Freie Universität Berlin

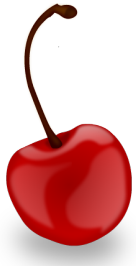
Webframeworks für Python

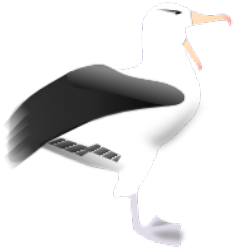
Vorstellung und Vergleich

Andreas Knöpfle, Bastien Sachs und Tobias Schmid

Institut für Informatik

28. November 2012





Bottle

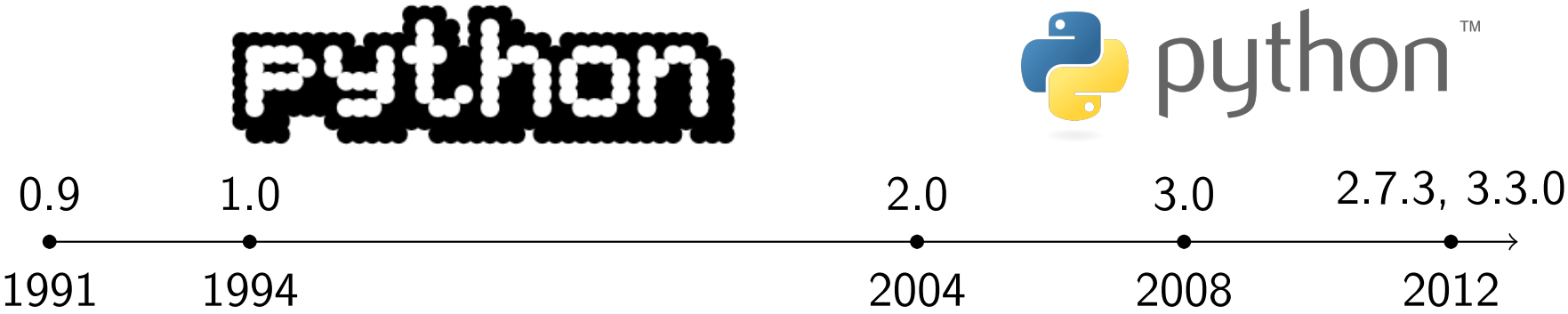
django

web.py

Gliederung

Quelle: [2]

Python – Programmiersprache fürs Web?



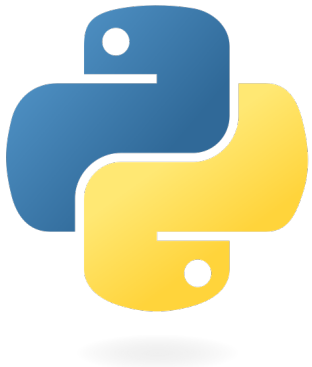
- Entwurfsphilosophie betont Programmlesbarkeit
  - reduzierte Syntax
  - relativ wenige Schlüsselwörter
- Multiparadigmen-Sprache (primär OOP/imperativ, auch funktional, AOP)
- dynamische Datentypen, garbage collection
- große, umfangreiche Standardbibliothek “batteries included”

Syntaxvergleich zu Java

```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);
}
```

Quelle: [3]

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

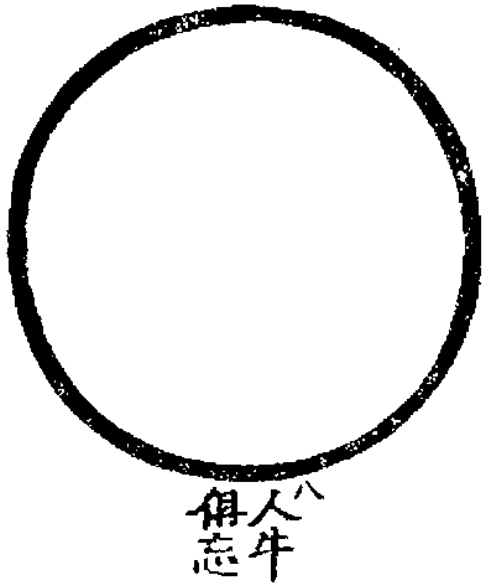


Quelle: [4]

Kernphilosophie – The Zen of Python

"Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
...
Readability counts.
...
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
..."

