

## Übungsblatt #06

Bearbeitungszeitraum: bis Donnerstag, 03.12.2009

*Was will uns dieses Übungsblatt sagen?*

1. Die Stärken von Perl liegen im Umgang mit Textdateien.
2. Eine typische Art von Problemstellung, für die Perl sehr gut geeignet ist, schauen wir uns mit diesem Übungsblatt an, indem wir mit Hilfe durch ein Perl-Skript eine Transformation von Textdateien von einem Quellformat in ein Zielformat vornehmen lassen.
3. Dabei soll die Aufgabenstellung nur als Beispiel verstanden werden: zum Umgang mit Datenbank-Ausgaben stehen in Perl natürlich deutlich bessere Möglichkeiten bereit.
4. Die regulären Ausdrücke, die wir beim `sed` kennengelernt haben, kommen in nahezu identischer Syntax auch in Perl zur Anwendung.
5. Listen, als ein Datentyp in Perl, sind ein sehr mächtiges Werkzeug im Umgang mit Daten. Mit nur minimalem Code lassen sich sehr komplexe Operationen darauf ausführen. Ihr könnt Euch ja mal überlegen, wieviel „Schreibarbeit“ für diese Aufgabe in Java, C++ oder einer ähnlichen Programmiersprache nötig wären.

### Erste Schritte mit Perl

Ihr erinnert Euch doch sicher noch an die SQL-Aufgaben von Übungsblatt Nr. 1, oder? Damals haben wir den MySQL-Client kennengelernt und verwendet.

Jetzt, wo Ihr von den Unix-Kommandos „`join`“, „`cut`“, „`sort`“ usw. so richtig begeistert seid, habt Ihr beschlossen, fortan nur noch CSV-Dateien zu benutzen. Wozu braucht man schliesslich einen Datenbank-Server à la MySQL wenn man Unix kennt? Das einzige Problem, das Ihr im Moment habt, ist allerdings die Frage, wie Ihr aus den MySQL-Tabellen schöne CSV-Dateien machen könnt.

„Halt!“, denkt die Scriptsprachlerin in Euch, „Ich kann doch nicht nur MySQL und Shell-Programmierung, sondern neuerdings auch Perl! Und steht Perl nicht für *Practical Extraction and Report Language*? Ich will doch etwas extrahieren... Da wäre es doch gelacht, wenn das nicht mit Perl ganz einfach ginge!“

Tja, und so kommt es, dass Ihr nun vor der Aufgabe sitzt, die MySQL-Ausgaben vom ersten Übungsblatt (die stehen auf unserer Vorlesungshomepage) mit Hil-

fe eines Perl-Skripts in CSV-Dateien übersetzen lassen zu wollen.

Zwar gibt es für den Umgang mit Datenbanken unter Perl sehr schöne Module, und eines davon werden wir demnächst auch kennenlernen. Aber für heute brauchen wir die noch nicht! Denn diese Aufgabe soll ein Paradebeispiel für einen Einsatzzweck von Perl in seiner „puren“ Form, also ohne weitere Hilfsmittel sein.

Irgendwie verstehe ich nicht,  
was ich machen soll...

Okay, dann nochmal etwas präziser: Auf der Homepage findet Ihr einige Dateien mit der Endung „`sql`“. Sie enthalten die Ausgaben einiger „`select * from . . .`“-Abfragen der Datenbank, die wir im Zusammenhang mit „Übungsblatt 1“ verwendet haben. Der Inhalt dieser Datei soll nun in eine CSV-Datei umformatiert werden, d.h. er soll durch ein Perl-Skript dergestalt „verarztet“ werden, dass er danach so aussieht, wie derjenige der Dateien mit der Endung „`csv`“, die ebenfalls auf der Homepage stehen.

Das ganze soll ausschließlich mit den Perl-Kenntnissen erledigt werden, die Ihr bislang bei uns erworben habt, also OHNE spezielle Datenbank-Module. Letztlich geht es bei dieser Aufgabe lediglich darum, Perl als mächtiges Werkzeug für die effiziente Bearbeitung von Textdateien kennenzulernen.

## Die Aufgabenstellung in einem Satz:

*Schreibt ein Perl-Skript, das eine MySQL-Ausgabe einliest und daraus eine CSV-Datei erstellt.*

## Ein paar kleine Hinweise noch:

- Euer Perl-Skript soll so flexibel wie möglich sein, das heisst, es soll mit allen SQL-Ausgaben, die wir auf der Homepage verlinkt haben, ohne spezielle Anpassungen zurecht kommen.
- Schön wäre es auch, wenn das Trennzeichen, das Ihr in Eurer CSV-Datei verwenden möchtet, als Parameter angegeben werden kann.
- Als Tip sei angemerkt, dass hier Listen sehr hilfreich sein könn(t)en. Wenn Ihr ganz genau schaut, findet Ihr sicher ein Zeichen, das sich als Trennsymbol für ein „split“ geradezu aufdrängt...
- Was wäre ein Übungsblatt ohne hinterlistige Fallen? Deshalb haben wir auch eine eingebaut. Ein kleiner Wink mit dem Zaunpfahl hierzu: denkt daran, dass man manchen Zeichen durch einen vorangestellten Backslash die Sonderbedeutung nehmen muss. Falls Ihr also mit regulären Ausdrücken arbeitet und nicht das gewünschte Ergebnis bekommt, denkt mal darüber nach, ob Euch nicht vielleicht ein solches Zeichen das Leben schwer macht.