Das Schweizer Taschenmesser für Python-Webentwickler

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

Über Mich

Über Mich

- Name: Armin Ronacher
- Werkzeug, Jinja, Pygments, ubuntuusers.de
- Python seit 2005
- WSGI Krieger der Ersten Stunde (naja fast)

Warum Python?

Warum Python?

- agil
- aktive Community
- unzählige Module
- herausragende Introspection Funktionalität
- **WSGI**

WSGI

WSGI

- **W**eb **S**erver **G**ateway **I**nterface
- Low-Level Interface zwischen Anwendung und Server
- erlaubt es Komponenten Framework–übergreifend zu verwenden
- CGI / FastCGI / SCGI / AJP / mod_python / mod_wsgi / twisted / standalone
- einfach und schnell

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

lighttpd

Apache

mod_fastcgi / mod_scgi

mod_wsgi

Deployment

lighttpd

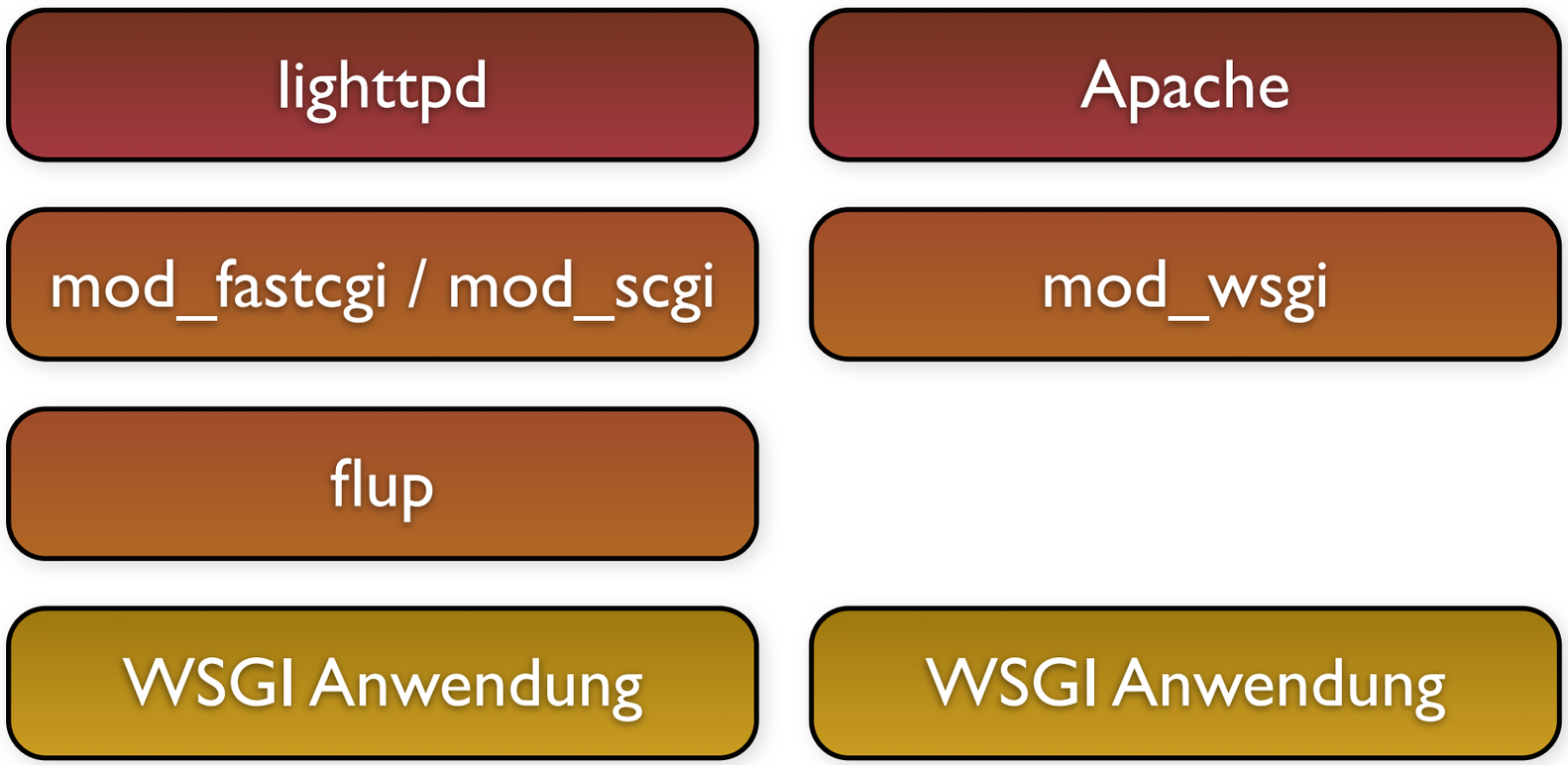
Apache

mod_fastcgi / mod_scgi

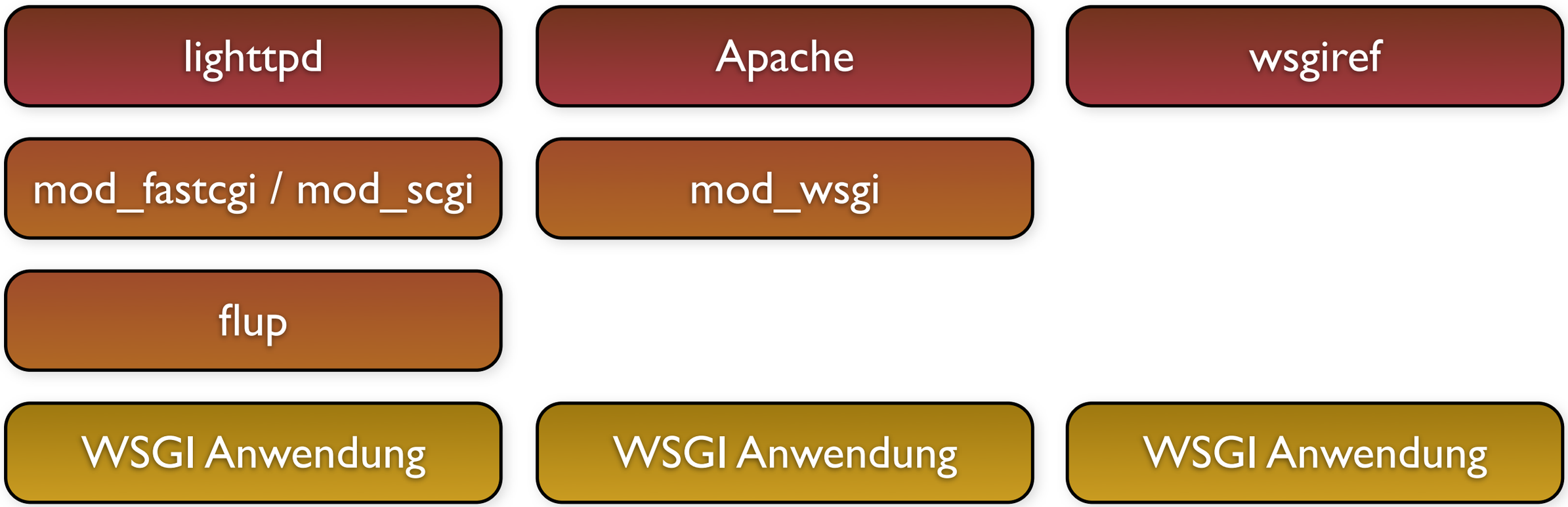
mod_wsgi

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 arbeiten zwischen Server und Anwendung

Middlewares

- Middlewares arbeiten zwischen Server und Anwendung
- Können eingehende und Ausgehende Daten abfangen oder verändern

Middlewares

- Middlewares arbeiten zwischen Server und Anwendung
- Können eingehende und Ausgehende Daten abfangen oder verändern
- nützlich um …
  - … Fehler zu protokolieren
  - … kaputte Serverdaten zu reparieren
  - … mehrere Anwendungen zu kombinieren

Aber....

Aber....

...eine Anwendung sollte nicht von einer Middleware abhängig sein.

Aber....

...eine Anwendung sollte nicht von einer Middleware abhängig sein.

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

Aber....

...eine Anwendung sollte nicht von einer Middleware abhängig sein.

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

Setup

Setup

- zentrale „Runner” Datei:

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- importiert Anwendung und Middlewares

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- importiert Anwendung und Middlewares
- verbindet sie und stellt ein `application` Objekt bereit oder ruft das Gateway auf.

Setup

- zentrale „Runner” Datei:
  - application.fcgi … mod_fastcgi
  - application.wsgi … mod_wsgi
  - run-application.py … standalone / wsgiref
- importiert Anwendung und Middlewares
- verbindet sie und stellt ein `application` Objekt bereit oder ruft das Gateway auf.

