Ecto made ~~simple~~... even simpler

Bernhard Stöcker

Recogizer Group GmbH

bernhard.stoecker@recogizer.de

October 11, 2017

Queries and Problems

Queries and Problems

Queries and Problems

```
def folder_by_parent(parent_id) do
  Folder
  |> where(^[{:parent_id, parent_id}])
  |> YourApp.Repo.all
end
```

Queries and Problems

```
def folder_by_parent(parent_id) do
  Folder
  |> where(^[{:parent_id, parent_id}])
  |> YourApp.Repo.all
end
```

What about the root folder?

Queries and Problems

```
def folder_by_parent(parent_id) do
  Folder
  |> where(^[{:parent_id, parent_id}])
  |> YourApp.Repo.all
end
```

What about the root folder?

```
folder_by_parent(nil)
```

```
# ** (ArgumentError) nil given for :parent_uuid.
# Comparison with nil is forbidden as it is unsafe.
# Instead write a query with is_nil/1, for example: is_nil(s.parent_uuid)
  (ecto) lib/ecto/query/builder/filter.ex:127: Ecto.Query.Builder.Filter.kw!/7
  (ecto) lib/ecto/query/builder/filter.ex:120: Ecto.Query.Builder.Filter.kw!/3
  (ecto) lib/ecto/query/builder/filter.ex:102: Ecto.Query.Builder.Filter.filter!/6
  (ecto) lib/ecto/query/builder/filter.ex:114: Ecto.Query.Builder.Filter.filter!/7
```

Queries and Problems

```
def folder_by_parent(nil) do
  query = from f in Folder,
          where: is_nil(field(f, :parent_uuid))
  YourApp.Repo.all(query)
end
def folder_by_parent(org_id) do
  Folder
  |> where(^[{:parent_id, org_id}])
  |> YourApp.Repo.all
end
```

Queries and Problems

What when I need data scoped to several organizations?

Queries and Problems

```
def folders_by_orgs(org_ids) do
  query = from f in Folder,
          where([f], field(f, :org_id) in ^org_ids)
  YourApp.Repo.all(query)
end
```

Queries and Problems

What do queries look like in real life?

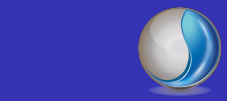
Queries and Problems

- Belongs to an organization from a list (thinking of authentication/authorization)
- Is a rootfolder
- Has some state (e.g. 'active')

Queries and Problems

```
def active_rootfolder(org_ids) do
  Folder
  |> fn queriable ->
    from m in queriable,
    where: is_nil(field(m, :parent_id))
  end.()
  |> where([m], field(m, :org_id) in ^org_ids)
  |> where(^[{:state, "active"}])
  |> YourApp.Repo.all
end
```

Queries and Problems



What if these statements can be expressed in a simple keyword list?

SimpleRepo

```
def active_rootfolder(org_ids) do
  Folder
  |> fn queriable ->
    from m in queriable,
    where: is_nil(field(m, :parent_id))
  end.()
  |> where([m], field(m, :org_id) in ^org_ids)
  |> where(^[{:state, "active"}])
  |> YourApp.Repo.all
end
```

```
def active_rootfolder(org_ids) do
  SimpleRepo.Query.scoped(
    Folder,
    [parent_id: nil, org_id: org_ids, state: "active"]
  )
  |> YourApp.Repo.all
end
```

SimpleRepo

Even shorter? Redefine Repo:

```
defmodule YourApp.Repo do
  use Ecto.Repo, otp_app: :your_app
  use SimpleRepo.Scoped, repo: __MODULE__
end
```

SimpleRepo

Even shorter? Redefine Repo:

```
defmodule YourApp.Repo do
  use Ecto.Repo, otp_app: :your_app
  use SimpleRepo.Scoped, repo: __MODULE__
end
```

In code:

```
def active_rootfolder(org_ids) do
  YourApp.Repo.all_scoped(
    Folder,
    [parent_id: nil, org_id: org_ids, state: "active"]
  )
end
```

SimpleRepo

Queries

- {key, nil}
- {key, value}
- {key, value_list}
- {key, {:like, pattern}}
- {key, {:not, nil}}
- {key, {:not, value}}
- {key, {:not, value_list}}
- {key, {:not_like, pattern}}
- {key, {:<, value}}
- {key, {:<=, value}}
- {key, {:>, value}}
- {key, {:>=, value}}

SimpleRepo

Scope Functions

- by_id_scoped\4
- one_scoped\3
- all_scoped\3
- update_scoped\5
- update_all_scoped\4
- delete_scoped\4
- delete_all_scoped\3
- aggregate_scoped\5

SimpleRepo

You like it? Then use it. And if you have more ideas: Contribute

- github.com/xofspades/simple_repo

SimpleRepo

You like it? Then use it. And if you have more ideas: Contribute

- github.com/xofspades/simple_repo
- github.com/xofspades/simple_controller

SimpleRepo used at

We are hiring:

- Backend Developer
- Data Scientist

Using:

- Scala
- Python
- Elixir
- R