Python для Web

WSGI, Django, TurboGears, Pylons

Юрий Юревич
http://gorod-omsk.ru/blog/pythy/

yury@yurevich.ru

Конференция по Ruby и Python.
Омск, 10 февраля 2007

Why Python

- Компактный (129 встроенных языковых конструкций, включая исключения, copyright и credits)
- Модульный
- Легкий для чтения
- Переносимый (Windows, Mac OS, *nix, Nokia 60)
- Обширная стандартная библиотека
- Большое сообщество

Что выбрать?

- Стандартная библиотека?
- ...
- Фреймворк? Какой?
  - Django
  - TurboGears
  - Pylons
- Zope?

Стандартная библиотека

`cgi`, `cookielib`, `urlparse`

- Плюсы
  - Всегда рядом
  - Поддержка WSGI (только в Python 2.5)
- Минусы
  - Слабое покрытие рутинных задач:
    - Валидация форм
    - Отсутствие шаблонного движка

Что дальше?

web.py, PSP, Quixote?

WSGI!

WSGI

Что это такое?

- Стандарт — *This document specifies a proposed standard interface between web servers and Python web applications or frameworks, to promote web application portability across a variety of web servers*, PEP 333
- Аналог Java Servlets

А подробнее?

- Приложение должно быть исполняемым (callable)
- Приложение принимает два параметра: переменные окружения и запрос
- Приложение выполянет запрос, указывая код возврата и заголовки ответа
- Приложение возвращает итератор с телом ответа

Пример, пожалуйста

Теория

- Приложение должно быть исполняемым [1]
- Приложение принимает два параметра: переменные окружения и запрос [1]
- Приложение выполянет запрос, указывая код возврата [2] и заголовки ответа [3]
- Приложение возвращает итератор с телом ответа [4]

Практика

```
1 | def wsgiapp(environ, start_response):
2 |     start_response('200 OK',
3 |         [('Content-type', 'text/plain')])
4 |     return ['Hello, World']
```

Преимущества WSGI

- Стандартизовано
- Просто и гибко
- Middleware
  - Некий аналог декоратора
  - Можно выстраивать цепочки middleware
- Широкая поддержка:
  - конечные веб-приложения:
    - MoinMoin (вики)
    - Trac (трекер, вики)
    - Roundap (трекер)
  - Инструменты для создания веб-приложений:
    - Django
    - TurboGears
    - Pylons
    - Zope 3

Пример WSGI middleware

Аутентификация/авторизация средствами barrel

```
1  | from barrel import cooper
2  | users = [('joe', 'foo'), ('sam', 'eggs')]
3  | roles = {'joe': ('admin', 'user'),
4  |          'sam': ('user', )}
5  | @cooper.formauth(users=users)
6  | @cooper.rolesauth(
7  |     roles_dict=roles, allowed_roles=('user'))
8  | def wsgiapp(environ, start_response):
9  |     start_response('200 OK',
10 |         [('Content-type', 'text/plain')])
11 |     user = environ['barrel.form.username']
12 |     user_roles = environ['barrel.roles']
13 |     return ['Hello, %s!' % user,
14 |             'Your roles are %s' % user_roles]
```

WSGI инструменты

- Управление URLами
  - Routes — Rails-like
  - Selector — Rails-like, regexp, неявно
  - Colubrid — Rails-like, regexp, по списку, неявно и др.
- Аутентификация, авторизация, сессии
  - AuthKit — http auth, form auth, openid
  - Barrel — http auth, form auth
  - WSGIAuth — http auth, IP, form auth, openid
  - Beaker — сессии, хранение: в файлах, БД, memcached
  - Dbstore — сессии, хранение в БД
  - WSGIState — сессии, управление кэшем, хранение: в файлах, БД, memcached
- Интересные
  - WSGIIntercept — тестирование WSGI-приложений
  - WSGISerialize — сериализация (JSON, XML-RPC, YaML, pickle)

Что такое фреймворк?

