The Swiss-Army Knife for Python Web Developers

Armin Ronacher — http://lucumr.pocoo.org/

About Me

About Me

- Name: Armin Ronacher
- Werkzeug, Jinja, Pygments, ubuntuusers.de
- Python since 2005
- WSGI warrior since the very beginning (well, not quite)

Why Python?

Why Python?

- agile
- active community
- countless modules
- powerful introspection functionality
- **WSGI**

WSGI

WSGI

- **W**eb **S**erver **G**ateway **I**nterface
- lowlevel interface between application and server
- allows to reuse code between applications
- CGI / FastCGI / SCGI / AJP / mod_python / mod_wsgi / twisted / standalone
- simple and fast

Hello World

Hello World

```
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [
        '<!DOCTYPE HTML>\n<title>Hello World</title>\n'
        '<h1>Hello World!</h1>'
    ]
```

Hello World

Request

```
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [
        '<!DOCTYPE HTML>\n<title>Hello World</title>\n'
        '<h1>Hello World!</h1>'
    ]
```

Hello World

Request

Response #1

```
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [
        '<!DOCTYPE HTML>\n<title>Hello World</title>\n'
        '<h1>Hello World!</h1>'
    ]
```

Response #2

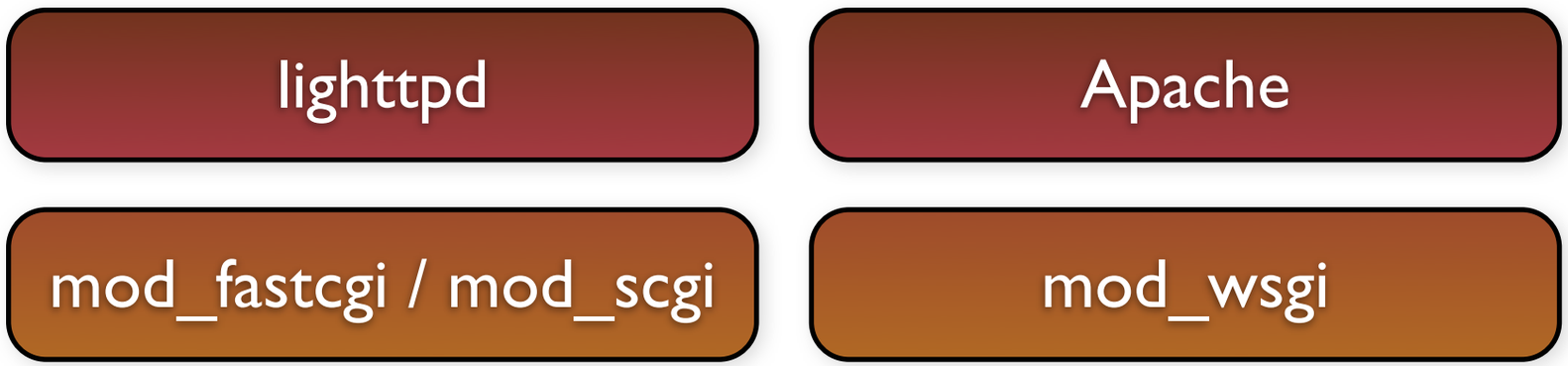
Deployment

Deployment

lighttpd

Apache

Deployment



Deployment

lighttpd

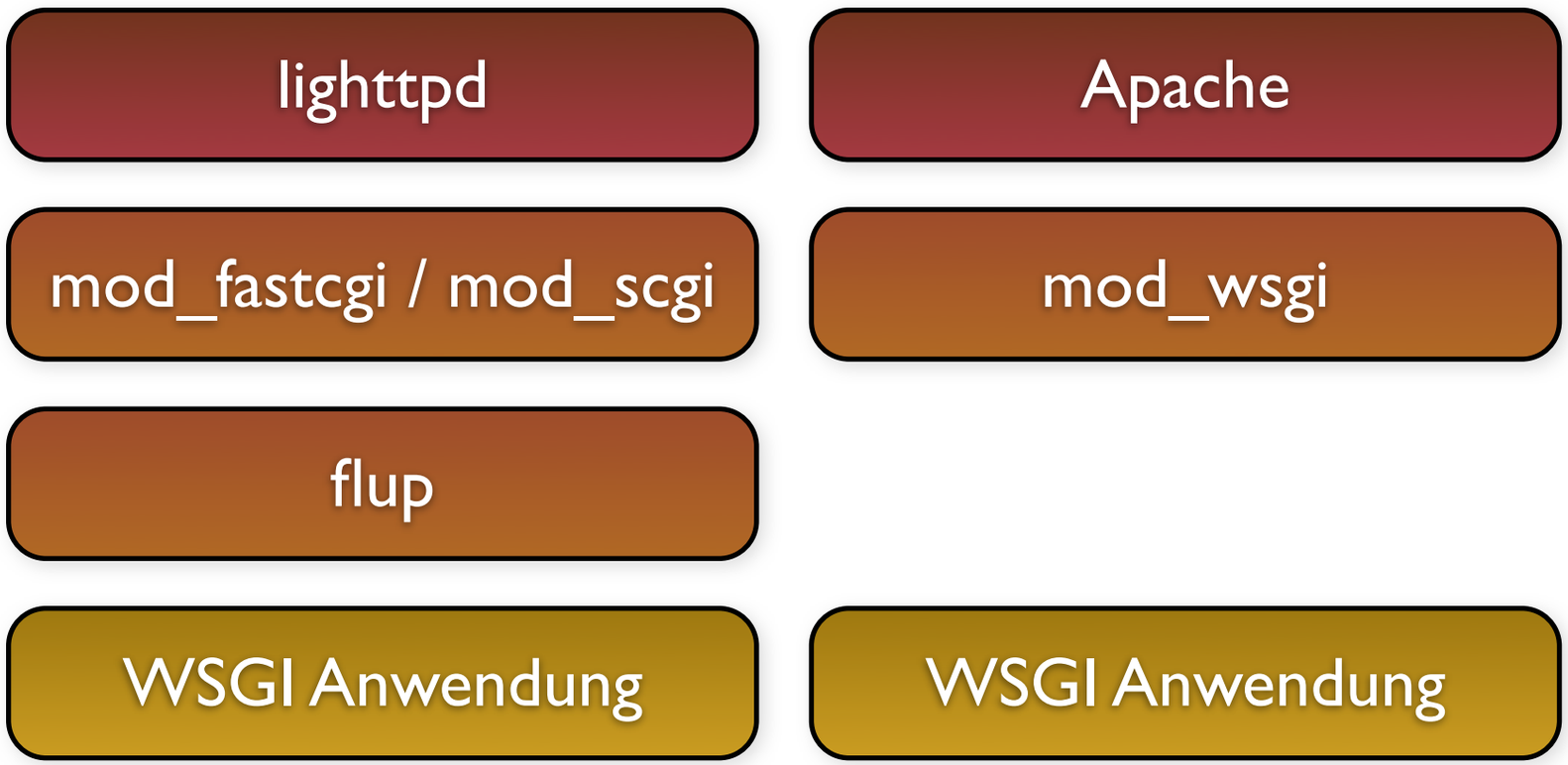
