REST in a Snap(let)

Building Web Resources in Haskell

Timothy Jones
github.com/zmthy

Background

CoffeeScript

node.js™

express

This Talk

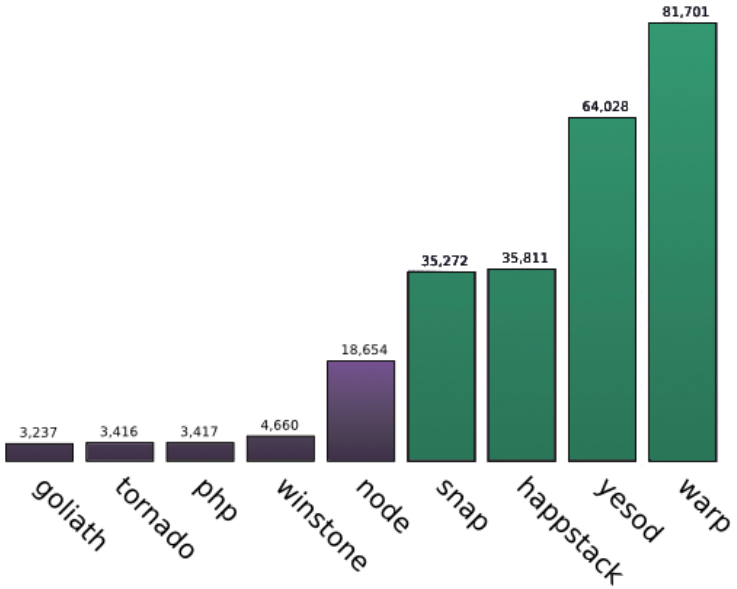
Why Haskell?

This ain't your Java's type system

Why Haskell?

This ain't your Java's type system

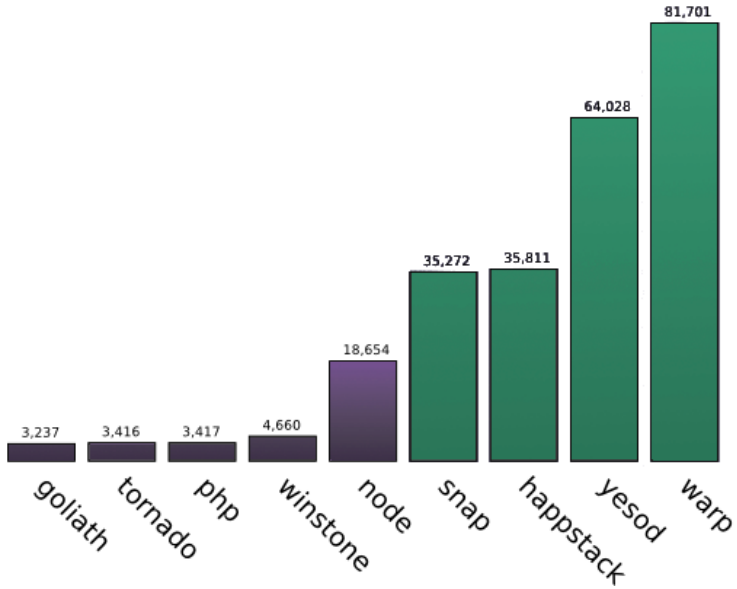
Runs natively



Why Haskell?

This ain't your Java's type system

Runs natively



Open Source!

Why Haskell?

