Rust 编程语言入门

杨旭，微软MVP
Rust、Go、C#开发者

8.3 String（上）

Rust 开发者经常会被字符串困扰的原因

- Rust 倾向于暴露可能的错误
- 字符串数据结构复杂
- UTF-8

字符串是什么

- Byte 的集合
- 一些方法
  - 能将 byte 解析为文本

字符串是什么

- Rust 的**核心语言层面**，只有一个字符串类型：字符串切片 **str**（或 &str）
- 字符串切片：对存储在其它地方、UTF-8 编码的字符串的引用
  - 字符串字面值：存储在二进制文件中，也是字符串切片

- **String** 类型：
  - 来自 **标准库** 而不是 核心语言
  - 可增长、可修改、可拥有
  - UTF-8 编码

通常说的字符串是指？

- String 和 &str
  - 标准库里用的多
  - UTF-8 编码
- 本节讲的主要是 String

其它类型的字符串

- Rust 的标准库还包含了很多其它的字符串类型，例如：OsString、OsStr、CString、CStr
  - String vs Str 后缀：**拥有**或**借用**的变体
  - 可存储不同编码的文本或在内存中以不同的形式展现
- Library crate 针对存储字符串可提供更多的选项

创建一个新的字符串（String）

- 很多 Vec<T> 的操作都可用于 String。
- String::new() 函数
  - （例子）
- 使用初始值来创建 String：
  - to_string() 方法，可用于实现了 Display trait 的类型，包括字符串字面值（例子）
  - String::from() 函数，从字面值创建 String （例子）
- UTF-8 编码的例子

更新 String

- **push_str()** 方法：把一个字符串切片附加到 String（例子）
- **push()** 方法：把单个字符附加到 String （例子）
- **+**：连接字符串（例子）
  - 使用了类似这个签名的方法 fn add(self, s: &str) -> String { … }
    - 标准库中的 add 方法使用了泛型
    - 只能把 &str 添加到 String
    - 解引用强制转换（deref coercion）
- **format!**：连接多个字符串（例子）
  - 和 println!() 类似，但返回字符串
  - 不会获得参数的所有权

再见