第 2 章

Rails 開発の準備作業

2.1　データベース管理システム（DBMS）の選択

PicoPlanner の開発に着手しましょう。前巻で作った SimpleGreeter とは異なり、今回はデータベースを利用しますので、最初の作業は**データベース管理システム**（DBMS）を選ぶことです。

データベースとは情報の集合です。情報の単なる寄せ集めではなく、検索や集計がすばやくできるように構造化されたものです。このデータベースを操作するソフトウェアがデータベース管理システムです。

Ruby on Rails の業界でよく使われるのは、次の 3 種類です。

- PostgreSQL
- MySQL
- SQLite3

このうち準備作業が最も楽なのが SQLite3 です。本番環境での運用には不向きですが学習用途としては十分ですので、本巻ではこれを採用します。

2.2　ラクダ、ヘビ、鎖

次に進む前に、アルファベットの表記法に関する話をします。

私たちがこの巻で作成する Web アプリの名前は PicoPlanner です。この名前は “pico” と “planner” というふたつの単語から構成されるわけですが、

“pico planner” のように単語間を空白文字で連結するのではなく、それぞれの単語の先頭を大文字に変えて直接連結しています。この表記法は**キャメルケース**（camel case）と呼ばれます。“camel” は「ラクダ」を意味する英単語です。大文字の部分が「ラクダのこぶ」に見えることに由来します。

“pico” と “planner” からひと続きの文字列を作る方法は他にもあります。ひとつは空白文字の代わりにアンダースコア（_）を用いて “pico_planner” のように連結するものです。この表記法を**スネークケース**（snake case）と呼びます。“snake” は爬虫類のヘビを意味します。

もうひとつが、マイナス記号（-）で単語間を連結する**チェーンケース**（chain case）という表記法です。“chain” は鎖という意味ですね。“pico” と “planner” から “pico-planner” という文字列が作られます。チェーンケースは CSS の `id` 属性や `class` 属性の値としてしばしば使われます。

2.3 新規 Rails アプリの作成

では、PicoPlanner のソースコードの骨格を作成しましょう。以下のコマンドを順に実行してください。

```
$ cd ~/projects
$ rails new pico_planner -B -C -T -d sqlite3
$ cd pico_planner
```

`rails new` コマンドの使い方については『初級①』で学習しました。第 1 引数にはソースコードを格納するディレクトリを指定します。通常、ディレクトリ名は全部小文字で表記するか、スネークケースで表記します。

上記のコマンドで使われているオプションについては 表 2.1 をご覧ください。`bundle install` コマンドを実行する前に `Gemfile` を書き換えたいので、オプション `-B` を指定しています。オプション `-C` と `-T` は、PicoPlanner では利用しないソースコードの生成を止めるために指定しています。オプション `-d` には DBMS の名前を指定します。オプション `-d` は省略しても構いません。この場合は、DBMS として SQLite3 が選ばれます。

表 2.1　rails new コマンドのオプション

オプション	意味
-B	bundle install の実行をスキップする
-C	ActionCable 関連のソースコードを生成しない
-T	テスト関連のソースコードを生成しない
-d sqlite3	DBMS として SQlite3 を選択
-d mysql	DBMS として MySQL を選択
-d postgresql	DBMS として PostgreSQL を選択

■コラム: アプリケーションの名前

rails new コマンドの第 1 引数に指定された文字列は、キャメルケースに変換された後、ソースコードの中でアプリケーションの名前として使われます。具体的には、次のふたつのファイルに埋め込まれます。

- app/views/layouts/application.html.erb
- config/application.rb

もし開発が進んでしまった後にアプリケーションの名前を変更したい場合は、上記のファイルの内容を書き換えてください。

2.4　Gemfile の編集

続いて、Gemfile を編集します。このファイルには、PicoPlanner が依存する（必要とする）Gem パッケージが列挙されています。

まず、次のように 1 行挿入してください。

Gemfile

```
1  source 'https://rubygems.org'
2
3  git_source(:github) do |repo_name|
4    repo_name = "#{repo_name}/#{repo_name}" unless repo_name.include?("/")
5    "https://github.com/#{repo_name}.git"
6  end
```

```
 7
 8 + ruby '2.3.3'
 9
10  # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
 :
```

