

Веб без фреймворков

Основы Веб-программирования

Кафедра Интеллектуальных Информационных Технологий, ИНФО, УрФУ

`http:
//lectures.uralbash.ru/6.www.sync/2.codding/index.html`

WSGI - это...?

Для разработки сайтов или Web-приложений на языке Python был утверждён стандарт взаимодействия между Python-приложениями и сервером (например Apache), названный WSGI (“Web Server Gateway Interface”).

Python

per-333

per-3333

Общие принципы

- Веб-сервер
- Разделение кода: **MVC, MTV, RV**
- Маршрутизация URL
- Шаблоны
- Пагинация
- Request/Response
- Статика
- Формы

Веб сервер

Задача Веб сервера - запускать Веб приложения.

Популярные WSGI Веб сервера:

- wsgiref
- Paste
- Waitress
- Gunicorn

wsgiref

```
from wsgiref.simple_server import make_server

def hello_world_app(environ, start_response):
    status = '200 OK' # HTTP Status
    headers = [
        ('Content-type', 'text/plain; charset=utf-8')
    ] # HTTP Headers
    start_response(status, headers)

    # The returned object is going to be printed
    return [b"Hello World"]

with make_server('', 8000, hello_world_app) as httpd:
    print("Serving on port 8000...")

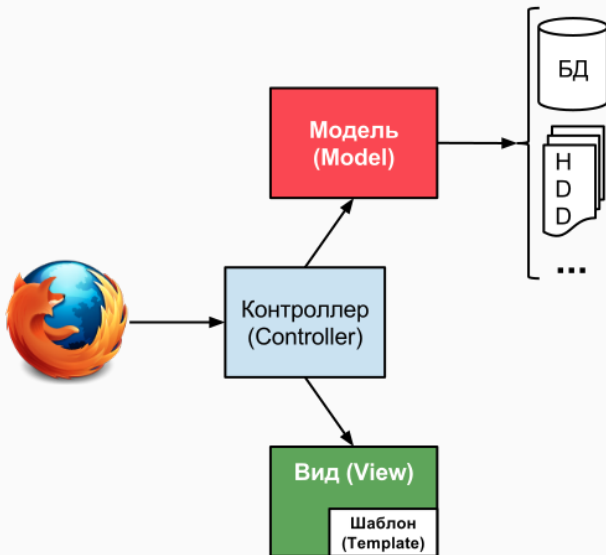
    # Serve until process is killed
    httpd.serve_forever()
```

Разделение кода: **MVC, MTV, RV**

Разделение кода на части

MVC (Model-View-Controller: модель-вид-контроллер) — шаблон архитектуры ПО, который подразумевает разделение программы на 3 слабосвязанных компонента, каждый из которых отвечает за свою сферу деятельности.

Разделение кода: MVC



Разделение кода: **MVC**

Классические **MVC** фреймворки:

- Ruby on Rails
- Pylons

Разделение кода: **MTV**

Фреймворк **Django** ввел новую терминологию **MTV**

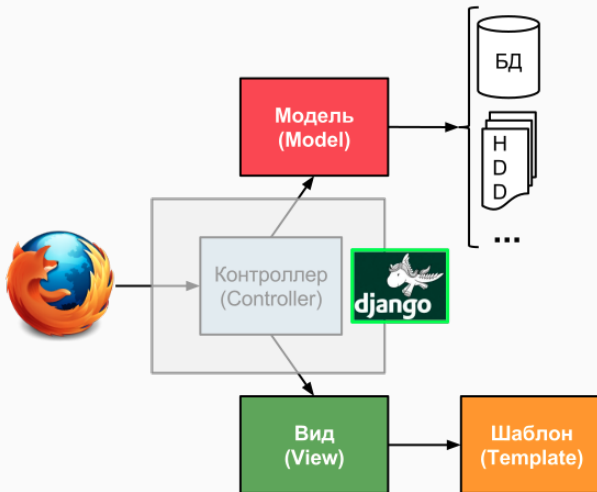
- **M -> M** Модели остались неизменными
- **V -> T** Представление назвали Templates
- **C -> V** Контроллеры назвали Views

Разделение кода: MTV

Tada! Django MTV



Разделение кода: MTV



Разделение кода: **MVC, MTV, RV**

Разработка без фреймворков дает вам возможность придерживаться любой архитектуры приложения и паттерна проектирования.

