第 9 章

rbenv コマンドの使い方

この章では *rbenv* コマンドを用いて、特定のバージョンの *ruby* をインストールしたり、選択したりする方法を学びます。

9.1 *rbenv* とは

rbenv コマンド *rbenv* は、コンピュータにインストールされた複数のバージョンの *ruby* から一つを選択するコマンドです。また ruby-build というプラグイン（拡張機能）を追加することにより、バージョンを指定して *ruby* をインストールすることができます。

9.2 *rbenv* のインストール

macOS の場合

以下の操作手順ではホームディレクトリにある設定ファイル .bash_profile の内容を書き換えます。操作ミスで ~/.bash_profile を壊してしまわないように、最初にバックアップファイル ~/.bash_profile.bak を作成してください。

```
$ touch ~/.bash_profile
$ cp -f ~/.bash_profile ~/.bash_profile.bak
```

途中で何か間違えたら、次のコマンドで ~/.bash_profile を元に戻すことが

できます。

```
$ cp -f ~/.bash_profile.bak ~/.bash_profile
```

ターミナルで以下のコマンドを順に実行してください。

```
$ brew install rbenv ruby-build
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash_profile
$ echo 'eval "$(rbenv init -)"' >> ~/.bash_profile
$ source ~/.bash_profile
```

echo で始まる二つのコマンドは間違いやすいので、特に注意して入力してください。

Ubuntu の場合

以下の操作手順ではホームディレクトリにある設定ファイル .bashrc の内容を書き換えます。操作ミスで ~/.bashrc を壊してしまわないように、最初にバックアップファイル ~/.bashrc.bak を作成してください。

```
$ touch ~/.bashrc
$ cp -f ~/.bashrc ~/.bashrc.bak
```

途中で何か間違えたら、次のコマンドで ~/.bashrc を元に戻すことができます。

```
$ cp -f ~/.bashrc.bak ~/.bashrc
```

ターミナルで以下のコマンドを順に実行してください。

```
$ sudo apt-get update
$ sudo apt-get -y install build-essential
$ sudo apt-get -y install git zlib1g-dev libssl-dev libreadline-dev
$ sudo apt-get -y install libxml2-dev libxslt-dev libsqlite3-dev
```

```
$ git clone git://github.com/sstephenson/rbenv.git ~/.rbenv
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
$ source ~/.bashrc
$ mkdir -p ~/.rbenv/plugins
$ cd ~/.rbenv/plugins
$ git clone git://github.com/sstephenson/ruby-build.git
```

echo で始まる二つのコマンドは間違いやすいので、特に注意して入力してください。

> *apt-get* コマンドで rbenv パッケージと ruby-build パッケージをインストールすることもできます。しかし、これらのパッケージは中身が古く、最新の *ruby* をインストールできません。本文に記載した手順で *rbenv* と *ruby-build* をインストールしてください。

9.3 *rbenv* コマンドの使い方

インストール可能な *ruby* のバージョン一覧表示

次のコマンドを実行すると、インストール可能な *ruby* のバージョンの一覧が表示されます。

```
$ rbenv install --list
```

その中に 2.4.1 というバージョン番号（2017 年 4 月 5 日時点での最新版）が含まれていることを確認してください。

> ■コラム: ruby-build プラグインの更新
>
> 新たにリリースされたバージョンの *ruby* をインストールしたい場合、ruby-build プラグイン ruby-build プラグインを更新する必要があります。

以下のコマンドをターミナルで順に実行してください。

```
$ cd ~/.rbenv/plugins/ruby-build
$ git pull
```

ターミナルで rbenv install --list コマンドを実行して、インストールしたいバージョンがリストに含まれているかどうかを確認してください。なお、*ruby* の新しいバージョンがリリースされてから、それが ruby-build プラグインに取り込まれるまでには多少の時間差があります。

バージョンを指定して *ruby* をインストールする

バージョン 2.4.1 の *ruby* をインストールするには、ターミナルで次のコマンドを実行してください。

```
$ rbenv install 2.4.1
```

インストールされた *ruby* のバージョン一覧表示

次のコマンドを実行すると、インストール済みの *ruby* のバージョンの一覧が表示されます。

```
$ rbenv versions
```

コマンドの結果は、例えば次のように表示されます。

```
* system (set by /Users/kuroda/.rbenv/version)
  2.4.1
```

