EBook Gratis

APRENDIZAJE phoenix-framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#phoenix-framework

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: phoenix-framework

It is an unofficial and free phoenix-framework ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official phoenix-framework.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con phoenix-framework

Observaciones

Esta sección proporciona una descripción general de qué es phoenix-framework y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro del marco de Phoenix, y vincular a los temas relacionados. Dado que la Documentación para phoenix-framework es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	Fecha de lanzamiento
0.1.1	2014-05-01
0.2.0	2014-05-01
0.2.1	2014-05-01
0.2.2	2014-06-05
0.2.3	2014-05-05
0.2.10	2014-05-22
0.2.11	2014-06-30
0.3.0	2014-07-01
0.3.1	2014-07-05
0.4.0	2014-08-31
0.4.1	2014-09-09
0.5.0	2014-10-14
0.6.0	2014-11-22
0.6.1	2014-11-30
0.6.2	2014-12-08
0.7.0	2014-12-10

Versión	Fecha de lanzamiento
0.7.1	2014-12-10
0.7.2	2014-12-11
0.8.0	2015-01-11
0.9.0	2015-02-12
0.10.0	2015-03-08
0.11.0	2015-04-08
0.12.0	2015-05-01
0.13.0	2015-11-15
0.13.1	2015-05-17
0.14.0	2015-06-30
0.15.0	2015-07-27
0.16.0	2015-08-06
0.16.1	2015-08-06
0.17.1	2015-08-27
1.0.0	2015-08-28
1.0.1	2015-09-03
1.0.2	2015-09-07
1.0.3	2015-09-29
1.0.4	2015-12-15
1.1.0	2015-09-16
1.1.1	2015-09-27
1.1.2	2016-01-09
1.1.3	2016-01-20
v1.2.0-rc.0	2016-04-29
v1.2.0-rc.1	2016-05-25

Versión	Fecha de lanzamiento
1.2.0	2016-06-23
1.2.2	2017-03-14
1.2.3	2017-03-15
1.2.4	2017-05-16
1.3.0-rc.1	2017-03-15
1.3.0-rc.2	2017-05-16

Examples

Instalación

El marco de Phoenix está escrito en Elixir , y el propio Elixir se basa en el lenguaje Erlang y aprovecha la máquina virtual de Erlang, conocida por ejecutar sistemas de baja latencia, distribuidos y tolerantes a fallas. Ambos lenguajes son requeridos para usar el framework phoenix. Siguiendo el siguiente paso para instalar el framework phoenix:

1. Instale Elixir en su máquina. Consulte Instalación de Elixir y cómo instalar la guía de Elixir .

2. Instale el gestor de paquetes **Hex** . Hex es una herramienta necesaria para ejecutar una aplicación de Phoenix e instalar las dependencias adicionales que podamos necesitar en el camino. Desde su terminal o ventana de control de comando, escriba:

```
$ mix local.hex
```

Este comando instalará o actualizará Hex, si ya lo tiene.

3. Instale Erlang en su máquina. Sin Erlang, el código de Elixir no se compilará porque Elixir usa la máquina virtual de Erlang para la compilación de código. Cuando instale Elixir, es probable que también haya instalado Erlang, pero si no es así, siga estas instrucciones en la guía de Elixir para instalar Erlang. Sin embargo, si tiene un sistema basado en Debian, es posible que deba instalar Erlang explícitamente.

```
$ wget https://packages.erlang-solutions.com/erlang-solutions_1.0_all.deb && sudo dpkg -i
erlang-solutions_1.0_all.deb
$ sudo apt-get update
$ sudo apt-get install esl-erlang
```

4. Instale el marco de Phoenix en su máquina. Una vez que tengamos Elixir y Erlang, estamos listos para instalar el archivo Phoenix Mix. Un archivo Mix es un archivo Zip que contiene una aplicación así como sus archivos BEAM compilados. Está vinculado a una versión específica de la aplicación. El archivo es lo que usaremos para generar una nueva aplicación básica de

