How to cancel an infinite stream from within the stream itself?

Asked 3 years, 9 months ago
Active 2 years, 1 month ago
Viewed 1k times

6

I'm trying to cancel an interval (`interval_timer`) after emptying a queue but not sure what is the right strategy.

```
let mut some_vars = vec![1, 2, 3, 4, 5, 6, 7, 8];
let interval_timer = tokio_timer::Timer::default();

let timer = interval_timer
    .interval(Duration::from_millis(1000))
    .map_err(|_| {
        println!("Errored out");
    });

let s = timer.for_each(move |_| {
    println!("Woke up");
    let item = some_vars.pop().unwrap();

    let f = futures::future::ok(item).map(|x| {
        println!("{:?}", x);
    });
    tokio::spawn(f)
});

tokio::run(s);
```

I tried `drop` as suggested in gitter but that ended up with an error:

```
let mut some_vars = vec![1, 2, 3, 4, 5, 6, 7, 8];
let mut interval_timer = tokio_timer::Timer::default();

let timer = interval_timer
    .interval(Duration::from_millis(1000))
    .map_err(|_| {
        println!("Errored out");
    });

let s = timer.for_each(move |_| {
    println!("Woke up");
    if some_vars.len() == 1 {
        drop(interval_timer);
    }

    let item = some_vars.pop().unwrap();

    let f = futures::future::ok(item).map(|x| {
        println!("{:?}", x);
    });
    tokio::spawn(f)
});

tokio::run(s);
```

The error:

```
error[E0507]: cannot move out of captured outer variable in an `FnMut` closure
--> src/main.rs:72:22
  |
60 |     let mut interval_timer = tokio_timer::Timer::default();
  |         ----------------- captured outer variable
...
72 |             drop(interval_timer);
  |                  ^^^^^^^^^^^^^^^ cannot move out of captured outer variable in an `FnMut` closure
```

stream rust future rust-tokio

edited Nov 5 '19 at 15:06

Shepmaster
305k • 59 • 824 • 1083

asked Mar 12 '18 at 12:57

3 Answers

Active Oldest Votes

3

For cases where you want to cancel a stream from *outside* of the stream, see stream-cancel.

For your specific case, it's easiest to convert your collection into a stream and zip it together with the interval timer. This way, the resulting stream naturally stops when the collection is empty:

```
use futures::{future, stream, Stream}; // 0.1.29
use std::time::Duration;
use tokio; // 0.1.22
use tokio_timer::Interval; // 0.2.11

fn main() {
  tokio::run({
    let some_vars = vec![1, 2, 3, 4, 5, 6, 7, 8];

    let timer =
      Interval::new_interval(Duration::from_millis(100)).map_err(|e| panic!("Error: {}", e));

    let some_vars = stream::iter_ok(some_vars.into_iter().rev());
    let combined = timer.zip(some_vars);

    combined.for_each(move |(_, item)| {
      eprintln!("Woke up");

      tokio::spawn(future::lazy(move || {
        println!("{:?}", item);
        Ok(())
      }));

      Ok(())
    })
  });
}
```

Otherwise, you can stop the stream by using `and_then` to both remove the value from the collection and control if the stream should continue:

```
use futures::{future, Stream}; // 0.1.29
use std::time::Duration;
use tokio; // 0.1.22
use tokio_timer::Interval; // 0.2.11

fn main() {
  tokio::run({
    let mut some_vars = vec![1, 2, 3, 4, 5, 6, 7, 8];

    let timer =
      Interval::new_interval(Duration::from_millis(100)).map_err(|e| panic!("Error: {}", e));

    let limited = timer.and_then(move |_| {
      if some_vars.len() <= 4 {
        Err(())
      } else {
        some_vars.pop().ok_or(())
      }
    });

    limited.for_each(move |item| {
      eprintln!("Woke up");

      tokio::spawn(future::lazy(move || {
        println!("{:?}", item);
        Ok(())
      }));

      Ok(())
    })
  });
}
```

Share
Improve this answer
Follow
answered Nov 5 '19 at 15:49

Shepmaster
305k 59 824 1083

Add a comment

0

I created a copy of Tokio's `Interval` struct, adding a reference to a method of my application to indicate when to interrupt early.

In my case, I want to interrupt the `Interval` to shutdown.

My Interval poll method looks like this:

```
fn poll(&mut self) -> Poll<Option<Self::Item>, Self::Error> {
    if self.session.read().unwrap().shutdown {
        return Ok(Async::Ready(Some(Instant::now())));
    }

    // Wait for the delay to be done
    let _ = match self.delay.poll() {
```

Then you need to keep a handle on the task (call `task = futures::task::current()` when running inside the timeout task).

At any point you can then call `task.notify()` to kick the interval into action and hit your break out code, interrupting the `Interval` early.

Inside `Interval` there is a `Delay` struct that can be modified, you could create an `Interval` that you can interrupt and change the timeout, this way you could interrupt once and then continue.

Share
Improve this answer
Follow
edited Nov 7 '19 at 11:16

answered Nov 4 '19 at 21:25

teknopaul
5,991 ● 2 ● 27 ● 18

Add a comment

-2

tokio_timer::Interval implements futures::Stream , so try to use the take_while method:

```
let s = timer
    .take_while(|()|
        future::ok(is_net_completed()))
    .for_each(move |_| {
        println!("Woke up");
        // ...
    })
```

Share
Improve this answer
Follow
edited Mar 15 '18 at 5:03

answered Mar 13 '18 at 12:18

wolandr
42 ● 4

Can you explain further how this *cancels* the repeating interval?
– Shepmaster
Mar 13 '18 at 12:34

I did try take_while, the issue I had with that was I couldn't use some_vars in take_while closure and also for_each (mutable) closure. If the ownership could be solved then it solves the immediate problem.
– opensourcegeek
Mar 14 '18 at 12:04

@Shepmaster If you need *pause* the interval stream then it could implemented by using filer method between `take_while` and `for_each` parts. Or exactly in `for_each` closure.
– wolandr
Mar 14 '18 at 13:24

@opensourcegeek So you need to synchronize shard object used for example types from `std::sync::*`
– wolandr
Mar 14 '18 at 13:25

I tried take_while and it does not interrupt the interval, it still fires on the same schedule, if you put a println!() in the take_while closure this is pretty clear
– teknopaul
Nov 4 '19 at 19:05

Add a comment

Your Answer

Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged stream rust future rust-tokio or ask your own question.

The Overflow Blog

- Sequencing your DNA with a USB dongle and open source code
- Don't push that button: Exploring the software that flies SpaceX rockets and...

Featured on Meta

- Providing a JavaScript API for userscripts
- Congratulations to the 59 sites that just left Beta

Related

How do I copy the contents of one stream to another?

795
How do I save a stream to a file in C#?

How do I generate a stream from a string?

What is the best way to decouple a caller of spawn from the spawned procedure?

Passing a closure to a recursive function

How to select between a future and stream in Rust?

How to send bytes::bytes::Bytes by futures:stream::Stream?

Hot Network Questions

Circuit analysis homework - LTspice results are different from calculations

Can a Pyromancer Sorcerer deal fire damage to a creature under the effect of Invulnerability?

How to install a package via `apt-get` without flagging it as manually installed

Why is 20m band waterfall showing signals every 50 kHz?

Help identify a short story about professor using voodoo doll to prevent the marriage of a much younger woman he loves - by Henry Slesar

more hot questions

Question feed