第 8 章

プライベートメソッドと部分テンプレート

この章のテーマはソースコードの改善です。同一の、あるいは非常に類似したコードが複数箇所に存在するとき、それらをプライベートメソッドや部分テンプレートとして抜き出すとソースコードが読みやすくなります。これは、Rails アプリケーション開発における基本的なテクニックです。

8.1 プライベートメソッド

この節では、PicoPlanner の開発を一時中断して、Ruby 言語自体の学習をします。まず、lib ディレクトリの下にサンプルプログラムを置くためのディレクトリ lessons を作成しましょう。

```
$ mkdir -p lib/lessons
```

lib/lessons ディレクトリに新規ファイル greeting.rb というファイルを次の内容で作成してください。

lib/lessons/greeting.rb (New)

```
1  class Greeting
2    def hello
3      puts message
```

```
  end

  private def message
    'Hello'
  end
end

g = Greeting.new
g.hello
```

ターミナルで次のコマンドを実行してください。

```
$ ruby lib/lessons/greeting.rb
```

すると、ターミナルに Hello という文字列が出力されます。

Ruby のメソッドは、**パブリックメソッド**（public method）、**プロテクテッドメソッド**（protected method）、**プライベートメソッド**（private method）に分類されます。def の前に何も書かなければパブリックメソッドになります。この観点から見たメソッドの違いを**可視性**（visibility）と呼びます。

> 本シリーズでプロテクテッドメソッドを使わない予定なので、ここでは解説しません。興味を持たれた方は、付録 C を参照してください。

上記の例では、Greeting クラスの hello メソッドはパブリックメソッド、message メソッドはプライベートメソッドです。

さて、プライベートメソッドには、同じクラスやサブクラスで定義されたメソッドの中からしか呼び出せない、という制限があります。この点を確かめるため、greeting.rb を次のように書き換えてください。

lib/lessons/greeting.rb

```
  :
g = Greeting.new
g.hello
puts g.message
```

もう一度このプログラムを実行すると、最後の行の puts g.message で次のようなエラーメッセージが出ます。

```
lib/lessons/greeting.rb:13:in `<main>': private method `message' called >
for #<Greeting:0x0055633f12ef10> (NoMethodError)
```

また、プライベートメソッドには、**レシーバ形式**で呼び出せないという別の制限があります。オブジェクトの後ろにドット記号とメソッド名を書いて呼び出すのがレシーバ形式です。

greeting.rb を次のように書き換えてください。

lib/lessons/greeting.rb

```
1   class Greeting
2     def hello
3 -     puts message
3 +     puts self.message
4     end
:
```

このプログラムを実行すると、次のようなエラーメッセージが出ます。

```
lib/lessons/greeting.rb:3:in `hello': private method `message' called for >
#<Greeting:0x0056311cd0d8a8> (NoMethodError)
        from lib/lessons/greeting.rb:12:in `<main>'
```

ところで、プライベートメソッドを定義する方法は他にもあります。次の例をご覧ください。

```
class K
  private
  def foo
    'foo'
  end

  def bar
    'Bar'
  end
end
```

2 行目に private とだけ書かかれています。private はクラスメソッドであり、これが引数なしで呼び出されるとそれ以降に定義されるメソッドの可視性が

すべてプライベートになります。したがって、上で定義されたクラス K の foo メソッドと bar メソッドはともにプライベートメソッドになります。

実は、Rails の書籍やブログ記事などでよく紹介されているのはこちらの書き方です。def キーワードの左に private メソッドを置く書き方ができるようになったのは、Ruby 2.1（2013 年 12 月リリース）からなので、まだ一般に普及していないのです。しかし、本書では新しい書き方を採用します。どのメソッドがプライベートであるのかが、より明確になると考えるからです。

■コラム: メソッド定義はシンボルを返す

Ruby 2.1 以前からクラスメソッド private に対してメソッド名をシンボルで指定して、メソッドの可視性を変更できました。例えば、次のように書けばクラス K の foo メソッドはプライベートメソッドになります。

```
class K
  def foo
    'foo'
  end
  private :foo
end
```

Ruby 2.1 で、def と end で囲まれたメソッド定義がメソッド名をシンボルで返すようになりました。そのおかげで、現在では def キーワードの左に private と書くことができるのです。

8.2 plan_item_params メソッドの定義

さて、plan_items コントローラの create アクションと update アクションのコードを改めてご覧ください。

app/controllers/plan_items_controller.rb

```
  :
21    def create
22      PlanItem.create!(
23        params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
```

```
24      )
25
26      redirect_to :plan_items
27    end
28
29    def update
30      plan_item = PlanItem.find(params[:id])
31      plan_item.update!(
32        params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
33      )
34
35      redirect_to :plan_items
36    end
37  end
```

