1. 启动和运行（Up and Running）

首个教程的目的是让Phoenix应用程序尽可能快地启动和运行。

在我们开始之前，请花一分钟阅读Installation Guide。通过预先安装任何必要的依赖项，我们才能顺利地启动和运行应用程序。

此时，我们应该已经安装了Elixir、Erlang、Hex和Phoenix archive。我们还应该安装PostgreSQL和node.js来构建一个默认的应用程序。

好，我们已经整装待发了！

我们可以从任何目录运行 `mix phx.new` 来启动我们的Phoenix应用程序。Phoenix将接受一个绝对路径或相对路径来作为我们新项目的目录。假设我们应用程序的名称是 `hello` ，让我们运行以下命令:

```
$ mix phx.new hello
```

> 在我们开始之前，关于webpack的说明：Phoenix将默认使用webpack进行资源管理。Webpack的依赖是通过node包管理器安装的，而不是mix。Phoenix将在 `mix phx.new` 任务的最后提示我们安装它们。如果在那时我们选择了不安装，并且后面也没通过npm install安装那些依赖项，我们的应用程序将在我们试图启动它时引发错误，资源可能无法正确加载。如果我们完全不想使用webpack，我们可以简单地传递 `--no-webpack` 给 `mix phx.new` 。
>
> 关于Ecto的说明：Ecto允许我们的Phoenix应用程序与数据存储进行通信，比如PostgreSQL、MySQL等。如果我们的应用程序不需要这个组件，我们可以通过传递 `--no-ecto` 标志给 `mix phx.new` 来跳过这个依赖项。此标志还可以与 `--no-webpack` 组合来创建一个骨架应用程序。
>
> 学习有关 `mix phx.new` 的更多内容，你可以阅读Mix Tasks指南。

```
mix phx.new hello
* creating hello/config/config.exs
* creating hello/config/dev.exs
```

```
* creating hello/config/prod.exs
...
* creating hello/assets/static/images/phoenix.png
* creating hello/assets/static/favicon.ico

Fetch and install dependencies? [Yn]
```

Phoenix生成应用程序所需的目录结构和所有文件。完成后，它会询问我们是否需要为我们安装依赖项。我们选择是。

```
Fetch and install dependencies? [Yn] Y
* running mix deps.get
* running mix deps.compile
* running cd assets && npm install && node node_modules/webpack/bin/webpack.js --mode development

We are almost there! The following steps are missing:

    $ cd hello

Then configure your database in config/dev.exs and run:

    $ mix ecto.create

Start your Phoenix app with:

    $ mix phx.server

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

    $ iex -S mix phx.server
```

一旦依赖项安装完毕，任务将提示我们切换到项目目录并启动应用程序。

Phoenix假设我们的PostgreSQL数据库将有一个postgres用户帐户，该帐户具有正确的权限和一个“postgres”的密码。如果情况不是这样，请参阅Mix Tasks指南，以学习更多关于 `mix ecto.create` 任务。

好吧，让我们试一试。首先，我们 `cd` 进入刚才创建的 `hello/` 目录中:

```
$ cd hello
```

现在我们将创建我们的数据库:

```
$ mix ecto.create
Compiling 13 files (.ex)
Generated hello app
The database for Hello.Repo has been created
```

> 注意：如果这是您第一次运行此命令，Phoenix可能还会要求安装Rebar。继续安装，因为Rebar用于构建Erlang包。

最后，我们将启动Phoenix服务器:

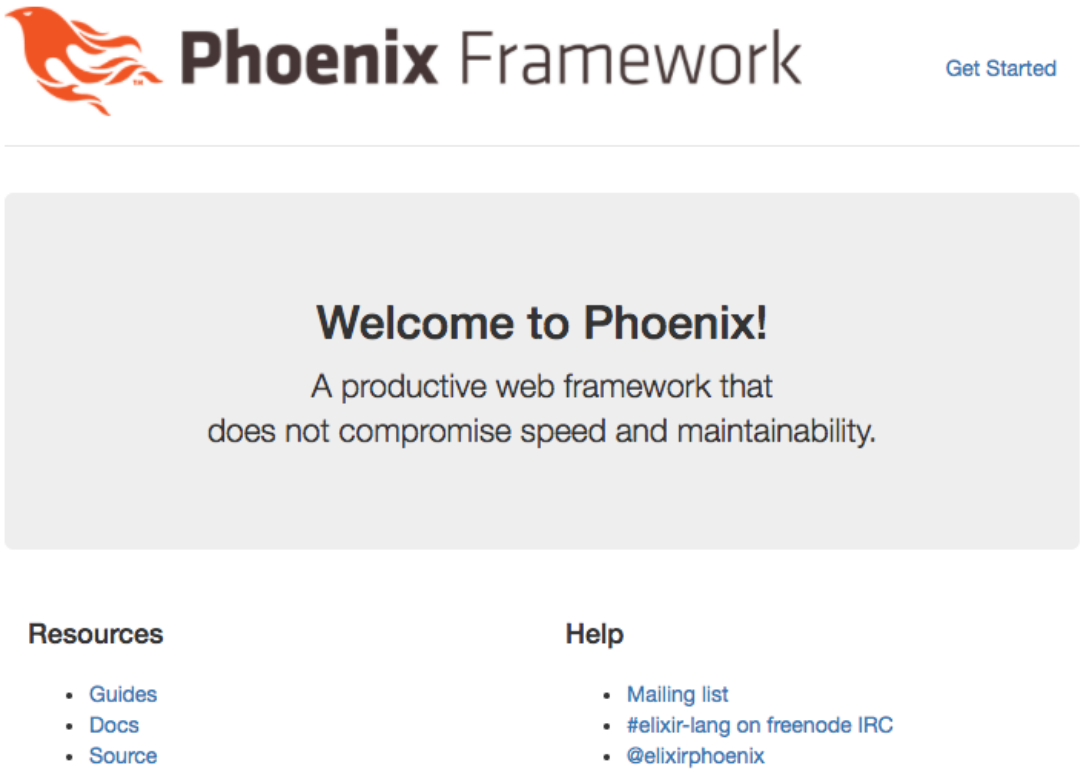
```
$ mix phx.server
[info] Running HelloWeb.Endpoint with cowboy 2.5.0 at http://localhost:4000

Webpack is watching the files…
...
```

如果我们在生成新应用程序时选择不让Phoenix安装我们的依赖项， `mix phx.new` 任务将提示我们当想要安装他们时采取的必要步骤。

```
Fetch and install dependencies? [Yn] n

We are almost there! The following steps are missing:

    $ cd hello
    $ mix deps.get
    $ cd assets && npm install && node node_modules/webpack/bin/webpack.js --mode development

Then configure your database in config/dev.exs and run:

    $ mix ecto.create

Start your Phoenix app with:
```

```
    $ mix phx.server

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

    $ iex -S mix phx.server
```

默认情况下，Phoenix接受4000端口的请求。如果我们将最喜欢的web浏览器指向http://localhost:4000，就会看到Phoenix Framework的欢迎页面。



如果你的屏幕看起来像上面的图片，恭喜你！你现在有了一个工作中的Phoenix应用程序。如果你看不到上面的页面，请尝试通过http://127.0.0.1:4000访问它，然后确保你的操作系统将“localhost”定义为“127.0.0.1”。

在本地，我们的应用程序在 `iex` 会话中运行。为了停止它，我们按下 `ctrl-c` 两次，就像我们通常停止 `iex` 一样。

下一步是稍微定制一下我们的应用程序，让我们了解一下Phoenix应用程序是如何组合在一起

的。