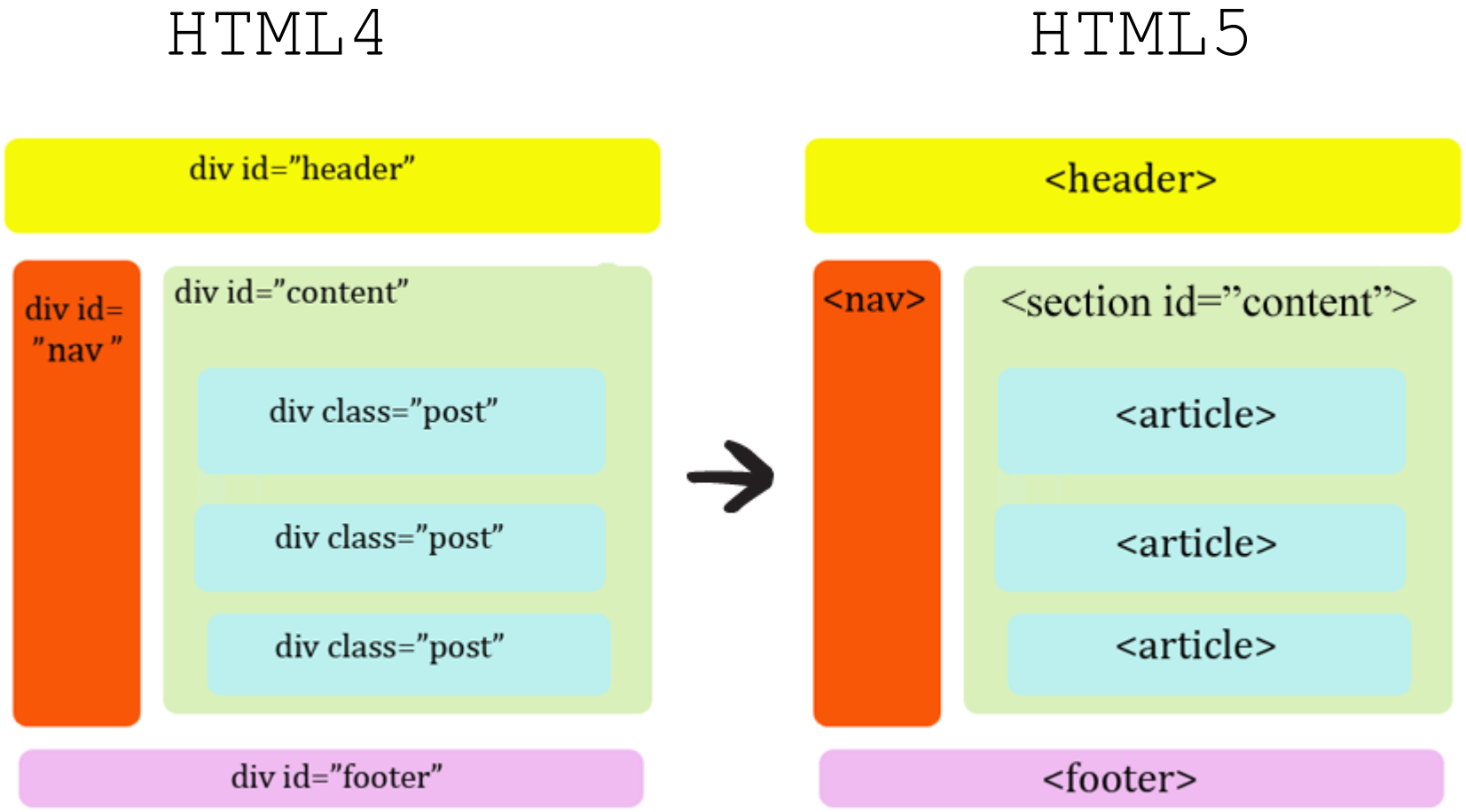
пример простой страницы

```
<!DOCTYPE html>                    <!-- тип документа -->
<html>                             <!-- начало документа -->
  <head>                               <!-- начало заголовка -->
    <meta charset="utf-8">
    <title>Главная страница</title>
  </head>
  <body>                             <!-- тело документа -->
        Привет
  </body>
</html>                             <!-- окончание документа -->
```

Как сделать шаблон вида ?



