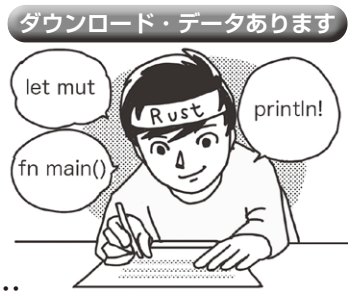
Rustプログラミング問題集



第5回 シリアライザ，ロガー，コマンドライン，HTTP… さまざまなクレートを使ってみる

中村 仁昭，懸川 岳

本連載ではRustを使った問題と回答形式でRustを学んでいきます．

今回は他言語でいうライブラリに相当するRustのクレートを使い，便利な機能を体験してみます．

（編集部）

● 実行環境

rustupでインストールした標準的な環境で動作確認しています．rustupでのインストール手順は公式ウェブ・ページ（https://www.rust-lang.org/ja/tools/install）を参照してください．インストールが難しい場合はウェブ・ブラウザの実行環境Rust Playground（https://play.rust-lang.org/）でも試せます．

1 JSON

serde_jsonクレートを使って構造体をシリアライズし，さらにシリアライズ結果から構造体にデシリアライズしてください．

シリアライズ元はi32型のx，yを持つ構造体Point，シリアライズ結果はJSON形式の文字列とします．

● 回答

JSONのシリアライズ/デシリアライズにserdeフレームワークとserde_jsonクレートを使います．Cargo.tomlに以下のように記載します．

```
[dependencies]
serde = { version = "1.0",
          features = ["derive"] }
serde_json = "1.0"
```

回答例を**リスト1**に示します．serdeはRustのデータ構造をシリアライズ/デシリアライズするためのフレームワークで，構造体（Point）に#[derive(Serialize, Deserialize)]を追加することでRustのトレイト機能により，コンパイル時にシリアライザ/デシリアライザ実装が自動で生成されます．このため効率的な変換が可能になっています．serdeはJSONの他にYAMLやMessagePackなどのデータ・フォーマットにも対応しています．

リスト1　構造体をJSON文字列にシリアライズ，デシリアライズする

```
use serde::{Deserialize, Serialize};

#[derive(Serialize, Deserialize, Debug)]
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 2 };

    let serialized = serde_json::to_string(&point)
                                                .unwrap();
    println!("serialized = {}", serialized);

    let deserialized: Point = serde_json::from_str(
                                &serialized).unwrap();
    println!("deserialized = {:?}", deserialized);
}
```

コンパイル時にシリアライザ/デシリアライザ実装が自動で生成される

2 ログ

env_loggerクレートを使ってロギングしてください．

● 回答

env_loggerクレートはRustの標準的なロギング・インターフェースのファサード[注1]を提供するlogクレートのロガー実装です．これを使うためには，Cargo.tomlに以下のような記載が必要です．

```
[dependencies]
log = "0.4"
env_logger = "0.6"
```

回答例を**リスト2**に示します．env_loggerは環

注1：デザイン・パターンの1つファサード・パターンのことで，共通インターフェースを提供し，どの実装を使用するかは利用者が選択する．