Hotrails

← Back to the list of chapters

Two ways to handle empty states with Hotwire

Published on March 01, 2022

In this chapter, we will learn two ways to handle empty states with Turbo. The first one uses Turbo Frames and Turbo Streams, and the second uses the only-child CSS pseudo-class.

Adding empty states to our Ruby on Rails applications

Empty states are an important part of our applications. When arriving on a web page for the first time, we won't have any data to help us guess what the page is used for. As a new user of an application, it's nice to have an image or a few sentences to explain the actions we can perform on the page.

If we destroy all the quotes in our quote editor, we currently have a blank page with just a title and a button. It would be nice to display an empty state when there are no quotes to help our users.

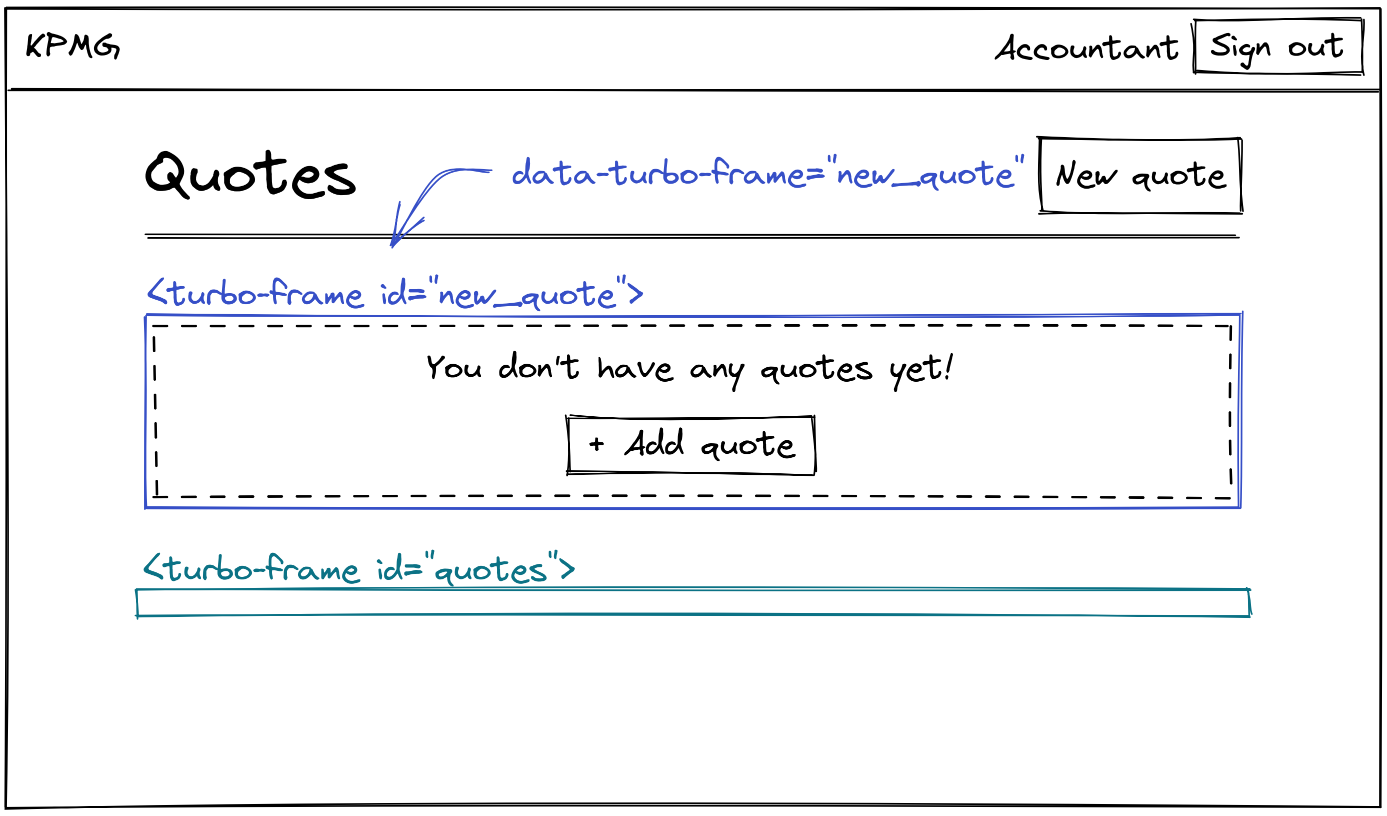
In this short chapter, we will learn two ways to add empty states to our Ruby on Rails 7 applications:

1. The first one will use Turbo Frames and Turbo Streams
2. The second one will use the `:only-child` CSS pseudo-class

Let's get started!

