Haskell-Überlebensratgeber

Henning Thielemann

2013-04-13

Viel wurde darüber geschrieben, wie toll man mit Haskell Probleme lösen kann und welche fortschrittlichen Eigenschaften Haskell besitzt. Die Foren und E-Mail-Verteiler sind gefüllt mit Ratschlägen, wie man zu noch höherer Glückseligkeit gelangen kann, wenn man nur möglichst viele GHC-Erweiterungen einschaltet. Nicht zuletzt wird in wissenschaftlichen Artikeln ausschweifend über die GHC-Erweiterungen der Zukunft philosophiert.

Die Fülle an verfügbaren Techniken erschlägt jeden Einsteiger und fordert selbst Fortgeschrittene. Mein Tutorium soll eine Bresche durch unwegsames Gelände schlagen: Mit welchen Lösungen werde ich morgen noch zufrieden sein? Wie mache ich meine Haskell-Programme sicher, wartbar, portierbar und für andere Programmierer verständlich? Was sollte ich auf jeden Fall vermeiden?

Folgende Themen möchte ich behandeln:

- Wie finde ich einen guten Namen für eine Variable oder eine Funktion?
- Lazy/non-strict, pure/impure, safe/unsafe, total/partial – Ist mir alles klar! Oder doch nicht?
- `throw`, `error`, `ErrorT`-Monad – Was brauche ich wofür?
- Wie entwerfe ich Typen sinnvoll?
- Wie entwerfe ich Klassen und wie organisiere ich deren Ausprägungen?
- Welche GHC-Erweiterungen brauche ich wirklich?
- Wofür darf ich unsafePerformIO verwenden?
- Welche Hackage-Pakete sind gut für mich und welche nicht?
- Wie verwende ich Kommentare und Haddock-Dokumentation sinnvoll?
- Wie spielen Versionsverwaltung und Cabal sinnvoll zusammen? Wie wähle ich Paketversionen und was bedeutet die Package-Versioning-Policy für mich?

Aufgebaut ist das Tutorium aus einer Reihe von Fragen und typischen Problemen, die ich gemeinsam mit den Teilnehmern diskutieren will. Dabei gibt es nicht immer ein Richtig oder Falsch, sondern mitunter nur einen Anstoß von mir, über das eine oder andere in Zukunft genauer nachzudenken. Ich werde die Themen nicht alle erschöpfend behandeln können, sondern mich danach richten, was den Teilnehmern und mir besonders wichtig erscheint. Die Themen, die ich nicht behandeln kann, können sich die Teilnehmer später anhand meiner Folien selber erarbeiten.

Das Tutorium richtet sich an praktizierende Haskell-Programmierer, nicht an Einsteiger, sondern an Gelegenheitsprogrammierer und fortgeschrittene Haskell-Programmierer. Ich denke aber, dass auch Experten die eine oder andere Anregung mitnehmen können.