Phoenix, a partir de la cual podemos construir. Aquí está el comando para instalar el archivo Phoenix:

```
$ mix archive.install https://github.com/phoenixframework/archives/raw/master/phoenix_new.ez
```

Puede descargar paquetes manualmente, si el comando anterior no funciona correctamente para usted. Descargue paquetes a su sistema de archivos de archivos Phoenix y ejecute el siguiente comando

```
mix archive.install /path/to/local/phoenix_new.ez
```

5 Plug, Cowboy y Ecto son componentes del marco de Phoenix, se instalarán automáticamente por mezcla, si dejas que Mix instale sus dependencias, cuando primero crearás proyectos de Phoenix. Además, si no permite que la mezcla descargue estos componentes, la combinación le indicará cómo hacerlo más adelante.

6. Instale Node.js (no menos que v5.0.0) en su máquina. Esta es una dependencia **opcional** . Node.js es necesario para instalar las dependencias de brunch.io . Phoenix utiliza Brunch.io para compilar activos estáticos (javascript, css, etc.), por defecto.

Podemos obtener node.js desde la página de descarga . Al seleccionar un paquete para descargar, es importante tener en cuenta que Phoenix requiere la versión 5.0.0 o superior.

Los usuarios de Mac OS X también pueden instalar node.js a través de homebrew .

Nota: no se sabe si io.js, que es una plataforma compatible con npm originalmente basada en Node.js, funciona con Phoenix.

Los usuarios de Debian / Ubuntu pueden ver un error que se ve así:

```
sh: 1: node: not found
npm WARN This failure might be due to the use of legacy binary "node"
```

Esto se debe a que Debian tiene binarios en conflicto para el nodo: vea la discusión sobre la siguiente pregunta SO

No se pueden instalar paquetes usando el administrador de paquetes de nodo en Ubuntu

Hay dos opciones para solucionar este problema, ya sea:

instalar nodejs-legacy:

```
$ apt-get install nodejs-legacy
```

o crear un enlace simbólico

```
$ ln -s /usr/bin/nodejs /usr/bin/node
```

7 Instale la base de datos (PostgreSQL) en su máquina. Phoenix configura las aplicaciones para usarlas de forma predeterminada, pero podemos cambiar a MySQL al pasar la `--database mysql` al crear una nueva aplicación. El wiki de PostgreSQL tiene guías de instalación para varios sistemas diferentes.

Postgrex es una dependencia directa de Phoenix y se utilizará para crear modelos. **Postgrex** se instalará automáticamente junto con el resto de dependencias cuando creará e iniciará el proyecto Phoenix.

8 inotify-tools (para usuarios de Linux) Este es un observador del sistema de archivos solo para Linux que Phoenix usa para la recarga de código en vivo. (Los usuarios de Mac OS X o Windows pueden ignorarlo).

Los usuarios de Linux necesitan instalar esta dependencia. Consulte la wiki de herramientas de inotificación para obtener instrucciones de instalación específicas de distribución.

Instalación de esqueleto

A veces desea una instalación sin nada excepto la configuración mínima de Phoenix. El siguiente comando te dará eso.

```
mix phoenix.new web --no-brunch --no-ecto
```

Nota: debe haber instalado Elixir, Erlang, Hex, Mix y el archivo Phoenix para la instalación de esqueleto

Creando proyecto Phoenix

Para crear su primer proyecto en el marco de Phoenix en este punto **, debe tener** instalados Elixir, Erlang, Hex y el archivo de Phoenix. También debe tener PostgreSQL y node.js instalados para crear una aplicación predeterminada.

Abra el terminal o el símbolo del sistema y vaya a la ubicación en su sistema de archivos donde desea **crear la aplicación** . `phoenix.new` es el comando mix que creará un nuevo proyecto para ti. Suponiendo que el nombre de nuestra aplicación es `hello_phoenix_world` , escriba

```
$ mix phoenix.new hello_phoenix_world
```

Alternativamente , podemos ejecutar mix phoenix.new desde cualquier directorio para iniciar nuestra aplicación Phoenix. Phoenix aceptará una ruta absoluta o relativa para el directorio de nuestro nuevo proyecto

```
$ mix phoenix.new /Users/username/work/elixir-projects/hello_phoenix_world
```

