Praktikum zu LV 7328 - Ruby:
Übung 07

Arbeiten mit regulären Ausdrücken

Organisatorisches

- Arbeitsverzeichnis:
  `~/lv/ruby/07/`
- Dateinamen:
  `07-registry.rb # kopieren, erweitern & abgeben`
- Werkzeuge:
```
ruby            # Der Interpreter
emacs           # mit Ruby-Mode. Auch X-Emacs ok
scite, irb, ri  # Optionale Tools, wie üblich
```
- Vorlagen:
  `07-registry.rb`

Organisatorisches

- Ablauf
  - Aufgabe 07 ist der erste Teil einer "Projektarbeit"
  - Weitere 1-2 Aufgaben werden folgen.
  - Quereinstieg ist möglich, da unabhängige Ziele verfolgt werden und jeweils Zwischenlösungen der früheren Aufgaben zur Verfügung stehen werden.
  - Derzeitiger Planungsstand:

```
07:    Scanner, mit regulären Ausdrücken
08:    Versch. Scripting, u.a. Datenbäume
xx:    Ein GUI-Browser für den Datenbaum
```

- Gegenstand / Ziel des Projekts:
  - Export/Import-Dateien der Windows Registry einlesen und in ein Datenmodell übertragen.

Hintergrund

- Ausgangspunkt
  - MS-Windows verwaltet eine große Zahl von Programm-einstellungen an zentraler Stelle in der so genannten "registry".
  - Mit "regedit" (Start/Ausführen/"regedit.exe") ist es u.a. möglich, Teile oder auch alle Einträge zu exportieren.
- Ziel
  - Ein (plattformunabhängiges) Ruby-Programm zur Verarbeitung derartiger Registry-Exportdateien.
- Weg
  - Teil 1: Entwickeln eines Scanners, der alle Zeilen der Exportdatei liest, in ihre Bestandteile zerlegt und daraus geeignete Datenobjekte anlegt.
  - Teil 2: Anordung dieser Objekte in einer Baumstruktur, Reproduktion des ursprünglichen Inhalts (Ausgabe)
  - Teil 3: Ein GUI-Browser für den Registry-Baum, basierend auf (2).

Hintergrund

- Methodik
  - Teil der Aufgabe ist es, sich in "fremden" Programmcode einzuarbeiten und diesen weiterzuentwickeln. Rechnen Sie daher auch Zeit dafür ein.
  - Kommentare in der Vorlage sind bewusst in Englisch gehalten, genauso wie die Vergabe von Klassen- und Methodennamen.
  - Diese Arbeitssituation ist typisch für spätere Projektarbeiten in der beruflichen Praxis - Üben bzw. Eingewöhnen lohnt sich also!
- Übergeordnetes Ziel
  - Skriptsprachen werden häufig für text-orientierte, plattform-übergreifende Aufgaben eingesetzt, häufig auch in der Rechneradministration - also wie in dieser Aufgabe.
  - Praxisrelevanz: Die Windows-Registry auf einem zentralen Unix-Server auszuwerten eröffnet den Administratoren interessante Möglichkeiten und ist daher eine praxisnahe Ruby-Aufgabe!

Vorbereitungen

- Vorlage kopieren
  - Damit die Aufgabe gut zu bewältigen ist, steht Ihnen eine Vorlage "`07-registry.rb`" im Dozentenverzeichnis zur Verfügung.
    >> Kopieren Sie diese Datei in Ihr Arbeitsverzeichnis. <<
    Ihre Aufgabe besteht i.w. in der Erweiterung dieser Datei.
- Eingabedaten präparieren
  - Für Ihre Tests stehen zwei Registry-Dateien bereit: "`klein.reg`" und "`alles.reg.bz2`".
  - Die Windows-Originale sind in UTF-16 codiert. Ihre Zeilenenden folgen der DOS/Windows-Konvention. Codieren Sie sie um:

```
$ iconv ...    # Von UTF-16 nach UTF-8 wandeln
$ dos2unix ... # Zeilenenden anpassen (für "diff" später)
```

  - Die genaue Bedienung von `iconv`, `dos2unix` und `bunzip2` bzw. `bzcat` entnehmen Sie bitte dem Unix-Manual.
  - "`alles.reg`" ist ca. 37 MB groß! Nutzen Sie die o.g. Entpacker.
  - Bei Platzproblemen verwenden Sie die fertigen (teils komprimierten) Dateiversionen des Dozenten ("klein-utf8.reg" und "alles-utf8.reg.bz2").
  - Testen Sie erst mit der großen Datei, wenn alles mit der kleinen gelingt!

