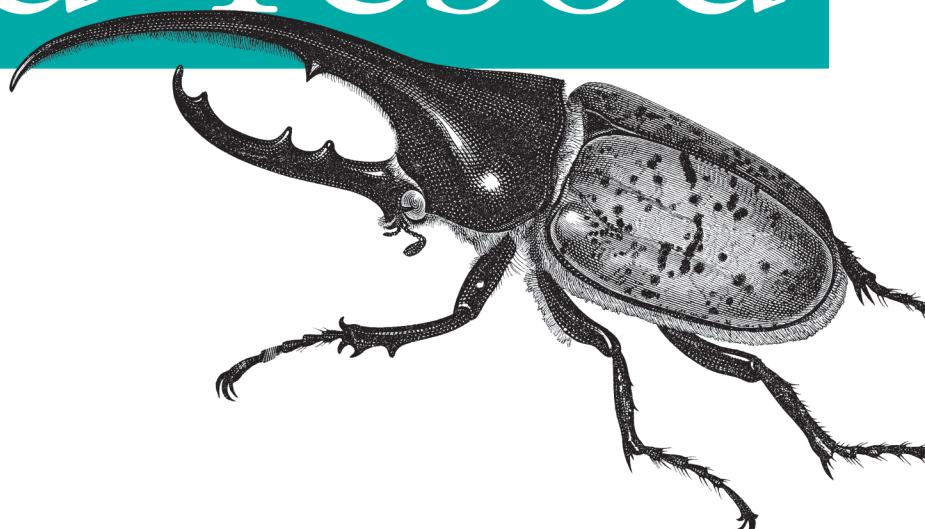


Safety-Driven Web Development

Free Sampler

Developing Web Applications with

Haskell and Yesod



O'REILLY®

Michael Snoyman

Want to read more?

You can [buy this book](#) at [oreilly.com](#)
in print and ebook format.

Buy 2 books, get the 3rd FREE!

Use discount code: OPC10

All orders over \$29.95 qualify for **free shipping** within the US.

It's also available at your favorite book retailer,
including the iBookstore, the [Android Marketplace](#),
and [Amazon.com](#).



O'REILLY®

Spreading the knowledge of innovators

[oreilly.com](#)

Developing Web Applications with Haskell and Yesod

by Michael Snoyman

Copyright © 2012 Michael Snoyman. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editor: Simon St. Laurent

Production Editor: Iris Febres

Proofreader: Iris Febres

Cover Designer: Karen Montgomery

Interior Designer: David Futato

Illustrator: Robert Romano

Revision History for the First Edition:

2012-04-20 First release

See <http://oreilly.com/catalog/errata.csp?isbn=9781449316976> for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *Developing Web Applications with Haskell and Yesod*, the rhinoceros beetle, the mountain apollo butterfly, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-31697-6

[LSI]

1334948660

Table of Contents

| | |
|---------------|----|
| Preface | ix |
|---------------|----|

Part I. Basics

| | |
|------------------------------|-----------|
| 1. Introduction | 3 |
| Type Safety | 3 |
| Concise | 4 |
| Performance | 4 |
| Modular | 4 |
| A Solid Foundation | 5 |
| Introduction to Haskell | 5 |
| 2. Haskell | 7 |
| Terminology | 7 |
| Tools | 8 |
| Language Pragmas | 9 |
| Overloaded Strings | 10 |
| Type Families | 11 |
| Template Haskell | 11 |
| QuasiQuotes | 13 |
| Summary | 13 |
| 3. Basics | 15 |
| Hello World | 15 |
| Routing | 16 |
| Handler Function | 17 |
| The Foundation | 18 |
| Running | 18 |
| Resources and Type-Safe URLs | 19 |
| The Scaffolded Site | 20 |