Salida

```
mix phoenix.new hello_phoenix_world
* creating hello_phoenix_world/config/config.exs
```

```
* creating hello_phoenix_world/config/dev.exs
* creating hello_phoenix_world/config/prod.exs
...
* creating hello_phoenix_world/web/views/layout_view.ex
* creating hello_phoenix_world/web/views/page_view.ex

Fetch and install dependencies? [Yn]
```

Phoenix generará la estructura de directorios para su proyecto y creará todos los archivos necesarios para la aplicación. Mix le preguntará si desea **instalar otras dependencias requeridas** . Digamos que sí a eso.

```
Fetch and install dependencies? [Yn] Y
* running mix deps.get
* running npm install && node node_modules/brunch/bin/brunch build
```

Una vez que **se instalen las dependencias** , la tarea le pedirá que cambie a nuestro directorio de proyecto e inicie la aplicación.

```
Move into your new project folder:

    $cd hello_phoenix_world
```

Ahora debe configurar el nombre de usuario y la contraseña de postgres, a menos que ya esté configurado con la contraseña y el usuario de postgres predeterminados. Edite su archivo `config/dev.exs` y configure el nombre de usuario y la contraseña:

```
# config/dev.exs
config :hello_phoenix_world, HelloPhoenixWorld.Repo,
  adapter: Ecto.Adapters.Postgres,
  username: "postgres",
  password: "postgres",
  database: "hello_phoenix_world_dev",
  hostname: "localhost",
  pool_size: 10

Now, create the database with the ecto mix task:

    $ mix ecto.create

We have a working application! Run your Phoenix application:

    $ mix phoenix.server

You can also run your app inside IEx (Interactive Elixir) as:

    $ iex -S mix phoenix.server

Load `http://localhost:4000` into your browser and you will see the default landing page of
your application.
```

Ahora, vamos a agregar hola mundo a la aplicación Phoenix. Abra el archivo `web/templates/page/index.html.eex` y reemplace los contenidos con lo siguiente y guarde el archivo:

```
<h2>Hello World</h2>
```

Si no ha abandonado el servidor, el nuevo código se compilará automáticamente y su navegador ahora debería mostrar su mensaje "Hola mundo".

Ahora puede crear un recurso CRUD .

Finalmente, para salir del servidor, escriba `ctrl-c crtl-c` (presione la `control key` y la tecla `c` juntas) dos veces seguidas.

Ejecutando Elixir / Phoenix en OSX

Elixir / fénix

Instale Homebrew primero:

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Luego, ejecutar `brew install elixir` instalará tanto el elixir como su dependencia: Erlang.

Instalar la mezcla con la `mix local.hex` .

Instale Phoenix según las instrucciones:

```
mix archive.install https://github.com/phoenixframework/archives/raw/master/phoenix_new.ez
```

Node.js

Puede instalar y administrar sus versiones de Node.js con NVM. Instala nvm con:

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.31.4/install.sh | bash
```

Si no está disponible el `curl` , puede instalarlo con el `brew install curl` . Entonces corre:

```
nvm install node
```

para descargar y compilar y la última versión de Node.js.

Base de datos

Descarga Postgres.app y ejecútalo . Cuando cree su proyecto Phoenix, en su archivo `config/dev.exs` , solo debe proporcionar un nombre para su base de datos; el adaptador usará los valores predeterminados para el resto:

```
config :myphoenixapp, MyPhoenixApp.Repo,
  adapter: Ecto.Adapters.Postgres,
  database: "myphoenixapp_dev",
  hostname: "localhost",
```

```
  pool_size: 10
```

Generando recursos para un modelo.

Para generar el esquema, la vista, el controlador, el archivo de migración para el repositorio, las plantillas de CRUD predeterminadas y los archivos de prueba para un modelo (como un andamio en Rails) se puede usar la tarea de mezcla `phoenix.gen.html` como esta:

```
mix phoenix.gen.html Book books title note:text pages:integer author_id:references:authors
```

Donde `Book` es el nombre del módulo, `books` se forma plural se utiliza para el esquema, seguida de campos de recursos: `title` (cadena por defecto), `note` (campo de texto), `pages` (entero), `author_id` que crea un `belongs_to` asociación con el modelo de Autor.

Lea Empezando con phoenix-framework en línea: https://riptutorial.com/es/phoenix-framework/topic/4996/empezando-con-phoenix-framework

Capítulo 2: Generar documentación del proyecto.

Examples

Razón fundamental

La correcta invocación de módulos de ayuda y funciones puede ser intimidante porque

- estos se generan dinámicamente (por ejemplo, cuando se crea un nuevo proyecto o se agrega un nuevo `resource`)
- no están documentados explícitamente (por ejemplo, `MyApp.ErrorHelpers.error_tag`)
- la documentación no cubre todos los ejemplos (por ejemplo, `MyApp.Router.Helpers.*_path` en `Phoenix.Router`).

Aunque los ayudantes creados están dispersos por todo el proyecto, su ubicación sigue una lógica sólida. Puede acostumbrarse a ellos bastante rápido y, afortunadamente, cuando genera un proyecto con Phoenix, el código se envía con la documentación a través de los atributos de los módulos `@doc` y `@moduledoc` Elixir.

Estos documentos no están limitados solo a ayudantes, sino que también puede

- vea su proyecto desglosado por submódulos / funciones / macros
- añade tu propia documentación
- busque las funciones que se generaron bajo el espacio de nombres de su proyecto (por ejemplo, `MyApp.Repo` contiene implementaciones de función de devolución de llamada de `Ecto.Repo`)

Generando los documentos

Para generar documentación a partir de su código fuente, agregue `ex_doc` como dependencia en su archivo `mix.exs` :

```
# config/mix.exs

def deps do
  [{:ex_doc, "~> 0.11", only: :dev}]
end
```

> Puede utilizar Markdown dentro de los `@doc` Elixir `@doc` y `@moduledoc` .

Luego, ejecute `mix deps.get` para obtener y compilar los nuevos módulos y generar la documentación del proyecto con documentos `mix docs` . Un ejemplo de salida es el oficial Elixir Docs .

Para servirlos de inmediato, utilice `mix docs --output priv/static/doc` y navegue hasta

`my_app_url_or_ip/doc/index.html` .

Lectura adicional:

- `ex_doc`
- Operadores de requisitos de versión (`Elixir.Version`)

El grueso de esta guía está referenciado en Recetas de Elixir .

Lea Generar documentación del proyecto. en línea: https://riptutorial.com/es/phoenix-framework/topic/5868/generar-documentacion-del-proyecto-

Capítulo 3: Uso de modelos ecto en phoenix.

Introducción

Cómo generar, editar y utilizar modelos ecto en los marcos de Phoenix.

Examples

Generar modelo de usuario desde la línea de comando

Para generar un modelo de usuario `json` con `username` , `password_hash` , `email_id` , `created_at` , `updated_at` , escriba

```
mix phoenix.gen.json User users username:string email_id:string password_hash:string
timestamps()
```

Migraciones del modelo ecto.

Cuando ejecuta `mix phoenix.gen.html` o `mix phoenix.gen.json` desde la línea de comandos, las migraciones se crean en `priv -> repo -> migrations` en su carpeta de proyecto.

Para ejecutar migraciones escriba `mix ecto.migrate` .

Para generar migraciones para su `mix ecto.gen migrations <model_name>` proyectos `mix ecto.gen migrations <model_name>`

Para generar migraciones para un repositorio diferente al predeterminado, ejecute una `mix ecto.gen migrations <model_name> -r <repo_name>`

Lea Uso de modelos ecto en phoenix. en línea: https://riptutorial.com/es/phoenix-framework/topic/10890/uso-de-modelos-ecto-en-phoenix-

Creditos

S. No	Capítulos	Contributors
1	Empezando con phoenix-framework	Community, helcim, penguin, Steve Pallen, SURAJ KUMAR, Svilen
2	Generar documentación del proyecto.	toraritte
3	Uso de modelos ecto en phoenix.	Faizan Ali