

次世代言語 CROSS Yesod web framework

伊東 勝利 /HIMA' (株) タイムインターメディア

自己紹介

- なまえ：伊東 勝利
- しょぞく：株式会社タイムインターメディア
- すきな言語：Haskell, Scheme
- しゅみ：水泳
- とくぎ：バブルリング



Yesod とは

- Web フレームワーク
- Haskell で開発されている
 - ◆ 強い型付け
 - ◆ 純粋 (side-effect free)
 - ◆ 速い
- ライブラリのおつまり

- フルスタック
 - ◆ Web サーバ
 - ◆ ORM
 - ◆ Add-on ライブラリ
 - ◆ auth
 - ◆ gravatar
 - ◆ widget etc.
- Yesod は Foundation

開発環境・コンパイラ・
IDE, ライブラリなど

- GHC7
- Yesod 自体がライブラリのおつまり
- IDE は特になし
- 通常の Haskell プログラミング環境例)

Emacs+haskell-mode+ghc-mod

メリット

こういうケースで長所が活きてくる
他にない技術、他より優れた方法

特長 1:Type-safe URLs

- すべての URL にはデータ型がある
 - ◆ 有効な URL は値として表現がある
- URL を中心に以下を自動的に同期
 - ◆ URL のパース
 - ◆ URL のレンダリング
 - ◆ URL からハンドラ関数ディスパッチ

例 :Type-safe URLs

```
mkYesod "MyApp" [parseRoutes|  
/ RootR GET  
/entry/#BlogId BlogPostR GET  
/entries/#Year/#Month BlogListR GET  
|]
```


メリット :Type-safe URLs

- パスを一箇所に記述
- データ型にもとづいて自動的にマーシャリング
- データ型の変更に対してコンパイラがエラーを捕捉

メリット :Type-safe URLs

- パスを一箇所に記述
- データ型にもとづいて自動的にマーシャリング
- データ型の変更に対してコンパイラがエラーを捕捉

特長 2:Compile-time templates

- やさしい構文
- コンパイル時に構文チェックされる
- Haskell の変数を直接使える
 - ◆ テンプレート中に書くための糊付けコードは不要
 - ◆ 自動的に型検査される
- シンプルな制御構文
- css と js もテンプレート
 - ◆ Debug mode では変更したら自動コンパイル
 - ◆ Quick development cycle

Hamlet(HTML)

```
<html>
  <head>
    <title>#{pageTitle} - My Site
    <link rel="stylesheet" href=@{StyleSheetR}>
  <body>
    <h1 .page-title> おともだちの一覧 :
    $if null friends
      <p> すまん . おともだちはいないや .
    $else
      <ul #friends-listing>
        $forall f <- friends
          <li>#{friendName f} (#{show $ friendAge f} 才)
    <footer>^{\copyright}
```


Lucius(CSS)

```
section.blog {  
  padding: 1em;  
  border: 1px solid #000  
  h1 {  
    color: #{headingColor};  
  }  
  background-image: url(@{MyBackgroundR});  
}
```


Julius(Javascript)

```
$(function(){  
  $("section.#{sectionClass}").hide();  
  $("#mybutton").click(function(){  
    document.location = "@{SomeRouteR}";  
    ^{addBling}  
  });  
});
```


メリット :Compile-time templates

- 簡単そうでしょ？
- ランタイムの表示エラーやリンク切れもない
- 良く使うテンプレートも部品化して埋め込み可能
- XSS Protection

例 : XSS Protection

```
name :: Text
```

```
name = "Michael <script>alert('XSS')</script>"
```

```
main :: IO ()
```

```
main = putStrLn $ renderHtml [shamlet|#{name}|]
```

出力 :

```
Michael &lt;script&gt;alert(&#39;XSS&#39;)&lt;/script&gt;
```


特長 3: Persistent template&EDSL

- 一箇所でエンティティの宣言
- 自動的に Haskell の型を生成し, 関数と SQL スキーマをマーシャリング
- 各テーブルから ID を分離
- 全てのマーシャリングと有効性検査はライブラリでハンドリング
- 自動マイグレーション
- SQL と MongoDB を容易に切り換え可能

