Уральский федеральный университет

Веб без фреймворков

Основы Веб-программирования

Кафедра Интеллектуальных Информационных Технологий, ИнФО, УрФУ

Ресурсы

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
pep-333
pep-3333

Общие принципы

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

Веб-сервер

Веб сервер

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**

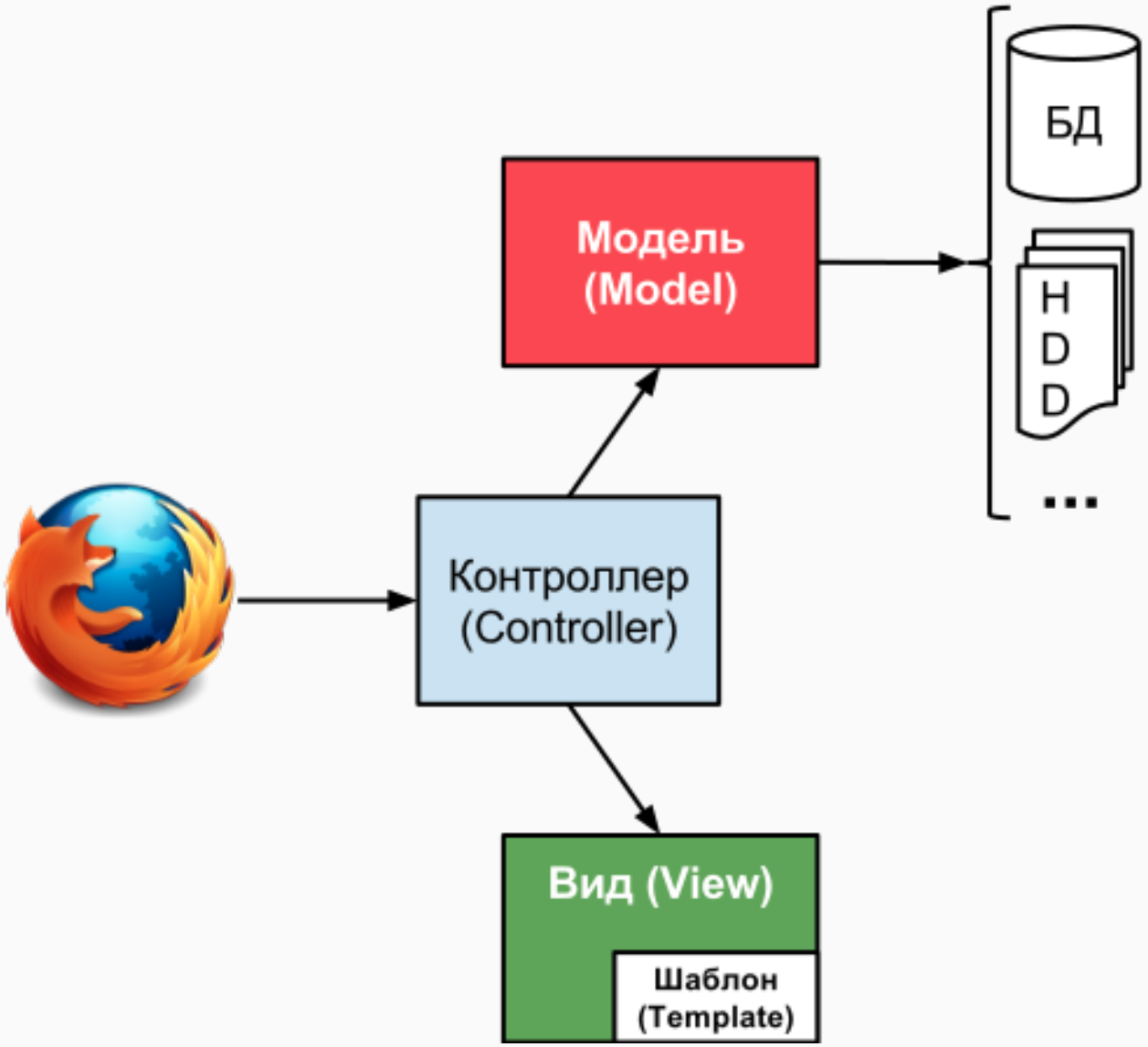
Разделение кода на части

Разделение кода: **MVC**

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**

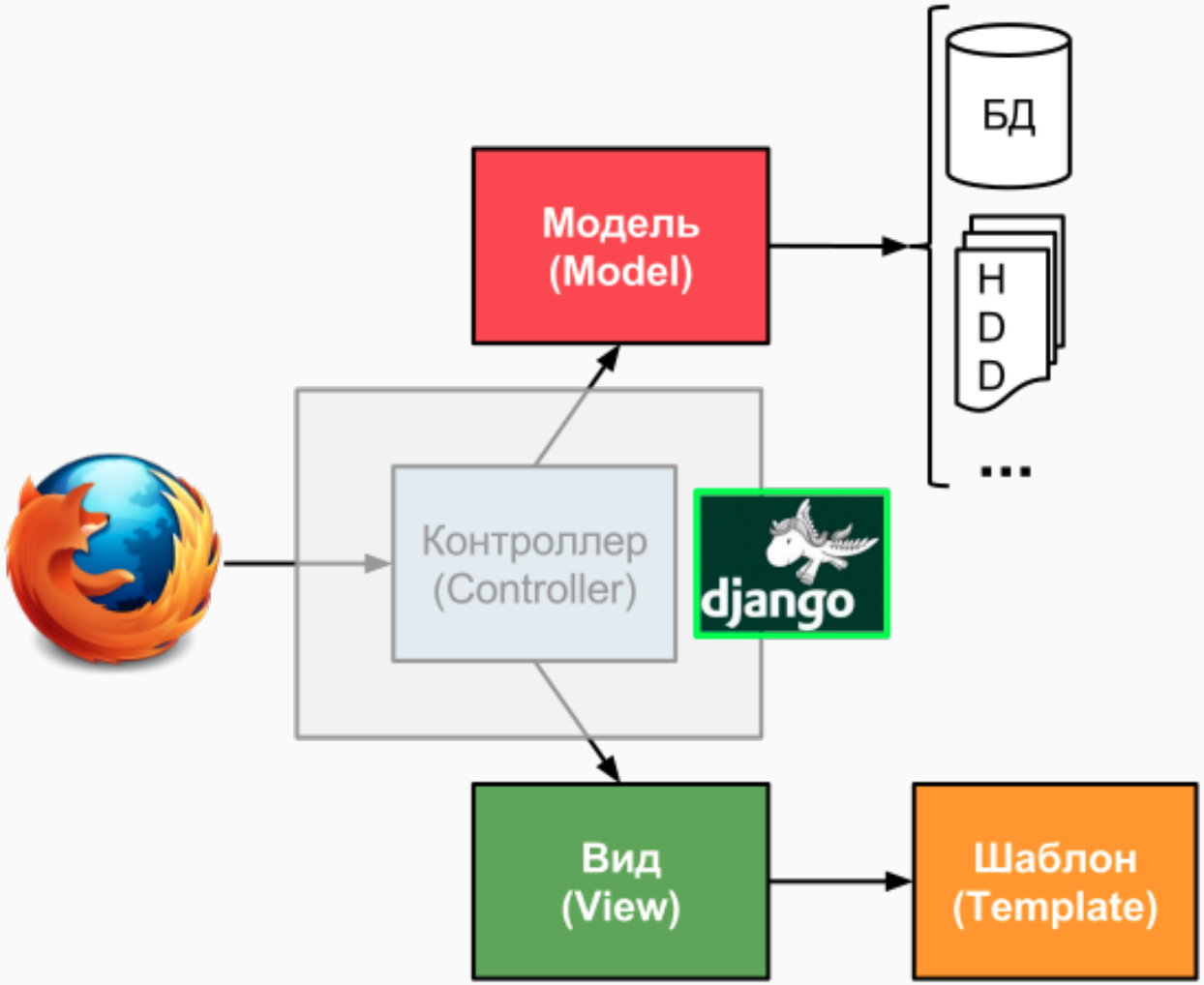
Фреймворк **Django** ввел новую терминологию **MTV**

