Working with Phoenix Framework

Ruben Amortegui
@ramortegui
https://www.rubenamortegui.com
https://github.com/ramortegui

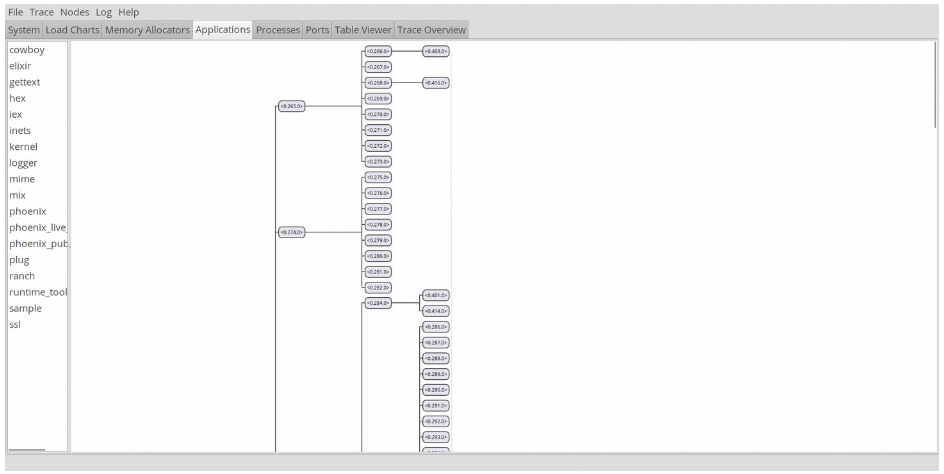
paradem

Phoenix

- Web framework
  - MVC architectural pattern.
  - Fast
  - Concurrent
  - Reliable

- Fast
  - Model of concurrency
  - Leverage pattern matching router functions
  - Templates are pre-compiled

- Concurrent
  - Elixir programming model makes reasoning about concurrent systems almost as easy as reasoning about single-threaded ones.

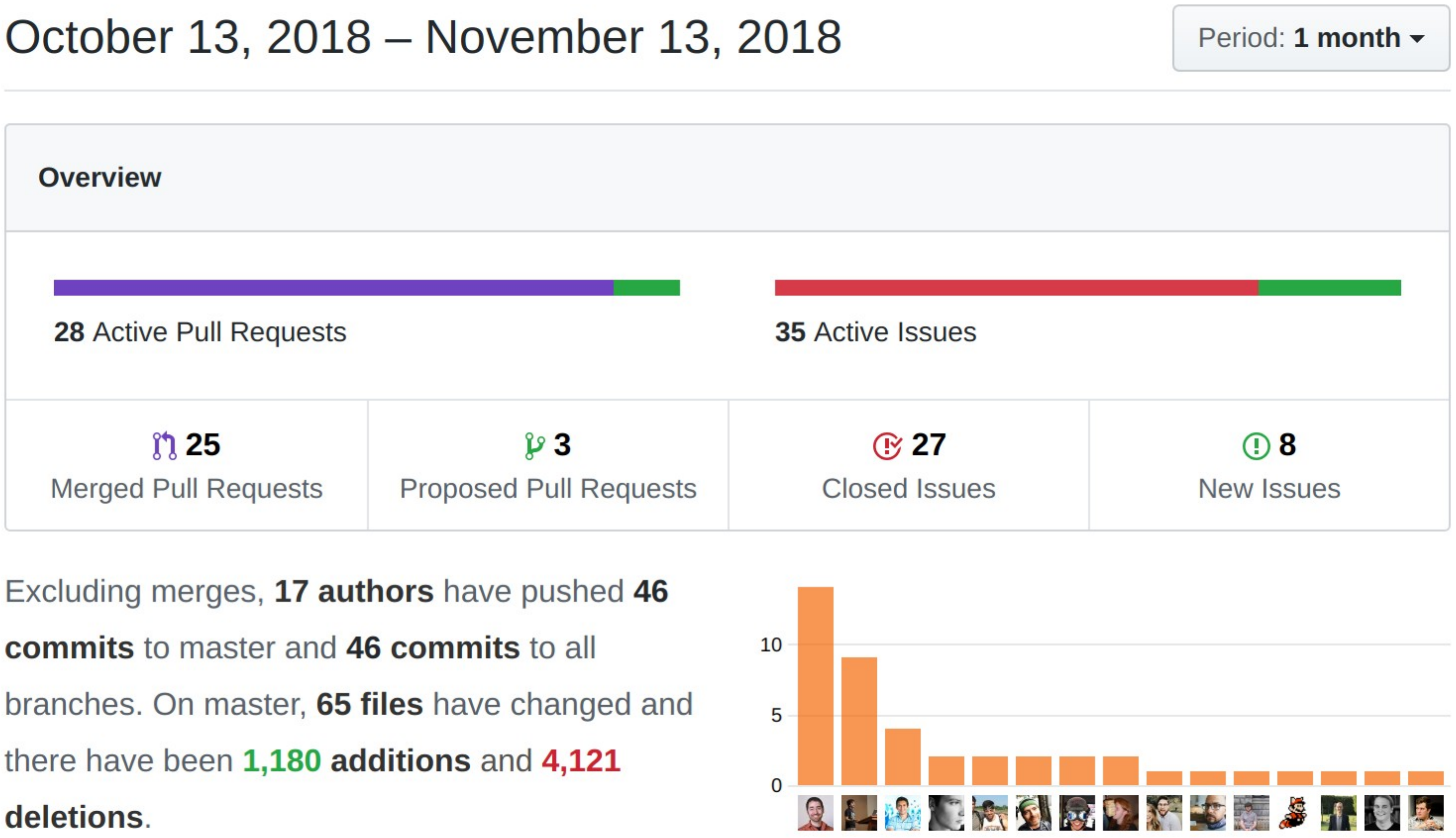


- Reliable
  - Based on Processes:
    - Linking structure
    - Communication
    - Supervision trees.

Status

- V 1.4.0 Nov. 7th 2018
  - Bug fixes
  - Cowboy 2 support
    - Http2
  - JSON library
    - \`jason\` instead of \`poison\`
  - Ecto 3.0
    - ecto_sql / ecto
  - Webpack
    - Remove brunch
  - Formatter integration
    - Added on Elixir 1.6

Insights



October 13, 2018 – November 13, 2018

Period: **1 month**

Overview

28 Active Pull Requests

35 Active Issues

25 Merged Pull Requests	**3** Proposed Pull Requests	**27** Closed Issues	**8** New Issues

Excluding merges, **17 authors** have pushed **46 commits** to master and **46 commits** to all branches. On master, **65 files** have changed and there have been **1,180 additions** and **4,121 deletions**.

