

Input and Output



```
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
```

Input and Output



Input/output in Haskell is based on a monad:

- The main program is `main::IO()`
- The `IO` type constructor is used to handle input/output.
- `IO` is instance of `Monad`.
- It is usually used with `do` notation.

Input and Output



Some basic operations:

```
getChar    :: IO Char      -- gets next character
getLine    :: IO String    -- gets next line
getContents :: IO String    -- get the entire input

putChar     :: Char -> IO () -- writes a character
putStr      :: String -> IO () -- writes text
putStrLn    :: String -> IO () -- writes text and a line break
print       :: Show a => a -> IO () -- writes any showable
```

`()` is a zero-field tuple and `()` is the only value of type `()`.
(\Leftrightarrow void of C).

Hello World Example



```
main = do
  putStrLn "What is your name?"
  name <- getline
  putStrLn $ "Hello " ++ name ++ "!"
```

Compilation and execution:

```
> ghc program.hs
[1 of 1] Compiling Main                ( program.hs, program.o )
Linking program ...

> ./program
What is your name?
James
Hello James!
```

Instructor Youtube Channel: Lucas Science

