CGI Programmierung mit **Ha**

Markus Schwarz

Überblick

- Was ist funktionale Programmierung
- Einführung in **Haskell**
- CGI-Programmierung mit **Haskell**
- Ein *etwas* größeres Beispiel

Was ist funktionale Programm

- Ein Programm ist eine Funktion
- Es gibt keine Zuweisungen
- Keine Seiteneffenkte

$\rightarrow$ kürzerer und besser wartbarer Code

Einführung in **Haskell**

- Funktionen
- Typen
- Listen
- Funktionen höherer Ordnung

Funktionen in **Haskell**

- Verknüpfung einer Definition mit einem Bezeich
- Bezeichner beginnen mit kleinem Buchstaben
- Bestehen meist aus einer Deklaration und einem
- Eine Funktionendeklaration hat beliebig viele fo
  meter und genau einen Typ

Beispiele für Funktionen

```
antwort :: Int
antwort = 42

inc :: Int -> Int
inc n = n + 1

fac :: Int -> Int
fac 0 = 1
fac n = n * fac (n - 1)
```

Typen

- Vordefinierte Datentypen (`Char`, `Int`, `Bool`, ...)
- Synonyme
- Algebraische Datentypen
- Klassen
- Im Gegensatz zu `C` ist **Haskell** streng typisiert

Datentypen (1)

- Synonyme

```
type Name = String
type Alter = Int
type Person = (Name, Alter)
```

- Auzählungstypen

```
data Monat = Jan | Feb | Mar | Apr | Mai | Jun
             Aug | Sep | Okt | Nov | Dez
```

Datentypen (2)

- Produkttypen

  ```
  data Person = Teilnehmer Name Alter
  ```

  - Es können beliebig viele Konstruktoren erzeu
  - Können nur durch Konstruktor erzeugt werde
  - Steigern die Lesbarkeit des Programmes

Listen in **Haskell**

- Elementarer Bestandteil der Sprache
- Definition einer Liste durch eckige Klammern.
- Elemente sind durch Kommata voneinander get
- Viele polymorophe Listenfunktionen

```
(++) :: [a] -> [a] -> [a]
(!!) :: [a] -> Int -> a
length :: [a] -> Int
```

Funktionen höherer Ordnu

- Funktionen als Parameter und/oder Rückgabew

- Umwandlung einer Liste von Integern in eine Liste
  mit Hilfe von `map`

```
map :: (a -> b) -> [a] -> [b]
alsString :: Int -> String
...
map alsString [1, 2, 42]
=> ["1", "2", "42"]
```

λ–Notation

- Definition anonymer Funktionen

- Definition dort, wo sie benutzt werden

```
map (\n -> n + 1) [1, 2, 3, 4]
```

CGI Programmierung mit **Ha**

- Hello World
- Abstraktion durch eine CGI Bibliothek
- Modellierung von HTML

Hello World

```
(1)   #! /usr/bin/runhugs

(2)   main :: IO()
(3)   main = putStr ("Content-type: text/html\n\n"
        ++ "<html>"
        ++ "<head><title>HelloWorld</title></head>
        ++ "<body><h1>Hello World</h1></body>"
        ++ "</html>")
```

Die CGI Bibliothek von Erik

Vorteile der Abstraktion

- Details des HTTP sind völlig transparent
- Dekodieren der Parameter erfolgt automatisch
- Lesbarere Skripte, da nur die für die Problemlösung relevanten Bestandteile enthalten sind.
- Bessere Möglichkeiten der Portabilität

Modellierung von HTML

```
data HTML
  = Text String
  | Element Tag [(Name, Value)] [HTML]

page :: String -> [(Name,Value)] -> [HTML] ->

h :: Int ->String -> HTML
hr :: HTML

ul :: [[HTML]] -> HTML
dl :: [(String,[HTML])] -> HTML

href :: URL -> [HTML] -> HTML
table :: String -> [String] -> [[ [HTML] ]] ->
```

Hello World (2)

- Anstatt direkt auf `stdout` zu schreiben, wird di
  benutzt.

- Modellieren der Seite mit Hilfe von Funktionen

```
(1) import CGI

    hello :: HTML
(2) hello = page "Hello World" [] [h1 "Hello W

(3) main :: IO ()
(4) main = wrapper (\env -> do return (Content
```

Umgebungsvariablen und Para

- Werden automatisch extrahiert und als eine Liste
  Wert Paaren übergeben

- Indizierter Zugriff ist möglich

  ```
  lookup "name" env
  ```

- Beispiel: Seite mit einer Liste alles Umgebungsv
  Parameter

Beispiel: Umgebungsvariab

```
seite :: [(Name, Value)] -> HTML
seite env = page "Umgebungsvariablen" []
  [
    h1 "Umgebungsvariablen",
    dl (map (\(dt,dd) -> (dt, [prose dd])) env)
  ]

main :: IO ()
main = wrapper(\env -> do return (Content(seite en
```

Fallbeispiel: Damenproble

- Wie lassen sich n Damen auf einem n*n Schach
  nen, ohne daß sie sich bedrohen?

- Schwerpunkt liegt auf der Modellierung von HT
  - Formular
  - Ausgabe

Das Formular

- Folgende Parameter sollen einstellbar sein
  - Die Anzahl der Damen (Listbox)
  - Welche Lösung soll angezeit werden (Freitex
  - Soll die Tabelle einen Rand haben oder nicht (
  - Welches Zeichen (Auswahlliste)
- Knopf zum Löschen der Eingaben
- Knopf zum Abschicken

Fazit

- **Haskell** ist für den Einsatz als CGI-Sprache gee
- Mit Hilfe von Biblotheken läßt sich die Arbeit ve
  - Kodieren / Dekodieren von Request und Res
  - Erzeugen von HTML
- Ungeeignet für Seiten mit viel Design

Vielen Dank

- Fragen
- Anmerkungen
- Kritik