Kostenloses eBook

LERNEN phoenix-framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#phoenix-framework

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit dem Phoenix-Framework

Bemerkungen

Dieser Abschnitt bietet einen Überblick über das, was phoenix-framework ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen innerhalb des Phoenix-Frameworks erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für das Phoenix-Framework neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Versionen

Ausführung	Veröffentlichungsdatum
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	30.06.2014
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

Ausführung	Veröffentlichungsdatum
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

Ausführung	Veröffentlichungsdatum
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

Installation

Das Phoenix-Framework ist in Elixir geschrieben. Elixir selbst basiert auf der Sprache Erlang und nutzt die Erlang-VM, die dafür bekannt ist, Systeme mit niedriger Latenz, verteilte und fehlertolerante Systeme auszuführen. Für die Verwendung von Phoenix Framework sind beide Sprachen erforderlich. Folgender Schritt zur Installation von Phoenix Framework:

1. Installieren Sie Elixir auf Ihrem Computer. Siehe Elixir-Installation und Informationen zur Installation der Elixir-Anleitung .

2. Installieren Sie den Hex- Paketmanager. Hex ist ein notwendiges Werkzeug, um eine Phoenix-App zum Laufen zu bringen und um zusätzliche Abhängigkeiten zu installieren, die wir unterwegs benötigen. Geben Sie im Terminal- oder Befehlssteuerungsfenster Folgendes ein:

```
$ mix local.hex
```

Dieser Befehl wird Hex installieren oder aktualisieren, falls Sie dies bereits getan haben.

3. Installieren Sie Erlang auf Ihrem Computer. Ohne Erlang kann Elixir-Code nicht kompiliert werden, da Elixir die VM von Erlang für die Code-Kompilierung verwendet. Wenn Sie Elixir installieren, haben Sie wahrscheinlich auch Erlang installiert. Falls dies nicht der Fall ist, befolgen Sie diese Anweisungen in der Elixir-Anleitung, um Erlang zu installieren. Wenn Sie jedoch ein Debian-basiertes System haben, müssen Sie Erlang möglicherweise explizit installieren.

```
$ 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. Installieren Sie das Phoenix Framework auf Ihrem Computer. Sobald wir Elixir und Erlang haben, können wir das Phoenix Mix-Archiv installieren. Ein Mix-Archiv ist eine Zip-Datei, die eine Anwendung sowie die kompilierten BEAM-Dateien enthält. Sie ist an eine bestimmte Version der

Anwendung gebunden. Das Archiv wird verwendet, um eine neue Basisanwendung von Phoenix zu erstellen, aus der wir erstellen können. Hier ist der Befehl zum Installieren des Phoenix-Archivs:

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

Sie können Pakete manuell herunterladen, wenn der obige Befehl für Sie nicht ordnungsgemäß funktioniert. Laden Sie Pakete in Ihr Dateisystem Phoenix-Archive herunter und führen Sie den folgenden Befehl aus

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

5 Plug, Cowboy und Ecto sind Komponenten des Phoenix-Frameworks. Sie werden automatisch von mix installiert, wenn Sie mix ihre Abhängigkeiten installieren lassen, wenn Sie zum ersten Mal Phoenix-Projekte erstellen. Wenn Sie Mix nicht zulassen, dass diese Komponenten heruntergeladen werden, erfahren Sie später, wie dies geschehen soll.

6. Installieren Sie Node.js (nicht weniger als Version 5.0.0) auf Ihrem Computer. Dies ist eine **optionale** Abhängigkeit. Zur Installation der Abhängigkeiten von brunch.io ist Node.js erforderlich. Brunch.io wird von Phoenix standardmäßig zum Kompilieren statischer Assets (Javascript, CSS usw.) verwendet.

Wir können node.js von der Download-Seite erhalten . Bei der Auswahl eines Pakets zum Herunterladen ist zu beachten, dass Phoenix Version 5.0.0 oder höher benötigt.

Benutzer von Mac OS X können node.js auch über Homebrew installieren.

Hinweis: io.js, eine npm-kompatible Plattform, die ursprünglich auf Node.js basiert, funktioniert nicht mit Phoenix.

Debian / Ubuntu-Benutzer sehen möglicherweise einen Fehler, der wie folgt aussieht:

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

Dies liegt daran, dass Debian in Konflikt stehende Binärdateien für Knoten hat: siehe die folgende SO-Frage

Pakete können nicht mit dem Knotenpaket-Manager in Ubuntu installiert werden

Es gibt zwei Möglichkeiten, dieses Problem zu beheben:

Installiere nodejs-Legacy:

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

oder erstellen Sie einen Symlink

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

7 Installieren Sie die Datenbank (PostgreSQL) auf Ihrem Computer. Phoenix konfiguriert Anwendungen standardmäßig so, dass sie standardmäßig verwendet wird. Wir können jedoch zu MySQL wechseln, indem `--database mysql` beim Erstellen einer neuen Anwendung das `--database mysql` Flag `--database mysql` . Das PostgreSQL-Wiki enthält Installationsanleitungen für verschiedene Systeme.

Postgrex ist eine direkte Abhängigkeit von Phoenix und wird zur Erstellung von Modellen verwendet. **Postgrex** wird automatisch zusammen mit den übrigen Abhängigkeiten installiert, wenn Sie das Phoenix-Projekt erstellen und starten.

8 inotify-tools (für Linux-Benutzer) Hierbei handelt es sich um ein Linux-only-Dateisystem-Watcher, das Phoenix für das Live-Laden von Code verwendet. (Benutzer von Mac OS X oder Windows können dies ignorieren.)

Linux-Benutzer müssen diese Abhängigkeit installieren. Bitte konsultieren Sie das inotify-tools-Wiki, um eine verteilungsspezifische Installationsanweisung zu erhalten.

Skeleton Installation

Manchmal möchten Sie eine Installation ohne etwas anderes als die minimale Phönix-Installation. Der folgende Befehl gibt Ihnen das.

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

Hinweis: Sie müssen Elixir, Erlang, Hex, Mix und das Phoenix-Archiv für die Skeleton-Installation installiert haben

Phoenix-Projekt erstellen

Für Ihr erstes Projekt in Phoenix Rahmen an dieser Stelle zu schaffen , **sollten Sie haben,** Elixir, Erlang, Hex und das installierte Phoenix - Archiv. Sie sollten auch PostgreSQL und node.js installiert haben, um eine Standardanwendung zu erstellen.

Öffnen Sie das Terminal oder die Eingabeaufforderung, und navigieren Sie zu dem Ort in Ihrem Dateisystem, an dem Sie die **Anwendung erstellen** möchten. `phoenix.new` ist der Mix-Befehl, der ein neues Projekt für Sie erstellt. Angenommen, der Name unserer Anwendung lautet `hello_phoenix_world` , und geben Sie dann ein

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

Alternativ können wir mix phoenix.new in einem beliebigen Verzeichnis ausführen, um unsere Phoenix-Anwendung zu booten. Phoenix akzeptiert entweder einen absoluten oder einen relativen Pfad für das Verzeichnis unseres neuen Projekts

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

Ausgabe

