Documentación

- Web Oficial
- Elixir guides
- Doc Oficial
- Elixir School (ES)
- Doc Oficial de Erlang/OTP
- Ebook Gratis – Aprendizaje Elixir Language

Modo Interactivo

Cuando tenemos instalado Elixir. Tendremos 3 ejecutables: iex, elixir, mix.

El primero que vamos a ver es iex o iex.bat en windows. Que significa Interactive Elixir. En este shell, podremos escribir cualquier expresión de Elixir y obtener su resultado.

Un ejemplo sencillo sería.

```
Erlang/OTP 21.0 [64-bit] [smp:2:2] [...]

Interactive Elixir (1.10.2) - press Ctrl+C to exit
iex(1)> 40 + 2
42
iex(2)> "hello" <> " world"
"hello world"
```

Para salir de iex, presione Ctrl + C dos veces.

Ejecutar scripts

Creamos un fichero llamado «simple.exs» y le añadimos la siguiente linea.

```
IO.puts "Hello world from Elixir"
```

Salvamos el fichero y ejecutamos en la consola.

```
$ elixir simple.exs
Hello world from Elixir
```

Crear un proyecto

Creemos nuestro primer proyecto con «mix new» desde la línea de comandos. Pasaremos el nombre del proyecto como argumento (kv, en este caso), y le diremos a Mix que nuestro módulo principal debería ser el KV en mayúsculas, en lugar del predeterminado, que habría sido Kv:

```
$ mix new kv --module KV
```

Lo que nos devolverá algo parecido a esto:

```
* creating README.md
* creating .formatter.exs
* creating .gitignore
* creating mix.exs
* creating lib
* creating lib/kv.ex
* creating test
* creating test/test_helper.exs
```

```
* creating test/kv_test.exs
```

El fichero responsable de la compilación del proyecto (Opciones, dependencias.. etc) es mix.exs y se genera automáticamente.

```
defmodule KV.MixProject do
  use Mix.Project

  def project do
    [
      app: :kv,
      version: "0.1.0",
      elixir: "~> 1.9",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about applications
  def application do
    [
      extra_applications: [:logger]
    ]
  end

  # Run "mix help deps" to learn about dependencies
  defp deps do
    [
      # {:dep_from_hexpm, "~> 0.3.0"},
      # {:dep_from_git, git: "https://github.com/elixir-lang/my_dep.git", tag: "0.1.0"},
    ]
  end
end
```

Ahora ya tenemos un proyecto. Se podría crea módulos y funciones. Para usarlo como librería de otra app. Pero queremos que se ejecute. Para ello modificamos el archivo mix.exs el apartado application. Tiene que ser algo parecido a esto.

```
  def application do
    [
      mod: {KV, []},
      extra_applications: [:logger]
    ]
```

```
  end
```

Eso hará que cuando ejecutemos la app nos llame a la función «start» del método «KV». Nos vamos a fichero «lib/kv.ex» y cambiamos su contenido que es de ejemplo por:

```
defmodule KV do
  @moduledoc """
  Documentation for `KV`.
  """

  @doc """
    Run to start app.
  """
  def start(_type, _args) do
    IO.puts "starting"
    :timer.sleep(1000)
    Task.start(fn -> :timer.sleep(1000); IO.puts("done sleeping") end)
  end
end
```

Una vez realizado esto. Ejecutamos en consola:

```
$ mix run
```

Lo que nos devolverá el siguiente texto con un pequeño delay.

```
Compiling 1 file (.ex)
starting
```