RESTful Interfaces und Templates mit Flask

Einleitung

Im Folgenden soll eine kleine Flask-Anwendung gebaut werden die ein RESTFul Interface implementiert. Das Thema dreht sich um den Stundenplan der FH Hof, den wir mit einem Python-Skript von der Website geparst haben, und als ISON im Ordner data/ bereitstellen.

Für diese Übung müsst ihr lediglich app.py anpassen. Momentan besteht diese aus dem Flask HelloWorld.

Ausführen könnt ihr eure Anwendung indem ihr in das Directory practice navigiert, und dort app.py ausführt. Wenn alles gut ging, dann könnt ihr unter http://localhost:5000 in euren Webbrowser eine Begrüßungsmessage sehen.

Testen könnt ihr die Aufgaben a), b) und c) indem ihr test.py ausführt:

```
$ python test.py a b c
** Querying: http://localhost:5000/api/count/all
** Querying: http://localhost:5000/api/count/Inf
** Querying: http://localhost:5000/api/list
** Querying: http://localhost:5000/api
...
Ran 4 tests in 0.016s
OK
```

Aufgabe **e)** können wir nach Termin gemeinsam testen.

Die Musterlösung stellen wir gegen Ende der Stunde dann auf Dropbox.

Aufgaben

a)

Erweitere app.py so dass eine Abfrage von http://localhost:5000/api/lnf/5 unseren aktuellen Stundenplan liefert. Der Studiengang Inf und das Semester soll dabei entsprechend durch andere Studiengänge wie BW ersetzt werden können.

Beispiel Antwort:

```
1 $ curl localhost:5000/api/Inf/5
2 {
3
        "Dienstag": [{
            "room": "FB102",
4
            "prof": "Prof. Dr. J\u00fcrgen Heym",
5
6
            "time": "11:30-13:00",
7
            "name": "RoutingSwitchingTroubleshooting 1",
8
            "note": "(Inf+MI+WI5)",
9
            "type": "FWM:1"
10
       },
11
12
13 }
```

Für das Abfragen eines Python Dictionaries mit dem Stundenplan könnt ihr die Funktion load(studiengang, semester) nutzen. Für das Umwandeln in valides JSON könnt ihr die eingebaute Funktion json.dumps(python_object) nutzen.

Wird ein Stundenplan von einem nichtexistierenden Kurs geholt, so soll ein leeres Dictionary zurückgegeben werden. ({}). Für nichtexistierende Kurse löst die load-Funktion eine NoSuchCourse Exception aus.

b)

http://localhost:5000/api/list courses soll eine sortierte Liste aller Studiengänge wiedergeben.

Beispiel:

```
$ curl localhost:5000/api/list_courses
["BBB", "BW", "GP", "IM", "Inf", ..., "Wing MT", "Wing WT"]
```

Zum Abfragen der unsortierten Kursliste könnt ihr die Funktion list_courses() verwenden.

c)

http://localhost:5000/api/count/studiengang soll die Anzahl der aktuell studierenden Semester für einen Studiengang ausgeben.

Für Inf wäre das beispielsweise **3** (== len('Inf1', 'Inf3', 'Inf5')).

Für den speziellen Studiengang all sollen alle Studiengänge gezählt werden.

Beispiel:

```
$ curl localhost:5000/api/count/Inf
3
$ curl localhost:5000/api/count/all
84
```

Zum Abfragen der Anzahl könnt ihr die bereitgestellte Funktion count(studiengang) nutzen. Das Argument studiengang ist optional. Lässt man es weg werden alle Studiengange gezählt.

d)

Erweitere die Flask-Anwendung um eine weitere URL:

http://localhost:5000/view/studiengang/semester.

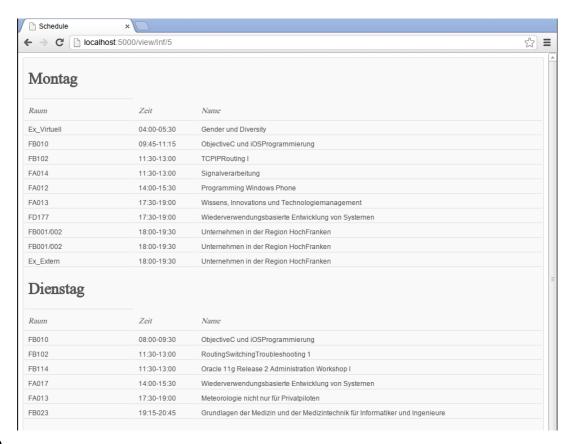
Diese soll bei Aufruf von beispielsweise view/Inf/5 unseren Stundenplan rendern.

Nutze dazu die Methode render_template() und schreibe ein Jinja2 Template dass fähig ist den Stundenplan als HTML zu rendern.

In templates/simple_table.html ist eine leichte Hilfestellung bzgl. HTML Tables. Ein brauchbares CSS ist bereits eingebaut.

Ihr müsst euch nicht ans Template halten, ihr könnt es beliebig verändern.

Beispiel Bild:



e)

Bitte backt uns eine Python-Torte:

http://www.cakefriday.de/2012/03/python-cake.html