Empty states with Turbo Frames and Turbo Streams

Before diving into the code, let's take some time to sketch what we will build. When a user has no quote, we want to display an empty state containing a "helpful message" and a call to action:

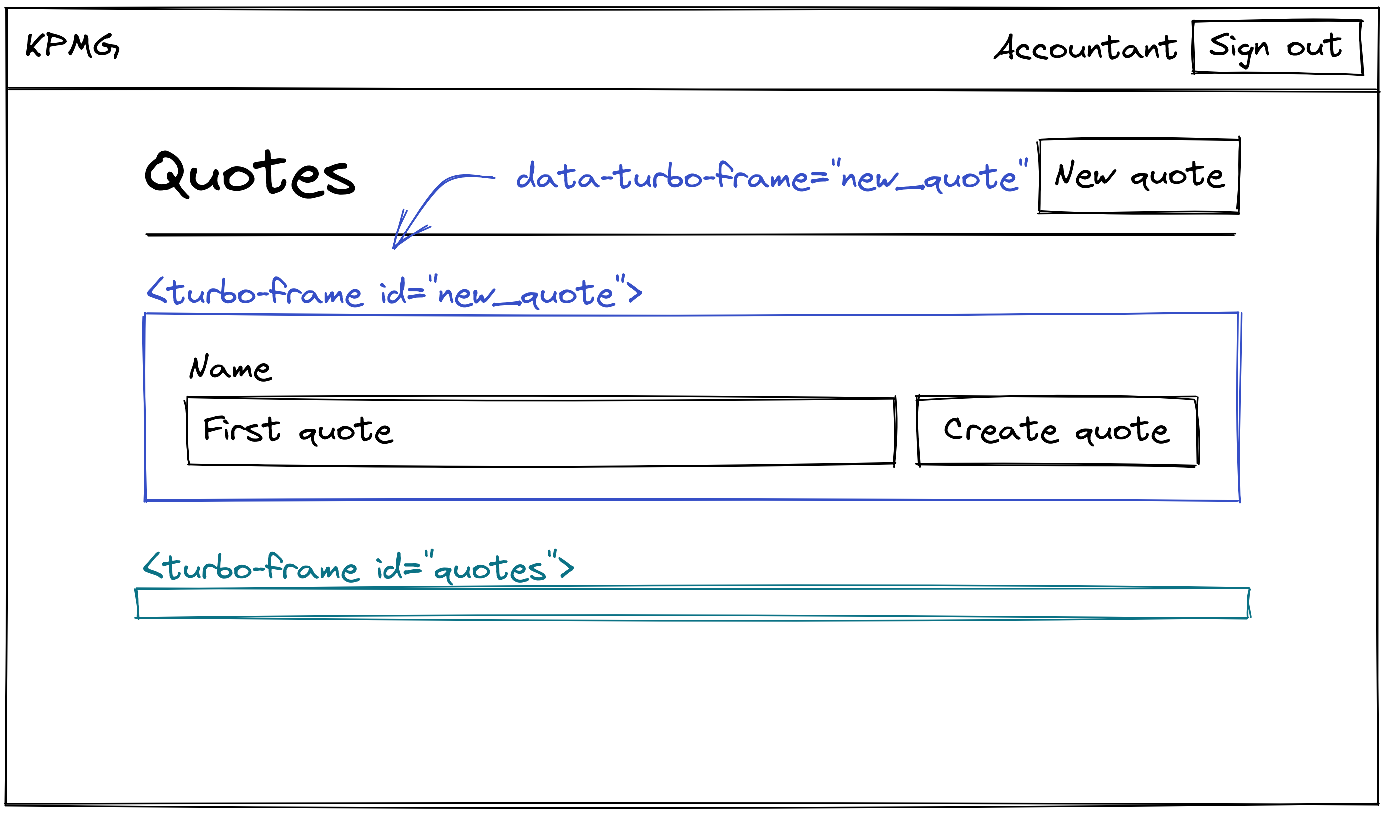

KPMG
Accountant
Sign out
Quotes
data-turbo-frame="new_quote"
New quote
<turbo-frame id="new_quote">
You don't have any quotes yet!
+ Add quote
<turbo-frame id="quotes">

