Praktische Informatik 3 WS 20/21

Christoph Lüth
Thomas Barkowsky
Tobias Brandt
Alexander Krug
Tarek Soliman
Robert Sachtleben

# 9. Übungsblatt

**Ausgabe:** 01.02.2021
**Abgabe:** 15.02.2021 12:00

Nachdem die Expansion der Kaffeehäuser des *Happy Coffeecup* mehr als erfolgreich verlief und auch die Transportwege des Kaffees in die einzelnen Kaffeehäuser optimiert wurde, gilt es nun, wie es sich für jedes gute Wirtschaftsunternehmen gehört, den maximalen Profit aus dieser Unternehmung zu schlagen. Da passt es nur allzu gut, dass das Rechnungswesen errechnet hat, dass das Unternehmen 3% mehr Steuern sparen könnte, wenn es den Latte Macchiato nicht direkt an den Endverbraucher verkauft, sondern den Kaffee an die einzelnen lokalen Kaffeehäuser absetzt. Deshalb soll das Unternehmen jetzt *verfranchised* werden.

## 9.1 *Kaffee im WWW* — *7 Punkte*

Um die einzelnen Läden des Franchise bekannt zu machen, möchte der Chef jedem Franchisenehmer eine Art *Happy Coffeecup Homepage Baukasten (HCHB)* zur Verfügung stellen (für den der Franchisenehmer natürlich auch noch bezahlen muss). Der alteingesessene Webentwickler des Happy Coffeecup hat vor Jahren schon einmal eine Webseite für das Unternehmen gebastelt[1]. Die Seite ist jedoch etwas in die Jahre gekommen und in statischem HTML geschrieben. Sie werden daher beauftragt, die Webseite *franchise-ready* (Wortlaut vom Chef) zu machen und auf ein modernes Framework zu portieren. Da Sie wissen, dass HMTL sich besonders gut in Haskell als DSL integrieren lässt, wählen Sie für diesen Job natürlich Haskell.

Die Webseite soll sich modular aus einzelnen Komponenten zusammensetzen lassen. Implementieren Sie dafür zunächst in `Website.hs` einen Kombinator `website :: String → String → Html → Html`.

`website title subtitle content` repräsentiert eine komplett gültige Instanz einer *Happy Coffeecup*-Webseite (inklusive Doctype und Metadaten) mit dem Titel und Untertitel `title` bzw. `subtitle` und dem Inhalt `content`, sodass der eigentliche Inhalt einer Webseite durch `content` repräsentiert ist. Das Layout und der Stil der Webseite soll identisch zu der alten Webseite sein. Denken Sie daran, dass Links möglichst nicht statisch kodiert werden sollten. Für Hamlet-Dateien existiert hier das Makro `@{C}`, wobei `C :: Route` ein Konstruktor ist, der einen Inhalt auf dem Server repräsentiert und dessen URL über einen Renderer definiert ist[2].

Um den Kombinator zu testen, modellieren Sie den Inhalt der alten `index.html` nach und übergeben Sie ihm `website`. Anschließend *servieren* Sie die gesamte Webseite unter dem Wurzelverzeichnis, indem Sie in `Main.hs` den Funktion `main` modifizieren. Mit dem Befehlt `stack run` aus dem Projektverzeichnis heraus, kann der Server gestartet werden und die Webseite dann unter `http://localhost:3000/` bewundert werden.

## 9.2 *Social Media* — *6 Punkte*

Beflügelt von diesem ersten Entwicklungserfolg gehen Sie zu Ihrem Chef, um Ihn zu überzeugen soziale Medien wie *TicToc* oder *Clubhouse*, in den *HCHB* zu integrieren. Der Chef ist begeistert; möchte aber natürlich (!) wieder Kosten sparen. Deshalb erteilt er Ihren Social Media Ambitionen erst einmal eine Absage. Er möchte aber trotzdem ein wenig mehr digitale Interaktion mit seinen koffeinsüchtigen Kunden - über die er aber die volle Kontrolle hat! Er verdonnert Sie also eine etwas kostensparendere und restriktivere[3] Art von Social Media zu implementieren: Ein Gästebuch.

