

第三课: 如何优雅的处理错误

`unwrap()`、`match()`、`Err`、`From`、?

苏林



回顾过去3次公开课的内容

1、第一课: Rust入门基本原理

2、第二课: 类型系统

今天公开课内容

- 1、聊一聊unwrap
- 2、rust如何处理错误
- 3、rust如何自定义错误
- 4、match多嵌套的问题
- 5、如何优雅的处理错误

使用unwrap

```
1 ▶ fn main() {  
2     let path: &str = "/sulin/test";  
3     println!("{}", read_file(path));  
4 }  
5  
6 fn read_file(path: &str) -> String {  
7     std::fs::read_to_string(path).unwrap()  
8 }
```

实际项目开发中, 到底是否应该避免unwrap()的出现

思考???

Rust错误处理示例

```
1  ▶ fn main() {  
2      let path: &str = "/sulin/test";  
3      match read_file(path) {  
4          Ok(file: String) => {  
5              println!("{}", file)  
6          }  
7          Err(e: Error) => {  
8              println!("{}", {}, path, e)  
9          }  
10     }  
11 }  
12  
13 fn read_file(path: &str) -> Result<String, std::io::Error> {  
14     std::fs::read_to_string(path)  
15 }
```

Rust错误处理示例

思考, 刚才的处理是否存在问题?

Rust如何自定义错误

Trait `std::error::Error`

```
pub trait Error: Debug + Display {  
    fn source(&self) -> Option<&(dyn Error + 'static)> { ... }  
  
    fn backtrace(&self) -> Option<&Backtrace> { ... }  
  
    fn description(&self) -> &str { ... }  
  
    fn cause(&self) -> Option<&dyn Error> { ... }  
}
```


Rust如何自定义错误

总结一下, 自定义一个error需要实现如下几步:

实现impl std::fmt::Display的trait,并实现fmt(...)方法。

实现impl std::fmt::Debug的trait, 一般直接添加注解即可: #[derive(Debug)]

实现impl std::error::Error的trait,并根据自身error级别是否覆盖std::error::Error中的source()方法。

match多嵌套的问题

```
1  fn read_file(path: &str) -> Result<String, std::io::Error> {  
2      std::fs::read_to_string(path)  
3  }  
4  
5  fn to_utf8(v: &[u8]) -> Result<&str, std::str::Utf8Error> {  
6      std::str::from_utf8(v)  
7  }  
8  
9  fn to_u32(v: &str) -> Result<u32, std::num::ParseIntError> {  
10     v.parse::<u32>()  
11 }  
12  
13 fn main() {  
14  
15 }
```

match多嵌套的问题

```
1  fn read_file(path: &str) -> Result<String, std::io::Error> {
2      std::fs::read_to_string(path)
3  }
4
5  fn to_utf8(v: &[u8]) -> Result<&str, std::str::Utf8Error> {
6      std::str::from_utf8(v)
7  }
8
9  fn to_u32(v: &str) -> Result<u32, std::num::ParseIntError> {
10     v.parse::<u32>()
11 }
12
13 fn main() {
14
15 }
```

QA环节

加群一起交流Rust & Databend