Quelle: [14]



Quelle: [5]

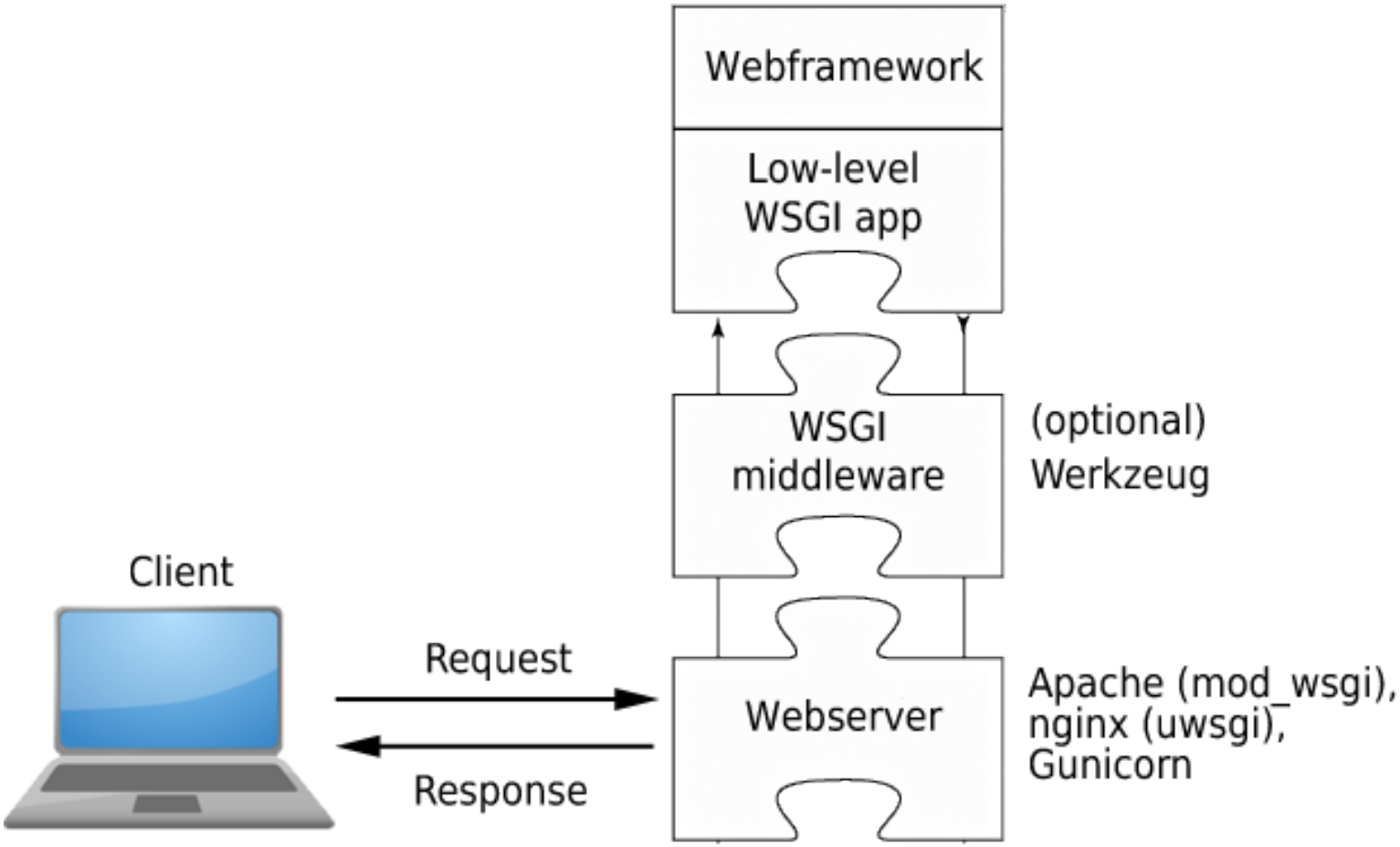
Softwarequalität im Plattformvergleich

Eigenschaft	+		-
Aufwand	**Python**	Ruby	Java, PHP
Lesbarkeit	**Python**	Ruby	Java, PHP
Modifizierbarkeit	**Python**	Ruby	Java, PHP
Sicherheit	Java, **Python**, Ruby	–	PHP
Werkzeuge	Java	PHP, **Python**, Ruby	

Ähnliche/sekundäre Eigenschaften: Geschwindigkeit, Skalierbarkeit und Speicherbedarf

Vereinfachte Übersicht auf Grundlage von *Comparing Web Development Platforms Through the Eyes of Professional Developers* [Har07]

Web Server Gateway Interface (WSGI)



WSGI Hello World App

```
def simple_app(environ, start_response):
    """Simplest possible application object"""
    status = "200 OK"
    response_headers = [("Content-type", "text/plain")]
    start_response(status, response_headers)
    return ["Hello World!\n"]
```

Python – Programmiersprache fürs Web!

- Eignung Programmiersprache für Aufgabe
- Fokus auf Produktivität & schnelle Entwicklung
- gute Softwarequalität in der Praxis
- breite Auswahl an Frameworks
- wenig Entwickler, aber leicht lernbar

Webframeworks für Python (Auswahl)

Full-Stack Frameworks

- **Django**
- **TurboGears**
- web2py
- Pylons/Pyramid
- Zope2
- Grok
- ...