HTTP Request/Response

- HTTP Request
  - Cowboy (Plug adapter)
    - Endpoint
    - Router
    - Controller
    - Views
    - Template
- HTTP Response
  - send_resp function

https://elixir-examples.github.io/examples/phoenix-framework-from-http-request-to-response

Plug

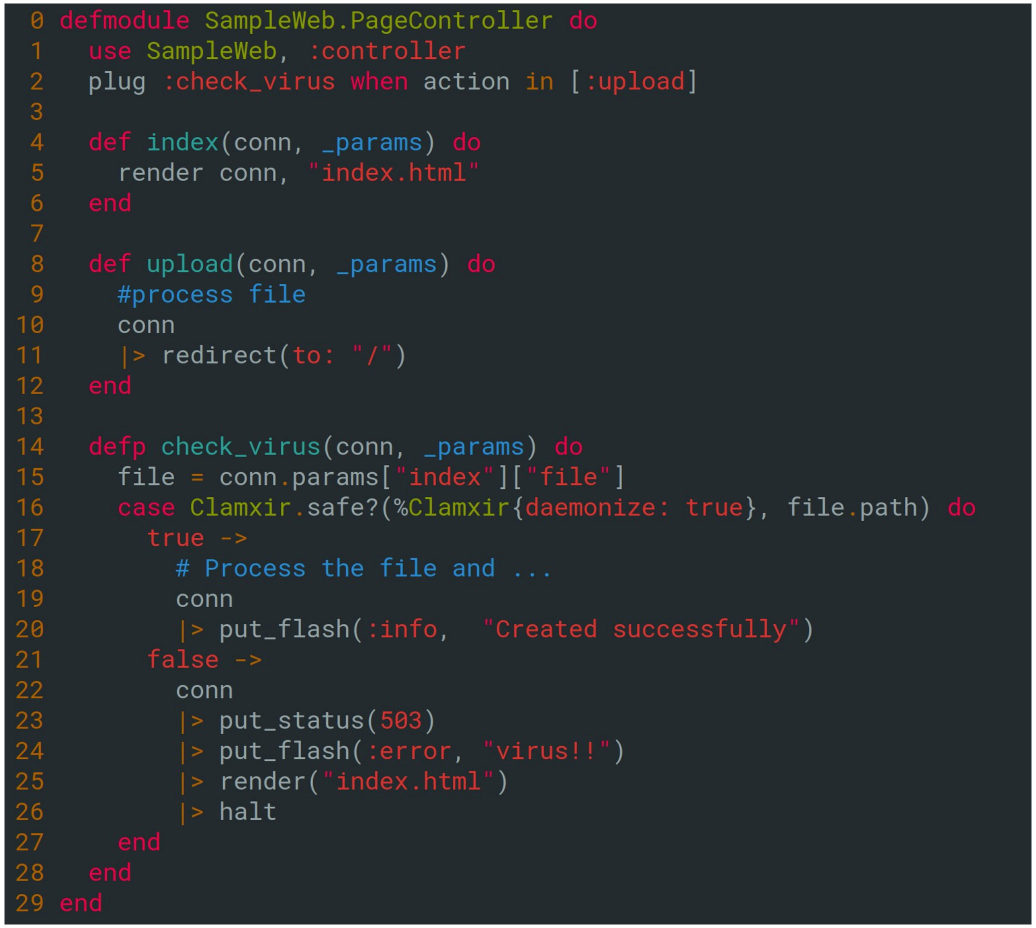
- Plug is a specification for composable modules in between web applications.
- It is also an abstraction layer for connection adapters of different web servers.

Plug

- Function
  - Specific for the module

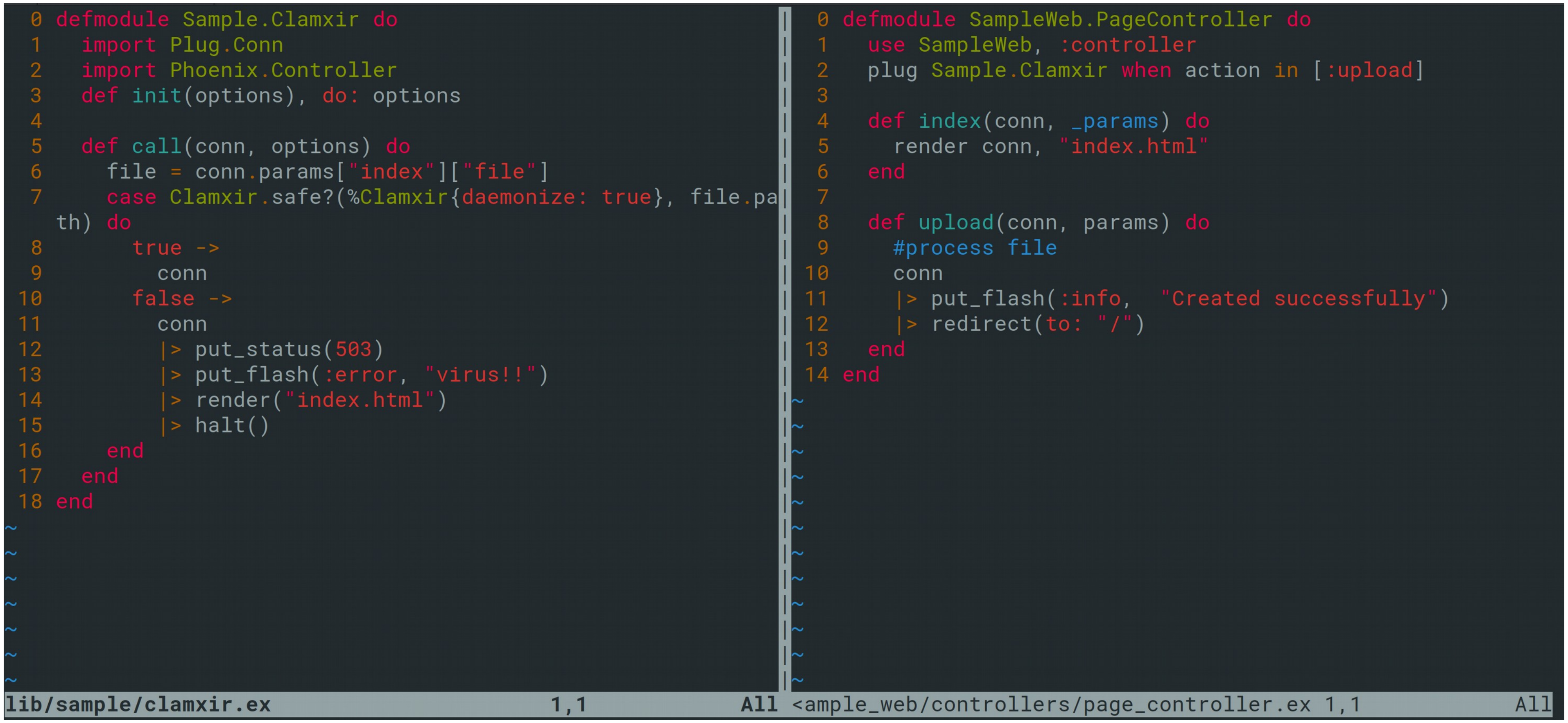
- Module
  - Shareable functionality
  - Load resource on controllers

