

Ruby on Rails 講義

第24回 道具箱

五十嵐邦明

twitter : igaiga555

<http://www.facebook.com/kuniaki.igarashi>



2013.12.5 at 一橋大学
ニフティ株式会社寄附講義
社会科学における情報技術と
コンテンツ作成IV

五十嵐邦明

講師

株式会社 spice life



twitter: igaiga555

<https://github.com/igaiga/>

<http://www.facebook.com/kuniaki.igarashi>

濱崎 健吾

Teaching Assistant
fluxflex, inc(米国法人)



twitter: hmsk

<https://github.com/hmsk/>

<http://www.facebook.com/hamachang>

コードを 書くための 道具箱



バージョン管理システム



thanks to ryopeko, the photo is <http://www.flickr.com/photos/martindale/59445824/>

こんなことありませんか？

あれ、コードを書き換えたら
動かなくなっちゃった・・・

動いてたときへ戻したいけど
分かなくなっちゃった・・・

そんなときに役に立つのが
バージョン管理システム
です。

もしも、こんな風に履歴が残っていて

■ book.rb 2012.12.15 14:10
authorメソッド作った

■ book.rb 2012.12.15 14:34
メモ機能を追加した

■ book.rb 2012.12.15 15:02
現在。素晴らしい機能を実装中。

時間の流れ

うまく動かなくなっちゃった・・・ってときに

■ book.rb 2012.12.15 14:10

authorメソッド作った

■ book.rb 2012.12.15 14:34

メモ機能を追加した

■ book.rb 2012.12.15 15:05

現在。素晴らしい機能は失敗に終わった。

時間の流れ

過去のある時点へ戻れたら便利ですよね？

book.rb 2012.12.15 14:10
authorメソッド作った

book.rb 2012.12.15 14:34
メモ機能を追加した

book.rb 2012.12.15 15:05
現在。素晴らしい機能は失敗に終わった。

ここへ
戻りたい

時間の流れ

バージョン管理システム
なら、できちゃいます！

※バージョン管理システムはざっくり言うとゲームのセーブポイントです。

では、使い方を見ていきましょう。

バージョン管理システム
にはいくつか種類があるのでですが、
今回は **git** という
バージョン管理システムを使います。

Rubyistがよく使うバージョン管理システムです。
Linuxを作ったLinusさんらが作者です。

※gitとRubyとは特に関係はありません。
独立して別々に使われます。

git インストール確認

git はshellでコマンドとして使います。

Win, Mac

RailsInstaller でインストールした人はすでにgitがインストールされています。

shell で以下のコマンドを打ってバージョンが表示されればインストール済みです。

\$ git --version

git version 1.7.5.2

※1.7.5.2の部分は異なります。

※Macの人でインストールされていない場合は、
以下などからDLしてインストールしてください。

<http://code.google.com/p/git-osx-installer/>

git 設定確認

git は使用者の名前とメールアドレスを履歴に残します。
以下のコマンドで設定を確認できます。

\$git config --global user.name

Kuniaki IGARASHI

\$git config --global user.email

igaiga@gmail.com

設定ができない場合は以下で設定できます。

git config --global user.name "Kuniaki IGARASHI"

git config --global user.email igaiga@gmail.com

※これらの情報はPC内に保存されます。使い方によっては公開されることもあるので、心配な場合は仮のものでも大丈夫です。

git 初期化

git はフォルダ単位で履歴（バージョン）を管理します。

試しに、新しいフォルダを作つてそこをgitでバージョン管理してみましょう。

前準備

```
$ mkdir フォルダ名
```

```
$ cd フォルダ名
```

フォルダを新規作成します。
作ったフォルダへ移動します。

git 初期化

```
$ git init
```

Initialized empty Git repository in フォルダ名

git init すると、そのフォルダをバージョン管理できるようになります。

※豆知識：バージョン管理されている場所を「リポジトリ」と呼びます。

履歴のポイントを作る

歴史に残したい状態でポイントを作ります。
このポイントを**コミット**と呼びます。

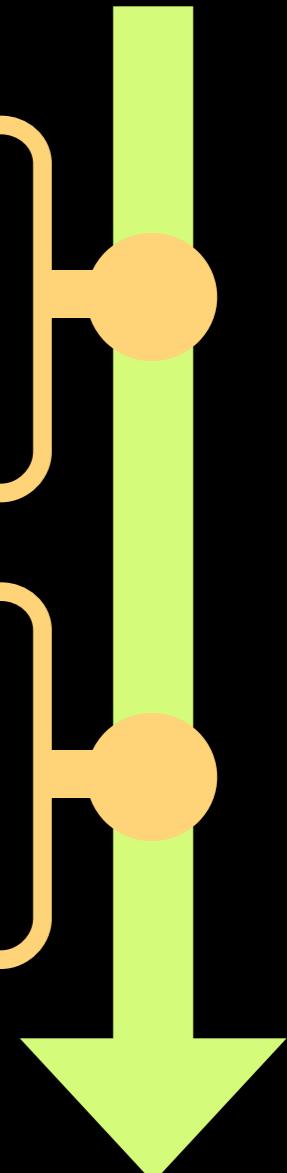
さきほどの図で言う●の部分です。

book.rb 2012.12.15 14:10

authorメソッド作った

book.rb 2012.12.15 14:34

メモ機能を追加した



コミットの方法

コミットしてみましょう。

前準備

エディタでファイルを作って、git init したフォルダへ保存しましょう。

hello.rb

puts "Hello world!"

ファイル名もファイルの内容も
自由でかまいません。
RubyのコードでなくてもOKです。

git コミット

\$ git add hello.rb

\$ git commit -m "hello.rb を追加"

hello.rb の部分はファイル名
"hello.rb を追加"の部分は
コメントです。自由に書けます。

```
[master (root-commit) ebeb9df] hello.rb を追加
```

```
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 hello.rb
```

コミットするには、git add した後で、git commit します。

履歴を見る

履歴を見てみましょう。先ほど作ったコミットがあるはずです。

履歴を見る

\$ git log

```
commit ebeb9df1625ef74856a2724cca54e87abf72dfbc
Author: Kuniaki IGARASHI <igaiga@gmail.com>
Date:   Sat Dec 15 21:44:22 2012 +0900
```

hello.rb を追加

さきほどのコミットがありました。
なにやら色々と書いてあります。（次のページへ続く）

履歴の見方

```
$ git log
```

```
commit ebcb9df1625ef74856a2724cca54e87abf72dfbc ←コミットハッシュ  
Author: Kuniaki IGARASHI <igaiga@gmail.com> ←コミットした人  
Date: Sat Dec 15 21:44:22 2012 +0900 ←コミットした日時
```

```
hello.rb を追加 ←コミットメッセージ
```