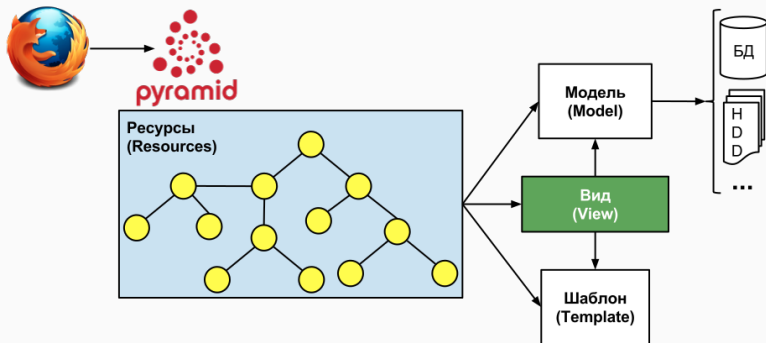
Разделение кода: **MVC, MTV, RV**

Это дает неоспоримую гибкость таким приложениям, оставляя ответственность по структуре ПО за разработчиком.

RV (Resources-View) –

дает ту же гибкость, накладывая минимальную архитектуру идеально вписывающуюся в ограничения Веб приложений.

Разделение кода: RV



Разделение кода: **RV**

“Мы считаем, что есть только две вещи: ресурсы (**Resource**) и виды (**View**). Дерево ресурсов представляет структуру сайта, а вид представляет ресурс. ”

“«**Шаблоны**» (**Template**) в реальности лишь деталь реализации некоторого вида: строго говоря, они не обязательны, и вид может вернуть ответ (Response) и без них. ”

“Нет никакого «**Контроллера**» (**controller**): его просто не существует. «**Модель**» (**Model**) же либо представлена деревом ресурсов, либо «доменной моделью» (domain model) (например, моделью **SQLAlchemy**), которая вообще не является частью каркаса. Нам кажется, что наша терминология более разумна при существующих ограничениях веб-технологий. ”

Pyramid only



Маршруты

или

Диспетчеризация URL

Маршруты: Сортировочная Ж/Д станция



Маршруты: Сортировочная Ж/Д станция

- **Поезда** похожи на HTTP запросы клиентов
- **Сортировочная станция** представляет из себя **маршруты**, только вместо поездов занимается диспетчеризацией HTTP запросов от клиентов
- **Пункт назначения** (завод или вокзал) можно сравнить с функцией (кодом), которая обрабатывает запрос

Маршруты: Определение

Маршруты определяют шаблоны URL и связывают их со своим кодом

Маршруты: Преимущества

В отличие от CGI, где все привязано к файловой системе, если вы измените свое решение по поводу конкретного URL, то просто поменяйте шаблон URL - код по-прежнему будет работать отлично и не понадобится менять какую-либо логику.

Маршруты: Регулярные выражения

```
from django.conf.urls import url

from . import views

urlpatterns = [
    url(r'^articles/2003/$', views.special_case_2003),
    url(r'^articles/([0-9]{4})/$', views.year_archive),
    url(r'^articles/([0-9]{4})/([0-9]{2})/$', views.month_archive),
    url(r'^articles/([0-9]{4})/([0-9]{2})/([0-9]+)/$', views.article_detail),
]
```

Маршруты: Django 2.0 - Сопоставление с образом

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/',
]
```

Маршруты: Сопоставление с образом

```
import selector

dispatch = selector.Selector()
dispatch.add('/', GET=BlogIndex)
dispatch.add('/add', GET=create, POST=create)
dispatch.add('/{id:digits}', GET=BlogRead)
dispatch.add('/{id:digits}/edit', GET=update, POST=update)
dispatch.add('/{id:digits}/delete', GET=delete)
```

Маршруты: Сравнение

Регулярные выражения

/

/article/add

^/article/(?P<id>d+)/\$

^/article/(?P<id>d+)/edit\$

^/article/(?P<id>d+)/delete\$

Сопоставление с образом

/

/article/add

/article/{id:digits}

/article/{id:digits}/edit

/article/{id:digits}/delete

Маршруты: Преимущества

- Регулярные выражения дают огромные возможности.
- Но из-за ограничений описанных в стандарте RFC 1738, большинство из них не нужны.
- Использование регулярных выражений затрудняет читабельность кода.

Шаблоны

Шаблоны имеют очень простое определение - в статические файлы вставляются куски кода, при прогоне таких файлов через специальный транслятор (препроцессор), код заменяется результатом его выполнения.

Шаблоны: C++

```
template< typename T >
T min( T a, T b )
{
    return a < b ? a : b;
}
```

```
int min( int a, int b )
{
    return a < b ? a : b;
}
```

```
long min( long a, long b )
{
    return a < b ? a : b;
}
```

Шаблоны: PHP

```
<html>
  <head> <title> Тестируем PHP </title> </head>
  <body>
    <?php
      echo '<h1>Hello, world!</h1>';
    ?>
    <br />
    <?php
      $colors = array("red", "green", "blue", "yellow");

      foreach ($colors as $value) {
        echo "* $value <br />\n";
      }
    ?>
  </body>
</html>
```

Шаблоны: PHP

