Web Programmierung

Webseiten scrapen mit Python

Dr. Uwe Ziegenhagen

24. September 2014

Inhalt

Kapitelüberblick

Python (Wikipedia)

- entwickelt Anfang der 1990er Jahre von Guido van Rossum
- universelle, üblicherweise interpretierte, höhere Programmiersprache
- unterstützt objektorientierte, aspektorientierte und funktionale Programmierung
- hat eine klare und übersichtliche Syntax
- Entwurfsphilosophie betont **Programmlesbarkeit**
- mein Zugang zu Python über `save.tv` Downloader:[1]
  - „Kann ich ja gut lesen!“
  - „Sauberer Aufbau!“
  - „Klammern braucht man ja wirklich nicht!“

[1] `http://www.radekw.com/blog/2009/04/23/savetv-downloader/`

Python

Installation

- Python 2 versus Python 3
- unter Linux und Mac OS X dabei
- `apt-get install python-setuptools`
- unter Windows
  - Download von `https://www.python.org/downloads/`
  - PATH-Variable anpassen (auch das `Scripts` Verzeichnis)
  - Installation der `setuptools`, Download von `https://pypi.python.org/pypi/setuptools`, mit `python ez_setup.py` installieren
- dann `easy_install beautifulsoup4` für die Installation von BS4
- dann `easy_install urllib3` für die Installation der URLlib

```
1  a=1
2  b=10
3
4  print(a+b)
5
6  # von 1 bis 9
7  for i in range(a,b):
8    print(i)
9
10 def volumen(a,b,c):
11   return a*b*c
12
13 print("Volumen:",volumen(2,3,4))
```

- Beauti. . . was? ⇒ Python-Bibliothek, um aus HTML und XML Informationen zu extrahieren.
- unterstützt Python 2.6+ und Python 3
- benutzt standardmäßig eigenen XML/HTML-Parser, unterstützt aber noch weitere
- extrahiert nicht nur aus dem Parse-Tree, sondern kann diesen auch ändern

```
1  from bs4 import BeautifulSoup
2  import urllib3
3
4  http = urllib3.PoolManager()
5  r = http.request('GET', 'http://uweziegenhagen.de/fom/3.html')
6  r = r.data
7  soup = BeautifulSoup(r)
8
9  print(soup.prettify())
10 print(soup.get_text())
11
12 print(soup.title)
13 print(soup.title.string)
14
15 print("Extraktion von Links")
16
17 for link in soup.find_all('a'):
18     print(link.get('href'))
```