- **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**

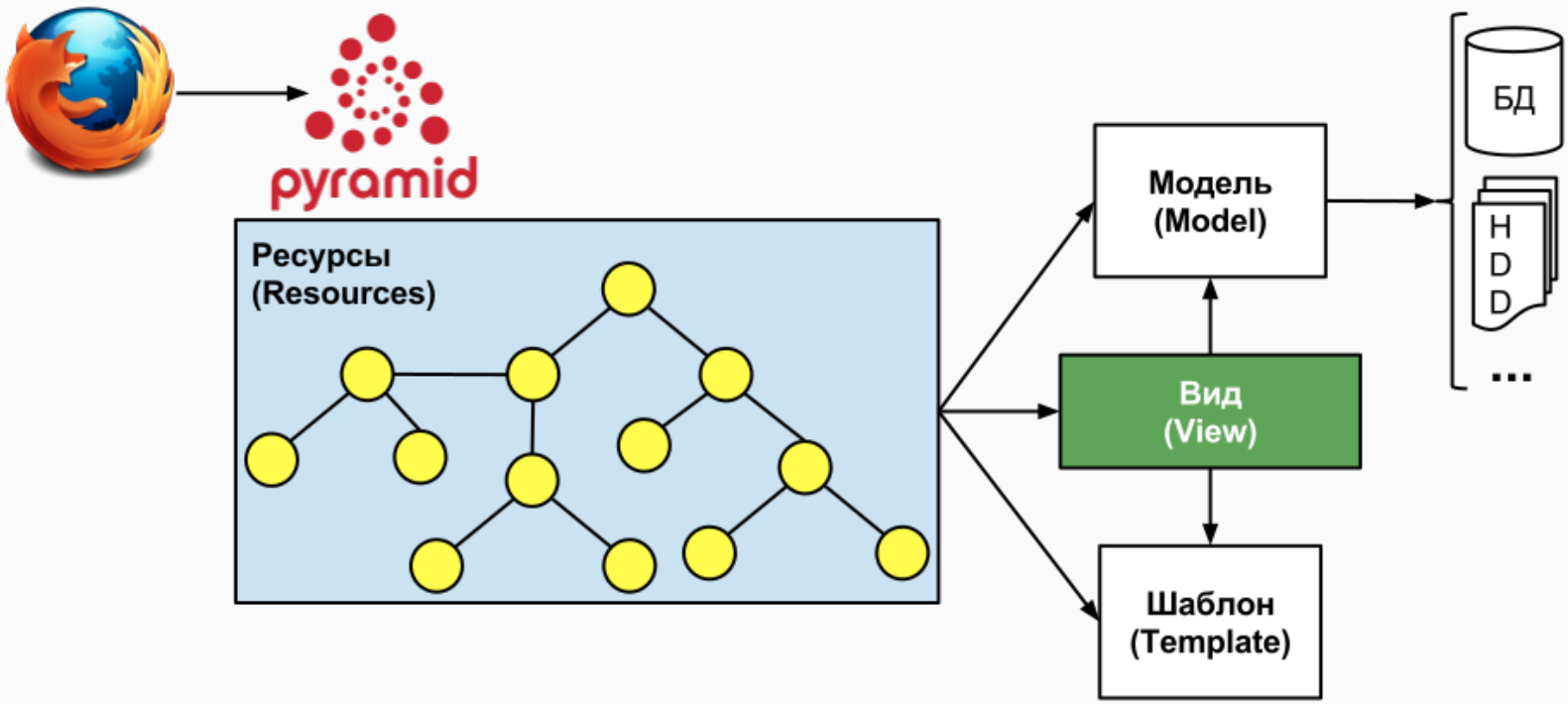
Это дает неоспоримую гибкость таким приложениям, оставляя ответственность по структуре ПО за разработчиком.

Разделение кода: **RV**

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. ”

Разделение кода: **RV**

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

Разделение кода: **RV**

“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. ”

Разделение кода: **RV**

Pyramid only

Маршруты

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 отличии 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})/$', views.month_
 url(r'^articles/([0-9]{4})/([0-9]{2})/([0-9]+)/$', vie
]
```

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_
    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=upda
dispatch.add('/{id:digits}/delete', GET=delete)
```

Маршруты: Сравнение

Регулярные выражения	Сопоставление с образом
/	/
/article/add	/article/add
`^/article/(?P<id>d+)/$`	`/article/{id:digits}`
`^/article/(?P<id>d+)/edit$`	`/article/{id:digits}/edit`
`^/article/(?P<id>d+)/delete$`	`/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: Определение

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>
```

Шаблоны: PHP

```
<body>

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

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

</body>
```

Шаблоны: Jinja2

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