第 3 章

Hello, world!

この章から、Ruby on Rails についての具体的な学習が始まります。素材として単純な Web アプリを開発します。その名も SimpleGreeter。ブラウザに「Hello, world!」と表示するだけの機能しかありません。単純ではありますが、これを作る過程から私たちは多くのことを学べます。

3.1 準備作業

この章では、Rails 開発の練習素材として「SimpleGreeter」という Web アプリを作ります。その準備として、まずは Rails アプリケーション群のソースコードを格納するためのディレクトリ `projects` をホームディレクトリの下に作ってください。

> OIAX BOOKS の『Ruby on Rails 5.0 インストール手順書』に沿って作業した場合、このディレクトリはすでに作られています。

ターミナルを開き、次のコマンドを実行してください。

```
$ mkdir -p ~/projects
```

そして、このディレクトリ以下で使用する *ruby* のバージョンを 2.3.1 とし

ます。

```
$ cd ~/projects
$ rbenv local 2.3.1
```

念のため *ruby* のバージョンを確認しましょう。

```
$ ruby -v
```

ターミナルに表示される結果は OS によって異なります（1 行目が OS X の場合、2 行目が Ubuntu の場合）。

```
ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin15]
ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-linux]
```

次に、このコマンドを実行してください。

```
$ rails -v
```

結果としてこう表示されれば OK です。

```
Rails 5.0.0
```

`5.0.0` がバージョン番号です。最後の桁の数字は違っていても構いません。このような結果にならない場合は、次のコマンドで Rails 5.0 をインストールしてください。

```
$ gem install rails -v 5.0
```

3.2 Rails アプリの骨格を生成する

では、SimpleGreeter の開発を始めましょう。最初に *rails* のサブコマンド `new` で Rails アプリの骨格を生成します。

通常 rails new コマンドは、アプリケーションの骨格を生成した後に、アプリケーションが依存する Gem パッケージを一括インストールしますが、オプション -B または --skip-bundle を指定すると、この処理がスキップされます。こうした方が、途中で問題が発生した場合の状況がつかみやすくなります。

```
$ rails new simple_greeter -B
```

引数 simple_greeter は Rails アプリのソースコードを設置するディレクトリのパスです。このパスがアプリの名前になりますので、綴りを間違えないように注意して入力してください。

ターミナルに次のような結果が表示されます。

```
      create
      create  README.md
      create  Rakefile
      create  config.ru
...
      create  vendor/assets/stylesheets
      create  vendor/assets/stylesheets/.keep
      remove  config/initializers/cors.rb
```

カレントディレクトリを simple_greeter に移動し、*bundle* コマンドを実行します。

```
$ cd simple_greeter
$ bundle
```

ターミナルに次のような結果が表示されます。

```
Fetching gem metadata from https://rubygems.org/
Fetching version metadata from https://rubygems.org/
Fetching dependency metadata from https://rubygems.org/
Resolving dependencies......
Using rake 11.2.2
...
Using rails 5.0.0
Using sass-rails 5.0.6
Bundle complete! 15 Gemfile dependencies, 63 gems now installed.
```

```
Use `bundle show [gemname]` to see where a bundled gem is installed.
```

そして、テキストエディタまたは IDE でこのディレクトリを開いてください。

> ディレクトリをテキストエディタや IDE で開く方法は、利用するソフトウェアの種類によって異なります。Atom を利用する場合は、ターミナルで `atom .` コマンドを実行してください。

3.3 とにかく作ってみよう

経路の設定

Rails アプリに新たな機能を追加するとき、まず最初に行うことは**経路** (route) の設定です。

> 実は、Rails 用語の“route”を「経路」と訳すのはあまり一般的ではありません。「ルート」または「ルーティング」と呼ぶのが普通です。筆者の旧著でも「ルーティング」を使ってきました。しかし、『Ruby on Rails 5.0 初級』シリーズでは、あえて漢字を用いた訳語を採用することにしました。「ルート」は“root”と混同しやすいし、“route”と“routing”はやはり異なる概念です。なるべくカタカナ語を減らしたいという意図もあります。違和感を持つ読者もいらっしゃるかもしれませんが、ご容赦ください。

経路という言葉の説明は後回しにします。テキストエディタで `config/routes.rb` を開いてください。初期状態では次のような内容です。

LIST config/routes.rb

```
1  Rails.application.routes.draw do
2    # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
3  end
```