各種情報が記録されています。

コミットハッシュはそのコミットを一意に示す英数字です。
このコミットの名前と言ってもいいでしょう。
このコミットを指すときは ebcb9d... と指定します。

もう1つコミットを作ってみましょう

前準備

エディタでさっきのファイルを変更して保存しましょう。

hello.rb

```
puts "foo"
```

git コミット

```
$ git add hello.rb
```

```
$ git commit -m "puts文を変更"
```

hello.rb の部分はファイル名
"puts文を変更"の部分は
コメントです。自由に書けます。

```
[master 6ba048b] puts文を変更
```

```
1 files changed, 1 insertions(+), 1 deletions(-)
```

コミットしたらgit log を見てみましょう。(つづく)

履歴を確認してみましょう

\$ git log

```
commit 6ba048b2967d1c04f271e12988e43f19f3f65def ←新しいコミット  
Author: Kuniaki IGARASHI <igaiga@gmail.com>  
Date: Sun Dec 16 09:56:42 2012 +0900  
が増えました
```

puts文を変更

```
commit ebeb9df1625ef74856a2724cca54e87abf72dfbc  
Author: Kuniaki IGARASHI <igaiga@gmail.com>  
Date: Sat Dec 15 21:44:22 2012 +0900
```

hello.rb を追加

このようにコミットが積み上がっていきます。
変更の内容をもう少し詳しく見るにはどうすればいいでしょうか。 (つづく)

履歴を詳しく見てみましょう

\$ git log -p

```
commit 6ba048b2967d1c04f271e12988e43f19f3f65def
Author: Kuniaki IGARASHI <iigaiga@gmail.com>
Date:   Sun Dec 16 09:56:42 2012 +0900
```

puts文を変更

```
diff --git a/hello.rb b/hello.rb  ↪hello.rb が変更された
index 08b85bf..a32c710 100644
--- a/hello.rb
+++ b/hello.rb
@@ -1 +1 @@
-puts "Hello world!" ← - から始まる行は削除されたことを意味します。
+puts "foo"           ← + から始まる行は追加されたことを意味します。
```

```
commit ebeb9df1625ef74856a2724cca54e87abf72dfbc
Author: Kuniaki IGARASHI <iigaiga@gmail.com>
Date:   Sat Dec 15 21:44:22 2012 +0900
```

コミットで変わった差分(diff)が表示されます。何を変えたか調べたいときに便利です。
また、以下の設定文を打つと結果がカラーになって見易いです。(1回だけ打てばずっと有効)

\$git config --global color.ui true

gitの現在状態を確認する

git 状態確認

```
$ git status
```

ファイルに変更がないとき

```
# On branch master  
nothing to commit (working directory clean)
```

ファイルが変更されているとき

```
# On branch master  
# Changes not staged for commit:  
#   (use "git add <file>..." to update what will be committed)  
#   (use "git checkout -- <file>..." to discard changes in  
working directory)  
#  
#       modified:   hello.rb  ←変更されているファイル名  
#  
no changes added to commit (use "git add" and/or "git commit -a")
```

gitの現在状態を確認する(つづき)

git 状態確認

```
$ git status
```

過去にコミットしていないファイルが存在するとき

```
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       iga.rb ←過去にコミットされていないけど存在してるファイル名
nothing added to commit but untracked files present (use "git add"
to track)
```

歴史を巻き戻す

過去のコミットまで戻してみましょう。

\$ git checkout [コミットハッシュ]

コミットハッシュは
git log で調べておきます。

例) **\$ git checkout ebbe9d**

コミットハッシュは全部打たなくとも
一意に決まる長さで大丈夫です。

Note: checking out 'ebbe9d'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b new_branch_name
```

HEAD is now at ebbe9df... hello.rb を追加

hello.rb をエディタで開くと、ebbe9d 時点の状態に戻ってることが分かります。
puts "Hello world!"

巻き戻した歴史を元に戻す

元に戻すには `checkout` で `master` を指定します。
※`hello.rb`のファイルも最新に戻るので、必要ならば別の名前でコピーしておきます。

```
$ git checkout master
```

```
Previous HEAD position was ebeb9df... hello.rb を追加  
Switched to branch 'master'
```

`hello.rb` をエディタで開くと、最新のコミット時点の状態に戻っています。

```
puts "foo"
```

參考：git 練習場 <http://try.github.com>



1.1 • Got 15 minutes and want to learn Git?

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

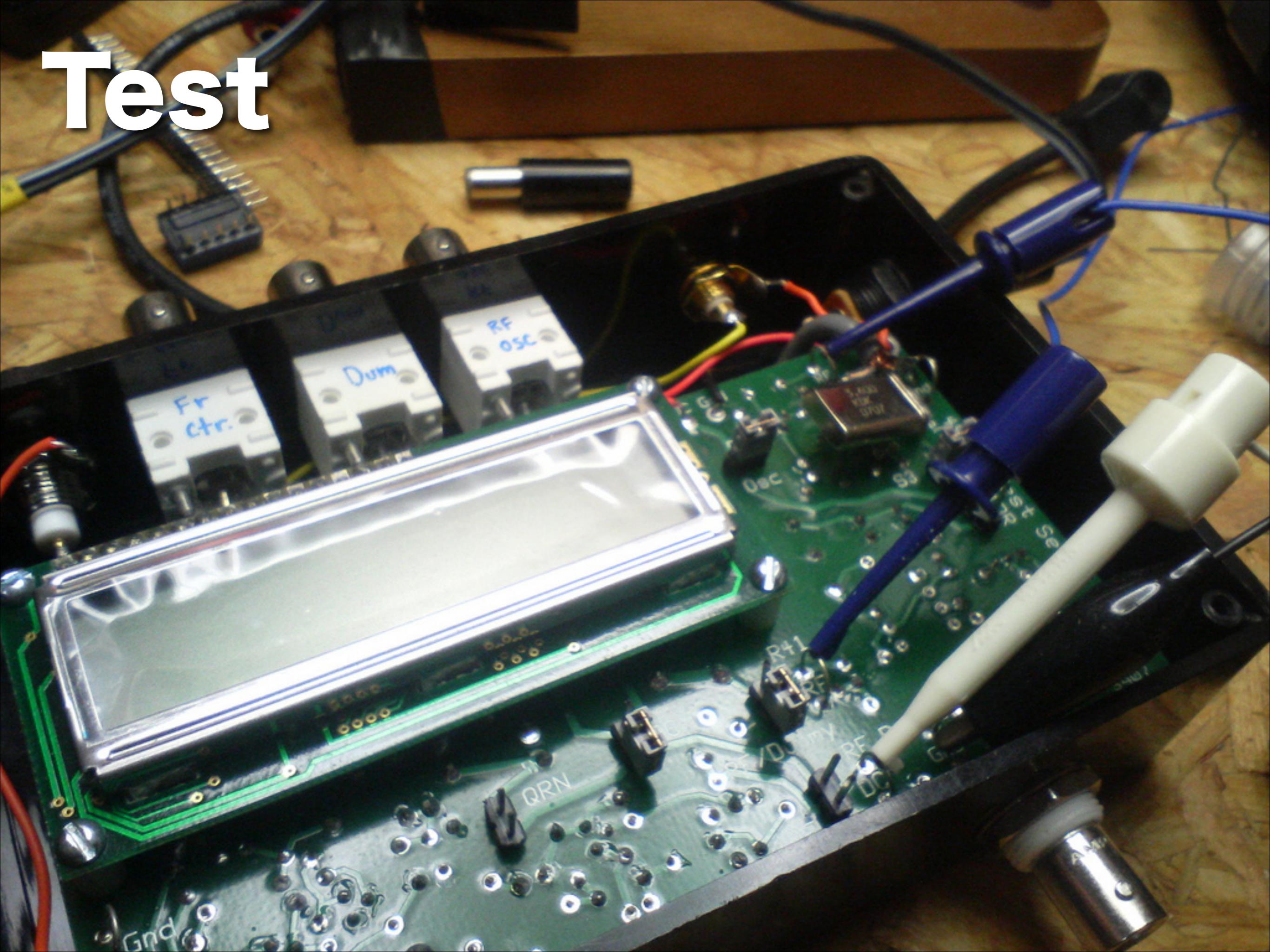
Our terminal prompt below is currently in an **octobox** directory. To initialize a Git repository here, type the following command:



➔ **git init**