Haskell on the Web

Snap

Get started quickly

```
import Snap

quickHttpServe (writeText "Hello, world!")
```

Snap

Get started quickly

```
import Snap

quickHttpServe (writeText "Hello, world!")
```

For bigger projects:

```
$ snap init
```

Snap Routing

Routes are data structures

```
serveUsers = path "/user"
    (method GET readUser <|> method POST createUser)
```

Snap Routing

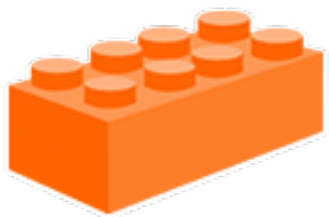
Routes are data structures

```
serveUsers = path "/user"
    (method GET readUser <|> method POST createUser)
```

They compose easily

```
routes = ifTop serveHomePage <|> serveUsers
```

Snaplets



Snaplets

Embeddable components

```
data MySite = MySite {
    heist :: Snaplet Heist,
    db    :: Snaplet Sqlite
}
```

Snaplets

Embeddable components

```
data MySite = MySite {
    heist :: Snaplet Heist,
    db    :: Snaplet Sqlite
}
```

MySite is a Snaplet too!

```
data YourSite = YourSite {
    subSite :: Snaplet MySite
}
```

Building a Snaplet for REST

CRUD

Building a Snaplet for REST

CRUD

Methods

PUT, DELETE...

Building a Snaplet for REST

CRUD

Methods

PUT, DELETE...

Content Negotiation

Accept: `application/json`

Don't hate the player, HATEOAS the game

Hypermedia

```
<link href="/person/tim">
    <method>GET</method>
    <method>PATCH</method>
</link>
```

Content negotiation is hard

Accept: `application/json; q=0.8, text/*; q=0.2`

Content-Type: `application/xml; charset=utf-8`

Foundational Packages

http-media

```
parseAccepts header >>= mapAccept
    [ ("text/html",         asHTML)
    , ("application/json", asJSON)
    ]
```

Foundational Packages

http-media

```
parseAccepts header >>= mapAccept
    [ ("text/html",         asHTML)
    , ("application/json", asJSON)
    ]
```

snap-accept

```
accept "text/html" asHTML
    <|> accept "application/json" asJSON
    <|> send406Error
```

snaplet-rest

Prototyping – up and running quickly

snaplet-rest

Prototyping – up and running quickly

Use a CRUD model to produce a REST interface

snaplet-rest

Prototyping – up and running quickly

Use a CRUD model to produce a REST interface

Doing the right thing should be easy

User Resource

```
data User = User {
    user :: Username,
    name :: String,
    age  :: Int
}

type Username = CaseInsensitive String
```

User Resource

```
data User = User {
    user :: Username,
    name :: String,
    age  :: Int
}

type Username = CaseInsensitive String

deriveJSON ''User
```

Aw CRUD

```
createUser :: User → m ()
```

Aw CRUD

```
createUser :: User → m ()

readUser   :: id → m [User]
```

Aw CRUD

```
createUser :: User → m ()

readUser   :: id → m [User]

deleteUser :: id → m Bool
```

Aw CRUD

```
createUser :: User → m ()

readUser   :: id → m [User]

deleteUser :: id → m Bool

updateUser :: id → diff → m Bool
```

Quite partial

```
data UserPart = UserPart {
    pName :: Maybe String,
    pAge  :: Maybe Int
}

deriveJSON ''UserPart
```

Quite partial

```
data UserPart = UserPart {
    pName :: Maybe String,
    pAge  :: Maybe Int
}

deriveJSON ''UserPart

type UserId = Either Username UserPart
```

Filling in the gaps

```
createUser :: User → m ()

readUser   :: UserId → m [User]

deleteUser :: UserId → m Bool

updateUser :: UserId → UserPart → m Bool
```

Putting it all together

```
serveUser = serveResource $ resource
    & addMedia  jsonFromInstances
    & setCreate createUser
    & setRead   readUser
    & setUpdate updateUser
    & setDelete deleteUser
```

A counter

```
data Count = Count { amount :: Int }
deriveJSON ''Count

data App = App { count :: Count }

serveCount = serveResource $ resource
    & addMedia  jsonFromInstances
    & setRead   (λ() → with count get)
    & setUpdate (λ() → with count · put)
```

What's next?

Automating partial structure

Nested resources

Online

github.com/zmthy/snaplet-rest

```
$ cabal update
$ cabal install snaplet-rest
```