EBook Gratuito

APPRENDIMENTO phoenix-framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#phoenix-framework

Sommario

Di

Capitolo 1: Iniziare con phoenix-framework

Osservazioni

Questa sezione fornisce una panoramica di ciò che phoenix-framework è, e perché uno sviluppatore potrebbe voler usarlo.

Dovrebbe anche menzionare eventuali soggetti di grandi dimensioni all'interno di phoenix-framework e collegarsi agli argomenti correlati. Poiché la documentazione di phoenix-framework è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data di rilascio
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
0.7.1	2014/12/10

Versione	Data di rilascio
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
1.2.0	2016/06/23

Versione	Data di rilascio
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

Installazione

Il framework Phoenix è scritto in Elixir e l'elisir stesso è basato sul linguaggio Erlang e sfrutta la VM Erlang, nota per l'esecuzione di sistemi a bassa latenza, distribuiti e fault-tolerant. Entrambe le lingue sono necessarie per l'utilizzo di framework phoenix. Seguendo il seguente passaggio per installare phoenix framework:

1. Installa Elixir sulla tua macchina. Vedi Installazione di elisir e come installare la guida di Elixir .

2. Installa il gestore di pacchetti **Hex** . Hex è uno strumento necessario per far funzionare un'app Phoenix e per installare eventuali dipendenze aggiuntive che potrebbero essere necessarie lungo il percorso. Dal terminale o dalla finestra di controllo comandi, digitare:

```
$ mix local.hex
```

Questo comando installerà o aggiornerà Hex, se già lo hai.

3. Installa Erlang sulla tua macchina. Senza Erlang, il codice Elixir non verrà compilato perché Elixir usa la VM di Erlang per la compilazione del codice. Quando installerai Elixir, probabilmente avrai installato anche Erlang, ma se non è così, segui queste istruzioni sulla guida di Elixir per installare Erlang. Tuttavia, se si dispone di un sistema basato su Debian, potrebbe essere necessario installare esplicitamente Erlang.

```
$ 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. Installa phoenix framework sulla tua macchina. Una volta che abbiamo Elixir ed Erlang, siamo pronti per installare l'archivio di Phoenix Mix. Un archivio Mix è un file Zip che contiene un'applicazione e i file BEAM compilati. È legato a una versione specifica dell'applicazione. L'archivio è ciò che useremo per generare una nuova applicazione base Phoenix da cui possiamo costruire. Ecco il comando per installare l'archivio Phoenix:

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

Puoi scaricare i pacchetti manualmente, se il comando precedente non funziona correttamente. Scarica i pacchetti nel tuo file system Archivi Phoenix ed esegui il seguente comando

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

5 Plug, Cowboy ed Ecto sono componenti del framework Phoenix, verranno installati automaticamente da mix, se si consente a mix di installare le sue dipendenze, quando si creeranno per la prima volta i progetti Phoenix. Inoltre, se non permetti a mix di scaricare questi componenti, mix ti spiegherà come farlo in seguito.

6. Installa Node.js (non meno di v5.0.0) sul tuo computer. Questa è una dipendenza **opzionale** . Node.js è necessario per installare le dipendenze di brunch.io . Brunch.io è utilizzato da Phoenix per la compilazione di risorse statiche (javascript, css, ecc.), Per impostazione predefinita.

Possiamo ottenere node.js dalla pagina di download . Quando si seleziona un pacchetto da scaricare, è importante notare che Phoenix richiede la versione 5.0.0 o successiva.

Gli utenti Mac OS X possono anche installare node.js tramite homebrew .

Nota: io.js, che è una piattaforma compatibile con NPM originariamente basata su Node.js, non è nota per funzionare con Phoenix.

Gli utenti Debian / Ubuntu potrebbero vedere un errore simile a questo:

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

Ciò è dovuto al fatto che Debian abbia binari in conflitto per il nodo: vedere la discussione sulla seguente domanda SO

Non è possibile installare pacchetti usando il gestore di pacchetti di nodi in Ubuntu

Esistono due opzioni per risolvere questo problema:

installa nodejs-legacy:

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

o creare un collegamento simbolico

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

7 Installa Database (PostgreSQL) sul tuo computer. Phoenix configura le applicazioni per usarle di default, ma possiamo passare a MySQL passando il flag `--database mysql` durante la creazione di una nuova applicazione. Il wiki PostgreSQL ha guide di installazione per un certo numero di sistemi diversi.

Postgrex è una dipendenza diretta di Phoenix e verrà utilizzata per creare modelli. **Postgrex** verrà installato automaticamente insieme al resto delle dipendenze quando creerai e **avvierai un** progetto Phoenix.

8 strumenti inotify (per utenti Linux) Si tratta di un watcher per filesystem Linux-only che Phoenix usa per ricaricare il codice live. (Gli utenti Mac OS X o Windows possono tranquillamente ignorarlo).

Gli utenti Linux devono installare questa dipendenza. Consultare il wiki di inotify-tools per le istruzioni di installazione specifiche della distribuzione.

Installazione scheletrica

A volte si desidera un'installazione senza nulla tranne l'impostazione phoenix minima nuda. Il comando seguente ti darà questo.

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

Nota: è necessario aver installato Elixir, Erlang, Hex, Mix e l'archivio Phoenix per l'installazione di scheletro

Creare un progetto Phoenix

Per creare il tuo primo progetto nel framework Phoenix, a questo punto **dovresti avere installato** Elixir, Erlang, Hex e l'archivio Phoenix. Dovresti inoltre avere PostgreSQL e node.js installati per creare un'applicazione predefinita.

Aprire il terminale o il prompt dei comandi e accedere al percorso sul file system in cui si desidera **creare l'applicazione** . `phoenix.new` è il comando mix che creerà un nuovo progetto per te. Supponendo che il nome della nostra applicazione sia `hello_phoenix_world` , quindi digita

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

In alternativa , possiamo eseguire il mix phoenix.new da qualsiasi directory per avviare la nostra applicazione Phoenix. Phoenix accetterà un percorso assoluto o relativo per la directory del nostro nuovo progetto

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

Produzione