まったく同一のソースコードが含まれていますね。23 行目と 32 行目です。

```
params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
```

> 26 行目と 35 行目の redirect_to :plan_items も重複していますが、短いコードなのでそのままにしておきます。

このようなケースでは、この部分をプライベートメソッドとして抜き出すとソースコードが読みやすくなります。plan_items_controller.rb を次のように書き換えてください。

app/controllers/plan_items_controller.rb

```
  :
35      redirect_to :plan_items
36    end
37 +
38 +  private def plan_item_params
39 +    params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
40 +  end
41  end
```

一般に、コントローラのメソッドの可視性は次のルールで決めます。

- アクションとして使われるものはパブリック
- そうでないものはプライベート

新たに定義した plan_item_params はアクションとして使わないので、プライベートメソッドとします。

では、plan_item_params メソッドを利用して create アクションのコードを書き換えましょう。

app/controllers/plan_items_controller.rb

```
  :
21    def create
22 -    PlanItem.create!(
23 -      params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
24 -    )
22 +    PlanItem.create!(plan_item_params)
23
24      redirect_to :plan_items
25    end
  :
```

同様に、update アクションも書き換えてください。

app/controllers/plan_items_controller.rb

```
  :
27    def update
28      plan_item = PlanItem.find(params[:id])
29 -    plan_item.update!(
30 -      params[:plan_item].permit(:name, :description, :starts_at, :ends_at)
31 -    )
29 +    plan_item.update!(plan_item_params)
30
31      redirect_to :plan_items
32    end
  :
```

かなり簡潔になりましたね。念のため、予定の新規追加機能と予定の変更機能が従来通り動くことを確認してください。

8.3 部分テンプレート

ERB テンプレートの一部を抜き出したものを**部分テンプレート**（partial）と呼びます。使い方を学ぶために、第 1 章で作成した lessons コントローラに新しいアクション hello を追加しましょう。

config/routes.rb を次のように書き換えてください。

config/routes.rb

```
Rails.application.routes.draw do
  root 'top#index'
  get 'lessons/form' => 'lessons#form'
  get 'lessons/register' => 'lessons#register'
+ get 'lessons/hello' => 'lessons#hello'
  resources :plan_items,
    only: [ :index, :new, :show, :edit, :create, :update ]
end
```

次に、app/controllers ディレクトリの lessons_controller.rb を次のように書き換えます。

app/controllers/lessons_controller.rb

```
:
  def register
    @user_name = params[:user_name]
  end

+ def hello
+ end
end
```

app/views/lessons ディレクトリに新規ファイル hello.html.erb を次の内容で作成します。

app/views/lessons/hello.html.erb (New)

```
1  <div class='card'>
2    <div class='card-block'>
3      <%= render 'content' %>
4    </div>
5  </div>
```

さらに、app/views/lessons ディレクトリに新規ファイル_content.html.erb を次の内容で作成します（ファイル名をアンダースコア（_）で始める点に注意）。

app/views/lessons/_content.html.erb (New)

```
1  <p class='card-text'>Hello, world!</p>
```

この節で使用した CSS クラス card、card-block、card-text は、Bootstrap の Card コンポーネントを生成するためのものです。詳しくは『初級①』第 9 章をご覧ください。

ブラウザで http://localhost:3000/lessons/hello を開くと、図 8.1 のような画面になります。

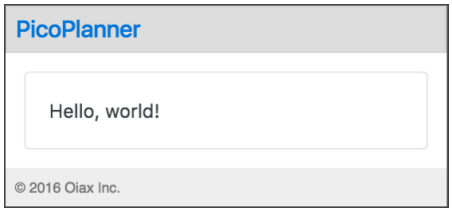

図 8.1　lessons#hello アクション

hello.html.erb の 3 行目をご覧ください。

```
<%= render 'content' %>
```

ERB テンプレートの中で render メソッドを呼び出すと、引数に指定された名前の部分テンプレートが HTML フラグメントに変換された上で、その場所に

埋め込まれます。部分テンプレートのファイル名は、部分テンプレートの名前の前にアンダースコア（_）を付け、拡張子として .html.erb を加えたものになります。上記のケースでは、部分テンプレートのファイル名は_content.html.erb になります。

原則として、部分テンプレートとそれが埋め込まれる先のテンプレートは同じディレクトリに置かれます。ここでは、app/views/lessons ディレクトリの _content.html.erb が、同ディレクトリにある hello.html.erb の 3 行目に埋め込まれました。

他のディレクトリにある部分テンプレートを埋め込みたい場合は、部分テンプレートの名前を app/views ディレクトリからの相対パスで指定します。例えば、app/views/shared ディレクトリの_notes.html.erb を部分テンプレートとして埋め込むなら、次のように記述します。

```
<%= render 'shared/notes' %>
```

