

# EPI: Interfejs Graficzny Podstawy Ruby on Rails

Aleksander Pohl

6 października 2010

# Ruby i Ruby on Rails

## Ruby

Japonia 1995, Yukihiro 'Matz' Matsumoto

<http://www.ruby-lang.org/en/>

**Ruby on Rails** – framework napisany w Ruby dla aplikacji opartych o bazę danych

- ▶ 2004 – początek projektu, David Heinemeier Hansson
- ▶ 2010 – Rails v. 3.0 (powstało z połączenia z projektem Merb)

<http://www.rubyonrails.org>

# Ruby on Rails – podstawowe założenia

- ▶ DRY – nie powtarzaj się
- ▶ Konwencja ponad konfiguracją: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ Wzorzec MVC – model – widok – kontroler
- ▶ Model – ActiveRecord: mapowanie obiektowo-relacyjne, migracje, zunifikowana obsługa wielu RDMS-ów, etc.
- ▶ Widok – ActionView: Rhtml, Haml, wsparcie dla Ajaxa
- ▶ Kontroler – ActionController: obsługa żądań, mapowanie adresów URL, cache'owanie, etc.
- ▶ Generatory – szablony dla modeli, rusztowań, migracji bazy danych, itp.

# Ruby on Rails – zasoby

## Dokumentacja:

- ▶ <http://apohllo.pl/dydaktyka/interfejsy/rails>
- ▶ <http://guides.rubyonrails.org/> – Rails Guides
- ▶ <http://apohllo.pl/guides/index.html> – Rails Guides (po polsku, omawia Rails 2.3.5)
- ▶ <http://railsapi.com> – działa szybko i jest aktualna

## Edytory:

- ▶ Vim + Rails plugin (w pracowniach)
- ▶ Aptana RadRails (na bazie Eclipse)
- ▶ NetBeans + Ruby plugin
- ▶ RubyMine (płatna)

# Szybkie wprowadzenie – system do obsługi biblioteki

Bardzo prosta biblioteka. Funkcjonalności:

1. przeglądanie listy książek
2. dodawanie książki
3. przeglądanie szczegółów książki
4. edytowanie szczegółów książki
5. usuwanie książki

# Plan działania


1. wygenerowanie struktury aplikacji  
`rails new library`
2. uruchomienie serwera  
`rails server`
3. konfiguracja bazy danych (opcjonalne)
4. dodanie modelu 'author'  
`rails generate scaffold author ..`
5. dodanie modelu 'book'  
`rails generate scaffold book ..`
6. dodanie wyboru autora przy edycji książki
7. powiązanie modeli 'author' i 'book'  
`belongs_to :author`
8. uwzględnienie autora przy wyświetlaniu książki
9. usunięcie powtórzeń kodu



# Tworzenie nowego projektu

1. przejdź do katalogu, w którym ma znaleźć się główny katalog projektu
2. `rails new library`
3. `cd library`
4. `rails server`
  - ▶ domyślny numer portu to 3000; jeśli wiele aplikacji działa w osobnych serwerach, to możemy zmienić go za pomocą opcji `-p`
  - ▶ to polecenie wywołuje wbudowany serwer zwany WEBRICK
  - ▶ w Windows: `ruby rails server`
  - ▶ aby uruchomić w tle: `rails server -d`
5. otwórz przeglądarkę i wprowadź adres `http://localhost:3000`

# Rails – okno powitalne



## Welcome aboard

You're riding Ruby on Rails!

[About your application's environment](#)

---

### Getting started

Here's how to get rolling:

- 1. Use `script/generate` to create your models and controllers**

To see all available options, run it without parameters.
- 2. Set up a default route and remove or rename this file**

Routes are set up in `config/routes.rb`.
- 3. Create your database**

Run `rake db:migrate` to create your database. If you're not using SQLite (the default), edit `config/database.yml` with your username and password.



