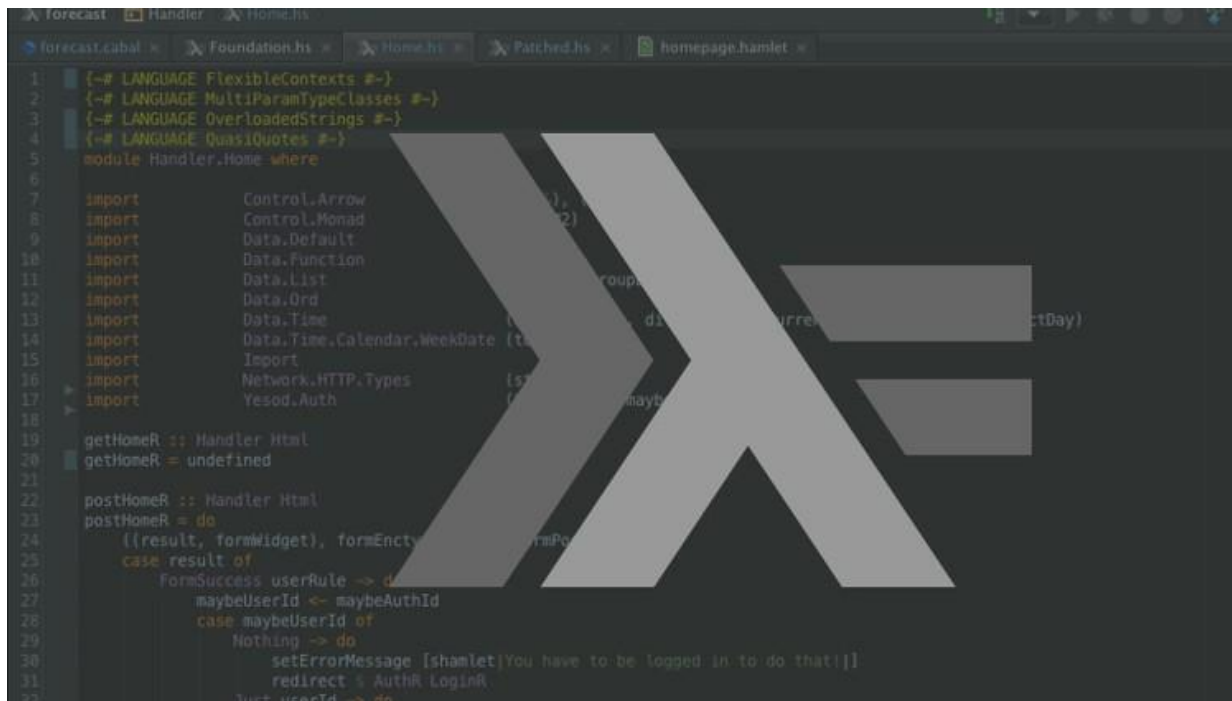


Access to Tuples



```
1 {-# LANGUAGE FlexibleContexts #-}
2 {-# LANGUAGE MultiParamTypeClasses #-}
3 {-# LANGUAGE OverloadedStrings #-}
4 {-# LANGUAGE QuasiQuotes #-}
5 module Handler.Home where
6
7 import Control.Arrow
8 import Control.Monad
9 import Data.Default
10 import Data.Function
11 import Data.List
12 import Data.Ord
13 import Data.Time
14 import Data.Time.Calendar.WeekDate
15 import Import
16 import Network.HTTP.Types
17 import Yesod.Auth
18
19 getHomeR :: Handler Html
20 getHomeR = undefined
21
22 postHomeR :: Handler Html
23 postHomeR = do
24   ((result, formWidget), formEnctype, formPost)
25   <- case result of
26     FormSuccess userRule -> do
27       maybeUserId <- maybeAuthId
28       case maybeUserId of
29         Nothing -> do
30           setErrorMessage [shamlet|You have to be logged in to do that!|]
31           redirect % AuthR.LoginR
32         Just userId -> do
```

Access to Tuples

For tuples of two elements, it can be accessed with `fst` and `snd`:

```
fst :: (a, b) -> a
snd :: (a, b) -> b

fst (3, "sense")    ↩ 3
snd (3, "sense")    ↩ "sense"
```

For general tuples, no accessor functions are defined

⇒ They can be easily created using patterns:

```
first (x, y, z) = x
second (x, y, z) = y
third (x, y, z) = z
```

```
first (x, _, _) = x
second (_, y, _) = y
third (_, _, z) = z
```

Access to Tuples

```
(3, 'z', False) :: (Int, Char, Bool)
(6, 9)          :: (Int, Int)
(True, (6, 9))  :: (Bool, (Int, Int))
```

```
mostFrequentCharacter :: String -> (Char, Int)
mostFrequentCharacter "AVATAR"  => ('A', 3)
```

```
timeDecomposition :: Int -> (Int, Int, Int)    -- hours, minutes, seconds
```

```
timeDecomposition seconds = (h, m, s)
  where
    h = div seconds 3600
    m = div (mod seconds 3600) 60
    s = mod seconds 60
```

Lucas Bazilio Youtube Channel