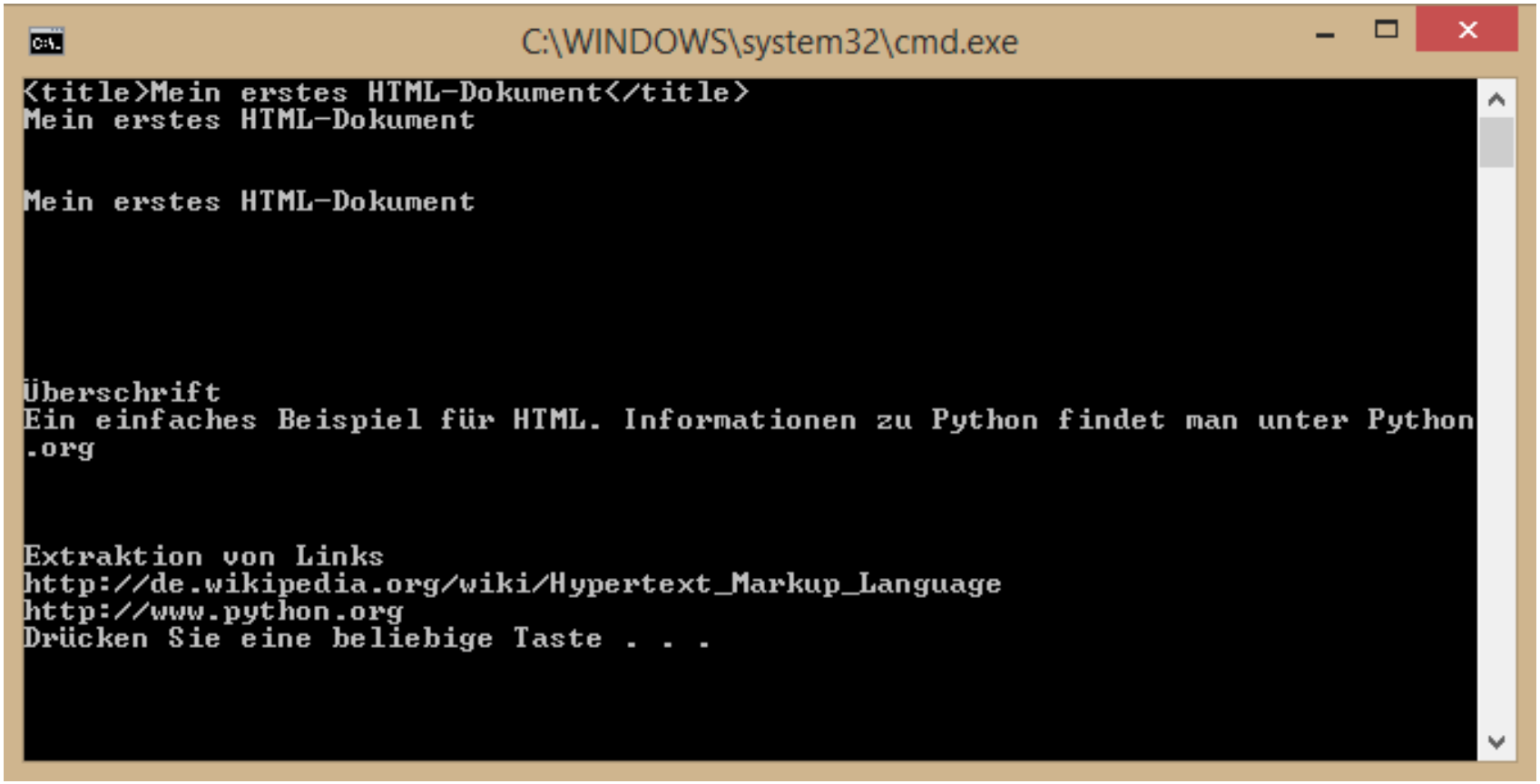

Abbildung: Ausgabe des BS4 Codes

Python

FMyLife mit BS4

```
1  # -*- coding: utf-8 -*-
2  from bs4 import BeautifulSoup
3  import urllib3
4  import sys
5
6  def textOf(soup):
7      return u''.join(soup.findAll(text=True))
8
9  http = urllib3.PoolManager()
10 r = http.request('GET', 'http://www.fmylife.com/?page=1')
11 r = r.data
12
13 soup = BeautifulSoup(r)
14
15 for item in soup.findAll('div', attrs={'class': 'post article'}):
16   item = textOf(item)
17   print(">",item[:item.find("FML#")])
```

Python

FMyLife mit BS4: Scraping der Startseite

```
1  # -*- coding: utf-8 -*-
2  from bs4 import BeautifulSoup
3  import urllib3
4  import sys
5  import codecs
6
7  # http://stackoverflow.com/questions/1752662/beautifulsoup-easy-way-to-
       to-obtain-html-free-contents
8  def textOf(soup):
9      return u''.join(soup.findAll(text=True))
10
11 http = urllib3.PoolManager()
12
13 with codecs.open("fmlife_20140906.txt", "w", "utf-8-sig") as ausgabe:
14   r = http.request('GET', 'http://www.fmylife.com/')
15   r = r.data
16   soup = BeautifulSoup(r)
17   for item in soup.findAll('div', attrs={'class': 'post article'}):
18     item = textOf(item)
19     ausgabe.write("\\item "+item[:item.find("FML#")]+"\n\n")
```

Python

FMyLife mit BS4: Scraping der anderen Seiten

```
1  # -*- coding: utf-8 -*-
2  from bs4 import BeautifulSoup
3  import urllib3
4  import codecs
5
6  def textOf(soup):
7      return u''.join(soup.findAll(text=True))
8
9  http = urllib3.PoolManager()
10
11 with codecs.open("fmlife_20140906.txt", "a", "utf-8-sig") as ausgabe:
12   for page in range(1,2132):
13     print(page)
14     r = http.request('GET', 'http://www.fmylife.com/?page='+str(page))
15     r = r.data
16     soup = BeautifulSoup(r)
17
18     for item in soup.findAll('div', attrs={'class': 'post article'}):
19       item = textOf(item)
20       ausgabe.write("\\item "+item[:item.find("FML#")]+"\n\n")
```

- Fragestellung: Wie entwickelt sich der Preis bei Amazon für einen bestimmten Artikel über die Zeit?
- Idee: Mit BS4 in regelmäßigen Abständen Preise extrahieren und wegschreiben (Datei, Datenbank)
- Ziel: Preis fällt unter bestimmte Grenze ⇒ Information per SMS/E-Mail
- Beispiel: Kopfhörer ATH-M50 von Audio Technica
- URL `http://www.amazon.de/dp/B000ULAP4U/`

Python

Amazon-Preise extrahieren

Python

Amazon-Preise extrahieren: Webseiten-Analyse mit Firebug

Python

Amazon-Preise extrahieren: Webseiten-Analyse mit Firebug

Python

Amazon-Preise extrahieren: BS4 Coding I

```
1  # -*- coding: utf-8 -*-
2  from bs4 import BeautifulSoup
3  import urllib3
4  import codecs
5
6  http = urllib3.PoolManager()
7
8  r = http.request('GET', 'http://www.amazon.de/dp/B000ULAP4U/')
9  r = r.data
10
11 soup = BeautifulSoup(r)
12
13 price = soup.find(id="priceblock_ourprice")
14 print(price)
```

⇒ Result `NONE`, scheinbar liefert Amazon andere Daten an Nicht-Browser aus

⇒ Let's fake a user agent!