```
$
```

Test



ぶっちゃけ、ブラウザからぽちぽち
動作確認するの、面倒なんですよね。
`rails s` も打たないといけないし。

そこで、動作確認(Test)もプログラム
にやらせてしまうことにしてます。
開発を速く巧く進める技です。

RSpec

プログラムでテストを書く仕組みの1つがRSpecです。
RSpecはGemになっているので簡単に使い始めることができます。

RailsアプリでRSpecを書く方法を見てみましょう。
例によっていつものアプリを作ります。
(既にできているアプリを使ってもOK)

前準備

```
$ rails new books_app
```

```
$ cd books_app
```

```
$ bundle exec rails g scaffold books title:string  
memo:text
```

```
$ bundle exec rake db:migrate
```

次の行も加えて実行してください。 (既にあるアプリを使う場合もこれだけ実行してください。)

```
$ bundle exec rake db:migrate RAILS_ENV=test
```

RSpec インストール

Gemfileに以下を追加します。
(たとえばsqlite3の下)

```
gem 'rspec-rails'
```

bundle install コマンドで変更したGemfileを反映させます。

```
$ bundle
```

(install は省略可能)

Rspecをインストールします。
shellで以下のコマンドを打ちます。

```
$ bundle exec rails g rspec:install
```

RSpec プロダクトコード実装

たとえばBookモデルが以下のようになっていて、`deco_title`というタイトルをデコる素敵なメソッドを実装していたとしましょう。

`app/models/book.rb`

```
class Book < ActiveRecord::Base
  def deco_title
    "*** " + title + " ***"
  end
end
```

たとえば、`book = Book.new(title: "3月のライオン")` のとき
`book.deco_title` と実行して `"*** 3月のライオン ***"` を得たいわけです。
テストなしはどううまく動くか不安で夜も眠れません。
テストコードを書いてみましょう！

※このBookクラスのように、アプリで実際の動作に使われるコードをプロダクトコードと読んだりもします。私は本番コードと読んじゅうことが多いです。

RSpec テストコード実装

テストコードは以下のように書けます。

spec/models/book_spec.rb

```
# -*- coding: utf-8 -*-
```

```
require 'spec_helper'
```

```
describe "Book" do ←Bookクラスの
```

```
  describe "#deco_title" do ←deco_titleメソッドについて(#はインスタンスメソッドの意)
```

```
    it "*** *** で囲まれたタイトルを返す" do ←こう動作すべき
```

```
      book = Book.new(:title => "3月のライオン") ←前準備
```

```
      book.deco_title.should == "*** 3月のライオン ***" ←テストコード コア部分
```

```
    end
```

```
  end
```

```
end
```

ポイントはこのコードです。 "should" がテストのためのメソッド。

book.deco_title.should == "* 3月のライオン ***"**

「book.deco_title は "*** 3月のライオン ***" と == (同じ) になるべき」
というテストコードです。

book.deco_title の結果が "*** 3月のライオン ***" の場合にテストを通過します。
異なる場合やうまく動かない場合はテストが失敗します。

RSpec テストコード実行

さきほど書いたテストコードを実行してみましょう。
実行には rspec コマンドを使い、
実行したいテストコードを指定します。

```
$ rspec spec/models/book_spec.rb
```

```
$ rspec spec/models/book_spec.rb
.
```

```
Finished in 0.00891 seconds
1 example, 0 failures
```

```
Randomized with seed 63328
```

テストが無事に通過すると、
こんな感じのスッキリしたグリーンの表示になります。

RSpec テストコード実行

```
$ rspec spec/models/book_spec.rb
```

```
$ rspec spec/models/book_spec.rb
```

```
F
```

```
*
```

```
Failures:
```

```
1) Book #deco_title *** *** で囲まれたタイトルを返す
```

```
Failure/Error: book.deco_title.should == "*** 3月のライオン ***"
```

```
TypeError:
```

```
  no implicit conversion of nil into String
```

```
# ./app/models/book.rb:3:in `+'
```

```
# ./app/models/book.rb:3:in `deco_title'
```

```
# ./spec/models/book_spec.rb:9:in `block (3 levels) in <top (required)>'
```

```
Finished in 0.00597 seconds
```

```
1 example, 1 failure
```

```
Failed examples:
```

```
rspec ./spec/models/book_spec.rb:6 # Book #deco_title *** *** で囲まれたタイトルを返す
```

```
Randomized with seed 46263
```

なにか問題があるとこんな感じの残念なレッドの表示になります。
プロダクトコード、もしくはテストコードを見直しましょう。

Cosmo

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#) [Forms](#)

Cosmo

An ode to Metro.

[Preview](#)

[Download](#)



Twitter Bootstrap

[Bootswatch](#) [News](#) [Gallery](#)

Readable

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#)

Readable

Optimized for legibility.

[Preview](#)

[Download](#)



Cyborg

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#) [Forms](#)

Cyborg

Jet black and electric blue.

[Preview](#)

[Download](#)



Bootswatch

[Bootswatch](#) [News](#) [Gallery](#) [Preview](#)

Simplex

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#) [Forms](#) [Misc](#)

Simplex

Minimal and minimalist.

[Preview](#)

[Download](#)



Journal

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#) [Forms](#)

Journal

Crisp like a new sheet of paper.

[Preview](#)

[Download](#)



Bootswatch

[Bootswatch](#) [News](#) [Gallery](#) [Preview](#)

Slate

A preview of changes in this swatch.

[Typography](#) [Navbar](#) [Buttons](#) [Forms](#) [Misc](#)

Slate

Shades of gunmetal gray.

[Preview](#)

[Download](#)



Twitter Bootstrapは
なんかええ感じの
デザインにしてくれる
お手軽ツールです。

前準備

例によっていつものscaffoldで作るアプリを題材にします。
前に作ったものがあればそれをそのまま使ってもOKです。