Plug as a Function



```elixir
defmodule SampleWeb.PageController do
  use SampleWeb, :controller
  plug :check_virus when action in [:upload]

  def index(conn, _params) do
    render conn, "index.html"
  end

  def upload(conn, _params) do
    #process file
    conn
    |> redirect(to: "/")
  end

  defp check_virus(conn, _params) do
    file = conn.params["index"]["file"]
    case Clamxir.safe?(%Clamxir{daemonize: true}, file.path) do
      true ->
        # Process the file and ...
        conn
        |> put_flash(:info,  "Created successfully")
      false ->
        conn
        |> put_status(503)
        |> put_flash(:error, "virus!!")
        |> render("index.html")
        |> halt
    end
  end
end
```

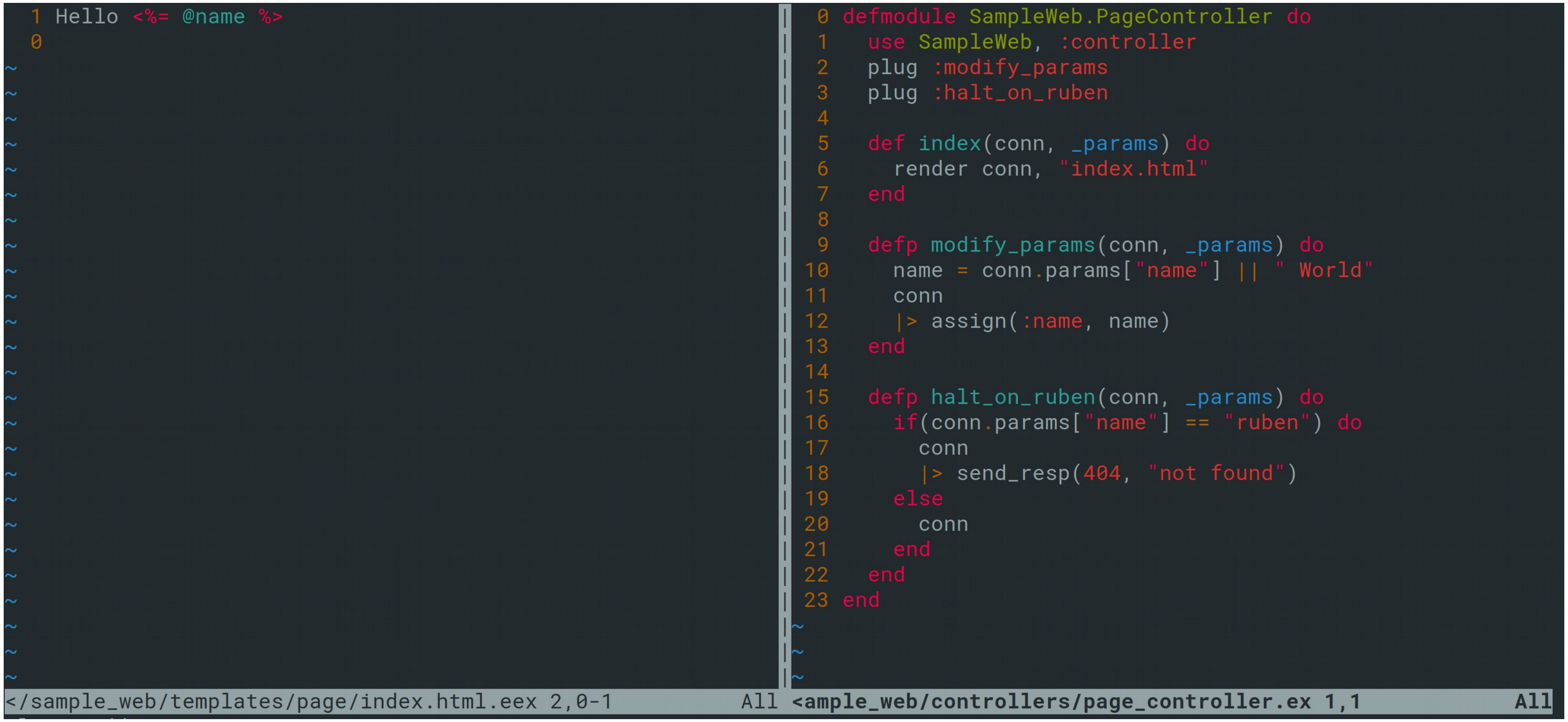
Plug as a Module



```
defmodule Sample.Clamxir do
  import Plug.Conn
  import Phoenix.Controller
  def init(options), do: options

  def call(conn, options) do
    file = conn.params["index"]["file"]
    case Clamxir.safe?(%Clamxir{daemonize: true}, file.path) do
      true ->
        conn
      false ->
        conn
        |> put_status(503)
        |> put_flash(:error, "virus!!")
        |> render("index.html")
        |> halt()
    end
  end
end
~
~
~
~
~
~
~
lib/sample/clamxir.ex                     1,1            All
```

```
defmodule SampleWeb.PageController do
  use SampleWeb, :controller
  plug Sample.Clamxir when action in [:upload]

  def index(conn, _params) do
    render conn, "index.html"
  end

  def upload(conn, params) do
    #process file
    conn
    |> put_flash(:info,  "Created successfully")
    |> redirect(to: "/")
  end
end
~
~
~
~
~
~
~
~
~
~
~
~
<ample_web/controllers/page_controller.ex 1,1            All
```