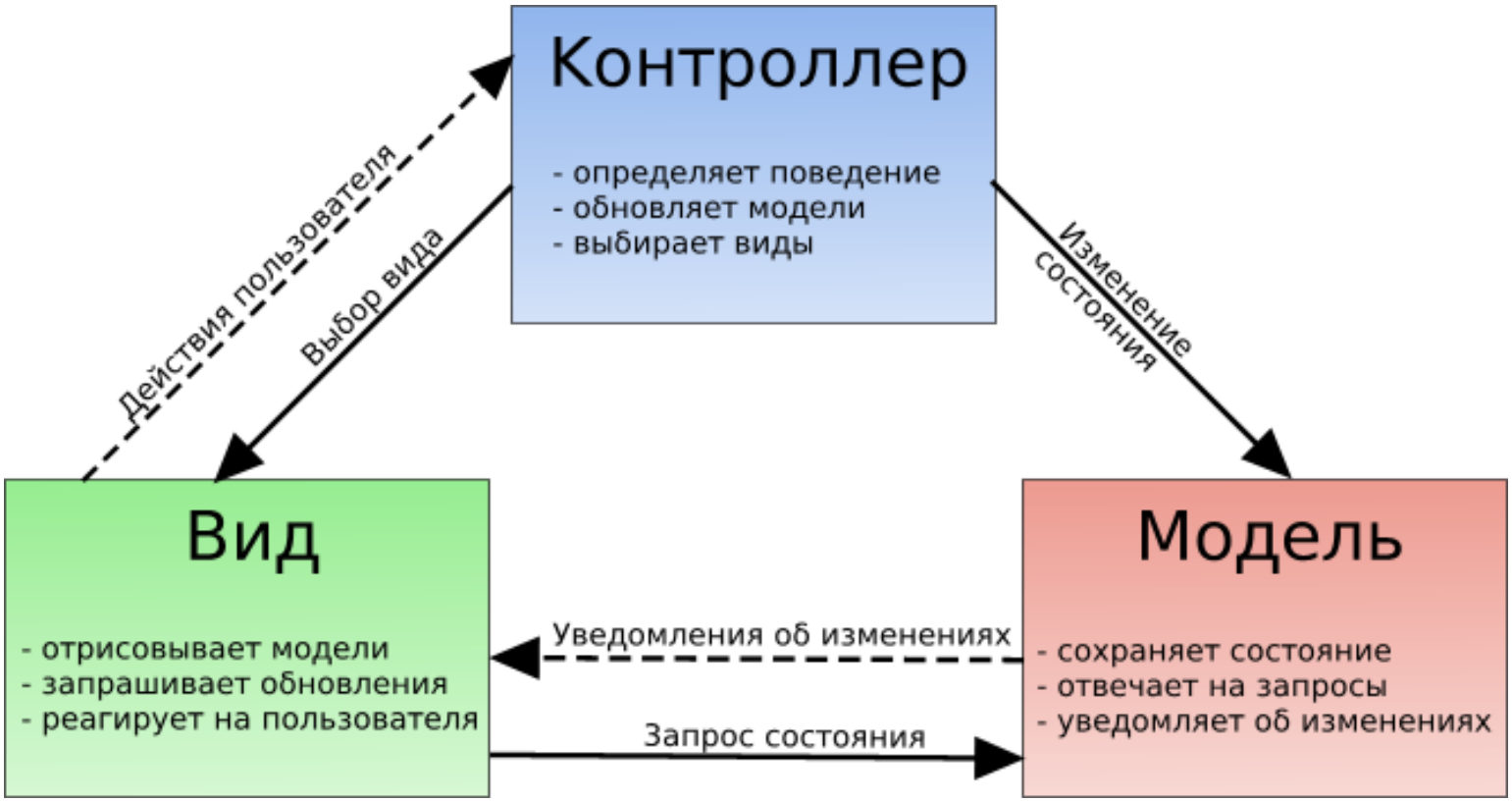
Фреймворк

- "Библиотека наоборот"
- Компоненты "на каждый день":
  - ORM
  - Шаблоны
  - Управление URLами
  - Валидация форм

Что такое MVC?

