Web Programming in Haskell using Yesod

Bangalore Haskell User Group Meetup
Xebia India

Dec 16, 2017

About Me

- Sibi
- Consultant for Xebia
- Internet Handle: ▸ psibi
- Mail: sibi@psibi.in
- ▸ http://psibi.in

Outline

Goal

- Viable choice
- Yesod's philosophy
- Survey of yesod related libraries, ORM backends etc.
- Real world experience

History

- Yesod is a Hebrew word
- Originally developed by Michael Snoyman
- Initial release: March 8, 2010
- License: MIT

Why Yesod?

- ~~R~~ evolutionary framework
- Avoid bugs at compile time
- DSLs for common tasks
- Performant

Persistent

- Database agnostic
- Data modeling
- Migrations

Declare tables

```
mkPersist [persist|
User
  name String
  age Int
Todo
  userId UserId
  task Text
  done Bool
|]
```

- Other syntax: sqltype, constraint, sql etc.
- Deriving json
- Reference: ▸ Persistent docs

CRUD

```
simonId <- insert $ "SPJ" 34
marlowId <- insert $ "Marlow" 35

insert_  $ simonId "Buy dragon book" False

simonTasks <- selectList
                [TodoUserId ==. simonId]
                [LimitTo 5]

(simon :: Maybe User) <- get simonId

delete marlowId
deleteWhere [TodoUserId ==. simonId]
```

Routing

- Single data type for representing URLs
- Mapping between route to handler is done
- Two way parsing functions created
- Sync between parsing and handler functions ensured

Route syntax

```
/  HomeR  GET  POST
/hello  HelloR
/fib/#Int  FibR  GET
```

- Canonical URLs (joinPath, cleanPath)
- Pieces: Static, Dynamic single, Dynamic multi

Routing internals

```
data MyAppRoute = HomeR | HelloR | FibR Int

renderMyAppRoute HomeR = []
renderMyAppRoute HelloR = ["hello"]
renderMyAppRoute (FibR int) =
    ["fib", toSinglePiece int]

parseMyAppRoute [] = Just HomeR
parseMyAppRoute ["hello"] = Just HelloR
parseMyAppRoute ["fib", int] = do
    fibInt <- fromSinglePiece int
    return $ FibR fibInt
parseMyAppRoute _ = Nothing
```

Template languages

- DSLs for templating
- Compile time syntax checked
- Variable interpolation
- Control structures for Hamlet

Hamlet

```
$doctype 5
<html>
  <head>
    <title>#{pageTitle} - My Site
      <link rel=stylesheet href=@{StylesheetR}>
    <body>
      <h1 .page-title>#{pageTitle}
      <p>Here is a list of your friends:
      $if null friends
          <p>Sorry, I lied
          <p>You don't have any friends.
      $else
          <ul>
              $forall Friend name age <- friends
                <li>#{name} (#{age} years old)
      <footer>^{copyright}
```

Lucius & Cassius (CSS)

Lucius

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

Cassius

```
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}
});
```

XSS Protection

```
name :: Text
name = "Sibi <script>alert('injected')</script>"

main :: IO ()
main = putStrLn $ renderHtml [shamlet|#{name}|]
```

Output:

```
Sibi &lt;script&gt;alert(&#39;injected&#39;)&lt;/script&gt;
```

XSS Protection

- blaze-html package
- ToMarkup typeclass
- Textual values are always escaped (see ToMarkup instances)
- Html values aren't escaped
- preEscapedToHtml

Widgets

- Glue between template languages
- Ability to re-use a single UI component
- Can perform IO operations (DB queries etc.)
- Controls the generation of end HTML (<body>,<head> tags)

Widget: Example

```
getHomeR = defaultLayout $ do
    setTitle "My Page Title"
    toWidget [lucius| h1 { color: green; } |]
    addScriptRemote "https://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"
    toWidget
        [julius|
            $(function() {
                $("h1").click(function(){
                    alert("You clicked on the heading!");
                });
            });
        |]
    toWidgetHead
        [hamlet|
            <meta name=keywords content="some sample keywords">
        |]
    toWidget
        [hamlet|
            <h1>Here's one way of including content
        |]
    [whamlet|<h2>Here's another |]
    toWidgetBody
        [julius|
            alert("This is included in the body itself");
        |]
```

Deploying Yesod apps

- Common advice: Don't compile on server
- Change config file for production (Personally used CPP)
- Files to deploy: executable, static folder
- Keter
- Hapistrano

Other parts

- Subsites
- Middlewares
- clientsession
- yesod-form
- yesod-auth
- yesod-fb
- yesod-sitemap
- websocket/eventsource support
- Lots of others in Hackage
- Real world experience

Want to contribute?

- Github: psibi/yesod-rest ▸ yesod-rest
- Github: psibi/wai-slack-middleware ▸ wai-slack
- yesod, persistent etc.

Questions... ?