Transform a Conn



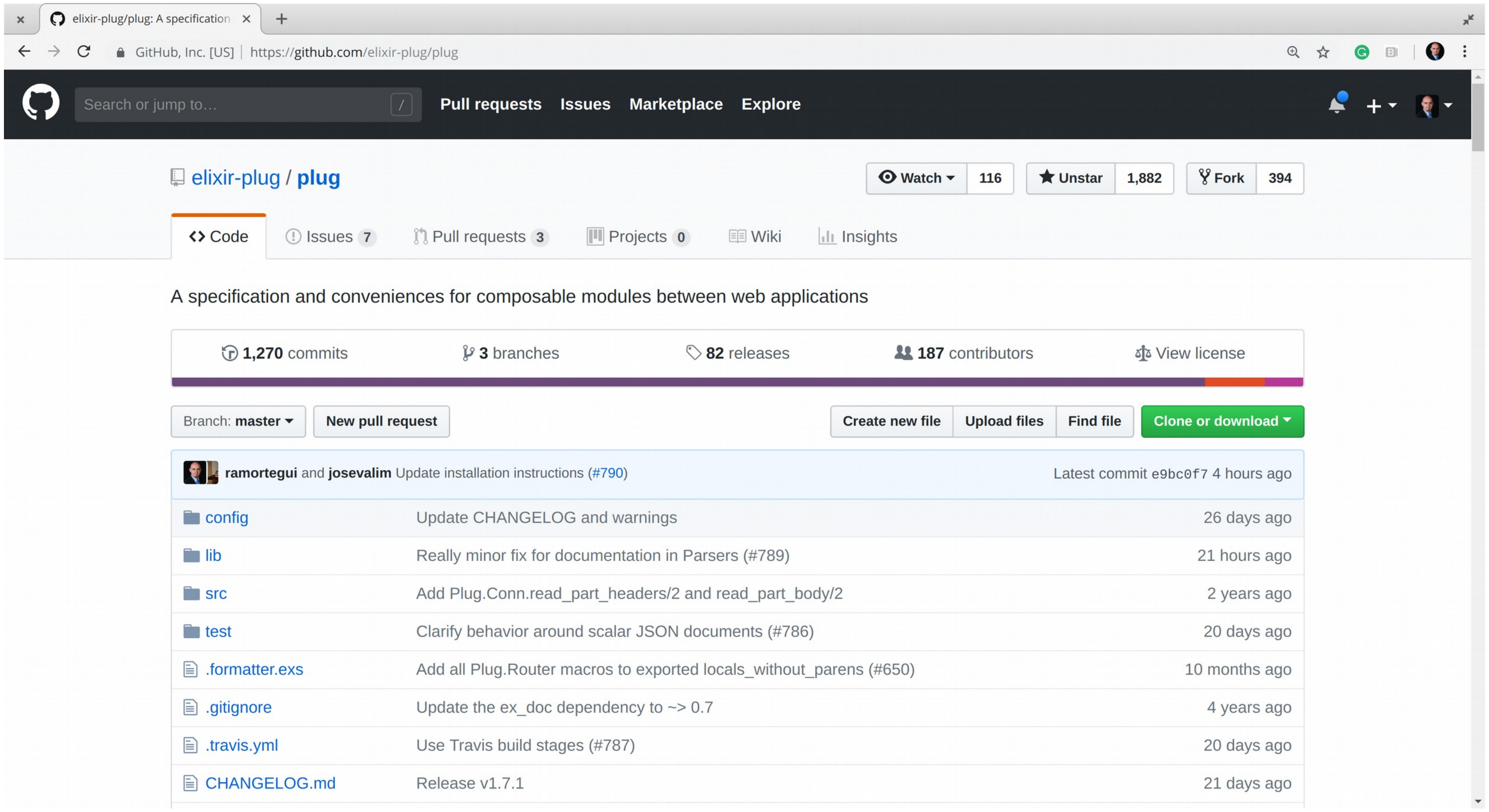
```
Hello <%= @name %>

</sample_web/templates/page/index.html.eex 2,0-1          All
```

```elixir
defmodule SampleWeb.PageController do
  use SampleWeb, :controller
  plug :modify_params
  plug :halt_on_ruben

  def index(conn, _params) do
    render conn, "index.html"
  end

  defp modify_params(conn, _params) do
    name = conn.params["name"] || " World"
    conn
    |> assign(:name, name)
  end

  defp halt_on_ruben(conn, _params) do
    if(conn.params["name"] == "ruben") do
      conn
      |> send_resp(404, "not found")
    else
      conn
    end
  end
end

<ample_web/controllers/page_controller.ex 1,1          All
```

HTTP Request/Response

- HTTP Request
  - Cowboy (Plug adapter)
    - Endpoint
    - Router
    - Controller
    - Views
    - Template
- HTTP Response
  - send_resp function

https://elixir-examples.github.io/examples/phoenix-framework-from-http-request-to-response



elixir-plug / plug

A specification and conveniences for composable modules between web applications

1,270 commits | 3 branches | 82 releases | 187 contributors | View license

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

ramortegui and josevalim Update installation instructions (#790) — Latest commit e9bc0f7 4 hours ago

config	Update CHANGELOG and warnings	26 days ago
lib	Really minor fix for documentation in Parsers (#789)	21 hours ago
src	Add Plug.Conn.read_part_headers/2 and read_part_body/2	2 years ago
test	Clarify behavior around scalar JSON documents (#786)	20 days ago
.formatter.exs	Add all Plug.Router macros to exported locals_without_parens (#650)	10 months ago
.gitignore	Update the ex_doc dependency to ~> 0.7	4 years ago
.travis.yml	Use Travis build stages (#787)	20 days ago
CHANGELOG.md	Release v1.7.1	21 days ago

Endpoint

- The start and end of the request life cycle
- Handles all aspects of requests up until the point where the router takes over
- Provides a core set of plugs to apply to all requests
- Dispatches requests into a designated router

Router

- Parses incoming requests and dispatches them to the correct controller/action, passing parameters as needed.
- Provides helpers to generate route paths or urls to resources.
- Defines named pipelines through which we may pass our requests.
- Pipelines - allow easy application of groups of plugs to a set of routes.