| | |
|---|-----------|
| Development Server | 20 |
| Summary | 21 |
| 4. Shakespearean Templates | 23 |
| Synopsis | 23 |
| Hamlet (HTML) | 23 |
| Cassius (CSS) | 24 |
| Lucius (CSS) | 24 |
| Julius (JavaScript) | 24 |
| Types | 24 |
| Type-Safe URLs | 25 |
| Syntax | 27 |
| Hamlet Syntax | 27 |
| Cassius Syntax | 31 |
| Lucius Syntax | 32 |
| Julius Syntax | 32 |
| Calling Shakespeare | 33 |
| Alternate Hamlet Types | 35 |
| Other Shakespeare | 37 |
| General Recommendations | 37 |
| 5. Widgets | 39 |
| Synopsis | 39 |
| What's in a Widget? | 40 |
| Constructing Widgets | 41 |
| Combining Widgets | 42 |
| Generate IDs | 42 |
| whamlet | 43 |
| Types | 44 |
| Using Widgets | 45 |
| Summary | 46 |
| 6. Yesod Typeclass | 49 |
| Rendering and Parsing URLs | 49 |
| joinPath | 50 |
| cleanPath | 51 |
| defaultLayout | 52 |
| getMessage | 53 |
| Custom Error Pages | 54 |
| External CSS and JavaScript | 54 |
| Smarter Static Files | 55 |
| Authentication/Authorization | 56 |
| Some Simple Settings | 57 |

| | |
|--------------------------------------|-----------|
| Summary | 57 |
| 7. Routing and Handlers | 59 |
| Route Syntax | 59 |
| Pieces | 60 |
| Resource Name | 61 |
| Handler Specification | 62 |
| Dispatch | 63 |
| Return Type | 63 |
| Arguments | 64 |
| The Handler Monad | 64 |
| Application Information | 65 |
| Request Information | 65 |
| Short Circuiting | 66 |
| Response Headers | 66 |
| Summary | 67 |
| 8. Forms | 69 |
| Synopsis | 69 |
| Kinds of Forms | 71 |
| Types | 72 |
| Converting | 74 |
| Create AForms | 74 |
| Optional Fields | 75 |
| Validation | 76 |
| More Sophisticated Fields | 77 |
| Running Forms | 78 |
| i18n | 79 |
| Monadic Forms | 80 |
| Input Forms | 82 |
| Custom Fields | 83 |
| Summary | 84 |
| 9. Sessions | 85 |
| ClientSession | 85 |
| Controlling Sessions | 86 |
| Session Operations | 87 |
| Messages | 87 |
| Ultimate Destination | 89 |
| Summary | 91 |
| 10. Persistent | 93 |
| Synopsis | 94 |

| | |
|--|------------|
| Solving the Boundary Issue | 94 |
| Types | 95 |
| Code Generation | 96 |
| PersistStore | 98 |
| Migrations | 99 |
| Uniqueness | 101 |
| Queries | 102 |
| Fetching by ID | 102 |
| Fetching by Unique Constraint | 103 |
| Select Functions | 103 |
| Manipulation | 105 |
| Insert | 105 |
| Update | 106 |
| Delete | 107 |
| Attributes | 108 |
| Relations | 110 |
| Closer Look at Types | 111 |
| Custom Fields | 113 |
| Persistent: Raw SQL | 113 |
| Integration with Yesod | 114 |
| Summary | 116 |
| 11. Deploying Your Webapp | 117 |
| Compiling | 117 |
| Warp | 117 |
| Configuration | 118 |
| Server Process | 120 |
| FastCGI | 120 |
| Desktop | 121 |
| CGI on Apache | 121 |
| FastCGI on lighttpd | 122 |
| CGI on lighttpd | 123 |
| <hr/> | |
| Part II. Advanced | |
| 12. RESTful Content | 127 |
| Request Methods | 127 |
| Representations | 128 |
| RepHtmlJson | 129 |
| News Feeds | 131 |
| Other Request Headers | 131 |
| Stateless | 132 |