Apache

mod_fastcgi / mod_scgi

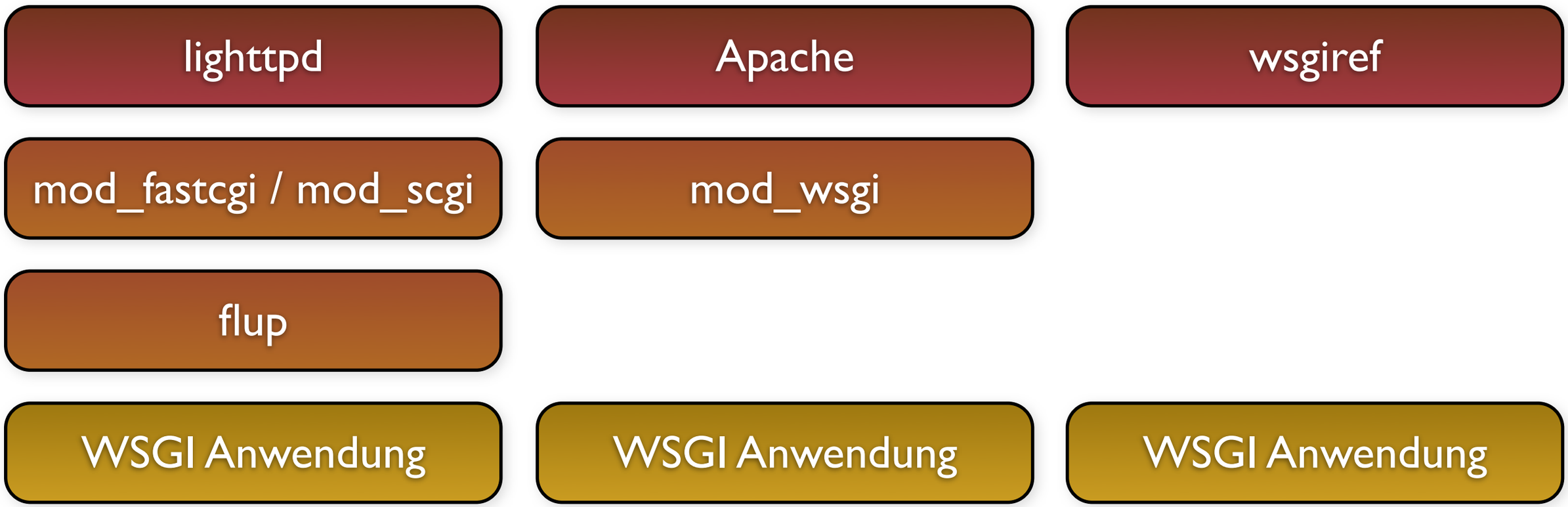
mod_wsgi

flup

Deployment


lighttpd
Apache
mod_fastcgi / mod_scgi
mod_wsgi
flup
WSGI Anwendung
WSGI Anwendung

Deployment


lighttpd
Apache
wsgiref
mod_fastcgi / mod_scgi
mod_wsgi
flup
WSGI Anwendung
WSGI Anwendung
WSGI Anwendung

Middlewares

Middlewares

- middlewares work between server and application

Middlewares

- middlewares work between server and application
- can manipulate incoming and outgoing data

Middlewares

- middlewares work between server and application
- can manipulate incoming and outgoing data
- useful to …
  - … log errors
  - … fix broken server data
  - … combine multiple applications

But....

But....

...an application shouldn’t depend on an middleware

But....

...an application shouldn’t depend on an middleware

http://dirtsimple.org/2007/02/wsgi-middleware-considered-harmful.html

But....

...an application shouldn’t depend on an middleware

http://dirtsimple.org/2007/02/wsgi-middleware-considered-harmful.html

Setup

Setup

- central „runner” file:

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- imports application and middlewares

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- imports application and middlewares
- combines it and creates and `application` object or calls the gateway

Setup

- central „runner” file:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- imports application and middlewares
- combines it and creates and `application` object or calls the gateway

```
from yourapplication import Application
from yourmiddleware import YourMiddleware
from vendormiddleware import VendorMiddleware

application = Application(configuration=here)
application = YourMiddleware(application, configuration=here)
application = VendorMiddleware(application)
```

Summary

Summary

- application: callable object

Summary

- application: callable object
  - `environ` … incoming data

Summary

- application: callable object
  - `environ` … incoming data
  - `start_response` … starts the response

Summary

- application: callable object
  - `environ` … incoming data
  - `start_response` … starts the response
  - `app_iter` … an iterator, each iteration sends data to the client

Summary

- application: callable object
  - `environ` … incoming data
  - `start_response` … starts the response
  - `app_iter` … an iterator, each iteration sends data to the client
- middleware: between application and gateway

Summary

- application: callable object
  - `environ` … incoming data
  - `start_response` … starts the response
  - `app_iter` … an iterator, each iteration sends data to the client
- middleware: between application and gateway
- gateway: translates WSGI to CGI etc.

