Rust 编程语言入门

杨旭，微软MVP
Rust、Go、C#开发者

5. struct

5.1 定义并实例化 struct

什么是 struct

- struct，结构体
  - 自定义的数据类型
  - 为相关联的值命名，打包 => 有意义的组合

定义 struct

- 使用 **struct** 关键字，并为整个 struct 命名
- 在花括号内，为所有**字段（Field）**定义**名称**和**类型**
- 例如：
- ```
  struct User {
      username: String,
      email: String,
      sign_in_count: u64,
      active: bool,
  }
  ```
  ```

实例化 struct

- 想要使用 struct，需要创建 struct 的实例：
  - 为每个字段指定具体值
  - 无需按声明的顺序进行指定
- 例子：
-
```
let user1 = User {
    email: String::from("someone@example.com"),
    username: String::from("someusername123"),
    active: true,
    sign_in_count: 1,
};
```

取得 struct 里面的某个值

- 使用点标记法：

- ```
  let mut user1 = User {
      email: String::from("someone@example.com"),
      username: String::from("someusername123"),
      active: true,
      sign_in_count: 1,
  };

  user1.email = String::from("anotheremail@example.com");
  ```
  ```

注意

- 一旦 struct 的实例是可变的，那么实例中所有的字段都是可变的

struct 作为函数的返回值

- 例子：

- ```
  fn build_user(email: String, username: String) -> User {
      User {
          email: email,
          username: username,
          active: true,
          sign_in_count: 1,
      }
  }
  ```
  ```

字段初始化简写

- 当字段名与字段值对应变量名相同时，就可以使用字段初始化简写的方式：

```
fn build_user(email: String, username: String) -> User {
    User {
        email,
        username,
        active: true,
        sign_in_count: 1,
    }
}
```

struct 更新语法

- 当你想基于某个 struct 实例来创建一个新实例的时候，可以使用 struct 更新语法：

- ```
  let user2 = User {
      email: String::from("another@example.com"),
      username: String::from("anotherusername567"),
      active: user1.active,
      sign_in_count: user1.sign_in_count,
  };
  ```

- ```
  let user2 = User {
      email: String::from("another@example.com"),
      username: String::from("anotherusername567"),
      ..user1
  };
  ```

Tuple struct

- 可定义类似 tuple 的 struct，叫做 tuple struct
  - Tuple struct 整体有个名，但里面的元素没有名
  - 适用：想给整个 tuple 起名，并让它不同于其它 tuple，而且又不需要给每个元素起名
- 定义 tuple struct：使用 struct 关键字，后边是名字，以及里面元素的类型
- 例子：
- ```
  struct Color(i32, i32, i32);
  struct Point(i32, i32, i32);
  let black = Color(0, 0, 0);
  let origin = Point(0, 0, 0);
  ```
- black 和 origin 是不同的类型，是不同 tuple struct 的实例。
  ```

Unit-Like Struct （没有任何字段）

- 可以定义没有任何字段的 struct，叫做 Unit-Like struct（因为与()，单元类型类似）
- 适用于需要在某个类型上实现某个 trait，但是在里面又没有想要存储的数据

struct 数据的所有权

- struct User {
  username: String,
  email: String,
  sign_in_count: u64,
  active: bool,
  }
- 这里的字段使用了 String 而不是 &str：
  - 该 struct 实例拥有其所有的数据
  - 只要 struct 实例是有效的，那么里面的字段数据也是有效的
- struct 里也可以存放引用，但这需要使用生命周期（以后讲）。
  - 生命周期保证只要 struct 实例是有效的，那么里面的引用也是有效的。
  - 如果 struct 里面存储引用，而不使用生命周期，就会报错（例子）。

再见