報道各位

2018年8月29日
株式会社インプレスR&D
https://nextpublishing.jp/

Haskellのウェブアプリケーションフレームワーク Yesodの入門書！

『Haskellで作るWebアプリケーション　遠回りして学ぶYesod入門』発行

技術書典シリーズ、8月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『Haskellで作るWebアプリケーション　遠回りして学ぶYesod入門』(著者:岡本 和樹、小山内 一由)を発行いたします。

『Haskellで作るWebアプリケーション　遠回りして学ぶYesod入門』
https://nextpublishing.jp/isbn/9784844398516

著者:岡本 和樹,小山内 一由
小売希望価格:電子書籍版　1600円(税別)／印刷書籍版　1800円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文76ページ
ISBN:978-4- 8443-9851-6
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

【Haskellのウェブアプリケーションフレームワーク Yesodの入門書！】

本書は、Haskellの入門書レベルの知識をもつ読者を対象とした、ウェブアプリケーションフレームワーク Yesodの入門書です。

比較的学習コストの高いYesodですが、本書を通じてYesodの基本的な知識とHaskellでのウェブアプリケーション開発に挑んで見ましょう！

〈本書の対象読者〉

Haskellの入門書は既に読みこなしているプログラマ
Haskellでウェブアプリを作ってみたいプログラマ
Haskellで事実上の標準ビルドツールとなっているStackを解説

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

Haskell のウェブアプリケーションフレームワークである Yesod をテンプレートを例に紹介

2.2　生成されたファイル

.dir-locals.elと.gitignoreはここでの本質ではないので省きます。リスト2.1は自動的に挿入されるコメントを削除しています。stack.yaml・hello.cabal・Main.ha・Foundation.hs・Application.hs・routes・Home.hs・Add.hsの順に示します。

ここでは、雰囲気を感じる程度で構いません。Haskellの入門書では見たことがない文法がいくつもあることが分かるでしょう。これが読めるようになることが本書の目標の1つです。

stack.yamlとhello.cabal

リスト2.1: stack.yaml

```
resolver: lts-5.15
packages:
- '.'
extra-deps: []
flags: {}
extra-package-dbs: []
```

リスト2.2: hello.cabal

```
name:                hello
version:             0.0.0
cabal-version:       >= 1.8
build-type:          Simple
extra-source-files: routes

executable           hello
  main-is:           Main.hs
  other-modules:     Application
                     Foundation

                     Add
                     Home

  ghc-options:       -Wall -fwarn-tabs -O2

  build-depends:  base
                , yesod-core

  ghc-options:    -threaded -O2 -rtsopts -with-rtsopts=-N
```

リスト2.1とリスト2.2は、ほぼ第1章「Stackとは」で解説した通りですね。extra-source-filesに指定されているroutesは後で出てきます。GHCのオプションの詳細はここでは解説しないので公式リファレンス等を参考にしてください。

Main.hs

リスト2.3: Main.hs

```
import Application () -- for YesodDispatch instance
import Foundation
import Yesod.Core

main :: IO ()
main = warp 3000 App
```

リスト2.3にmain関数があります。Yesod.Coreモジュールにあるwarp関数で3000番ポートを指定してサーバーを起動します。warpの型は次の通り。AppはYesodDispatch型クラスのなんらかのインスタンスだということが分かりますが、詳細は次項のリスト2.4で解説します。

```
warp :: YesodDispatch site => Int -> site -> IO ()
```

ところでimport Application ()の行は何もインポートしていませんし、消してしまってもよいでしょうか？いえいえ、このコードがあるのとないのではプログラムの意味が変わります。Applicationモジュールでは型クラスのインスタンス定義があるのですが、その型クラスと型は別モジュールで定義されているのでインポートするもの自体はないのです。

Foundation.hs

リスト2.4: Foundation.hs

```
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE TemplateHaskell   #-}
{-# LANGUAGE TypeFamilies      #-}
{-# LANGUAGE ViewPatterns      #-}
module Foundation where

import Yesod.Core

data App = App

mkYesodData "App" $(parseRoutesFile "routes")
```


