Rust 编程语言入门

杨旭，微软MVP
Rust、Go、C#开发者

10.3 Trait（上）

Trait

- Trait 告诉 Rust 编译器：
  - 某种类型具有哪些并且可以与其它类型共享的功能
- Trait：抽象的定义共享行为
- Trait bounds（约束）：泛型类型参数指定为实现了特定行为的类型
- Trait 与其它语言的接口（interface）类似，但有些区别。

定义一个 Trait

- Trait 的定义：把方法签名放在一起，来定义实现某种目的所必需的一组行为。
  - 关键字：trait
  - 只有方法签名，没有具体实现
  - trait 可以有多个方法：每个方法签名占一行，以；结尾
  - 实现该 trait 的类型必须提供具体的方法实现
- （例子）

在类型上实现 trait

- 与为类型实现方法类似。
- 不同之处：
  - impl **Xxxx for** Tweet { … }
  - 在 impl 的块里，需要对 Trait 里的方法签名进行具体的实现
- （例子）

实现 trait 的约束

- 可以在某个类型上实现某个 trait 的前提条件是：
  - 这个类型 或 这个 trait 是在本地 crate 里定义的
- 无法为外部类型来实现外部的 trait：
  - 这个限制是程序属性的一部分（也就是**一致性**）。
  - 更具体地说是**孤儿规则**：之所以这样命名是因为父类型不存在。
  - 此规则确保其他人的代码不能破坏您的代码，反之亦然。
  - 如果没有这个规则，两个 crate 可以为同一类型实现同一个 trait，Rust 就不知道应该使用哪个实现了。

默认实现

- （例子）
- 默认实现的方法可以调用 trait 中其它的方法，即使这些方法没有默认实现。
- （例子）
- 注意：无法从方法的重写实现里面调用默认的实现。

再见