第 15 章

ビューモジュール

この章では、Phoenix 独特の仕組みであるビューモジュールを用いて、HTML 文書のタイトルを動的に生成する方法について解説します。

15.1　トップページの作成

ModestGreeter の開発を再開します。トップページを作りましょう。「RAVT」の順で作業を進めていきます。

まず、経路を設定します。テキストエディタで web/router.ex を次のように変更してください。

web/router.ex

```
  :
12    scope "/", ModestGreeter do
13      pipe_through :browser
14
15 +    get "/", TopController, :index
16      get "/hello/:name", HelloController, :show
17      get "/hello", HelloController, :show
18    end
19  end
```

次に、top コントローラのソースコードを新規作成し、index アクションを作ります。

web/controllers/top_controller.ex (New)

```
1  defmodule ModestGreeter.TopController do
2    use ModestGreeter.Web, :controller
3
4    def index(conn, _params) do
5      render conn, "index.html"
6    end
7  end
```

ビューモジュールを作成します。

web/views/top_view.ex (New)

```
1  defmodule ModestGreeter.TopView do
2    use ModestGreeter.Web, :view
3  end
```

top コントローラで使用するテンプレートを置くディレクトリを作成します。

```
$ mkdir -p web/templates/top
```

index アクションのためのテンプレートを作成してください。

web/templates/top/index.html.eex (New)

```
1  <div class="jumbotron m-1">
2    <h1 class="display-3">ModestGreeter</h1>
3    <p class="lead">ModestGreeter へようこそ！ </p>
4  </div>
```

Bootstrap の **Jumbotron コンポーネント**を利用しています。ページの広い範囲を使って Web サイトの名称、ロゴ、キャッチコピーなどを目立たせるためのコンポーネントです。jumbotron クラスを指定した div 要素が Jumbotron の範囲となります。m-1 クラスについてはすでに 第 11 章 で紹介しました。要素の四辺のマージンを 1rem にするクラスです。

5 行目の display-3 は、フォントサイズを調整するためのクラスです。display-1 から display-4 までのクラスが定義されており、フォントサイズ

は順に 6rem、5.5rem、4.5rem、3.5rem となります。クラス名に含まれる数字が大きくなるほど、フォントサイズが小さくなる点に注意してください。

さあ、うまくトップページが表示されるでしょうか。mix phoenix.server コマンドでサーバを起動し、ブラウザで http://localhost:4000 を開いてみましょう。

あらら、ダメですね。エラーが出てしまいました（図 15.1）。

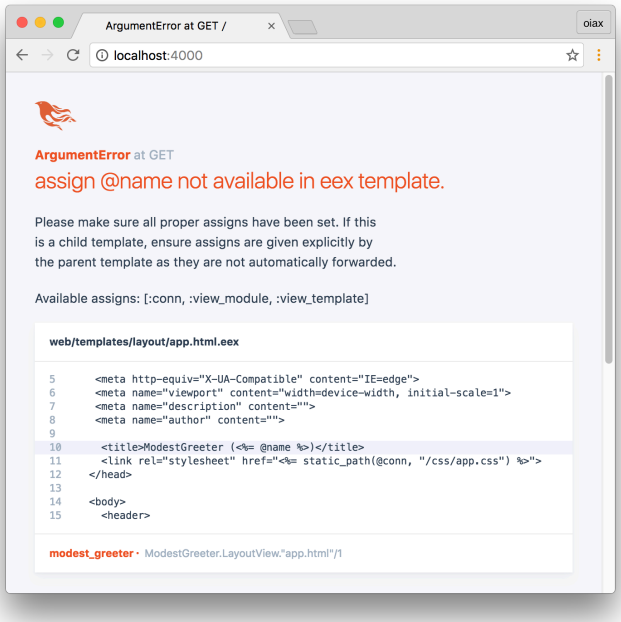

図 15.1　トップページでエラー発生

エラーメッセージには「@name が得られない（not available）」と出ています。原因は、web/templates/layout ディレクトリにあるレイアウトテンプレート app.html.eex の 10 行目です。

```
    <title>ModestGreeter (<%= @name %>)</title>
```

テンプレートに現れる @ はなんでしたっけね。@ はマクロであり、@name と書

くことでアクションの render 関数に指定されたキーワードリストの :name キーの値を参照できます。確かに、さきほど作成した index アクションでは render 関数にキーワードリストを指定していません。

とすれば、index アクションで render 関数の第 3 引数を指定することでエラーを回避できます。top_controller.ex を次のように書き換えてください。

web/controllers/top_controller.ex

```
  :
4     def index(conn, _params) do
5 -     render conn, "index.html"
5 +     render conn, "index.html", name: "Top"
6     end
7   end
```

これでトップページのタイトルは「ModestGreeter (Top)」になります。しかし、筆者としてはトップページのタイトルは単に「ModestGreeter」としたいと考えています。どうすればいいでしょうか。

