Rust勉強会

TOMLと`Cargo.toml`

Naoya Tezuka & Takuto Nishimura

TOML

TOMLってなんですか？

TOML = Tom’s Obvious, Minimal Language

Tomさん = このフォーマットを考案したうちの一人，Tom Preston-Wernerのこと
設定ファイルのフォーマットの1種で、JSONとかyamlとかがライバル

これがTOMLだ！

```toml
# This is a TOML document
title = "TOML Example"
[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00-08:00
[database]
enabled = true
ports = [ 8000, 8001, 8002 ]
```

実装されている言語

最近の言語には大体実装済み!

> TOML already has implementations in most of the most popular programming languages in use today: **C, C#, C++, Clojure, Dart, Elixir, Erlang, Go, Haskell, Java, Javascript, Lua, Objective-C, Perl, PHP, Python, Ruby, Swift, Scala**...

詳しくはここ

TOMLの構文

テーブル(の配列)

```toml
[[products]]
name = "Hammer"
sku = 738594937

[[products]]

[[products]]
name = "Nail"
color = "gray"
```

```
# {
#   "products": [
#     { "name": "Hammer", "sku": "738594937"},
#       {},
#       { "name": "Nail", "sku": 284758393, "color": "gray" },
#   ]
# }
```

TOMLの構文

コメント

```
# This is a TOML comment
```

文字列

```
str1 = "Please call me \"TOM\"\n"
str2 = """
    You can use \
    multi-line strings. \
    """
str3 = 'single quotes mean "no escape"'
```

TOMLの構文

数(bin, oct, hexもあるよ)

```
int1 = 32
float2 = -3.14
inf3 = +inf
not4 = nan
bool5 = true
```

日付と時間

```
# 現地の日付
ld1 = 1979-05-27
# 現地の時間
lt2 = 21:59:44
```

YAML vs JSON

vs TOML

TOMLの良いところ1

仕様書が小さい

YAML：28スクロール！
JSON：6スクロール
TOML：7スクロール

実際にはTOMLの仕様書は殆どがサンプルコード

→ 本当に**言語仕様が小さい**!!

TOMLの良いところ2

コメントが書きやすい

YAML：# で簡単(アレ？)
JSON：かけません！！
TOML：# で簡単

TOMLの良いところ3

人間にとって読みやすい

```
- YAML:
 - インデント: 変えないとねえ
```

```
"JSON": {"クオーテーション多すぎ！": {"カッコも多すぎ": "ぴえん"}}
```

```
TOML = "読みやすい"
```

TOMLの良いところ4

パースしやすい(らしい)

YAML：めっちゃむずい
JSON：簡単
TOML：簡単

TOMLのまとめ

- TOMLは人間に優しいフォーマット
  - 読み書きしやすい
  - コメントも書ける
  - 仕様が小さい
- 機械にも優しい
  - パースしやすい
- JSONとYAMLのいいとこどり

`Cargo.toml`

`Cargo.toml`はmanifest file

manifest file = パッケージのメタデータを格納するファイル

```toml
# example of Cargo.toml
[package]
name = "hello_world" # the name of the package
version = "0.1.0"    # the current version, obeying semver
edition = "2021"
authors = "Sekai Aisatsu<helloWorld@example.com>"

[dependencies]
serde_json = "1.0"

[build-dependencies]
structopt = "0.3"
```

全内容の一覧はここ

`Cargo.toml`の役割

≃パッケージマネージャ`Cargo`の設定ファイル

- 依存ライブラリとそのバージョン
- ビルドターゲットのパスの指定
- コンパイル時のフラグ指定
- `features`(条件付きコンパイル)や`option`の指定

などを`Cargo.toml`に記述すれば，後は`Cargo`におまかせ！

`cargo-edit` を使おう

- `Cargo.toml` は `cargo-edit` を使って編集できるよ
  - もちろん、みんな各自のエディタから直接いじることもできるんだけど...
- `$ cargo upgrade` というコマンドを叩けば、`[dependencies]` にあるパッケージのバージョンを自動で更新してくれる！！
- `features` の追加も `--features` フラグで可能

`cargo-edit`の使い方

- `$ cargo add {パッケージ名}`：tomlにパッケージを追加する。バージョンを指定しても良いし、指定しなければ最新のものが追加される。
- `$ cargo rm {パッケージ名}`：tomlファイルに記載されているパッケージを削除する
- `$ cargo upgrade {パッケージ名}`：tomlファイルの特定のパッケージを最新(または指定したバージョン)にアップデート

公式ドキュメントはここ

`Cargo.toml`まとめ