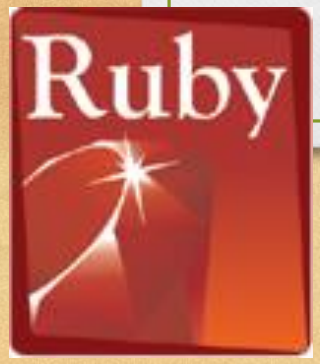
Ruby on Rails (RoR) って何? 体験会

北海道大学 大学院理学院

宇宙理学専攻

荻原 弘尭



目次

- はじめに
- Ruby on Rails(基本)
- 現在稼働中のサービスについての紹介
- 今から始めるののおすすめの文献

はじめに

諸注意

- Ruby on Rails について全てわかるわけではありません．正しくないこともしゃべるかもしれません．ご了承ください．
  - この頃触れていないし…
- この資料は 2016/07/22 に三上さんが発表した資料を手直ししただけのものです．聞いたことがある人はすいません．
- 興味のある人は各自勉強してください
- 質問・コメントはその都度していただいて結構です

なぜ話そうと思ったか

- Ruby on Rails は巷や PSG 内でも結構使われている
  - PSG の中でも結構重要なものとなってしまった (と思う)
    - suu
    - Prrs
- しかし, 今の PSG の世代の子達はあまり知る機会がなかったと思う
  - ミーティングとかで話したときに置いてきぼりにならないようになってもらいたい
- つまり消え行く者の老婆心

今日伝えたいこと

- Ruby on Rails というものがあり, PSG でも使われている
- 大まかな Ruby on Rails の仕組み
  - 動かなくてもどう動かないか管理者へ報告できるようになってほしい
- 何かを作るということに興味持ってもらいたい
  - RoR プロジェクト内でも凍結したプロジェクトが多くある. できるならそれを蘇らせてくれ！

実は多くの人が触れている
Ruby on Rails

- みなさんがよく使っているWebアプリケーションが Ruby on Rails を用いて作られている

COOKPAD

github

食べログ

rakuten

惑星宇宙グループでも…

- 現在運用中
  - INEX レポート投稿システム -suu-
    - 管理者: 村橋
  - PSG 施設予約システム -Prrs-
    - 管理者: 荻原
- 凍結されたもの
  - Redmine -EPredmine-
    - 管理者: 三上
  - 書籍管理システム(ベータ版) -EPbook-
    - 管理者: 三上
- 全てjet サーバで動いています
  - https://jet.ep.sci.hokudai.ac.jp/

- suu
- epredmine
- prrs
- epbook

更新情報

- **2016.07.21** [epbook] 書籍管理システム ベータ版の運用を開始(0.1.0).
- **2016.06.05** [epredmine] epredmine の運用を再開(2.0.0), Rails4 に対応.
- **2016.04.13** [suu] レポート投稿システムの改修(2.0.2). 詳細
- **2016.03.08** [suu] 2016 年度用レポート投稿システム稼働開始(2.0.1).
- **2016.03.05** [suu] OS アップグレードに伴い, レポート投稿システムの大幅改修(2.0.0). 詳細
- **2016.03.05** [prrs] PSG 施設予約システム (3.0 -どらきち-) 運用開始
- **2016.03.04** [ozuma] OS アップグレードに伴い, 凍結中
- **2016.03.04** JET サーバのOS をjessie にアップグレード
- **2015.10.05** [prrs] PSG 施設予約システム の本運用開始
- **2015.09.26** [suu] 予約名の上限撤廃

copyright © inex 2016 All rights reserved.

Ruby on Rails

Ruby on Rails (Rails, RoR)

- Ruby 言語で記述され, Ruby 環境で動作するWeb アプリケーションフレームワーク
  - Ruby: オブジェクトスクリプト言語
- 開発者
  - David Heinemeier Hansson
    - デンマークのプログラマ
- 開発年
  - 2004 年7 月に最初のバージョンが公開
    - 最新安定版: Rails 5.1.4 (2017/09/07)
    - 前安定版: Rails 4.2.7 (2016/07/13)

Web アプリケーションフレームワーク

- フレームワーク(枠組み)
  - 「開発・運用・意思決定を行う際に、その基礎となる規則・構造・アイデア・思想などの集合のこと。」(Wikipedia, フレームワーク, 2017年3月6日)
- アプリケーションフレームワーク
  - 再利用可能なクラス・ライブラリの集合
    - アプリケーションを新規作成する際に，再利用可能なコードをまとめておくことで開発者の手間を省くことができる
- Web アプリケーションフレームワーク
  - Web サイト等の開発をする際に用いられるアプリケーションフレームワーク