Werkzeug

Werkzeug

- unicode handling

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing
- URL dispatching

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing
- URL dispatching
- HTTP parsing

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing
- URL dispatching
- HTTP parsing
- development server

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing
- URL dispatching
- HTTP parsing
- development server
- autoreloader

Werkzeug

- unicode handling
- form data / file uploads / url parameter parsing
- URL dispatching
- HTTP parsing
- development server
- autoreloader
- countless small helpers

What is it not?

What is it not?

- ORM

What is it not?

- ORM
- template engine

What is it not?

- ORM
- template engine
- form-validation

What is it not?

- ORM
- template engine
- form-validation
- i18n / l10n

What is it not?

- ORM
- template engine
- form-validation
- i18n / l10n
- component architecture

What is it not?

- ORM
- template engine
- form-validation
- i18n / l10n
- component architecture
- a framework

Why not?

Why not?

- such things exist already

Why not?

- such things exist already
- you can combine them

Why not?

- such things exist already
- you can combine them
- everybody wants something else

Why not?

- such things exist already
- you can combine them
- everybody wants something else
- cherry picking!

What does it look like?

What does it look like?

```
>>> from werkzeug import Request, create_environ
>>> environ = create_environ('/index.html?foo=bar&foo=baz&blah=42')
>>> request = Request(environ)
>>> request.args['foo']
u'bar'
>>> request.args.getlist('foo')
[u'bar', u'baz']
>>> request.args.get('blah', type=int)
42
>>> request.path
u'/index.html'
>>> request.method
'GET'
```

DumpIt!

a pastebin in 15 minutes

What do we use?

What do we use?

- Werkzeug

What do we use?

- Werkzeug
- Jinja

What do we use?

- Werkzeug
- Jinja
- sqlite3

What do we use?

- Werkzeug
- Jinja
- sqlite3
- Pygments

What do we use?

- Werkzeug WSGI
- Jinja
- sqlite3
- Pygments

What do we use?

- Werkzeug — WSGI
- Jinja — Templates
- sqlite3
- Pygments

What do we use?

- Werkzeug — WSGI
- Jinja — Templates
- sqlite3 — Database
- Pygments

What do we use?

- Werkzeug — WSGI
- Jinja — Templates
- sqlite3 — Database
- Pygments — Code Highlighting

What do we use?

- Werkzeug — WSGI
- Jinja — Templates
- sqlite3 — Database
- Pygments — Code Highlighting

```
easy_install Werkzeug
easy_install Jinja
easy_install Pygments
```

#0: Database

#0: Database

schema.sql

```
CREATE TABLE pastes (
    id INTEGER NOT NULL,
    code TEXT,
    lang VARCHAR(40),
    PRIMARY KEY (id)
);
```

#1: Imports

#1: Imports

```
import sqlite3
from os import path
from werkzeug import Request, Response, redirect
from werkzeug.exceptions import HTTPException, NotFound
from werkzeug.routing import Map, Rule
from jinja import Environment, FileSystemLoader
from pygments import highlight
from pygments.lexers import get_lexer_by_name, TextLexer
from pygments.formatters import HtmlFormatter
```

#2: Configuration

#2: Configuration

```
DATABASE = '/path/to/dumpit.db'
PYGMENTS_STYLE = 'pastie'
LANGUAGES = [
    ('text',            'No Highlighting'),
    ('python',          'Python'),
    ('c',               'C')
]
TEMPLATES = path.join(path.dirname(__file__), 'templates')

jinja_env = Environment(loader=FileSystemLoader(TEMPLATES))
pygments_formatter = HtmlFormatter(style=PYGMENTS_STYLE)

def render_template(template_name, **context):
    template = jinja_env.get_template(template_name)
    return template.render(context)
```