15.2 document_title 関数の定義

ここで登場するのが**ビューモジュール**（view module）です。まず、いま行ったばかりの変更を元に戻します。

web/controllers/top_controller.ex

```
  :
5 -     render conn, "index.html", name: "Top"
5 +     render conn, "index.html"
  :
```

そして、HelloView モジュールのソースコードを次のように書き換えてください。

web/views/hello_view.ex

```
1   defmodule ModestGreeter.HelloView do
2     use ModestGreeter.Web, :view
```

```
 3 +
 4 +   def document_title(assigns) do
 5 +     "ModestGreeter (#{assigns.name})"
 6 +   end
 7   end
```

document_title という名前の関数を定義しました。説明は後回しにします。続いて、TopView モジュールのソースコードを次のように書き換えてください。

web/views/top_view.ex

```
 1   defmodule ModestGreeter.TopView do
 2     use ModestGreeter.Web, :view
 3 +
 4 +   def document_title(_assigns) do
 5 +     "ModestGreeter"
 6 +   end
 7   end
```

引数を 1 個取りますが、関数の本体で使わないので仮引数の名前をアンダースコア（_）で始めています。

では、これらの関数をテンプレートの中で使ってみます。まず、TopController の index アクションのテンプレートを次のように書き換えます。

web/templates/top/index.html.eex

```
 1   <div class="jumbotron m-1">
 2 -   <h1 class="display-3">ModestGreeter</h1>
 2 +   <h1 class="display-3"><%= document_title(assigns) %></h1>
 3     <p class="lead">ModestGreeter へようこそ！ </p>
 4   </div>
```

そして、レイアウトテンプレート app.html.eex を次のように書き換えます。

web/templates/layout/app.html.eex

```
10 -     <title>ModestGreeter (<%= @name %>)</title>
10 +     <title><%= @view_module.document_title assigns %></title>
```

ブラウザの画面は図 15.2 のように変化します。エラーが解消しました。ブラウザのタブに表示されているタイトルも「ModestGreeter」となっています。

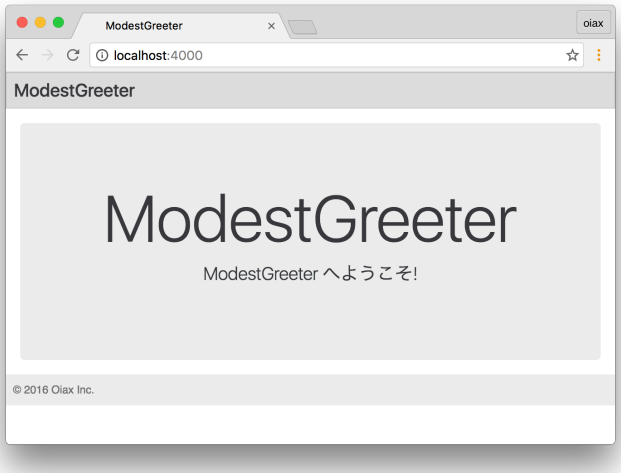

図 15.2　ModestGreeter のトップページ

念のため、アドレスバーの URL を `http://localhost:4000/hello` に変えて、従来通りの表示が保たれていることを確認してください（画面キャプチャ省略）。

15.3　document_title 関数の説明

では、ふたつの `document_title` 関数について説明します。まず、`HelloView.document_title` 関数のコードをご覧ください。

```
def document_title(assigns) do
  "ModestGreeter (#{assigns.name})"
end
```

仮引数 `assings` には、`render` 関数の第 3 引数に与えられたキーワードリストの要素をすべて持つマップがセットされます。キーワードリストそのものが渡っ

てくるのではなく、マップに変換されています。そのため、assigns.name のようにドット記号を用いて値にアクセスできます。この関数は "ModestGreeter (Alice)" のような文字列を返します。

次に、TopView.document_title 関数のコードをご覧ください。

```
def document_title(_assigns) do
  "ModestGreeter"
end
```

この関数は常に "ModestGreeter" という文字列を返します。引数に与えられたマップは使用しないので、仮引数の名前をアンダースコアで始めています。

続いて、web/templates/top/index.html.eex の変更箇所をご覧ください。

```
<h1 class="display-3"><%= document_title(assigns) %></h1>
```

TopView モジュールの document_title 関数を呼び出しています。TopView は TopController に対応するビューモジュールであるため、モジュール名なしで関数を呼び出せます。

テンプレートの中では常に assigns という変数を参照することができます。この変数にはマップがセットされており、このマップは render 関数の第 3 引数に与えられたキーワードリストの要素をすべて持っています。

最後に、レイアウトテンプレートの変更箇所をご覧ください。

```
    <title><%= @view_module.document_title assigns %></title>
```

@view_module は、現在使われているコントローラに対応するビューモジュールを返します。つまり、現在のコントローラが TopController なら TopView を、HelloController なら HelloView を返します。したがって、@view_module.document_title で、どちらかのビューモジュールの document_title 関数を指し示すことになります。