このファイルを次のように書き換えてください。

LIST config/routes.rb

```
1  Rails.application.routes.draw do
2 -  # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
2 +  get 'hello' => 'hello#show'
3  end
```

追加された行は、'hello' と 'hello#show' を矢印で結んでいるように見えますね。この点を記憶にとどめて、次に進みましょう。

アクションの作成

次に**アクション**（action） を作ります。アクションを作るには、まず**コントローラ**（controller） を用意する必要があります。コントローラはアクションの「入れ物」です。

app/controllers ディレクトリの下に新規ファイル hello_controller.rb を作り、テキストエディタで次の内容を書き込んでください。

LIST app/controllers/hello_controller.rb (New)

```
1  class HelloController < ApplicationController
2    def show
3    end
4  end
```

アルファベットの大文字と小文字の違いやスペースの有無に注意して入力してください。1 行目に見える Hello と 2 行目に見える show という文字列を覚え

ておいてください。詳しくは次章で説明しますが、2〜3 行目に書かれているものがアクションです。

テンプレートの作成

続いて、**テンプレート**（template） を作ります。「ひな型」という意味です。何のひな型かと言えば、HTML 文書のひな型です。

まず、app/views ディレクトリの下に hello サブディレクトリを作成してください。

```
$ mkdir -p app/views/hello
```

そして、テキストエディタで app/views/hello ディレクトリの下に新規ファイル show.html.erb を作り、次の内容を書き込んでください。

LIST app/views/hello/show.html.erb (New)

```
1  <p>Hello, world!</p>
```

ディレクトリ名に hello、ファイル名に show という見覚えのある文字列が含まれていますね。

以上で、ブラウザの画面に「Hello, world!」と表示する準備が整いました。

Rails サーバーを起動する

まず、ターミナルで次のコマンドを実行してください。

```
$ rails s
```

ターミナルには次のような結果が表示されます。

```
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:>
3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.6.0 (ruby 2.3.1-p112), codename: Sleepy Sunday Serenity
```

```
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
```

ターミナルに `Listening on tcp://localhost:3000` と出力されれば、Rails サーバーがブラウザからの要求を受け付ける状態になっています。なお、Rails サーバーを終了するにはキーボードで `Ctrl` キーと `C` を同時に押してください。

結果の 1 行目に「Puma」という文字列が見えます。Rails 4.2 まではここに「WEBrick」と書かれていました。WEBrick は Web サーバーの機能を提供する *ruby* の標準ライブラリで、主に Web アプリの開発環境で利用されます。他方、Puma は本番環境での実績を多く持つ本格的な Web サーバーです。Rails 5.0 からは開発環境におけるデフォルトの Web サーバーが Puma になりました。

ブラウザで表示を確認する

ブラウザのアドレスバーに `http://localhost:3000/hello` と入力して Enter キーを押してください。すると、ブラウザの画面が 図 3.1 のように変化します。

最近のブラウザでは、私たちがアドレスバーに `http://` で始まる文字列（URL）を入力しても、先頭にある `http://` を自動で隠してしまいます。そのため、図 3.1 ではアドレスバーに `localhost:3000/hello` と表示されています。

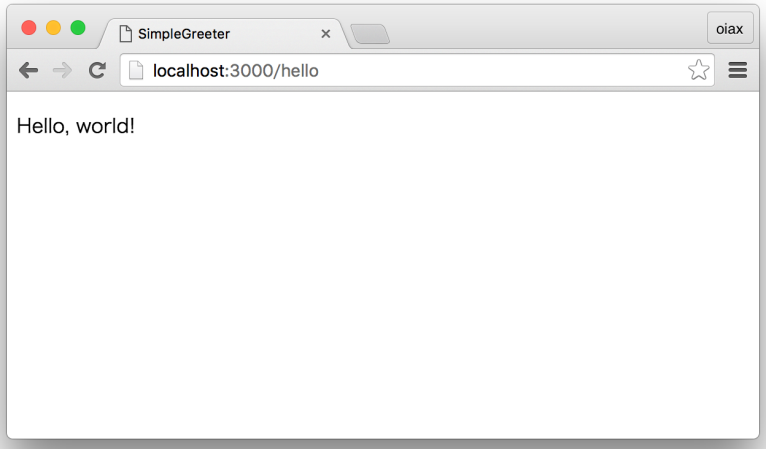

図 3.1　「Hello, world!」メッセージの表示

ページのソースを表示する

ブラウザの画面を右クリックし、コンテキストメニューから「ページのソースを表示」を選んでください。すると、次のようなテキストが画面に表示されるはずです（一部省略）。

```
<!DOCTYPE html>
<html>
  <head>
    <title>SimpleGreeter</title>
    <meta name="csrf-param" content="authenticity_token" />
...
  </head>

  <body>
    <p>Hello, world!</p>

  </body>
</html>
```

