

Tutorial: Microservices in Haskell

Alexander Abushkevich

March 2015

REST and DB access

alexanderaa.github.io/haskell-microservices

Agenda

- ▶ Microservices - definition and introductory notes
- ▶ Quick overview - request/response cycle and associated type conversions
- ▶ Focus on a variety of libraries, which help to (de)serialize JSON/XML/...

Agenda

- ▶ Microservices - definition and introductory notes
- ▶ Quick overview - request/response cycle and associated type conversions
- ▶ Focus on a variety of libraries, which help to (de)serialize JSON/XML/...
- ▶ Focus on a variety of libraries, which help to (de)serialize DB data
- ▶ REST in Haskell
- ▶ Putting all pieces together - compile and run the resulting microservice

Microservices - definition

```
1  val range f t =  
2      let  
3          fun range' acc f t =  
4              if (f >= t)  
5                  then acc  
6                  else (range' (f :: acc) (f+1) t)  
7      in  
8          range' [] f t  
9      end
```


Applicative functors

```
Prelude> import Control.Applicative
Prelude Control.Applicative> :t (<$>)
(<$>) :: Functor f => (a -> b) -> f a -> f b
```

Declare data types

Message, which contains text s

```
data MId          = MId Integer           deriving (Eq, Ord, Show)
data EmailAddress = EmailAddress String    deriving (Eq, Ord, Show)
data MText        = MText String          deriving (Eq, Ord, Show)
data MStatus = MNew
           | MRead
           | MDeleted
           deriving (Eq, Ord, Show)

data Message = Message { mId      :: MId
                        , mText    :: MessageText
                        , mStatus  :: MessageStatus
                        , mDated   :: UTCTime }
                        deriving (Eq, Ord, Show, Typeable)
```


Aeson conversions

```
1 instance AE.FromJSON Message where
2     parseJSON (AE.Object v) =
3         Message <$> (v AE..: "id"          )
4                     <*> (v AE..: "message"  )
5                     <*> (v AE..: "status"   )
6                     <*> (v AE..: "dated"    )
```

Add simple quickcheck test

```
prop_OptionJSON :: Option -> Bool
prop_OptionJSON o = (((AE.decode . AE.encode) (Just o)) ==
```

PostgreSQL-simple

Questions?