8.4　部分テンプレートにローカル変数を渡す

render メソッドの第 2 引数にハッシュを指定すると、そのハッシュの要素がローカル変数として部分テンプレートに渡ります。

hello.html.erb を次のように書き換えてください。

app/views/lessons/hello.html.erb

```
1   <div class='card m-1'>
2     <div class='card-block'>
3 -     <%= render 'content' %>
3 +     <%= render 'content', name: 'Alice' %>
4 +     <%= render 'content', name: 'Bob' %>
5     </div>
6   </div>
```

4 行目の render メソッドの第 2 引数に { name: 'Alice' } というハッシュが指定されています。そのため、部分テンプレートの中ではローカル変数 name で文字列 'Alice' を参照できます。

_content.html.erb を次のように書き換えてください。

app/views/lessons/_content.html.erb

```
1 - <p class='card-text'>Hello, world!</p>
1 + <p class='card-text'>Hello, <%= name %>!</p>
```

ブラウザで http://localhost:3000/lessons/hello を開くと、図 8.2 のような画面になります。

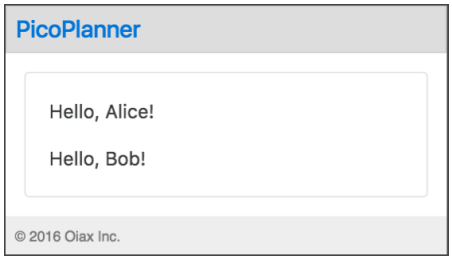

図 8.2　部分テンプレートにパラメータを渡す

_content.html.erb の <%= name %> と書かれた場所に 'Alice' や 'Bob' などの文字列が埋め込まれています。

8.5　フォーム用テンプレートの整理

では、部分テンプレートを用いて、PicoPlanner の ERB テンプレートを整理していきましょう。まず、app/views/plan_items ディレクトリの new.html.erb と edit.html.erb の重複を解消します。new.html.erb の 2〜23 行を切り取ります。

app/views/plan_items/new.html.erb

```
1    <%= form_for @plan_item do |f| %>
2 -    <div class='form-group'>
3 -      <%= f.label :name, '件名' %>
4 -      <%= f.text_field :name, class: 'form-control', required: true %>
:
17 -   <div class='form-group'>
18 -     <%= f.label :ends_at, '終了日時' %>
```

```
19 -      <div class='input-group date datetime-picker'>
20 -        <%= f.text_field :ends_at, class: 'form-control', required: true %>
21 -        <span class='input-group-addon'><i class='fa fa-calendar'></i></span>
22 -      </div>
23 -    </div>
 2      <div class='form-group'>
 3        <%= f.submit '追加', class: 'btn btn-success' %>
 4      </div>
 5    <% end %>
```

同じディレクトリに新規ファイル_fields.html.erb を作成し、切り取ったコードを貼り付けます。貼り付けた後で、インデントを半角スペース 2 個分減らしてください。

app/views/plan_items/_fields.html.erb (New)

```
 1    <div class='form-group'>
 2      <%= f.label :name, '件名' %>
 3      <%= f.text_field :name, class: 'form-control', required: true %>
 :
16    <div class='form-group'>
17      <%= f.label :ends_at, '終了日時' %>
18      <div class='input-group date datetime-picker'>
19        <%= f.text_field :ends_at, class: 'form-control', required: true %>
20        <span class='input-group-addon'><i class='fa fa-calendar'></i></span>
21      </div>
22    </div>
```

new.html.erb を次のように書き換えます。

app/views/plan_items/new.html.erb

```
 1    <%= form_for @plan_item do |f| %>
 2 +    <%= render 'fields', f: f %>
 3      <div class='form-group'>
 4        <%= f.submit '追加', class: 'btn btn-success' %>
 5      </div>
 6    <% end %>
```

render メソッドの第 2 引数に { f: f } というハッシュを指定しました。こ

れで、部分テンプレートの中でローカル変数 f を通じてフォームビルダー f にアクセスできます。

new.html.erb と同様に、edit.html.erb の 2〜23 行を切り取ります。

app/views/plan_items/edit.html.erb

```
 1   <%= form_for @plan_item do |f| %>
 2 -   <div class='form-group'>
 3 -     <%= f.label :name, '件名' %>
 4 -     <%= f.text_field :name, class: 'form-control', required: true %>
 :
17 -   <div class='form-group'>
18 -     <%= f.label :ends_at, '終了日時' %>
19 -     <div class='input-group date datetime-picker'>
20 -       <%= f.text_field :ends_at, class: 'form-control', required: true %>
21 -       <span class='input-group-addon'><i class='fa fa-calendar'></i></span>
22 -     </div>
23 -   </div>
 2     <div class='form-group'>
 3       <%= f.submit '更新', class: 'btn btn-success' %>
 4     </div>
 5   <% end %>
```