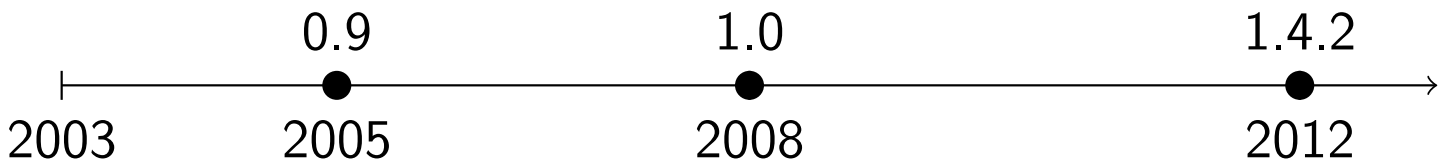
Microframeworks

- **Bottle**
- **Tornado**
- **Flask**
- **web.py**
- CherryPy
- ...

django

Quelle: [12]

> *"The web framework for perfectionists with deadlines"*



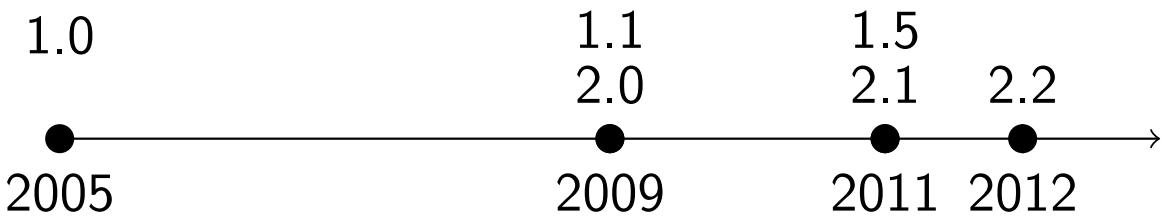
Wichtige Designprinzipien:

- Don't repeat yourself (DRY)
- Explicit is better than implicit

Turbogears

Quelle: [13]

> *"The next generation web framework that scales with you."*



- Aus extern gepflegten Python-Modulen zusammengesetzt
- Module sind austauschbar

Persistenz

SQL-ORMs

- Django built-in ORM (Django)
- SQLAlchemy (Pylons, Turbogears 2)
- SQLObject (Pylons, TurboGears 2)
- DAL (web2py)
- ...

MongoDB ORMs

- MongoDB Engine (Django)
- Ming (TurboGears 2)

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

Persistenz

Django

```
from django.db import models

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

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

```
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

- Formulare werden durch die Helperklasse Form erzeugt
- Können aus dem Model abgeleitet werden
- Validatoren werden ebenfalls aus dem Model abgeleitet

```
from django.forms import ModelForm

class EmployeeForm(ModelForm):
    class Meta:
        model = Employee

# Formular erzeugen
form = EmployeeForm()

# Formular zum editieren erzeugen
employee = Employee.objects.get(pk=1)
form = EmployeeForm(instance=employee)
```

Formulare, Validierung

Django

```
<form action="{% url employee_add %}" method="post">
  {{ form.as_p }}  {% csrf_token %}
  <input type="submit" value="Submit" />
</form>
```

```
<form action="/employee/add/" method="post">
<p><label for="id_name">Name:</label> <input id="id_name"
    type="text" name="name" maxlength="60" /></p>
<p><label for="id_boss">Boss:</label> <input type="checkbox"
    name="boss" id="id_boss" /></p>
<p><label for="id_concern">Concern:</label> <select
    name="concern" id="id_concern"><option value=""
    selected="selected">---------</option><option
    value="1">Concern object</option>
</select></p>  <div style='display:none'><input type='hidden'
    name='csrfmiddlewaretoken'
    value='mgPxkm0c21WH81X0LIZEFoBxrDhCY2oW' /></div>
  <input type="submit" value="Submit" />
</form>
```

Formulare, Validierung

Turbogears 2

- Hier kann *DBSprockets* verwendet werden
- DBSprockets unterstützt *SQLAlchemy*
- Funktionsweise analog zu Django
- Alternative: *Sprox*, *ToscaWidgets*

```
from dbsprockets.declaratives import FormBase

class EmployeeForm(FormBase):
    __model__ = Employee

form = EmployeeForm()
```

Templates

Templates in Python

- Text-basiert:
  - Django template language
  - Cheetah
  - Mako (füher 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-Vererbung
- Trennung von Programmlogik und -darstellung
- Unterstützt Filter und elementare Kontrollstrukturen

