Hotwire

by Jackson Pires

Conhecendo o Turbo Stream

Conhecendo o Turbo Stream

No exemplo base podemos ver que o Turbo Drive continua funcionando.

O que queremos agora é alterar múltiplas partes do documento HTML.

Ao tentar isolar com turbo frame os dois locais que vamos atualizar, percebemos que temos um problema… os IDs do turbo frame não podem colidir, e ao mesmo tempo temos dois locais para atualizar. É aqui que entra o Turbo Stream.

Conhecendo o Turbo Stream

Vamos começar pensando seguinte… O Turbo Stream não precisa de `turbo_frame_tag` pois ele pode trabalhar diretamente com elementos HTML.

Então vamos começar usando uma DIV para envolver o que queremos atualizar (dentro das partials). O importante aqui é termos um ID único para os elementos que vamos atualizar.

Conhecendo o Turbo Stream

```
<div id="users_status_count">

  <h2><%= "Actives: #{@users.where(active: true).count} / Inactives:
#{@users.where(active: false).count} " %></h2>

</div>
```

OU

```
<%= content_tag(:div, id: :users_status_count do %>

  <h2><%= "Actives: #{@users.where(active: true).count} / Inactives:
#{@users.where(active: false).count} " %></h2>

<% end %>
```

Conhecendo o Turbo Stream

```
<%= content_tag(:div, id: "change_status_button_#{dom_id(user)}") do %>
  <p>
    <% user_status_label = user.active ? "Inactivate!" : "Activate!" %>
    <%= button_to user_status_label, change_status_user_path(user) %>
  </p>
<% end %>
```

Conhecendo o Turbo Stream

Remover da action `change_status` em `users_controller.rb`

~~`redirect_to users_url`~~

OU use o `respond_to`

```
respond_to do |format|
  format.turbo_stream
  format.html { redirect_to users_url}
end
```

Conhecendo o Turbo Stream

Criar um arquivo: `views/users/change_status.turbo_stream.erb`

Como funcionam as atualizações do Turbo Stream?

Como funcionam as atualizações do Turbo Stream?

Documentação: https://turbo.hotwired.dev/handbook/streams

Referência das sete possíveis ações: https://turbo.hotwired.dev/reference/streams

Como funcionam as atualizações do Turbo Stream?

```
# views/users/change_status.turbo_stream.erb

<%= turbo_stream.replace :users_status_count do %>
  <%= render "users_status_count" %>
<% end %>

<%= turbo_stream.replace
"change_status_button_#{dom_id(@user)}" do %>
  <%= render "change_status_button", user: @user %>
<% end %>

<%= turbo_stream.replace dom_id(@user) do %>
  <%= render "user", user: @user %>
<% end %>
```

Como funcionam as atualizações do Turbo Stream?

Também é possível fazer respostas `in line` para o turbo stream, direto do controller, algo como:

```
  format.turbo_stream do
    render turbo_stream: turbo_stream.replace
:users_status_count do
       render "users_status_count"
    end
     ...
  end
```

Persistindo um estado entre páginas

Persistindo um estado entre páginas

Use o data attribute abaixo no wrapper a ser persistido.

```
data-turbo-permanent // "data-turbo-permanent": ''
```

Importante notar…

Importante notar…

- `button_to` vs `link_to`.
- Observar requisição `TURBO_STREAM` no log do Rails.
- Observar requisição no dev tools com `Accept` contendo `text/vnd.turbo-stream.html` e é assim que o Rails sabe qual template renderizar `(.html.erb ou .turbo_stream.erb)` em conjunto com o `respond_to`.
- Quando usamos turbo stream a header **`Turbo-Frame`** não é enviada na requisição.