アプリケーションフレームワークのメリット

- 開発生産性の向上
  - コード開発に規約があるので品質均質化
  - 比較的役割分担がしやすい
- メンテナンス性に優れる
  - コードの一貫性による可読性の向上
- 先端の技術トレンドに対応しやすい
  - フレームワーク規模での対応が可能
    - HTML5 など
- 一定以上の品質が期待できる
  - もともとある程度できているものから作るため

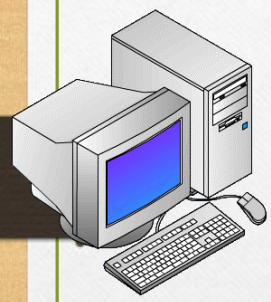
アプリケーションフレームワークのデメリット

- 制約が多いので慣れるのに時間がかかる
- こんなアプリケーションはフレームワークを導入しないほうがいい
  - 小規模なその場限りのアプリケーション
    - 開発コスト 大

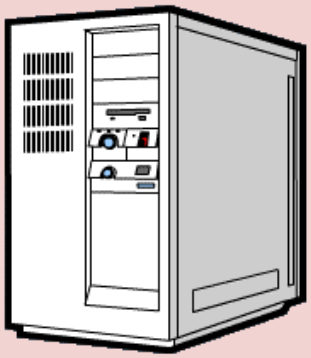
Rails プログラミングに必要な環境

- Ruby
- HTTP サーバ
  - Apache, WEBrick…
- Database
  - SQLite, MySQL…
- Ruby on Rails

Rails プログラミングに必要な環境



クライアント
(Web ブラウザ)



HTTP サーバ

Ruby on Rails

Ruby

SQLite

Database

とりあえず動かしてみる？

- 目の前に情報実験機があるみなさんぜひRails を動かしてみましょう
  - これ以降の作業は

    Debian 9 stretch で行われたものです (ただし Debian 8 Jessie でもできると思う)

    - Windows, Mac は知らないのでやりたい人は調べてください

    簡単な導入しかやりません.

    - Apache 立ち上げていろいろしたいときはもっと作業が必要となるので今日はやりません

Rails の準備 (@ Debian 9, stretch)

- Ruby のインストール

  # apt-get install ruby ruby-dev

- SQLite のインストール

  # apt-get install sqlite3 libsqlite3-dev

- その他必要パッケージインストール

  # apt-get install zlib1g-dev make gcc g++

- Ruby のバージョンのチェック

  # ruby –v

  ruby 2.3.3p222 (2016-11-21)　(rails 5 を使うには ruby 2.2.2 以降が必要 Jessie 等で ruby 2.1.5 の場合は ruby のバージョンを上げるか以下で rails4 をインストールしてください

- Rails のインストール(時間がかかる)

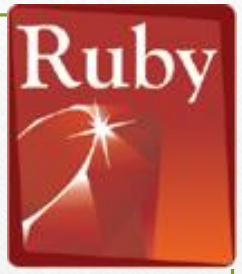
  # gem install rails

  デフォルトだと 最新盤 5.1.4 が入る. ruby が古くてはいらない場合は

  # gem install rails –v 4.0

  としてください.

Rails インストールの合間に Gem とは



- Gem: 宝石の意
- Ruby で使われるライブラリやアプリケーションのこと
  - RubyGems と呼ばれるパッケージ管理ツールを使ってダウンロードしたりできる
- Rails では Bundler と呼ばれるものでGem パッケージの種類やバージョンをそろえることができる
  - コマンドは bundle install

  Rails ではGemfile と呼ばれる必要パッケージの記載されたファイルに基づいてインストールが行われる.

