

Phoenix per principianti

di Paolo Montrasio
paolo.montrasio@connettiva.eu

Slide a
connettiva.eu/phoenix-per-principianti.pdf

Ovvero...

2005: Conoscenza(Ruby) == 0 → Rails

2014: Conoscenza(Elixir) == 0 → Phoenix

Imparare Elixir mentre si impara Phoenix

- Le guide di Phoenix

<http://www.phoenixframework.org/docs/overview>

- I generatori per i controller

Per ambientarsi

- Phoenix è MVC
- È giovane ma si innalza sulle spalle dei giganti
- Per chi conosce Rails:
 - Models e Controllers → Models e Controllers
 - Views → Template
 - Helpers → Views (circa)
 - Cables (Rails 5) → Channels (da sempre)
 - ActiveRecord → Ecto
 - Migrations → Migrations

Differenze

- Ecto
 - Lo schema va dichiarato
 - Changeset per modifiche e validazioni
- Controller singoli (UserController ma /users)
- app/ → web/
- config/database.yml → config/<env>.exs
- config/routes.rb → web/router.ex

Creare una web app

<http://www.phoenixframework.org/docs/up-and-running>

```
mix phoenix.new bologna_2015
```

```
git add .gitignore config/ lib/ mix.* package.json  
priv/ README.md test/ web/
```

```
git commit -a -m "Demo for today!"
```

config/dev.exs

```
config :bologna_2015, Bologna_2015.Repo,  
  adapter: Ecto.Adapters.Postgres,  
  username: "bologna_2015",  
  password: "RJP4Q1_2vPYX4UOR",  
  database: "bologna_2015_dev",  
  hostname: "localhost",  
  pool_size: 10
```

Lancio della web app

```
$ mix phoenix.run # rails s
```

```
$ iex -S mix # rails c + rails s
```

```
http://localhost:4000
```

```
$ mix -h # rake -T
```

Debug

```
ieX -S mix phoenix.server
```

```
include IEx.pry
```

```
IEx.pry nel punto dove ispezionare lo stato
```


web/router.ex

```
defmodule Bologna_2015.Router do
  use Bologna_2015.Web, :router
  scope "/", Bologna_2015 do
    pipe_through :browser
    get "/", PageController, :index
    resources "/users", UserController
  end
end
```

Restful routes

```
$ mix phoenix.routes
```

```
page_path  GET    /                Bologna_2015.PageController :index
user_path  GET    /users          Bologna_2015.UserController :index
user_path  GET    /users/:id/edit Bologna_2015.UserController :edit
user_path  GET    /users/new      Bologna_2015.UserController :new
user_path  GET    /users/:id      Bologna_2015.UserController :show
user_path  POST   /users          Bologna_2015.UserController :create
user_path  PATCH /users/:id      Bologna_2015.UserController :update
           PUT    /users/:id      Bologna_2015.UserController :update
user_path  DELETE /users/:id      Bologna_2015.UserController :delete
```

Path helpers

```
iex(1)> Bologna_2015.Router.Helpers.user_path(  
    Bologna_2015.Endpoint, :index)
```

"/users"

```
iex(2)> Bologna_2015.Router.Helpers.user_path(  
    Bologna_2015.Endpoint, :show, 1)
```

"/users/1"

Più compatto

```
iex(3)> import Bologna_2015.Router.Helpers
```

```
nil
```

```
iex(4)> user_path(Bologna_2015.Endpoint, :index)
```

```
"/users"
```

```
iex(5)> user_path(Bologna_2015.Endpoint, :show, 1)
```

```
"/users/1"
```

Ancora più compatto

```
iex(8)> alias Bologna_2015.Endpoint  
nil
```

```
iex(9)> user_path(Endpoint, :index)  
"/users"
```

```
iex(10)> user_path(Endpoint, :show, 1)  
"/users/1"
```

URL con parametri

```
iex(11)> user_path(Endpoint, :index,  
                  order: "reverse")
```

```
"/users?order=reverse"
```

```
iex(12)> user_url(Endpoint, :index,  
                  order: "reverse")
```

```
"http://localhost:4000/users?order=reverse"
```