Паттерн «Модель-вид-контроллер» (MVC)

- Модель - «что» (DB/ORM)
- Вид - «как выглядит» (шаблоны)
- Контроллер - «как получается» (логика)



Django

«Веб-фреймворк для перфекционистов с дедлайнами»

- ORM - собственный
- Шаблоны - собственные
- Управление URLами - собственное, на регулярных выражениях
- Валидация форм - собственная

Django: преимущества и недостатки

Преимущества

- Всё вместе - «don't worry — be happy»
- Хорошая документация
- Готовые решения в `django.contrib`:
  - `django.contrib.auth` — система аутентификации/авторизации
  - `django.contrib.syndication` — RSS/Atom фиды
  - `django.contrib.comments` — Комментарии
  - `django.contrib.admin` — Авто-админка

Недостатки

- Монолитный
- Не очень удобные формы (встречаем newforms?)
- ORM (ждем SQLAlchemy?)
- Многовато «магии»

TurboGears

«Мегафреймворк для быстрой веб разработки, то что вы искали»

- ORM - SQLObject (по умолчанию), SQLAlchemy
- Шаблоны - Kid (по умолчанию), Cheetah, Genshi, Django templates, etc
- Управление URLами - CherryPy, по именам методов
- Валидация форм - FormEncode

TurboGears: преимущества и недостатки

Преимущества

- Компонентая архитектура
- Неплохая документация
- Качественные вспомогательные инструменты (TG ToolBox):
  - ModelDesigner — дизайнер моделей
  - CatWalk — аналог админки Django
  - WidgetBrowser — обзор виджетов (мини-приложения)

Недостатки

- Не очень удобное и прозрачное управление URLами
- Не продуманная архитектура

Pylons

«Легкий веб-фреймворк, ориентированный на гибкость и скорость разработки»

- ORM - SQLObject, SQLAlchemy
- Шаблоны - Myghty (по умолчанию), Kid, Cheetah, Genshi, etc
- Управление URLами - Routes, порт с RoR
- Валидация форм - FormEncode

Pylons: преимущества и недостатки

Преимущества

- Компонентая архитектура, основанная на WSGI
- Тщательно подобранные компоненты (Paste, WebHelpers, AuthKit)
- Легкая смена компонент

Недостатки

- Не полная документация

Фреймворки: итоги

- Django — «don't worry — be happy»
  - Ниша: CMS, контент-сайт
  - Диагноз: быстрый старт
  - Проблемы: монолитность
- TurboGears — «запчасти есть, винтиков маловато»
  - Ниша: AJAX-powered сайт, контент-сайт
  - Диагноз: любителям «покрутить винтики»
  - Проблемы: связность компонент
- Pylons — «всё правильно сделал»
  - Диагноз: быстро, удобно
  - Проблемы: документация, порог вхождения

