Phoenix LiveView

Paul Valckenaers
Bram Van Impe

Phoenix auth

- Maak een Phoenix LiveView server (met postgress database)
- Voeg authentication toe aan de server:

  mix phx.gen.auth Accounts User users
- Open web browser
  - *http://localhost:4000/*
- Open een code editor
  - *code .*

router.ex

```
scope "/", Deel2Web do
  pipe_through [:browser, :require_authenticated_user]

  get "/users/settings", UserSettingsController, :edit
  put "/users/settings", UserSettingsController, :update

  live "/prive", MyPage

end
```

my_page.ex

```
defmodule Deel1Web.MyPage do
  use Phoenix.LiveView

  def mount(_params, _session, socket), do: { :ok, socket}

  def render(assigns) do
    ~H"""
    <h1> --------- Hello World ------------ </h1>
    """
  end
end
```

my_page1.ex

```
def mount( params , session, socket),
  do: {:ok, assign(socket, sessie: inspect(session), param: inspect(params))}


def render(assigns) do
  ~H"""
  <pre> Deze web page is persoonlijk... </pre>
  <p><b> ASSIGNS : </b> <%= inspect(assigns) %> </p>
  <p><b> SESSION :  </b> <%= @sessie %> </p>
  <p><b> PARAMS : </b> <%= @param %> </p>
  """
end
```

my_page2.ex

```
def mount( _params , session, socket) do
   user = Accounts.get_user_by_session_token(session["user_token"])
   {:ok, assign(socket, sessie: inspect(session), user: user.email)}
 end

def render(assigns) do
   ~H"""
   <pre> Deze web page is persoonlijk... </pre>
   <p> GEBRUIKER IS : <%= @user %> </p>
  ____________________________________________
   <p> <%= inspect(assigns) %> </p>
   """
end
```

Assignments

- Open de ‘prive-page’ in meerdere tabs
- Indien mogelijk, in meerdere browsers
  - Firefox, edge, chromium, brave…
- Wat zie je in de “assigns”, “session”?

- Bekijk de accounts bestanden
  - Startpunt voor …

GenServer

- LiveView is een (soort) GenServer
- Callback functions
  - handle_event
  - handle_info
  - handle_call
  - handle_cast

GenServer

```
defmodule Deel2.MyServer1 do
  use GenServer
def start_link(args) do
  {:ok, pid} = GenServer.start_link(__MODULE__, args)
  Process.register(pid, :ikke)
  {:ok, pid}
End
#### callback functions ####
def init(args), do: {:ok, initial_state = args}
```

GenServer / call

```
# User functions
  def get(server_pid), do: GenServer.call(server_pid, :get)

# Callback functions
  handle_call(:get, _from, state),
    do: {:reply, reply = state, state}
```

GenServer / cast

```
# User functions
  def inc(server_pid), do: GenServer.cast(server_pid, :inc)

# Callback functions
  handle_cast(:inc, _from, state),
    do: {:noreply, new_state = state + 1}
```

GenServer / info

Messages from anywhere

send(server_pid, 1)

send(:ikke, 2)

Callback functions

handle_info(msg, state),

do: {:noreply, new_state = state + msg}

Application.ex - supervisors

```
def start(_type, _args) do
    Children = [ ...
      # Start a worker by calling: Deel2.Worker.start_link(arg)
      # {Deel2.Worker, arg}
      {Deel2.MyServer, 0}, # module name & initialization parameter
      …
    ]
    opts = [strategy: :one_for_one, name: Deel2.Supervisor]
    Supervisor.start_link(children, opts)
  end
```

Assigments

- Iex -S mix phx.server en dan
  - Test de gen_server
  - Crash de server
  - Process.whereis(:ikke)
- Pas de GenServer aan
  - inc(amount)
  - List → inc(hd) , reset(), get(), …

Phoenix.PubSub

- MyPage
  - alias Phoenix.PubSub
  - PubSub.subscribe(<server>, <topic>)
  - handle_info

Phoenix.PubSub

Application

```
{Phoenix.PubSub, name: Deel2.PubSub},

%{
    id: Phoenix.PubSub.Supervisor2,
    start: {Phoenix.PubSub.Supervisor, :start_link, [[name: :hi]]},
    type: :supervisor
  },
```

Assigments

- Iex -S mix phx.server en dan

  Gebruik de 2 PubSub servers

  om op de web page informatie te tonen

  Hergebruik de code van vorige les om van de gebruiker informatie naar de gen_server voor List te sturen