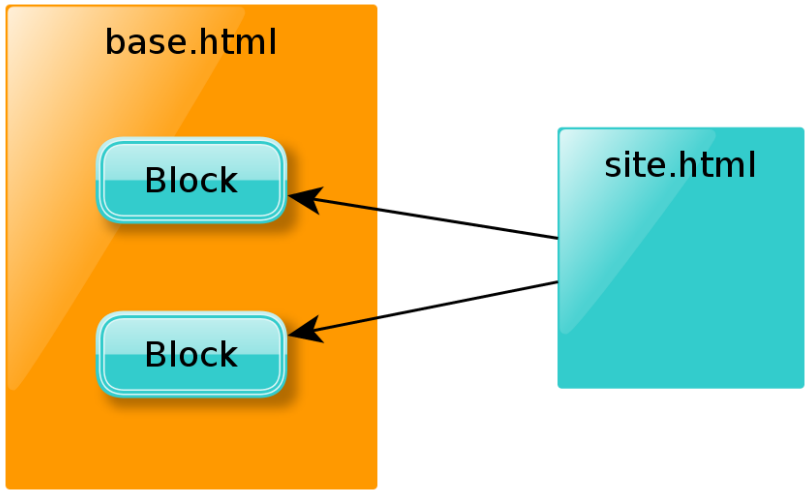
base.html

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

---

site.html

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

{% block main %}
    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

Django

- Unterstützt Textübersetzung, Datums-, Zeit und Zahlenformatierung, Zeitzonen und Pluralisierung
- Markieren der Texte mit *ugettext*
- Übersetzung innerhalb von Python, Templates und JavaScript

```
from django.utils.translation import ugettext as _

def my_view(request):
    output = _("Welcome to my site.")
    return HttpResponse(output)
```

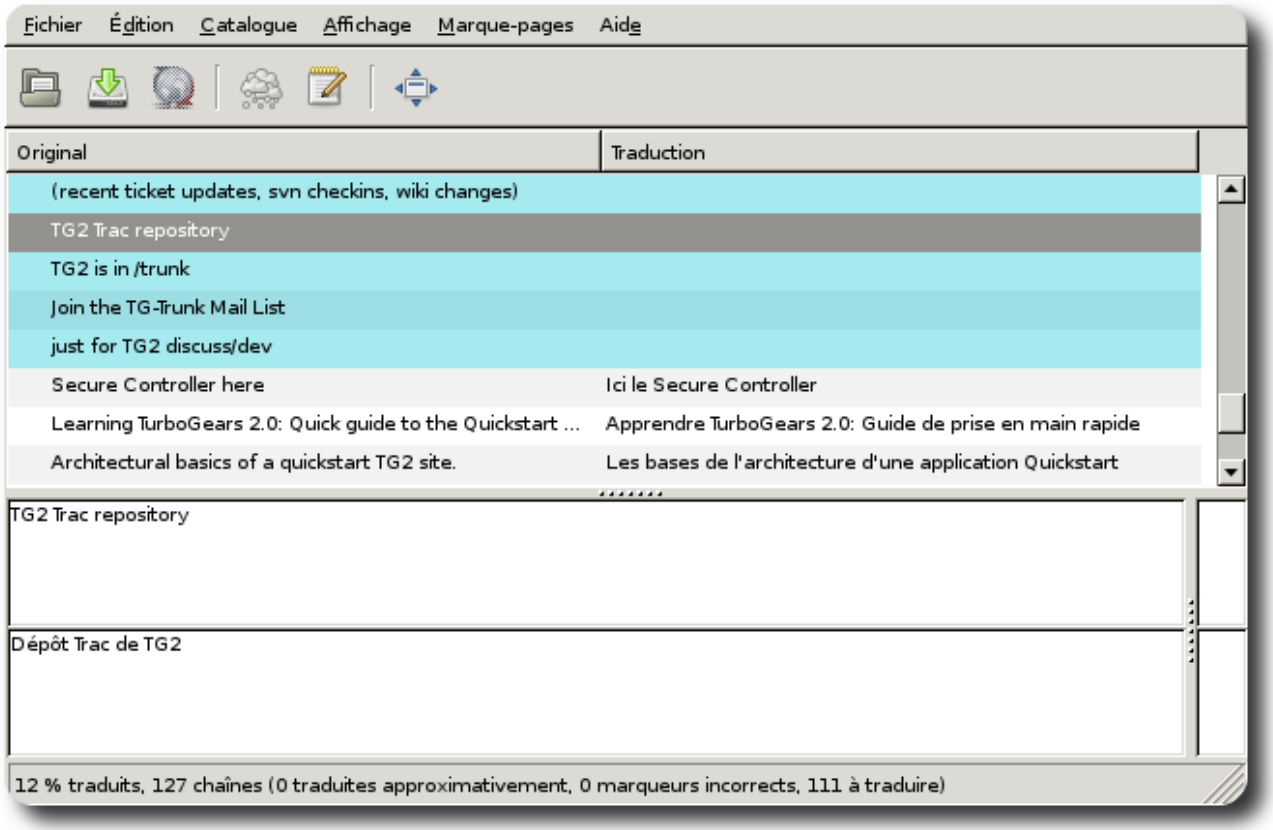
- Werkzeuge zum Erstellen neuer Sprachdateien:
  django-admin.py makemessages -l de

```
#: path/to/python/module.py:23
msgid "Welcome to my site."
msgstr ""
```