```
$ rails new books_app
```

```
$ cd books_app
```

```
$ bundle exec rails g scaffold books  
title:string memo:text
```

```
$ bundle exec rake db:migrate
```

Listing books

Title

Memo

路地恋花 京都の路地を舞台にした恋愛短編集 [Show](#) [Edit](#) [Destroy](#)

ちはやふる 高校の競技カルタ部を描く火花散らす青春 [Show](#) [Edit](#) [Destroy](#)

[New Book](#)

地味～

New book

Title

路地恋花

Memo

京都の路地を舞台にした恋愛短編集

地味～

Create Book

Back

そこで Twitter Bootstrap

<http://twitter.github.com/bootstrap/>

Bootstrap

Sleek, intuitive, and powerful front-end framework for faster and easier web development.

[Download Bootstrap](#)[GitHub project](#) [Examples](#) [Extend](#) [Version 2.2.2](#)

まずは
DLします。

[Follow @twbootstrap](#)

45.3K followers



17K

Introducing Bootstrap.

Need reasons to love Bootstrap? Look no further.



By nerds, for nerds.

Made for everyone.

Packed with features.

ファイルコピー

DLしたbootstrap.zipを解凍して、
css/bootstrap.css
を適用したいアプリの
app/assets/stylesheets/
フォルダへコピーします。

コード修正

app/views/layouts/application.html.erb

<%= stylesheet_link_tag "application", media: "all", "data-turbolinks-track" => true %>
の上に以下の行を追加します。

```
<%= stylesheet_link_tag "bootstrap", media: "all" %>
```

<%= yield %> を以下で書き換えます。

```
<div class="container">  
  <%= yield %>  
</div>
```

コード修正

app/views/layouts/application.html.erb

<body> の次の行に以下を追加します。 (上部メニューバーになります)

```
<nav class="navbar navbar-default" role="navigation">
<div class="navbar-header">
  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-navbar-1">
    </button>
  <a class="navbar-brand" href="#">The Books App</a>
</div>

<div class="collapse navbar-collapse" id="bs-navbar-1">
  <ul class="nav navbar-nav">
    <li class="active"><a href="#">Link</a></li>
    <li><a href="#">Link</a></li>
  </ul>
</div>
</nav>
```

コード修正

app/views/layouts/application.html.erb

</body> の前の行に以下を追加します。(下部フッターになります)

```
<footer><div class="container">  
Hitotsubashi Univ.  
</div></footer>
```

コード修正

app/assets/stylesheets/application.css

ファイルの一番最後に以下を追加します。

...
*= require_tree .
*/

の下です。

```
footer { margin-top:100px; }  
table,td,th { vertical-align: middle !important; border: none !important; }  
th { border-bottom: 1px solid #DDD !important; }
```

コード 修正後

app/views/layouts/application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>BooksApp</title>
  <%= stylesheet_link_tag "bootstrap", media: "all" %>
  <%= stylesheet_link_tag "application", media: "all", "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>
</head>
<body>
  <nav class="navbar navbar-default" role="navigation">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-navbar-1">
        </button>
      <a class="navbar-brand" href="#">The Books App</a>
    </div>

    <div class="collapse navbar-collapse" id="bs-navbar-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link</a></li>
        <li><a href="#">Link</a></li>
      </ul>
    </div>
  </nav>

  <div class="container">
    <%= yield %>
  </div>

  <footer><div class="container">
    Hitotsubashi Univ.
  </div></footer>
</body>
</html>
```

コード 修正後

app/assets/stylesheets/application.css

```
/*
 * This is a manifest file that'll be compiled into application.css, which will include all the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of the
 * compiled file, but it's generally better to create a new file per style scope.
 *
 *= require_self
 *= require_tree .
 */
footer { margin-top:100px; }
table,td,th { vertical-align: middle !important; border: none !important; }
th { border-bottom: 1px solid #DDD !important; }
```

Listing books

Title	Memo	
路地恋花	京都の路地を舞台にした恋愛短編集	Show Edit Destroy
ちはやふる	高校の競技かるた部を描く火花散らす青春	Show Edit Destroy

[New Book](#)

ちょっと
いい感じ～

New book

Title

Memo

Create Book

Back

Hitotsubashi Univ.

ちょっと
いい感じ～

Twitter Bootstrap More

もっと使ってみたい方は、
ドットインストールの解説動画がお勧めです。

http://dotinstall.com/lessons/basic_twitter_bootstrap_v3

自由制作演習

自由に作るものを作り、
作って発表する演習です。

要件

- ▶ Rails または Ruby のコードを書いてください
- ▶ (Railsを使わずに)Ruby だけでもOKです
- ▶ 発表時間は5分間(それより短ければOK)
- ▶ 発表には以下を必ずいれてください。
- ▶ 動機：なぜそのプログラムを作ろうと思ったか
- ▶ プログラムのデモ

- ▶ 発表後に発表資料とコード群を提出してください
- ▶ 発表しない場合、評価は原則として不可とします
- ▶ なので、発表日に出席できない場合は五十嵐へ相談してください

※ルイス先生の講義を受講している方は、その一部をここで作ってもらってもかまわないです。別のものを作ってもOKです。

スケジュール

12/19, 26, 1/9 制作期間

出席はしてもしなくてもOKです(出席扱い)

五十嵐は講義時間中は部屋にいるので、
質問、相談など気軽にどうぞ

1/16 発表日(最終講義日)

8:50から開始するので朝がんばって来て！

発表資料とコード群の提出方法はこの日に連絡します

今までの道具のおさらい Ruby編

- ▶ **Excel, csv ファイルの読み込み**
 - ▶ spreadsheet gem (資料第12回 Gem)
- ▶ **twitter データの収集**
 - ▶ twitter gem (資料第12回 Gem)
- ▶ **Sinatra (Webフレームワーク)**
 - ▶ sinatra gem (資料第13回Sinatraで作る簡単Webアプリ)
- ▶ **iPhoneへPush通知**
 - ▶ おまけ(講義資料ページに置いてあります)

今までの道具のおさらい

Rails編

▶ 一番小さなRailsアプリ(資料第18回)

▶ Web画面を作つて情報を表示するときに

▶ CSS & HTML (資料第19回)

▶ デザインをかっこよくしたいときに

▶ CRUD (資料第20回)

▶ 新規作成、表示、更新、削除の4つの基本動作

▶ new & create (資料第21回)

▶ 新規作成について詳しく

▶ model (資料第22回)

▶ データを保存したいときに

▶ 画像投稿アプリ(資料第23回)

▶ 画像を投稿できるようにする、gemの使い方、Bundler

どんなものつくればいいのかのヒント

- ▶ 自分の生活を少し便利にするもの
 - ▶ 情報が見れるWebアプリ
 - ▶ スマートフォンや携帯へ情報発信
- ▶ データを収集したり解析したり
 - ▶ 例)政府統計
 - ▶ <http://www.e-stat.go.jp/estat/html/GL02100101.html>
 - ▶ 例)twitter などのWeb上の情報