```
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 erstellt die Verzeichnisstruktur für Ihr Projekt und erstellt alle für die Anwendung erforderlichen Dateien. Mix fragt Sie, ob Sie **andere erforderliche Abhängigkeiten installieren** möchten. Sagen wir ja dazu.

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

Nach der **Installation von Abhängigkeiten werden** Sie aufgefordert, in unser Projektverzeichnis zu wechseln und die Anwendung zu starten.

```
Move into your new project folder:

    $cd hello_phoenix_world
```

Sie müssen jetzt den Postgres-Benutzernamen und das Kennwort einrichten, sofern er nicht bereits mit dem Standardbenutzer- und Postgres-Kennwort eingerichtet wurde. Bearbeiten Sie Ihre `config/dev.exs` Datei und legen Sie den Benutzernamen und das Passwort fest:

```
# 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.
```

Jetzt können wir der Phoenix-Anwendung eine hallo-Welt hinzufügen. Öffnen Sie die Datei `web/templates/page/index.html.eex` , ersetzen Sie den Inhalt durch Folgendes und speichern Sie die Datei:

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

Wenn Sie den Server nicht beendet haben, wird der neue Code automatisch kompiliert und Ihr Browser sollte jetzt die Meldung "Hello World" anzeigen.

Sie können jetzt eine CRUD-Ressource erstellen .

`ctrl-c` zum Verlassen des Servers zum `ctrl-c` zweimal hintereinander `ctrl-c crtl-c` (drücken Sie die `control key` und die `c` Taste).

Elixir / Phoenix unter OSX ausführen

Elixier / Phoenix

Installieren Sie zuerst Homebrew :

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

Bei der Ausführung von `brew install elixir` werden dann sowohl Elixir als auch seine Abhängigkeit installiert - Erlang.

Mix mit `mix local.hex` .

Installieren Sie Phoenix gemäß den Anweisungen:

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

Node.js

Sie können Ihre Node.js-Versionen mit NVM installieren und verwalten. Installiere nvm mit:

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

Wenn `curl` nicht verfügbar ist, können Sie es mit `brew install curl` . Dann renne:

```
nvm install node
```

Herunterladen und Kompilieren der neuesten Version von Node.js.