Controllers

- Provide functions called actions to handle requests
- Actions:
  - Prepare data and pass it into views
  - Invoke rendering via views
  - Perform redirects

Views

- Render templates
- Act as a presentation layer
- Define helper functions, available in templates, to decorate data for presentation

Templates

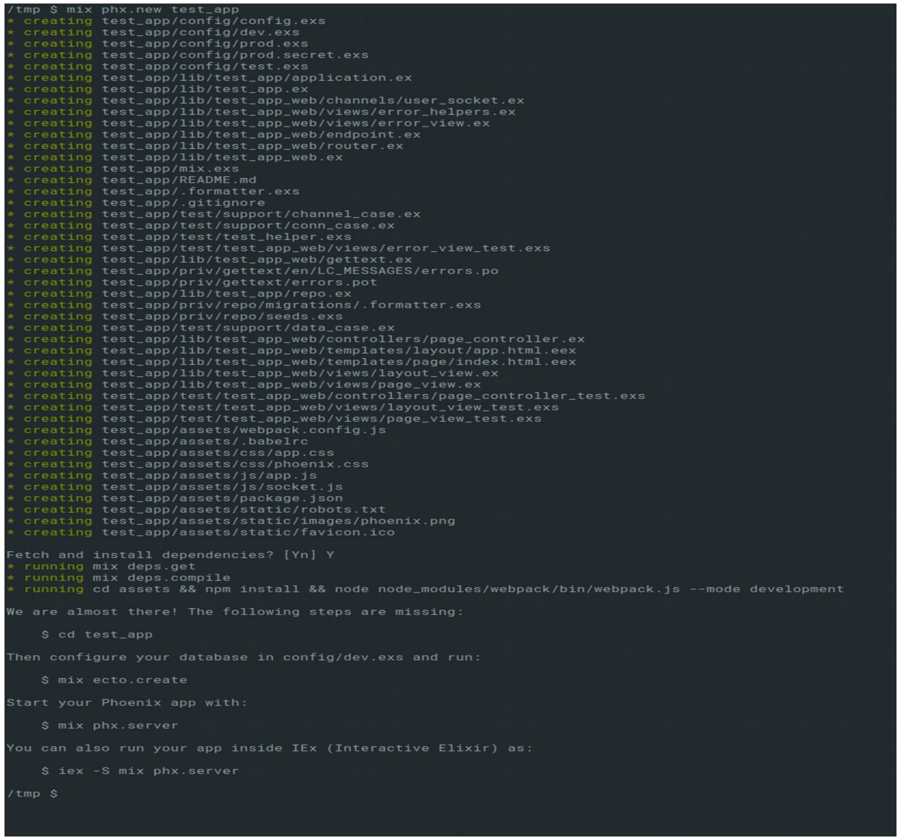
- Files containing the contents that will be served in a response.
- Provide the basic structure for a response, and allow dynamic data to be substituted in.
- Are pre-compiled and fast.

Channels and PubSub

- Channels
  - Manage sockets for easy real time communication
  - Are analogous to controllers except that they allow bi-directional communication with persistent connections
- PubSub
  - Underlies the channel layer and allows clients to subscribe to topics
  - Abstracts the underlying PubSub adapter for third-party PubSub integration

Default Structure and Files

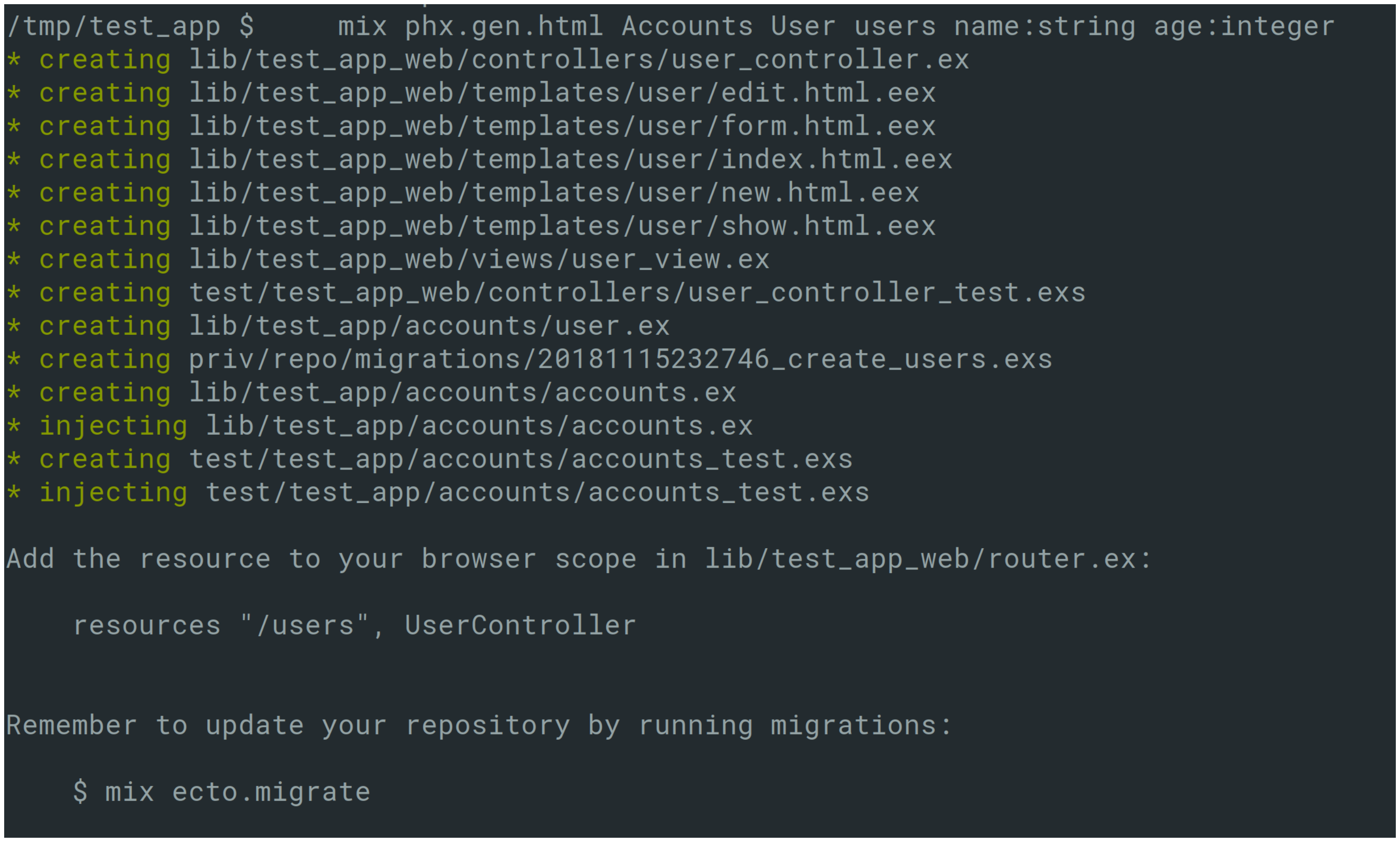
– mix phx.new test_app



```
/tmp $ mix phx.new test_app
* creating test_app/config/config.exs
* creating test_app/config/dev.exs
* creating test_app/config/prod.exs
* creating test_app/config/prod.secret.exs
* creating test_app/config/test.exs
* creating test_app/lib/test_app/application.ex
* creating test_app/lib/test_app.ex
* creating test_app/lib/test_app_web/channels/user_socket.ex
* creating test_app/lib/test_app_web/views/error_helpers.ex
* creating test_app/lib/test_app_web/views/error_view.ex
* creating test_app/lib/test_app_web/endpoint.ex
* creating test_app/lib/test_app_web/router.ex
* creating test_app/lib/test_app_web.ex
* creating test_app/mix.exs
* creating test_app/README.md
* creating test_app/.formatter.exs
* creating test_app/.gitignore
* creating test_app/test/support/channel_case.ex
* creating test_app/test/support/conn_case.ex
* creating test_app/test/test_helper.exs
* creating test_app/test/test_app_web/views/error_view_test.exs
* creating test_app/lib/test_app_web/gettext.ex
* creating test_app/priv/gettext/en/LC_MESSAGES/errors.po
* creating test_app/priv/gettext/errors.pot
* creating test_app/lib/test_app/repo.ex
* creating test_app/priv/repo/migrations/.formatter.exs
* creating test_app/priv/repo/seeds.exs
* creating test_app/test/support/data_case.ex
* creating test_app/lib/test_app_web/controllers/page_controller.ex
* creating test_app/lib/test_app_web/templates/layout/app.html.eex
* creating test_app/lib/test_app_web/templates/page/index.html.eex
* creating test_app/lib/test_app_web/views/layout_view.ex
* creating test_app/lib/test_app_web/views/page_view.ex
* creating test_app/test/test_app_web/controllers/page_controller_test.exs
* creating test_app/test/test_app_web/views/layout_view_test.exs
* creating test_app/test/test_app_web/views/page_view_test.exs
* creating test_app/assets/webpack.config.js
* creating test_app/assets/.babelrc
* creating test_app/assets/css/app.css
* creating test_app/assets/css/phoenix.css
* creating test_app/assets/js/app.js
* creating test_app/assets/js/socket.js
* creating test_app/assets/package.json
* creating test_app/assets/static/robots.txt
* creating test_app/assets/static/images/phoenix.png
* creating test_app/assets/static/favicon.ico

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 test_app

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

/tmp $
```