I18N und L10N

Turbogears 2

- Internationalisierung mit Python Modul *gettext*
- Funktionsweise analog zu Django
- .po-Sprachdateien können auch mit Hilfsmitteln bearbeitet werden



Quelle: [11]

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')
```

Routing

TurboGears 2

> *"... By default you don't need to think about Routes at all ..."*

- Object Dispatch, und built in Routes Integration
- kann überschrieben werden
- Verwendet Routes (Python Implementierung des Rails-Routes-System)

Routing Beispiel TurboGears 2

```
class RootController(BaseController):

    @expose()
    def index(self):
        return "<h1>Hello World</h1>"

    @expose()
    def default(self, *args, **kw):
        return "This page is not ready"
```

```
http://localhost:8080/index => Hello World

http://localhost:8080/hello => This page is not ready
```

Sicherheitmechanismen

Django

- SQL-Injection
  - automatisches Escape von Benutzereingaben
- Cross-Site Scripting (XSS)
  - automatisches Escape von Benutzereingaben
- Cross Site Request Forgery (CSRF)
  - Eingebaute Helper mit CSRF Token
- Session Forging/Hijacking
  - Session Informationen niemals in URL
  - Daten nicht direkt in Cockies speichern (sondern über Session ID)
  - Escape von Session Daten, wenn in Template dargestellt
  - Eingebauter Schutz gegen Brute-Force Session-Angriff

Scaffolding

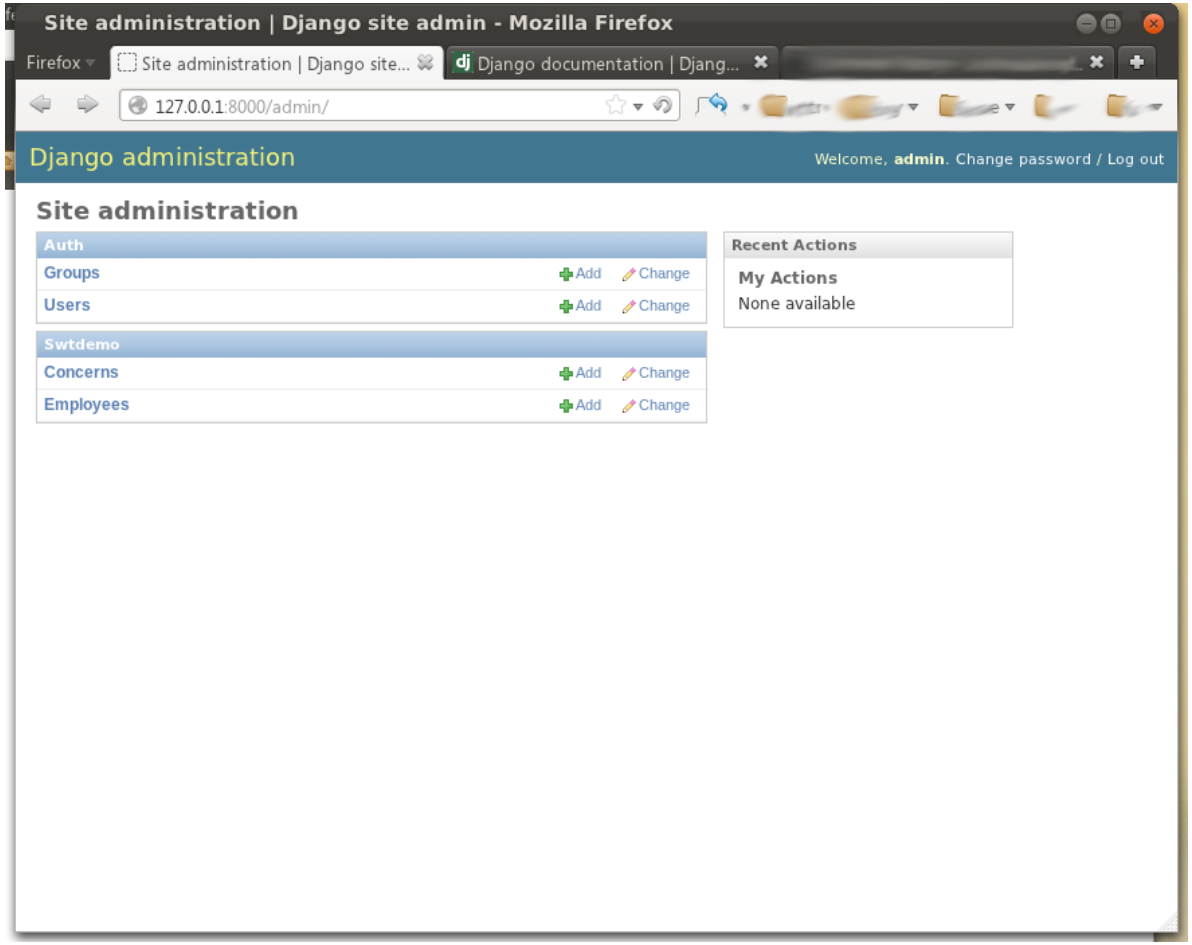
Django

- Generic-Views (siehe Beispiel-Applikation)
- django-common-helpers plugin (https://github.com/Tivix/django-common)
  - create app
  - create models
  - create views
  - create templates
  - create forms
  - create urls
  - create tests

Extras

Django Administration

- Eingebaute Administrationsoberfläche
- Registrierte Modelle können verwaltet werden



Kriterienübersicht

	Django	TurboGears 2
Bootstrapping	✓	✓
Scaffolding	div. Apps	tgcrud
Persistenz	✓	SQLAlchemy
Formulare	✓	dbsprockets
Validierung	✓	FormEncode
Routing	✓(RegEx-Patterns)	Routes
Templates	✓	✓
L10N	✓	✓
I18N	✓	✓
Sicherheit	✓	modulabhängig
Unit testing	✓	nose
Caching	✓	Beaker
Plugins	1525+	

vorhanden
über Module zuschaltbar

Quelle: [6]

> *"... microframework for Python based on Werkzeug, Jinja2 and good intentions..."*

Quelle: [10]

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

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/
```

