Skip to main content

Stack Overflow

1. About
2. Products
3. For Teams

1. Stack Internal Implement a knowledge platform layer to power your enterprise and AI tools.
2. Stack Data Licensing Get access to top-class technical expertise with trusted & attributed content.
3. Stack Ads Connect your brand to the world’s most trusted technologist communities.
4. Releases Keep up-to-date on features we add to Stack Overflow and Stack Internal.
5. About the company Visit the blog

Loading…

current community

Stack Overflow

help chat

- Meta Stack Overflow

your communities

Sign up or log in to customize your list.

more stack exchange communities

company blog

3. Log in
4. Sign up

AI Assist is now on Stack Overflow. Start a chat to get instant answers from across the network. Sign up to save and share your chats.

Home

Questions

AI Assist

Tags

Challenges

7. Chat

Articles

Users

Companies

Collectives

Communities for your favorite technologies. Explore all Collectives

Returning reference to trait rather than struct

Ask Question

Asked 10 years ago

Modified 10 years ago

Viewed 218 times

2

I'm running into an issue trying to return a reference to a trait in Rust. I can return a reference to a struct OK.

This is a contrived example to reproduce the problem, but I have a `Person` struct defined as:

```
struct PersonStruct<'a> {
   age: u32,
   photo: Option<PhotoStruct<'a>>
}
```

I have a `Person` trait too, and I am trying to write a method on this trait to return a `Photo` trait, rather than a `PhotoStruct`.

The full source is below, but the issue is that this method works fine:

```
    fn get_photo_struct(&'a self) -> Option<&'a PhotoStruct<'a>> {
       let photo_ref = self.photo.as_ref();
       photo_ref
   }
```

... and this one won't compile ...

```
    fn get_photo_trait(&'a self) -> Option<&'a Photo<'a>> {
       let photo_ref = self.photo.as_ref();
       photo_ref
   }
```

The difference is, the first returns the struct and the second returns the trait. There is an implementation of the trait for the struct.

The compilation error is:

```
src/lib.rs:40:9: 40:18 error: mismatched types:
expected `core::option::Option<&'a Photo<'a> + 'a>`,
   found `core::option::Option<&PhotoStruct<'_>>`
(expected trait Photo,
   found struct `PhotoStruct`) [E0308]
```

I'd appreciate any pointers on what I'm doing wrong or what approach I should be taking instead.

Full source:

```
use std::any::Any;
use std::vec::Vec;

struct PhotoStruct<'a> {
   photo_bytes: &'a [u8]
}

struct PersonStruct<'a> {
   age: u32,
   photo: Option<PhotoStruct<'a>>
}

trait Photo<'a> {
   fn get_bytes(&self) -> &'a[u8];
}

impl<'a> Photo<'a> for PhotoStruct<'a> {
   fn get_bytes(&self) -> &'a[u8] {
       return self.photo_bytes;
   }
}

trait Person<'a> {
   fn get_age(&self) -> u32;
   fn get_photo_struct(&'a self) -> Option<&'a PhotoStruct<'a>>;
   fn get_photo_trait(&'a self) -> Option<&'a Photo<'a>>;
}

impl<'a> Person<'a> for PersonStruct<'a> {
   fn get_age(&self) -> u32 {
       self.age
   }
   fn get_photo_struct(&'a self) -> Option<&'a PhotoStruct<'a>> {
       let photo_ref = self.photo.as_ref();
       photo_ref
   }
   fn get_photo_trait(&'a self) -> Option<&'a Photo<'a>> {
       let photo_ref = self.photo.as_ref();
       photo_ref
   }
}
```

- rust

Share

Improve this question

Follow

edited Nov 12, 2015 at 2:31

Shepmaster

439k116116 gold badges1.3k1.3k silver badges1.5k1.5k bronze badges

asked Nov 12, 2015 at 2:23

andygrove

11811 silver badge77 bronze badges

2

2

On side note: You usually shouldn't insert `Struct` into your type names... Just call it `Person` or `Photo`. Just a small hint on better style...