これが、Rails サーバーからブラウザに送られてきたものです。HTML という言語で記述されているので、これを **HTML 文書**（HTML document） と呼びます。HTML に関しては 第 5 章 で説明します。

第 15 章

画像

本章では、Web アプリ上に写真やイラストなどの画像を表示する方法と、Web アプリにファビコン（ブックマークやショートカット用のアイコン）を設定する方法について解説します。

15.1 `img` 要素

HTML 文書に**画像**（image）を挿入するには `img` 要素を使用します。この要素の `src` 属性には画像ファイルの URL を、`alt` 属性には**代替文字列**（alternative text）を指定します。代替文字列は、何らかの理由で画像ファイルにアクセスできない場合に画像の代わりにブラウザの画面に表示されます。音声読み上げブラウザも代替文字列を利用します。

HTML 文書では **PNG**、**JPEG**、**GIF**、**SVG** という 4 種類のファイル形式の画像がよく使われます。通常、ファイルの拡張子には、`png`、`jpeg`、`gif`、`svg` 等ファイル形式の名前を小文字にしたものを使います。ただし、JPEG 形式のファイルの拡張子には `jpg` もよく使われます。

各ファイル形式の特徴を箇条書きでまとめます：

PNG フルカラー（約 1,677 万色）の画像を扱える。透過処理が可能。

JPEG フルカラー（約 1,677 万色）の画像を扱える。写真向き。

GIF 最大 256 色の画像しか扱えないが、透過処理とアニメーション処理が可能。

SVG "Scalable Vector Graphics" の略。拡大・縮小しても品質が劣化しない。

次に示すのは、アメリカ航空宇宙局（NASA）が配布している地球の画像を表示する img 要素の記述例です。

```
<img src="https://www.oiax.jp/books/files/nasa_earth.jpg" alt="地球の写真">
```

img 要素に width 属性と height 属性に値を指定することにより、画像がブラウザ上に表示される幅と高さを調節できます。これらの属性には幅と高さのピクセル数を「100」のような整数値で指定します。

```
<img src="https://www.oiax.jp/books/files/nasa_earth.jpg" alt="地球の写真"
  width="240" height="240">
```

15.2 image_tag メソッド

Rails アプリ上に画像を表示する場合、img 要素を直接記述する代わりに image_tag メソッドを使用して img 要素を生成するのが一般的です。

このメソッドを用いる場合、画像ファイルは app/assets/images ディレクトリに置きます。ターミナルで次のコマンドを実行してください。

```
$ wget https://www.oiax.jp/books/files/robot.svg
$ mv robot.svg app/assets/images
```

そして、hello#show アクション用のテンプレートを次のように書き換えます。

LIST app/views/hello/show.html.erb

```
  :
4   <div class="card m-a-1">
5 +   <%= image_tag 'robot.svg', alt: 'Robot', size: '320x320' %>
6     <div class="card-block card-inverse card-success">
  :
```

image_tag メソッドの第 1 引数に画像のファイル名 robot.svg を、オプション alt に代替文字列を指定します。必要に応じて、オプション size に表示上の