Rails サーバ立ち上げ，動作確認

- test1 ディレクトリの作成(test だと怒られる)

  $ rails new test1 (sudo のパスワードが聞かれる)

- 必要なgem のインストール

  # gem install therubyracer

  $ cd test1

  Gemfile をエディタで開き，”therubyracer” の行をコメントイン

  $ vim Gemfile

  gem ‘therubyracer’, platform: ruby

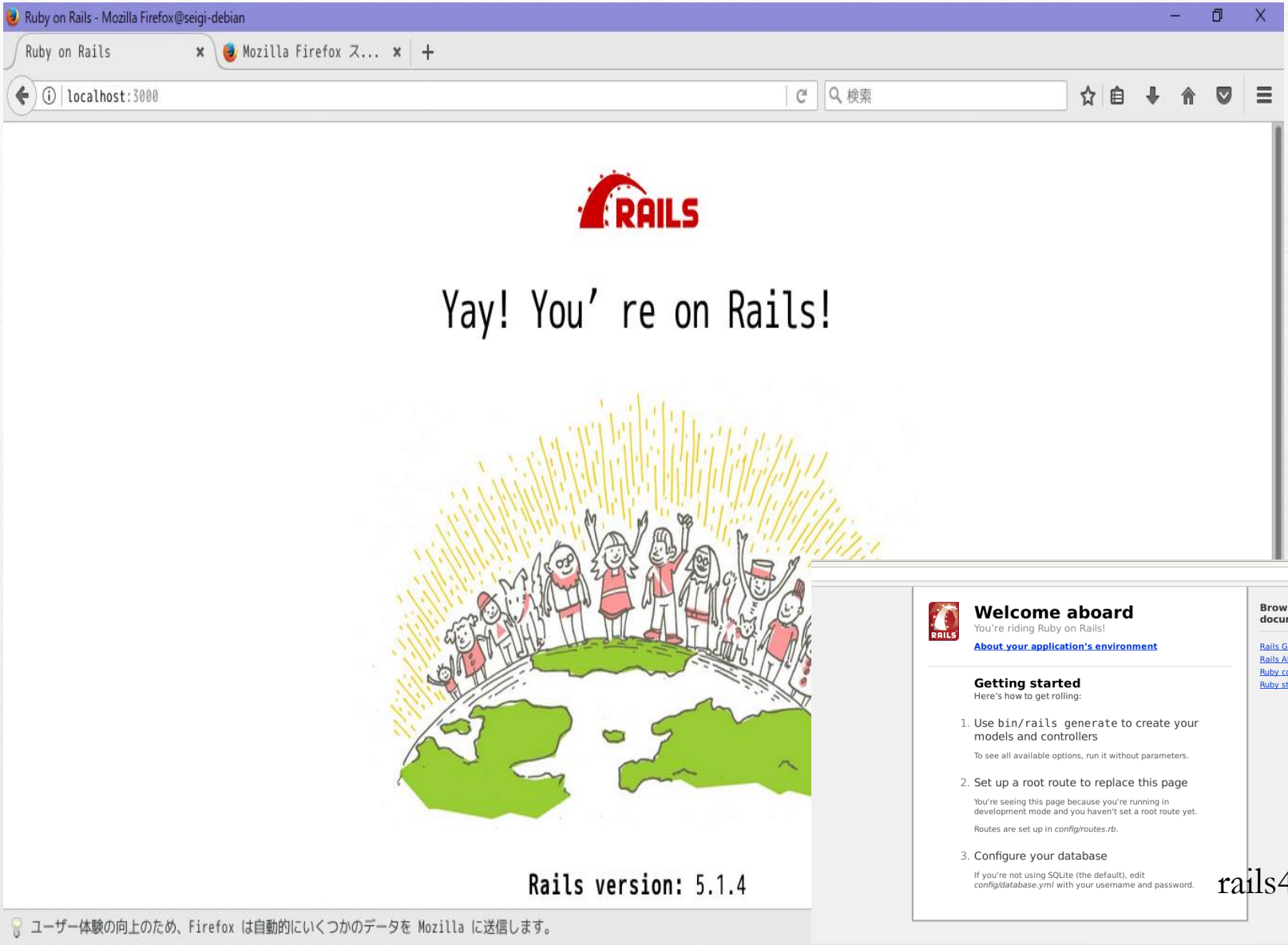
  Bundler でインストール

  $ bundle install

- test1 ディレクトリ上でRails サーバを立ち上げる

  $ rails server

- WEBrick で動作を確認
  - ブラウザを開いて，”http://localhost:3000/” と打ち込むと…


Ruby on Rails - Mozilla Firefox@seigi-debian
Ruby on Rails
Mozilla Firefox ス...
localhost:3000
検索
RAILS
Yay! You're on Rails!
Rails version: 5.1.4
ユーザー体験の向上のため、Firefox は自動的にいくつかのデータを Mozilla に送信します。
Welcome aboard
You're riding Ruby on Rails!
About your application's environment
Getting started
Here's how to get rolling:
1. Use bin/rails generate to create your models and controllers
To see all available options, run it without parameters.
2. Set up a root route to replace this page
You're seeing this page because you're running in development mode and you haven't set a root route yet.
Routes are set up in config/routes.rb.
3. Configure your database
If you're not using SQLite (the default), edit config/database.yml with your username and password.
Browse the documentation
Rails Guides
Rails API
Ruby core
Ruby standard library

