ERICH KADERKA

RUBY STORIES VII

PRIMEHAMMER.COM

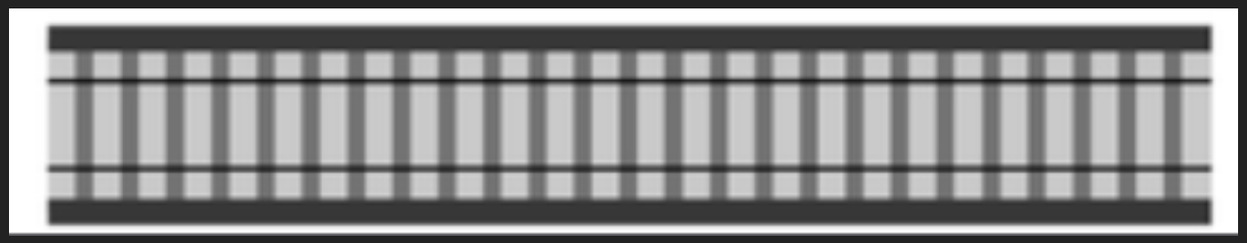
RAILWAY ORIENTED PROGRAMMING

WHAT IS RAILWAY ORIENTED PROGRAMMING?

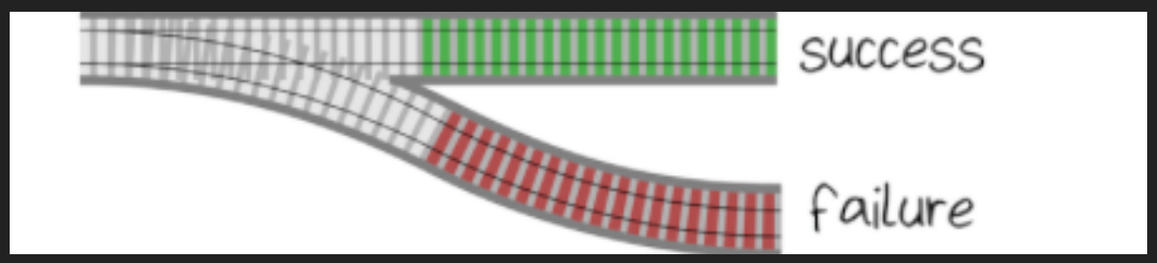
- The term itself was defined by Scott Wlaschin
- Many examples in functional programming assume that you are always on happy path. But in real world you must deal with validations, logging, errors, exceptions etc
- Railway Oriented Programming is a functional approach to error handling

WHY RAILWAY?

- One track function (happy path)

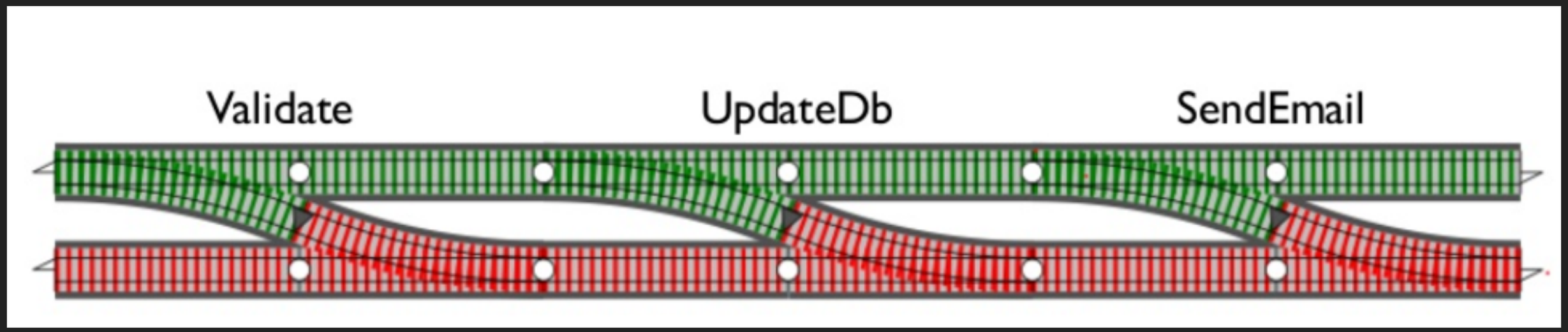


- Two track



REAL WORLD EXAMPLE

- Validate - Update DB - Send Email - Notify
- On success we pass through to the next function, on error we switch (bye-pass)