「こんなものを作りたいかも」
みたいな ふわっ とした質問でも、
話してたら案が出てくることも。
なので気軽に相談してください。

発表

発表は受講者と講師による
採点投票方式にします。

評価軸は以下の2つです。

- ・エンターテイメント点（面白さ）
- ・学術点（興味深さ、意義、技術力）

それぞれ1位の人を褒め称えます。

付録

デバッグの方法

- ▶ ブラウザに表示されるエラーを読む
- ▶ rails server の出力を確認
- ▶ ログを確認
- ▶ ログを出力
- ▶ pryを使う
- ▶ Chrome のデバッグ機能を使う

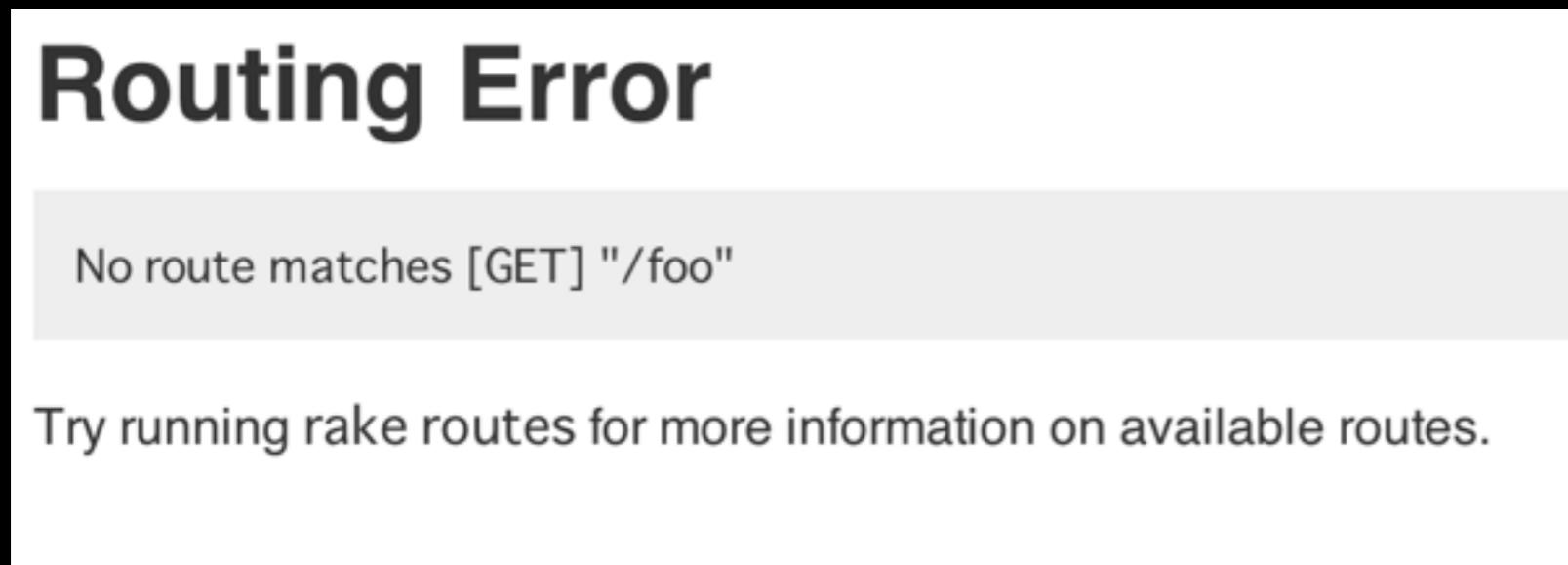
ブラウザに表示されるエラーを読む

エラー例：rails server が起動していない



エラー例：そんなURL知らない

Routing Error



ブラウザに表示されるエラーを読む

NameError in BooksController#index

undefined local variable or method `bar' for #<BooksController:0x007f94ddcccd28>

Rails.root: /Users/igarashi/work/scaffold1115/books_app

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

app/controllers/books_controller.rb:5:in `index'

Request

Parameters:

None

[Show session dump](#)

[Show env dump](#)

Response

Headers:

None

エラー例：

**app/controllers/
books_controller.rb**

の5行目で bar っていう知らないものがあるのでエラーだよ。

rails server の出力を確認

rails s を実行した shell を見ると、アクセスのたびに何かが
出力されていることが分かります。

```
$ bundle exec rails s
=> Booting WEBrick
=> Rails 3.2.9 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2012-11-17 13:03:00] INFO  WEBrick 1.3.1
[2012-11-17 13:03:00] INFO  ruby 1.9.3 (2012-04-20) [x86_64-darwin11.3.0]
[2012-11-17 13:03:00] INFO  WEBrick::HTTPServer#start: pid=9602 port=3000
```

```
Started GET "/books" for 127.0.0.1 at 2012-11-17 13:03:05 +0900
```

```
Connecting to database specified by database.yml
```

```
Processing by BooksController#index as HTML
```

```
Book Load (0.4ms)  SELECT "books".* FROM "books"
```

```
Rendered books/index.html.erb within layouts/application (3.6ms)
```

```
Completed 200 OK in 76ms (Views: 37.8ms | ActiveRecord: 2.1ms)
```

```
Started GET "/assets/books.css?body=1" for 127.0.0.1 at 2012-11-17 13:03:06 +0900
```

```
0
```

```
Served asset /books.css - 304 Not Modified (1ms)
```

```
[2012-11-17 13:03:06] WARN Could not determine content-length of response body.
```

```
Set content-length of the response or set Response#chunked = true
```

```
Started GET "/assets/scaffolds.css?body=1" for 127.0.0.1 at 2012-11-17 13:03:06 +0900
```

```
0
```

rails server の出力を確認

エラーが起こるとその旨も表示されます。

エラー例: /foo にアクセスしたけど、そんなURL知らない

```
Started GET "/foo" for 127.0.0.1 at 2012-11-17 13:16:44 +0900
```

```
ActionController::RoutingError (No route matches [GET] "/foo"):  
  actionpack (3.2.9) lib/action_dispatch/middleware/debug_exceptions.rb:21:in `call'  
  actionpack (3.2.9) lib/action_dispatch/middleware/show_exceptions.rb:56:in `call'  
  railties (3.2.9) lib/rails/rack/logger.rb:32:in `call_app'  
  railties (3.2.9) lib/rails/rack/logger.rb:16:in `block in call'  
  activesupport (3.2.9) lib/active_support/tagged_logging.rb:22:in `tagged'  
  railties (3.2.9) lib/rails/rack/logger.rb:16:in `call'  
  actionpack (3.2.9) lib/action_dispatch/middleware/request_id.rb:22:in `call'  
  rack (1.4.1) lib/rack/methodoverride.rb:21:in `call'  
  rack (1.4.1) lib/rack/runtime.rb:17:in `call'  
  activesupport (3.2.9) lib/active_support/cache/strategy/local_cache.rb:72:in `call'  
  rack (1.4.1) lib/rack/lock.rb:15:in `call'  
  actionpack (3.2.9) lib/action_dispatch/middleware/static.rb:62:in `call'  
  railties (3.2.9) lib/rails/engine.rb:479:in `call'  
  railties (3.2.9) lib/rails/application.rb:223:in `call'  
  rack (1.4.1) lib/rack/content_length.rb:14:in `call'  
  railties (3.2.9) lib/rails/rack/log_tailer.rb:17:in `call'  
  rack (1.4.1) lib/rack/handler/webrick.rb:59:in `service'  
  /Users/igarashi/.rvm/rubies/ruby-1.9.3-p194/lib/ruby/1.9.1/webrick/httpserver.  
rb:138:in `service'
```