rails4 の場合

ディレクトリ構造

- アプリケーションルート以下の構造は基本的に以下のようになる

```
test1/
  |--app (アプリケーションメインフォルダ)
       |--models -- *.rb
       |--controllers -- *_controllers.rb
       |--views  -- 各クラス -- *.html.erb
       …
  |--db (データベース置き場)
       …
  Gemfile (必要なGem ファイルの定義)
       …
```

ソフトウェアアーキテクチャ

MVC パターンを採用

- Model
  - データベースとのデータのやりとり
- View
  - データを表示(HTML)
- Controller
  - ユーザの入力に対し応答・処理

MVC の基本的なシナリオ

Web ブラウザ

Controller

Database

View

Model

MVC の基本的なシナリオ

Web ブラウザ

①UI を通してパラメータを入力

⑦View のテンプレートを html 文書としてブラウザに出力

Controller

Database

⑤Model からデータを取得

④データのやりとり

⑥結果を View に渡す

②結果を統合したテンプレートを返す

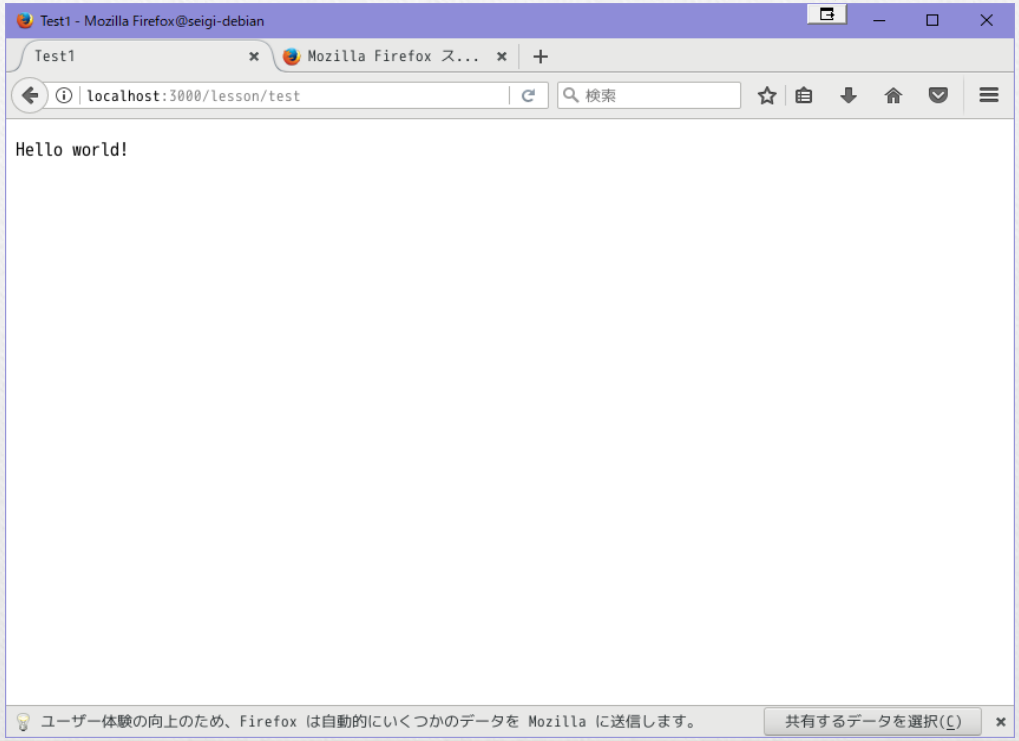
②Controller がユーザの操作に応じたデータのやりとりを指示

View

Model

多分よくわかんないと思うので，簡単な例で試す1

- 画面に Hello world! と表示する (ただし, 今回は model と DB のやり取りは行っていない.).



多分よくわかんないと思うので，簡単な例で試す1

- lesson という Controller を追加
  $ rails generate controller lesson
- Controller にアクション (指令) を追加
  $ cd test1/app/controller
  $ vim lesson_controller.rb
  - class LessonController < ApplicationController do end の間に以下を追加
    - def test
      @var = 'hello world!'
      end
- Controller のアクションがどこにあるのかを設定 (ルーティング)
  $ cd test1/config/
  $ vim routes.rb
  - Rails.application.routes.draw do end の間に以下を追
    - get "lesson/test"
- test.html.erb という View を追加
  $ cd test1/app/view/lesson
- $ vim test.html.erb
  - <%= @var %>
- ブラウザで http://localhost:3000/lesson/test を開く

だけど…結構大変

- DB, Model, Controller (routes.rb), View いろいろファイルを作成しないといけない
  - 先ほどは練習のために適当だったが Rails ではファイル名，アクション名などもルールがある (CoC)

CoC(Convention over Configuration)
- 設定より規約
- 慎重に設計された規約に従うことにより，設定が不要(あるいは軽減)

- 一から全部作っていくのはなかなか厳しい…

そんなあなたに…

Scaffolding 機能!!

Scaffolding 機能 (スキャフォールディング)

- Scaffolding: 足場
- 基本機能をあらかじめ実装したアプリケーションの骨格を作成する機能
- 基本的な仕様を簡単に手早く作れる

Scaffolding 機能による開発

- アプリケーションルートに移動

  $ cd test1

- 関連ファイルの作成

  $ rails generate scaffold book title:string price:integer published:date

```
ogihara@seigi-debian: ~/test1/config
ogihara@seigi-debian:~/test1/config$ vim routes.rb
ogihara@seigi-debian:~/test1/config$ rails generate sca
ffold book title:string price:integer published:date
Running via Spring preloader in process 22938
      invoke  active_record
      create    db/migrate/20180119040516_create_books.
rb
      create    app/models/book.rb
      invoke    test_unit
      create      test/models/book_test.rb
      create      test/fixtures/books.yml
      invoke  resource_route
       route    resources :books
      invoke  scaffold_controller
      create    app/controllers/books_controller.rb
      invoke    erb
      create      app/views/books
      create      app/views/books/index.html.erb
      create      app/views/books/edit.html.erb
      create      app/views/books/show.html.erb
      create      app/views/books/new.html.erb
      create      app/views/books/_form.html.erb
      invoke    test_unit
      create      test/controllers/books_controller_tes
t.rb
      invoke    helper
      create      app/helpers/books_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/books/index.json.jbuilder
      create      app/views/books/show.json.jbuilder
      create      app/views/books/_book.json.jbuilder
      invoke  test_unit
      create    test/system/books_test.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/books.coffee
      invoke    scss
      create      app/assets/stylesheets/books.scss
      invoke  scss
      create    app/assets/stylesheets/scaffolds.scss
ogihara@seigi-debian:~/test1/config$
```

Scaffold により作成, 修正されるファイル

- app 以下
  - views/books/index.html など
  - controllers/books_controller.rb
  - models/book.rb など
- config 以下
  - routes.rb
- db 以下
  - migrate/YYYYMMDD_create_books.rb

その他多数…

app/views/books/index.html.erb

- 勝手にHTMLファイルが作成される

```
<h1>Listing Books</h1>

<table>
  <thead>
    <tr>
      <th>Title</th>
      <th>Price</th>
      <th>Published</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <% @books.each do |book| %>
      <tr>
        <td><%= book.title %></td>
        <td><%= book.price %></td>
        <td><%= book.published %></td>
        <td><%= link_to 'Show', book %></td>
        <td><%= link_to 'Edit', edit_book_path(book) %></td>
        <td><%= link_to 'Destroy', book, method: :delete, data: { confirm: 'Are you sure?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>
```

マイグレーションファイル
db/migrate/YYYYMMDD_create_books.rb

- データベース(テーブル)の作成に用いるファイル

```
class CreateBooks < ActiveRecord::Migration
  def change
    create_table :books do |t|
      t.string :title
      t.integer :price
      t.date :published

      t.timestamps null: false
    end
  end
end
```

