第 9 章

Bootstrap と Font Awesome

本章では、Bootstrap と Font Awesome を導入することにより、SimpleGreeter の外観を整えます。

9.1 Bootstrap の導入

CSS/JavaScript ライブラリ **Bootstrap** を用いると、パソコン、スマートフォン、タブレット PC など画面サイズの異なる端末に対応した Web ページを効率的に制作できるようになります。SimpleGreeter に Bootstrap を導入しましょう。

> 本書では Bootstrap のバージョン 4（Bootstrap 4）を採用します。本書の最終更新時点（2017 年 1 月）では Bootstrap 4 はまだ正式にリリースされていません。今後、仕様が変更される可能性があります。

テキストエディタで `Gemfile` を次のように書き換えてください。

Gemfile

```
  :
49    gem 'spring-watcher-listen', '~> 2.0.0'
50  end
51 +
```

```
52 + gem 'bootstrap', '4.0.0.alpha6'
53 + gem 'tether-rails'
```

Gem パッケージ bootstrap と tether-rails をインストールします。後者は、Bootstrap が依存する JavaScript ライブラリ Tether を使うためのパッケージです。

ターミナルで Ctrl-C を入力して Rails サーバーを止めてから、Gem パッケージ群をインストールします。

```
$ bundle
```

app/assets/stylesheets ディレクトリの application.css を削除します。

```
$ 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

```
1  @import "bootstrap";
2  @import "*";
```

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><%= @document_title %></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。

9.2 Card コンポーネント

Web デザイナーたちが Bootstrap を採用する第一の目的は、レスポンシブ Web デザインを実現することです。すなわち、ブラウザの画面サイズに応じてレイアウトを最適化することです。しかし、Bootstrap が用意する多彩な**コンポーネント**（component） を使うという目的もあります。

本章では、そのひとつ **Card コンポーネント**を紹介します。

基本的な構成

app/views/hello ディレクトリの show.html.erb を次のように変更してください。

app/views/hello/show.html.erb

```
:
4 - <p>Hello, <%= @name %>!</p>
4 + <div class="card">
5 +   <div class="card-block">
6 +     <p>Hello, <%= @name %>!</p>
7 +   </div>
```

```
8 + </div>
```

変更点は以下のとおりです：

1. p 要素を div 要素で二重に囲んだ。
2. 外側の div 要素の class 属性に card を指定した。
3. 内側の div 要素の class 属性に card-block を指定した。

div 要素は、いわば「透明な箱」のような存在です。「段落」という意味を持つ p 要素と異なり、特別な意味を持ちません。主に、スタイルの適用範囲を区切るために用いられます。

class 属性を用いると、HTML の要素に**クラス**を設定できます。クラスとはスタイルに名前をつけたものと考えてください。ただし、クラスには他の役割もあります。

Bootstrap は数多くのクラスを定義しています。例えば、card クラスは「四辺を薄い灰色の角丸枠線で囲む」というスタイルを持ちます。また、card-block クラスには「四辺のパディングの幅を 1.25rem にする」というスタイルが設定されています。

Rails サーバーを起動してブラウザをリロードすると、図 9.1 のような画面に切り替わります。

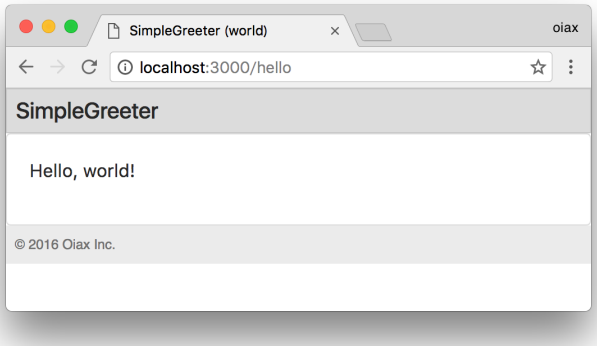

図 9.1　Card コンポーネント（初期状態）

このように Bootstrap を導入すると、スタイルシートを書かなくても HTML 文書の外観を変えることができます。

card-text クラス

さて、図 9.1 を注意深く観察すると、「Hello, world!」というメッセージの上下のマージンが均等でないことに気付きます。しかし、show.html.erb を次のように変更すればこの問題が解消されます。

app/views/hello/show.html.erb