ELIXIR / PHOENIX APP

```elixir
def update_user_and_send_email(%User{} = user, attrs) do
  user
  |> update_user(attrs)
  |> send_notification_email()
  |> notify_admin()
end

def update_user(%User{} = user, attrs) do
  user
  |> User.changeset(attrs)
  |> Repo.update()
end

def send_notification_email(user) do
  {:error, "Invalid API key"}
end

def notify_admin(user) do
  {:ok}
end
```

```
iex(17)> Scotty.Accounts.update_user_and_send_email(user, %{age: 13})
[debug] QUERY OK db=9.3ms
UPDATE "users" SET "age" = $1, "updated_at" = $2 WHERE "id" = $3 [13, {{2019, 1, 28}, {16, 9, 22, 541727}}, 1
{:ok}
```

ROP

- Hex package
- Introduces new operator - >>> (just a simple macro)

```
def update_user_and_send_email(%User{} = user, attrs) do
  user
  |> update_user(attrs)
  >>> send_notification_email()
  >>> notify_admin()
end
```

ROP RESULT

- Returns tuple with the error
- Doesn't execute notify function

```
iex(3)> Scotty.Accounts.update_user_and_send_email(user, %{age: 21})
[debug] QUERY OK db=2.3ms
UPDATE "users" SET "age" = $1, "updated_at" = $2 WHERE "id" = $3 [21, {{2019, 1, 28}, {16, 28, 13, 118535}}, 1]
{:error, "Invalid API key"}
```

EXCEPTION

- Handle exception with try_catch function

```
def update_user_and_send_email(%User{} = user, attrs) do
  user
  |> update_user(attrs)
  >>> try_catch(send_notification_email())
  >>> notify_admin()
end

def send_notification_email(user) do
  raise "Exception"
end
```

```
iex(9)> Scotty.Accounts.update_user_and_send_email(user, %{age: 21})
[debug] QUERY OK db=8.0ms
UPDATE "users" SET "age" = $1, "updated_at" = $2 WHERE "id" = $3 [21, {{2019, 1, 28}, {16, 34, 19, 758025}}, 1]
{:error, %RuntimeError{message: "Exception"}}
```

TRAILBLAZER RAILS EXAMPLE

- Trailblazer operation - service object, remove logic from the controller and model and provide separate and, streamline object for it
- Call api several times, create user, assign role, revert changes

```
class User::CreateFromTeamInvite < Trailblazer::Operation
  step :update_rs_user
  step :create_rs_user
  step :create_person
  step :assign_rs_user_roles
  fail :revert_changes
```

TB CODE EXAMPLE

- Railway.pass!, Railway.fail! (fast options, fail step)

```ruby
def assign_rs_user_roles(options, realsavvy_client:, team_invite:, **)
  realsavvy_user_id = options['user'].realsavvy_id

  begin
    realsavvy_client.assign_roles_to_user(params: {
      "id" => realsavvy_user_id,
      "roles" => [])]
    })
    Railway.pass!
  rescue => e
    options['error'] = e
    Railway.fail!
  end
end

def revert_changes(options, team_invite:, **)
  options['user'].destroy if options['user']
  team_invite.status = 'pending'
  team_invite.save!
end
```

RESOURCES

- Web (F#) https://fsharpforfunandprofit.com/rop/
- Elixir presentation https://speakerdeck.com/cjbell88/railway-programming-in-elixir-with-with
- Domain Modeling Made Functional (book, DDD) https://pragprog.com/book/swdddf/domain-modeling-made-functional