この記述は *ruby* のバージョンを 2.3.3 に固定するためのものです。こう書くことにより、2.3.3 以外の *ruby* で PicoPlanner を動かそうとした時にエラーが出て止まります。

ruby の挙動はバージョンごとに若干異なります。ある環境で開発した PicoPlanner を別の環境で動かすときに、なるべくトラブルが起こらないようにするため、*ruby* のバージョンを揃えます。

続いて、Gemfile からナンバー記号（#）で始まる行をすべて除去します。これらの行にはコメントが書いてあるだけです。これらのコメントを残しておいてもほとんど役に立ちません。開発が進むにしたがって、Gemfile の記述はどんどん増えていくことになりますので、初期段階ですっきりさせておきましょう。

Gemfile

```
 1  source 'https://rubygems.org'
 2
 3  git_source(:github) do |repo_name|
 4    repo_name = "#{repo_name}/#{repo_name}" unless repo_name.include?("/")
 5    "https://github.com/#{repo_name}.git"
 6  end
 7
 8  ruby '2.3.3'
 9
10  gem 'rails', '~> 5.0.1'
11  gem 'sqlite3'
12  gem 'puma', '~> 3.0'
13  gem 'sass-rails', '~> 5.0'
14  gem 'uglifier', '>= 1.3.0'
15  gem 'coffee-rails', '~> 4.2'
16
17  gem 'jquery-rails'
18  gem 'turbolinks', '~> 5'
```

```
19  gem 'jbuilder', '~> 2.5'
20
21  group :development, :test do
22    gem 'byebug', platform: :mri
23  end
24
25  group :development do
26    gem 'web-console', '>= 3.3.0'
27    gem 'listen', '~> 3.0.5'
28    gem 'spring'
29    gem 'spring-watcher-listen', '~> 2.0.0'
30  end
31
32  gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
```

次に、Gemfile の 10 行目を次のように書き換えます。

Gemfile

```
   :
10 - gem 'rails', '~> 5.0.1'
10 + gem 'rails', '= 5.0.1'
   :
```

通常は、Rails のバージョン番号を '~> 5.0.1' のように指定します。この場合、bundle update コマンドにより Rails のバージョンは 5.0. で始まるもっとも大きな数字のところまでアップグレードされます。しかし、本書の内容はバージョン 5.0.1 で動作確認されていますので、トラブルを避けるために Rails のバージョンを 5.0.1 に固定します。

続いて、Gemfile から coffee-rails に関する行を除去します。

Gemfile

```
   :
15 - gem 'coffee-rails', '~> 4.2'
   :
```

JavaScript の代わりに CoffeeScript というプログラミング言語を使用するための Gem パッケージですが、PicoPlanner では不要です。

さらに、Gemfile から末尾の 2 行を削除します。

Gemfile

```
  :
31 -
32 - gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
```

tzinfo-data は Windows 環境および JRuby 環境でのみ必要な Gem パッケージです。

最後に、Gem パッケージ群をインストールします。

```
$ bundle install
```

2.5 データベースの作成

ターミナルで次のコマンドを実行してください。

```
$ ls db
```

次のような結果が出力されます。

```
seeds.rb
```

これが、現在の db ディレクトリの中身です。seeds.rb ファイルがあるだけです。ここで、次のコマンドを実行します。

```
$ rails db:create
```

すると、ターミナルに次のように表示されます。

```
Created database 'db/development.sqlite3'
Created database 'db/test.sqlite3'
```

db ディレクトリの中身を確認してみましょう。

```
$ ls db
```

次のように出力されますね。

```
development.sqlite3  seeds.rb  test.sqlite3
```

拡張子 .sqlite3 を持つふたつのファイルが新たに作られました。これらがデータベースの実体です。

アプリケーションの開発時に使用するのは development.sqlite3 です。test.sqlite3 は自動テストを行うときに使うものですので、この『Ruby on Rails 5.0 初級』シリーズで利用する機会はありません。

2.6 Bootstrap と Font Awesome の導入

Bootstrap と Font Awesome を導入します。以下、導入手順を簡潔に紹介します。詳しくは『初級①』の第 9 章を参照してください。

まずは、Gemfile の書き換え。

Gemfile

```
  :
18    gem 'jbuilder', '~> 2.5'
19 +
20 +  gem 'bootstrap', '4.0.0.alpha6'
21 +  gem 'tether-rails'
22 +  gem 'font-awesome-sass'
23
24    group :development, :test do
  :
```