Sketch of the Quotes#index page with the empty state

We will add the empty state within the Turbo Frame with id `new_quote`, so that:

- If the user clicks on the "New quote" button within the header, Turbo will replace the Turbo Frame with id `new_quote` on the `Quotes#index` page with the one extracted from the `Quotes#new` page, thanks to the `data-turbo-frame="new_quote"` data attribute.
- If the user clicks on the "Add quote" button within the empty state, Turbo will replace the Turbo Frame with id `new_quote` from the `Quotes#index` page with the one extracted from the `Quotes#new` page, as the link is within this Turbo Frame.

As described above, when the user clicks either of the two links, the empty state will be replaced by the new quote form. The state of the page when the user clicks on the "New quote" or the "Add quote" button is described in the following sketch:

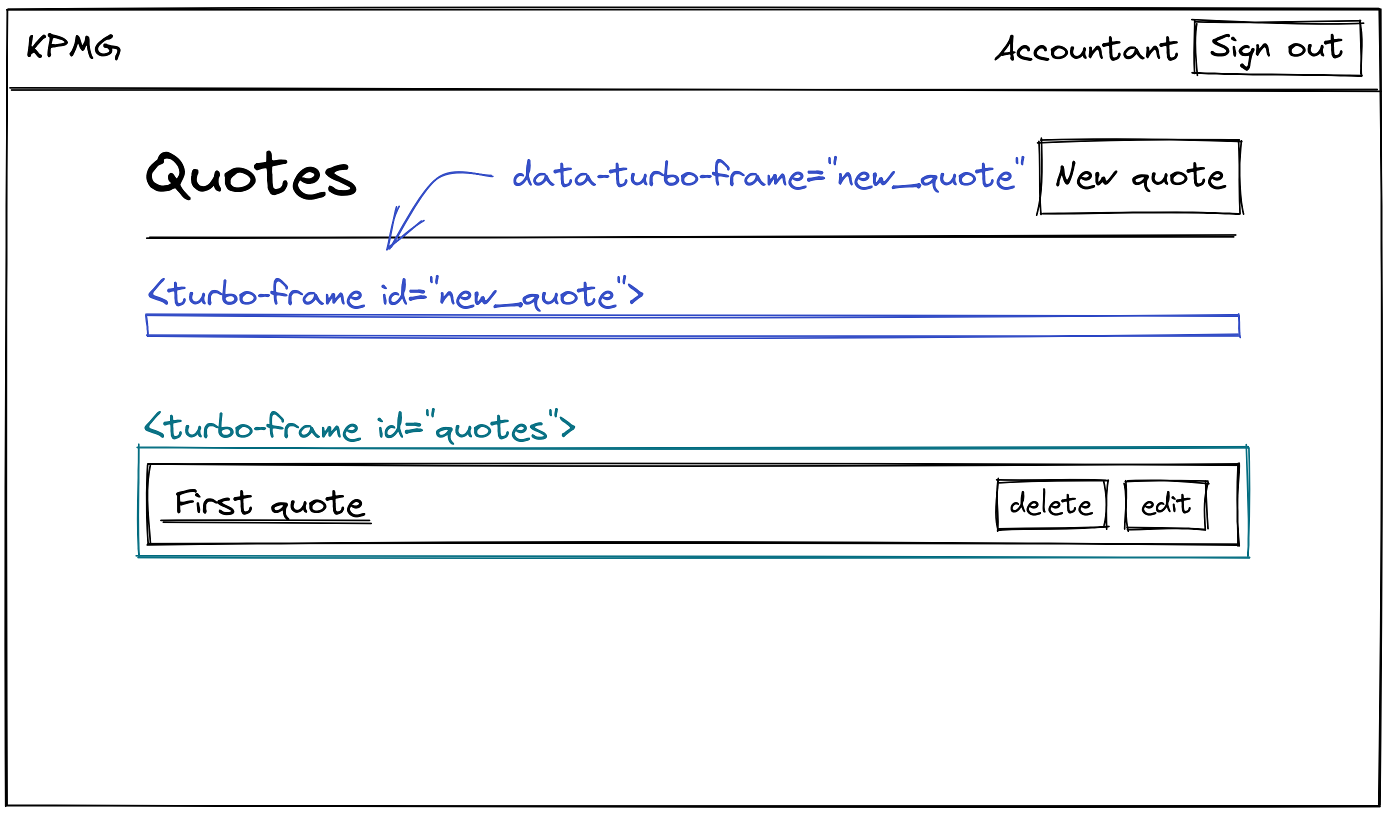

KPMG
Accountant
Sign out
Quotes
data-turbo-frame="new_quote"
New quote
<turbo-frame id="new_quote">
Name
First quote
Create quote
<turbo-frame id="quotes">

