Rust 编程语言入门

杨旭，微软 MVP
Rust、Go 开发者

19.2 高级 Trait

在 Trait 定义中使用关联类型来指定占位类型

- 关联类型（associated type）是 Trait 中的类型占位符，它可以用于 Trait 的方法签名中：
  - 可以定义出包含某些类型的 Trait，而在实现前无需知道这些类型是什么
- （例子）

关联类型与泛型的区别

泛型

- 每次实现 Trait 时标注类型
- 可以为一个类型多次实现某个 Trait（不同的泛型参数）

关联类型

- 无需标注类型
- 无法为单个类型多次实现某个 Trait

默认泛型参数和运算符重载

- 可以在使用泛型参数时为泛型指定一个默认的具体类型。
- 语法：<PlaceholderType=ConcreteType>
- 这种技术常用于运算符重载（operator overloading）
- Rust 不允许创建自己的运算符及重载任意的运算符
- 但可以通过实现 std::ops 中列出的那些 trait 来重载一部分相应的 运算符
- （例子）

默认泛型参数的主要应用场景

- 扩展一个类型而不破坏现有代码
- 允许在大部分用户都不需要的特定场景下进行自定义

完全限定语法（Fully Qualified Syntax）如何调用同名方法

- （例子）
- 完全限定语法：<Type as Trait>::function(receiver_if_method, netx_arg, ...);
  - 可以在任何调用函数或方法的地方使用
  - 允许忽略那些从其它上下文能推导出来的部分
  - 当 Rust 无法区分你期望调用哪个具体实现的时候，才需使用这种语法

使用 supertrait 来要求 trait 附带其它 trait 的功能

- 需要在一个 trait 中使用其它 trait 的功能：
  - 需要被依赖的 trait 也被实现
  - 那个被间接依赖的 trait 就是当前 trait 的 supertrait
- （例子）

使用 newtype 模式在外部类型上实现外部 trait

- 孤儿规则：只有当 trait 或类型定义在本地包时，才能为该类型实现这个 trait
- 可以通过 newtype 模式来绕过这一规则
  - 利用 tuple struct（元组结构体）创建一个新的类型
- （例子）

再见