実際に Yesod を使って簡単な Web アプリケーションの開発を体験

第6章　わいわいWAI

PythonにWSGIが、RubyにRackがあるようにHaskellにはWAI[1]があります。このWAIのAPIを直接使うことで、Yesodが背後で何をしているかを見てみましょう。

WAI・WSGI・Rackは、Webサーバーと Webアプリケーションの間のインターフェースについての仕様です。これに従うことで、Webアプリケーションの実装を変えずに、Webサーバーを別の物に切り替えることが可能になります。

念のためですが、Pythonで書いたアプリケーションをHaskellに切り替える、といったことはできません。あくまでそれぞれの言語内でのWebサーバー・Webアプリケーションの実装についての話です。

……というのが理想で、実際のところHaskellでは大御所Webアプリケーションフレームワーク（Yesod・Snap・Happstack）ごとにその仕様が存在します。仕様の制定がWebアプリケーションフレームワークの作成よりも後で、各々の陣営が仕様を制定したからだったはずだと筆者は考えています。

Yesod陣営ではその名もWeb Application Interface（WAI）という仕様を制定し、その事実上の標準である実装がWarp[2]です。

それではWarpを使って簡単なWebアプリケーションを作っていきましょう。

6.1　Hello, WAI!

まずはHello World、ということでGET要求に対して次に示す応答をするプログラムから始めます。

リスト6.1: これから実装するHTTP応答

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Date: Wed, 11 May 2016 01:23:14 GMT
Server: Warp/3.2.2
Content-Type: text/plain

Hello, Web!
```

1 wai：Stackage Serverhttps://www.stackage.org/package/wai
2 warp：Stackage Serverhttps://www.stackage.org/package/warp

リスト6.2: Main.hs

```
{-# LANGUAGE OverloadedStrings #-}

import Network.Wai              (Application, responseLBS)
import Network.HTTP.Types       (ok200)
import Network.Wai.Handler.Warp (run)

app :: Application
app _ respond =
  respond $ responseLBS
    ok200
    [("Content-Type", "text/plain")]
    "Hello, Web!"

main :: IO ()
main = run 8080 app
```

このプログラムはwai・warp・http-typesパッケージに依存します。

コードの意味については初見でもだいたい理解できるのではないかと思いますが、appの中でステータスコードとコンテンツタイプそして本体を指定しています。mainではポート番号を指定してWebサーバーを起動しています。新出のAPIの詳細を見ていきましょう。

```
type Application = Request
                -> (Response -> IO ResponseReceived)
                -> IO ResponseReceived
responseLBS :: Status -> ResponseHeaders -> ByteString -> Response
ok200 :: Status
run :: Port -> Application -> IO ()
```

Application型はアプリケーション全体を表す型で、その実装は、第1引数でRequestを受け取り、Responseを生成して、第2引数の関数に渡すようにします。

responseLBSはステータスコード・レスポンスヘッダー・本体を渡すとResponseが生成されます。LBSはLazy ByteStringを表します。

6.2　ルーティング

次はパスによるルーティングをしてみましょう。パスはpathInfo関数で取得できます。型を次に示します。

＜＜目次＞＞

＜＜著者紹介＞＞

岡本 和樹
代数的データ型と副作用の分離に惚れ込んで、宣伝活動をしている。街中の λ 形を探すのが日課。Twitter・GitHub は kakkun61。

小山内 一由
2010 年ごろに Haskell に触れ、面倒なチェックを機械におこなわせる魅力を知り Haskell を続けている。2017 年より日本 Haskell ユーザーグループに参加。

＜＜販売ストア＞＞

電子書籍:
Amazon Kindle ストア、楽天 kobo イーブックストア、Apple iBookstore、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍:
Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス
※ 各ストアでの販売は準備が整いしだい開始されます。
※ 全国の一般書店からもご注文いただけます。

【株式会社インプレス R&D】 https://nextpublishing.jp/

株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。
※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証1部9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp