Informatik I

21. Das WWW befragen

Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

10.01.2014

Motivation

Motivation

- Oft braucht ein Programm Informationen, die es im WWW einfach zu finden gibt.

Motivation

- Oft braucht ein Programm Informationen, die es im WWW einfach zu finden gibt.
- Dazu müsste man bloß kurz eine Webseite aufrufen und ein Detail nachschlagen.

Motivation

- Oft braucht ein Programm Informationen, die es im WWW einfach zu finden gibt.
- Dazu müsste man bloß kurz eine Webseite aufrufen und ein Detail nachschlagen.
- Zum Beispiel wollen wir die aktuelle Temperatur wissen.

Motivation

- Oft braucht ein Programm Informationen, die es im WWW einfach zu finden gibt.
- Dazu müsste man bloß kurz eine Webseite aufrufen und ein Detail nachschlagen.
- Zum Beispiel wollen wir die aktuelle Temperatur wissen.
- Könnte das nicht ein kleines Skript für uns tun?

Motivation

- Oft braucht ein Programm Informationen, die es im WWW einfach zu finden gibt.
- Dazu müsste man bloß kurz eine Webseite aufrufen und ein Detail nachschlagen.
- Zum Beispiel wollen wir die aktuelle Temperatur wissen.
- Könnte das nicht ein kleines Skript für uns tun?
- Auf `http://www.wetteronline.de` findet man die aktuelle Temperatur ziemlich weit oben auf der Seite.

Webseiten und HTML

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.
- HTML-Formatanweisungen kommen normalerweise in Paaren: z.B. `<h1>` und `</h1>` schließen eine Überschrift ein.

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.
- HTML-Formatanweisungen kommen normalerweise in Paaren: z.B. `<h1>` und `</h1>` schließen eine Überschrift ein.
- Generell wird eine öffnende Markierung <*mark*> durch eine schließende Markierung abgeschlossen: </*mark*>.

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.
- HTML-Formatanweisungen kommen normalerweise in Paaren: z.B. `<h1>` und `</h1>` schließen eine Überschrift ein.
- Generell wird eine öffnende Markierung <*mark*> durch eine schließende Markierung abgeschlossen: </*mark*>.
- Bei der öffnenden Markierung werden oft noch weitere Attribute angegeben, z.B. `<table border=''2''>`.

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.
- HTML-Formatanweisungen kommen normalerweise in Paaren: z.B. `<h1>` und `</h1>` schließen eine Überschrift ein.
- Generell wird eine öffnende Markierung <*mark*> durch eine schließende Markierung abgeschlossen: </*mark*>.
- Bei der öffnenden Markierung werden oft noch weitere Attribute angegeben, z.B. `<table border=''2''>`.
- Außerdem können die Dateien weitere Formatanweisungen (CSS) und aktive Komponenten (Javascript) enthalten.

HTML

- Alle Webseiten bestehen aus Texten (und Bildern) mit HTML-Formatanweisungen (*Hypertext markup language*).
- Die HTML-Anweisungen beschreiben, wie bestimmte Textteile erscheinen sollen.
- HTML-Formatanweisungen kommen normalerweise in Paaren: z.B. `<h1>` und `</h1>` schließen eine Überschrift ein.
- Generell wird eine öffnende Markierung <*mark*> durch eine schließende Markierung abgeschlossen: </*mark*>.
- Bei der öffnenden Markierung werden oft noch weitere Attribute angegeben, z.B. `<table border=''2''>`.
- Außerdem können die Dateien weitere Formatanweisungen (CSS) und aktive Komponenten (Javascript) enthalten.
- Eine gute Einführung findet sich z.B. auf `http://de.selfhtml.org/`.

Genereller Aufbau einer Webseite

HTML page

```
<!DOCTYPE html>
<html>
<head>
<meta ...>
</head>
<body>
...
</body>
</html>
```

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.
- Normalerweise findet man schnell ein Pattern, das zutreffend ist.

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.
- Normalerweise findet man schnell ein Pattern, das zutreffend ist.
- Schauen wir uns den Quellcode der `http://www.wetteronline.de/freiburg`-Seite an.

