Ruby on rails

1 Cel laboratoriów

Wybór środowiska programistycznego lub Instalowanie Ruby on Rail framework i zapoznanie się z podstawami funkcjonowania i używania.

2 RoR

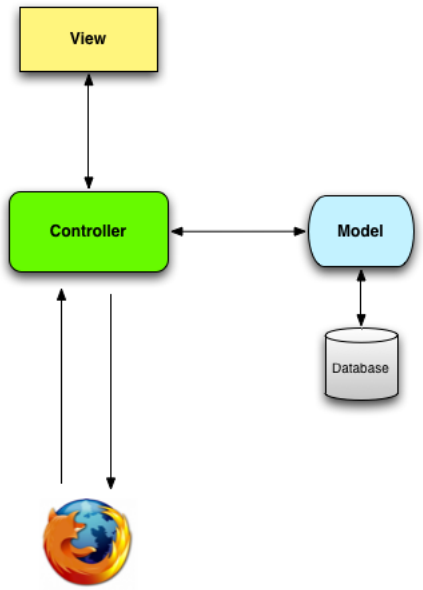
Ruby on Rails to framework do tworzenia dynamicznych stron internetowych napisany w języku programowania Ruby. Od debiutu w 2004 roku, Ruby on Rails szybko stał się jednym popularny. RoR jest wykorzystywany przez firmy takie jak Airbnb, Basecamp, Disney, GitHub, Hulu, Kickstarter, Shopify, Twitter i Yellow Pages.

Zalety Ruby on Rails

1. Jest w oprogramowaniem otwartym, dostępnym na licencji MIT, w efekcie jest darmowy.
2. Sukces zawdzięcza eleganckiej i kompaktowej konstrukcji.
3. poprzez wykorzystanie plastyczności języka Ruby, RoR jest skuteczny i wygodny. Kod aplikacji jest zwięzły i czytelny.
4. Możliwość wykorzystania do rapid prototyping.
5. Generowanie kodu dla programisty - scaffolding.
6. Mało SQLa — wykorzystanie mapowania w języku Ruby.
7. Jest modularny.
8. SQLite jako domyślną bazą danych (nie wymaga serwera, bez konfiguracji, transakcyjna, relacyjna baza danych). W konsekwencji nie trzeba konfigurować bazy danych, by zacząć kodować.

RoR pracuje wg wzorca nazwanego „Model-View-Controller”, czyli podziale całej aplikacji na osobne części:

1. Model — reprezentuje dane, z którymi pracuje aplikacja — baza danych.



Rysunek 1: Schemat prezentujący architekturę MVC (źródło: Ruby on Rails Tutorial 3rd edt.)

2. Widok (view) — reprezentacja danych, należy to utożsamić z tym co widzi użytkownik (strony WWW).

3. Kontroler (controller) — zapewnia interakcję pomiędzy modelem i widokiem, jest odpowiedzialny za sterowanie, interakcję stron z bazą danych.

Podczas interakcji z aplikacją Rails, przeglądarka wysyła żądanie, które jest odbierany przez serwer WWW i przekazywane do kontrolera Rails, który jest odpowiedzialny za to, co dalej robić. W niektórych przypadkach, kontroler natychmiast wygeneruje widok, który jest szablonem i który zostanie przekonwertowany do formatu HTML i wysłany jako odpowiedź do przeglądarki. Dla dynamicznych stron, czyli w innych przypadkach, kontroler współpracuje z modelem. Po wywołaniu modelu, kontroler renderuje widok i wysyła pełną stronę WWW w formacie HTML do przeglądarki.

3 Wybór środowiska

Możliwe jest tworzenie aplikacji lokalnie (instalacja Ruby) lub w chmurze np. Cloud9 (należy podać numer karty kredytowej nawet dla konta darmowego), Heroku lub Codeanywhere.

3.1 Instalacja

Najlepiej posłużyć się instrukcją ze strony `http://installrails.com`.