インストール。

```
$ bundle install
```

不要なファイルを削除。

```
$ rm app/assets/stylesheets/application.css
```

同ディレクトリに新規ファイル application.scss（拡張子に注意）を作成し、次の内容を書き込みます。

app/assets/stylesheets/application.scss (New)

```
1  @import 'bootstrap';
2  @import 'font-awesome-sprockets';
3  @import 'font-awesome';
4  @import '*';
```

app/assets/stylesheets ディレクトリに application.scss 以外のファイルが存在しないと、4 行目の @import '*'; でエラーになるため、空の SCSS ファイルを作ります。

```
$ touch app/assets/stylesheets/main.scss
```

app/assets/javascripts ディレクトリの application.js を書き換えます。

app/assets/javascripts/application.js

```
 :
13  //= require jquery
14  //= require jquery_ujs
15  //= require turbolinks
16 + //= require tether
17 + //= require bootstrap-sprockets
18  //= require_tree .
```

app/views/layouts ディレクトリの application.html.erb を編集します。

app/views/layouts/application.html.erb

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4 +     <meta name='viewport' content='width=device-width, initial-scale=1'>
5       <title>PicoPlanner</title>
:
```

ビューポート（viewport） に関する meta 要素を追加しました。ビューポートとは、スマートフォン用ブラウザにおける表示領域と拡大／縮小に関する設定です。width=device-width は「表示領域の幅を端末メーカー推奨の値に合わせる」という意味です。また、initial-scale=1 は「初期状態では拡大／縮小をしない」という意味です。

2.7 README.md の編集

準備作業の仕上げとして、README.md を書き換えます。初期状態はこのようになっています。

README.md

```
1   # README
2
3   This README would normally document whatever steps are necessary to get the
4   application up and running.
:
```

いったんすべて削除して、次の内容を記入してください。

README.md

```
1   # PicoPlanner: 簡易予定表管理システム
2
3   これは『Ruby on Rails 5.0 初級②』の学習用 Rails アプリケーションです。
4
5   ## 稼働条件
6
7   * macOS 10.12、 OS X v10.11 または Ubuntu 16.04
```

```
 8  * Ruby 2.3.3
 9
10  ## インストール方法
11
12  ```text
13  $ gem install bundler
14  $ bundle install
15  $ rails db:setup
16  ```
17
18  ## 起動方法
19
20  ```text
21  $ rails s
22  ```
```

README.md には、Rails アプリケーションの概要、稼働条件、インストール方法、起動方法、などを **Markdown** で記述します。

Markdown はマークアップ言語の一種、つまり HTML の仲間です。1 行目のナンバー記号（#）は、HTML の h1 要素に相当する見出しを示します。5 行目にある 2 個のナンバー記号（##）は、HTML の h2 要素に相当する見出しを示します。7 行目と 8 行目のアスタリスク（*）は、箇条書き項目を示します。

12〜16 行には、整形済みテキスト（HTML の pre 要素に相当）が記述されています。

````
```text
$ gem install bundler
$ bundle install
$ rails db:setup
```
````

3 個のバッククォート（```）で囲まれた範囲については、空白文字や改行がそのまま維持され、等幅フォントで表示されます。なお、12 行目末尾の text は、内容がプレーンテキストであることを示しています。Ruby のソースコードを整形済みテキストとして表示したい場合はここに ruby と書いてください。Markdown の処理システムによっては、Ruby 言語向けにシンタックスハイライト（可読性向上のために部分的に色やフォントを変える効果）が施されます。

テキストエディタ Atom をお使いの方は、ctrl + shift + M で Markdown 文書をプレビューできます（図 2.1）。

図 2.1　Markdown 文書のプレビュー

■コラム: README.md の存在意義

Rails アプリケーションを動かすだけであれば、README.md の中身は初期状態のままで構わないし、消してしまっても大丈夫です。しかし、Rails

アプリケーションの開発が完了または中断してしばらく経過すると、作った本人の記憶が急速に薄れていきますので、Rails アプリケーションに関する基本的な情報を README.md の中にまとめておくことをお勧めします。

なお、アプリケーションのソースコードを GitHub のようなサービスで管理するときには、このファイルがプロジェクトのトップページとして使われることになります。