画像の幅と高さをアルファベットの 'x' で連結して指定してください。

ブラウザで SimpleGreeter のトップページで「Alice」リンクをクリックすると、図 15.1 のような画面が表示されます。

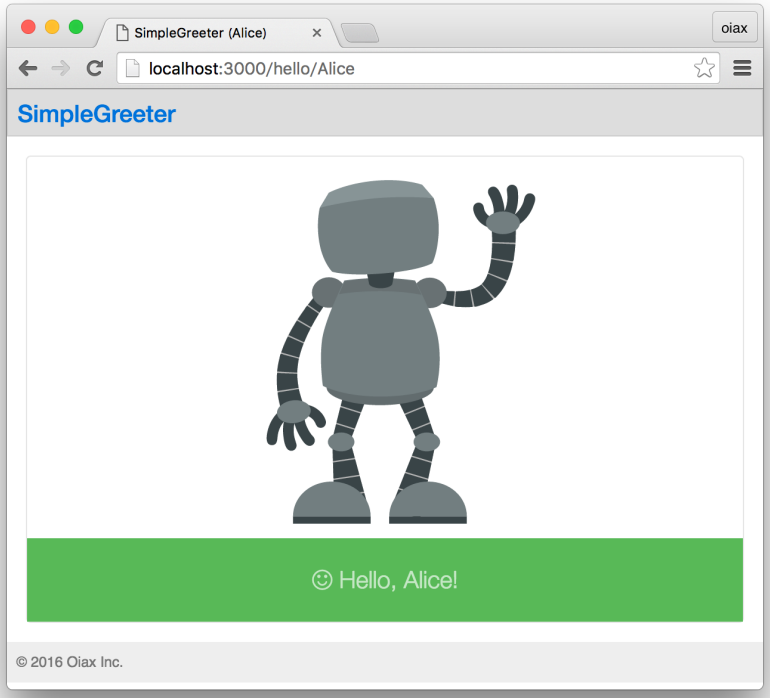

図 15.1　メッセージの上にロボットの画像を表示

15.3　画像表示幅の調節

実は、さきほど見たページを iPhone 5 で表示すると 図 15.2 左側のようにロボットのイラストが少し右寄りになります。

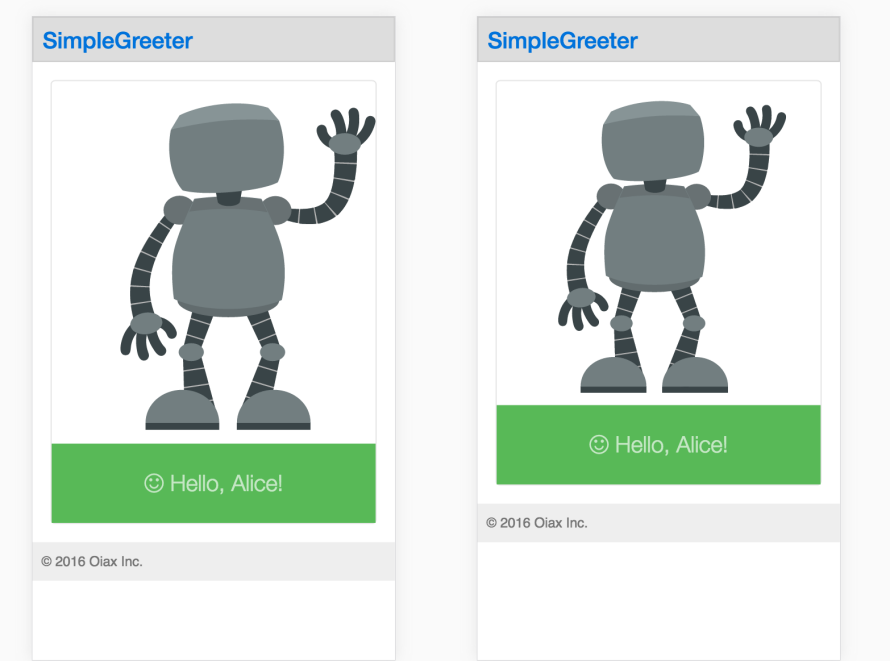

図 15.2　スマートフォン向けに画像の表示幅を調節

iPhone 5 の画面は 320px 幅の画像がちょうど収まる大きさなので（縦長表示の場合）、Card コンポーネントの内側に収まらないのです。そこで、`show.html.erb` を次のように修正します。

LIST app/views/hello/show.html.erb

```
:
5 -     <%= image_tag 'robot.svg', alt: 'Robot', size: '320x320' %>
5 +     <%= image_tag 'robot.svg', alt: 'Robot',
6 +       style: 'width: 100%; max-width: 320px' %>
:
```

`image_tag` メソッドから `size` オプションを取り去り、代わりに `style` オプションに CSS でスタイルを指定しました。まず、`width` プロパティに `100%` という値を指定することで、画像が親要素の幅いっぱいに表示されるようにします。これで iPhone 5 のような表示幅の狭いスマートフォンで画像が中央に揃い

ます（図 15.2 右側）。

ただし、この指定だけでは幅の広いブラウザで表示した時にイラストが大きくなりすぎます。そこで、max-width プロパティに 320px という値を指定しました。このプロパティは要素の表示幅の最大値を示します。