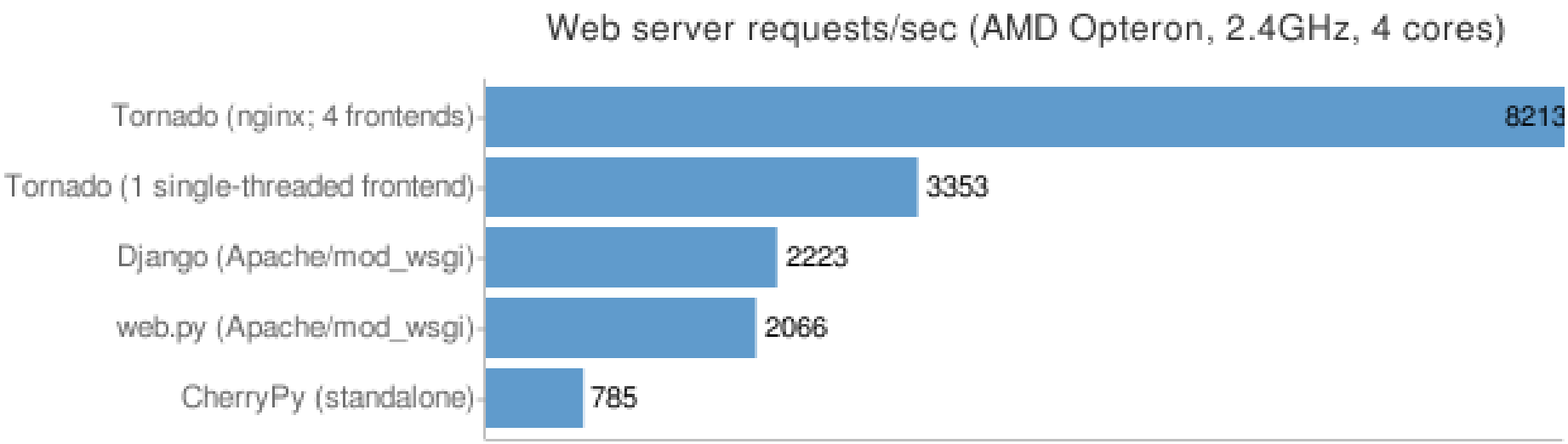
Tornado

Quelle: [7]

- Webframework und Webserver für Python

> *"... 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..."*

Quelle: [11]



Quelle: [8]

Quelle: [7]

Was kann/ist Tornado noch ?

- Core Web-Framework
  - Template System (tornado.template)
  - Internationalisierung (tornado.locale)
  - Unit Testing (tornado.testing)
- Einige weitere Hilfsmittel, Services, ...
  - Nicht blockierender HTTP-Client
  - Login mit OpenID und OAuth (3rd party)
  - MySQL client Wrapper
  - ...

