

言語間で比較する エラーの通知と処理

nsfisis (いまむら)

第 154 回 PHP 勉強会@東京

自己紹介

nsfisis (いまむら)



@ デジタルサーカス株式会社

話すこと、 話さないこと

- 話すこと
 - 様々な言語におけるエラー通知の方法
- 話さないこと
 - 君たちはどう生きるか (PHP におけるベストプラクティス)

取り上げる言語

- C
- C++
- Java
- Go
- Rust



c

C: 成功 / 失敗を戻り値で表す

```
#include <stdbool.h>

bool do_something() {
    // ...
    if (success) {
        return true;
    } else {
        return false;
    }
}
```

PHP では `mkdir()` など

C: 成功 / 失敗を戻り値で表す

- **Bad** 処理結果があるときに使えない
- **Bad** エラーに情報を載せられない
- **Bad** エラーハンドリングを省略できる
 - **Bad** 省略したとき、後続の処理が一切止まらない

C: 処理結果または失敗を返す

```
something* do_something() {  
    // ...  
    if (success) {  
        return result;  
    } else {  
        return NULL;  
    }  
}
```

PHP では `fopen()` など

C: 処理結果または失敗を返す

- **Good** 処理結果があるときに使える
- **Bad** エラーに情報を載せられない
 - 注 : 動的型付き言語の場合はその限りでない
- **Bad** エラーハンドリングを省略できる
 - **Good** 省略したとき、処理結果を使おうとすると止まる
 - **Bad** 省略したとき、後続の処理が進みうる

C: 引数で処理結果を受け取る

```
#include <stdbool.h>

bool do_something(something* result) {
    // ...
    if (success) {
        *result = ...;
        return true;
    } else {
        return false;
    }
}
```

PHP では `preg_match()` など

C: 引数で処理結果を受け取る

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Bad エラーハンドリングを省略できる
 - Good 省略したとき、処理結果を使おうとすると止まる
 - Bad 省略したとき、後続の処理が進みうる

C: グローバル状態から通知する

```
some_error error;  
  
void do_something() {  
    // ...  
    if (success) {  
        error = 0;  
    } else {  
        error = 42;  
    }  
}
```

PHP では `json_decode()` と `json_last_error()` など

C: グローバル状態から通知する

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Bad エラーハンドリングを省略できる
 - Bad 省略したとき、後続の処理が一切止まらない

C++

C++: 例外を投げる

```
#include <stdexcept>

void do_something() {
    // ...
    if (!success) {
        throw std::runtime_error{"..."};
    }
}
```

PHP では `DateTimeImmutable::__construct()` など

C++: 例外を投げる

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Bad エラーハンドリングを省略できる
 - Good 省略したとき、そこで止まる



Java

Java: 検査例外

```
void doSomething() throws SomeException {  
    // ...  
    if (!success) {  
        throw new SomeException(...);  
    }  
}
```

PHP では `@throws` が (一応) 近い

Java: 検査例外

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Good エラーハンドリングを省略するとコンパイルエラーになる
 - Good 省略ができない

Java: 検査例外

- **Good** 処理結果があるときに使える
- **Good** エラーに情報を載せられる
- **Good** エラーハンドリングを省略するとコンパイルエラーになる
 - **Good** 省略ができない
- **Bad** どこでどの例外が投げられるのかわかりにくい
 - 注 : 例外自体の問題ではなく、C++ や Java が採用している文法の問題
- **Bad** 大域脱出は処理の動きが複雑になる
- **Bad** 例外クラスの継承ツリー設計が困難
- **Bad** 回復不能なエラーまで例外で表される
- 検査例外固有の問題がいくつかある (ここでは割愛)

Go

Go: 多值返却

```
func doSomething() (int, error) {  
    // ...  
    if success {  
        return 42, nil  
    } else {  
        return 0, errors.New("...")  
    }  
}  
  
result, err := doSomething()  
if err != nil {  
    // ...  
}
```

Go: 多值返却

```
<?php
function do_something(): array {
    // ...
    if ($success) {
        return [42, null];
    } else {
        return [null, new SomeError(...)];
    }
}

[$result, $err] = do_something();
if (isset($err)) {
    // ...
}
```

Go: 多値返却

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Bad エラーハンドリングを省略できる
 - Good linter での検知は可能

Go: panic

```
func doSomething() int {  
    // ...  
    if !success {  
        panic("...")  
    }  
    return 42  
}
```

回復不能なエラー。通常のエラーと同じ方法では捕まえない

Go: 多値返却 + panic

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Bad エラーハンドリングを省略できる
 - Good linter での検知は可能
- Good 特殊な構文なし、大域脱出なし
- Good エラー値のカテゴリズに継承が使われない
- Good 回復可能なエラーと回復不能なエラーを区別できる
- Bad 静的型付き言語以外だと、C と同レベルの安全性になる

Rust

Rust: 代数的データ型

```
fn do_something() -> Result<i32, SomeError> {  
    // ...  
    if success {  
        Ok(42)  
    } else {  
        Err(SomeError::new())  
    }  
}
```

PHP では union 型が一番近い

Rust: 代数的データ型 + panic

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Good エラーハンドリングを省略するとコンパイルエラーになる
 - Good 処理結果がない場合でも、linter での検知は可能
- Good 特殊な構文なし、大域脱出なし
- Good エラー値のカテゴリズに継承が使われない
- Good 回復可能なエラーと回復不能なエラーを区別できる

Rust: 代数的データ型 + panic

- Good 処理結果があるときに使える
- Good エラーに情報を載せられる
- Good エラーハンドリングを省略するとコンパイルエラーになる
 - Good 処理結果がない場合でも、linter での検知は可能
- Good 特殊な構文なし、大域脱出なし
- Good エラー値のカテゴリズに継承が使われない
- Good 回復可能なエラーと回復不能なエラーを区別できる
- Bad 静的型付き言語以外だと、C と同レベルの安全性になる
- Bad 言語側にそれ相応のシンタックスシュガーがないと書きづらい

PHP

我らが PHP は ...

C 言語の素朴な値による通知と例外をミックスしたキメラ
どう設計すべきかは今回のスコープ外

我らが PHP は ...

C 言語の素朴な値による通知と例外をミックスしたキメラ
どう設計すべきかは今回のスコープ外
君たちはどう生きるか