Lifetime issues with Stream and async traits

help

ar3s3ru August 2, 2020, 10:36pm 1

Hello there!
I'm having some issues with a recent project of mine.

Here's the intro: I got a `Data` object and I want to generate an "infinite" `Stream` for it. This stream is generated by a `Calculator` object (which returns the `impl Stream<Data>`) and applies a `Filter` logic over it before returning the `Stream`.

This `Filter` is reaching out to external sources (e.g. webservices), thus the signature is similar to:

```
fn filter(&self, data: &Data) -> BoxFuture<Result<bool, SomeError>>;
```

I've recreated this example in the Playground: Rust Playground

Two problems I'm having:

1. **Lifetimes:** for some reason, I can't call the `Filter::filter` function from the `Calculator` due to some lifetime mismatches (I guess due to `filter` using `&Data`).
2. **Early stop:** is there a way to bubble-up the `Err` from the `Filter` to the `Calculator` caller? Ideally, I'd like for the whole operation to fail if `Filter` has failed.

Many thanks

drewkett August 3, 2020, 2:32am 2

Is there a reason you are taking `Data` by ref but have `Copy` derived for it? I'm not sure if that's just a consequence of the simplification of the problem, but its a bit odd. The nice thing about `Copy` is that you can pass by value without worrying about ownership, but is really only practical for small structs that don't require ownership.

Here's a version of your problem that compiles. I switched `filter` to `filter_map` which lets you return a stream of `Result<Data,()>`, which is probably what you want to answer your second questions. When processing the iterator, you can just stop the iterator at the first error result. (This is what `.collect()` does for example). I also switched it to pass by value for `filter_map` though which simplified the lifetime situation which sort of side steps question 1, but is what I would do if I have a struct that implements `Copy`. Without a bit of clarity on requirements of Data though, this may not be feasible.

Playground Link

EDIT:
It just dawned on me that `filter(..)` takes the item by reference which is why you had references in there. In which case, I'm assuming the struct is small and derives `Copy`, so I think the playground link is a good way forward.

drewkett August 3, 2020, 4:35am 3

For completeness, here's a version using `filter` that fixes the lifetime issues. The error message wrt lifetimes was related to the `async {}` closure. This version fixes it by avoiding the closure altogether. I can't say I fully understand it, but I believe its something to do with the closure requiring the borrow of `Data` to survive across the `.await` and the compiler being unable to determine the lifetime of the anonymous closure.

Playground Link

system Closed November 1, 2020, 4:35am 4

This topic was automatically closed 90 days after the last reply. We invite you to open a new topic if you have further questions or comments.

Related topics

Topic		Replies	Views	Activity
Passing async function to filter() for futures::stream Stream help	6	7293	August 7, 2020	
Stream filter and reference lifetime help	2	693	September 21, 2021	
Can't infer lifetime due to conflicting requirements when returning a stream help	13	749	August 5, 2020	

Difficulty with trait function lifetimes help	3	363	May 2, 2024
Lifetime error with async closure help	9	1296	August 21, 2020