Generators – phx.gen.html

mix phx.gen.html Accounts User users name:string age:integer



```
/tmp/test_app $     mix phx.gen.html Accounts User users name:string age:integer
* creating lib/test_app_web/controllers/user_controller.ex
* creating lib/test_app_web/templates/user/edit.html.eex
* creating lib/test_app_web/templates/user/form.html.eex
* creating lib/test_app_web/templates/user/index.html.eex
* creating lib/test_app_web/templates/user/new.html.eex
* creating lib/test_app_web/templates/user/show.html.eex
* creating lib/test_app_web/views/user_view.ex
* creating test/test_app_web/controllers/user_controller_test.exs
* creating lib/test_app/accounts/user.ex
* creating priv/repo/migrations/20181115232746_create_users.exs
* creating lib/test_app/accounts/accounts.ex
* injecting lib/test_app/accounts/accounts.ex
* creating test/test_app/accounts/accounts_test.exs
* injecting test/test_app/accounts/accounts_test.exs

Add the resource to your browser scope in lib/test_app_web/router.ex:

    resources "/users", UserController


Remember to update your repository by running migrations:

    $ mix ecto.migrate
```

Generators – phx.gen.html

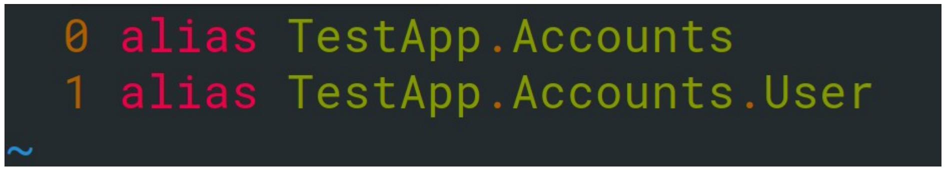
- Migration
- Model
- Context
- View
- Template
- Tests

Developer tips

- How to get help?
  - mix help
  - mix help phx.new
- Inside iex
  - h function/arity
  - Eg: iex> h IO.puts

Developer tips

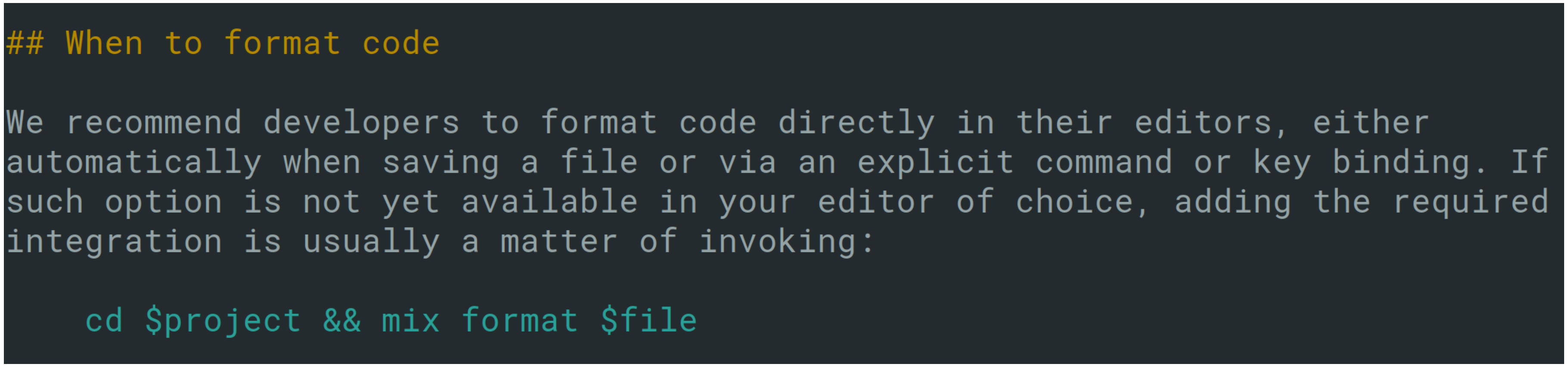
- .iex.exs



```
alias TestApp.Accounts
alias TestApp.Accounts.User
```

