Netzwerkprogrammierung – Network Programming

REST / FLASK

Jan Krüger
jkrueger@cebitec.uni-bielefeld.de

Alexander Sczyrba
asczyrba@cebitec.uni-bielefeld.de

REST / FLASK

- REST
  - Was ist das WWW ?
  - RPC Protokolle
- PIP
- FLASK
  - HalloWelt.py

Was ist eigentlich das WWW ?

- Komponenten für einen menschlichen Betrachter
  - Texte (i.d.R. HTML)
  - Bilder
  - Filme / Musik / Tondokumente
- Browser als klassischerClient, heutzutage „Apps“
- Maschinenauswertbare Informationen
  - Rohdaten
  - Datenübertragung
  - Funktionen

RPC Architekturen

- RMI
  - Sprachabhängigkeit (z.B. Java-RMI)
  - OS Abhängigkeit
- CORBA
  - Industriestandard
  - Sehr flexibel und maechtig → sehr komplex
- XMLRPC
  - XML
- SOAP
  - W3C Empfehlung : `ws-*`
  - XML → grosser Overhead
  - Komplex, unzählige Erweiterungen

Das muss doch einfacher gehen? REST

1) Adressierbarkeit

   z.B. https://www.kaufladen.de/warenkorb/kunde/123456

2) Zustandslosigkeit

3) Einheitliche Schnittstelle

   z.B. HTTP → GET/PUT/POST/DELETE

4) Entkopplung von Resourcen und Repräsentation

Beispiel / Aufgabe

Sucht Euch auf dem BiBiServ (https://bibiserv.cebitec.uni-bielefeld.de) ein einfaches Werkzeug mit Beispiel aus (z.B. Dialign) :

1) Probiert das Beispiel über den Browser aus
2) Probiert das gleiche Beispiel mit der Kommandozeile aus (WebService – Tutorial)
3) Was nehmt Ihr aus der Übung (inhaltlich) mit ?

PIP – Python Install Package

- Vereinfacht die Installation von PyPa Paketen
- Support für die jeweils letzte Version ab Python 2.6
- Unterstützt (und installiert) virtualenv
  - `pip install virtualenv` :-)
- Unterstützt die Installation von Paketen ins Benutzerverzeichnis (keine Rootrechte notwendig)
  - `pip install –user $USER <package>`

Was ist FLASK ?

- Python Web Application Framework
- Installation via
  `pip install –user flask`
  (für python2 UND python3)
- BuiltIn Server zu Test/Debug Zwecke
- sehr gute Dokumentation :
  http://flask.pocoo.org/docs/0.11/
- einfach : Python Module (Klassen) mit Annotation
- Templates

HalloWelt.py

- Code

```
from flask import Flask
app = Flask(__name__)

@app.route( / )
def hello_world():
        return “Hallo Welt!“
```

- Flask starten :

```
$ export FLASK_APP=HalloWelt.py
$ python[3] -m flask run
```

- Browser:
  http://127.0.0.1:5000

Aufgabe

- Implementiert eine Klasse „Rechner“ das die folgenden Funktionalitäten implementiert und via REST (mit FLASK) publiziert.
  - Addition / Subtraktion
  - Multiplikation / Division
  - Summe
- Welche Probleme müssen berücksichtigt werden ?
- Testet den Client mittels CURL