Sketch of the Quotes#index page with the empty state replaced

When the user submits a valid form, the behavior does not change:

1. The created quote is *prepended* to the list of quotes
2. The HTML contained in the Turbo Frame with id `new_quote` is removed

This behavior is described in the following sketch:


KPMG
Accountant
Sign out
Quotes
data-turbo-frame="new_quote"
New quote
<turbo-frame id="new_quote">
<turbo-frame id="quotes">
First quote
delete
edit

Sketch of the Quotes#index page without the empty state

If we refresh the page, **the empty state should not be visible anymore as there is at least one quote on the page**.

Now that the requirements are clear let's start coding! The first thing we have to do is display the empty state only when there are no quotes on the page. To do this, let's create an **empty state partial** that we can then use on the `Quotes#index` page:

```
<%# app/views/quotes/_empty_state.html.erb %>

<div class="empty-state">
  <p class="empty-state__text">
    You don't have any quotes yet!
  </p>

  <%= link_to "Add quote", new_quote_path, class: "btn btn--primary" %>
</div>
```

We can now render this empty state only when the current user has no quote on the `Quotes#index` page:

```
<%# app/views/quotes/index.html.erb %>

<%= turbo_stream_from current_company, "quotes" %>

<div class="container">
  <div class="header">
    <h1>Quotes</h1>
    <%= link_to "New quote",
                new_quote_path,
                class: "btn btn--primary",
                data: { turbo_frame: dom_id(Quote.new) } %>
  </div>

  <%= turbo_frame_tag Quote.new do %>
    <% if @quotes.none? %>
      <%= render "quotes/empty_state" %>
    <% end %>
  <% end %>

  <%= turbo_frame_tag "quotes" do %>
    <%= render @quotes %>
  <% end %>
</div>
```

Before we test in the browser, let's also style our empty state to make it a bit nicer:

```
// app/assets/stylesheets/components/_empty_state.scss

.empty-state {
  padding: var(--space-m);
  border: var(--border);
  border-style: dashed;
  text-align: center;

  &__text {
    font-size: var(--font-size-l);
    color: var(--color-text-header);
    margin-bottom: var(--space-l);
    font-weight: bold;
  }
}
```

Let's not forget to import this file into our manifest file for the styles to apply:

```
// app/assets/stylesheets/application.sass.scss

// All the previous code
@import "components/empty_state";
```

We are now ready to experiment in the browser.

On the `Quotes#index` page, let's delete all the quotes. We can then click either on the "New quote" or "Add quote" button. We should see that the form to create a new quote replaces the empty state. If we submit a valid form, the created quote is *prepended* to the list of quotes, and the empty state isn't visible anymore!

However, there is a small improvement we could make. If we delete the quote we just created, the empty state does not appear back on the screen. **We want the empty state to be always present when there is no quote on the page**. To do this, we must instruct the `destroy.turbo_stream.erb` view to update the content of the Turbo Frame with id `new_quote` with the content of the `quotes/empty_state` partial when there are no quotes on the page:

```
<%# app/views/quotes/destroy.turbo_stream.erb %>

<%= turbo_stream.remove @quote %>
<%= render_turbo_stream_flash_messages %>

<% unless current_company.quotes.exists? %>
  <%= turbo_stream.update Quote.new do %>
    <%= render "quotes/empty_state" %>
  <% end %>
<% end %>
```

With those lines of code added, our empty state now behaves as we want! Perfect!

However, there is one small issue with the current implementation that we could easily overlook. Since Chapter 5 and Chapter 6, we are streaming quotes creations, updates, and deletions made by all the users of our company on the `Quotes#index` page. Therefore, **if someone creates a quote while we are on the empty page, the quote will be added to the list, and the empty state will still be visible on the page**.

Let's discuss why this use case is important and how we can solve this in the next section.

