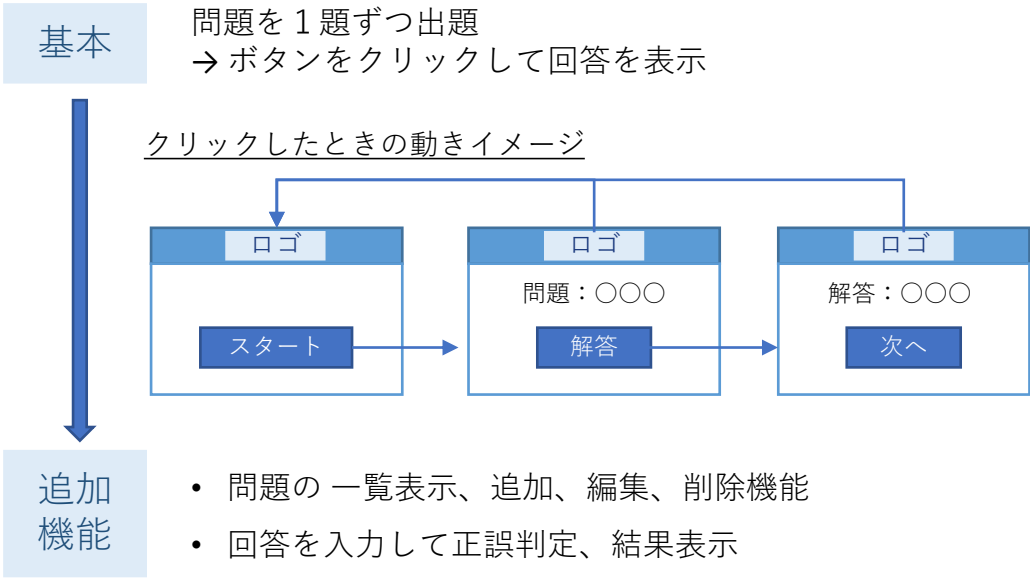


クイズアプリの作成手順

作成イメージ：<https://polar-forest-09833.herokuapp.com/>



～基本アプリの作成～

■ 作成するコントローラ、モデル

quizzesコントローラ

homeアクション・・・トップ画面表示
questionアクション・・・問題をランダムに 1 題出題
answerアクション・・・解答を表示

Quizモデル

問題と解答、解説（必要に応じて）を保存する表が必要

「表」のイメージ

問題を入れる 解答を入れる 解説を入れる（必要に応じて）

question	answer	description

縦列 1 つ 1 つを次から
「カラム」と言います

<条件>

question・・・文字列。入力必須。
answer・・・文字列。入力必須。
description・・・文字列。

※ コントローラ名、モデル名等は自由に決めても OK！

■ 作成手順

以下の流れで作成する

- ① Rails アプリの枠作成
- ②コントローラ作成
- ③トップ画面のルーティング設定
- ④ ページ遷移をさせる
- ⑤ モデル作成
- ⑥ データを保存してみる
- ⑦ コントローラで問題をランダムに 1 題選択する
- ⑧ 問題を表示する
- ⑨ 解答ページへのリンクを作成する
- ⑩ 解答を表示する
- ⑪ トップ画面へのリンクを作成する

■ 基本アプリの作成ステップ

① Rails アプリの枠作成

rails new ○○

アプリ名を英語で入力しよう

例：rails new quizapp

※ コントローラ名やモデル名とは違うものにしよう



アプリのテンプレートが自動生成される。

下記リンク「Railsアプリの表示」以降（rails new techpitgramは除く）を実行（Yay! You're on Rails!が表示されるところまで）

<https://sites.google.com/clack.ne.jp/manual/techpitgram%E7%92%B0%E5%A2%83%E6%A7%8B%E7%AF%89?#h.rj23hlerw6q3>



Yay! You're on Rails!



② コントローラ作成

rails g controller ○○ △△ △△ △△ △△

○○部分にコントローラ名、
△△部分にアクション名を入れる

※ アクション名は列挙できる。なくてもOK。

例：rails g controller quizzes home question answer

quizzesコントローラ

homeアクション・・・トップ画面表示

questionアクション・・・問題をランダムに1題出題

answerアクション・・・解答を表示



コントローラファイルやビューファイルを自動生成してくれる。例の場合だと、

- ・ quizzes_controller.rb (app/controllers/quizzes/の下)
- ・ home.html.erb (app/view/quizzes/の下)
- ・ question.html.erb (app/views/ quizzes/の下)
- ・ answer.html.erb (app/views/ quizzes/の下)

を自動生成してくれる。

→ ファイルの中身も確認しよう！

(② コントローラ作成 の続き)