Die Aufgabe

- A) Reguläre Ausdrücke entwickeln
  - Methode "scan" ist nur ein Rumpf. Derzeit wird nur eine (triviale) Eingabezeile erkannt. Entwickeln Sie für jede Zeilenart einen passenden regulären Ausdruck, sodass die Zeile nicht mehr in der Kategorie "*Unaccounted*" landet!
  - Beachten Sie, dass es Sonderfälle gibt. Ergänzen Sie Ruby-Code nach Bedarf, um auch diese zu berücksichtigen.
- B) Daten extrahieren, Datenmodell aufbauen
  - Die Programmvorlage enthält bereits vieles, was zum Aufbau eines Datenmodells erforderlich ist. Extrahieren Sie die Daten aus den Eingabezeilen und formen Sie daraus die für die Zeilen

    ```
    curr_node.store (..., RegType::....new(...) )
    ```

    noch ausstehenden Eingaben.
- C) Report erweitern
  - Beachten Sie die Zählerstände aus der Scan-Phase. Erzeugen Sie Angaben aus Ihrem Datenmodell! Sind beide Angaben konsistent?

Die Aufgabe

- D: Ziel für die Teile A-C, mit "klein-utf8.reg" zu ermitteln:
  - Wie viele Einträge ("nodes") hat Ihr Datenmodell (ohne "root")?
  - Wie viele "Hex"-Daten mit bzw. ohne explizite Datentyp-Angabe gab es?
  - Wie viele DWord-Daten lagen vor, und wie viele String-Angaben?
  - Wie viele "Hex"-Fortsetzungszeilen enthielt Ihre Datei?
- Bemerkung
  - Diese Angaben sollte Ihr Programm ausgeben. Ermittlung mit anderen Werkzeugen wie "wc" oder "grep" zählt nicht (liegt aber nahe zwecks Austesten Ihres Programms).
- E (*): Führen Sie (D) für die große Testdatei durch
  - Lösen Sie auch ggf. erst dann auftretenden neuen "Probleme".
  - Ermitteln Sie den Speicherbedarf von Ruby, z.B. indem Sie das Programm gegen Ende mit "sleep 60" eine Minute anhalten und in dieser Zeit mittels "top" die Spalte "SIZE" auslesen.

Details zu Regedit-Exporten

- Das Regedit-Format ist nicht offengelegt, aber leicht nachzuvollziehen:
  - Jeder Eintrag beginnt mit einer pfad-artigen ID-Zeile:
    ```
    [HKEY_LOCAL_MACHINE\HARDWARE\ACPI]
    ```
  - Darauf folgen optionale Datenzeilen zu diesem Eintrag
  - Er wird beendet durch eine Leerzeile
- Datenformate
  - Eine Datenzeile besteht aus einem key/value-Paar
  - Der "key" ist immer ein String (normalerweise mit "..." begrenzt)
    ```
    "mykey"=...
    Ausnahme: @=... (@ steht für den leeren String)
    ```
  - Es gibt drei Datentypen für "value":
    ```
    "mykey1"="somestring"      (String-Typ)
    "mykey2"=dword:00000002    (4-Byte-Wert)
    "mykey3"=hex:00,a0,ff,2b   (Bytesequenz, Default-Typ)
    ```


Details zu Regedit-Exporten

- Datentypen
  - Genau genommen unterstützt Windows eine große Zahl von Datentypen. Diese Datentypen werden mit einer 31-bit-Zahl verwaltet.
  - Non-Standard Datentypen werden im Hex-Format ausgegeben, wobei der numerische Datentyp als Parameter (hex-Darstellung ohne führende Nullen) erscheint:
    ```
    "mykey4"=hex(9):00,a0,ff,2b   (Bytesequenz, Typ 0x9)
    ```
- Abfolge der Einträge
  - Alle Einträge bilden eine **Baumstruktur**, wie ihre Pfade schon nahe legen.
  - Sie erscheinen in Reihenfolge „WLR“, Elternknoten gehen ihren Kindknoten also stets voran. Die Einträge erscheinen sortiert, was den Aufbau eines eigenen Datenbaums vereinfachen kann.
- Weitere Informationen z.B. in:
  ```
  http://web.cs.mun.ca/~michael/regutils/doc/regedit.html
  ```

Details zu Regedit-Exporten

- Beispiel-Einträge, aus der o.g. Quelle:

```
[HKEY_USER\network]
"RestoreDiskChecked"=dword:00000000
"RestorePrinterChecked"=dword:00000001
"FictitiousName"=hex:23,a3,f4

[HKEY_USER\network\Persistent]
@="some value"
```