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()
```

Quelle: [9]

> *"... a fast, simple and lightweight WSGI micro web-framework for Python ..."*

Quelle: [12]

- Was kann/ist Bottle ?
  - Routing
  - Eingebaute Template-Engine (unterstützt auch mako, jinja2 und cheetah)
  - Keine Abhängigkeiten (alles in einem Modul)
  - Eingebauter Entwicklungsserver

Bottle Beispiel

```
from bottle import route, run, template

@route('/hello/:name')
def index(name='World'):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

Quelle: [10]

> *"... is a web framework for Python that is as simple as it is powerful ..."*

Quelle: [13]

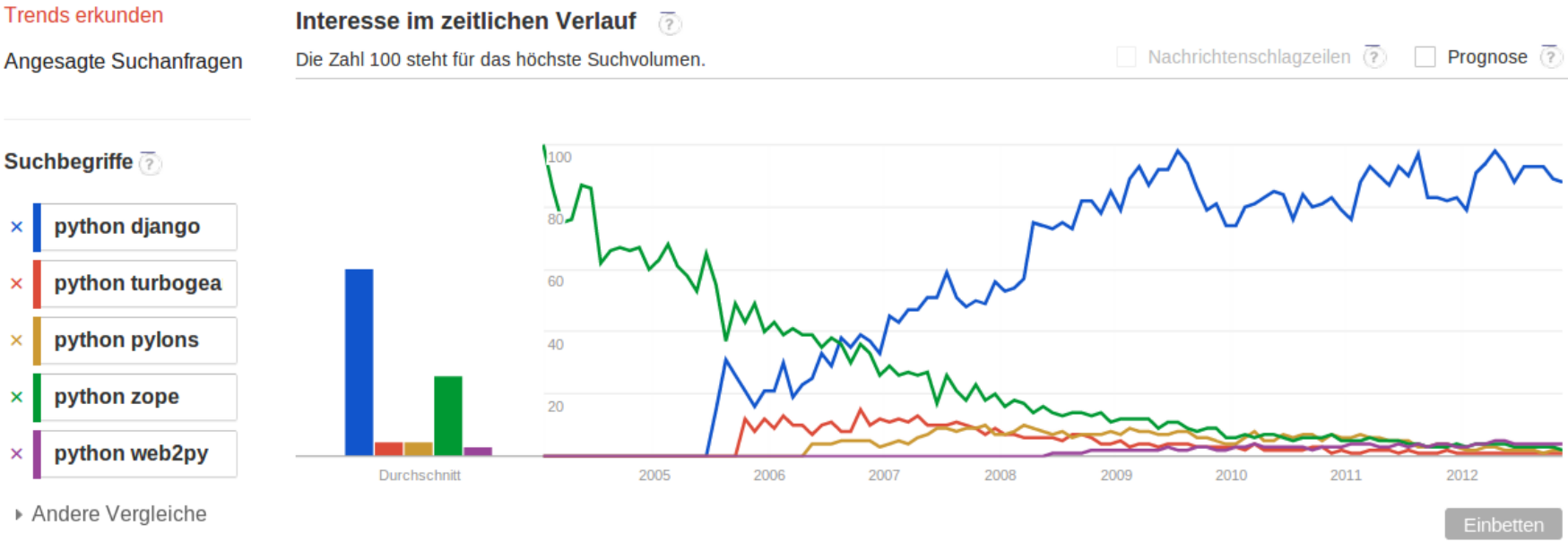
- Was kann/ist web.py ?
  - Eigene Template Sprache (angelehnt an Python)
  - Eingebaute Formularunterstützung
  - Abstraktion der Datenbank API ohne ORM

