創好リナの

Programming Seminar

7bit

API

2018.09.30

APIを作ろう

- モジュール化という考え方
- Luaのモジュール化
- LoggingAPI見本

今日は解説少なめ

でも大事

APIを作ろう

モジュール化 #とは

- 似たような処理を毎回書くのは辛い
- 一つのファイルに全て書くのは辛い

-> モジュール化することで解決

モジュール化 #とは

- 似たような処理を使い回せるようにする
- 処理をファイルに分ける
- 同じ物事に関する処理をまとめたものがAPIと呼ばれる
  - クラスのようなもの
- `fs`や`turtle`がAPI
- ライブラリとも言う
  - 今はAPIと言うとRESTなどWebシステムのイメージが強いため、ライブラリと呼ぶ

Luaのモジュール化

- JavaScriptと似てる(Node.js | ES6)
- `require()`関数でモジュール名を指定して使う
- モジュール側はオブジェクトをreturnで返す
- モジュール名
  - パス区切りは`.` (相対パスは使えない)
  - 拡張子(`.lua`)は付けない

- main.lua

```lua
local sub = require("hoge.sub")

sub.func() -- sub module
```

- hoge/sub.lua

```lua
local tbl = {}

function tbl.func()
  print("sub module")
end

return tbl
```

しかし!!!!!

CCLuaにはrequire()が無い!!!

のは昔の話だった…

1.12版にはあります

今日の問題は作りづらいので

Logging API を作ってみよう

Logging API を作ってみよう

- 簡単なログを取るAPIを作ってみる
- ファイルや標準出力にログを出力する

ログの重要性

- ログを取らないとエラーや、予期せぬ動作があったときに特定し辛い
- 今どこを処理してるかリアルタイムで分かる
- 安心感がある

仕様

- `log.trace()` / `log.debug()` / `log.info()` `log.warn()` / `log.error()` / `log.fatal()`
  - それぞれのレベルでログが出力
- 出力形式
  - `[レベル][時間][コンピュータラベル]: メッセージ`
- `log.file = ファイル名` でファイル出力に変更