#3: Dispatching

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])

def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])

def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])

def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

http://localhost:5000/
http://localhost:5000/42
http://localhost:5000/42/raw
http://localhost:5000/pygments.css

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])
```

```
def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])

def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#3: Dispatching

```
url_map = Map([
    Rule('/', endpoint='new_paste'),
    Rule('/<int:id>', endpoint='show_paste'),
    Rule('/<int:id>/raw', endpoint='download_paste'),
    Rule('/pygments.css', endpoint='pygments_style')
])


def application(environ, start_response):
    request = Request(environ)
    request.db = sqlite3.connect(DATABASE)
    url_adapter = url_map.bind_to_environ(environ)
    try:
        endpoint, values = url_adapter.match()
        response = globals()[endpoint](request, **values)
        if isinstance(response, basestring):
            response = Response(response, mimetype='text/html')
    except HTTPException, error:
        response = error
    return response(environ, start_response)
```

#4: „Views“

#4: „Views“

```
def new_paste(request):
    if request.method == 'POST':
        code = request.form.get('code')
        lang = request.form.get('lang')
        if code and lang:
            paste = Paste(lang, code)
            paste.save(request.db)
            return redirect(str(paste.id))
    return render_template('new_paste.html', languages=LANGUAGES)
```

#4: „Views“

```
def new_paste(request):
    if request.method == 'POST':
        code = request.form.get('code')
        lang = request.form.get('lang')
        if code and lang:
            paste =
            paste.s
            return
    return render_t
```

```
def show_paste(request, id):
    paste = Paste.get(request.db, id)
    if paste is None:
        raise NotFound()
    return render_template('show_paste.html', paste=paste)
```

#4: „Views“

```python
def new_paste(request):
    if request.method == 'POST':
        code = request.form.get('code')
        lang = request.form.get('lang')
        if code and lang:
            paste =
            paste.s
            return
    return render_t
```

```python
def show_paste(request, id):
    paste = Paste.get(request.db, id)
    if paste is None:
        raise NotFound()
    return render_template('show_paste.html', paste=paste)
```

```python
def download_paste(request, id):
    paste = Paste.get(request.db, id)
    if paste is None:
        raise NotFound()
    return Response(paste.code)
```

#4: „Views“

```
def new_paste(request):
    if request.method == 'POST':
        code = request.form.get('code')
        lang = request.form.get('lang')
        if code and lang:
            paste =
            paste.s
            return
    return render_t
```

```
def show_paste(request, id):
    paste = Paste.get(request.db, id)
    if paste is None:
        raise NotFound()
    return render_template('show_paste.html', paste=paste)
```

```
def download_paste(request, id):
    paste = Paste.get(request.db, id)
    if paste is None:
        raise NotFound()
    return Response(paste.code)
```

```
def pygments_style(request):
    return Response(pygments_formatter.get_style_defs(),
                    mimetype='text/css')
```

#5: Model

#5: Model

```
class Paste(object):

    def __init__(self, lang, code, id=None):
        self.lang = lang
        self.code = code
        self.id = id

    @property
    def highlighted_code(self):
        try:
            lexer = get_lexer_by_name(self.lang)
        except ValueError:
            lexer = TextLexer
        return highlight(self.code, lexer, pygments_formatter)

    @classmethod
    def get(cls, con, id):
        cur = con.cursor()
        cur.execute('select lang, code, id from pastes where id = ?', [id])
        row = cur.fetchone()
```

#5: Model

```
    @classmethod
    def get(cls, con, id):
        cur = con.cursor()
        cur.execute('select lang, code, id from pastes where id = ?', [id])
        row = cur.fetchone()
        if row:
            return cls(*row)

    def save(self, con):
        cur = con.cursor()
        if self.id is None:
            cur.execute('insert into pastes (lang, code) values (?, ?)',
                        [self.lang, self.code])
            self.id = cur.lastrowid
        else:
            cur.execute('update pastes set lang = ?, code = ? where '
                        'id = ?', [self.lang, self.code, self.id])
        con.commit()
```

