



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?

2. Webframeworks für Python

3. Fazit

Farbdeutung bei Vergleichen

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

Python – Programmiersprache fürs Web?



- Entwurfsphilosophie betont Programmlesbarkeit
 - ▶ Blöcke durch gleiche Einrückung begrenzt
 - ▶ relativ wenige Schlüsselwörter
- objektorientierte, aspektorientierte und funktionale Programmierung
- dynamische Datentypen, garbage collection
- große, umfangreiche Standardbibliothek “batteries included”

Python – Programmiersprache fürs Web?

eventl. Codebeispiele, Gegenüberstellungen, Erklärung

Web Server Gateway Interface (WSGI)

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

Python – Programmiersprache fürs Web!

Zusammenfassung der genannten Punkte

Webframeworks für Python

Full-Stack Frameworks

- Django

Microframeworks

Webframeworks für Python

Full-Stack Frameworks

- Django
- TurboGears

Microframeworks

Webframeworks für Python

Full-Stack Frameworks

- Django
- TurboGears
- web2py

Microframeworks

Webframeworks für Python

Full-Stack Frameworks

- Django
- TurboGears
- web2py
- Pylons/Pyramid

Microframeworks

Webframeworks für Python

Full-Stack Frameworks

- Django
- TurboGears
- web2py
- Pylons/Pyramid

Microframeworks

- Bottle

Webframeworks für Python

Full-Stack Frameworks

- Django
- TurboGears
- web2py
- Pylons/Pyramid

Microframeworks

- Bottle
- CherryPy

Hauptteil:

Vor- und Nachteile einiger weniger Frameworks aufzeigen

(konkrete) Lösungsansätze für bestimmte Probleme/Vergleichskriterien

SQL-ORMs

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

MongoDB ORMs

- Django MongoDB Engine (Django)
- Ming (TurboGears)

Persistenz in Django

```
from django.db import models
class Item(models.Model):
    name = models.CharField(max_length=60)
    created = models.DateTimeField(auto_now_add=True)
    priority = models.IntegerField(default=0)
    difficulty = models.IntegerField(default=0)
    done = models.BooleanField(default=False)
```

Templates, I18N, L10N

- Template Engines:
 - Python built-in
 - Django template language
 - Cheetah (Django, Turbogears, Pylons)
 - Myghty
- I18N + 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

Konfiguration, Routing

TurboGears

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

Routing Beispiel Django

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

Requests:

```
/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')
```

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
Konfiguration		
Routing	mächtig, evtl. Overload	einfach,anpassbar

Formulare, Validierung

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

Sicherheitsmechanismen

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

Bootstrapping, Scaffolding, Erweiterbarkeit

Extras: WebServices, Caching, Tests

Kriterienübersicht

- * Vergleichstabellen (Django vs. ...)

Fazit

Je Anforderungen an das Webframework (“Taste”)

...

Quellen der Abbildungen

1. Innenhof Informatik <http://www.flickr.com/photos/bennybenny/3597853896/>

alle URLs aufgerufen am 14. November 2012.

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>

Ende der Präsentation

- Vielen Dank für Ihre Aufmerksamkeit.
- offene Fragen?
- Diskussion
 - Kritik, Anregungen