



# Praktikum zu LV 7328 - Ruby:

## Übung 07

Projektarbeit, Teil 2  
Ausgabeformatierung,  
Umgang mit Datenbäumen



## Organisatorisches



- Arbeitsverzeichnis:  
`~/kurse/ruby/07/`
- Dateinamen:  
`07-registry.rb` # kopieren, erweitern & abgeben
- Werkzeuge:  
`ruby` # Der Interpreter  
`emacs` # mit Ruby-Mode. NICHT X-Emacs  
`scite` # Ein portabler Editor, auch  
# mit Ruby-Mode
- Vorlagen:  
`07-registry.rb`



- Ablauf
  - Aufgabe 07 ist der zweite Teil einer "Projektarbeit"
  - Eine weitere Aufgabe soll noch folgen.
  - Quereinstieg ist möglich, da unabhängige Ziele verfolgt werden und jeweils Zwischenlösungen der früheren Aufgaben zur Verfügung stehen werden.
  - Derzeitiger Planungsstand:
    - 06: Scanner, mit regulären Ausdrücken
    - 07: 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.



- Siehe gleichnamige Abschnitte der Aufgabe 06.



## Vorbereitungen



- Vorlage kopieren
  - Sie können mit Ihrer Datei "06-registry.rb" weiterarbeiten, indem Sie diese unter "07-registry.rb" in den neuen Ordner kopieren.
  - Falls Sie keine befriedigende Lösung bei Aufgabe 06 erreichten, steht Ihnen eine Vorlage unter "07-registry.rb" im Dozentenverzeichnis zur Verfügung.
  - Ihre Aufgabe besteht i.w. in der Erweiterung dieser Datei.
- Eingabedaten
  - Für Ihre Tests verwenden Sie dieselben Dateien "klein.reg" und "alles.reg.bz2" wie in Aufgabe 06 bzw. deren mit `iconv`, `dos2unix` und `bunzip2` bzw. `bzcat` angepassten Versionen "klein-utf8.reg" und "alles-utf8.reg.bz2".
  - Erneut: Testen Sie erst mit der großen Datei, wenn alles mit der kleinen gelingt!



## Die Aufgabe



### A: Ausgabe der eingelesenen Daten

- Implementieren Sie Methoden `to_s` für die Klassen `RegType::Hex`, `RegType::DWord`, `RegType::String`, `RegNode`
- Geben Sie am Programmende jeden Knoten dann einfach mittels "puts" aus (welches implizit `RegNode#to_s` verwendet), indem Sie über jeden Knoten von "root" iterieren - analog zum Zählen der Knoten zuvor.
- Testen Sie das Ergebnis:

```
$ 07-registry klein-utf8.reg > tmp1
$ diff tmp1 klein-utf8.reg | more
```
- Die Daten sollten inhaltlich übereinstimmen. Die Ausgabe langer Hex-Ketten in einer Zeile wird akzeptiert.



## Die Aufgabe



B (\*): Ausgabe der eingelesenen Daten, Forts.

- Erreichen Sie nun exakte Übereinstimmung bei "diff"
- Ausnahme:  
Die Kopfzeilen "Windows Registry Editor ..." + Leerzeile dürfen fehlen.

C (\*): Große Testdatei:

- 1) Gelingt der vorher/nachher-Vergleich auch hier?
- 2) Bestimmen Sie die Laufzeiten
  - für den scan-Teil und
  - für den Ausgabeteil des Programms  
bei Anwendung auf "alles-utf8.reg"



## Die Aufgabe



D: Entwickeln Sie die Methode "RegNode#add" weiter:

- Sie soll nun einen echten Datenbaum erzeugen, der die Struktur der Pfade widerspiegelt.
- Optional: Optimieren Sie Ihren "add"-Algorithmus, so dass er "alles-rtf8.reg" innerhalb weniger Minuten verarbeitet.

- Hinweise

- Generell: Sie können die vorgegebenen Klassen auch erweitern und umgestalten, wenn Ihr Lösungsweg dies erfordert - die Vorlage ist als Hilfe gedacht, nicht als "Beengung".
- Zu A: Beim Rekonstruieren des Hex-Formats haben sich String#unpack und Array#collect bewährt!
- Zu D: Ergänzen Sie bei Bedarf weitere Methoden zu RegNode, wenn dadurch "add" einfacher zu implementieren ist.