Lukas Kalbertodt – Lukas Kalbertodt

2015-11-12 08:20:30 +00:00

Commented Nov 12, 2015 at 8:20

2

If we are commenting on style, I'd recommend dropping the `get_` prefixes and the redundant `let` statements.

Shepmaster – Shepmaster

2015-11-12 13:33:03 +00:00

Commented Nov 12, 2015 at 13:33

Add a comment |

1 Answer

Sorted by: Reset to default

Highest score (default) Trending (recent votes count more) Date modified (newest first) Date created (oldest first)

2

You can cast it to a `&Photo`:

```
fn get_photo_trait(&'a self) -> Option<&'a Photo<'a>> {
   let photo_ref = self.photo.as_ref();
   photo_ref.map(|photo| photo as &Photo)
}
```

Share

Improve this answer

Follow

answered Nov 12, 2015 at 2:37

antoyo

12k99 gold badges5454 silver badges9090 bronze badges

Sign up to request clarification or add additional context in comments.

Comments

Add a comment

Your Answer

Thanks for contributing an answer to Stack Overflow!

- Please be sure to *answer the question*. Provide details and share your research!

But *avoid* …

- Asking for help, clarification, or responding to other answers.
- Making statements based on opinion; back them up with references or personal experience.

To learn more, see our tips on writing great answers.

Sign up or log in

Sign up using Google

Sign up using Email and Password

Submit

Post as a guest

Name

Email

Required, but never shown

Post as a guest

Name

Email

Required, but never shown

Post Your Answer Discard

By clicking “Post Your Answer”, you agree to our terms of service and acknowledge you have read our privacy policy.

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

- rust

See similar questions with these tags.

- The Overflow Blog
  Introducing Stack Overflow AI Assist—a tool for the modern developer
  Treating your agents like microservices

- Featured on Meta
  Chat room owners can now establish room guidelines
  AI Assist is now available on Stack Overflow
  Policy: Generative AI (e.g., ChatGPT) is banned

Related

4
How to have a reference to a trait and still use original struct?
5
How to use a trait object to refer to struct that has generic methods
49
How do I return an instance of a trait from a method?
0
How to return an instance of a trait?
2
How to write a trait that has a method returning a reference and implement it correctly?
1
Using trait methods from a structure implementation
4
How can a Rust trait object return another trait object?
1
Rust: Trait as a return type
1
Returning an object that implements trait from a method
2
Rust: returning trait object in default method implementation

Hot Network Questions

- A hilarious parody of Western movies where an entire town collapses into tunnels
- PSE Advent Calendar 2025 (Day 6): Reindeer with presents
- Is there a term for forming emotional or relational bonds with AI chatbots? If not, is “cyberpomorphic” a valid neologism?
- How to protect 100-year-old wooden floors for the next 100 years?
- Unity HDRP custom pass stencil buffer not working in build mode
- Why are left and right cosets different in the non-normal case?
- is it possible to keep the space when using url in latex
- Is "We will review application starting from..." a deadline?
- Why is "cmd | xargs" to join lines of cmd with spaces wrong
- I2C bus problems when using two distance sensors in Arduino
- King defense setups
- Vertical alignment of equations inside tabularx
- PSE Advent Calendar 2025 (Day 5): Cinnamon overdose
- The fourth moment of the Riemann zeta function without absolute values
- What's the point of swapping polarity between mod slots?
- Would it be plausible to have a flat eye with a moving pupil?
- Explain this seeming contradiction in Euclid Book 1 Proposition 16
- How to reduce GAM sensitivity to sparse data
- Best Welding Method to Fix a Damaged Steel Surly Frame
- Displaying features in a layer of geometrytype = GeometryCollection25D
- Do something and "win"
- Why does black allow white to have a protected passed central pawn here?
- TV program where real actors interact with 2-Dimensional characters
- Could a world tree have different biomes?

more hot questions

Question feed

Subscribe to RSS

Question feed