Cos'è un Endpoint?

```
lib/bologna_2015/endpoint.ex
```

```
defmodule Bologna_2015.Endpoint do
  use Phoenix.Endpoint, otp_app: :bologna_2015
  ...
  plug Plug.Session,          # i plug modificano conn
    store: :cookie,
    key: "_bologna_2015_key",
    signing_salt: "PgkCXiY6"

  plug Bologna_2015.Router # definito in web/router.ex
end
```

Scoping delle rotte

```
scope "/admin" do
  resources "/users", Admin.UserController
end
```

| | | | | |
|-----------|--------|-----------------------|-----------------------------------|---------|
| user_path | GET | /admin/users | Bologna_2015.Admin.UserController | :index |
| user_path | GET | /admin/users/:id/edit | Bologna_2015.Admin.UserController | :edit |
| user_path | GET | /admin/users/new | Bologna_2015.Admin.UserController | :new |
| user_path | GET | /admin/users/:id | Bologna_2015.Admin.UserController | :show |
| user_path | POST | /admin/users | Bologna_2015.Admin.UserController | :create |
| user_path | PATCH | /admin/users/:id | Bologna_2015.Admin.UserController | :update |
| | PUT | /admin/users/:id | Bologna_2015.Admin.UserController | :update |
| user_path | DELETE | /admin/users/:id | Bologna_2015.Admin.UserController | :delete |

I controller

```
def show(conn, %{"id" => id}) do
  user = Repo.get!(User, id)
  render(conn, "show.html", user: user)
end
```

```
o anche:      conn
               |> assign(:user, user)
               |> render("show.html")
```

Dev procedurali: attenzione!

Funziona: conn
 |> assign(:user, user)
 |> render("show.html")

Non funziona: assign(conn, :user, user)
 render(conn, "show.html")

API JSON

```
def show(conn, %{"id" => id}) do
  user = Repo.get!(User, id)

  json conn, %{ id: user.id, email: user.email, inserted_at:
    user.inserted_at, updated_at: user.updated_at }
end
```

GET /admin/users/1

```
{"updated_at":"2015-10-10T09:47:04.528266Z",
"inserted_at":"2015-10-10T09:47:04.528266Z",
"id":1,"email":"paolo.montrasio@connettiva.eu"}
```

Redirect

```
def delete(conn, %{"id" => id}) do
  user = Repo.get!(User, id)
  Repo.delete!(user)
  conn
  |> put_flash(:info, "User deleted successfully.")
  |> redirect(to: user_path(conn, :index))
end
```

Cos'è un flash?

web/templates/layout/app.html.eex

```
<p class="alert alert-info" role="alert">  
  <%= get_flash(@conn, :info) %>  
</p>
```

```
<p class="alert alert-danger" role="alert">  
  <%= get_flash(@conn, :error) %>  
</p>
```

```
<%= @inner %>
```

Porting di una app a Phoenix

- Customers analytics per CheckBonus
<http://checkbonus.it/>
- Web app Rails
- Le pagine fanno richieste a Rails per mostrare tabelle e grafici
- Risposte JSON



Modelli

```
$ mix phoenix.gen.html Retailer retailers name:string internal_id:integer
* creating web/controllers/retailer_controller.ex
* creating web/templates/retailer/edit.html.eex
* creating web/templates/retailer/form.html.eex
* creating web/templates/retailer/index.html.eex
* creating web/templates/retailer/new.html.eex
* creating web/templates/retailer/show.html.eex
* creating web/views/retailer_view.ex
* creating test/controllers/retailer_controller_test.exs
* creating priv/repo/migrations/20150919101354_create_retailer.exs
* creating web/models/retailer.ex
* creating test/models/retailer_test.exs
```

Rotte e migrazioni

Aggiungere `resources "/retailers"`, `RetailerController`

Altrimenti:

```
$ mix ecto.migrate
```

```
== Compilation error on file web/controllers/retailer_controller.ex ==
```

```
** (CompileError) web/controllers/retailer_controller.ex:25: function  
retailer_path/2 undefined
```

```
(stdlib) lists.erl:1337: :lists.foreach/2
```

```
(stdlib) erl_eval.erl:669: :erl_eval.do_apply/6
```

```
(elixir) lib/kernel/parallel_compiler.ex:100: anonymous fn/4 in  
Kernel.ParallelCompiler.spawn_compilers/8
```


Migrazioni con Ecto

```
$ mix ecto.migrate          # up
```

```
$ mix ecto.rollback        # down di uno
```

<http://hexdocs.pm/ecto/Ecto.html>

Adapter per PostgreSQL, MySQL, MariaDB, MSSQL, MongoDB.

Il modello generato

```
defmodule Bologna_2015.Retailer do
```

```
  use Bologna_2015.Web, :model
```

```
  schema "retailers" do
```

```
    field :name, :string
```

```
    field :internal_id, :integer
```

```
    timestamps
```

```
    has_many :shops, Bologna_2015.Shop
```

```
    has_many :visits, Bologna_2015.Visit
```

```
  end
```

```
  @required_fields ~w(name)
```

```
  @optional_fields ~w(internal_id)
```

```
  def changeset(model, params \ \ :empty) do
```

```
    model
```

```
    |> cast(params, @required_fields,
```

```
              @optional_fields)
```

```
  end
```

```
end
```

Changeset e validazioni / 1

```
iex(5)> alias Bologna_2015.Retailer
```

```
iex(6)> changeset = Retailer.changeset(%Retailer{}, %{})
```

```
%Ecto.Changeset{action: nil, changes: %{}, constraints: [],
```

```
errors: [name: "can't be blank"], filters: %{},
```

```
model: %Bologna_2015.Retailer{__meta__: #Ecto.Schema.Metadata<:built>,
```

```
internal_id: nil, id: nil, inserted_at: nil, name: nil, updated_at: nil},
```

```
optional: [:internal_id], opts: nil, params: %{}, repo: nil,
```

```
required: [:name],
```

```
types: %{internal_id: :integer, id: :id, inserted_at: Ecto.DateTime,
```

```
name: :string, updated_at: Ecto.DateTime}, valid?: false, validations: []}
```

```
iex(7)> changeset.valid?
```

```
false
```

```
iex(8)> changeset.errors
```

```
[name: "can't be blank"]
```

Changeset e validazioni / 2

```
iex(10)> params = %{name: "Joe Example"}
```

```
%{name: "Joe Example"}
```

```
iex(11)> changeset = Retailer.changeset(%Retailer{}, params)
```

```
%Ecto.Changeset{action: nil, changes: %{name: "Joe Example"}, constraints: [],
```

```
errors: [], filters: %{}},
```

```
model: %Bologna_2015.Retailer{__meta__: #Ecto.Schema.Metadata<:built>,
```

```
internal_id: nil, id: nil, inserted_at: nil, name: nil, updated_at: nil},
```

```
optional: [:internal_id], opts: nil, params: %{"name" => "Joe Example"},
```

```
repo: nil, required: [:name],
```

```
types: %{internal_id: :integer, id: :id, inserted_at: Ecto.DateTime,
```

```
name: :string, updated_at: Ecto.DateTime}, valid?: true, validations: []}
```

```
iex(12)> changeset.valid?
```

```
true
```

```
iex(13)> changeset.errors
```

```
[]
```

Validazioni

```
def changeset(model, params \\ :empty) do
  model
  |> cast(params, @required_fields, @optional_fields)
  |> validate_confirmation(:password)
  |> validate_length(:password, min: 12)
  |> validate_number(:age)
  |> validate_inclusion(:age, 18..130)
  |> validate_format(:email, ~r/@/)
end
```

Registrazione e autenticazione

- L'ostacolo più grande all'adozione di Phoenix
- No framework con copertura di tutto lo use case
 - Registrazione
 - Invio mail di attivazione
 - Non ho ricevuto il link di attivazione
 - Ho perso la password
 - Faccio login / faccio logout
 - Mi autentico con FB / Tw / G+ / OAuth

Soluzioni

- Addict <https://github.com/trenpaxter/addict>
 - POST JSON per registrazione, login, logout, recupero e reset password: OK per SPA.
 - Mail via Mailgun
- Passport <https://github.com/opendrops/passport>
 - No routes, no controllers: un SessionManager da usare nel proprio codice
- Do it yourself
<http://nithinbekal.com/posts/phoenix-authentication/>

Do It Yourself: solo la login

/admin/users/5



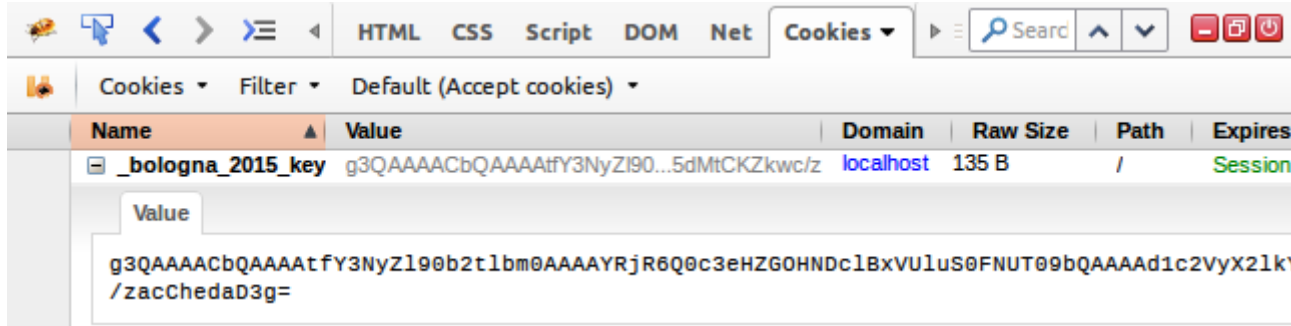
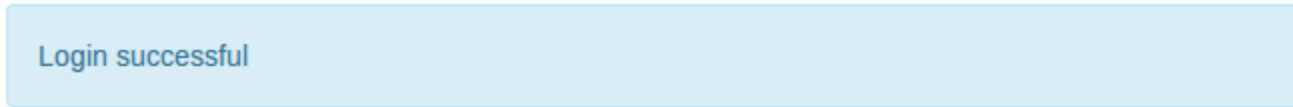
/sessions/new

Email

Password

→ /sessions/create

/admin/users/5



I file necessari

resources "/sessions", SessionController,
only: [:new, :create, :delete]

web/models/user.ex

web/controllers/session_controller.ex

web/views/session_view.ex

web/templates/session/new.html.eex

lib/bologna_2015/authentication.ex

lib/bologna_2015/must_be_logged_in.ex

Modello e cifratura password

```
schema "users" do
  field :email, :string
  field :encrypted_password, :string
end

@required_fields ~w(email encrypted_password)

def hash(plaintext) do
  Base.encode16(:crypto.hash(:sha256, to_char_list(plaintext)))
end
```

<https://www.djm.org.uk/cryptographic-hash-functions-elixir-generating-hex-digests-md5-sha1-sha2/>

Inserimento utenti / 1

Barando, direttamente nel db (PostgreSQL)

```
create extension pgcrypto;  
  
insert into users (email, encrypted_password,  
inserted_at, updated_at) values  
('paolo.montrasio@connettiva.eu',  
upper(encode(digest('password', 'sha256'),'hex')),  
now(), now());
```

Inserimento utenti / 2

Correttamente, in Elixir

```
$ iex -S mix
```

```
alias Bologna_2015.User
```

```
changeset = User.changeset(%User{  
  %{email: "paolo.montrasio@connettiva.eu",  
    encrypted_password: User.hash("password")})
```

```
alias Bologna_2015.Repo
```

```
Repo.insert(changeset)
```

Form di login

```
<form action="/sessions" method="post">  
  <input type="hidden" name="_csrf_token"  
    value="<%= get_csrf_token() %>">
```

Email

```
<input name="user[email]" type="email" value="" />
```

Password

```
<input name="user[password]" type="password" />
```

```
<input type="submit" value="Sign in" />
```

```
</form>
```

Controller per le sessioni

```
def create(conn, %{ "user" => %{ "email" => email, "password" => password }) do
  case User.find(email, password) do
    [user] ->
      fetch_session(conn)
      |> put_session(:user_id, user.id)           # user.id nella sessione per i controller
      |> put_flash(:info, "Login successful")
      |> redirect(to: page_path(conn, :index))
    [] ->
      fetch_session(conn)
      |> put_flash(:error, "Login failed")
      |> redirect(to: session_path(conn, :new))
  end
end

def find(email, password) do
  enc_pwd = hash(password)
  query = from user in User,
  where: user.email == ^email and
  user.encrypted_password == ^enc_pwd,
  select: user
  Repo.all(query)
end
```

Plug di autenticazione

```
defmodule Bologna_2015.Plugs.Authentication do
  import Plug.Conn
  alias Bologna_2015.User
  alias Bologna_2015.Repo
  def init(default), do: default
  def call(conn, _default) do
    user = nil
    user_id = get_session(conn, :user_id)
    unless user_id == nil do
      user = Repo.get(User, user_id)
    end
    assign(conn, :current_user, user)
  end
end
# conn.assigns[:current_user]
```

web/router.ex

```
defmodule Bologna_2015.Router do
  use Bologna_2015.Web, :router

  pipeline :browser do
    plug :accepts, ["html"]
    plug :fetch_session
    plug :fetch_flash
    plug :protect_from_forgery
    plug :put_secure_browser_headers
    plug Bologna_2015.Plugs.Authentication
  end
end
```

Plug di autorizzazione

```
defmodule Bologna_2015.Plugs.MustBeLoggedIn do
```

```
  import Plug.Conn
```

```
  import Phoenix.Controller
```

```
  def init(default), do: default
```

```
  def call(conn, _default) do
```

```
    if conn.assigns[:current_user] == nil do
```

```
      conn
```

```
      |> put_flash(:info, "You must be logged in")
```

```
      |> redirect(to: "/") |> halt
```

```
    else
```

```
      conn
```

```
    end
```

```
  end
```

```
end
```

```
web/controllers/admin/user_controller.ex
```

```
defmodule Bologna_2015.Admin.UserController do  
  use Bologna_2015.Web, :controller
```

```
  plug Bologna_2015.Plugs.MustBeLoggedIn
```


Funziona? mix test

```
defmodule Bologna_2015.SessionControllerTest do
  use Bologna_2015.ConnCase
  alias Bologna_2015.User
  @valid_attrs %{"email" => "paolo.montrasio@connettiva.eu", "password" => "password"}
  setup do
    conn = conn()
    {:ok, conn: conn}
  end
  test "creates session and redirects when data is valid", %{conn: conn} do
    changeset = User.changeset(%User{}, %{email: @valid_attrs["email"],
                                          encrypted_password: User.hash(@valid_attrs["password"])})
    {:ok, user } = Repo.insert(changeset)
    conn = post conn, session_path(conn, :create), user: @valid_attrs
    assert redirected_to(conn) == page_path(conn, :index)
    assert get_session(conn, :user_id) == user.id
  end
end
```

API JSON – di nuovo e meglio

```
pipeline :api do
```

```
  plug :accepts, ["json"]
```

```
  scope "/api", Bologna_2015, as: :api do
```

```
    resources "/retailers", API.RetailerController,  
              only: [:index] do
```

```
      resources "/visits", API.VisitController,  
                only: [:index]
```

```
    ...
```

```
api_retailer_path      GET /api/retailers          Bologna_2015.API.RetailerController :index  
api_retailer_visit_path GET /api/retailers/:retailer_id/visits Bologna_2015.API.VisitController :index
```

Visit: migrazione e modello

```
defmodule Bologna_2015.Repo.Migrations.CreateVisit do
```

```
  use Ecto.Migration
```

```
  def change do
```

```
    create table(:visits) do
```

```
      add :retailer_id, :integer
```

```
      add :started_at, :timestamp
```

```
      add :duration, :integer
```

```
    end
```

```
  end
```

```
end
```

```
defmodule Bologna_2015.Visit do
  use Bologna_2015.Web, :model
  schema "visits" do
    belongs_to :retailer, Bologna_2015.Retailer
    field :started_at, Ecto.DateTime
    field :duration, :integer
  end
  @required_fields ~w(retailer_id, started_at,
                       duration)
  @optional_fields ~w()
  def changeset(model, params \\ :empty) do
    model
    |> cast(params, @required_fields,
            @optional_fields)
  end
end
```