実際に出来上がったものをブラウザで見てみる

- マイグレーションファイルの実行によるテーブルの作成

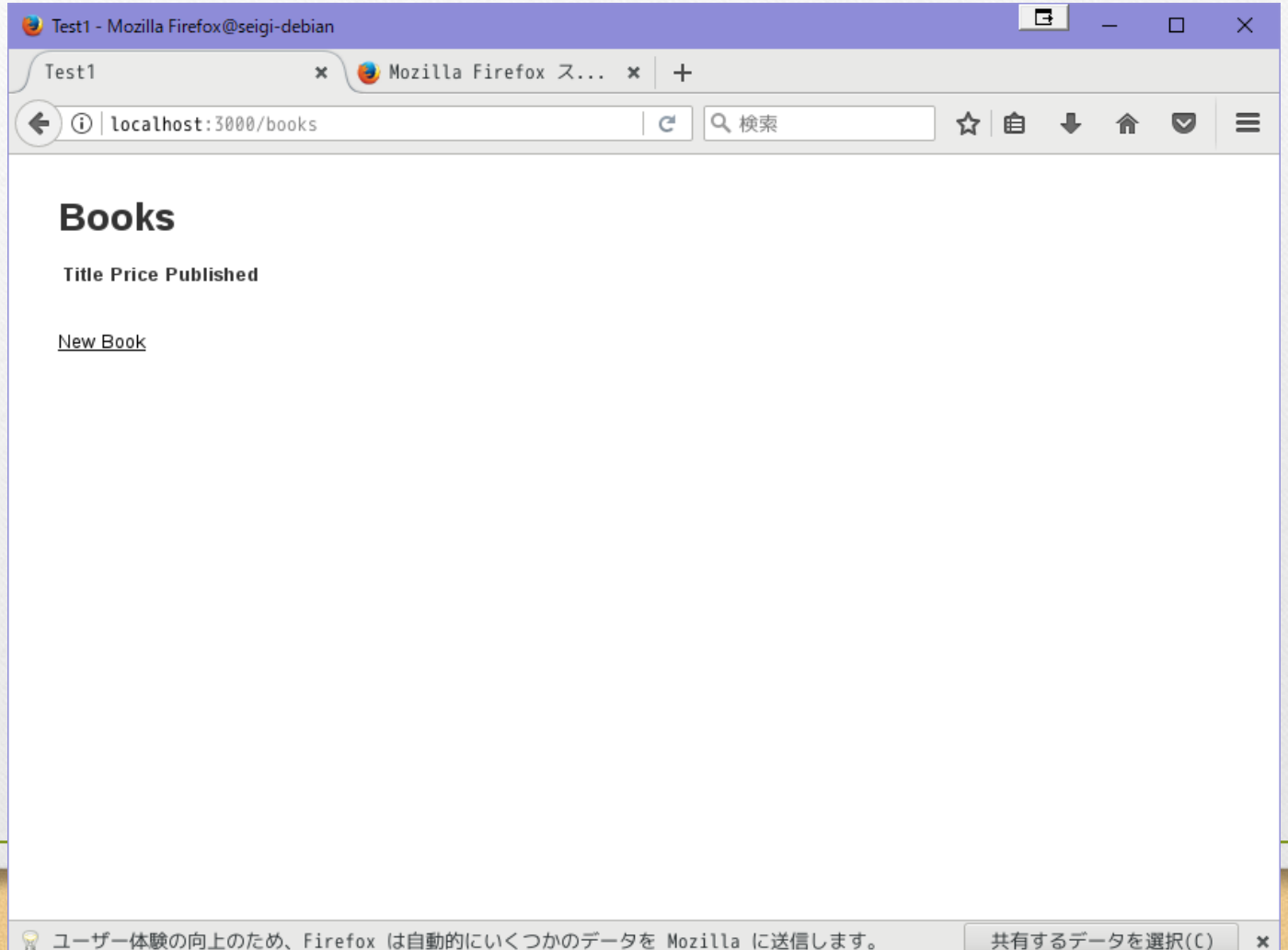
  $ rake db:migrate

- サーバ起動, ブラウザ確認

  $ rails s

  ブラウザで http://localhost:3000/books にアクセス

ブラウザでアクセス


Test1 - Mozilla Firefox@seigi-debian
Test1
Mozilla Firefox ス...
localhost:3000/books
検索
Books
Title Price Published
New Book
ユーザー体験の向上のため、Firefox は自動的にいくつかのデータを Mozilla に送信します。
共有するデータを選択(C)

手を動かしてみる

- 書籍の登録や編集などをやってみよう.