Нужно его разметить(блочная верстка).



Пример верстки макета.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>nav</title>
 </head>
 <body>
  <header>
    <h1>Чебурашка и крокодил Гена</h1>
  </header>
  <nav><a href="1.html">Чебурашка</a> | <a
href="2.html">Гена</a> |
       <a href="3.html">Шапокляк</a> | <a
href="4.html">Лариска</a></nav>
  <article>
    <h2>Добро пожаловать!</h2>
  </article>
   <footer> Мой сайт @2022 г.</footer>
 </body>
</html>
```

Как сделать красиво ?

CSS3 – каскадные таблицы стилей предназначены для задания элементам оформления: размеры блоков, фон, цвета, рамки, отступы, шрифты, эффекты и т.д.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Глобальные стили</title>
  <style>    <!-- способ 1 , в загловке html ->
   H1 {
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
 <body>
  <h1>Hello, world!</h1>
 </body>
</html>
```

Подключение стилей

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Стили</title>
  <link rel="stylesheet" href="mysite.css">
  <link rel="stylesheet" href="http://www.htmlbook.ru/main.css">
 </head>
 <body>
   <h1>Заголовок</h1>
   <p>Текст</p>
 </body>
</html>
```

Содержимое mysite.css

```
# файл mysite.css
H1 {
   font-size: 120%;
   font-family: Verdana, Arial, Helvetica, sans-
   serif;
   color: #333366;
}
```

После того как мы создали красивую страницу. Мы должны ее отдать по http запросу.

Можно отдать страницу прямо в роутере но такой подход очень грамоздкий и плохо сопровождаемый.

```
from app import app

@app.route('/')
@app.route('/index')
def index():
    user = { 'nickname': 'Miguel' } # выдуманный
пользователь
    return '''
       <html>
           <head>
              <title>Home Page</title>
           </head>
           <body>
             <h1>Hello, ''' + user['nickname'] + '''</h1>
           </body>
       </html>
     '''
```

Лучший способ воспользоваться шаблонизатором

создадим файл в папке /templates/index.html

```
<html>
  <head>
    <title>{{title}} - microblog</title>
  </head>
  <body>
      <h1>Hello, {{user.nickname}}!</h1>
  </body>
</html>
```

Добавим функцией render_template

В функцию render_template передаем шаблон и данные. Функция сама сопоставляет данные с метками в шаблоне и в итоге отдает сгенерированную страницу.

