RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

Bifrost

Staking Derivatives

A parachain designed for staking liquidity

web3.0 bootcamp

project supported by web3 foundation grants program

substrate builders program

https://bifrost.finance

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

如何使用过程宏简化代码

whoami

```
#[derive(WhoAmI)]
struct WhoAmI {
  #[quote = " 没人关心 "]
  name: "Jianping Deng( 邓建平 )",
  #[quote = " 没人关心 "]
  work_for: "Bifrost",
  #[quote = " 没人关心 "]
  job: "Rust Engineer",
  #[quote = " 叔叔，我们不约 "]
  make_friend: "Dengjianping(github)",
}
```

Agenda

- 什么是过程宏
- 为什么要用过程宏
- 例子
- Takeaways

什么是过程宏

From The Rust Reference:
Procedural macros allow creating syntax extensions as execution of a function.

过程宏分类

- Function-like macros: my_func_like!()

- Derive macros: #[derive(MyDerive)]

- Attribute macros: #[my_attri_macro]

必备的第三方库

- syn，解析 AST 里的 tokens
- quote，重构 AST
- proc-macro/2，支持过程宏的编写

基本编写过程

- 解析 AST
- 处理 Tokens
- 重新生成 AST

为什么要用过程宏

- 复用代码
- 增强代码的表达力
- 更深入学习 rust

例子

```
pub async fn get_acount(url: &str, account_name: AccountName)
  -> Result<GetAccount, Box<dyn std::error::Error>>
{
  let response = reqwest::Client::new()
        .post(&url)
        .json(&account_name)
        .send()
        .await?;

  if response.status() == reqwest::StatusCode::OK {
     response.json().await.map_err(|_| crate::Error::ParseJsonError)?;
  } else {
     Err(crate::Error::HttpRequestError)
  }
}
```

例子

```
pub async fn get_action…
pub async fn get_abi…
pub async fn push_action…
// …
```

例子

```rust
#[derive(Debug, Clone, Serialize)]
pub struct GetAccountParams {
  pub url: String,
  pub account_name: AccountName,
}
```

例子

```
#[derive(Debug, Clone, Serialize, RPC)]
#[rpc(api="v1/chain/get_account", http_method="POST", returns="GetAccount")]
pub struct GetAccountParams {
  pub url: String,
  pub account_name: AccountName,
}
```

例子

```
#[proc_macro_derive(RPC, attributes(rpc))]
pub fn derive_rpc(item: TokenStream) -> TokenStream {
  let derive_input = parse_macro_input!(item as DeriveInput);
  // ...
  todo!()
}
```

例子

```
// 提取属性
let mut api = String::new();
let mut http_method = String::new();
let mut returns = String::new();
derive_input.attrs.iter().for_each(|attr| {
  // 一些解析步骤
  api = get_api_address(attr); // 获取 api 地址
  http_method = get_method_type(attr); // 获取 method 类型
  returns = get_return_type(attr); // 获取返回类型
});
```

例子

```
// 重新构造 AST
let expanded = quote! {
  impl #impl_generics #struct_name #ty_generics #where_clause {
    if http_method == "post" {
      pub async fn post(&self )
        -> Result<#returns, Box<dyn std::error::Error>>
      { /* reqwest 实现 post 请求 */ }
    }
    if http_method == "get" {
      pub async fn get(&self )
        -> Result<#returns, Box<dyn std::error::Error>>
      { /* reqwest 实现 get 请求 */ }
    }
  }
};
```

例子

```
#[derive(RPC)]
#[rpc(api="v1/chain/get_abi", http_method="GET", returns="GetAbi")]
pub struct GetAbiParams {
  pub url: String,
  pub account_name: AccountName,
}

#[derive(RPC)]
#[rpc(api="v1/chain/get_block", http_method="GET", returns="GetBlock")]
struct GetBlockParams {
  pub url: String,
  pub block_num_or_id: String,
}
```

例子 2

```
// actix-web
use identity: actix_identity::Identity;
pub(crate) async fn my_view(
  db: web::Data<PgPool>,
  identity: Identity
) -> Result<HttpResponse, crate::ErrorKind> {
  // ...
  if let Some(_) = identity.identity() {
     // do something
  } else {
     Err(crate::ErrorKind::IdentityExpired)
  }
}
```

例子 2

```
# Python, Django
@login_required
def my_view(request):
    # do something
```

例子 2

```
#[login_required]
pub(crate) async fn my_view(
  db: web::Data<PgPool>,
  identity: Identity
) -> Result<HttpResponse, crate::ErrorKind> {
  // do something
}
```

例子 2

```
#[proc_macro_attribute]
pub fn login_required(_: TokenStream, func: TokenStream) -> TokenStream {
  let func = parse_macro_input!(func as ItemFn);


  // ...
  todo!()
}
```

例子 2

```
// 获取参数，并提取参数， identity: Identity
let (identity_param, identity_type) = func.sig.inputs.iter().filter_map(|i| {
  match i {
     // pat_type => identity: Identity
     FnArg::Typed(ref pat_type) => {
        // 获取 identity 的 pattern
     }
     // ...
  }
}).collect()[0];
```

例子 2

```
let caller = quote!{
  #func_vis #asyncness fn #func_name #func_generics(#func_inputs) #func_output {
    fn is_expired(#identity_param: &#identity_type) -> bool {
      #identity_param.identity().is_some()
    }
    if !is_expired(&#identity_param) {
      Err(crate::ErrorKind::IdentityExpiredError)
    } else {
      #func_block
    }
  }
};
```

例子 2

```
#[login_required]
pub(crate) async fn my_view(
  db: web::Data<PgPool>,
  identity: Identity
) -> Result<HttpResponse, crate::ErrorKind> {
  // do something
}
```

更多例子

```
#[derive(ReadBytes, WriteBytes, Digest, NumBytes, SerializeData)]
struct MyFoo{ … }


#[timeit]
fn foo() { … }


#[timeout(secs = 10)]
fn bar() { … }
```

Takeaways

- 审视代码复用部分。
- 做好宏的设计，尽量简介优雅。可以参考 serde, rocket 等等的设计。
- 文档和测试。
- 注意宏的卫生。
- 调试工具， cargo-expand, dbg!, println! 。
- 关注大神 @dtolnay ，多看官方文档，社区交流。
- more

Thanks!

Enjoy Rust!

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