例 : Persistent template & EDSL

```
mkPersist [persist|
```

```
Person
```

```
    name Text
```

```
    age Int Maybe
```

```
BlogPost
```

```
    title Text
```

```
    author PersonId
```

```
|]
```


例 : Persistent CRUD

```
RunMigration migrateAll
```

```
cutseaId <- insert $ Person " いとうかつとし" $ Just 35
```

```
keikoId <- insert $ Person " いとうけいこ" Nothing
```

```
insert $ BlogPost " 最初の投稿" cutseaId
```

```
insert $ BlogPost " もいっこ投稿" cutseaId
```

```
onePost <- selectList [BlogPostAuthorId==.cutseaId][LimitTo 1]
```

```
liftIO $ print (onePost :: [(BlogPostId, BlogPost)])
```

```
update cutseaId [PersonAge +=. Just 1]
```

```
cutsea <- get cutseaId
```

```
delete cutseaId
```

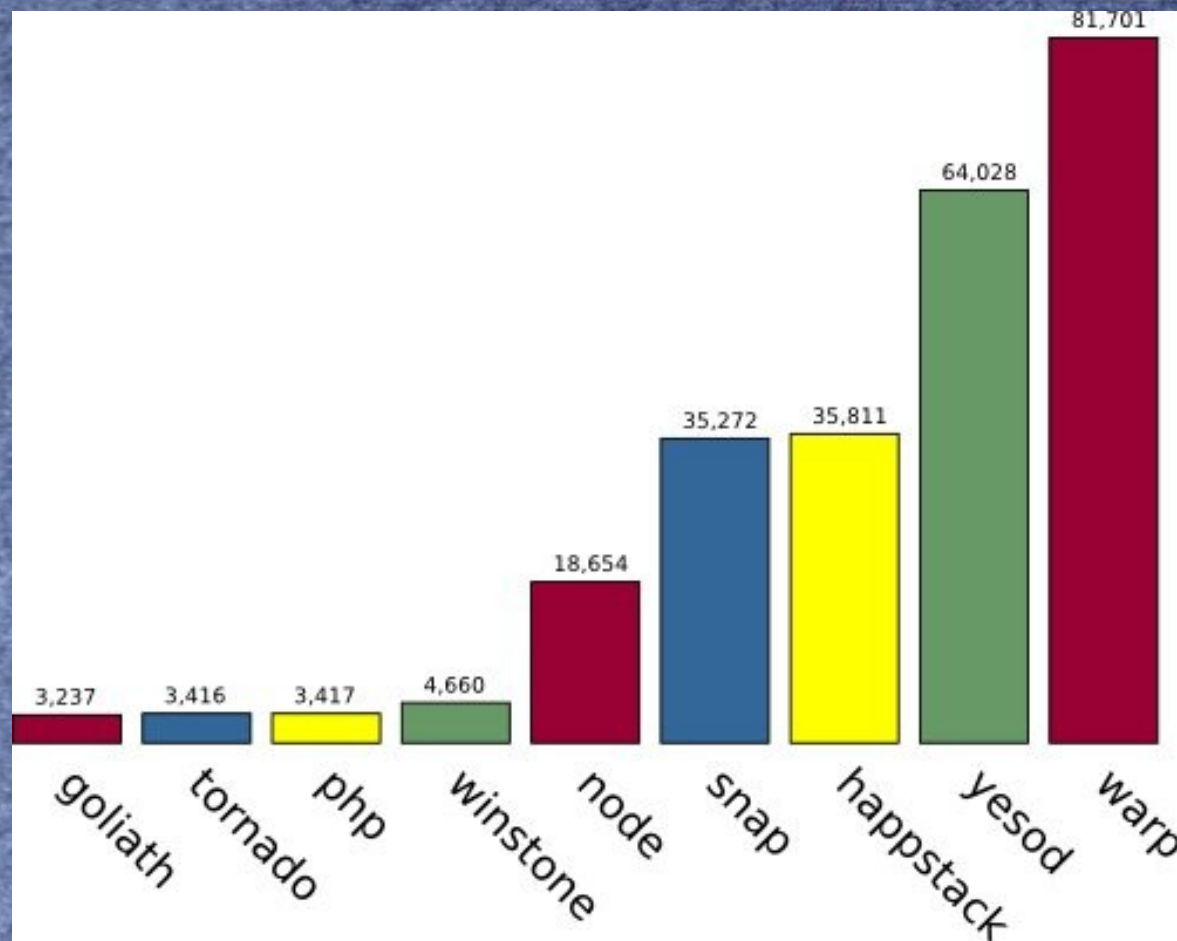
```
deleteWhere [BlogPostAuthorId==.cutseaId]
```


