Rack

やまもとじゅん

Version 0.1, 2018/12/16

目次

Rackはサーバとアプリケーション／フレームワーク間のインターフェース（HTTPリクエストとレスポンス）を可能なかぎり簡単な方法でラッピングし、応用の利くインターフェイスを提供する。

Chapter 1. Rackの目的

インターフェースが統一されていれば，サーバやフレームワークの組み合わせが自由になる

指定したファイルを独自の **Ruby DSL** として読み込み、DSLで指定した様々なミドルウェア、アプリケーションを組み合わせてWebサーバを立ち上げることができる **rackup** というコマンドを提供するライブラリ

NOTE	Ruby on Railsをインストールしている場合はすでに必須モジュールとして入っています

Chapter 2. Rackアプリケーションとして最低限必要なこと

- callというメソッドを持っていること
- callメソッドの引数としてWebサーバからのリクエストを受けること
- callメソッドは，次の要素を含むレスポンスを返すこと
  - ステータスコード
  - レスポンスヘッダ（Hash）
  - レスポンスボディ（Array）

Chapter 3. 初めての Rack

simple_app.rb

```
class SimpleApp
  def call(env)
    p env
    case env['REQUEST_METHOD']
    when 'GET'
      [
        200,
        { 'Content-Type' => 'text/html' },
        ['<html><body><form method="POST"><input type="submit" value="見たい？"
/></form></body></html>']
      ]
    when 'POST'
      [
        200,
        { 'Content-Type' => 'text/html' },
        ['<html><body>何見てんだよ</body></html>']
      ]
    end
  end
end
```

NOTE 基本的にはconfig.ruに色々書いて，アプリケーション側にはサーバ依存のコードは書かない。（依存コードを書くと汎用性が下がり意味が薄くなる）

3.1. config.ru & rackup

config.ruという **rackup**ファイル を用意し，rackupといういうコマンドを使ってアプリケーションを起動

config.ru

```
require './simple_app.rb'
run SimpleApp.new
```

```
rackup config.ru
```

NOTE **.ru**: rackupファイルと呼びます。拡張子のruはおそらくrackupの略でしょう

Chapter 4. Rack::Request/Rack::Response

Rack::Request と **Rack::Response** というラッパーが用意されている。HashやArrayよりは少し扱いやすくなる。

```
require 'rack/request'
require 'rack/response'

class SimpleApp
  def call(env)
    req = Rack::Request.new(env)

    body = case req.request_method
           when 'GET'
             '<html><body><form method="POST"><input type="submit" value="見たい？"
/></form></body></html>'
           when 'POST'
             '<html><body>何見てんだよ。</body></html>'
           end

    res = Rack::Response.new { |r|
      r.status = 200
      r['Content-Type'] = 'text/html;charset=utf-8'
      r.write body
    }
    res.finish
  end
end
```

Chapter 5. Rack::URLMap

Rack標準添付の「アプリケーション」。Rack::Builderにも上手く組み込まれている。
パスとアプリケーションのマッピングを保持しておき，パスに応じてリクエストを登録してあるアプリケーションに振り分けてくれます。

```
require 'simple_app'
require 'rack/lobster'

map '/simple' do
  run SimpleApp.new
end

map '/lobster' do
  # Rackをインストールすると
  # サンプルとして付いてくるアプリケーション
  # ちょっと面白い
  run Rack::Lobster.new
end
```

Chapter 6. 参考にした資料

- Rackとは何か @gihyo.jp
- Rails と Rack @Ruby on Rails ガイド