← Volver a la lista de capítulos

Introducción al tutorial de Turbo Rails

Publicado el 5 de enero de 2022

En este capítulo, explicaremos lo que vamos a aprender, veremos el producto terminado y pondremos en marcha nuestra nueva aplicación Rails 7.

¿Qué vamos a construir?

En este tutorial, crearemos un **editor de citas reactivo de una sola página con solo una línea de JavaScript personalizado** para la animación del mensaje flash. Puede ver el proyecto terminado en la página del editor de citas de este sitio web. ¡Adelante! Cree una cita, haga clic en ella y comience a agregar fechas y elementos para ver lo que crearemos.

Con el editor de cotizaciones, puede crear, actualizar y eliminar cotizaciones. Al hacer clic en una cotización, accede a la página para crear, actualizar y eliminar fechas. En cada fecha, puede agregar partidas. Cada vez que crea, actualiza o elimina una partida, se actualiza el total de la cotización.

Este proyecto está fuertemente inspirado en un proyecto que tuve que desarrollar con React. Cuando `turbo-rails` se lanzó en diciembre de 2020, la primera línea del README era: ***"Turbo te da la velocidad de una aplicación web de una sola página sin tener que escribir JavaScript"*** .

Trenes calientes

reconstruí el editor de citas con Turbo. **Me sorprendió lo fácil que era trabajar**

con él . ¿Y la mejor parte? ¡No más React, no más Redux, no más Formik! En cambio, podía trabajar con las herramientas que amo y conozco bien: Ruby on Rails y Simple Form. ¿Aburrido? Sí, pero podía obtener los beneficios de React con una décima parte del esfuerzo.

¿Por qué aprender Ruby on Rails 7 con Turbo?

Con el lanzamiento de Ruby on Rails 7 en diciembre de 2021, **Hotwire** , que es la combinación de **Stimulus y Turbo** , se convirtió en el marco de interfaz predeterminado para las aplicaciones Rails.

Stimulus ya existe desde hace un tiempo y es una biblioteca muy conocida en el ecosistema Rails. Por otro lado, Turbo y su integración con Ruby on Rails son herramientas completamente nuevas con características impresionantes.

- En primer lugar, todos los clics en enlaces y envíos de formularios son ahora solicitudes AJAX, lo que acelera nuestras aplicaciones gracias a *Turbo Drive* . **Obtenemos este beneficio de forma gratuita, ya que no requiere ningún trabajo; solo tenemos que importar la biblioteca** .
- En segundo lugar, ahora es muy fácil crear aplicaciones dinámicas dividiendo las páginas en diferentes partes con *Turbo Frames* , con solo unas pocas líneas de código . Desarrollamos nuestros controladores CRUD como lo hacíamos antes y, con solo agregar unas pocas líneas de código, **podemos reemplazar o cargar de forma diferida partes independientes de la página** .
- En tercer lugar, resulta **trivial agregar funciones en tiempo real** con la ayuda de *Turbo Streams* . ¿Quieres agregar notificaciones en tiempo real a tu aplicación, crear un juego multijugador en tiempo real o un sistema de monitoreo de errores en tiempo real? ¡La parte en tiempo real requiere solo unas pocas líneas de código con Turbo!

Turbo acelera las aplicaciones Rails, reduce la cantidad de JavaScript que tenemos que escribir y facilita el trabajo con funciones en tiempo real. Lo mejor de todo es que es fácil de aprender y, **al leer este tutorial, sabrás todo lo que hay que saber sobre las tres partes de Turbo** .

¿A quién va dirigido este libro?

En este libro:

- Crear controladores CRUD
- Crea nuestro sistema de diseño
- Configurar la autenticación con la gema Devise
- Obtenga más información sobre Turbo Drive, Turbo Frames y Turbo Streams

Si ya estás familiarizado con los puntos 1 a 3 y quieres aprender sobre el 4, ¡este tutorial es para ti!

Configuración de la aplicación

Vamos a crear nuestra nueva aplicación Rails. Usaremos Sass como preprocesador CSS para crear nuestro sistema de diseño, esbuild para agrupar nuestra línea única de JavaScript y una base de datos PostgreSQL para poder implementar nuestra aplicación en Heroku al final del tutorial.

Ahora podemos crear nuestra aplicación:

```
rails new quote-editor --css=sass --javascript=esbuild --database=postgr
```

Este tutorial fue escrito para la versión 1.0.X de Turbo-Rails (la versión de Turbo-Rails que se lanzó al mismo tiempo que Rails 7).

Como este tutorial se escribió en la época de Rails 7.0.0, asegurémonos de usar la versión de Turbo-Rails que se usaba en ese momento para evitar problemas inesperados. Actualicemos nuestro Gemfile bloqueando la versión de Turbo-Rails:

```
# Gemfile

gem "turbo-rails", "~> 1.0"
```

Ahora podemos ejecutar `bundle install` para instalar la versión correcta de la gema.

Ahora que nuestra aplicación está lista, escribamos el `bin/setup` comando para instalar las dependencias y crear la base de datos:

```
bin/setup
```

Ahora podemos ejecutar el servidor Rails y los scripts que precompilan el CSS y el código JavaScript con el `bin/dev` comando:

```
bin/dev
```

Ahora podemos ir a `http://localhost:3000` y deberíamos ver la pantalla de inicio de Rails.

Nota sobre los scripts `bin/setup` y el `bin/dev` :

Es una buena práctica tener un `bin/setup` script sólido para configurar la aplicación para nosotros: instalar las gemas, las dependencias de JavaScript, crear, migrar y sembrar la base de datos.

Es casi imprescindible cuando se trabaja en equipo, ya que debería ser fácil para los nuevos desarrolladores configurar el entorno de desarrollo. Por supuesto, podríamos documentar el proceso, pero la documentación queda obsoleta, ¡pero el código no! Si el script falla, alguien lo arreglará.

Incluso cuando trabajamos en un proyecto pequeño, usar el script tiene sus ventajas. Cada vez que necesitamos empezar desde cero, sabemos `bin/setup` que nos respalda.

El `bin/dev` script instala Foreman localmente y ejecuta la aplicación según el `Procfile.dev` archivo. Al ejecutar el `bin/dev` comando, ejecutamos tres comandos a la vez:

```
# Procfile.dev

web: bin/rails server -p 3000
js: yarn build --watch
css: yarn build:css --watch
```

Ya conocemos el primer comando `bin/rails server -p 3000` para iniciar el servidor Rails. Los otros dos comandos `yarn build --watch` y `yarn build:css --watch` están definidos en la sección de scripts del `Package.json` . Se encargan de precompilar nuestro código CSS y JavaScript

antes de entregarlos al flujo de activos. La `--watch` opción está aquí para garantizar que el código CSS y JavaScript se compile cada vez que guardemos un archivo CSS/Sass o JavaScript.

Ambos scripts se encuentran en la `/bin` carpeta de tu aplicación Rails si quieres echarles un vistazo.

Ya está todo listo para que comencemos a codificar. Comenzaremos a crear nuestra aplicación en el próximo capítulo. ¡Nos vemos allí!

Siguiente →