Controller

- app/controllers/books_controller.rb

```ruby
class BooksController < ApplicationController
  before_action :set_book, only: [:show, :edit, :update, :destroy]

  # GET /books
  # GET /books.json
  def index
    @books = Book.all
  end

  # GET /books/1
  # GET /books/1.json
  def show
  end

  # GET /books/new
  def new
    @book = Book.new
  end

  # GET /books/1/edit
  def edit
  end
```

index アクション
- all メソッドを使ってデータベースからデータを取得

new アクション
- new メソッドを使ってデータの格納先変数を作成

Book クラスは DB とやり取りできるオブジェクト
Model によって作成されている
(Model とのやり取り部分)

Book テーブル

Title	Price	Published
Book1	3000	XX corp.
Book2	2500	YY corp.

Title	Price	Published
Book1	3000	XX corp.
Book2	2500	YY corp.

テーブルにデータ格納場所を作成

Controller

- app/controllers/books_controller.rb

```ruby
class BooksController < ApplicationController
  before_action :set_book, only: [:show, :edit, :update, :destroy]

  # GET /books
  # GET /books.json
  def index
    @books = Book.all
  end

  # GET /books/1
  # GET /books/1.json
  def show
  end

  # GET /books/new
  def new
    @book = Book.new
  end

  # GET /books/1/edit
  def edit
  end
```

show, edit アクション
- 何も書いていない？

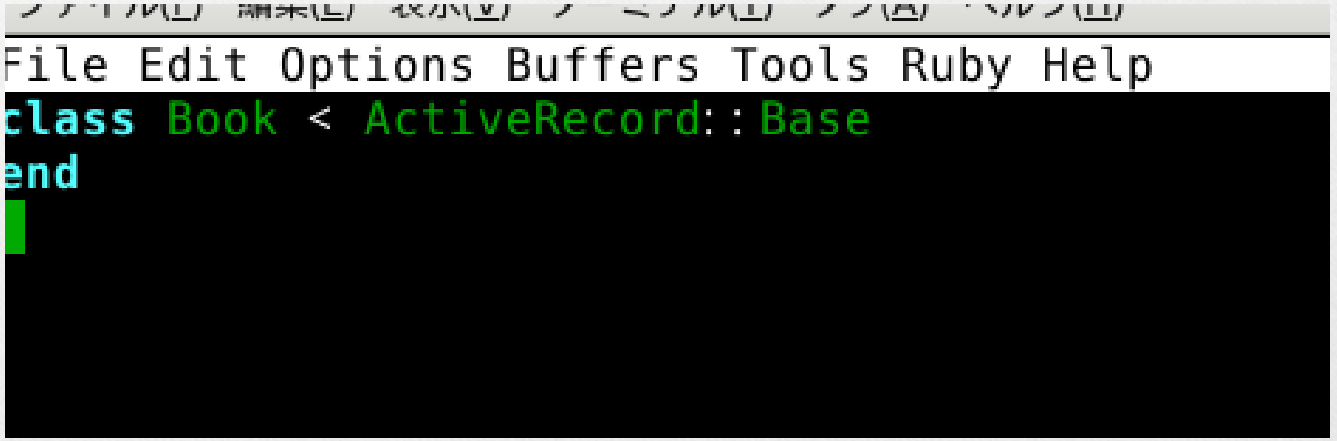
DRY(Don't Repeat Yourself)
- 同じことを繰り返さない
- 定義などの作業は一回だけで済ませろ

↓ 一番下

```ruby
  private
    # Use callbacks to share common setup or constraints between actions.
    def set_book
      @book = Book.find(params[:id])
    end
```

Model

- app/models/book.rb
  - ほとんど何も書いていない…
    - Book というクラス (Ruby) オブジェクトとBook というデータベース (db) を繋げている



  - やりたいことなどの仕様はだいたいここに書き足していき, Controller では呼び出すだけにするのが望ましいらしい
- 今日は詳しくは語らない

ここまででできること

- 書籍の
  - 一覧表示(index)
  - 詳細表示(show)
  - 新規作成(new)
  - 登録(create)
  - 編集(edit)
  - 更新(update)
  - 削除(destroy)

Scaffold 機能だけでこれだけのことができる

充実したアプリケーションの開発のために本当はもう少し語りたいこと

- アクションの追加
- Model によるパラメータのvalidation
- プラグイン
- javascript
- ルーティング
  - RESTful なルーティング
- Apache にRails アプリケーションをのせる
  - Passenger
- 複数のデータの関連付け(アソシエーション)