User Agent

- optionaler Parameter im HTTP-Header
- definiert in RFC 2616
- Zweck
  - Statistische Auswertungen
  - Fehlersuche
  - unterschiedliche Auslieferung von Inhalten je Client-Typ

Beispiele

```
1 Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
2 Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.0
3 Mozilla/5.0 (compatible; MSIE 9.0; Windows Phone OS 7.5; Trident/5.0;
      IEMobile/9.0)
```

Python

Amazon-Preise extrahieren: BS4 Coding III

```
1  # -*- coding: utf-8 -*-
2  from bs4 import BeautifulSoup
3  import urllib3
4  import codecs
5
6  def textOf(soup):
7      return u''.join(soup.findAll(text=True))
8
9  http = urllib3.PoolManager()
10 header=urllib3.util.make_headers(keep_alive=True, user_agent="Mozilla
       /5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.0")
11
12 r = http.request('GET', 'http://www.amazon.de/dp/B000ULAP4U/',
13 headers=header)
14 r = r.data
15
16 soup = BeautifulSoup(r)
17 price = soup.find(id="priceblock_ourprice")
18 price = textOf(price)
19 print(price)
```

Python

Amazon-Preise extrahieren: Ergebnis

Abbildung: Ergebnis der Extraktion

⇒ Kann jetzt weiterverarbeitet und gespeichert werden.

By the Way...

Alternative über `wget` und `sed` I

- **wget** = Programm zum Herunterladen von Inhalten aus dem Internet, verfügbar für alle Plattformen
- mehr als 100 mögliche Kommandozeilenparameter
- einfachste Form `wget <URL>`
- Beispiel: „**wget www.amazon.de/dp/B000ULAP4U/**“ lädt HTML als „B000ULAP4U“ herunter
- Alternative: cURL (`http://de.wikipedia.org/wiki/CURL`), kann auch hochladen

By the Way...

Alternative über wget und sed II

```
1  E:\>wget http://www.amazon.de/dp/B000ULAP4U/
2  SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
3  syswgetrc = c:/progra~1/wget/etc/wgetrc
4  --2014-09-17 06:01:54-- http://www.amazon.de/dp/B000ULAP4U/
5  Resolving www.amazon.de... 178.236.7.219
6  Connecting to www.amazon.de|178.236.7.219|:80... connected.
7  HTTP request sent, awaiting response... 301 MovedPermanently
8  Location: http://www.amazon.de/Technica-ATH-M50-Professional-Studio-
       Kopfh%C3%B6r
9  er-schwarz/dp/B000ULAP4U [following]
10 --2014-09-17 06:01:54-- http://www.amazon.de/Technica-ATH-M50-
       Professional-Stud
11 io-Kopfh%C3%B6rer-schwarz/dp/B000ULAP4U
12 Connecting to www.amazon.de|178.236.7.219|:80... connected.
13 HTTP request sent, awaiting response... 200 OK
14 Length: unspecified [text/html]
15 Saving to: 'B000ULAP4U'
16
17     [   <=>                                 ] 303.395     314K/s   in 0,9s
```

Extraktion von Inhalten mit sed

Alternative über `wget` und `sed` II

- **sed** = ultimativer **S**tream **Ed**itor
- Standard unter Unix/Linux, auch für Win 32 erhältlich

sed -n -e 's/<Muster>/<Textneu>/' <Dateiname>

- **-e**: Befehle in Kommandozeile (siehe **-f** Option)
- **s/Muster/Textneu/** substitute **Muster** mit **Textneu**
- **<Dateiname>**: Datei, die durchsucht werden soll
- Beispiel: **satz.txt** mit folgendem Inhalt: „Die Kneipe am Ende der Strasse.“
- **sed -e "s/Kneipe/Lokalität/" satz.txt**
- Ergebnis: „Die Lokalität am Ende der Strasse.“

Extraktion von Inhalten mit sed

Alternative über `wget` und `sed` III

Extraktion des Preises aus der Webseite

- gesuchtes Stück:

```
1 <span id="priceblock_ourprice" class="a-size-medium a-color-
      price">
2 EUR 139,00
3 </span>
```

- Nutzung von sed mit Subpattern

Extraktion von Inhalten mit sed

Alternative über `wget` und `sed` IV

```
1 sed -n -e "s/\(<span id=\"priceblock_ourprice\" class=\"a-size-
    medium a-color-price\">EUR \)\([0-9,]*\)\(<\/span>\)/\2/p"
    B000ULAP4U
```

- **sed -n -e**: **-n** um alles zu unterdrücken, was nicht Pattern ist. **-e** Angabe des Patterns folgt
- 1. Subpattern für den **** Bereich
- 2. Subpattern für den Preis
- 3. Subpattern für **** Bereich
- **\2** für Rückgabe des zweiten Subpatterns
- **p** für die Ausgabe