```
from flask import render_template
from app import app
@app.route('/')
@app.route('/index')
def index():
    user = { 'nickname': 'Miguel' } # выдуманный пользователь
    return render_template("index.html",
        title = 'Home',
        user = user)
```

Загружаем стили

Если в шаблоне есть css , картинки и др. медия ресурсы то создаем папку **static** и система сама их отдаст по указанным относительным ссылкам в шаблоне.

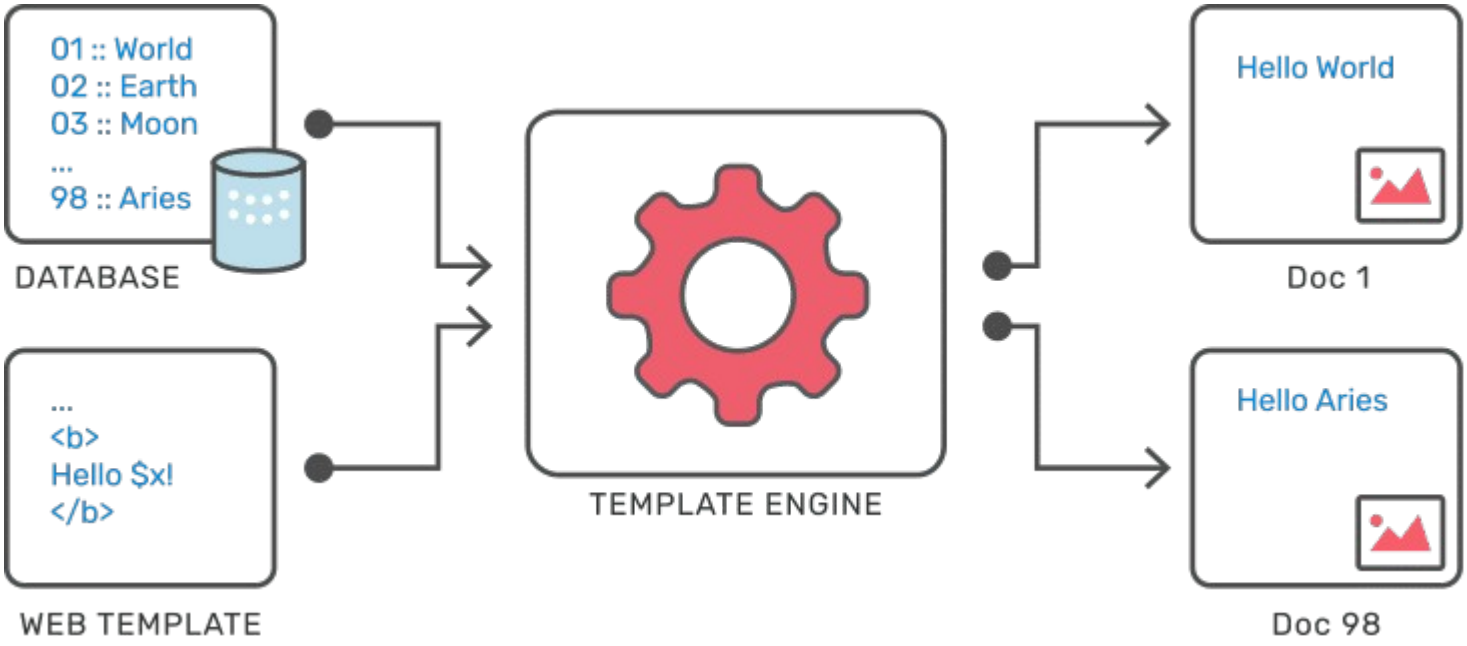
Пример:

```
<html lang="ru" >
 <head>
   <meta charset="utf-8">
   <title>Flask Parent Template</title>
   <link rel="stylesheet" href="{{ url_for('static',
filename='css/default.css') }}">
 </head>
 <body>
    <header>
        <h1 class="logo">First Web App</h1>
        <nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav>
    </header>
    {% block content %}
    {% endblock %}
 </body>
</html>
```

Шаблонизация Jinja

Шаблонизатор — это инструмент, который позволяет проще писать разметку, делить её на компоненты и связывать с данными. Главное преимущество шаблонизаторов — они избавляют от необходимости писать повторяющийся код несколько раз.

Template engine (Jinja)



Base Template

```
# Шаблонизация дает возможность создавать базовый шаблон и
наследовать его дочерними шаблонама. А также разбивать
страницу на блоки.
# файл base.html
<!DOCTYPE html>
<html lang="en">
<head>
    {% block head %}
    <link rel="stylesheet" href="style.css" />
    <title>{% block title %}{% endblock %} - My
Webpage</title>
    {% endblock %}
</head>
<body>
    <section>{% block content %}{% endblock %}</section>
    <footer>
        {% block footer %}
        &copy; Copyright 2008 by <a
href="http://domain.invalid/">you</a>.
        {% endblock %}
    </footer>
</body>
</html>
```

Child Template

```
# Создаем станицу  page.html на базе родительского шаблона
# Переопределяем содержимое блоков.
{% extends "base.html" %}
{% block title %}Index{% endblock %}
{% block head %}
    {{ super() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
{% block content %}
    <h1>Index</h1>
    <p class="important">
      Welcome to my awesome homepage.
    </p>
{% endblock %}
```

Генерируем page.html

```
from flask import render_template
from app import app
@app.route('/')
@app.route('/index')
def index():
    return render_template("page.html")
```

Итог

- Создаем шаблоны страницы в html и храним их в папке **templates**
- Медия ресурсы css, img, audio, video и т.д храним в папке **static**
- Отдаем красивую страницу через шаблонизатор Jinja методом **render_template**

Ссылки для самостоятельного изучения HTML5/CSS3

- http://htmlbook.ru/html5
- http://htmlbook.ru/css3
- https://jinja.palletsprojects.com/en/3.0.x/templates/