| | |
|--|------------|
| Summary | 132 |
| 13. Yesod's Monads | 135 |
| Monad Transformers | 135 |
| The Three Transformers | 136 |
| Example: Database-Driven Navbar | 137 |
| Example: Request Information | 139 |
| Summary | 140 |
| 14. Authentication and Authorization | 141 |
| Overview | 141 |
| Authenticate Me | 142 |
| Email | 145 |
| Authorization | 149 |
| Conclusion | 151 |
| 15. Scaffolding and the Site Template | 153 |
| How to Scaffold | 153 |
| File Structure | 154 |
| Cabal File | 154 |
| Routes and Entities | 155 |
| Foundation and Application Modules | 155 |
| Import | 156 |
| Handler Modules | 157 |
| widgetFile | 157 |
| defaultLayout | 158 |
| Static Files | 158 |
| Conclusion | 159 |
| 16. Internationalization | 161 |
| Synopsis | 161 |
| Overview | 163 |
| Message Files | 164 |
| Specifying Types | 165 |
| RenderMessage Typeclass | 165 |
| Interpolation | 166 |
| Phrases, Not Words | 167 |
| 17. Creating a Subsite | 169 |
| Hello World | 169 |

Part III. Examples

| | |
|---|-----|
| 18. Blog: i18n, Authentication, Authorization, and Database | 175 |
| 19. Wiki: Markdown, Chat Subsite, Event Source | 185 |
| 20. JSON Web Service | 193 |
| Server | 193 |
| Client | 194 |
| 21. Case Study: Sphinx-Based Search | 197 |
| Sphinx Setup | 197 |
| Basic Yesod Setup | 198 |
| Searching | 200 |
| Streaming xmppipe Output | 203 |
| Full Code | 206 |

Part IV. Appendices

| | |
|------------------------------------|-----|
| A. monad-control | 213 |
| B. Conduit | 223 |
| C. Web Application Interface | 255 |
| D. Settings Types | 259 |
| E. http-conduit | 261 |
| F. xml-conduit | 267 |

Introduction

Since web programming began, people have been trying to make the development process a more pleasant one. As a community, we have continually pushed new techniques to try and solve some of the lingering difficulties of security threats, the stateless nature of HTTP, the multiple languages (HTML, CSS, JavaScript) necessary to create a powerful web application, and more.

Yesod attempts to ease the web development process by playing to the strengths of the Haskell programming language. Haskell's strong compile-time guarantees of correctness not only encompass types; referential transparency ensures that we don't have any unintended side effects. Pattern matching on algebraic data types can help guarantee we've accounted for every possible case. By building upon Haskell, entire classes of bugs disappear.

Unfortunately, using Haskell isn't enough. The Web, by its very nature, is *not* type safe. Even the simplest case of distinguishing between an integer and string is impossible: all data on the Web is transferred as raw bytes, evading our best efforts at type safety. Every app writer is left with the task of validating all input. I call this problem *the boundary issue*: as much as your application is type safe on the inside, every boundary with the outside world still needs to be sanitized.

Type Safety

This is where Yesod comes in. By using high-level declarative techniques, you can specify the exact input types you are expecting. And the process works the other way as well: using a process of type-safe URLs, you can make sure that the data you send out is also guaranteed to be well formed.

The boundary issue is not just a problem when dealing with the client: the same problem exists when persisting and loading data. Once again, Yesod saves you on the boundary by performing the marshaling of data for you. You can specify your entities in a high-level definition and remain blissfully ignorant of the details.

Concise

We all know that there is a lot of boilerplate coding involved in web applications. Wherever possible, Yesod tries to use Haskell's features to save your fingers the work:

- The forms library reduces the amount of code used for common cases by leveraging the Applicative type class.
- Routes are declared in a very terse format, without sacrificing type safety.
- Serializing your data to and from a database is handled automatically via code generation.