```
:
5     <div class="card-block">
6 -     <p>Hello, <%= @name %>!</p>
6 +     <p class="card-text">Hello, <%= @name %>!</p>
7     </div>
```

card-text クラスは面白い効果を持ちます。これは基本的には何のスタイルも設定しません。ただし、親要素の最後の子要素である場合は、下辺のマージンが 0 に設定されます。

Bootstrap はデフォルトで p 要素の下辺のマージンを 1rem に設定しています。複数の段落が重なった時に段落と段落の間を少し空けるためです。しかし、最後の段落ではこの 1rem のマージンが邪魔になります。

ブラウザをリロードすると 図 9.2 のような画面に切り替わります。

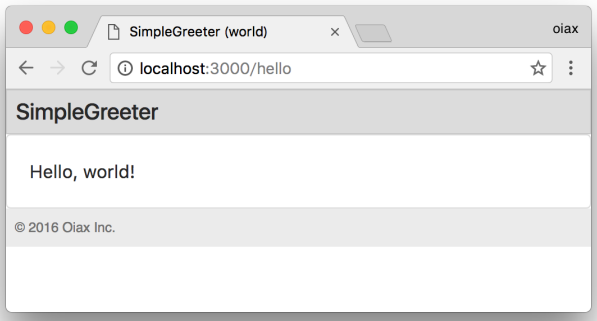

図 9.2　Card コンポーネント（card-text クラスを適用）

マージンの調整

現状の Card コンポーネントは上下左右の辺にマージンがないため「カード」のように見えません。そこで、show.html.erb を次のように変更します。

app/views/hello/show.html.erb

```
  :
4 - <div class="card">
4 + <div class="card m-3">
  :
```

class 属性の値を空白文字で区切れば、ひとつの要素に複数のクラスを与えることができます。外側の div 要素に対して m-3 というクラスを追加しました。

さて、m-3 のような英字と数字をハイフンで連結したクラスは、間隔 (spacing) の設定に使われます。

1 文字目の m はマージンを意味します。この文字を p で置き換えると、パディングを設定するクラスになります。

2 文字目の数字は間隔の幅を表します。0 が 0、1 が 0.25rem、2 が 0.5rem、3 が 1rem、 4 が 1.5rem、 5 が 3rem を意味します。

「上辺のみ」あるいは「左辺と右辺のみ」のように対象となる辺を絞ってマージ

ンあるいはパディングを設定したい場合は、m-t-1 のように 2 文字目で対象となる辺を表します（表 9.1）。

表 9.1　間隔設定クラスの 2 文字目の意味

文字	対象となる辺
t	上辺
r	右辺
b	下辺
l	左辺
x	左辺と右辺
y	上辺と下辺

ここでは、Card コンポーネントの四辺の余白を 1rem に揃えています。その結果、ブラウザの表示は 図 9.3 のようになります。

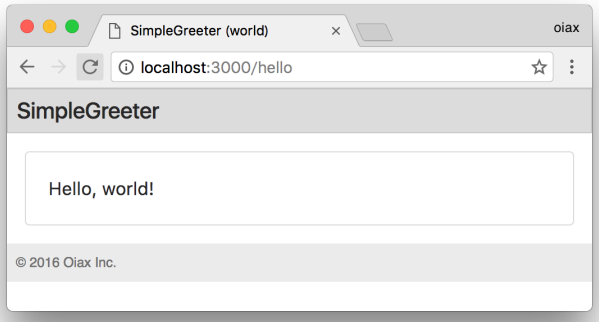

図 9.3　Card コンポーネント（マージンを調整）

9.3　背景色、テキストの配置、フォントサイズを調整

さらに外観の調整を続けましょう。show.html.erb を次のように変更してください。

app/views/hello/show.html.erb