#6: Templates

#6: Templates

layout.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Dump It!</title>
    <link rel="stylesheet" href="/static/style.css" type="text/css">
    <link rel="stylesheet" href="/pygments.css" type="text/css">
  </head>
  <body>
    <div id="header">
      <h1>Dump It!</h1>
    </div>
    <div id="page">
      {% block body %}{% endblock %}
    </div>
  </body>
</html>
```

#6: Templates

layout.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Dump It!</title>
    <link
    <link
  </head>
  <body>
    <div
      <h1
    </div
    <div
      {%
    </div
  </body>
</html>
```

new_paste.html

```
{% extends "layout.html" %}
{% block body %}
  <h2>New Paste</h2>
  <form action="" method="post">
    <p><textarea name="code" rows="8" cols="50"></textarea></p>
    <p><select name="lang">
    {% for code, name in languages %}
      <option value="{{ code }}">{{ name }}</option>
    {% endfor %}
    </select><input type="submit" value="Paste"></p>
  </form>
{% endblock %}
```

#6: Templates

layout.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Dump It!</title>
    <link
    <link
  </head>
  <body>
    <div
      <h1
    </div
    <div
      {%
    </div
  </body>
</html>
```

new_paste.html

```
{% extends "layout.html" %}
{% block body %}
  <h2>New Paste</h2>
  <form action="" method="post">
    <p><textarea name="code" rows="8" cols="50"></textarea></p>
    <p><select name="lang">
    {% for code, name in languages %}
      <option value="{{ code }}">{{ name }}</op
    {% endfor %}
    </select><input
  </form>
{% endblock %}
```

show_paste.html

```
{% extends "layout.html" %}
{% block body %}
  <h2>Paste #{{ paste.id }}</h2>
  <div class="paste">
    {{ paste.highlighted_code }}
  </div>
{% endblock %}
```

Development Server

Development Server

```
if __name__ == '__main__':
    from werkzeug import run_simple, SharedDataMiddleware
    application = SharedDataMiddleware(application, {
        '/static': path.join(path.dirname(__file__), 'static')
    })
    run_simple('localhost', 4000, application)
```

Development Server

```
if __name__ == '__main__':
    from werkzeug import run_simple, SharedDataMiddleware
    application = SharedDataMiddleware(application, {
        '/static': path.join(path.dirname(__file__), 'static')
    })
    run_simple('localhost', 4000, application)
```

```
mitsuhiko@nausicaa:~/DumpIt$ sqlite3 /path/to/dumpit.db < schema.sql
mitsuhiko@nausicaa:~/DumpIt$ python dumpit.py runserver
 * Running on http://localhost:4000/
```

„Dump It!“ In Action

„Dump It!“ In Action

„Dump It!“ In Action

More Than One Way

More Than One Way

- Templates: XML / Text-based / Sandbox

More Than One Way

- Templates: XML / Text-based / Sandbox
- Daten: SQL / CouchDB / Filesystem

More Than One Way

- Templates: XML / Text-based / Sandbox
- Daten: SQL / CouchDB / Filesystem
- AJAX: JSON / XML / HTML Fragments

More Than One Way

- Templates: XML / Text-based / Sandbox
- Daten: SQL / CouchDB / Filesystem
- AJAX: JSON / XML / HTML Fragments
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameters

More Than One Way

- Templates: XML / Text-based / Sandbox
- Daten: SQL / CouchDB / Filesystem
- AJAX: JSON / XML / HTML Fragments
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameters
- Dispatching: Controller / View-Functions

More Than One Way

- Templates: XML / Text-based / Sandbox
- Daten: SQL / CouchDB / Filesystem
- AJAX: JSON / XML / HTML Fragments
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameters
- Dispatching: Controller / View-Functions
- Auth: Apache / LDAP / OpenID

http://werkzeug.pocoo.org/

http://lucumr.pocoo.org/talks/ltgraz08/