How to return a reference in Rust Closure

Asked 1 year, 9 months ago

Active 1 year, 9 months ago

Viewed 309 times

3

I have the following rust code cannot be compiled.

```
struct Person {
    name : String,
    age : u8,
}


fn main() {
    let p = Person{ name: "Nobody".to_string(), age : 24};

    let age = |p : &Person| p.age;
    let name = |p : &Person | &p.name;

    println! ("name={}, age={}" , name(&p), age(&p));
}
```

And the compiler gave the following error message.

```
   Compiling playground v0.0.1 (/playground)
error[E0495]: cannot infer an appropriate lifetime for borrow expression due to conflicting requirements
  --> src/main.rs:11:31
   |
11 |     let name = |p : &Person | &p.name;
   |                               ^^^^^^^
   |
note: first, the lifetime cannot outlive the anonymous lifetime #1 defined on the body at 11:16...
  --> src/main.rs:11:16
   |
11 |     let name = |p : &Person | &p.name;
   |                ^^^^^^^^^^^^^^^^^^^^^^
note: ...so that reference does not outlive borrowed content
  --> src/main.rs:11:31
   |
11 |     let name = |p : &Person | &p.name;
   |                               ^^^^^^^
note: but, the lifetime must be valid for the expression at 2:29...
  --> src/main.rs:13:5
   |
13 |     println! ("name={}, age={}" , name(&p), age(&p));
   |     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: ...so type `(&&std::string::String, &u8)` of expression is valid during the expression
  --> src/main.rs:13:5
   |
13 |     println! ("name={}, age={}" , name(&p), age(&p));
   |     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   = note: this error originates in a macro outside of the current crate (in Nightly builds, run with -Z external-macro-backtrace for more info)

error: aborting due to previous error
```

I tried to add the lifetime for `name` closure.

```
let name<'a> = |p : &'a Person | -> &'a String { &'a p.name };
```

but still got the compiler error

```
   Compiling playground v0.0.1 (/playground)
error: expected one of `:`, `;`, `=`, `@`, or `|`, found `<`
  --> src/main.rs:12:13
   |
12 |     let name<'a> = |p : &'a Person | -> &'a String { &'a p.name };
   |             ^ expected one of `:`, `;`, `=`, `@`, or `|`

error: aborting due to previous error
```

Just want to know how to write the correct code.

rust

Share

Improve this question

Follow

asked Mar 28 '20 at 6:03

haoel

33 ● 4

Does this answer your question? github.com/rust-lang/rust/issues/22340
– Đorđe Željić
Mar 28 '20 at 9:09

Add a comment

2 Answers

Active Oldest Votes

2

One other solution is to give an explicit type to your closure. Unfortunately, you can't use its actual type, but you can cast it to a function pointer.

Remember that the issue is that the compiler isn't able to correctly deduce that the lifetime of the output is tied to the lifetime of the input (it could be an instance of this bug, but I'm not at all sure). We can fix that by making the lifetimes explicit.

```
struct Person {
    name: String,
    age: u8,
}

fn main() {
    let p = Person {
        name: "Nobody".to_string(),
        age: 24,
    };

    let age = |p: &Person| p.age;
    // Our only changes are right here.
    let name: for<'a> fn(&'a Person) -> &'a String = |p: &Person| &p.name;

    println!("name={}, age={}", name(&p), age(&p));
}
```

(playground)

In fact, it's possible to be slightly less explicit than this. The compiler is fine figuring out the types of the input and output. It's just the lifetimes it has trouble with. So replacing that line with `let name: for<'a> fn(&'a _) -> &'a _ = |p: &Person| &p.name;` also works (playground).

Share
Improve this answer
Follow
answered Mar 28 '20 at 20:23
SCappella
8,311 1 17 29

Add a comment

2

For me it is more important to understand the source of the problem that find a workaround, so step by step. let's start from something that works:

```
struct Person {
    name: String,
    age: u8,
}

fn get_name<'a>(person: &'a Person) -> &'a str {
    &person.name
}

fn main() {
    let p = Person {
        name: "Nobody".to_string(),
        age: 24,
    };

    let age = |p: &Person| p.age;
    let name = get_name;

    println!("name={}, age={}", name(&p), age(&p));
}
```

There is no issues when using a function instead of a clousure. In this case, the compiler is able to check that lifetime requirements are ok.

But when trying to use a closure for `name` :

```
let name = |p : &Person | &p.name;
```

You get the `cannot infer an appropriate lifetime` error.

Why?

A closure captures its environment: some opaque struct has to be created by the compiler and such struct has to be callable.

I'm non fully aware of the internal details, but something along these lines would be created when desugaring your closure:

```
struct OpaqueType<'a> {
    // a PhantomData because you don't capure nothing
    // just to make explicit that struct lifetime bind to environment
    // if you would had captured some integer:
    // captured_int: &'a i32,
    captured_int: PhantomData<&'a i32>,

}

impl<'a> OpaqueType<'a> {
    fn call<'b>(&'b self, person: &'a Person) -> &'a str {
        &person.name
    }
}
```

And looking at `call` it is apparent that when a clusure argument is a reference there are two unrelated lifetimes at play.

Finally the answer: how to return a reference

Also note that in your case, not declaring the argument type and using the helper function `get_name`, works:

```
// let name = |p| &p.name; // does not work, not enough info to infer p type
let name = |p| get_name(p);
```

My guess is that in such case the compiler, following some inference path, is able to desugar in a way that lifetimes are bounded as expected.

Share
Improve this answer
Follow
answered Mar 28 '20 at 11:29
attdona
13.7k 6 38 52

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 rust 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

Visit chat

Related

How to include and use Rust libraries

Why doesn't println! work in Rust unit tests?

Unable to coerce &String to &str

Allowing reference lifetime to outlive a closure

"the type does not fulfill the required lifetime" when using a method in a thread

0

how can I implement an external trait on an internal trait in Rust?

How to impl add for a generic in Rust

why rustc compile complain my simple code "the trait std::io::Read is not implemented for Result<File, anyhow::Error>"

can't use implemented trait in other file rust

Hot Network Questions

On what basis do countries repay international loans?

Schrödinger's cat program

Do all observations arise from probability distributions?

What caused this crash landing?

Why did the JWST solar array deploy early?

more hot questions

Question feed