Zope 3

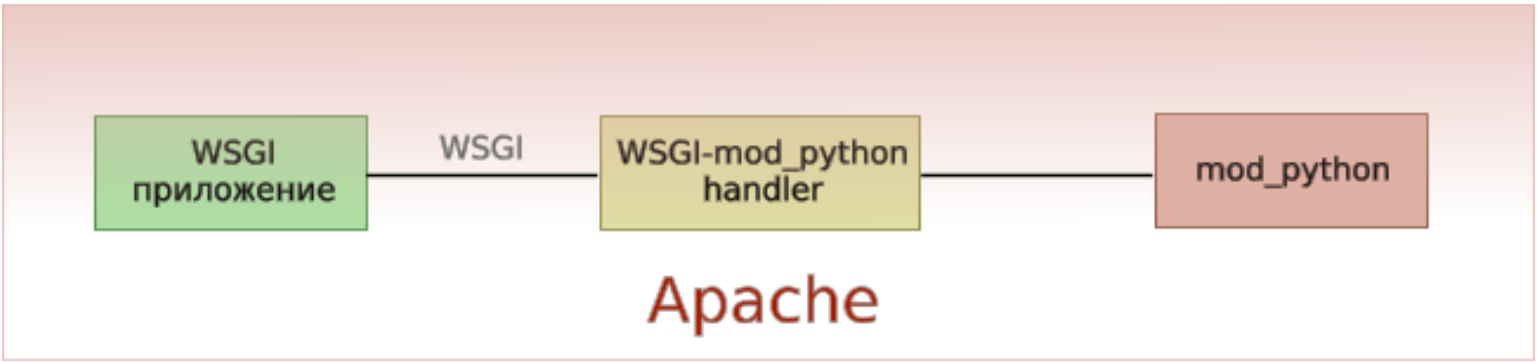
OO сервер приложений

- Объектная БД — ZODB
- Объектное управление URLами (ecquisition)
- Шаблоны ZPT (TAL, template attributes language)

Развертывание WSGI приложений

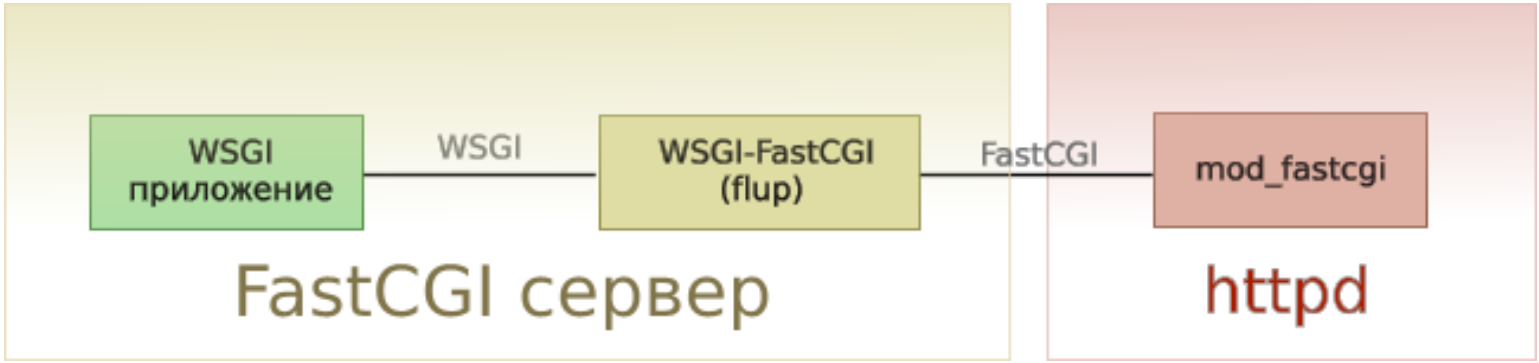
Подробнее см. в блоге http://gorod-omsk.ru/blog/pythy/2007/02/05/deploying-wsgi-app/

Apache+mod_python



- Скорость — 14x
- Удобство — 4
- В плюс — простота
- В минус — объем памяти
- **Итог — «дешево и сердито»**

Apache/lighttpd/nginx+FastCGI



- Скорость — 5x (Apache) – 12x (lighttpd/nginx)
- Удобство — 3
- В плюс — отношение скорость/память
- В минус — сложность настройки
- **Итог — «оптимально»**

Twisted Web2

- Скорость — 7x
- Удобство — 5
- В плюс — «всё включено»
- В минус — скорость отдачи статики
- **Итог — «для внутреннего пользования»**

CherryPy WSGI server

- Скорость — 24x
- Удобство — 4
- В плюс — скорость
- В минус — скорость отдачи статики, сложность настройки
- **Итог — «потенциально интересно»**

Заключение

Почему ваш следующий проект должен быть на Python/WSGI

- Быстрый старт
- Легкая поддержка
- Стандартизированная платформа
- Качественные инструменты на любой вкус
- Гибкие схемы развертывания

Спасибо

Спасибо за внимание

Вопросы?