コントローラやビューファイルの他に、rails g controller でアクションを指定していると、ルーティングも自動設定してくれる。

config/routes.rb

を確認しよう。

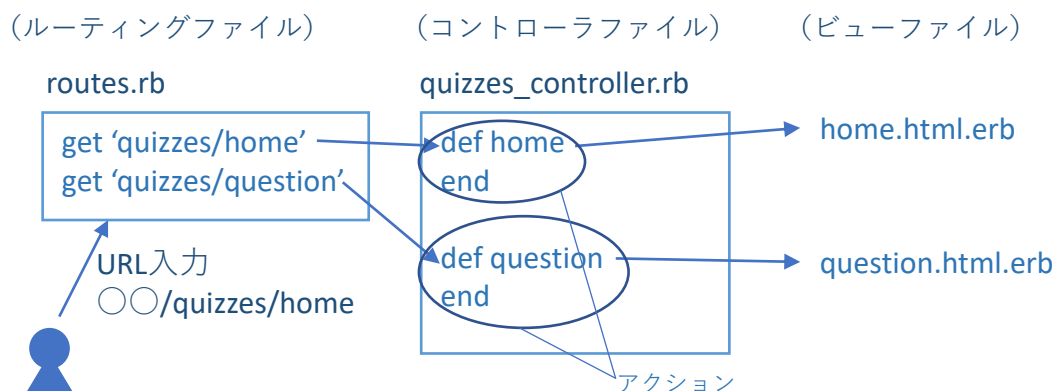
「Yay!You're on Rails」のページを表示させて、URLの一番最後 (?以降がある場合は、?以降を削除して) に

- /quizzes/home
- /quizzes/question
- /quizzes/answer

を追記して、それぞれの画面が表示されるか確認しよう。

例：https://〇〇.vfs.cloud9.ap-northeast-1.amazonaws.com/quizzes/home

自動生成内容のイメージ



確認出来たら、app/views/quizzes/home.html.erbをトップ画面用に編集してみよう。また、app/assets/stylesheets/quizzes.cssを編集してcss適用も試してみよう。

③ トップ画面のルーティング設定

現状は、

`https://○○.vfs.cloud9.ap-northeast-1.amazonaws.com/quizzes/home`
のようにしないとトップ画面が表示されないが、これを
`https://○○.vfs.cloud9.ap-northeast-1.amazonaws.com`
でトップ画面表示ができるよう変更する。

`config/routes.rb`
で
`get “コントローラ名/アクション名”`
が列挙されていると思うが、そのうちの1つ（トップ画面にしたいもの）について、
`root ‘コントローラ名#アクション名’`
に変更する。
設定したら、`https://○○.vfs.cloud9.ap-northeast-1.amazonaws.com`に
アクセスして確認。

④ ページ遷移をさせる

トップ画面から問題画面に移動するためのボタンを作成する。

`app/views/quizzes/home.html.erb`
(`app/views/コントローラ名/アクション名.html.erb`)
を編集する。

`link_to` を使っても、`button_to` を使ってもOK。
下記リンクなどを参考に（もちろん自分で調べてもOK!）やってみよう。

ボタンの作り方参考↓

https://pikawaka.com/rails/button_to

書いたらちゃんとページ遷移できるか試してみよう。

⑤ モデル作成

ここでは、データを保存できるようにする。

```
rails g model モデル名 カラム名:型 カラム名:型
```

カラム名: 型はいくつでも列挙できる

型は文字列とか数値とかある。文字列であればstring。

例：rails g model Quiz question:string answer:string description:string

問題を入れる 解答を入れる 解説を入れる（必要に応じて）

question	answer	description

<条件>

question・・・文字列。空欄はNG

answer・・・文字列。空欄はNG

description・・・文字列。

※ モデル名の部分を間違えたらややこしいけど、「カラム名:型名」は後からでも簡単に追加できるので間違えても気にしないで先に進んでください



app/models/quiz.rb

db/migrate/〇〇.rb

が自動生成される。

db/migrate/〇〇.rb が、データを格納するテーブルを作成するための定義ファイル。

中身を確認しよう。

(⑤ モデル作成 の続き)

マイグレーションファイルの中見例

```
class CreateQuizzes < ActiveRecord::Migration[6.0]
  def change
    create_table :quizzes do |t|
      t.string :question, null: false
      t.string :answer, null: false
      t.string :description
    end
    t.timestamps
  end
end
```

t.型名 :カラム名

で追加するカラムを設定できる。

例：t.string :answer でanswerカラムを文字列で作ることになる

t.integer :answer なら整数が解答になるようなanswerカラムを作れる

「null: false」で「入力必須（NULLはNG）」を指定できる。

マイグレーションファイルの中身を必要に応じて、
追記・修正し、

```
rails rb:migrate
```

データを保存するテーブル（表）の作成に成功したら、

```
===== ○○○ =====
create_table(○○)
-> ○s
===== ○○○ =====
```

のように表示される。

rails aborted! など、長々とした表示がされたら自分で調べる
か、講師に声をかける！

(⑤ モデル作成 の続き)