rails server の出力を確認

エラーが起こるとその旨も表示されます。

**app/controllers/books_controller.rb の5行目で
bar っていう知らないものがあるのでエラーだよ。**

```
Started GET "/books" for 127.0.0.1 at 2012-11-17 13:21:10 +0900
Connecting to database specified by database.yml
Processing by BooksController#index as HTML
Completed 500 Internal Server Error in 1ms

NameError (undefined local variable or method `bar' for #<BooksController:0x007f
940dccc28>):
  app/controllers/books_controller.rb:5:in `index'
```

```
Rendered /Users/igarashi/.rvm/gems/ruby-1.9.3-p194/gems/actionpack-3.2.9/lib/a
ction_dispatch/middleware/templates/rescues/_trace.erb (0.9ms)
Rendered /Users/igarashi/.rvm/gems/ruby-1.9.3-p194/gems/actionpack-3.2.9/lib/a
ction_dispatch/middleware/templates/rescues/_request_and_response.erb (0.7ms)
Rendered /Users/igarashi/.rvm/gems/ruby-1.9.3-p194/gems/actionpack-3.2.9/lib/a
ction_dispatch/middleware/templates/rescues/diagnostics.erb within rescues/layout (9.3ms)
```

ログの確認

log/development.log

上記のファイルにログが出力されます。内容は rails server が出力しているものと似ていますが、ログの方が詳しい情報が出ていることもあります。

エディタで開いて確認してもいいですが、ログが追記されるごとにリロードする必要があります。
画面に次々と更新された内容を表示するには以下のコマンドが便利です。

```
$ tail -f log/development.log
```

※ tailはlinux, mac のみ。winではできません。

ログを出力

ソースファイル中で変数の中身を表示したり、処理がどこまで進んだか知りたい場合にはログを出力するのが便利です。

ここでは2つの方法を説明します。

shellに表示 p

Ruby の p メソッドがRailsでも使えます。

pメソッドは文字列を rails server の shell に表示します。

※表示だけでlog/development.logには出力しないので注意

```
class BooksController < ApplicationController
```

```
  def index
```

```
    @books = Book.all
```

```
    p '*****' ← 探し易いように目立つ文字を表示すると便利
```

```
    p @books ← p メソッドで@booksの中身を表示
```

```
...
```

```
  end
```

```
end
```

```
Started GET "/assets/application.js?body=1" for 127.0.0.1 at 2012-11-17 19:20:50  
+0900
```

```
Served asset /application.js - 304 Not Modified (25ms)
```

```
[2012-11-17 19:20:50] WARN Could not determine content-length of response body.  
Set content-length of the response or set Response#chunked = true
```

```
[2012-11-17 19:20:51] WARN Could not determine content-length of response body.  
Set content-length of the response or set Response#chunked = true
```

```
"*****"
```

```
[#<Book id: 1, title: "ドラえもん", memo: "猫型ロボットとの冒険浪漫譚", created_at: "2012-11-15 12:59:01", updated_at: "2012-11-15 12:59:01">]
```

ログに表示 Rails.logger

Rails.logger.error を使うとlog/development.log へ変数の中身や文字列を出力できます。
使い方は pと同じです。
※rails server のshellには表示しません。

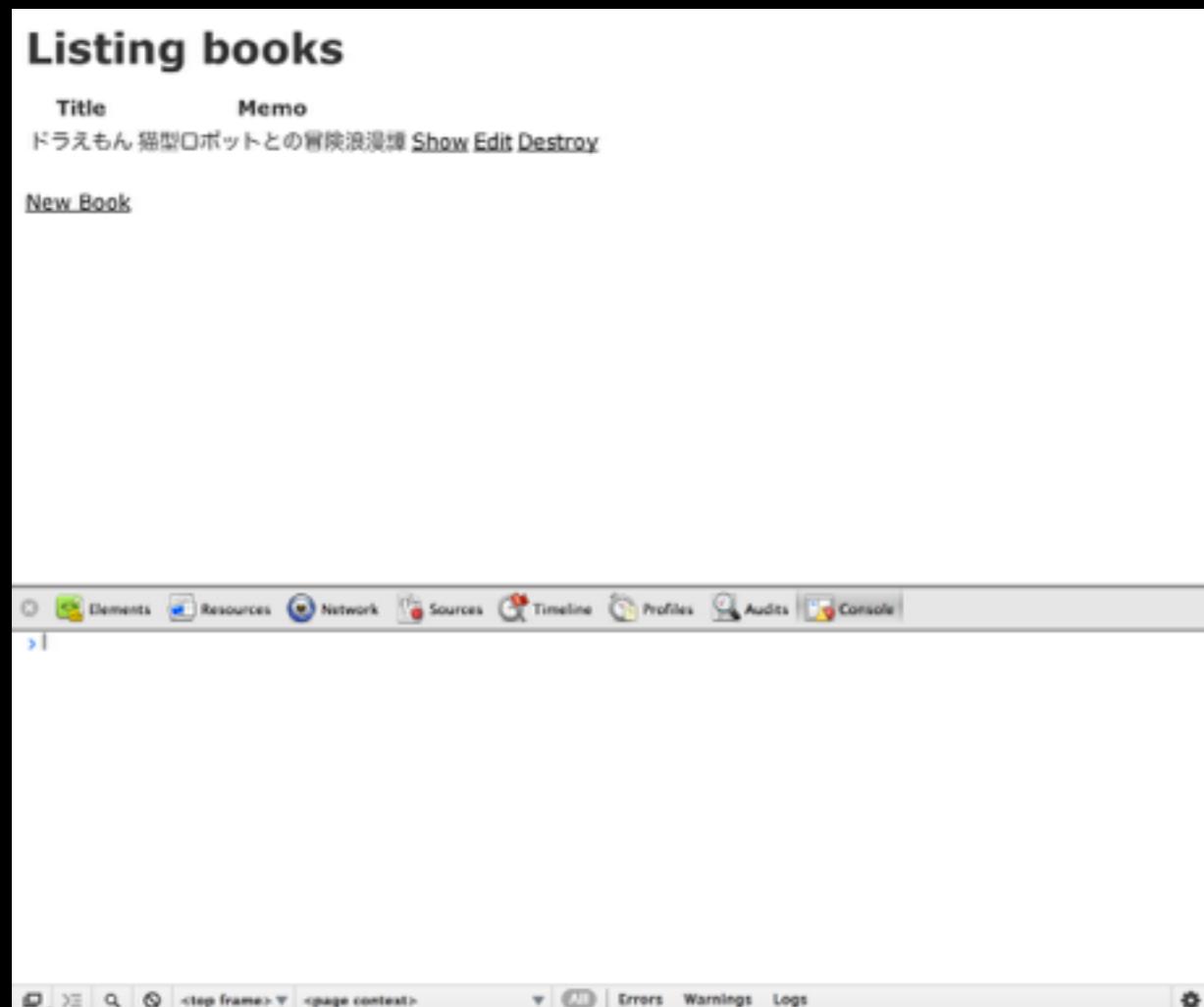
```
class BooksController < ApplicationController
  def index
    @books = Book.all
    Rails.logger.error '*****'
    Rails.logger.error @books
  ...
  end
end
```

