



Webframeworks für Python

Vorstellung und Vergleich

Andreas Knöpfle, Bastien Sachs und Tobias Schmid

Institut für Informatik

28. November 2012

Einstiegs-Idee:

viele Logos der Webframeworks für Python

Stichpunkte:

- In Python geschriebene Webframework gibt es wie „Sand am Meer“
- Warum? Einblick gibt unser Vortrag

Gliederung

1. Python – Programmiersprache fürs Web?

1.1. Codebeispiele/Gegenüberstellungen

2. Webframeworks für Python

2.1. Django

2.2. Vergleichskriterien

3. noch mehr Frameworks ;-)

4. Fazit

Python – Programmiersprache fürs Web?



- Entwurfsphilosophie betont Programmlesbarkeit
 - ▶ Blöcke durch gleiche Einrückung begrenzt
 - ▶ relativ wenige Schlüsselwörter
- Multiparadigmen-Sprache (primär OOP, funktional, AOP)
- dynamische Datentypen, garbage collection
- große, umfangreiche Standardbibliothek “batteries included”

Syntaxvergleich zu Java

```
int value = 0;
String title = String.valueOf(value);
if ("0".equals(title)) {
    throw new Exception("Error");
}
for (int i = 1; i < 10; i++) {
    System.out.printf("row %s", i);
}
```

```
value = 0
title = str(value)
if title == "0":
    raise Exception("Error")
for i in range(1, 10):
    print "row {}".format(i)
```



Quelle: [??]



Quelle: [??]

Web Server Gateway Interface (WSGI)

* Apache (mod_wsgi), nginx (uWSGI), Gunicorn
Bild

Python – Programmiersprache fürs Web!

Zusammenfassung der genannten Punkte

Webframeworks für Python (Auswahl)

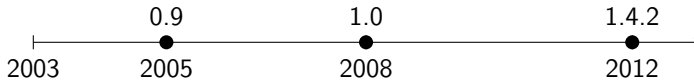
Full-Stack Frameworks

- Django
- TurboGears
- web2py
- Pylons/Pyramid
- ...

Microframeworks

- Bottle
- CherryPy
- ...

"The web framework for perfectionists with deadlines"



Farbdeutung bei Vergleichen

Hauptteil:

Vor- und Nachteile einiger weniger Frameworks aufzeigen

(konkrete) Lösungsansätze für bestimmte Probleme/Vergleichskriterien

sehr gute Lösung
in manchen Fällen evtl. nicht optimal
nicht optimale Lösung

SQL-ORMs

- Django built-in ORM (Django)
- SQLAlchemy (Pylons)
- SQLAlchemy (Pylons, TurboGears)
- DAL (web2py)
- ...

MongoDB ORMs

- MongoDB Engine (Django)
- Ming (TurboGears)

Full-Stack Frameworks haben entweder eigenes ORM oder ermöglichen den Einsatz von *SQLAlchemy* oder *SQLObject*.

Persistenz Beispiel Django

```
from django.db import models
```

```
class Employee(models.Model):  
    name = models.CharField(max_length=60)  
    boss = models.BooleanField(default=False)  
    concern = models.ForeignKey(Concern)
```

```
class Concern(models.Model):  
    name = models.CharField(max_length=40)
```

```
concern = Concern.objects.create(name="Musterfirma")  
Employee.objects.create(name="Mustermann", concern=concern)
```

```
# Alle Mitarbeiter des Konzerns zum Chef machen  
Employee.objects.filter(concern=concern).update(boss=True)
```

Formulare, Validierung

- `django.forms`
 - ▶ HTML form widget
 - ▶ Field validation
 - ▶ ...

Templates

Templates in Python

- Text-basiert:
 - ▶ Django template language
 - ▶ Cheetah
 - ▶ Mako (früher Myghty)
 - ▶ Jinja2
- XML-basiert:
 - ▶ Genshi
 - ▶ SimpleTAL

Es gibt noch sehr viel mehr Template-Engines
(siehe <http://wiki.python.org/moin/Templating>)

Templates

Django template language

- Template-Inheritance
- Unterstützt Filter und Kontrollstrukturen

```
<body>
  <div id="content">
    {% block content %}{% endblock %}
  </div>
</body>
```

```
{% extends "base.html" %}
```

```
{% block content %}
  Inhalt: {{ content }}
{% endblock %}
```

Templates

Cheetah

- Wird von allen Full-Stack-Frameworks unterstützt
- Große und aktive Community
- Templates können von Python-Klassen abgeleitet werden

```
<table>
  #for $client in $clients
    <tr>
      <td>$client.surname, $client.firstname</td>
      <td><a
        href="mailto:$client.email">$client.email</a></td>
    </tr>
  #end for
</table>
```


