haskell-scripting Documentation

Kostiantyn Rybnikov

Oct 11, 2018

Contents:

Haskell is becoming more and more easy in terms of a language in which you can write small programs that usually are launched from command-line and do some helpful work. Unlike regular programming techniques, scripting is focused on “getting things done” in a fastest possible way, so usage of techniques like erroring out early and using exceptions is not considered a bad style.

This repository’s goal is to become a one-stop shop serving as a cookbook on scripting in Haskell, going from most simple and easy towards more complicated and framework-heavy patterns, capturing most common data formats and libraries used in scripting (“most common” metric was chosen arbitrary). PRs and Issues proposing improvements are welcome!

CHAPTER 1

Prerequisites

- install the stack tool
- run `stack setup` to install GHC

1.1 Simple Scripts

Simplest script in haskell looks like this: [simple.hs](./simple.hs):

```
#!/usr/bin/env stack
-- stack --resolver=lts-12.7 script

main :: IO ()
main = do
  putStrLn "hello, scripting!"
```

Just mark this file as executable (via `chmod +x ./simple_stdin_stdout.hs`) and launch as usual:

```
$ ./simple_stdin_stdout.hs
hello, scripting!
```

You can **compile your script** to get an executable like this:

```
$ stack ghc --resolver=lts-12.7 ./simple_stdin_stdout.hs
Linking simple_stdin_stdout ...
$ ./simple_stdin_stdout
hello, scripting!
```

With a `stack script` shown above, you can use whatever packages are present in a snapshot without listing them out. For example: simple_with_dep.hs:

```
#!/usr/bin/env stack
-- stack --resolver=lts-12.7 script
```

(continues on next page)

(continued from previous page)

```
import Database.Redis

main :: IO ()
main = do
  putStrLn "hello, scripting!"
```

If you want to use a non-snapshot dependency, you can use a `stack runghc` (or `stack ghc`) command with dependency packages being listed in a `--package` argument: simple_stdin_stdout_nonsnapshot_dep.hs:

```
#!/usr/bin/env stack
-- stack --resolver=lts-12.7 --package corenlp-parser script

import NLP.CoreNLP

main :: IO ()
main = do
  putStrLn "hello, scripting!"
```

Notes/Caveats:

- TODO compiling with Wall/Werror and other flags
- buffering (per-line? per-byte? need to set buffering mode)
- best practice: upon reading/writing to stdin/stdout, usage of *String* is bad, better use *Text*, best – bytestrings (TODO: citation on encoding problems needed)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search