メリット : Persistent template & EDSL

- 簡単そうでしょ？
- クエリの記述が EDSL なので Haskell の全能力を使って合成できる
- コンパイル時にクエリも型検査される
- No SQL injection

メリット：速い

- Warp ベンチマーク（超古いですけど）



Pong benchmark, extra large instance, requests/second

メリット：速い

- Web Application Interface(WAI)
- Blaze-builder
- Enumerator(-> Conduit)
- Multi-threaded runtime
- Haskell は速いよ！(by snoyman)

メリット：モジュラリティ

- Widgets

- ◆ HTML/CSS/Javascript をコンポーネント化
- ◆ 複数箇所に含めることができる
- ◆ DB アクセスも含めることができる
- ◆ `<body>` と `<head>` の両方同時に影響を与える
- ◆ CSRF Protection

例 :Widgets

```
existingLinks :: Widget
```

```
existingLinks = do
```

```
  links <- lift $ runDB $ selectList [] []
```

```
  toWidget [lucius|li{list-style-type:none;}|]
```

```
  [whamlet|
```

```
<ul>
```

```
  $forall link <- links
```

```
    <a href=@{linkUrl $ snd link}>#{linkTitle $ snd link}
```

```
|]
```


例 : Widgets

- 使う側のテンプレート :

```
<h2> 登録済みのリンク  
^{existingLinks}
```

- 出力

```
<html><head><title></title><style>.message{color:red;}li{  
list-style-  
type:none;}</style></head><body>../body>
```


デメリット

- 現状 MySQL, Oracle, SQL Server などはサポートしてないので、これらの DB に縛りがある場合

適用事例

- Trough Suite Solutions:
 - ◆ Pruduction yesod site at Emerson(Social Knowledge Base)
 - ◆ Warp Webserver powering Dell's context-sensitive help
 - ◆ Various yesod libraries used at Cisco and LifeTech

適用事例

- Very active, friendly community, lots of them making sites too.
 - ◆ yesodweb - Yesod 公式サイト
 - ◆ haskellers.com - Haskell の SNS
 - ◆ TKProf - GHC プロファイラ可視化
 - ◆ Kestrel - WIKI(某短期大学案件)
 - ◆ BISocie - BTS(某短期大学案件)
 - ◆ その他

www.yesodweb.com/wiki/powerd-by-yesod

コミュニティの動向

- 議論の場を web-devel(Haskell の Web 一般) な ML から Google Group(Yesod web Framework) に移行
- 主に Google Group で議論して yesodweb 上の blog で技術の紹介やリリースアナウンスなどがされる
- 現在最新は 0.9.4.1(2011/12/27) で 1 ~ 6 ヶ月おきに 0.1 ずつバージョンアップ
- 実装技術 (スタイル) 的には Enumerator から Conduit へシフト中
- 1 月あたりに予定されていた Yesod-1.0(安定版) のリリースは見送り
- Yesod-0.10 で Conduit 化した後にあらためて 1.0 へというリスケジューリングが発生 ← イマココ!!

Yesod で Hello

Yesod で Hello

特長の分るコード例

特長の解説

- アプリケーションが Hello/Links というデータ型になっている
- Yesod アプリにするのに Yesod クラスのインスタンスにしている
- defaultLayout でサイトのデフォルトページを設定できます
 - ◆ Yesod クラスのメソッド

特長の解説

- `selectList [][]` だけでも Link テーブルを `select` できてる
 - ◆ 型推論でクエリ対象テーブルを判断可能
 - ◆ `LinkUrl` や `linkTitle` の使用から判断できる
- テンプレート中の `@{AddLinkR}` も型検査されている
 - ◆ 間違っていればエラーになるのでリンク切れのままリリースすることはない

特長の解説

- HTML/CSS/Javascript を (DB アクセスなどロジックも) コンポーネント化できる (Widget)
- urlField,emailField など便利なフィールドも用意されている