■コラム: 画像ファイルのフィンガープリント

本文中で私は「image_tag メソッドを使用して img 要素を生成する」と書きましたが、実際にどのような HTML フラグメントが作られているのかを確認してみましょう。図 15.1 を開いている状態でブラウザの「ページのソースを表示」機能を使って調べると、次のようなコードになっていることがわかります。ただし、src 属性の値は長すぎるので一部省略しています（... の部分）。

```
<img alt="Robot" width="320" height="320"
  src="/assets/robot-e31c3ae...f5fa5a1.svg" />
```

image_tag メソッドの第 1 引数に指定したファイル名は robot.svg でしたが、ファイル名の本体と拡張子の間にマイナス記号と 64 桁の 16 進数が挿入されています。また、画像ファイルを置いたディレクトリは app/assets/images でしたが、URL パスのディレクトリ部分は /assets/ になっています。

64 桁の 16 進数は、SHA-256 というアルゴリズムで計算された画像ファイルの**フィンガープリント**（fingerprint）です。英語の“fingerprint”は「指紋」を意味し、画像の中身が少しでも変化するとこの値はまったく異なる値になります。

フィンガープリントの役割はキャッシュの制御です。これがないとファイル名が変化しないため、画像の内容が更新された時でもブラウザがキャッシュに保存された古いファイルを利用することがあります。フィンガープリントのおかげで、新しい画像が確実にブラウザに表示されるのです。

15.4 ファビコンと Web クリップアイコンの指定

本巻の締めくくりとして、SimpleGreeter にファビコンと Web クリップアイコンを設定しましょう。

ファビコン（favicon）とは、ブラウザで特定の Web サイトを開いた時にタブに表示される小さなアイコンです。また、ブラウザのブックマークやお気に入りをリスト表示する際にもこのアイコンが使われます。他方、**Web クリップアイコン**は、スマートフォンのブラウザで Web サイトをホーム画面に追加した時に表示されるアイコンです。

ターミナルで次のコマンドを実行してください。

```
$ wget https://www.oiax.jp/books/files/simple_greeter.ico
$ wget https://www.oiax.jp/books/files/robot256.png
$ mv simple_greeter.ico robot256.png app/assets/images
```

ファビコンは **ICO 形式**の画像ファイルです。拡張子は `ico` とします。Web クリップアイコンには、256 ピクセルの幅と高さを持つ JPEG 形式または PNG 形式の画像ファイルを用意してください。

次に、レイアウトテンプレートを次のように書き換えます。

LIST app/views/layouts/application.html.erb

```
  :
  9       <%= javascript_include_tag 'application', 'data-turbolinks-track': '
        reload' %>
 10 +     <%= favicon_link_tag('simple_greeter.ico') %>
 11 +     <%= favicon_link_tag('robot256.png', rel: 'apple-touch-icon') %>
 12     </head>
  :
```

追加された二つの `favicon_link_tag` メソッドは、次のような HTML フラグメントを生成します（ファイル名の一部を省略しています）。

```
<link rel="shortcut icon" type="image/x-icon" href="/assets/favicon-
84047ae...141ea09.ico" />
<link rel="apple-touch-icon" type="image/x-icon" href="/assets/robot256-
eeebb62...35ea802.png" />
```

HTML の link 要素の役割は、HTML 文書に関連するファイルを指定することです。必ず head 要素の中に置かれます。CSS ファイルを読み込むために用いられるほか、上記の例のようにファビコンや Web クリップアイコンを指定するのにも用いられます。なお、link 要素は空要素の一種です。

ブラウザをリロードするとタブにファビコンが表示されます（図 15.3）。

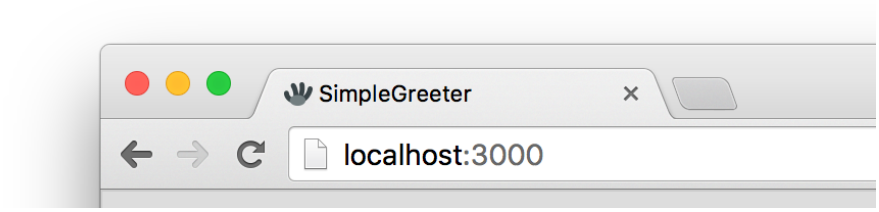

図 15.3　タブにファビコンを表示

以上で、本書『Ruby on Rails 5.0 初級①』はおしまいです。

次巻『Ruby on Rails 5.0 初級②』は 2016 年秋の刊行予定です。データベース、モデル、リソースなどが主なテーマとなります。