Generazione controller

mix phoenix.gen.json **API.Visit** visits --no-model

* creating web/controllers/**api/visit_controller.ex**

* creating web/views/api/visit_view.ex

* creating test/controllers/api/visit_controller_test.exs

* creating web/views/changeset_view.ex

Sostituire alias Bologna_2015.API.Visit

Con alias Bologna_2015.Visit

Il controller

```
def index(conn, _params) do
  retailer_id = conn.assigns[:retailer].id # da dove arriva?
  query = from visit in Visit,
    where: visit.retailer_id == ^retailer_id,
    select: visit

  visits = Repo.all(query)
  render(conn, "index.json", visits: visits) # dov'è il template?
end
```

Assign del modello

```
plug :assign_retailer
```

```
defp assign_retailer(conn, _options) do  
  retailer = Repo.get!(Bologna_2015.Retailer,  
    conn.params["retailer_id"])  
  assign(conn, :retailer, retailer)  
end
```

Il template / 1

```
# web/views/api/visit_view.ex
def render("index.json", %{visits: visits}) do
  %{data: render_many(visits, Bologna_2015.API.VisitView,
    "visit.json")}
end

# render_many? Circa equivalente a
Enum.map(visits, fn user ->
  render(Bologna_2015.API.VisitView, "visit.json", visit: visit)
end)
```

II template / 2

```
# web/views/api/visit_view.ex
```

```
def render("visit.json", %{visit: visit}) do
```

```
  %{id: visit.id}
```

```
end
```

```
- %{id: visit.id}
```

```
+ %{started_at: visit.started_at, duration: visit.duration}
```


La richiesta

GET /retailers/1/visits

```
{"data": [  
  {"started_at": "2015-09-29T20:11:00Z", "duration": 6},  
  {"started_at": "2015-09-29T20:41:00Z", "duration": 6},  
  ...  
]}
```

Benchmark Phoenix

```
query = from visit in Visit,  
  where: visit.retailer_id == ^retailer_id,  
  select: visit  
visits = Repo.all(query)
```

(252), 147, 134, 145, 133, 142 → **media 140 ms**
per 5000+ visits

Benchmark Rails

```
visits = Visit.where(  
  retailer_id: params[:retailer_id]).  
  pluck(:started_at, :duration)
```

(149), 117, 112, 124, 109, 122 → **media 116 ms**

Benchmark Rails

```
visits = Visit.where(  
  retailer_id: params[:retailer_id]).  
  pluck(:started_at, :duration)
```

(149), 117, 112, 124, 109, 122 → **media 116 ms**

Ma è un confronto onesto?

```
select * vs select started_at, duration
```

Benchmark Rails select *

```
visits = Visit.where(  
  retailer_id: params[:retailer_id])
```

(265), 236, 233, 230, 259, 282 → **media 248 ms**

Benchmark Phoenix

```
query = from visit in Visit,  
  where: visit.retailer_id == ^retailer_id,  
  select: [visit.started_at, visit.duration]  
visits = Repo.all(query)
```

(193), 85, 72, 79, 70, 68 → **media 74 ms**

Benchmark: riassunto

select * from visits

| | | | |
|---------|-----|----|--------|
| Phoenix | 140 | ms | |
| Rails | 248 | ms | x 1.71 |

select started_at, duration from visits

| | | | |
|---------|-----|----|--------|
| Phoenix | 74 | ms | |
| Rails | 116 | ms | x 1.56 |

Benchmark: riassunto

select * from visits

| | | | |
|---------------|------|----|--------|
| Phoenix | 140 | ms | |
| Rails | 248 | ms | x 1.71 |
| Ruby senza AR | 219 | ms | |
| PostgreSQL | 2.97 | ms | |

select started_at, duration from visits

| | | | |
|---------------|------|----|--------|
| Phoenix | 74 | ms | |
| Rails | 116 | ms | x 1.56 |
| Ruby senza AR | 88 | ms | |
| PostgreSQL | 3.47 | ms | |

Fastidi

- `alias / import / require` all'inizio di ogni file
- Mancanza di un framework di autenticazione
- Dover chiamare ogni tanto Erlang
- Dover scrivere due volte lo schema, nella migrazione e nel modello

Delizie

- Hot reload
- iex -S mix
- Channels
 - <https://medium.com/@azzarcher/the-simplicity-and-power-of-elixir-the-ws2048-case-b510eaa568c0>

Domande e contatti

Paolo Montrasio

paolo.montrasio@connettiva.eu

Slide a

connettiva.eu/phoenix-per-principianti.pdf