```
<body>

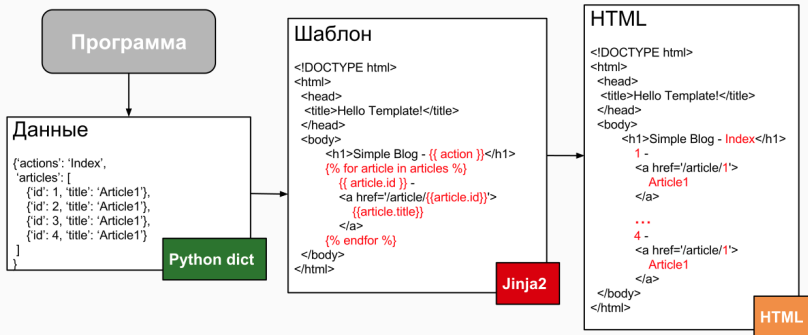
<h1>Hello, world!</h1>
<br />

* red <br />
* green <br />
* blue <br />
* yellow <br />

</body>
```

Jinja2 — самый популярный шаблонизатор в языке программирования Python. Автор Armin Ronacher из команды <http://www.pocoo.org/>, не раз приезжал на конференции в Екатеринбург с докладами о своих продуктах.

Шаблоны: Jinja2



Шаблоны: Jinja2 - {{ выражения }}

```
from jinja2 import Template

template = Template('Hello {{ name }}!')
print(template.render(name='Вася'))
```

Hello Вася!

Шаблоны: Jinja2 - {# комментарии #}

```
{# Это кусок кода
    который стал временно не нужен,
    но удалять жалко
    {% for user in users %}
        ...
    {% endfor %}
#}
```

Шаблоны: Jinja2 - {% операторы %}

```
from jinja2 import Template
text = '{% for item in range(5) %}Hello {{ name }}! {%
template = Template(text)
print(template.render(name='Вася'))
```

Hello Вася! Hello Вася! Hello Вася! Hello Вася! Hello Вася!

Шаблоны: Jinja2 - модули

```
from jinja2 import Template
template = Template(
    "{% set a, b, c = 'foo', фyy'', 'föö' %}"
)
m = template.module
print(m.a)
print(m.b)
print(m.c)
```

foo

фyy

föö

Шаблоны: Jinja2 - чтение из файла

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;"
  </head>
  <body>
    {% for item in range(5) %}
      Hello {{ name }}!
    {% endfor %}
  </body>
</html>
```

Шаблоны: Jinja2 - чтение из файла

```
from jinja2 import Template

html = open('foopkg/templates/0.hello.html').read()
template = Template(html)
print(template.render(name='Петя'))
```

Шаблоны: Jinja2 - чтение из файла

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;"
  </head>
  <body>
```

Hello Петя!

Hello Петя!

Hello Петя!

Hello Петя!

Hello Петя!

Шаблоны: Jinja2 - {% extends "наследование" %}

```
<head>
    {% block head %}
        <link rel="stylesheet" href="style.css" />
        <title>{% block title %}
            {% endblock %} — My Webpage</title>
        <meta charset='utf-8'>
    {% endblock %}
</head>
<body>
    <div id="content">{% block content %}
        {% endblock %}</div>
    <div id="footer">
        {% block footer %}
            &copy; Copyright 2008 by
            <a href="http://domain.invalid/">you</a>.
        {% endblock %}
    </div>
```

Шаблоны: Jinja2 - {% extends "наследование" %}

```
{% 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 {{ name }} to my awesome homepage.
    </p>
{% endblock %}
```


Шаблоны: Jinja2 - {% extends "наследование" %}

```
from jinja2 import Environment, FileSystemLoader

env = Environment(loader=FileSystemLoader('.'))
template = env.get_template('index.html')
print(template.render(name='Петя'))
```

Шаблоны: Jinja2 - {% extends "наследование" %}

```
<!DOCTYPE html>
<html lang="en">
<head>

    <link rel="stylesheet" href="style.css" />
    <title>Index — My Webpage</title>
    <meta charset='utf-8'>

    <style type="text/css">
        .important { color: #336699; }
    </style>

</head>
```

Шаблоны: Jinja2 - {% extends "наследование" %}

```
<body>
  <div id="content">
    <h1>Index</h1>
    <p class="important">
      Welcome Петя to my awesome homepage.
    </p>
  </div>
  <div id="footer">