```
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 genererà la struttura di directory per il tuo progetto e creerà tutti i file necessari per l'applicazione. Mix ti chiederà se vuoi **installare altre dipendenze richieste** . Diciamo di sì.

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

Una volta **installate le dipendenze** , l'attività richiederà di cambiare nella directory del progetto e avviare l'applicazione.

```
Move into your new project folder:

    $cd hello_phoenix_world
```

Ora è necessario impostare il nome utente e la password postgres a meno che non sia già configurato con la password postgres e postgres postgres predefinita. Modifica il tuo file `config/dev.exs` e imposta il nome utente e la password:

```
# 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.
```

Ora, aggiungiamo Hello World all'applicazione Phoenix. Apri il file `web/templates/page/index.html.eex` e sostituisci il contenuto con quanto segue e salva il file:

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

Se non si è usciti dal server, il nuovo codice verrà automaticamente compilato e il browser dovrebbe ora visualizzare il messaggio "Hello World".

Ora puoi creare una risorsa CRUD .

Infine, per uscire dal server, digitare `ctrl-c crtl-c` (premere il `control key` e il tasto `c` insieme) due volte di seguito.

Esecuzione di Elixir / Phoenix su OSX

Elisir / Phoenix

Installa Homebrew prima:

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

Quindi eseguendo `brew install elixir` installerà sia Elixir che la sua dipendenza, Erlang.

Installa mix con `mix local.hex` .

Installa Phoenix come da istruzioni:

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

Node.js

Puoi installare e gestire le tue versioni di Node.js con NVM. Installa nvm con:

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

Se `curl` non è disponibile, è possibile installarlo con `brew install curl` . Quindi esegui:

```
nvm install node
```

per scaricare e compilare e l'ultima versione di Node.js.