アスタリスク記号（*）のついたバージョンが現在選択されています。ここで system というのは、「あなたのコンピュータにもとからあった *ruby*」という意味です。

ruby のバージョンを選択

ruby のバージョンを選択するには、次に挙げる *rbenv* の三つのサブコマンドの一つを利用します。

- `shell`
- `local`
- `global`

`shell` サブコマンド

次のコマンドを実行すると、*ruby* のバージョン 2.4.1 が選択されます。

```
$ rbenv shell 2.4.1
```

本当に選択されたかどうかを確認するため、次のコマンドを実行してみましょう。

```
$ 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 macOS の場合、2 行目が Ubuntu の場合）。

```
ruby 2.4.1p111 (2017-03-22 revision 58053) [x86_64-darwin15]
ruby 2.4.1p111 (2017-03-22 revision 58053) [x86_64-linux]
```

なお、バージョン番号の後ろに付いている `p0` は「パッチレベル」という番号です。この番号は、*ruby* の各バージョンに対してバグ修正パッチ（修正データ）が通算で何回適用されたかを表します。私たちがこの番号を気にする必要はありません。

`rbenv shell` コマンドの効果は、ターミナルを閉じるか次のコマンドを実行すると消えます。

```
$ rbenv shell --unset
```

■コラム: 環境変数 RBENV_VERSION

rbenv shell コマンドは、環境変数 RBENV_VERSION にバージョン番号をセットすることで *ruby* のバージョンを選択しています。

「環境変数」は実行中の個々のプログラムに付随するデータであり、プログラムが終了すると消えてしまいます。ターミナル終了時に rbenv shell コマンドの効果がなくなるのは、そのためです。

local サブコマンド

次のコマンドを実行すると、*ruby* のバージョン 2.4.1 が選択されます。

```
$ rbenv local 2.4.1
```

このコマンドは、カレントディレクトリに.ruby-version という名前のテキストファイルを作り、そこに *ruby* のバージョン番号を書き込みます。

.ruby-version の効果はサブディレクトリにも及びます。例えば、projects ディレクトリの下に x と y という二つのサブディレクトリがあるとします。ここで、projects ディレクトリの直下に.ruby-version というファイルがあり、そこに 2.3.3 と書いてあれば、x ディレクトリでも y ディレクトリでも *ruby* のバージョン 2.3.3 が選択されます。

もし、y ディレクトリではバージョン 2.4.1 の *ruby* を使いたければ、y ディレクトリで rbenv local 2.4.1 コマンドを実行します。この結果、2.4.1 という内容を持つファイル.ruby-version が作られ、y ディレクトリ以下でバージョン 2.4.1 の *ruby* が選択されるようになります。

なお、shell サブコマンドによるバージョン選択の効果は、local サブコマンドよりも強い点に気をつける必要があります。つまり、カレントディレクトリにファイル.ruby-version があっても、環境変数 RBENV_VERSION がセットされていればそちらが優先されます。

global サブコマンド

次のコマンドを実行すると、*ruby* のバージョン 2.4.1 が選択されます。

```
$ rbenv global 2.4.1
```

このコマンドは、~/.rbenv/ディレクトリにある version という名前のテキストファイルに *ruby* のバージョン番号を書き込みます。

global サブコマンドによるバージョン選択の効果は、その名の通りグローバルに（全体的に）適用されます。ただし、あるディレクトリにファイル .ruby-version があれば、そのディレクトリ以下ではそちらが優先されます。

つまり、バージョンを選択する三つのサブコマンドの「強さ」は次の式で表現されることになります。

```
shell > local > global
```

rbenv global コマンドによるバージョン選択を初期状態に戻すには、次のコマンドを実行します。

```
$ rbenv global system
```

結果として、「あなたのコンピュータにもとからあった *ruby*」が選択されます。

9.4 *rbenv* の更新

Mac macOS の場合

rbenv のバージョン番号は次のコマンドで調べられます。

```
$ rbenv --version
```

2017 年 4 月 5 日現在、バージョン番号は次のように表示されます。

```
rbenv 1.1.0
```

rbenv を最新版に更新するには、ターミナルで以下のコマンドを順に実行してください。

```
$ brew update
$ brew upgrade rbenv ruby-build
$ cd ~/.rbenv/plugins/ruby-build
$ git pull
```

Ubuntu の場合

rbenv のバージョン番号は次のコマンドで調べられます。

