

Real World Haskell

Tobias Höppner

SoSe 2013

Inhaltsverzeichnis

1	VL I	2
1.1	Motivation	2
1.2	Was passiert hier?! - der kleine Webserver	2
1.2.1	der kleine Webserver	2
1.2.2	Einbinden von Modulen	2
1.2.3	Do-Notation	2
1.2.4	\$-Operator	3
1.2.5	!!-Operator	3
1.3	der größere Webserver	3
1.4	builds	3
1.4.1	mit ghc	3
1.4.2	mit cabal	3
1.5	Generics in Haskell	4
1.6	Stdlib - System.IO	4
1.7	Stdlib - System.Environment	4
1.8	Kommentare und Haddock	4
1.9	Keywords	4

Kapitel 1

VL I

1.1 Motivation

Warum eigentlich Haskell?

Haskell Compiler ist mächtig. Weil die Semantik und Typsystem wilde Sachen erlaubt. Wilde Sachen ermöglichen korrekte Software und sind meist sogar effizienter.

1.2 Was passiert hier?! - der kleine Webserver

1.2.1 der kleine Webserver

Was nicht behandelt wurde:

- Fehlerfälle, Exceptions Haskell unterstützt Exceptions
- Effizienz

1.2.2 Einbinden von Modulen

import am Anfang der Datei

- System.IO
- Control.Monad (forever)
- Text.Printf
- Network
- Control.Exception
- Control.Concurrent

1.2.3 Do-Notation

```
1 main = do
2   putStrLn "hallo _user !! "
```

```

3  putStrLn "xxxx"
4  main = p "x" >> p "x"

    ist das gleiche wie

1  main :: IO()
2  main = do
3    args <- getArgs
4    read ((!!0) args)
5    let x = read ((!! 1) args)

```

Typen

listenOn: $_ \leftarrow IO _$

1.2.4 \$-Operator

```

1  f a b

    a ist eine Fkt. g x k
    b ist eine Fkt. k fv

1  f g x k k f v

```

1.2.5 !!-Operator

Gibt das angegebene Element aus der Liste zurück.

```

1  (!!) :: [a] -> Int -> a
2  let xs = []
3  ys = [1,2,4]
4  zs = [1..1378]
5
6  zs !! 0

```

1.3 der größere Webserver

1.4 builds

1.4.1 mit ghc

```

1  ghc x.hs

```

Wird unübersichtlich für mehrere Dateien / Module.

1.4.2 mit cabal

```

1  cabal configure
2  cabal build
3  cabal install
4  \end{listlisting}

```

```

5
6 Projekte werden als \textbf{.cabal} gespeichert, sind
  eleganter und man kann schneller testen.
7
8 \section{.(Punkt)-Operator}
9 \begin{lstlisting}
10 (.) :: (b -> c) -> (a -> b) -> (a -> c)
11 f . g
    entspricht

1 (\x -> f ( g x))

```

1.5 Generics in Haskell

```

1 List e
2 m k v

```

1.6 Stdlib - System.IO

- Textinput / Textoutput
 - Print
 - getLine
 - getChar

1.7 Stdlib - System.Environment

- getArgs

1.8 Kommentare und Haddock

```

1 — einfacher Kommentar
2 {- mehrzeiliger Kommentar -}
3 {- mehrzeiliger Kommentar
4   {- verschachtelter Kommentar -}
5 -}
6 — | haddock kommentar

```

1.9 Keywords

Programming Guidelines sind brauchbar

```

1 main = do
2   args <- getArgs
3   case args of
4     [] -> ...

```

```
5      [ "-x" ] -> ...
6      [ "-x", b ] -> ...
```