misskey-rs 紹介

`misskey-rs` は何？

- Misskey API を Rust から叩く
- Rust とは？
  - むかしのスライドを参照してください
  - https://coord-e.github.io/slide-coins20lt-rust-intro/
- Misskey API とは？
  - Misskey をプログラムから操作するためのやつ

？やればええやんけ

- Misskey API を Rust から叩く
  - 叩けばいいのでは？
- そうもいかない

まずリクエスト

```
let body = json!({
    "limit": 30
});
let client = reqwest::Client::new();
let res = client.post("https://missing.coord-e.dev/api/users")
    .json(&body)
    .send()
    .await?;
```

まあそれはそう

最初のユーザーのユーザー名を取り出す

```
res[0]["username"]
```

このコードがやっていること:

- 帰ってきたのが配列だと仮定して（そうじゃない場合があるから）
  - 本当に配列じゃなかったらクラッシュ
- 配列の 0 番目がオブジェクトだと仮定して（そうじゃない場合があるから）
  - 本当にオブジェクトじゃなかったらクラッシュ
- キー `"username"` がなかったらクラッシュ

オイオイ w

- 本当に配列じゃなかったら
  - 配列だが？
- 配列の 0 番目がオブジェクトだと仮定して
  - そうだが？
- `"username"` がなかったらクラッシュ
  - あるが？

型をつける

```
struct User {
  pub id: Id<User>,
  pub username: String,
  // あるかないかわからないやつは Option で！
  pub name: Option<String>,
}

// リクエストにも型をつけておくとエラーが少なくていいね
struct Request {
  pub limit: Option<u8>,
  pub since_id: Option<Id<User>>,
  pub until_id: Option<Id<User>>
}
```

そうすると `users` API は `Request` を受け取ってエラーか `Vec<User>` を返す関数と見ることができるな？

`misskey-rs` がやっていることその１

- リクエスト・レスポンスに型をつける
  - 「これを送るとふつうはこれが返ってくる」
- JS と同じような使い心地に
  - 補完も効く ← ！

`misskey-rs` v0.1 見た目

```rust
let note = client
    .request(
        misskey::endpoint::notes::create::Request::builder()
            .visibility(Visibility::Home)
            .text("Hello, Misskey")
            .build(),
    )
    .await?
    .into_result()?;

println!("{}", note.text.unwrap());
```

だるい

楽になると思っていたんだけど、ダメだった…

- Misskey API を覚えている必要はある
- 結果が二重の `Result` があってしんどい
  - 通信のエラー + Misskey が返すエラー

`misskey-rs` がやっていることその 2: 高レベル API

普通にライブラリっぽく使えたらいいよね

`misskey-rs` v0.2 見た目

```rust
// ノート
let note = client.create_note("Hello, Misskey").await?;

// リプ
client.reply(&note, "Hey, this is Cisskey").await?;

// リアクション
client.react(&note, ":2e2_face:").await?;
```

よさそう

"API バインディング" から "クライアントライブラリ" になった

ページネーション

- `notes/local-timeline` とか
  - 量が多いのでほしいだけ返してもらう
  - TL をスクロールするとロードが走るとおもうんだけどあれ
- この仕組みは多くのエンドポイントで使われている
  - 基本的にスクロールするとさらに読み込むタイプのやつはこれ

ページネーション

- これ、クソデカ遅延読み込み配列として扱えたらうれしくね？
- ストリームとして実装

170:6 error 不適切表現「クソ」が含まれています。

ページネーション見た目

すべての（！）ノートを画面に出力

```
client.local_notes(..).try_for_each(|note| async {
  println!("{}", note.text.unwrap_or_default());
  Ok(())
}).await?;
```

さらに…

`note1` と `note2` の間のノートをすべて取得とか

```
client.local_notes(note1..note2)
```

`time1` と `time2` の間のノートをすべて取得とか

```
client.local_notes(time1..time2)
```

あるノートの前後 n ノートを取るとかもできる

ストリーミング API

リアルタイムに取得

- ノートが来たら全部にリアクション

```
client.local_timeline().await?.try_for_each(|note| async move {
  client.react(&note, "👀").await
}).await?;
```

ストリーミング API

- 自動フォロバ

```
client.main_stream().await?.try_for_each(|event| async {
  if let MainStreamEvent::Followed(user) = event {
      println!("followed from @{}", user.username);

      if !client.is_following(&user).await? {
          client.follow(&user).await?;
      }
  }
  Ok(())
}).await?;
```

それ以外

- Rust から Misskey を扱うためのライブラリを書いた
  https://coord-e.com/post/2020-12-19-misskey-rs.html

v0.3 リリースに向けて

- 最新の Misskey に対応
  - レジストリとかは実装した
- なんかバグ修正
- ライブラリのアプデ

おしまい