

Web applications in Haskell

Bc. Pavel Dvořák
FI MUNI

The primary objective

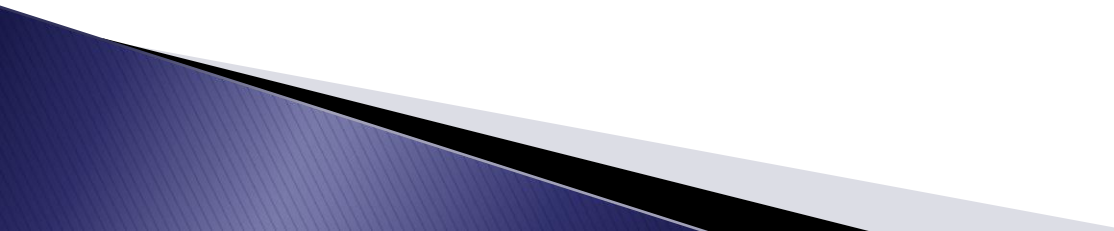
- ▶ To describe methods used for development of web applications written in the Haskell programming language.

Hypertext Transfer Protocol

- ▶ The fundamental communication protocol of the Web.
- ▶ Client sends an HTTP request to a web server. The server returns a proper HTTP response.
- ▶ Such an interaction is stateless.
- ▶ The HTTP specification determines what should such a request and a response look like.

```
communicate :: Request -> IO Response
```

Web application development

- ▶ Our web application has to be run along with a web server that listens on a specified port.
 - ▶ Every time when the server receives a request, the server passes the request to the application and waits for the result.
 - ▶ In Haskell, there are various possibilities for web content serving.
- 

FastCGI

- ▶ Common Gateway Interface is a link between a web server and a web application.
- ▶ With a right configuration, we can run Haskell on one of the widely used web servers such as Apache or lighttpd.
- ▶ Unlike the regular CGI, FastCGI is able to process more requests at once, which reduces the overhead.

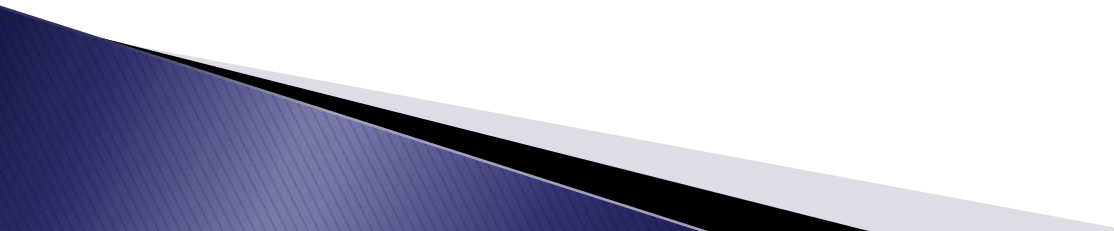
```
main :: IO ()  
main = runFastCGIConcurrent 8 $ output "Hello, world!"
```

WAI + Warp

- ▶ WAI is a web interface, Warp is a web server.
- ▶ They are both written completely in Haskell and take advantage of the Iteratee enumerator.
- ▶ The combination is the fastest native way for running Haskell web applications.

```
main :: IO ()
main = run 8000 (const . return $
                responseLBS statusOK [] "Hello, world!")
```

A web framework

- ▶ A collection of libraries designated for a specified task, in this case for web application development.
 - ▶ Provides facilities for DBMS, template processing, URL mapping...
 - ▶ In Haskell, there are more than a dozen of web frameworks available.
- 

Happstack

HAPPSTACK
A Haskell Web Framework

- ▶ One of the oldest Haskell frameworks.
- ▶ A relatively large piece of software; there is a lite version available, though.
- ▶ The state of the application can be saved and retrieved using the MACID monad.

Snap



- ▶ A young web framework.
- ▶ It provides reusable web components called snaplets, which are similar to Java applets.
- ▶ That means that the framework is very customisable.

Yesod



- ▶ Also a young framework.
- ▶ The WAI and Warp packages originated from Yesod.
- ▶ It employs massively Template Haskell, a metaprogramming extension.

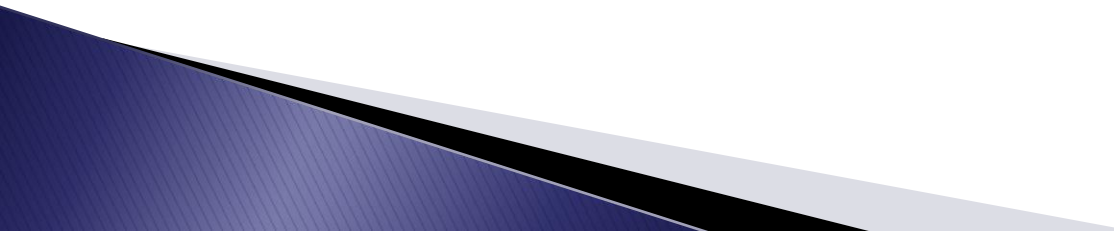
The lesson

- ▶ There is no single way to write a web application in Haskell.

The secondary objective

- ▶ To improve substantially one of the described Haskell web frameworks.