web.py Beispiel

```
import web

urls = (
    '/(.*)', 'hello'
)
app = web.application(urls, globals())

class hello:
    def GET(self, name):
        if not name:
            name = 'World'
        return 'Hello, ' + name + '!'

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

Beliebtheit Full-Stack-Frameworks

Google Trends

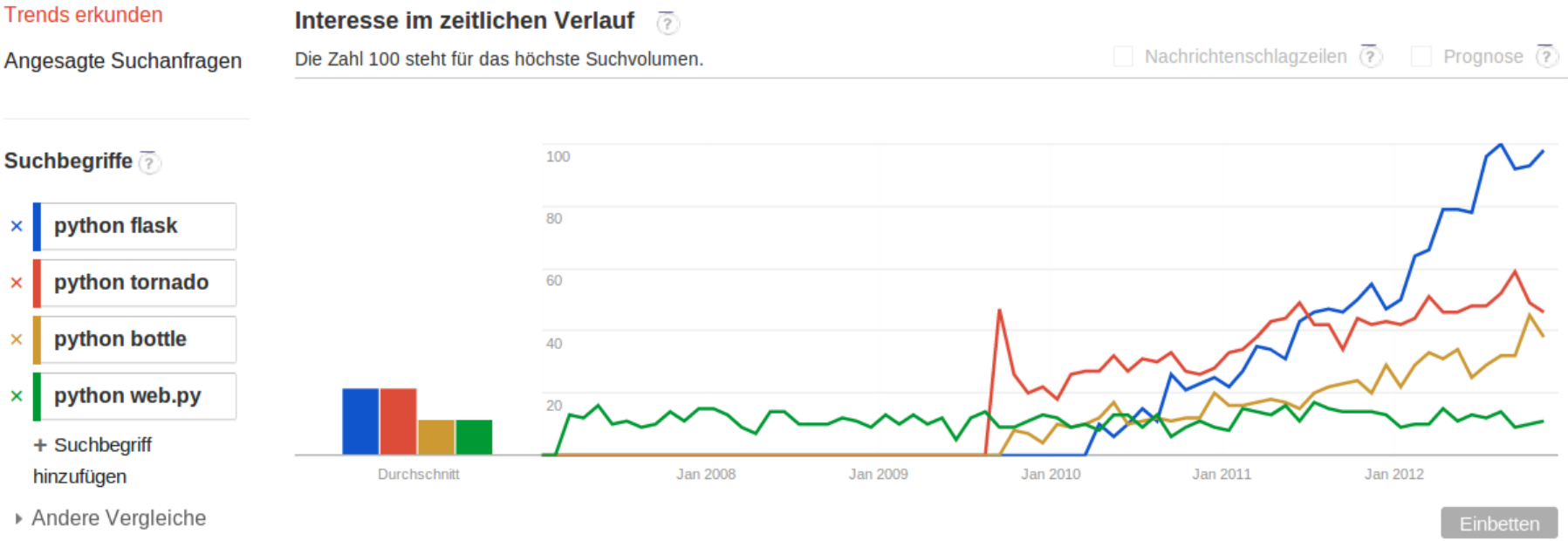


Anzahl der Themen in den jeweiligen Mailinglisten:

- Django: 37967
- web2py: 19329
- Turbogears 2: 9959
- Pylons: 4966

Beliebtheit Microframeworks

Google Trends



Fazit

- Je nach Vorliebe, Anforderungen und Vorkenntnissen
- Jedes Framework hat Stärken/Schwächen

Quelle: [14]

Quellen

[Har07] Will Hardy. *Comparing Web Development Platforms Through the Eyes of Professional Developers*. Techn. Ber. Freie Universität Berlin, Institut für Informatik, Nov. 2007.
URL: http://www.mi.fu-berlin.de/inf/publications/techreports/tr2007/B-07-14/index.html.

- 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
- https://docs.djangoproject.com/en/dev/topics/
- https://docs.djangoproject.com/en/dev/misc/design-philosophies/
- http://www.djangobook.com/en/2.0/chapter20.html

alle URLs aufgerufen am 20. November 2012.

Quellen der Abbildungen

1. Innenhof Informatik http://www.flickr.com/photos/bennybenny/3597853896/
2. Albino Python http://www.flickr.com/photos/34525959@N08/3522274687/
3. Logo Java http://de.wikipedia.org/w/index.php?title=Datei:Java-Logo.svg&oldid=95163279
4. Logo Python http://www.python.org/community/logos/
5. Bild Zen http://de.wikipedia.org/w/index.php?title=Datei:Ochsenbild08Haikon.png&oldid=101311361
6. Flask Logo http://flask.pocoo.org/static/logo.png
7. Tornado Logo http://www.tornadoweb.org/static/tornado.png
8. Tornado Statistik (Bret Taylor, 10. September 2009) http://http://developers.facebook.com/blog/post/301/
9. Bottle Logo http://bottlepy.org/docs/dev/_static/logo_nav.png
10. web.py Logo http://webpy.org/static/webpy.gif
11. po-Editor http://www.turbogears.org/2.2/docs/_images/poedit.png
12. Django Logo https://www.djangoproject.com/m/img/logos/django-logo-positive.png
13. Turbogears Logo http://turbogears.org/2.0/docs/_static/tg.png
14. Python kids http://www.flickr.com/photos/8113820@N05/2458633114/

alle URLs aufgerufen am 14. November 2012.

Quellen der Zitate

10. Flask `http://flask.pocoo.org/`
11. Tornado `http://www.tornadoweb.org/documentation/overview.html`
12. Bottle `http://bottlepy.org/docs/dev/`
13. web.py `http://webpy.org/`
14. The Zen of Python `http://www.python.org/dev/peps/pep-0020/`

alle URLs aufgerufen am 27. November 2012.