Improved Navigation in Angular 7 with switchMap

Nate Lapinski Follow
Dec 20, 2018 · 4 min read

Pushing Devils

With PR #25740 all navigations performed by Angular's router are merged into a single, observable stream. There can also be only one active navigation at any time. This provides the benefits of making navigations faster and more predictable. Although these changes are mostly internal, they do affect how we should think about navigations and routing in our applications.

> *This is a major refactor of how the router previously worked. There are a couple of major advantages to this refactor and the future work will be built on top of it.*

These changes were released as part of Angular 7.0. In this article we'll examine these changes and see how they can be used. We'll also see how `switchMap` enforces that only one navigation can exist at any one time.

Understanding Navigation

A navigation happens whenever the URL changes. This can be the result of some imperative action (a service calling `navigate`, or a guard returning a UrlTree) or some other action such as a user clicking on a `[routerLink]` directive.

Once a navigation starts, it runs through the following stages:

1. Process redirects
2. URL to path matching
3. Run guards and resolvers
4. Render components and update the browser location

If you'd like to know more, I've written an article explaining each stage.

The Problem

Before #25740, it was possible for multiple navigations to run at once. As you can imagine, this could cause problems.

Consider the following scenario:

- A user clicks on link X. A new navigation with an `id` of 1 starts.
- During the navigation to X, guards and resolvers run asynchronously, and take 10 seconds to complete.

- During those 10 seconds, the user gets fed up, and clicks on link Y, which starts a new navigation to Y, with `id` of 2.
- Navigation 2 must wait on navigation 1's guards and resolvers to finish (the results of which will be ignored).
- Maybe one of the guards in navigation 1 fails and initiates a redirect while navigation 2 is waiting. It's now difficult to tell where the user will land.

Jason Aden discussed this exact problem in his talk at AngularConnect 2018, which I recommend checking out.

In short, managing simultaneous navigations was messy, both inside and outside of the framework. With the changes introduced in 25740, there can be only one active navigation at any one time.

As we'll see, this simplifies matters and makes navigation easier to reason about.

The Changes

The biggest internal changes in 25740 are as follows:

- Each stage of the navigation cycle is now represented by its own operator (redirects, route recognize, etc)
- These operators are run through a `switchMap` which ensures that only the most recent navigation is considered. And, it will cancel and clean up any pending navigations.

The Details

One map can make all the difference

As mentioned previously, the new "one navigation at a time" rule is enforced by the mighty `switchMap` operator. By creating a single observable stream and pipelining all navigations in the router through `switchMap` instead of a `mergeMap` any new navigation will cause the previous navigation to be cancelled and cleaned up by `switchMap`.

> ** Refactor `switchMap` instead of the previous `mergeMap` to ensure new navigations cause a cancellation and clean up of already running navigations*
> *- excerpt from PR 25740*

If you aren’t already aware of how powerful `switchMap` is, or how it works, I recommend checking out Nicholas Jamieson’s article on the topic.

Custom operators

Starting with 25740, different parts of the navigation process have been refactored into custom operators, which live under `/packages/router/src/operators`. You can see that each of the navigation stages mentioned previously are now represented by operators:

- apply_redirects.ts
- recognize.ts
- check_guards.ts and resolve_data.ts
- activate_routes.ts

Internally, transitions between navigations are represented by the `NavigationTransition` type. An observable of `NavigationTransition` called `router.transitions` is used with the main `router.navigations` observable to handle new navigation. This is where `switchMap` is being used to cancel any current navigations whenever a new navigation happens.

```
private setupNavigations(transitions: Observable<Naviga
    return transitions.pipe(
        filter(t => t.id !== 0),
        // Extract URL
        map(t => ({...t, extractedUrl: this.urlHandling
         // Using switchMap so we cancel executing navi
```

You can see the entire pipeline if you check out `setupNavigations` in `router.ts`.

Benefits

With these new changes, any new navigation will automatically cancel and cleanup any pending navigations. This means fewer memory leaks and less time spent waiting for guards and resolvers to finish running for a stale navigation. In larger applications these benefits can really make a difference.

Fortunately, all of these changes are internal. So there isn't really anything you need to change in your applications. Just be aware that you can have only one active navigation at a time.

Summary

With the changes in 25740:

- There is only one active navigation at any time.
- When a new navigation is triggered, any pending navigation is cancelled and cleaned up.
- Internally, each stage of the navigation cycle has been broken up into separate operators.
- These changes should provide faster, more reliable navigation.

Happy navigating!