Rust 编程语言入门

杨旭，微软 MVP
Rust、Go 开发者

15 智能指针

相关的概念

- 指针：一个变量在内存中包含的是一个地址（指向其它数据）
- Rust 中最常见的指针就是“引用”
- 引用：
  - 使用 &
  - 借用它指向的值
  - 没有其余开销
  - 最常见的指针类型

智能指针

- 智能指针是这样一些数据结构：
  - 行为和指针相似
  - 有额外的元数据和功能

引用计数（reference counting）智能指针类型

- 通过记录所有者的数量，使一份数据被多个所有者同时持有
- 并在没有任何所有者时自动清理数据

引用和智能指针的其它不同

- 引用：　　只借用数据
- 智能指针：　　很多时候都拥有它所指向的数据

智能指针的例子

- String 和 Vec<T>
- 都拥有一片内存区域，且允许用户对其操作
- 还拥有元数据（例如容量等）
- 提供额外的功能或保障（String 保障其数据是合法的 UTF-8 编码）

智能指针的实现

- 智能指针通常使用 struct 实现，并且实现了：
  - Deref 和 Drop 这两个 trait
- Deref trait：允许智能指针 struct 的实例像引用一样使用
- Drop trait：允许你自定义当智能指针实例走出作用域时的代码

本章内容

- 介绍标准库中常见的智能指针
  - Box<T>：在 heap 内存上分配值
  - Rc<T>：启用多重所有权的引用计数类型
  - Ref<T> 和 RefMut<T>，通过 RefCell<T> 访问：在运行时而不是编译时强制借用规则的类型

- 此外：
  - 内部可变模式（*interior mutability* pattern）：不可变类型暴露出可修改其内部值的 API
  - 引用循环（*reference cycles*）：它们如何泄露内存，以及如何防止其发生。

再见