そして、edit.html.erb を次のように書き換えます。

app/views/plan_items/edit.html.erb

```
 1   <%= form_for @plan_item do |f| %>
 2 +   <%= render 'fields', f: f %>
 3     <div class='form-group'>
 4       <%= f.submit '更新', class: 'btn btn-success' %>
 5     </div>
 6   <% end %>
```

ブラウザで予定追加フォームと予定変更フォームを開いて、表示内容が変化していないことを確認してください。

8.6　予定リスト用テンプレートの整理

次に、app/views/plan_items ディレクトリの index.html.erb のコードを書き換えます。コードに重複があるわけではありませんが、次の節で利用したいパーツを先回りして部分テンプレート化しておきます。

index.html.erb の 8〜10 行を切り取ります。

app/views/plan_items/index.html.erb

```
:
7       <div class='col-xs-4 hidden-md-up text-xs-right'>
8 -       <%= link_to [ :edit, item ] do %>
9 -         <i class='fa fa-pencil-square fa-lg'></i>
10 -      <% end %>
8       </div>
:
```

app/views/plan_items ディレクトリに新規ファイル_xs_toolbar.html.erb を作成し、切り取った内容を貼り付けます（インデントを半角スペース 6 個分減らします)。

app/views/plan_items/_xs_toolbar.html.erb (New)

```
1   <%= link_to [ :edit, item ] do %>
2     <i class='fa fa-pencil-square fa-lg'></i>
3   <% end %>
```

そして、index.html.erb を次のように書き換えます。

app/views/plan_items/index.html.erb

```
:
7       <div class='col-xs-4 hidden-md-up text-xs-right'>
8 +       <%= render 'xs_toolbar', item: item %>
9       </div>
:
```

index.html.erb の 18〜20 行を切り取ります。

app/views/plan_items/index.html.erb

```
  :
17      <div class='col-md-3 hidden-sm-down text-xs-right'>
18 -      <%= link_to [ :edit, item ], class: 'btn btn-secondary btn-sm' do %>
19 -        <i class='fa fa-pencil-square'></i> 変更
20 -      <% end %>
18      </div>
  :
```

app/views/plan_items ディレクトリに新規ファイル_md_toolbar.html.erb を作成し、切り取った内容を貼り付けます（インデントを半角スペース 6 個分減らします)。

app/views/plan_items/_md_toolbar.html.erb (New)

```
1   <%= link_to [ :edit, item ], class: 'btn btn-secondary btn-sm' do %>
2     <i class='fa fa-pencil-square'></i> 変更
3   <% end %>
```

そして、index.html.erb を次のように書き換えます。

app/views/plan_items/index.html.erb

```
  :
17      <div class='col-md-3 hidden-sm-down text-xs-right'>
18 +      <%= render 'md_toolbar', item: item %>
19      </div>
  :
```

ブラウザで予定リストページを開いて、表示内容が変化していないことを確認してください。

8.7 予定の詳細ページに「変更」リンクを設置

前節で作った部分テンプレートを利用して、予定の詳細ページに「変更」リンクを設置します。

app/views/plan_items/show.html.erb

```
1    <div class='container-fluid plan-item'>
2      <div class='row'>
3        <div class='col-xs-12 hidden-md-up text-xs-right'>
4          <%= link_to :plan_items do %>
5            <i class='fa fa-list fa-lg'></i>
6          <% end %>
7 +        <%= render 'xs_toolbar', item: @plan_item %>
8        </div>
9        <div class='col-md-12 hidden-sm-down text-xs-right'>
10         <%= link_to :plan_items, class: 'btn btn-secondary btn-sm' do %>
11           <i class='fa fa-list'></i> 予定表へ戻る
12         <% end %>
13 +       <%= render 'md_toolbar', item: @plan_item %>
14       </div>
15     </div>
```

ブラウザで予定の詳細ページを開くと、スマホモードでは図 8.3 のような表示になります。

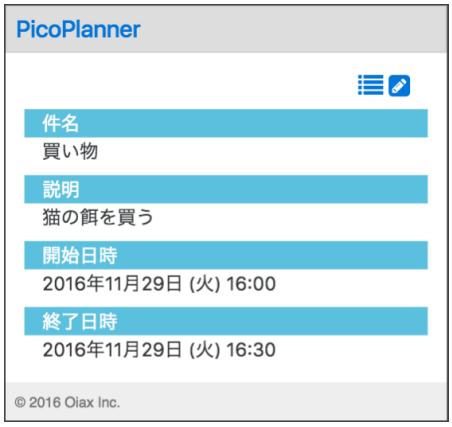

図 8.3　予定の詳細ページに「変更」リンクを設置（スマホモード）

通常モードでは図 8.4 のような表示になります。

図 8.4　予定の詳細ページに「変更」リンクを設置（通常モード）