Empty states with the only-child CSS pseudo-class

Before we talk about the second way of handling empty states with Turbo, let's try to reproduce the issue in the browser. Let's navigate to the `Quotes#index` page and delete all the quotes. Let's then create a new quote **from the console**. The created quote is broadcasted to our view, and we see that **both the created quote the empty state are visible on the page**.

Note: Our example with *quotes* seems a bit cumbersome. However, let's imagine for a second those *quotes* were *notifications*, like for example, Github notification:

1. When we have no notifications on the page, we want to see the empty state
2. When a notification is pushed to our view, we want the empty state to disappear
3. When we delete the notification, we want the empty state to appear back on the screen

That's the behavior we will implement in this section, and notifications are a great use case.

Let's analyze the issue here. As explained in Chapter 5 and Chapter 6, thanks to the `broadcasts_to` method in the `Quote` model:

- When a quote is created, the content of the `quotes/_quote.html.erb` partial is prepended to the list of quotes
- When a quote is deleted, the quote is removed from the list

By default, there is no mention of empty states. While we *could* play with the callbacks and override the default options of the `broadcasts_to` method, there is an elegant way to achieve what we want, thanks to the `:only-child` pseudo-class in CSS. The behavior we want to achieve is the following:

- When the empty state is the **only child of the quotes list**, we want it to be visible
- When the empty state is **not the only child** of the quotes list, we want it to be **invisible**

The behavior we want is slightly different from the first method as we won't replace the empty state with the new quote form this time.

Let's start coding! First, we need to move the content of the `quotes/empty_state` partial to the quotes list:

```
<%# app/views/quotes/index.html.erb %>

<%= turbo_stream_from current_company, "quotes" %>

<div class="container">
  <div class="header">
    <h1>Quotes</h1>
    <%= link_to "New quote",
                new_quote_path,
                class: "btn btn--primary",
                data: { turbo_frame: dom_id(Quote.new) } %>
  </div>

  <%= turbo_frame_tag Quote.new %>

  <%= turbo_frame_tag "quotes" do %>
    <%= render "quotes/empty_state" %>
    <%= render @quotes %>
  <% end %>
</div>
```

Then, we have to use the `:only-child` pseudo-class in our CSS to show the empty state when it is the only child of the Turbo Frame with id `quotes` and hide it when it is not:

```
// app/assets/stylesheets/components/_empty_state.scss

.empty-state {
  padding: var(--space-m);
  border: var(--border);
  border-style: dashed;
  text-align: center;

  &__text {
    font-size: var(--font-size-l);
    color: var(--color-text-header);
    margin-bottom: var(--space-l);
    font-weight: bold;
  }

  &--only-child {
    display: none;

    &:only-child {
      display: revert;
    }
  }
}
```

We use what the BEM methodology calls a *modifier* here for our `.empty-state--only-child` CSS class because we want to support the two methods presented in this chapter with the same `.empty-state` class.

In our empty state partial, we need the "Add quote" link to **explicitly target** the Turbo Frame with the id of `new_quote` as it is no longer a child of the Turbo Frame. We can achieve this thanks to the `data-turbo-frame="new_quote"` data attribute:

```
<%# app/views/quotes/_empty_state.html.erb %>

<div class="empty-state empty-state--only-child">
  <p class="empty-state__text">
    You don't have any quotes yet!
  </p>

  <%= link_to "Add quote",
              new_quote_path,
              class: "btn btn--primary",
              data: { turbo_frame: dom_id(Quote.new) } %>
</div>
```

We can also reset the content of the `destroy.turbo_stream.erb` view as we don't need any custom behavior anymore:

```
<%# app/views/quotes/destroy.turbo_stream.erb %>

<%= turbo_stream.remove @quote %>
<%= render_turbo_stream_flash_messages %>
```

We can now test the behavior in our browser:

- When we have quotes in the list, the empty state is not visible
- When we don't have quotes in the list, the empty state is visible

The best part is that we could achieve this behavior with CSS only!

Wrap up

In this chapter, we saw two methods to manage empty states with Turbo Frames and Turbo Streams.

In the first method, we used Turbo Frames and Turbo Streams to precisely add/remove the empty state on the `Quotes#index` page when required. While this method is great in most cases, it might not be the best fit when HTML is broadcasted to the view.

In the second method, we leveraged the power of the `:only-child` CSS pseudo-class to do all the work for us. We didn't have to write any custom Turbo/Turbo Rails related code!

In the following three chapters, we will work on the `Quotes#show` page to finalize our quote editor! See you there!

← previous next →