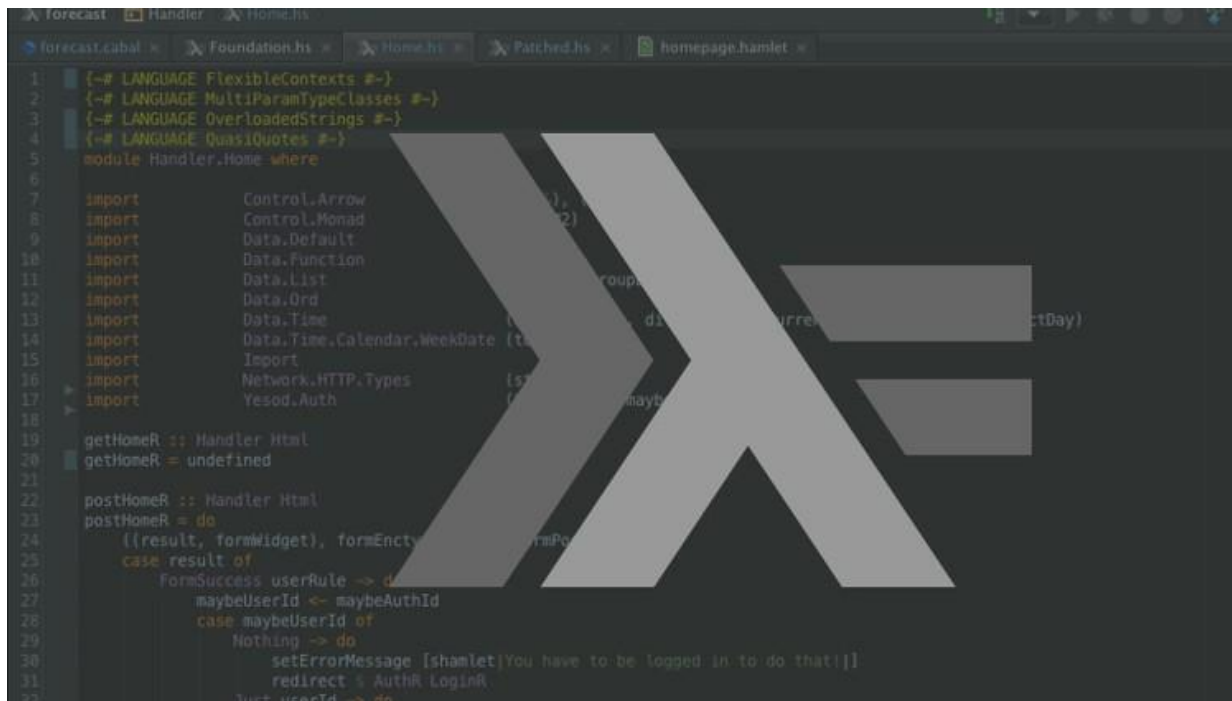


Advanced Types



```
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) <- runFormPost
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
```

Either Data Type

```

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 (toDayTime, fromDayTime, currentDay)
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) <- runFormPost
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

```

Either a b



The polymorphic type `Either a b` is predefined like this:

```
data Either a b = Left a | Right b
```

Either a b



The polymorphic type `Either a b` is predefined like this:

```
data Either a b = Left a | Right b
```

It Expresses two possibilities for a value:

- a value of type `a` (with the `Left` constructor), or
- a value of type `b` (with the `Right` constructor)

Either a b



The polymorphic type `Either a b` is predefined like this:

```
data Either a b = Left a | Right b
```

It Expresses two possibilities for a value:

- a value of type `a` (with the `Left` constructor), or
- a value of type `b` (with the `Right` constructor)

Applications:

- Indicate that a value can alternatively be of two types.
- Report an error. Usually:
 - `a` is a `String` and is the error diagnosis.
 - `b` is of the type of the expected result.
 - **Mnemonic:** right also means correct.

Examples



Let's see some examples in practise.