← 目印
←@booksの中身を表示

Chromeでデバッグ

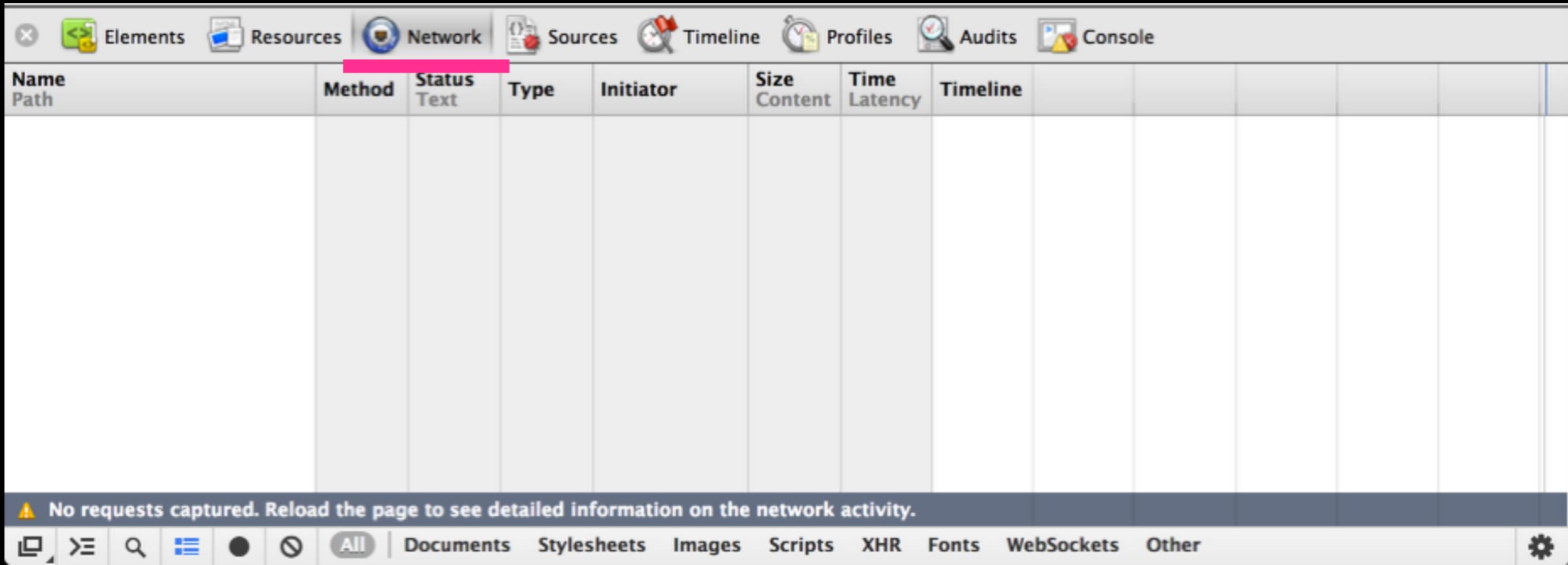
Chrome の デベロッパーツールを使うとリクエストの情報が表示できたりして便利です。

Chrome の 表示 - 開発/管理 - デベロッパーツールを選んでデベロッパーツールを起動します。



←こんなのが
表示されます。

Chromeでデバッグ



リクエストの情報を表示させるにはデベロッパーツールの Network タブを押して表示しておきます。
この状態で表示させたいリクエストを行います。
例として、/books ページへアクセスしてみましょう。

Chromeでデバッグ

The screenshot shows the Network tab in the Chrome DevTools. It lists several requests:

Name Path	Method	Status	Type	Initiator	Size Content	Time Latency	Timeline	150ms	225ms	300ms	375ms	450ms
books	GET	304 Not Mod	text/h...	Other	321B 1.24KB	19ms 18ms	<input checked="" type="checkbox"/>					
scaffolds.css /assets	GET	(failed)	Pending	books:7 Parser	13B 0B	5ms 0.0 days						
books.css /assets	GET	304 Not Mod	text/css	books:6 Parser	252B 0B	21ms 21ms	<input checked="" type="checkbox"/>					
books.js /assets	GET	304 Not Mod	application/javascript	books:8 Parser	252B 32B	16ms 16ms	<input checked="" type="checkbox"/>					
jquery.js /assets	GET	304 Not Mod	application/javascript	books:8 Parser	252B 260.63K	28ms 25ms	<input checked="" type="checkbox"/>					
application.css /assets	GET	304 Not Mod	text/css	books:5 Parser	252B 513B	28ms 24ms	<input checked="" type="checkbox"/>					

At the bottom, there are filter buttons for All, Documents, Stylesheets, Images, Scripts, XHR, Fonts, WebSockets, and Other, along with a clear button (X) and a settings gear icon.

なんかいろいろ表示されました。一番最初のbooksをクリックしてみましょう。

(まっさらにしてもう1回実行したいときは、一番下の マークをクリックすると、表示をクリアすることができます。)

Chromeでデバッグ

The screenshot shows the Chrome DevTools Network tab. On the left, a list of resources is shown, including 'books' (HTML), 'scaffolds.css' (CSS), 'books.css' (CSS), 'books.js' (JavaScript), 'jquery.js' (JavaScript), and 'application.css' (CSS). On the right, a detailed view of a request for 'books' is displayed. The 'Headers' tab is selected. The request URL is 'http://localhost:3003/books', the method is 'GET', and the status code is '304 Not Modified'. The 'Request Headers' section lists various HTTP headers. The 'Cookie' header contains a long session ID. At the bottom, there are filters for 'Documents', 'Stylesheets', 'Images', 'Scripts', 'XHR', 'Fonts', and 'WebSockets'.