15.4 「このサイトについて」ページの作成

続いて、「このサイトについて」ページを作ります。URL パスは /about とし、TopController の about アクションでリクエストを受けることにします。まず、経路設定です。

web/router.ex

```
 :
12    scope "/", ModestGreeter do
13      pipe_through :browser
14
15      get "/", TopController, :index
16 +    get "/about", TopController, :about
17      get "/hello/:name", HelloController, :show
18      get "/hello", HelloController, :show
19    end
20  end
```

TopController に about アクションを追加します。

web/controllers/top_controller.ex

```
 1  defmodule ModestGreeter.TopController do
 2    use ModestGreeter.Web, :controller
 3
 4    def index(conn, _params) do
 5      render conn, "index.html"
 6    end
 7 +
 8 +  def about(conn, _params) do
 9 +    render conn, "about.html"
10 +  end
11  end
```

ビューモジュールの変更は後回しにします。テンプレートのファイルを次のような内容で新規作成してください。

app/templates/top/about.html.eex (New)

```
1  <div class="m-1">
2    <h1>このサイトについて</h1>
3    <p>ModestGreeter は『Elixir/Phoenix 初級①』の学習用 Phoenix アプリです。</p>
4  </div>
```

ブラウザのアドレスバーに http://localhost:4000/about と入力すると、図 15.3 のような画面になります。

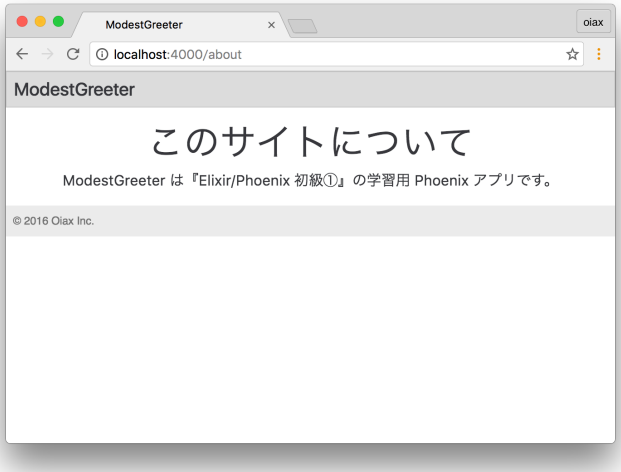

図 15.3　「このサイトについて」を表示

15.5　case による条件分岐

ビューモジュールの変更に進む前に、Elixir での条件分岐について簡単に説明します。いろんな書き方があるのですが、本巻では case マクロを使った方法を紹介します。

> Elixir の条件分岐に関しては、次巻『初級②』で改めて詳しく解説します。

次の Elixir コードをご覧ください。

```
n = 1
case n do
  1 -> IO.puts "A"
  2 -> IO.puts "B"
  _ -> IO.puts "C"
end
```

このコードを Elixir スクリプトとして実行すると A という文字が出力されます。変数 n の値を 1 でも 2 でもない値、例えば 5 に変えて実行すると C が出力されます。

case マクロを用いると、対象となる式に対して複数個の値を列挙し、式がどの値と一致するかで処理を分岐させることができます。対象となる式は case と do に記述します。そして、do と end の間では、値と式の組を列挙します。その際、矢印記号（->）の左側に値、右側に式を置きます。

このコードでは、対象となる式 n に対して、1、2、_ という値を用意しました。式 n の値が 1 に等しければ、IO.puts "A" という式が評価され、2 に等しければ、IO.puts "B" という式が評価されます。最後の _ は「任意の値」を意味します。そのため n の値が 1 でも 2 でもないときに、式 IO.puts "C" が評価されます。

Ruby をご存知の方は、次の Ruby コードと比較すると分かりやすいでしょう。

```
n = 1
case n
  when 1 then puts "A"
  when 2 then puts "B"
  else puts "C"
end
```

15.6　ページタイトルの切り替え

さて、「このサイトについて」ページのタイトルはトップページと同じです。タイトルをアクションごとに変えるにはどうすればいいでしょうか。

実は、document_title 関数に引数として渡されるマップには、render 関数から渡されるキーワードリストの要素以外にもいろんな情報が含まれています。そ

のひとつがテンプレートの名前です。この情報は :view_template というキーでマップから取り出せます。それを見て処理を分ければいいのです。

では、TopView モジュールのソースコードを次のように書き換えてください。

web/views/top_view.ex

```
 1   defmodule ModestGreeter.TopView do
 2     use ModestGreeter.Web, :view
 3
 4 -   def document_title(_assigns) do
 4 +   def document_title(assigns) do
 5 -     "ModestGreeter"
 5 +     case assigns.view_template do
 6 +       "about.html" -> "ModestGreeter - このサイトについて"
 7 +       _ -> "ModestGreeter"
 8 +     end
 9     end
10   end
```

前節で学んだ case マクロを用いて処理を分岐させています。比較の対象となる式は assigns.view_template です。その値が "about.html" と一致する場合は、文字列 "ModestGreeter - このサイトについて" を関数の戻り値とします。そうでない場合は、文字列 "ModestGreeter" を返します。