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.
- Normalerweise findet man schnell ein Pattern, das zutreffend ist.
- Schauen wir uns den Quellcode der `http://www.wetteronline.de/freiburg`-Seite an.
- Seite anwählen, dann rechts klicken und `Quelltext anschauen` wählen; ggfs. Text vorher markieren.

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.
- Normalerweise findet man schnell ein Pattern, das zutreffend ist.
- Schauen wir uns den Quellcode der `http://www.wetteronline.de/freiburg`-Seite an.
- Seite anwählen, dann rechts klicken und `Quelltext anschauen` wählen; ggfs. Text vorher markieren.
- Nach dem Text suchen.

Wie bekommt man die Information?

- Man kann sich den Quellcode der Webseite anschauen.
- Normalerweise findet man schnell ein Pattern, das zutreffend ist.
- Schauen wir uns den Quellcode der `http://www.wetteronline.de/freiburg`-Seite an.
- Seite anwählen, dann rechts klicken und `Quelltext anschauen` wählen; ggfs. Text vorher markieren.
- Nach dem Text suchen.
- Pattern konstruieren!

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.
- Bei uns ist folgende Zeile relevant:
  `<span ...id="current-weather"> aktuell 8&deg;C etwas Regen</span>`

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.
- Bei uns ist folgende Zeile relevant:
  `<span ...id="current-weather"> aktuell 8&deg;C etwas Regen</span>`
- Möglicher regulärer Ausruck:
  `r'<span[^>]*id="current-weather">\s+aktuell\s+(\d+)&deg;C'`

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.
- Bei uns ist folgende Zeile relevant:
  `<span ...id="current-weather"> aktuell 8&deg;C etwas Regen</span>`
- Möglicher regulärer Ausruck:
  `r'<span[^>]*id="current-weather">\s+aktuell\s+(\d+)&deg;C'`
- . . . zumindest solange wir keine negativen Temperaturen bekommen . . .

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.
- Bei uns ist folgende Zeile relevant:
  `<span ...id="current-weather"> aktuell 8&deg;C etwas Regen</span>`
- Möglicher regulärer Ausruck:
  `r'<span[^>]*id="current-weather">\s+aktuell\s+(\d+)&deg;C'`
- . . . zumindest solange wir keine negativen Temperaturen bekommen . . .
- Aber wie kommen wir an die Webseite?

Regulären Ausdruck konstruieren

- Am besten nach id=. . . *name* schauen, da diese eindeutig auf der HTML-Seite sind.
- Bei uns ist folgende Zeile relevant:
  `<span ...id="current-weather"> aktuell 8&deg;C etwas Regen</span>`
- Möglicher regulärer Ausruck:
  `r'<span[^>]*id="current-weather">\s+aktuell\s+(\d+)&deg;C'`
- . . . zumindest solange wir keine negativen Temperaturen bekommen . . .
- Aber wie kommen wir an die Webseite?

→ `urllib`

Das `urllib`-Paket

Das `urllib`-Paket

- Das `urllib`-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.

Das `urllib`-Paket

- Das `urllib`-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:

Das `urllib`-Paket

- Das `urllib`-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:
  - `urllib.request`: Enthält Funktionen und Klassen zum Zugriff auf Ressourcen im Internet.

Das `urllib`-Paket

- Das `urllib`-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:
  - `urllib.request`: Enthält Funktionen und Klassen zum Zugriff auf Ressourcen im Internet.
  - `urllib.parse`: Unterstützt das Parsen von URLs (*Universal Resource Locators*).