`null: false` のように、カラムに入力制限（必須入力、文字数制限など）を付ける場合は、モデルファイルも編集する。

quizモデルであれば、

`app/models/quiz.rb`

に、

`validates :カラム名, :presence: true`

を設定する（`presence`は「存在」という意味。`presence: true`で必須入力を意味する）

```
class Quiz < ApplicationRecord
  validates :question, presence: true
  validates :answer, presence: true
end
```

⑥ データを保存してみる

`rails c`

で、データの中身を見たり、保存したりの操作を（そういった画面を作ることなく）直接できるようになる。

`rails c` のあとは、

```
record = Quiz.new(question: "〇〇", answer: "〇〇")
record.save()
```

Quizモデルでなく独自のモデルを作った場合は、そのモデル名を入れる

カラム名: 入れたい内容
カラム名を変更した場合は、そのカラム名にする。必須入力カラム以外はなくてもOK.

モデル名.all

で、保存した内容が表示されたら成功。終わったら

`exit`

で、操作モードから抜け出せる。

⑦ コントローラで問題をランダムに 1 題選択する

app/controllers/コントローラ名.rb
を編集する。

該当のコントローラのアクションで、
「保存されているデータから 1 つランダムに問題を選択する」
のを実現する。

quizzesコントローラのquestionアクションなら、ココを編集

```
class QuizzesController < ApplicationController
  def home
  end

  def question
  end
end
```

ランダム選択の参考はこちら↓

<https://qiita.com/mailok1212/items/360d15eeab3bf9465f42>

先ほどのrails cで試してからコントローラに書き写すと確実。

@quiz = Quiz.○○

(変数名 = モデル名.○○)

のように、変数名の頭に「@」を付ける点に注意 (ビューファイル
に中身を渡すため)

quizzes#question

@quiz

question	answer	description

※ quizzes#question は「quizzesコントローラのquestionアクション」の意味。
書くとき長いので、以降でも時々こう書きます。

⑧ 問題を表示する

apps/views/コントローラ名/アクション名.html.erb を

<https://prog->

[8.com/slides?displayed_id=2925&lesson=70%2C74%2C76%2C77%2C82%2C89%2C86%2C96%2C92%2C93%2C107](https://prog-8.com/slides?displayed_id=2925&lesson=70%2C74%2C76%2C77%2C82%2C89%2C86%2C96%2C92%2C93%2C107)

を参考編集し、⑦で取得したデータの問題文部分 (上の例だと
@quizのquestionカラムの中身) を表示する。

@quizのquestionカラムの中身を取り出したい場合は

@quiz.question

になることに注意。

⑨ 解答ページへのリンクを作成する

問題画面のあとに解答画面に移動できるようなボタンを作る。

やり方は④と同じ。

例の場合だと、


quizzesコントローラの answerアクション
へ飛ぶように
app/views/quizzes/question.html.erb
を編集する。

⑩ 解答を表示する

問題への解答を表示するには、「解答を表示するアクションで直前に表示されていた問題に関する情報を取ってくる」必要がある。

やり方は複数考えられると思うが、今回はgetメソッドで、paramsを使ってビューからコントローラに、表示されていた問題文のidをコントローラに送ることを考える（getメソッドやparamsは分からなくてもとりあえず先に進んでOK）。

※ ややこしいので省略していたが、作ったテーブルには自動的にidのカラムもある。他にも、created_at, updated_at（データの作成時刻、更新時刻を記録するカラム）も自動的に作られる。



id	question	answer	description

paramsを使ったビューからコントローラへの値の受け渡しは↓
解答ページへのリンクを button_to で作成した場合は

https://pikawaka.com/rails/button_to#params

link_to で作成した場合は

<https://qiita.com/ATORA1992/items/566d76a7092bff40df4c>

が参考になる。

⑧も参考に、

app/controllers/コントローラ名.rb
を編集して、該当の問題文を取得し、
app/views/コントローラ名/アクション名.html.erb
を編集して解答を表示しよう。

⑪ トップ画面へのリンクの作成

どの画面からもトップ画面へ遷移するようなリンクを作りたい。
そういうときは、

app/views/layouts/application.html.erb
を編集する。

application.html.erbの初期状態

```
<!DOCTYPE html>
<html>
  <head>
    <title>Quizapp</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>

    <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track': 'page_load' %>
    <%= javascript_pack_tag 'application', 'data-turbolinks-track': 'page_load' %>
  </head>
  <body>
    <%= yield %>
  </body>
</html>
```

application.html.erb は（特に指定がない場合）常に表示され、
<%= yield %> に、今まで編集したような .html.erb ファイルの中身が、
そのまま入る。

なので、常に表示するようなヘッダーやフッターはここに書くと
良い。

ここに、トップ画面へ遷移するようなリンクの入ったヘッダーを作
成しよう。

ちなみに、せっかくなのでロゴを作りたい方は

<https://ja.cooltext.com/>
がなかなか使えます（多分調べれば他にもいろいろある）

これで基本アプリは終了！

追加機能はお好みで！