Banca dati

Scarica Postgres.app ed eseguilo . Quando crei il tuo progetto Phoenix, nel tuo file `config/dev.exs` , devi solo fornire un nome per il tuo database - l'adattatore userà i valori predefiniti per il resto:

```
config :myphoenixapp, MyPhoenixApp.Repo,
  adapter: Ecto.Adapters.Postgres,
  database: "myphoenixapp_dev",
  hostname: "localhost",
  pool_size: 10
```

Generazione di risorse per un modello

Per generare schema, vista, controller, file di migrazione per il repository, modelli CRUD predefiniti e file di test per un modello (come un ponteggio in Rails), è possibile utilizzare l'attività mix `phoenix.gen.html` questo modo:

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

Dove `Book` è il nome del modulo, i `books` sono la forma plurale utilizzata per lo schema, seguita dai campi delle risorse: `title` (stringa per impostazione predefinita), `note` (campo di testo), `pages` (numero intero), `author_id` che crea un'associazione `belongs_to` al modello Autore.

Leggi Iniziare con phoenix-framework online: https://riptutorial.com/it/phoenix-framework/topic/4996/iniziare-con-phoenix-framework

Capitolo 2: Genera documentazione di progetto

Examples

Fondamento logico

L'invocazione corretta dei moduli e delle funzioni di supporto può intimidire perché

- questi sono generati dinamicamente (ad esempio, quando si crea un nuovo progetto o si aggiunge una nuova `resource`)
- non sono documentati esplicitamente (es. `MyApp.ErrorHelpers.error_tag`)
- la documentazione non copre tutti gli esempi (ad es `MyApp.Router.Helpers.*_path` in `Phoenix.Router`).

Sebbene gli helper creati siano sparsi per tutto il progetto ma la loro posizione segue una logica solida. Puoi `@doc` a loro abbastanza velocemente e fortunatamente, quando generi un progetto con Phoenix, il codice viene spedito con la documentazione tramite gli attributi del modulo `@doc` e `@moduledoc` di Elixir.

Questi documenti non sono limitati ai soli aiutanti, ma puoi anche farlo

- vedi il tuo progetto suddiviso per sottomoduli / funzioni / macro
- aggiungi la tua documentazione
- cercare tutte le funzioni che sono state generate sotto lo spazio dei nomi del tuo progetto (ad esempio, `MyApp.Repo` contiene implementazioni di funzioni di callback da `Ecto.Repo`)

Generare i documenti

Per generare documentazione dal tuo codice sorgente, aggiungi `ex_doc` come dipendenza nel tuo file `mix.exs` :

```
# config/mix.exs

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

> Puoi utilizzare Markdown negli `@doc` Elixir `@doc` e `@moduledoc` .

Quindi, esegui `mix deps.get` per recuperare e compilare i nuovi moduli e generare la documentazione del progetto con i documenti di `mix docs` . Un esempio di output è il documento ufficiale di Elixir .

Per servirli immediatamente usa i `mix docs --output priv/static/doc` e vai a `my_app_url_or_ip/doc/index.html` .

Letture aggiuntive:

- `ex_doc`
- Operatori del requisito della versione (`Elixir.Version`)

La maggior parte di questa guida fa riferimento alle Ricette con elisir .

Leggi Genera documentazione di progetto online: https://riptutorial.com/it/phoenix-framework/topic/5868/genera-documentazione-di-progetto

Capitolo 3: Uso di modelli Ecto in phoenix

introduzione

Come generare, modificare e utilizzare i modelli ecto nei framework phoenix.

Examples

Generare il modello utente dalla riga di comando

Per generare un modello utente `json` con `username` , `password_hash` , `email_id` , `created_at` , `updated_at` , type

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

Migrazioni del modello ecto

Quando si esegue il `mix phoenix.gen.html` o si `mix phoenix.gen.json` dalla riga di comando, le migrazioni vengono create in `priv -> repo -> migrations` nella cartella del progetto.

Per eseguire le migrazioni digitare `mix ecto.migrate` .

Per generare migrazioni per il tuo progetto, `mix ecto.gen migrations <model_name>`

Per generare migrazioni per un repository diverso da quello predefinito, si eseguono le `mix ecto.gen migrations <model_name> -r <repo_name>`

Leggi Uso di modelli Ecto in phoenix online: https://riptutorial.com/it/phoenix-framework/topic/10890/uso-di-modelli-ecto-in-phoenix

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con phoenix-framework	Community, helcim, penguin, Steve Pallen, SURAJ KUMAR, Svilen
2	Genera documentazione di progetto	toraritte
3	Uso di modelli Ecto in phoenix	Faizan Ali