Persistent

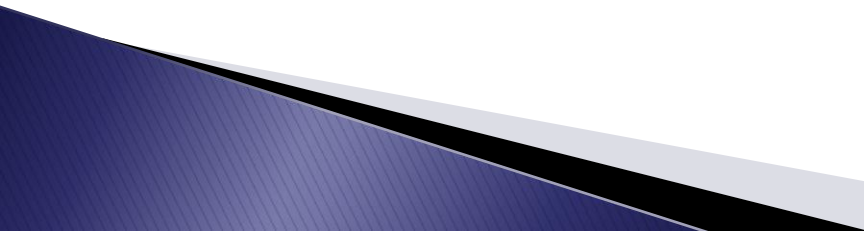
- ▶ A unified interface between one of the several database management systems and the Yesod web framework.
 - ▶ Persistent derives a database scheme from a data structure in our web application and automatically generates database queries.
 - ▶ Currently, it supports PostgreSQL, SQLite, MySQL, and MongoDB.
- 

Persistent — an example

```
share [mkPersist sqlSettings, mkMigrate "migrateAll"] [persist|
Person
  firstName String
  lastName String
  age Int
  PersonName firstName lastName
|]

main :: IO ()
main = withSqliteConn ":memory:" $ runSqlConn $ do
  runMigration migrateAll
  johnId <- insert $ Person "John" "Doe" 26
  x <- selectList [PersonAge >. 21] [LimitTo 10]
  liftIO $ print x
  return ()
```

CouchDB

- ▶ A NoSQL, document-oriented database system written in Erlang.
 - ▶ Reliable, fault-tolerant, highly concurrent.
 - ▶ It provides a RESTful web service together with a user-friendly web interface called Futon.
 - ▶ Every piece of information is encoded into the JSON format.
 - ▶ Data transformation done by JavaScript views.
- 

CouchDB in Haskell

- ▶ For accessing the database, we can utilize the `Database.CouchDB` module.
- ▶ It encodes the data into the JSON format using the `Text.JSON` module.
- ▶ All the interactions are encapsulated inside a custom monad.

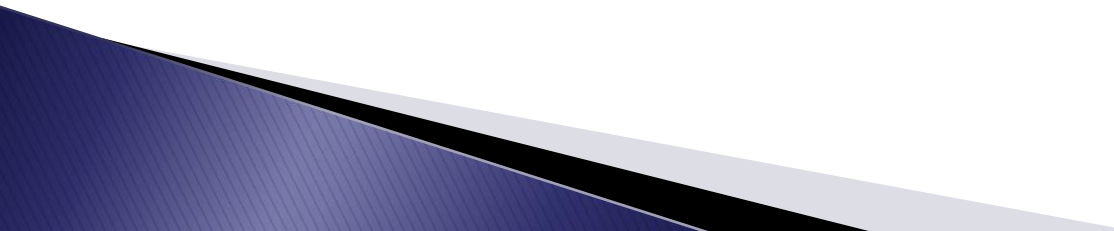
```
data CouchMonad a =  
    CouchMonad (CouchConn -> IO (a, CouchConn))
```


CouchDB in Haskell — an example

```
> conn <- createCouchConn "localhost" 5984
> let sp = db "south_park"
> let eric = JSObject $ toJSObject
    [ ("name", JSString $ toJSString "Eric Cartman"),
      ("age", JSRational False 9) ]
> (doc, rev) <- runCouchDBWith conn $ newDoc sp eric

> do {(Just (_, _, x)) <- runCouchDBWith conn $ getDoc
    sp doc; putStrLn . render $ pp_value x}
{"_id": "7d4ffcae98cdba9a7f6992470a00115e",
 "_rev": "1-28be4e4fca34e9811b4fbc85eb7aaea4",
 "name": "Eric Cartman", "age": 9}
```

Our contribution

- ▶ We have implemented the Persistent interface for CouchDB using the Database.CouchDB module, which had had to be fixed first.
 - ▶ The basic element — a custom Reader monad.
 - ▶ Our module generates necessary JavaScript code for data filtering.
 - ▶ All the Persistent's functionality has been successfully covered.
- 

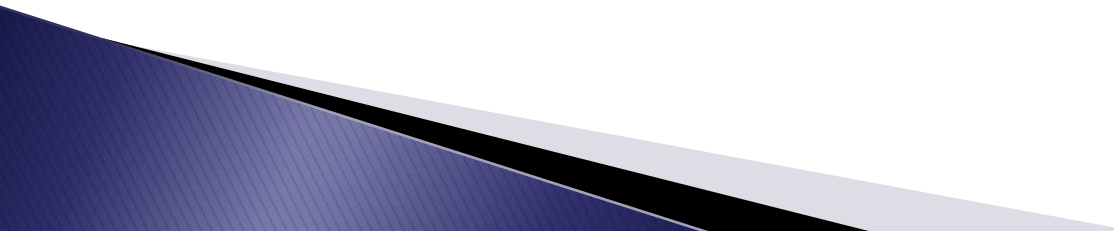
The reaction of a Yesod developer



 **gregwebs** commented

First of all - awesome!

Further work

- ▶ Refactor the module in accordance with the recent Persistent's conceptual changes.
 - ▶ Improve efficiency of the code. (Other changes to the CouchDB library are probably needed.)
 - ▶ Test the interface extensively in order to declare it stable.
- 



Do you have any questions? >>

Thank you for your attention.