[The Rust Programming Language Forum](https://users.rust-lang.org/)

Struct having a reference to a value in one of its variables

help

frostfortune February 25, 2019, 4:49pm 1

I am making a struct which is basically just a vector and a reference to one of the values in the vector.

this is an example to show the problem

rust_error778×726 25.1 KB

if i remove the lifetime specifiers and the reference to the selected_item all together that error doesnt even show.
can anyone explain what is going on here or help me understand why i cant do this?

RustyYato February 25, 2019, 4:57pm 2

You have managed to create a self-referential list.
Here's what happened.

```
// You created a list, this list's lifetime parameter ('a) is currently unbound
let mut list = List::new();

// You add an item to the list, and set selected_item
// This binds the lifetime 'a to the lifetime of list
// Note: this means that because you are mutable borrowing here
// Rust will think that you are mutable borrowing for the entirety of 'a
// Now because 'a == lifetime of list, you are now mutable borrowing
// for the rest of list's lifetime
list.add_item(1);

// You try to get an immutable borrow of list, which conflicts with the
// mutable borrow from earlier
list.print_selected();
```

Note:
even though I am talking about (im)mutable references, it will be much more useful to think of `&mut T` and &Tas compile time locks (similar idea to thread locks). Where `&mut T` means unique lock, and &T means shared lock.

You can see more about this in this blog post

Blog post: Rust, a unique perspective tutorials

> Rust: A unique perspective is my attempt to explain Rust's memory safety guarantees in terms of unique versus shared access to memory. These aren't new ideas, but I couldn't find much existing documentation that introduces them to beginning and intermediate Rust programmers, so I wrote my own.

3 Likes

frostfortune February 25, 2019, 5:02pm 3

Thank you for the explanation. I will be sure to check that post out.

1 Like

singron February 26, 2019, 4:54am 4

Here is a rust playground link for your original example.

The previous reply is completely correct. I'd just like to add that it's specifically very difficult to use structs where the fields borrow from each other (e.g. the `selected_item` borrowing from `items`). I've made a modified playground to the above where I replaced `selected_item: Option<&'a mut i32>` with `selected_item_idx: Option<usize>`, which works more easily in this case.

nour February 27, 2019, 10:47am 5

frostfortune:

> I am making a struct which is basically just a vector and a reference to one of the values in the vector.

Hi,
The guys have already pointed the problem, so I will add a very very small note : in such positions, you can use an "index" not "a reference", i.e. you keep the position of the selected element in the list not the reference to it and when no element is selected you can keep an unrealistic value (such as -1) than you can change this integer index freely because it become independant from the vector (from Rust point of view).

baumanj March 3, 2019, 5:56pm 6

The downside of an index is since it’s completely independent from rustc’s point of view, it can become totally invalid and it’s your code’s responsibility to check that.

1 Like

Related topics

Topic	Replies	Views	Activity
How to run a method that requires a mutable borrow of self and pass it non-mutable references of self fields help	7	1192	August 15, 2020
How to create a struct with other field's reference	5	442	July 9, 2024
How to handle a reference in a struct to another member in the struct help	5	4497	May 8, 2022
Beginner question on lifetimes help	7	459	January 26, 2023
Lifetime Mismatches in Rust help	8	720	November 7, 2022