Name
Path

books

scaffolds.css
/assets

books.css
/assets

books.js
/assets

jquery.js
/assets

application.css
/assets

Headers Preview Response Cookies Timing

Request URL: <http://localhost:3003/books>

Request Method: GET

Status Code: 304 Not Modified

▼ Request Headers [view source](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Charset: UTF-8,*;q=0.5

Accept-Encoding: gzip,deflate,sdch

Accept-Language: ja,en-US;q=0.8,en;q=0.6

Cache-Control: max-age=0

Connection: keep-alive

Cookie: _blog_app_session=BAh7B0kiD3Nlc3Npb25faWQG0gDg4ZjMzNjAwYjA0BjsAVEkiEF9jc3JmX3Rva2VuBjsARKkiMXcw0RGRNa3pYdC9jL2swUkU9BjsARg%3D%3D--fe120d0b40641889p_session=BAh7B0kiD3Nlc3Npb25faWQG0gZFRkkiJTY3NjNlNBjsAVEkiEF9jc3JmX3Rva2VuBjsARKkiMVRkemdsVGNLWEhuSET17RIIIORicAR0%2D%2D--8c3c1013d86568h2f65817h00ec7c8h

All Documents Stylesheets Images Scripts XHR Fonts WebSockets

リクエストの詳細が表示されます。リクエスト URLが <http://localhost:3003/books> であること、HTTPメソッドが GET であることが分かります。

Chromeでデバッグ

The screenshot shows the Chrome DevTools interface with the Network tab selected. The Response tab is highlighted with a pink background. On the left, a list of resources is shown with icons indicating their type (e.g., CSS, JS, Images). The main area displays the HTML response for the 'books' resource. The code is color-coded, with tags in purple and attributes in blue. Line numbers are visible on the left side of the code.

Name	Path	Content
books		<!DOCTYPE html> <html> <head> <title>BooksApp</title> <link href="/assets/application.css?body=1" media="all" rel="stylesheet" type="text/css"> <link href="/assets/books.css?body=1" media="all" rel="stylesheet" type="text/css"> <link href="/assets/scaffolds.css?body=1" media="all" rel="stylesheet" type="text/css"> <script src="/assets/jquery.js?body=1" type="text/javascript"></script> <script src="/assets/jquery_ujs.js?body=1" type="text/javascript"></script> <script src="/assets/books.js?body=1" type="text/javascript"></script> <script src="/assets/application.js?body=1" type="text/javascript"></script> <meta content="authenticity_token" name="csrf-param" /> <meta content="TdZglTcKXHnHKl7HJzX/XUr5I8z69DgmTslUL642sEE=" name="csrf-token" /> </head> <body> <h1>Listing books</h1> <table>
scaffolds.css	/assets	
books.css	/assets	
books.js	/assets	
jquery.js	/assets	
application.css	/assets	

ちなみに、Responseタブを表示させるとリクエストの結果返ってきたHTML(=現在表示されているページ)が表示されます。

ほかにも便利な機能がいろいろありますがまたの機会に・・・

pryでデバッグ

pry はプログラムの実行を止めて、その場所でRubyのコードを実行することができる、irbのような対話環境です。

前準備

pryを使うためには、まず前準備として Gemfile に gem 'pry' を追加します。位置は例えば gem 'sqlite3' の次の行あたり。

Gemfile

gem 'pry', group: [:development, :test] ←追加

※ gem 'pry' だけでも動作します。上記は開発モード(:development)とテストモード(:test)だけでpryを使い、サービス運用モード(:productionといいます)ではpryを使わない設定にしています。

Railsは何も指定しない場合は通常developmentモードで動作しています。

追加したら、shell で bundle コマンドを実行してpryをインストールします。

\$ bundle

pryでデバッグ

次に、ソースコードで止めたい場所へ
binding.pry と書きます。

例として、/books へアクセスされたときに停止するように
BooksController の index アクションに書いてみましょう。

```
class BooksController < ApplicationController
  def index
    @books = Book.all
    binding.pry ←追記
...
  end
end
```

rails server を再起動して、ブラウザでアクセスしてみましょう。
<http://localhost:3000/books>

pryでデバッグ

ブラウザでアクセス後、**rails server** のshellを見るとプログラムが停止してコマンドを打てるようになっています。
irbのようにRubyのコードが実行できます。

```
From: /Users/igarashi/work/books_picture_app/app/controllers/books_controller.rb
@ line 8 BooksController#index:

  6: def index
  7:   @books = Book.all
=> 8:   binding.pry
  9: end
```

pryでデバッグ

```
Started GET "/books" for 127.0.0.1 at 2013-12-01 17:46:52 +0900
Processing by BooksController#index as HTML

From: /Users/igarashi/work/books_picture_app/app/controllers/books_controller.rb @ line 8 BooksController#index:

6: def index
7:   @books = Book.all
=> 8:   binding.pry
9: end

[1] pry(#<BooksController>) > p @books
Book Load (0.2ms)  SELECT "books".* FROM "books"
#<ActiveRecord::Relation [#<Book id: 2, title: "スタートアップRuby", memo: "Rubyの入門書", picture: "startup_ruby.png", created_at: "2013-12-01 06:46:53", updated_at: "2013-12-01 06:46:53">, #<Book id: 4, title: "4月は君の嘘", memo: "ピアノを巡る青春物語", picture: nil, created_at: "2013-12-01 08:46:39", updated_at: "2013-12-01 08:46:39">]
=> [#<Book id: 2, title: "スタートアップRuby", memo: "Rubyの入門書", picture: "startup_ruby.png", created_at: "2013-12-01 06:46:53", updated_at: "2013-12-01 06:46:53">,
 #<Book id: 4, title: "4月は君の嘘", memo: "ピアノを巡る青春物語", picture: nil, created_at: "2013-12-01 08:46:39", updated_at: "2013-12-01 08:46:39">]
```

コードを実行できるので、変数の中身を `p` で表示できます。
(実は、`p` を打たなくても `@books` と打つだけでも表示できる。)

`pry` を終了してプログラムを再開するには `exit` と打ちます。

`params`を表示したり、`Book.where` で検索を試してみたり、工夫次第で便利な道具として使うことができます。

參考資料

書式

Rubyコード

`puts "abc"`

実行結果

"abc"

実行結果

文中では `#=>` で書きます

`p 1+2 #=> 3`

shellコマンド

`$ ls`

自習用Rails資料

特に教科書は使いませんが、自習用には以下の資料をお勧めします。

▶ RailsTutorial (web)

▶ <http://railstutorial.jp/?version=4.0>

▶ ドットインストール(web)(動画)

▶ http://dotinstall.com/lessons/basic_rails_v2

▶ Rails Guide (web)(English)

▶ <http://guides.rubyonrails.org/>

▶ RailsによるアジャイルWebアプリケーション開発 第4版

▶ <http://www.amazon.co.jp/dp/4274068668/>



▶ 改訂新版 基礎Ruby on Rails

▶ <http://www.amazon.co.jp/dp/4844331566/>



▶ たのしいRuby

▶ <http://www.amazon.co.jp/dp/4797372273/>



講義資料置き場

過去の資料がDLできます。

<https://github.com/igaiga/hitotsubashi-ruby-2013>

雑談・質問用facebookグループ

<https://www.facebook.com/groups/hitotsubashi.rb>

- ・加入/非加入は自由です
- ・加入/非加入は成績に関係しません
- ・参加者一覧は公開されます
- ・参加者はスタッフ(講師・TA)と昨年、今年の受講者です
- ・書き込みは参加者のみ見えます
- ・希望者はアクセスして参加申請してください
- ・雑談、質問、議論など何でも気にせずどうぞ～
- ・質問に答えられる人は答えてあげてください
- ・講師陣もお答えします
- ・入ったら軽く自己紹介おねがいします