In Yesod, we have two kinds of code generation. To get your project started, we provide a scaffolding tool to set up your file and folder structure. However, most code generation is done at compile time via meta programming. This means your generated code will never get stale, as a simple library upgrade will bring all your generated code up-to-date.

But for those who like to stay in control, and know exactly what their code is doing, you can always run closer to the compiler and write all your code yourself.

Performance

Haskell's main compiler, the GHC, has amazing performance characteristics, and is improving all the time. This choice of language by itself gives Yesod a large performance advantage over other offerings. But that's not enough: we need an architecture designed for performance.

Our approach to templates is one example: by allowing HTML, CSS, and JavaScript to be analyzed at compile time, Yesod both avoids costly disk I/O at runtime and can optimize the rendering of this code. But the architectural decisions go deeper: we use advanced techniques such as conduits and builders in the underlying libraries to make sure our code runs in constant memory, without exhausting precious file handles and other resources. By offering high-level abstractions, you can get highly compressed and properly cached CSS and JavaScript.

Yesod's flagship web server, Warp, is the fastest Haskell web server around. When these two pieces of technology are combined, it produces one of the fastest web application deployment solutions available.

Modular

Yesod has spawned the creation of dozens of packages, most of which are usable in a context outside of Yesod itself. One of the goals of the project is to contribute back to the community as much as possible; as such, even if you are not planning on using

Yesod in your next project, a large portion of this book may still be relevant for your needs.

Of course, these libraries have all been designed to integrate well together. Using the Yesod Framework should give you a strong feeling of consistency throughout the various APIs.

A Solid Foundation

I remember once seeing a PHP framework advertising support for UTF-8. This struck me as surprising: you mean having UTF-8 support isn't automatic? In the Haskell world, issues like character encoding are already well addressed and fully supported. In fact, we usually have the opposite problem: there are a number of packages providing powerful and well-designed support for the problem. The Haskell community is constantly pushing the boundaries finding the cleanest, most efficient solutions for each challenge.

The downside of such a powerful ecosystem is the complexity of choice. By using Yesod, you will already have most of the tools chosen for you, and you can be guaranteed they work together. Of course, you always have the option of pulling in your own solution.

As a real-life example, Yesod and Hamlet (the default templating language) use `blaze-builder` for textual content generation. This choice was made because blaze provides the fastest interface for generating UTF-8 data. Anyone who wants to use one of the other great libraries out there, such as `text`, should have no problem dropping it in.

Introduction to Haskell

Haskell is a powerful, fast, type-safe, functional programming language. This book takes as an assumption that you are already familiar with most of the basics of Haskell. There are two wonderful books for learning Haskell, both of which are available for reading online:

- [Learn You a Haskell for Great Good!](#)
- [Real World Haskell](#)

Yesod relies on a few features in Haskell that most introductory tutorials do not cover. Though you will rarely need to understand how these work, it's always best to start off with a good appreciation for what your tools are doing. These are covered in the next chapter.

O'Reilly Ebooks—Your bookshelf on your devices!



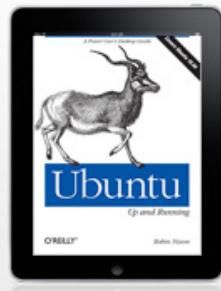
PDF



ePub



Mobi



APK



DAISY

When you buy an ebook through oreilly.com you get lifetime access to the book, and whenever possible we provide it to you in five, DRM-free file formats—PDF, .epub, Kindle-compatible .mobi, Android .apk, and DAISY—that you can use on the devices of your choice. Our ebook files are fully searchable, and you can cut-and-paste and print them. We also alert you when we've updated the files with corrections and additions.

Learn more at ebooks.oreilly.com

You can also purchase O'Reilly ebooks through the iBookstore, the [Android Marketplace](#), and [Amazon.com](#).

O'REILLY®

Spreading the knowledge of innovators

oreilly.com