Instalacja z użyciem gems:

```
gem install rails -v 4.2.2
```

3.2 Cloud9

Oto kroki do rozpoczęcia pracy ze środowiskiem:

1. Zarejestruj się na Cloud9 pod adresm `http://c9.io` UWAGA: podczas rejestracji do konta studenckiego należy podać numer karty kredytowej.
2. Kliknij na „Go to your Dashboard"
3. Wybierz opcję „Create New Workspace"
4. Utworzyć obszar roboczy o nazwie „rails-tutorial", ustaw ją na „Private to the people I invite" i wybierz ikonę Rails Tutorial
5. Po zakończeniu tworzenia obszaru roboczego, zaznacz go i kliknij „Start editing"

3.3 Codeanywhere

1. Zarejestruj się na Codeanywhere pod adresm `http://codeanywhere.com` .
2. W obszarze zarządzania projektami wybierz „Create a new Project".
3. Nadaj mu nazwę projektRoR
4. Otwórz projekt
5. Wybrać środowisko „Ruby na Ubuntu" i nadać nazwę obszaru roboczego np:. „Obszar-Roboczy".
6. Po zakończeniu tworzenia obszaru roboczego należy używać terminala do wykonywania poleceń do tworzenia aplikacji RoR.
7. Zmienić ustawienia, by pozwolić na wyświetlenie wyników — renderowanie aplikacji poprzez:
   (a) Kliknij prawym przyciskiem myszy na box „ObszarRoboczy" i wybierz „Config".
   (b) W otwartym pliku konfiguracyjnym zmień:

```
"commands": [
"rails server"
],
```

   na

```
"commands": [
"rails server --binding=0.0.0.0"
],
```

   (c) W otwartym pliku konfiguracyjnym zmień:

```
"cwd": "~/workspace",
```

na

```
"cwd": "~/workspace/hello_app",
```

(d) W otwartym pliku konfiguracyjnym zmień:

```
"environment": {
"PORT": "80"
},
```

na

```
"environment": {
"PORT": "3000"
},
```

8. W przeglądarce odblokować wyskakujące okienka (w takim oknie będzie można oglądać wynik działania aplikacji).

4 Pierwszy program „Hello world"

4.1 Tworzenie aplikacji

Praktycznie wszystkie aplikacje Rails rozpoczyna się w ten sam sposób, poleceniem „rails new", które tworzy projekt w architekturze MVC.

Wykonać polecenie tworzące szkielet aplikacji RoR.

```
$ rails new hello_app
```

Układ katalogów i plików jest ustandaryzowany, co ułatwia pracę. Schemat pokazany jest w Tablicy 1.

Po utworzeniu nowej struktury aplikacji następuje automatyczne instalowani bibliotek (gems) przy użyciu narzędzia Bundler. Bundler może być uruchamiany ręcznie. Taka konieczność występuję, gdy chcemy dodać inne niż standardowe biblioteki. Aby dodać bibliotekę:

Otwórz w edytorze

```
Gemfile
```

przyjrzyj się zawartości.

Aby zainstalować zmiany:

```
$ cd hello_app/
$ bundle install
```

Tablica 1: Zawartość aplikacji RoR

File/Directory	Przeznaczenie
app/	Główny kod aplikacji włączając model, views i controllers oraz pomoc
app/assets	zasoby aplikacji, takie jak kaskadowe arkusze stylów (CSS), pliki JavaScript i obrazy
bin/	pliki wykonywalne
config/	konfiguracja aplikacji, np. jaka db
db/	pliki bazy danych
doc/	dokumentacja do aplikacji
lib/	moduły bibliotek
lib/assets	zasoby biblioteczne takie jak kaskadowe arkusze stylów (CSS), pliki JavaScript i obrazy
log/	pliki z logami
public/	Dane udostępniane publicznie (na przykład za pośrednictwem przeglądarek internetowych), takie jak strony błędów (statyczne elementy strony)
bin/rails	Program do generowania kodu, otwierając sesję konsoli, lub uruchomieniem serwera lokalnego
test/	testy aplikacji
tmp/	pliki tymczasowe
vendor/	Kody innych firm, takich jak wtyczki i gems
vendor/assets	Zasoby niezbędna dla vendor
README.rdoc	krótki opis aplikacji
Rakefile	Zadania użytkowe dostępne poprzez polecenie „rake”
Gemfile	wymagane biblioteki gems dla aplikacji
Gemfile.lock	Lista bibliotek gems wykorzystywanych w celu zapewnienia, że wszystkie kopie aplikacji wykorzystują te same wersje gem
config.ru	Plik konfiguracyjny dla RACK a Ruby Webserver Interface
.gitignore	wzorce do pominięcia przez Git

4.2 Uruchamianie aplikacji

Aby uruchomić aplikację wystarczy użyć wbudowanego serwera www. Urchamia go polecenie „rails server" lub „rails s". np.:

```
$ cd ~/hello_app/
$ rails server
=> Booting WEBrick
=> Rails application starting on http://localhost:3000
=> Run ‘rails server -h‘ for more startup options
=> Ctrl-C to shutdown server
```

Od tej pory ten terminal jest zajęty poprzez działający serwer. Do dalszej pracy potrzebny będzie kolejny terminal.