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

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

Zusammenfassung

Zusammenfassung

- Anwendung: aufrufbares Objekt

Zusammenfassung

- Anwendung: aufrufbares Objekt
  - `environ` … eingehende Daten

Zusammenfassung

- Anwendung: aufrufbares Objekt
  - `environ` … eingehende Daten
  - `start_response` … startet Senden der Daten

Zusammenfassung

- Anwendung: aufrufbares Objekt
  - `environ` … eingehende Daten
  - `start_response` … startet Senden der Daten
  - `app_iter` … Iterator, jede Iteration sendet Daten zum Browser

Zusammenfassung

- Anwendung: aufrufbares Objekt
  - `environ` … eingehende Daten
  - `start_response` … startet Senden der Daten
  - `app_iter` … Iterator, jede Iteration sendet Daten zum Browser
- Middleware: zwischen Anwendung und Gateway

Zusammenfassung

- Anwendung: aufrufbares Objekt
  - `environ` … eingehende Daten
  - `start_response` … startet Senden der Daten
  - `app_iter` … Iterator, jede Iteration sendet Daten zum Browser
- Middleware: zwischen Anwendung und Gateway
- Gateway: übersetzt WSGI nach CGI usw.

Werkzeug

Werkzeug

- Unicode Handling

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter
- URL Dispatching

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter
- URL Dispatching
- HTTP Parsing

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter
- URL Dispatching
- HTTP Parsing
- Entwicklungsserver

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter
- URL Dispatching
- HTTP Parsing
- Entwicklungsserver
- Autoreloader

Werkzeug

- Unicode Handling
- Form Daten / File Uploads / URL Parameter
- URL Dispatching
- HTTP Parsing
- Entwicklungsserver
- Autoreloader
- unzählige kleine Helferlein

Was ist es nicht?

Was ist es nicht?

- ORM

Was ist es nicht?

- ORM
- Template Engine

Was ist es nicht?

- ORM
- Template Engine
- Form-Validation

Was ist es nicht?

- ORM
- Template Engine
- Form-Validation
- i18n / l10n

Was ist es nicht?

- ORM
- Template Engine
- Form-Validation
- i18n / l10n
- Component Architecture

Was ist es nicht?

- ORM
- Template Engine
- Form-Validation
- i18n / l10n
- Component Architecture
- ein Framework

Warum nicht?

Warum nicht?

- gibt es alles schon

Warum nicht?

- gibt es alles schon
- kann man wunderbar kombinieren

Warum nicht?

- gibt es alles schon
- kann man wunderbar kombinieren
- jeder will was anderes

Warum nicht?

- gibt es alles schon
- kann man wunderbar kombinieren
- jeder will was anderes
- Cherry Picking!

Wie schaut es aus?

Wie schaut es aus?

```
>>> 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!

Ein Pastebin in 15 Minuten

Was wird verwendet?

Was wird verwendet?

- Werkzeug

Was wird verwendet?

- Werkzeug
- Jinja

Was wird verwendet?

- Werkzeug
- Jinja
- sqlite3

Was wird verwendet?

- Werkzeug
- Jinja
- sqlite3
- Pygments

Was wird verwendet?

- Werkzeug WSGI
- Jinja
- sqlite3
- Pygments

Was wird verwendet?

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

Was wird verwendet?

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

Was wird verwendet?

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

Was wird verwendet?

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

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

#0: Datenbank

#0: Datenbank

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: Konfiguration

#2: Konfiguration

```
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 %}
```

Entwicklungsserver

Entwicklungsserver

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

Entwicklungsserver

```
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 Aktion

„Dump It!“ In Aktion

„Dump It!“ In Aktion

Mehr als ein Weg

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox
- Daten: SQL / CouchDB / Dateisystem

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox
- Daten: SQL / CouchDB / Dateisystem
- AJAX: JSON / XML / HTML Fragmente

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox
- Daten: SQL / CouchDB / Dateisystem
- AJAX: JSON / XML / HTML Fragmente
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameter

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox
- Daten: SQL / CouchDB / Dateisystem
- AJAX: JSON / XML / HTML Fragmente
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameter
- Dispatching: Controller / View-Funktionen

Mehr als ein Weg

- Templates: XML / Text-basiert / Sandbox
- Daten: SQL / CouchDB / Dateisystem
- AJAX: JSON / XML / HTML Fragmente
- URLs: Regular Expressions / Werkzeug Routing / Routes / Objekt-basierend / Query Parameter
- Dispatching: Controller / View-Funktionen
- Auth: Apache / LDAP / OpenID

http://werkzeug.pocoo.org/

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