```
$ rbenv --version
```

2017 年 4 月 5 日現在、バージョン番号は次のように表示されます。

```
rbenv 1.1.0-2-g4f8925a
```

rbenv を最新版に更新するには、ターミナルで以下のコマンドを順に実行してください。

```
$ cd ~/.rbenv
$ git pull
$ cd plugins/ruby-build
$ git pull
```

第 20 章

React

本章では、JavaScrip フレームワーク React の概要を説明したのち、React を Webpacker を通じて Rails アプリケーションに組み込む手順を解説します。

20.1 React とは

React（リアクト）は、米 Facebook 社が中心となって開発されているオープンソースの JavaScript フレームワークです。2013 年に公開され、2015 年頃から大きな注目を集めるようになりました。

仮想 DOM（virtual DOM）と呼ばれる独特のレンダリング機構のおかげで、ネイティブアプリのようなユーザーインターフェースを効率的に実装することができます。

目立つ特徴として、**JSX** と呼ばれる独自の構文を持つ点が挙げられます。JSX は JavaScript に対する拡張構文です。JSX の構文が使われている例を次に示します。

```
React.render(
  <p>Hello, world!</p>,
  document.getElementById('message')
);
```

一見すると普通の JavaScript プログラムですが、2 行目に突如として HTML の断片（`<p>Hello, world!</p>`）が現れます。このような書き方を許すのが JSX です。

React はしばしば **Redux**（リダックス）という別の JavaScript ライブラリと組み合わせて使われます。このライブラリを用いると Web アプリケーションの「状態（state)」を洗練された方法で管理することができます。

20.2 React の導入

では、前章（第 19 章）終了時点での AirBoy のソースコードをコピーして、それに React を組み込んでみましょう。ターミナルで以下のコマンドを順に実行してください。

```
$ cd ~/projects
$ cp -r air_boy air_boy_react
$ cd air_boy_react
$ rails webpacker:install:react
```

結果として、`package.json` に `react` という項目が追加され、`node_modules` ディレクトリに React がインストールされます。

> 執筆時点（2017 年 4 月）における、React の最新バージョンは `15.4.2` です。

また、`app/javascript/packs` ディレクトリの下に `top_react.jsx` というファイルが作られます。これは、JSX 構文を用いた JavaScript プログラムのサンプルです。

20.3 React プログラミングの例

続いて、React が正常に機能するかどうかを確認するため、簡単な JavaScript プログラムを JSX 構文で書いてみます。

準備作業

まず、ターミナルで次のコマンドを実行します。

```
$ rails g controller top counter
```

すると、ターミナルに次のような結果が表示されます。

```
       create  app/controllers/top_controller.rb
        route  get 'top/counter'
      invoke  erb
...
       create      app/assets/javascripts/top.coffee
      invoke    scss
       create      app/assets/stylesheets/top.scss
```

JSX ファイルを作る

次に、app/javascript/packs ディレクトリに新規ファイル counter.jsx を次の内容で作成します。

app/javascript/packs/counter.jsx

```
1  import React from 'react'
2  import ReactDOM from 'react-dom'
3
4  class Counter extends React.Component {
5    constructor(props) {
6      super(props)
7      this.state = { counter: 0 }
8      this.increment = this.increment.bind(this)
9    }
10
11   increment() {
12     this.setState({ counter: this.state.counter + 1 })
13   }
14
15   render() {
16     return <div>
17       <div>Counter = {this.state.counter}</div>
18       <button onClick={this.increment}>Click me!</button>
19     </div>
20   }
21 }
```

```
22
23  document.addEventListener('DOMContentLoaded', () => {
24    ReactDOM.render(
25      <Counter />,
26      document.getElementById('counter')
27    )
28  })
```

webpack-watcher の起動

ターミナルで次のコマンドを実行します。