Das `urllib`-Paket

- Das `urllib`-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:
  - `urllib.request`: Enthält Funktionen und Klassen zum Zugriff auf Ressourcen im Internet.
  - `urllib.parse`: Unterstützt das Parsen von URLs (*Universal Resource Locators*).
- Die wichtigste Funktionen aus `urllib.request` ist:

Das urllib-Paket

- Das urllib-Paket bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:
  - urllib.request: Enthält Funktionen und Klassen zum Zugriff auf Ressourcen im Internet.
  - urllib.parse: Unterstützt das Parsen von URLs (*Universal Resource Locators*).
- Die wichtigste Funktionen aus urllib.request ist:
  - urlopen(url, data=None, timeout, *, cafile=None, capath=None, cadefault=False): Stellt ein Datei-ähnliches Objekt zur Verfügung. url ist die URL (HTTP/HTTPS-Adresse), auf die zugegriffen werden soll; data sind zusätzliche Daten, die bei einer Anfrage geschickt werden; timeout ist ein optionaler Parameter für eine obere Zeitschranke. Die anderen Parameter sind für Zertifikate (bei HTTPS).

Das `urllib`-Paket

- Das `urllib-Paket` bietet komfortable Schnittstellen, um auf Ressourcen im WWW zuzugreifen.
- Das Paket enthält mehrere Module:
  - `urllib.request`: Enthält Funktionen und Klassen zum Zugriff auf Ressourcen im Internet.
  - `urllib.parse`: Unterstützt das Parsen von URLs (*Universal Resource Locators*).
- Die wichtigste Funktionen aus `urllib.request` ist:
  - `urlopen(url, data=None, timeout, *, cafile=None, capath=None, cadefault=False)`: Stellt ein Datei-ähnliches Objekt zur Verfügung. `url` ist die URL (HTTP/HTTPS-Adresse), auf die zugegriffen werden soll; `data` sind zusätzliche Daten, die bei einer Anfrage geschickt werden; `timeout` ist ein optionaler Parameter für eine obere Zeitschranke. Die anderen Parameter sind für Zertifikate (bei HTTPS).
  - Nach `urlopen` kann man auf dem resultierenden Objekt `read`-Methoden anwenden und erhält bytes zurück.

Eine Webseite anschauen

wetter.py

```
from urllib.request import urlopen

showlines = 10
remotefile = urlopen("http://www.wetteronline.de/")
# method to get info about connection
print(remotefile.info())
# read all lines
remotedata = remotefile.readlines()
remotefile.close()
for line in remotedata[:showlines]:
    print(line)
```

Die Temperatur checken

temperature.py

```
from urllib.request import urlopen
import re

rf = urlopen("http://www.wetteronline.de/freiburg")
rd = rf.read().decode('utf8')
rf.close()
rx = re.compile(
   r'<span[^>]*id="current-weather">\s+aktuell\s+(\d+)&deg;C',
   re.I+re.M)
print("Die Temperatur beträgt zur Zeit",
      rx.search(remotedata).group(1),
      "Grad Celsius")
```

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?
- Aber Vorsicht:

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?
- Aber Vorsicht:
  - Webdesigner ändern gerne öfter mal das Seitenlayout.

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?
- Aber Vorsicht:
  - Webdesigner ändern gerne öfter mal das Seitenlayout.
  - Seitenbetreiber lieben das Scraping nicht, speziell wenn es zu starker Belastung des Webservers führt.

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?
- Aber Vorsicht:
  - Webdesigner ändern gerne öfter mal das Seitenlayout.
  - Seitenbetreiber lieben das Scraping nicht, speziell wenn es zu starker Belastung des Webservers führt.
  - Das umfangreiche Kopieren und auf eigener Webseite zur Verfügung stellen ist im Übrigen Missbrauch!

Web-Scraping

- Auf diese Weise, die man *Scraping* nennt, kann man beliebige interessante Informationen von Webseiten sammeln und z.B. per E-Mail verschicken.
- Zum Beispiel: Was gibt es heute in der Mensa?
- Aber Vorsicht:
  - Webdesigner ändern gerne öfter mal das Seitenlayout.
  - Seitenbetreiber lieben das Scraping nicht, speziell wenn es zu starker Belastung des Webservers führt.
  - Das umfangreiche Kopieren und auf eigener Webseite zur Verfügung stellen ist im Übrigen Missbrauch!
- Manche Seitenbetreiber bieten auch Webservices an, über die man dann per definierter Schnittstelle maschinenlesbar Daten bekommen kann (Beispiel: der Wetterservice `www.wunderground.com`).