Developer Tips

- mix format



```
## When to format code

We recommend developers to format code directly in their editors, either
automatically when saving a file or via an explicit command or key binding. If
such option is not yet available in your editor of choice, adding the required
integration is usually a matter of invoking:

    cd $project && mix format $file
```

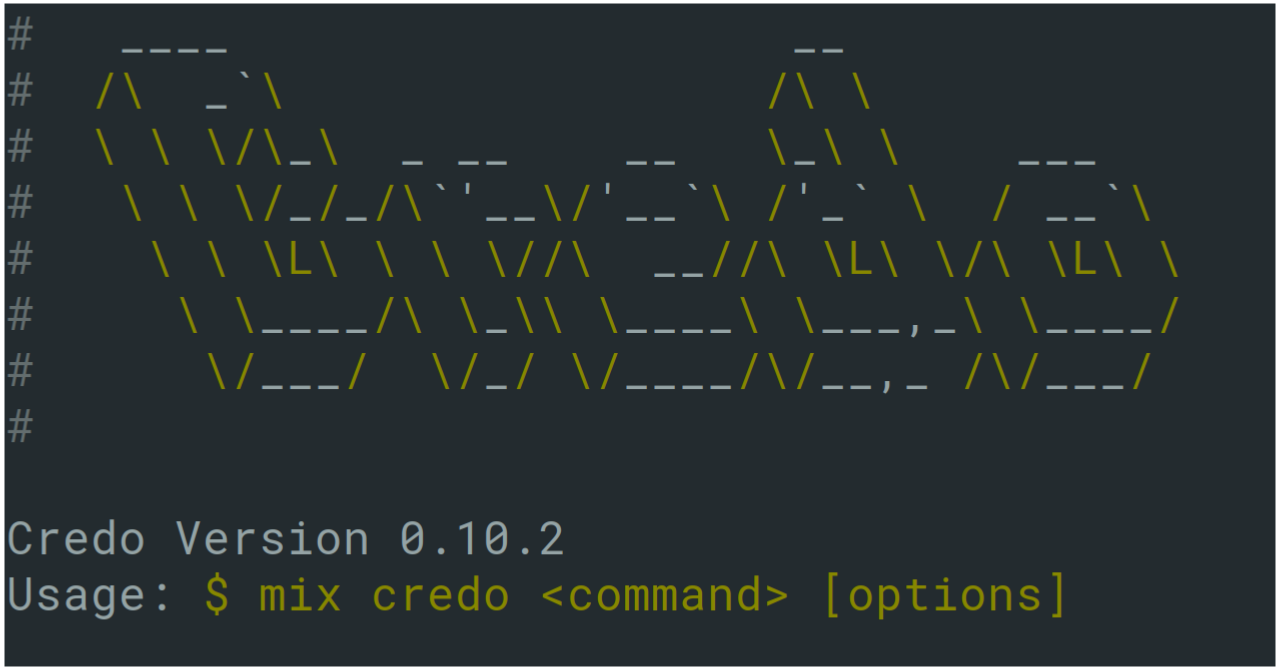
Developer Tips

- Working in a team
  - Migrations
  - Javascript dependencies
  - Mix dependencies

- Add mix tasks for updates
  - mix update

Developer Tips

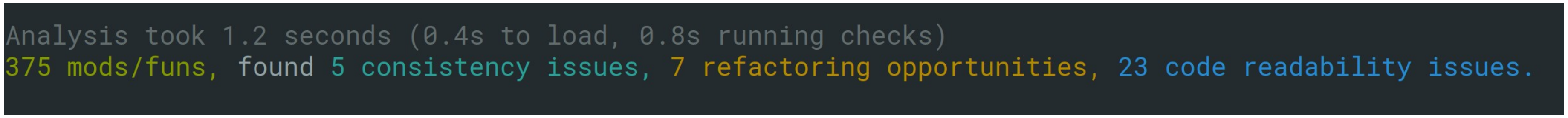
- Code Analyzer
  - https://github.com/rrrene/credo



```
#   ____                   __
#  /\  _`\                /\ \
#  \ \ \/\_\  _ __    __  \_\ \    ___
#   \ \ \/_/_/\`'__\/'__`\ /'_` \  / __`\
#    \ \ \L\ \ \ \//\  __//\ \L\ \/\ \L\ \
#     \ \____/\ \_\\ \____\ \___,_\ \____/
#      \/___/  \/_/ \/____/\/__,_ /\/___/
#

Credo Version 0.10.2
Usage: $ mix credo <command> [options]
```

Credo

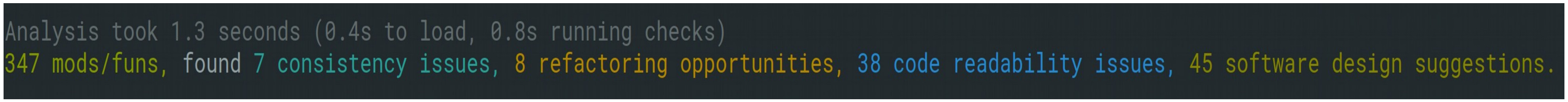
- mix credo --all



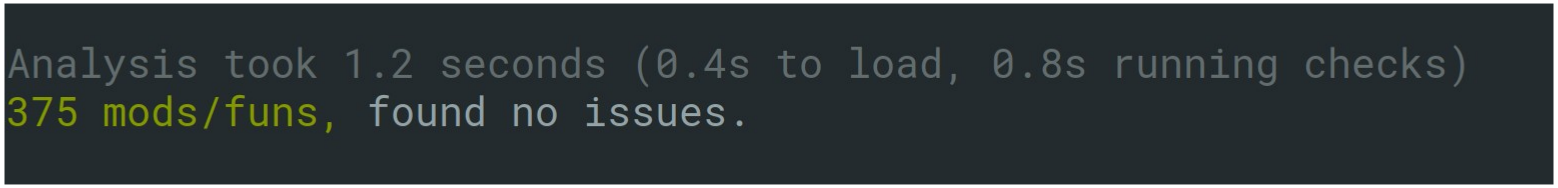
```
Analysis took 1.2 seconds (0.4s to load, 0.8s running checks)
375 mods/funs, found 5 consistency issues, 7 refactoring opportunities, 23 code readability issues.
```

Credo

- mix credo –strict --all



```
Analysis took 1.3 seconds (0.4s to load, 0.8s running checks)
347 mods/funs, found 7 consistency issues, 8 refactoring opportunities, 38 code readability issues, 45 software design suggestions.
```



```
Analysis took 1.2 seconds (0.4s to load, 0.8s running checks)
375 mods/funs, found no issues.
```