    &copy; Copyright 2008 by <a href="http://domain">

  </div>
</body>
</html>
```

Пагинация

Пагинация — разделение контента на страницы.

OpenNET

www.opennet.ru

Поиск (теги):

[НОВОСТИ \(+\)](#) [КОНТЕНТ](#) [WIKI](#)

Блок пагинации на форуме opennet.ru


[форумы](#) [помощь](#) [поиск](#) [регистрация](#) [майллист](#)


[Создать новую тему](#)


☐ Раскрыть нити
⇒ Пометить прочитанным


1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 | 9 | 10 | [Архив](#) | [Избранное](#) | [Мое](#) | [Новое](#) | [≡](#) | [≡](#) | [Г](#)

Форум WEB технологии

 **Как CGI-скрипту указать имя возвращаемого архива** (web-разработка, CGI) [NEW](#) [\[<<\]](#) [\[≡\]](#) [\[>>\]](#)
Здравствуйте, Кратко вопрос звучит так: может ли CGI-скрипт вернуть вместе с данными, имя
[...](#)

 **JSON** (Apache, http-серверы) [NEW](#) [\[<<\]](#) [\[≡\]](#) [\[>>\]](#)
Чё им надо? [code] 4. Set up a service so that as soon as the file /var/log/sizes.log changes, the m

 **WEB сервер своими руками (CGI) !** (web-разработка, CGI) [NEW](#) [\[<<\]](#) [\[≡\]](#) [\[>>\]](#)
Други, Помогите пожалста. Вот такая задачка: Есть железка PLC – На ней нет какой либо ОС
возможность работать с [...](#)

 **построение графиков по данным из БД под СУБД MS SQL (PHP)** [NEW](#) [\[<<\]](#) [\[≡\]](#) [\[>>\]](#)
Всех приветствую! Вообще есть данные числовые, хранятся в СУБД MS SQL. Мне нужно в лок

Samsung Galaxy A3 (2017)
SM-A320F

от 14 950 ₽

до -27%

4.0 114 отзыва

Samsung Galaxy A7 (2017)
SM-A720F

от 21 790 ₽

4.0 44 отзыва

Samsung Galaxy Note 8
64GB

от 57 490 ₽

4.5 11 отзывов

Samsung Galaxy J5 (2017)
16GB

от 14 550 ₽

4.5 31 отзыв

Samsung Galaxy J1 (2016)
SM-J120F/DS

от 5 690 ₽

4.0 303 отзыва

Samsung Galaxy J3 (2017)

от 10 450 ₽

4.0 11 отзывов

Показать еще

1 2 3

Вперед >

Показывать по 12 ▾

Samsung

☐ Galax

☐ Galax

☐ Galax

☐ Galax

☐ Galax

☐ Galax

☐ Galax

☐ Andr

☐ iOS

▾ Диаг

▾ Разр

☐ 4G L

☐ 2 SIM

▾ Емко
акку

Пагинация: Python

pip install **paginate**

<https://github.com/Pylons/paginate>

Пагинация: Python

```
from paginate import Page
page = values.get('page', 1)
paged_articles = Page(
    ARTICLES,
    page=page,
    items_per_page=8,
)
```

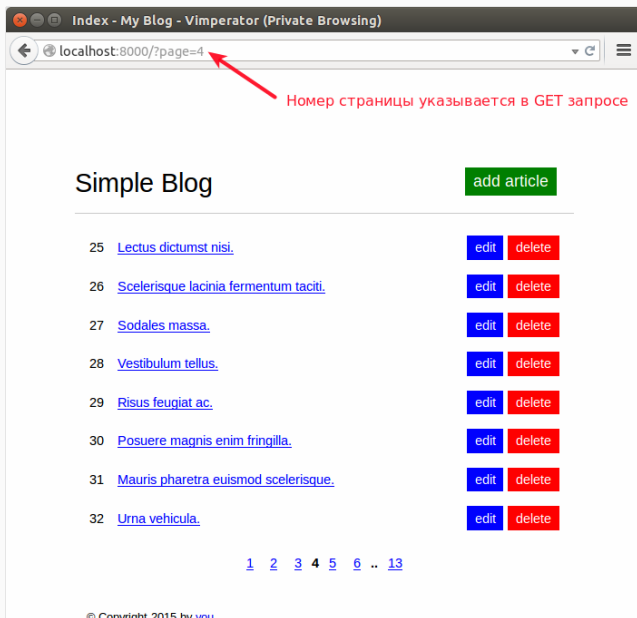
Пагинация: Python

```
{% for article in articles %}
  <div class="blog-list__item">
    <div class="blog-list__item-id">
      {{ article.id }}</div>
      <a href="/article/{{ article.id }}"
        class="blog-list__item-link">{{ article.title }}</a>
      <div class="blog-list__item-action">
        <a href="/article/{{ article.id }}/edit"
          class="blog-list__item-edit">edit</a>
        <a href="/article/{{ article.id }}/delete"
          onclick="return confirm_delete();"
          class="blog-list__item-delete">delete</a>
      </div>
    </div>
  </div>
{% endfor %}
```

Пагинация: Python

```
<div class="paginator">
    {{ articles.pager(url="?page=$page") }}
</div>
```

Пагинация: Python



WebOb — библиотека сериализующая HTTP запрос (текст) в объект и наоборот объект в HTTP ответ (текст).