今日は語らない

現在稼働中のサービスについての紹介

惑星宇宙グループで稼働しているRails サービス

- INEX レポート投稿システム -suu-
  - 管理者: 村橋
- PSG 施設予約システム -prrs-
  - 管理者: 荻原

- 全てjet サーバで動いています
  - https://jet.ep.sci.hokudai.ac.jp/

INEX レポート投稿システム suu

- INEX課題提出用レポート投稿システム
  - 2013/04/01- 稼働開始
- Ruby on Rails のバージョン: 4.2.1
- 管理者 : 村橋(4 代目)
- 機能
  - レポート投稿・更新・コメント付与
  - ファイルアップロード

PSG 施設予約システム Prrs

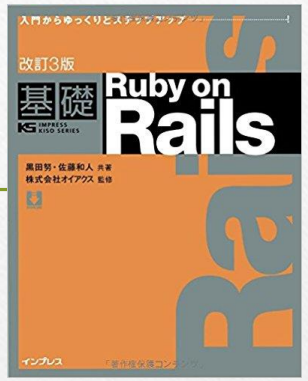
- 惑星宇宙グループの施設予約管理システム
  - 2016/03/05- 稼働開始
- Ruby on Rails のバージョン: 4.2.5.1
- 管理者 : 荻原(2 代目)
- 機能
  - 施設ごとの予約管理
  - 要望等のコメント掲示板みたいなもの

まとめ

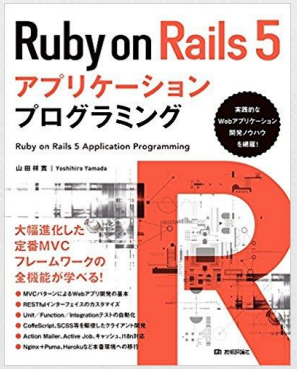
- Ruby on Rails の基本の基本について解説・実際に手を動かした
  - 簡単にデータベースを用いたウェブアプリケーションを作れる Web アプリケーションフレームワーク
  - MVC
  - DRY, CoC
- EPnetFaN のウェブアプリケーションのほとんどが Rails で提供されている
  - 動いているものもあれば止まっているものもある
    - できるなら皆さんで再開してください

今から始める人へおすすめの文献

- おすすめ図書
  - 改訂3版基礎 Ruby on Rails (KS IMPRESS KISO SERIES) , 黒田 勉, 佐藤和人 (2015)
    - 古いが私が勉強した本. Ruby から詳しく載っておりアプリを作りながら教える内容なので初心者におすすめ
  - Ruby on Rails 5 アプリケーションプログラミング, 山田祥寛, 技術 評論社 (2017)
    - 比較的新しいので最新の情報も載っている分かって読めばより分かる
- おすすめ Web ページ
  - Ruby on Rails チュートリアル, Michae Hartl , (2018/01/19 訪問時), http://railstutorial.jp/



https://images-na.ssl-images-amazon.com/images/I/41AkxXsz%2B5L._SX391_BO1,204,203,200_.jpg



https://images-na.ssl-images-amazon.com/images/I/51vycwIasvL._SX392_BO1,204,203,200_.jpg

参考文献

- 黒田 勉, 佐藤和人, 2013, 改訂新版基礎 Ruby on Rails, インプレス
- 山田祥寛，2014, Ruby on Rails 4 アプリケーションプログラミング，技術 評論社
- **PSG 施設予約システム, 2018/01/19 アクセス, https://jet.ep.sci.hokudai.ac.jp/prrs**
- Ruby on Rails | A web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern.**, 2018/01/19 アクセス**, http://rubyonrails.org/
- **Ruby on Rails チュートリアル 実例を使ってRailsを学ぼう, Michael Hartl, 2018/01/19 アクセス, https://railstutorial.jp/**
- **Ruby on Rails の事例まとめ（日本有名サイト編）, 吉田光利, 2018/01/19 アクセス, http://skillhub.jp/blogs/177**
- **Ruby on Rails をほんの少し語ってみる, 三上峻. 2016, https://www.ep.sci.hokudai.ac.jp/~epnetfan/zagaku/2016/0722/pub/**
- **Welcome to JET sever, 2018/01/19 アクセス, https://jet.ep.sci.hokudai.ac.jp/**
- **フレームワーク Wikipedia, 2017, https://ja.wikipedia.org/wiki/%E3%83%95%E3%83%AC%E3%83%BC%E3%83%A0%E3%83%AF%E3%83%BC%E3%82%AF**