Na serwerze lokalnym, należy wpisać w przeglądarce adres `http://localhost:3000`.

Aby zobaczyć informacje na temat pierwszej aplikacji, kliknij na link „About your application's environment".

4.3 Hello, world!

W pierwszym zastosowaniu MVC, zrobimy maleńkie zmiany, dodając akcję kontrolera do renderowania napisu "Hello, World!".

Jak wskazuje ich nazwa, działania kontrolera są zdefiniowane wewnątrz controllers.

```
$ ls app/controllers/*_controller.rb
```

Otwórz w edytorze plik „application_controller.rb" i dopisz do niego:

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  def hello
    render text: "hello, world!"
  end
end
```

Po zdefiniowaniu działania, które zwraca renderowany tekst, musimy wskazać Rails bieżące miejsce jako aktulane, zamiast domyślnej strony. Aby to zrobić, będziemy edytować „routes", który działa przed kontrolerem i decyduje, gdzie wysłać zapytania, które przychodzą z przeglądarki. Zatem: chcemy zmienić domyślną stronę, root route, która określa stronę wyświetlaną z głównego adresu strony.

Należy otworzyć plik „config/routes.rb" zawierający zakomentowane linijki. W linijce

```
# root 'welcome#index'
```

"welcome"to nazwa kontrolera a "index"jest działaniem/metodą w ramach tego kontrolera. Aby aktywować ścieżkę root, usuń znak komentarza, a następnie zastąp go kodem:

```
# You can have the root of your site routed with "root"
root 'application#hello'
```

Odśwież przeglądarkę wyświetlającą stronę główną projektu hello_app.

4.4 Hello, world! — statycznie

Utwórz w katalogu „public" plik „index.html". Ten folder jest zawsze przeszukiwany jako pierwszy przed przejściem do obsługi stron dynamicznych. Do pliku wpisz:

```
<!DOCTYPE html>
<html>
<head><title>Totally Static</title></head>
<body><h2>Will never generate dynamic content :(</h2></body>
</html>
```

5 Wdrożenie

Nawet na tak wczesnym etapie, mamy już zamiar wdrożyć prawie pustą aplikację Rails.

W chwili obecnej jest wiele możliwości, hostów lub serwerów wirtualnych. Można wykorzystać Heroku, który jest platformą hostingową zbudowaną specjalnie do wdrażania Rails i innych aplikacji internetowych. Heroku ułatwia wdrażanie aplikacji Rails jeżeli Twój kod źródłowy jest pod kontrolą wersji z użyciem Git. Darmowy Heroku jest wystarczający na potrzeby testowania aplikacji.

Należy utworzyć i skonfigurować nowe konto Heroku. Pierwszym krokiem jest zarejestrowanie się na Heroku. Następnie sprawdzenie, czy system ma już zainstalowanego klienta wiersza polecenia Heroku:

```
$ heroku version
```

Jeżeli nie, to ze strony swojego konta przejdź do instrukcji konfiguracji i zostanie Ci przedstawiona ścieżka pobrania Heroku Toolbelt.

Następnie należy się zalogować hosta i podać swój klucz SSH:

```
$ heroku login
$ heroku keys:add
```

Logowanie jest niezbędne, by móc skojarzyć pracę Git i Heroku. Następnie wykonaj polecenie:

```
$ heroku create
```

by utworzyć na serwerze miejsce, w którym będzie przechowywana aplikacja. Komenda Heroku tworzy nową subdomenę tylko dla naszej aplikacji, dostępną do natychmiastowego oglądania. Nie ma tam jeszcze nic.

Aby zainstalować aplikację, pierwszym krokiem jest użycie Git „push" do głównej gałęzi Heroku:

```
$ git push heroku master
```

Aby obejrzeć wynik: aplikację wystawioną na zewnątrz uruchom:

```
$ heroku open
```

6 Zadania

1. Zmień treść strony, aby wyświetlić "Witaj świecie!" zamiast "Hello World!".
2. Dodać drugą akcję/metodę o nazwie goodbye, która renderuje tekst "Do zobaczenia!". Edytuj plik root route, tak że strona główna trafia do metody goodbye zamiast do hello.

7 Więcej o kontrolerach

Kontroler zawiera akcje (metody w Ruby) i przetwarza zapytania z przeglądarki. Kontroler może być wygenerowany w sposób szybki wraz z odpowiadającym mu widokiem „rails generate controller controller_name [action1 action2]". Wykonaj:

```
$ rails generate controller greeter hello
```

Sprawdź zawartość katalogu controllers i zawartość nowego pliku oraz zawartość katalogu views. Wyświetl zawartość strony: `http://localhost:3000/greeter/hello`. Pliki w views wyglądają jak HTML, ale mają rozszerzenie erb. Jest to osadzony html w Ruby. Aby w tym pliku wykonać kod Ruby należy zapisać <% ruby code %> lub <%= ruby code %>.

Dodaj do view hello.html.erb zapis:

```
<% random_names = ["Alex", "Joe", "Michael"] %>
<h1>Hej, <%= random_names.sample %></h1>
<p>Teraz jest: <%=Time.now%> </p>
```

8 Routes

Zapytania przeglądarki muszą zostać przekierowane (routed) do kontrolera. A wykonuje to element Route.

Ścieżka do metody/działania "hello" została wygenerowana automatycznie (config/routes.rb) przy generowaniu kontrolera.

Zadanie poniższe pokazuje jak samodzielnie dodać akcję/motedę wewnątrz kontrolera i obsłużyć ją, tak by została wyświetlona w przeglądarce.

1. Dodaj w pliku greeter_controller.rb nową metodę „goodbye”.
2. W podkatalogu /viewes/greeter/ dodaj nowy plik goodbye.html.erb. Dodaj kod wyświetlający pożegnanie dla losowo wygenerowanych imion z tablicy imion.
3. Wyedytuj plik config/routes.rb i dodaj:

   ```
   get 'greeter/goodbye'
   ```

4. Aby przetestować ścieżki w danej web aplikacji wpisz w terminalu, gdy bieżącym katalogiem jest katalog aplikacji np. „hello_app”:

   ```
   $ rake routes
   ```

9 Kilka ciekawostek

Jeżeli metoda w kontrolerze jest pusta i ma niepowtarzalną nazwę to można taką metodę usunąć z kontrolera np. zmień zawartość kontrolera greeter_controller.rb

```
class GreeterController < ApplicationController
  def hello
  end
  #def goodbye
  #end
end
```

Po tej operacji nadal strony działają jak poprzednio.

Zasadą w Rails jest przeniesienie jak największej możliwej porcji kodu dynamicznego do kontrolera, by odciążyć views.

Wpisz do pliku greeter_controller.rb:

```
def hello
   random_names = ["Ala", "Ola", "Iza"]
   @name = random_names.sample
   @time = Time.now
   @times_displayed ||= 0
   @times_displayed += 1
 end
```

Wszystkie zmienne instancji będą widoczne w obrębie view dla metody hello. Wpisz w pliku hello.html.erb:

```
<h1>Hej, <%= @name %></h1>
<p>Jest teraz godzina <%= @time %> </p>
<p>Ta strona była odwiedzona <%= @times_displayed %> razy</p>
```

Sprawdź działanie strony. Zmienna times_displayed jest inicjowana za każdym razem, gdy przeglądarka wyśle zapytanie. Takie dane mogą być przechowywane w środowisku sesji lub bazie danych.

10 Helpers

Helpers to rodzaj makro, narzędzie do formatowania Views czyli interfejsu.

Zmienna time wyświetla czas w niesformatowanej formie. Wartość czas jest dostępna przez zmienną @time. Zadanie takie realizuje się w elemencie helper. Wszystko co jest zdefiniowane w helper jest widoczne dla wszystkich views nie tylko greeter views.

Otwórz plik greeter_helper.rb i dopisz:

```
def formatted_time(time)
  time.strftime("%I:%M%p") # time string in AM/PM format
end
```

Następnie w odpowiednim pliku views wywołaj funkcję formatted_time(@time) zamiast @time. Sprawdź zmiany.

Do dyspozycji jest bardzo użyteczny wbudowany helper „link_to". Użycie:

```
link_to name path
```

Jest to generator linku (znacznik a w html), który wyświetla na stronie „name" i przekierowuje do „path". Path może być wyrażeniem lub ścieżką z pliku routes.rb kończący się „_url" (pałna ścieżka) lub „_path"(ścieżka względna). Można zamiast ścieżki wykorzystać zmienną, której zmiana wartości zmieni przekierowanie bez zbytecznych dużych zmian w kodzie!

Dodaj w pliku hello.html.erb kod:

```
<p><%= link_to "Google", "http://www.google.com" %></p>
<p><%= link_to "Goodbye", greeter_goodbye_path %></p>
```

„greeter_goodbye", to prefix. Można go odczytać listy rake routes.

11 Literatura obowiązkowa

1. https://www.railstutorial.org
2. http://mislav.uniqpath.com/poignant-guide/book/