## Rails – struktura katalogów

- ▶ app: kod źródłowy
- ▶ config: konfiguracja
- ▶ db: schemat bazy danych
- ▶ doc: dokumentacja
- ▶ lib: dodatkowe biblioteki
- ▶ log: logi
- ▶ public: obrazki, css, js – treści statyczne
- ▶ script: specjalne skrypty aplikacji
- ▶ test: automatyczne testy
- ▶ tmp: pliki tymczasowe
- ▶ vendor: dodatki (pluginy)

# Katalog 'app'

Railsy wykorzystują wzorzec projektowy Model-View-Controller i jest on odzwierciedlony w strukturze katalogów:

- ▶ **models** – klasy zawierające logikę biznesową
- ▶ **controllers** – kontrolery spajające widoki z danymi
- ▶ **views** – widoki zorganizowane w katalogach odpowiadających kontrolerom
- ▶ **helpers** – metody pomocnicze wykorzystywane w widokach
- ▶ **mailers** – dodatkowe moduły pomocne w wysyłaniu i odbieraniu poczty elektronicznej

## Konfiguracja bazy danych (opcjonalne)

Określamy parametry połączenia z bazą danych w pliku `config/database.yml`. W tym momencie wprowadzamy zmiany tylko w sekcji zatytułowanej `'development'`:

```
development:  
  adapter: sqlite3  
  database: db/development.sqlite3  
  pool: 5  
  timeout: 5000  
  encoding: utf8
```

Następnie wywołujemy polecenie: `$ rake db:create`  
Aby usunąć istniejącą bazę danych: `$ rake db:drop`

# Tworzenie rusztowania dla modelu 'author'

Przechodzimy do katalogu libarary i generujemy rusztowanie:

```
$ rails generate scaffold author first_name:string  
last_name:string
```

- ▶ model 'author' posiada atrybuty imię i nazwisko, które są łańcuchami znaków
- ▶ zostanie wygenerowana migracja, która dodaje do bazy danych schemat odpowiadający modelowi 'author'
- ▶ zostanie również wygenerowany podstawowy kontroler wraz z widokami do manipulowania autorami

# Podgląd pliku migracji

Otwieramy plik `xxx_create_authors.rb` w katalogu `db/migrate`.

```
class CreateAuthors < ActiveRecord::Migration
  def self.up
    create_table :authors do |t|
      t.string :first_name
      t.string :last_name
      t.timestamps
    end
  end

  def self.down
    drop_table :authors
  end
end
```

# Podgląd pliku rusztowania

Otwieramy plik `authors_controller.rb` w katalogu `app/controllers`.

```
class AuthorsController < ApplicationController
  # GET /authors
  # GET /authors.xml
  def index
    @authors = Author.all

    respond_to do |format|
      format.html # index.html.erb
      format.xml { render :xml => @authors }
    end
  end

  # GET /authors/1
  # GET /authors/1.xml
  def show
    @author = Author.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.xml { render :xml => @author }
    end
  end
  # ...
end
```



# Uruchamiamy migrację

```
$ rake db:migrate
== CreateAuthors: migrating =====
-- create_table(:authors)
   -> 0.0041s
== CreateAuthors: migrated (0.0042s) =====
```

- ▶ rake jest narzędziem podobnym do GNU Make, który pozwala na uruchamianie zadań, takich jak np. wywołanie migracji bazy danych
- ▶ aby polecenie wykonało się poprawnie trzeba być w katalogu `library!`
- ▶ polecenie powoduje zmianę schematu bazy danych zgodnie z wygenerowaną wcześniej migracją

# Wygenerowany schemat bazy

Jeśli nie było problemów z biblioteką bazy danych sqlite3 możemy zobaczyć wygenerowane tabele:

```
$ sqlite3 db/development.sqlite3
SQLite version 3.6.2
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .schema

SHOW CREATE TABLE authors;
CREATE TABLE 'authors' ('id' INTEGER PRIMARY KEY
  AUTOINCREMENT NOT NULL, 'first_name' varchar(255),
  'last_name' varchar(255));
SHOW CREATE TABLE schema_migrations;
CREATE TABLE 'schema_migrations' ('version'
  varchar(255) NOT NULL);
```

## Rusztowanie dla modelu 'author'

Uruchamiamy serwer (`rails server`) i otwieramy przeglądarkę na adresie **`http://localhost:3000/authors`**

Generator rusztowania utworzył m.in. następujące pliki:

- ▶ **`app/controllers/authors_controller.rb`** – definiuje akcje `index`, `show`, `new`, `create`, `edit`, `update` oraz `destroy`
- ▶ **`app/models/author.rb`** – dynamicznie (przez mechanizm ORM) mapuje książkę do tabeli w bazie danych (**Uwaga:** nazwa modelu jest w liczbie pojedynczej, a tabeli – w liczbie mnogiej.)
- ▶ **`app/views/authors/*`** – widoki w plikach `html.erb` (html z osadzonym kodem Rubiego)

## Tworzenie rusztowania dla modelu 'book'

```
$ rails generate scaffold book title:string  
  author:references
```

Następnie (w katalogu głównym!):

```
$ rake db:migrate
```

Oglądamy wynik pod adresem (pamiętając o serwerze):

**<http://localhost:3000/books>**

Próba utworzenia nowej książki zakończy się niestety porażką!

# Zmiana sposobu tworzenia książek

Dodajemy poniższy kod na początku pliku `app/controllers/books_controller.rb`:

```
class BooksController < ApplicationController
  before_filter :find_authors, :only => [:new, :edit, :update, :create]

  # GET /books
  # ...
```

Oraz na jego końcu (przed słowem kluczowym `end`):

```
protected
def find_authors
  @authors = Author.find(:all).map do |author|
    [ author.first_name + ' ' + author.last_name, author.id]
  end
end
```

## Zmiana sposobu tworzenia książek – cd.(2)

Modyfikujemy plik `app/views/books/_form.html.erb`

```
<div class="field">
  <%= f.label :author %><br />
  <%= f.text_field :author %>
</div>
```

przez:

```
<div class="field">
  <%= f.label :author_id %><br />
  <%= f.select :author_id, @authors %>
</div>
```

Teraz zaglądamy pod `http://localhost:3000/books/new`

# Definiowanie nowej relacji w modelu 'book'

Zaglądamy do `app/models/book.rb`:

```
class Book < ActiveRecord::Base
  belongs_to :author
end
```

Dzięki zastosowaniu `author:references` przy generowaniu szkieletu, zdefiniowana jest relacja wiele-do-jednego pomiędzy książką a autorem.

Informację o tej relacji musimy jednak również dodać do modelu `author`:

```
class Author < ActiveRecord::Base
  has_many :books
end
```

# Poprawienie sposobu wyświetlania szczegółów książki

Zmieniamy widok książki `app/views/books/show.html.erb`:

```
<p>
  <b>Title:</b>
  <%=h @book.title %>
</p>

<p>
  <b>Author:</b>
  <%=h @book.author.first_name + ' ' +
    @book.author.last_name %>
</p>

<%= link_to 'Edit', edit_book_path(@book) %> |
<%= link_to 'Back', books_path %>
```

Teraz oglądamy szczegóły książki!

# Usuwanie powtórzeń kodu (DRY!)

Aby wyświetlić imię i nazwisko autora, w plikach **books\_controller.rb** oraz **show.html.erb** użyliśmy podobnego kodu. Było to po prostu połączenie imienia i nazwiska w jeden łańcuch.

Aby usunąć to powtórzenie zdefiniujemy nową metodę w modelu 'author', która będzie działać jak wirtualny atrybut.

Dodajmy zatem metodę **full\_name** w pliku **app/models/author.rb**:

```
class Author < ActiveRecord::Base
  has_many :books

  def full_name
    "#{self.first_name} #{self.last_name}"
  end
end
```



# Usuwanie powtórzeń

Następnie w **show.html.erb** zastępujemy:

```
<%=h @book.author.first_name + ' ' + @book.author.last_name %>
```

przez:

```
<%=h @book.author.full_name %>
```

A w **books\_controller.rb** zastępujemy:

```
@authors = Author.find(:all).map do |author|  
  [ author.first_name + ' ' + author.last_name, author.id]  
end
```

przez:

```
@authors = Author.find(:all).map do |author|  
  [ author.full_name, author.id]  
end
```

Teraz sprawdzamy czy wszystko działa jak należy!

# Zadanie

Zmodyfikuj listing książek tak aby zawierał imię i nazwisko autora!

# Podziękowania

Dla:

- ▶ Agnieszki Figiel, za udostępnienie prezentacji w postaci plików źródłowych
- ▶ Marka Kowalcze oraz Jakuba Kuźmy z grupy SRUG (srug.pl), za pomoc przy kolorowaniu składni w Latex'u