```
$ bin/webpack-watcher
```

すると、ターミナルに図 20.1 のようなメッセージが出力されて停止しますので、そのままにしておきます。

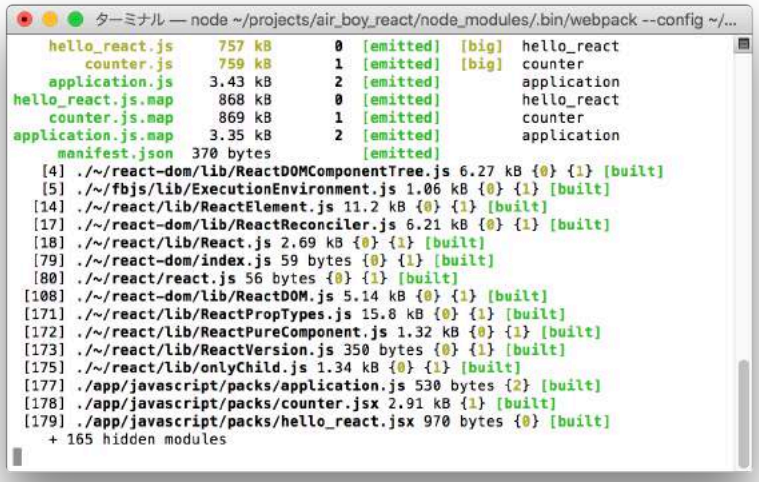

図 20.1　webpack-watcher を起動

webpack-dev-server の起動

ここで、ターミナルをもうひとつ開きます。そして、以下のコマンドを順に実行してください。

```
$ cd ~/projects/air_boy_react
$ bin/webpack-dev-server
```

すると、ターミナルに図 20.2 のようなメッセージが出力されて停止しますので、そのままにしておきます。

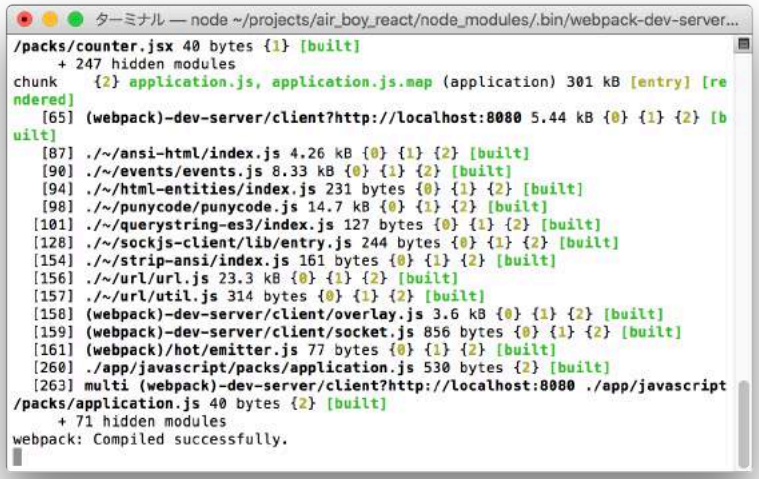

図 20.2　webpack-dev-server を起動

HTML テンプレートの書き換え

次に、土台となる Web ページのテンプレートを書き換えます。テキストエディタで app/views/top ディレクトリにある counter.html.erb を次のように

書き換えてください。

app/views/top/counter.html.erb

```
1 - <h1>Top#counter</h1>
2 - <p>Find me in app/views/top/counter.html.erb</p>
1 + <div id="counter"></div>
2 + <%= javascript_pack_tag "counter" %>
```

Rails サーバーの起動

さらに、別のターミナルを開き、以下のコマンドを順に実行して、Rails サーバーを起動します。

```
$ cd ~/projects/air_boy_react
$ rails s
```

動作確認

ブラウザで http://localhost:3000/top/counter を開くと、図 20.3 のような画面が現れます。

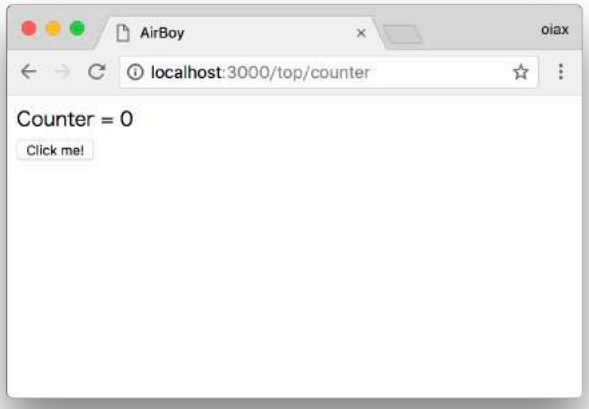

図 20.3　ブラウザによる動作確認

マウスで「Click me!」ボタンを数回クリックしてみてください。クリックするたびに「Counter =」の右側の数字が 1 ずつ増えていけば成功です。