```
  :
  4   <div class="card m-3">
  5 -   <div class="card-block">
  5 +   <div class="card-block card-inverse card-success">
  6 -     <p class="card-text">Hello, <%= @name %>!</p>
  6 +     <p class="card-text text-center lead">Hello, <%= @name %>!</p>
  7     </div>
  8   </div>
```

card-inverse クラスは、内側の要素の文字色を半透明の白にします。ただし、内側の要素に card-text クラスが指定されている必要があります。

card-success クラスは背景色を濃い緑（#5cb85c）にします。success の部分を他の名前で置き換えれば、様々な色を背景色として利用できます（表 9.2）。

表 9.2　背景色を設定するクラス

クラス名	背景色	16 進数表示
card-primary	青	#0275d8
card-success	緑	#5cb85c
card-info	水色	#5bc0de
card-warning	オレンジ色	#f0ad4e
card-danger	赤	#d9534f

Bootstrap は“blue”とか“red”などの具体的な名前ではなく“primary”とか“danger”などの抽象的な名前を用いてクラスを定義しています。表 9.2 の 3 列目に書いてあるとおり、これらの色はすべて中間色なので、具体的な色の名前で呼ぶと紛らわしいからです。なお、これらの名前は「アダ名」みたいなものです。人間が覚えやすいように仮に付けられているに過ぎません。英語の“danger”は「危険」を意味しますが、赤い色を危険なものにしか使っていけないわけではありません。

text-center は、テキストを中央寄せするクラスで、lead はフォントサイズを少し大きめ（1.25rem）にするクラスです。

ブラウザをリロードすると 図 9.4 のような画面に切り替わります。

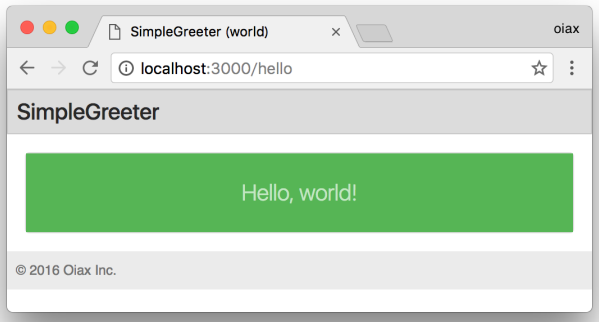

図 9.4　Card コンポーネント（背景色とフォントサイズを調整）

グレースケールのディスプレイやペーパーバック版で本書をお読みの方には分かりませんが、図 9.4 で「Hello, world!」というメッセージは緑色の背景の上に描かれています。

また、Chrome のデベロッパーツール（付録 A 参照）を利用して、SimpleGreeter がスマートフォンでどのように表示されるのかを確認してください。

9.4　Font Awesome の導入

Font Awesome を利用すると HTML タグを書くだけで 図 9.5 のようなさまざまなアイコンを Web ページの中で使用することができます。

図 9.5　Font Awesome のアイコンの例

利用可能なアイコンのリストは `http://fontawesome.io/icons/` で閲覧できます。

導入の手順はとても簡単です。まず、テキストエディタで Gemfile に 1 行追加します。

Gemfile

```
  :
52  gem 'bootstrap', '~> 4.0.0.alpha6'
53  gem 'tetehr-rails'
54 + gem 'font-awesome-sass'
```

ターミナルで Ctrl-C を入力して Rails サーバーを止めてから、追加された Gem パッケージ font-awesome-sass をインストールします。

```
$ bundle
```

app/assets/stylesheets ディレクトリの application.scss を編集します。

app/assets/stylesheets/application.scss

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

9.5 Font Awesome の利用

これで Font Awesome が導入されました。では、使ってみましょう。

app/views/hello ディレクトリの show.html.erb を次のように変更します。

app/views/hello/show.html.erb

```
  :
6 -     <p class="card-text text-center lead">Hello, <%= @name %>!</p>
6 +     <p class="card-text text-center lead">
7 +       <i class="fa fa-smile-o"></i>
8 +       Hello, <%= @name %>!
9 +     </p>
  :
```

class 属性に fa クラスを指定した i 要素が Font Awesome によってアイコン表示に変えられます。i 要素の内容は空にしてください。

fa-で始まるアイコン固有のクラス名を class 属性に指定することにより、表示するアイコンを選択します。ここでは「smile-o」という名前のアイコンを表示したいので fa-smile-o というクラス名を指定しました。

Rails サーバーを起動しブラウザをリロードすると、図 9.6 のように「Hello, world!」メッセージの前に「笑顔」のアイコンが表示されます。

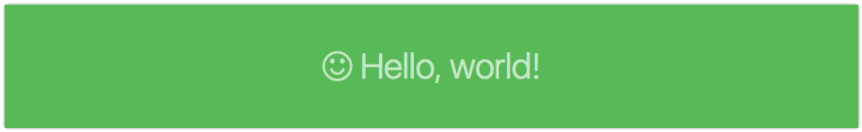

図 9.6　Font Awesome を使ってアイコンを表示