Datenbank

Laden Sie Postgres.app herunter und führen Sie es aus. Wenn Sie Ihr Phoenix-Projekt in Ihrer Datei `config/dev.exs` erstellen, müssen Sie nur einen Namen für Ihre Datenbank `config/dev.exs` Der Adapter verwendet für den Rest Standardwerte:

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

Ressourcen für ein Modell generieren

Um Schema, Ansicht, Controller, Migrationsdatei für das Repository, Standard-CRUD-Vorlagen und Testdateien für ein Modell (wie ein Gerüst in Rails) zu `phoenix.gen.html` können Sie die `phoenix.gen.html` Task wie `phoenix.gen.html` :

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

Wo `Book` der Modulname ist, sind `books` Pluralformen, die für ein Schema verwendet werden, gefolgt von Ressourcenfeldern: `title` (standardmäßig Zeichenfolge), `note` (Textfeld), `pages` (Ganzzahl), `author_id` , wodurch eine `belongs_to` zu Autor erstellt wird.

Erste Schritte mit dem Phoenix-Framework online lesen: https://riptutorial.com/de/phoenix-framework/topic/4996/erste-schritte-mit-dem-phoenix-framework

Kapitel 2: Projektdokumentation erstellen

Examples

Begründung

Der korrekte Aufruf von Hilfsmodulen und Funktionen kann einschüchternd sein, weil

- Diese werden dynamisch generiert (z. B. beim Erstellen eines neuen Projekts oder Hinzufügen einer neuen `resource`).
- Sie werden nicht explizit dokumentiert (z. B. `MyApp.ErrorHelpers.error_tag`).
- Die Dokumentation deckt nicht alle Beispiele ab (z `MyApp.Router.Helpers.*_path` B. `MyApp.Router.Helpers.*_path` in `Phoenix.Router`).

Die erstellten Helfer sind zwar im gesamten Projekt verstreut, ihr Standort folgt jedoch einer festen Logik. Sie können sich recht schnell daran gewöhnen, wenn Sie ein Projekt mit Phoenix generieren. Der Code wird mit der Dokumentation über die Elixir-Attribute `@doc` und `@moduledoc` .

Diese Dokumente sind nicht nur auf Helfer beschränkt, sondern auch Sie können dies tun

- sehen Sie Ihr Projekt nach Submodulen / Funktionen / Makros auf
- fügen Sie Ihre eigene Dokumentation hinzu
- `MyApp.Repo` Funktionen, die im Namensraum Ihres Projekts generiert wurden (z. `MyApp.Repo` enthält `MyApp.Repo` Callback-Funktionsimplementierungen von `Ecto.Repo`).

Generieren der Dokumente

Um aus Ihrem Quellcode Dokumentation zu generieren, fügen Sie `ex_doc` als Abhängigkeit in Ihre `mix.exs` Datei ein:

```
# config/mix.exs

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

> Sie können Markdown in den Attributen Elixir `@doc` und `@moduledoc` .

Führen Sie anschließend `mix deps.get` , um die neuen Module `mix deps.get` zu kompilieren und die Projektdokumentation mit `mix docs` generieren. Ein Beispiel für eine Ausgabe sind die offiziellen Elixir Docs .

Um sie sofort zu bedienen, verwenden Sie `mix docs --output priv/static/doc` und navigieren Sie zu `my_app_url_or_ip/doc/index.html` .

Zusätzliche Lektüre:

- `ex_doc`

- Versionsanforderungsoperatoren (`Elixir.Version`)

Der Großteil dieses Handbuchs wird aus Elixir-Rezepten referenziert.

Projektdokumentation erstellen online lesen: https://riptutorial.com/de/phoenix-framework/topic/5868/projektdokumentation-erstellen

Kapitel 3: Verwendung von Ecto-Modellen in Phoenix

Einführung

Erstellen, Bearbeiten und Verwenden von Ecto-Modellen in Phoenix-Frameworks

Examples

Generieren Sie das Benutzermodell über die Befehlszeile

Um ein `json` Benutzermodell mit `username` , `password_hash` , `email_id created_at email_id` , `created_at` , `updated_at email_id` , geben `created_at updated_at`

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

Migrationen des ecto-Modells

Wenn Sie `mix phoenix.gen.html` oder `mix phoenix.gen.json` über die Befehlszeile `mix phoenix.gen.json` , werden Migrationen in `priv -> repo -> migrations` in Ihrem Projektordner erstellt.

Um Migrationen `mix ecto.migrate` .

Um Migrationen für Ihr Projektmix zu generieren, `mix ecto.gen migrations <model_name>`

Um Migrationen für ein anderes Repository als das Standardmaskensystem zu generieren, führen Sie `mix ecto.gen migrations <model_name> -r <repo_name>`

Verwendung von Ecto-Modellen in Phoenix online lesen: https://riptutorial.com/de/phoenix-framework/topic/10890/verwendung-von-ecto-modellen-in-phoenix

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit dem Phoenix-Framework	Community, helcim, penguin, Steve Pallen, SURAJ KUMAR, Svilen
2	Projektdokumentation erstellen	toraritte
3	Verwendung von Ecto-Modellen in Phoenix	Faizan Ali