Entwickeln Sie deshalb zunächst eine Funktion `gbContent :: GuestBook → Html`, die Sie später `website` übergeben können. Die Gästebuch-Seite soll das Feedback aller User mit ihrem Namen und einem Kommentar darstellen. Unter den Kommentaren soll es ebenso ein Formular geben, welches die zuvor genannten Daten entgegennimmt. Das Formular soll via `POST`-Request die Daten an einen von Ihnen gewählten Endpunkt leiten. Im Order `old` finden Sie eine statische Vorlage für das Gästebuch an der Sie sich orientieren können.

Anschließend müssen Sie `Route` einen neuen Konstruktor `Guestbook` hinzufügen, der das Gästebuch repräsentiert, sowie in `renderUrl` diesem Konstruktor zu einem neuen Pfad für das Gästebuch zeigen lassen, sodass Sie in Ihren Hamlet-Datei `@{Guestbook}` verwenden können.

[1] Zu finden im Ordner `old`
[2] In diesem Projekt heißt der Renderer renderUrl
[3] sowie etwas in die Jahre gekommende



Version 1.0 vom 24.01.2021

Um das Gästebuch persistent[4] für alle User des Servers zu halten, benötigt das Programm irgendeine Art von persistenten Speicher. Es ist Ihnen überlassen wie Sie diesen realisieren, empfohlen wird aber z.B. vordefinierte Lösungen wie `MVar` zu nutzen.

Modifizieren Sie also die `main`-Funktion so, dass Sie vor Serverstart einen Speicher für das Gästebuch initialisiert. Fügen Sie dann zusätzlich dem Server Endpunkte für das Gästebuch hinzu. Einen `GET` Endpunkt, der die URL für das Gästebuch bestimmt[5], und einen `POST` Endpunkt, der die Daten des Formulars entgegennimmt, das Gästebuch um den neuen Gästebucheintrag erweitert und anschließend wieder auf das Gästebuch weiterleitet.

## 9.3 *Rechnung, bitte!* *7 Punkte*

Das Gästebuch steht also. Sehr gut. Nun fehlt nur noch eine Möglichkeit wie die Kunden auch online ihren *Chai Latte* ordern können. Dafür werden zwei weitere Webseiteninhalte benötigt: eine für die Auswahl der zu bestellenden Getränke und eine Seite, die dem Kunden die Rechnung seiner bestellten Getränke anzeigt.

Erstellen Sie beide nötigen Webseiten inklusive der notwendigen Endpunkte und Funktionalität auf dem Server. Die Rechnungsseite soll den Preis für jedes Getränk, sowie den Gesamtpreis in einer Tabelle auszeichnen. Beachten Sie dabei, dass die Tabelle nur Getränke beinhalten soll, die auch tatsächlich bestellt worden sind. Im Ordner `old` finden Sie ebenfalls statische Versionen des Bestellvorgangs an denen Sie sich orientieren können. Datentypen und Funktionen, die für die Bestellung nützlich sein könnten, stehen bereits im Modul `Types` bereit.

*Hinweise:*

1. In diesem Übungsblatt geht es vor allem darum, dass Sie sich in fremde Bibliotheken einarbeiten können. Die Bibliothek des Servers ist unter `https://hackage.haskell.org/package/scotty-0.12` zu finden. `Hamlet`, die HTML-DSL, ist Teil von `shakespeare` und ist unter `https://hackage.haskell.org/package/shakespeare-2.0.25` zu finden. Die Übersichtsseite verlinkt ebenfalls einen Übersichtsartikel für `Hamlet`-Templates. Der `Html`-Typ wird von der Bibliothek `blaze-html` definiert; zu finden unter: `https://hackage.haskell.org/package/blaze-html-0.9.1.2`.
2. Statische Daten wie Bilder und CSS-Dateien finden sie in dem Ordner `static`. All diese Dateien werden vom Server bereits auf den angegebenen Pfaden bereitgestellt.
3. Bei der Implementierung der Persistenz des Gästebuchs kann es nützlich sein, innerhalb der Actionmonade beliebige IO-Aktionen auszuführen. Dies ist mit der Funktion `liftIO :: IO a → ActionM a` möglich.
4. Sollten Sie es für nötig halten, optische Änderungen an dem vorgegebenen Layout und Design vorzunehmen, finden Sie die Dokumentation des integrierten CSS-Frameworks unter `https://bulma.io/documentation/`.

[4] jedenfalls solange der Server läuft
[5] Sollte mit dem in `renderUrl` übereinstimmen