Single File Elixir Programs

Brendan Sechter

2016-01-12

I was reading the learn you some Erlang for great good book and realized that escript functionality is easier to use in erlang than elixir. This post covers single file elixir programs. Single file programs are useful for problems that are a little too complicated for iex, but not complicated enough for a full blown mix project.

Thanks to everyone in #elixir-lang on Freenode who answered my questions about command line arguments and escript.

Software Versions

```
$ date
January  8, 2016 at 05:35:06 PM JST
$ uname -a
FreeBSD mirage.sennue.com 11.0-CURRENT FreeBSD 11.0-CURRENT #0 r287598: Thu Sep 10 14:45:48 JST
$ elixir --version
Erlang/OTP 18 [erts-7.2.1] [source] [64-bit] [async-threads:10] [hipe] [kernel-poll:false]
Elixir 1.2.0
$ erl -version -eval '' -noshell
Erlang (ASYNC_THREADS,HIPE) (BEAM) emulator version 7.2.1
```

Instructions

All of the following scripts need to be executable to run.

```
chmod +x my_script
./my_script
```

A hello world erlang escript looks like this.

```
#!/usr/bin/env escript
main(_) ->
  io:put_chars("Hello, World! [Erlang]\n").
```

The corresponding elixir file looks like this.

```
#!/usr/bin/env elixir
IO.puts "Hello, World! [Elixir]"
```

Simple enough, but the above is only useful for programs that do not need user input. A single file erlang script that echoes command line arguments looks like this.

```
#!/usr/bin/env escript
main(Args) ->
  io:put_chars("Hello, escript!\n"),
  print_args(Args).
```

```
print_args([]) -> ok;
print_args(Args) ->
  io:put_chars("\nArgs:\n"),
  lists:map(fun(Arg) -> io:format("  ~s~n", [Arg]) end, Args).
```

The corresponding elixir file looks like this.

```
#!/usr/bin/env elixir
defmodule Script do
  def main(args) do
    IO.puts "Hello, Elixir!"
    print_args(args)
  end

  def print_args([]), do: :ok
  def print_args(args) do
    IO.puts("\nArgs:")
    args
    |> Enum.each(fn s -> IO.puts("  " <> s) end)
  end
end

Script.main(System.argv)
```

The erlang version automatically routes commandline arguments to main. The elixir version does not. Slightly unhandy for organized programs, but this allows free floating code in elixir.

To go a step beyond single file programs, a full blow elixir escript program can be tested with a mix task.

```
# lib/mix/tasks/main.ex
defmodule Mix.Tasks.Main do
  use Mix.Task

  def run(args) do
    Mix.Task.run "app.start"
    Mix.Project.get.project[:escript][:main_module].main(args)
  end
end
```

Mix.Task.run “app.start” starts all of the applications in the project. Mix.Project.get.project[:escript][:main_m runs the escipt main function with the supplied command line arguments.

This way the command line application can be tested without recompiling.

```
mix main arg_a arg_b arg_c
```

References:

- Learn You Some Erlang for Great Good!
- Erlang escript
- IRC #elixir-lang on Freenode
- Writing a Command Line Application in Elixir
- Elixir, Create Command Line Tools