7结构体字段问题

2020年3月9日 12:14

1.结构体字段转移所有权
结构体的部分字段是可以移动所有权的,剩余的字段是可以访问的,结构体的整体不能访问.

2.使用结构体更新语法从其他实力创建实例。
```
let user2 = User {
    email: String::from("another@example.com"),
    username: String::from("anotherusername567"),
    ..user1
};
```
..语法表示了剩余未显示设置值的字段应有与给定实例对应字段相同的值。

细节注意:

```
#[derive(Debug)]
struct User{
    username: String,
    email: String,
    sign_in_count: u64,
    active: bool,
}

let user1 = User{
    username: String::from("hades"),
    email: String::from("caojinyu@msn.com"),
    sign_in_count: 10,
    active: false,
};

let user2 = User {
    username: String::from("tom"),
    ..user1   // user1的部分元素的所有权会发生转移
};
```

因为user1的email所有权发生了转移，所以整体的user1不能被使用。
部分使用的话，user1.email是不能使用的，但是user1.username可以使用，没有发生所有权转移，user1.sign_in_count，user1.active可以使用的，因为实现了Copy。
```
println!("{}", user1.email); //error
println!("{}", user1.username); // right
println!("{}", user1.sign_in_count); // ok
println!("{}", user1.active); // ok
```

如果我们想要继续整体的使用user1的话，要实现Clone。
```
#[derive(Debug, Clone)]
struct User{
    username: String,
    email: String,
    sign_in_count: u64,
    active: bool,
```

```
}

let user1 = User{
    username: String::from("hades"),
    email: String::from("caojinyu@msn.com"),
    sign_in_count: 10,
    active: false,
};

let user2 = User {
    username: String::from("tom"),
    ..user1.clone()   // user1临时克隆了一份，原来的user1的全部依然可以使用的。
};
```

3.元组结构体

元组结构体有着结构体名称提供的含义，但是没有具体的字段名称，只有字段的类型。

适用于:
当我们想给整个元组起一个名字，并使元组称为与其他元组不用的类型时，元组结构体是非常有用的。

```
struct Type(i32, String, f64);
```