Credo

```
Analysis took 1.2 seconds (0.4s to load, 0.8s running checks)
375 mods/funs, found no issues.
```

Developer Tips

- CircleCI, TravisCI
  - Run tests
  - Check format
  - Analyze code

Libraries

- Authentication
  - Guardian (protection and function callbacks for authentication)
    - Implements JSON Web Token
    - https://github.com/ueberauth/guardian

Libraries

- Authorization and Resource loads
  - Canary
    - https://github.com/cpjk/canary
  - Bodyguard
    - https://github.com/schrockwell/bodyguard
  - Policy Wonk(*)
    - https://github.com/boydm/policy_wonk

Libraries

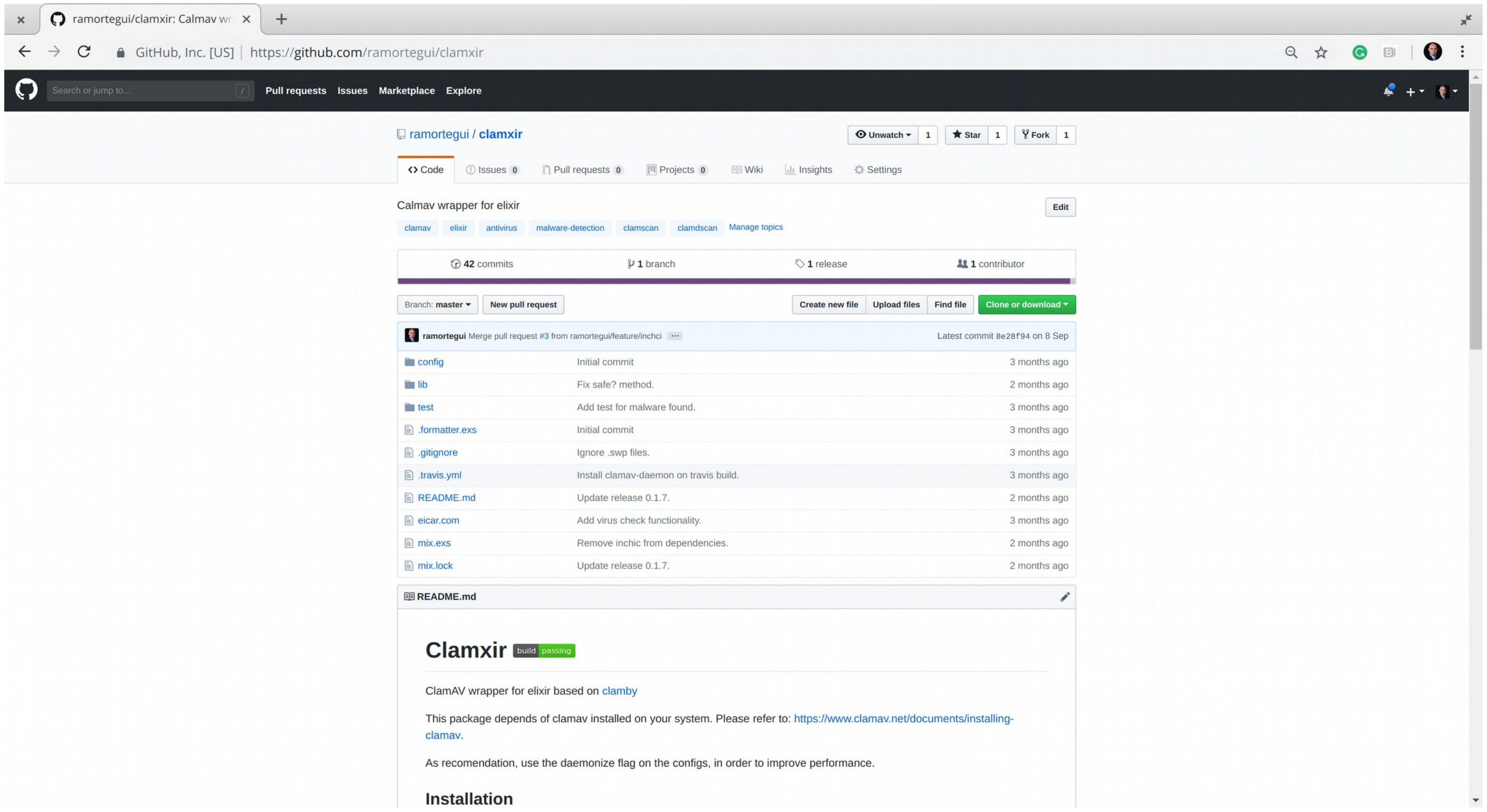
- Tests
  - ExUnit
  - Tags
  - Setup
  - ex_machina

Libraries

- Deployments
  - Server
    - Distillery
      - Build releases
      - https://github.com/bitwalker/distillery
    - Edeliver
      - Build and deploy elixir apps with hot code upgrade.
      - https://github.com/edeliver/edeliver
  - Heroku
    - https://hexdocs.pm/phoenix/heroku.html

Other Libraries

- Wrappers
  - Clamxir: ClamAV wrapper based on Clamby.



Calmav wrapper for elixir

ramortegui Merge pull request #3 from ramortegui/feature/inchci … Latest commit 8e28f94 on 8 Sep

config	Initial commit	3 months ago
lib	Fix safe? method.	2 months ago
test	Add test for malware found.	3 months ago
.formatter.exs	Initial commit	3 months ago
.gitignore	Ignore .swp files.	3 months ago
.travis.yml	Install clamav-daemon on travis build.	3 months ago
README.md	Update release 0.1.7.	2 months ago
eicar.com	Add virus check functionality.	3 months ago
mix.exs	Remove inchic from dependencies.	2 months ago
mix.lock	Update release 0.1.7.	2 months ago

README.md

Clamxir build passing

ClamAV wrapper for elixir based on clamby

This package depends of clamav installed on your system. Please refer to: https://www.clamav.net/documents/installing-clamav.

As recomendation, use the daemonize flag on the configs, in order to improve performance.

Installation

Summary

- HTTP Request/Response with phoenix is based on transformation of the structure of %Plug.Conn{...}, and is fast, no magic.
- Phoenix is based on Plugs, and understanding plugs will make your work with Phoenix a breeze.

References

- https://hexdocs.pm/phoenix/overview.html
- https://hexdocs.pm/phoenix/plug.html
- Programming phoenix 1.4. Chris McCord, Bruce Tate, José Valim. V Oct 19. 2018

Thanks!

Q & A?

@ramortegui

https://www.rubenamortegui.com

https://github.com/ramortegui