I18N und L10N

- gettext

Konfiguration, Routing

Django

- URLconf (URL Konfiguration)
- einfaches Mapping zwischen URL-Patterns (Regex)
- Wenn der Ausdruck passt, ruft Django den View auf
- settings.py für alle weiteren Konfigurationen
 - ▶ Datenbankeinstellungen
 - ▶ E-Mail und Fehlermeldungseinstellungen
 - ▶ i18n und URL Einstellungen
 - ▶ Applikations and Middleware Einstellungen

Routing Beispiel Django

```
urlpatterns = patterns('news.views',  
    (r'^articles/2003/$', 'special_case_2003'),  
    (r'^articles/(\d{4})/$', 'year_archive'),  
    (r'^articles/(\d{4})/(\d{2})/$', 'month_archive'),  
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'article_detail'),  
)
```

/articles/2005/03/	=> month_archive(request, '2005', '03')
/articles/2005/3/	=> no match
/articles/2003/	=> special_case_2003(request, '2003')
/articles/2003	=> no match
/articles/2003/03/03/	=> article_detail(request, '2003', '03', '03')

Konfiguration, Routing

TurboGears

- Object Dispatch, und built in Routes Integration
- kann überschrieben werden

Routing Beispiel TurboGear 2.0

```
def setup_routes(self):
    map = Mapper(
        directory=config['pylons.paths']['controllers'],
        always_scan=config['debug'])

    # Setup a default route for the root of object dispatch
    map.connect('*url', controller='root',
               action='routes_placeholder')
    config['routes.map'] = map
```

Vergleich Konfiguration, Routing

	Django	TurboGear 2.0
Routing	mächtig	einfach,anpassbar

Sicherheitsmechanismen

- django: <http://www.djangobook.com/en/2.0/chapter20.html>

Bootstrapping, Scaffolding, Erweiterbarkeit

Extras: WebServices, Caching, Tests

“... microframework for Python based on Werkzeug, Jinja2 and good intentions. . .”

- Flexibilität für Entwickler
- Entwicklungsserver und Debugger
- Unittest-Unterstützung
- RESTful
- Template-Sprache Jinja2
- inspiriert von Sinatra (Ruby)



Quelle: [2]

Flask

Minimalbeispiel einer Webanwendung

hello.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

```
$ python hello.py
* Running on http://127.0.0.1:5000/
```

Tornado

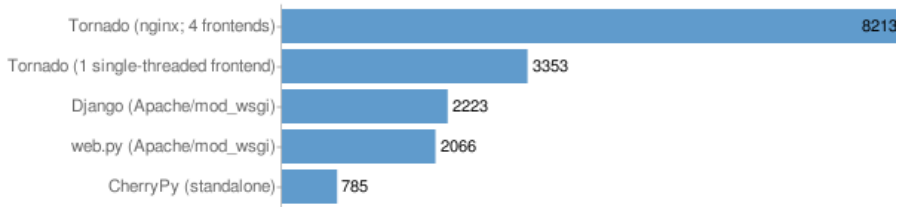
“... FriendFeed’s web server is a relatively simple, non-blocking web server written in Python... Tornado is an open source version of this web server and some of the tools we use most often at FriendFeed...”

- Webframework und Webserver für Python
- Große Anzahl gleichzeitiger Verbindungen



Tornado Performance

Web server requests/sec (AMD Opteron, 2.4GHz, 4 cores)



Quelle: [4]

Tornado

Minimalbeispiel einer Webanwendung

hello.py

```
from tornado import ioloop, web

class MainHandler(web.RequestHandler):
    def get(self):
        self.write("Hello, world")

application = web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
    application.listen(8888)
    ioloop.IOLoop.instance().start()
```


Kriterienübersicht

- * Vergleichstabellen (Django vs. ...)

Fazit

Je Anforderungen an das Webframework (“Taste”)

...

Quellen

- <http://www.infoworld.com/d/application-development/pillars-python-six-python-web-frameworks-compared-169442>
- <http://wiki.python.org/moin/WebFrameworks>
- <http://wiki.python-forum.de/Web-Frameworks>
- <http://blog.ianbicking.org/turbogears-and-pylons.html>

alle URLs aufgerufen am 20. November 2012.

Quellen der Abbildungen

1. Innenhof Informatik <http://www.flickr.com/photos/bennybenny/3597853896/>
2. Fask Logo <http://flask.pocoo.org/static/logo.png>
3. Tornado Logo <http://www.tornadoweb.org/static/tornado.png>
4. Tornado Statistik (Bret Taylor, 10. September 2009)
<http://http://developers.facebook.com/blog/post/301/>

alle URLs aufgerufen am 14. November 2012.

Ende der Präsentation