LV 4342
Skriptsprachen-Praktikum
Übung 07

Arbeiten mit regulären Ausdrücken,

Aspekte nebenläufiger Programmierung

Mini-Projekt

Organisatorisches

- Arbeitsverzeichnis:

```
~/lv/skriptspr/07/
```

- Dateinamen:

```
07-spider.rb      # neu erstellen & abgeben
```

- Werkzeuge:

```
ruby              # Der Interpreter
emacs             # mit Ruby-Mode. Auch X-Emacs ok
scite, irb, ri    # Optionale Tools, wie üblich
                  # mit Ruby-Mode
komodo            # IDE für Ruby u.a. Skriptsprachen
```

Die Aufgabe

- Ein einfacher Web Crawler / Spider
  - Schreiben Sie ein Ruby-Programm, das einen URL von der Kommandozeile annimmt, die zugehörige (X)HTML-Datei einliest und analysiert – und alle gefundenen **Links** zu anderen HTML-Seiten **rekursiv weiterverfolgt**
  - Jede Seite soll nur einmal besucht werden
  - Beachten Sie Kommentare – auskommentierte Links *nicht* verfolgen!
  - Verfolgen Sie nur Anker-Elemente („a“) mit Verweisen auf andere (X)HTML-Seiten – keine anderen Elemente, keine anderen Dateitypen wie PDF, keine Javascript-Links
  - Ignorieren Sie „https“-Links u.a., verfolgen Sie nur das Schema „http“.
  - Sammeln Sie neben den Links weitere statistische Daten der besuchten Seite – Details siehe unten.

Details

- Hintergrund-Information
  - (X)HTML-Dokumente im WWW sind komplexe Textdaten
  - Ihre Analyse ist mit Hilfe von Regulären Ausdrücken besonders effizient möglich
  - Der Zugriff auf HTML-Daten via http ist mit Hilfe von Standardbibliotheken auf einfache Weise möglich.

- Aufruf (Beispiel)

```
$ ./07-spider.rb http://www.somehost.xy/
```

- Hinweise für das Abrufen von WWW-Seiten
  - Standard-Bibliothek „URI“ zum Parsen von URLs
  - Standard-Bibliothek „Net::HTTP“ zum Lesen von Web-Seiten
  - Infos unter http://www.ruby-doc.org/stdlib/

Details

- Abbruchbedingung
  - Wenn 1000 verschiedene Seiten besucht wurden, soll auf jeden Fall abgebrochen werden
  - Abbruch ferner bei Strg-C (Tipp: „rescue Interrupt“)
  - Beim Beenden immer (Tipp: „ensure“) ausgeben:

```
Anzahl gefundener / besuchter Seiten
Liste der besuchten URLs
```

- Ausgabe
  - Geben Sie zu jeder neu besuchten Seite deren URL und ein paar Zeilen mit den Statistik-Daten aus.
  - Damit erhalten Sie gleich auch einen Fortschrittsindikator

- Proxy
  - Beachten Sie den Proxy innerhalb der FH. Tipp: Es gibt Unterstützung für Proxies in der Bibliothek Net::HTTP !

Details / Statistik

- Ermitteln Sie zu jeder besuchten Seite
  - Die Anzahl Kommentare **<!-- … -->**
  - Die Anzahl Start-Tags, End-Tags und im Fall von XHTML von Empty-Element-Tags. Erinnerung:

```
<abc>                                   Start-Tag
<abc att1="value1" att2 = 'value2' >    … mit Attributen

</abc>                                  End-Tag

<br/>                                   Empty-Elem.-Tag
<abc att1="value1" att2 = 'value2'/>    … mit Attributen
```

  - Die Anzahl „a“-Links mit href-Attribut **Text**
  - Optional: Die Anzahl Worte im Nutztext. Tipp: \w genügt

Vereinfachungen

- Schleifenerkennung?
  - Reale Seiten können sich gegenseitig zitieren. Echte Spider erkennen dies, hier können wir aber darauf verzichten.

- Relative Links
  - Die Verarbeitung relativer Links ist etwas mühevoll. Sie können sich gerne daran versuchen, aber Auslassen relativer Links wie

    ```
    <a href="../next.html">Naechste Seite</a>
    ```

    ist auch in Ordnung.

Ergänzung für Woche 2

- Parallelisieren Sie Ihren Code
  - Die Verfolgung neuer Links soll parallel von separaten Threads ausgeführt werden, um die **I/O-Begrenzung** zu lindern.
  - Vorsicht: Die Zahl der Threads kann geometrisch wachsen und muss begrenzt werden!
  - Vermeiden Sie Kollisionen beim simultanen Aktualisieren der Liste der schon besuchten URLs mithilfe von Mutex / Locking, vgl. Vorlesung

- Lohnt sich‘s?
  - Erreichen Sie die Maximalzahl zu besuchender Seiten nun schneller?
  - Suchen Sie nun „breiter“, d.h. eine größere Vielfalt an URLs? Ggf.: Warum?

- **Freiwilliger Teil**
  - Diese Ergänzung ist freiwillig und wird nicht bewertet (aber begrüßt!)

Abgaberegeln

- Mini-Projekt
  - Die Aufgabe lässt Ihnen mehr Freiheiten als die bisherigen Übungen – nutzen Sie den Spielraum zum Experimentieren, ergänzen Sie die Aufgabe um Ziele, die Sie interessieren.
  - Es kommt nicht auf völlig korrekte Statistiken an, auch nicht auf das Aufsuchen wirklich aller verlinkten Seiten – viel wichtiger ist, Reguläre Ausdrücke zum Bewältigen realer Aufgaben mit „echten“ Daten einzusetzen.
- Abgabefrist
  - Diesmal zwei Wochen – analysieren Sie die HTML-Texte mit Regulären Ausdrücken für die Statistik in Woche 2, aber ermitteln Sie die Links bereits in Woche 1.
  - Elementare Spider-Funktionen (ohne Statistik) sind nach der ersten Woche bereits auf Nachfrage vorzuführen (auf dem FH-Rechner)!