





大阪国際大学経営情報学部 石川高行

RAILS を教える順序



この発表の要


- 最短距離の教育
- default 認識
- 到達目標の設定



教育者の思考停止

「どうしてこの順序で教えるんですか？」
「私がこの順序で教わったからだ」


- 技巧的な因数分解 → 解の公式
- 微分 → 積分
- 直観的な極限 → ϵ - δ (→ 超実数)
- 直滑降 → プルーク → シュテム → パラレル



情報教育での思考停止

- C 言語至上主義
 - 「なにはともあれ、まずはC言語」
 - 「C++ を学ぶ前にCの知識が必要です」
 - 「GUI programmingなんてまだまだ早い。Cで関数を学んで、C++でオブジェクト指向を学んで、それでやっとGUIに手を出すんだ」


思いつく限りの回り道をさせているとしか見えない



初心者には (C 言語の例)

- float の代わりに double だけで充分じゃない?
- 共用体 (union) なんて教えなくてもいいんじゃない?
- 文字型配列の代わりに文字列クラスで充分じゃない?(Cにはなくても他の言語には大抵ある)

最短距離の教育を!



教える順序と認識 (1)

- どちらの順序が良い?
 - 「if という条件分岐があります」「else 節を付け加えることもできます」
 - 「if ~ else という条件分岐があります」「else 節は省略することもできます」

どちらから教えるか、で学習者の default 認識は違ってくる

大阪国際大学

教える順序と認識 (2)

- どちら(の順序)が良い?
 - 「MVCや永続化なんて知らないよ。プログラムはスタンドアローンで動けばいいんじゃないの?」
 - 「え、他人とデータ共有? データの永続化? じゃあ、ネットワークを介すからライブラリを探して、と.....面倒だなあ。」
 - 「MVCは普段から意識しているよ。」
 - 「他人とデータ共有? データの永続化? 最初からそのつもりでRailsを使っているよ。」

大阪国際大学

MVC をどう教えるか (1)

「使わないと損をする Model-View-Controller」
<http://www.jac-net.com/~tarzan/smalltalkers/mvc/mvc.html>

大阪国際大学

MVC をどう教えるか (2)

- Model とは
 - スタンドアローンのゲームのデータは、ゲーム機の中
 - ゲーム機1台に Model 1つ
 - 自分がラスボス倒しても友人の世界は別
 - ネットワークゲームのデータは、サーバの中
 - サーバ1台に Model 1つ (あるいはいくつか)
 - 友人と一緒にラスボスを倒せる

身近な例を使う、の原則

大阪国際大学

MVC をどう教えるか (3)

- View とは
 - FFやDQで、地上を歩いても空を飛んでも、見え方が異なるだけで世界は同じ
 - mixiをPCで開いても携帯電話で開いても、見え方が少々異なるだけで見えてみるものは同じ

大阪国際大学

MVC をどう教えるか (4)

- Controller とは
 - gamepad も joystick も、好みに応じて使い分ければ良い。
 - このたとえばイマイチ。
 - Ruby on Rails では server への HTTP request が controller. (そう言い切って良いだろうか?)

大阪国際大学

永続化を default 意識に (1)

- DHH 『AWDwR』

```
drop table if exists products;
create table products (
  id          int          not null auto_increment,
  title       varchar(100) not null,
  description text         not null,
  image_url   varchar(200) not null,
  price       decimal(10,2) not null,
  date_available datetime  not null,
  primary key (id)
);
```

永続化を default 意識に (2)

- 『ライド・オン・Rails』 p.33

```
class CreateBooks < ActiveRecord::Migration
  def self.up
    create_table :books do |t|
      t.column :isbn, :string
      t.column :name, :string
      t.column :author, :string
      t.column :manufacturer, :string
      t.column :release_on, :date
    end
  end

  def self.down
    drop_table :books
  end
end
```

永続化を default 意識に (3)

- Model の定義はなるべく1度で (Rails の中だけで) 終わておきたい
- 永続化のためだけに手間をかけたくない (学習者に「その手間が面倒だ」と感じさせたら、default 意識にならない)
- db:migrate で DB に自動的に反映

M から作るか C から作るか

- 「Rolling with Ruby on Rails」

<http://blog.livedoor.jp/zep716/archives/24193487.html>

- ruby script\$generate controller MyTest
- http://127.0.0.1:3000/My_Test/
- 「Unknown Action」

```
class MyTestController < ActionController::Base
  def index
    @hello = "Hello World!"
  end
end
```

“Hello, World!” 病?

MVC の M が最重要

- M のない VC なんて虚しい
 - generate Model
 - db:migrate
 - generate scaffold
 - WEBrick
- この順序が王道ではないか (論拠薄弱、議論の余地あり)


重要な folders

- “rails” command で多くの folders が生成されるが、重要なのはここ

```
rails
  app
    : アプリケーションの根となるファイルを置くディレクトリ
  config
    : コントローラファイルのディレクトリ
  lib
    : ヘルパーファイルのディレクトリ
  public
    : セーブファイルのディレクトリ
  tmp
    : テンプレートファイルのディレクトリ
  vendor
    : 外部ライブラリーやプラグインを置くディレクトリ
  vendor/assets
    : 外部ライブラリーやプラグインの静的コンテンツ
  vendor/cache
    : 各種静的ファイルのディレクトリ
  vendor/javascript
    : 静的コンテンツのディレクトリ
  vendor/stylesheets
    : プラグインの静的コンテンツ
  vendor/translations
    : プラグインの静的コンテンツ
  vendor/vendor
    : プラグインの静的コンテンツ
```


helper の存在意義 (1)

- 「『<%=link_to.....』だなんて面倒だ。直接『<a href="....."』って書かせてほしい。」
- 誰か、こういう学習者に説明してあげて下さい。




到達目標の設定 (1)

- とりあえずどこまで教えたら Rails 入門と言えるか
 - DB 操作経験のない大学生を対象と想定
 - 枝葉は切る、幹は残す
 - しかし、「どこからが枝葉か」の判断はかなり主観的



到達目標の設定 (2)

- とりあえずどこまで教えたら Rails 入門と言えるか
 - 1対多の関係を作れるところまで
 - scaffold に頼らずに作る部分が出てくる
 - 例: 「レシピには、カテゴリー (デザートのような) を指定できるようにしたい。そして、特定のカテゴリーに属するレシピだけを一覧できるようにしたい。これができるようにするには、データベースのカテゴリーテーブルの追加と、所属するカテゴリーを示すフィールドのレシピテーブルへの追加が必要になる。」 (前出「Rolling with RoR」)



test は?

- 「agile programming として test は良い習慣。学習の最初から習慣化すべき。」
- 「Rails 本で最も躓きやすい部分が test。test に通らなくても稼動すること多いから、最短距離のために test は省略すべき。」

どちらの考えを支持しますか?



Rails 検定 (1)

- 学習者に到達度の「見える化」→検定
 - 技能を公的に保証するものでなくてよい
 - 学内検定で充分
- 勝手に作ってしまう
 - 皆さんからの案があったら伺いたい



Rails 検定 (2)

- 5級: Ruby, Rails, DB を install できる
- 4級: DB に database, table, user を設定し、権限を設定できる (GUI 使用可)
- 3級: 単一 model を generate, migrate, generate scaffold し WEBrick を起動できる
- 2級: view を変更することができる
- 1級: 1対多の関係を適切に実装できる



Rails 学習者の standard

- 以下の blog は必見? 他には?
 - masuidrive on rails
<http://blog.masuidrive.jp/>
 - \(.v.)ノくまくまー
<http://wota.jp/ac/>
 - 川口...) <2nd life
<http://d.hatena.ne.jp/secondlife/>
- mixi の Rails 初心者質問箱は?
- 他には何があるだろう?