

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

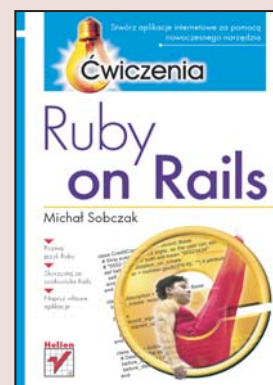
FRAGMENTY KSIĄŻEK ONLINE

Ruby on Rails. Ćwiczenia

Autor: Michał Sobczak

ISBN: 83-246-0661-0

Format: A5, stron: 192



Stwórz aplikacje internetowe za pomocą nowoczesnego narzędzia

- Poznaj język Ruby
- Skorzystaj ze środowiska Rails
- Napisz własne aplikacje

W dobie rosnącej popularności Linuksa, niestabnącej popularności systemu Windows i obecności na rynku innych systemów operacyjnych aplikacje „biurkowe”, wymagające konkretnego środowiska uruchomieniowego, tracą swoją pozycję. Ich miejsce zajmują aplikacje intranetowe bądź internetowe – instalowane na serwerach sieciowych, wymagające po stronie użytkownika jedynie przeglądarki WWW. Rozwiązanie takie jest niezwykle wygodne również dla twórców aplikacji, ponieważ zdecydowanie upraszcza proces wprowadzania nowych wersji systemu oraz jego konserwacji. Istnieje wiele technologii ułatwiających tworzenie takich aplikacji. Jedną z nowości na rynku jest zyskująca coraz większe uznanie Ruby on Rails.

Dzięki książce „Ruby on Rails. Ćwiczenia” opanujesz podstawy tworzenia aplikacji internetowych za pomocą tej technologii. Nauczysz się programować w języku Ruby: poznasz jego elementy i zasady projektowania obiektowego, metody przetwarzania danych tekstowych, pracy z plikami i katalogami oraz obsługi błędów i wyjątków. Przeczytasz także o środowisku Rails, instalowanym na serwerze aplikacji. Wykonując ćwiczenia z ostatnich rozdziałów, zrealizujesz projekt aplikacji służącej do zarządzania czasem w technice Ruby on Rails.

- Instalacja interpretera Ruby
- Podstawowe elementy języka Ruby
- Konstrukcje warunkowe i sterujące
- Programowanie obiektowe
- Obsługa wyjątków
- Korzystanie z wyrażeń regularnych
- Instalacja środowiska Rails
- Generowanie adresów URL
- Szablony RHTML
- Wysyłanie poczty elektronicznej

**Poznaj już dziś technologię, dzięki której tworzenie aplikacji
będzie bardzo wydajne i przyjemne**



Spis treści

	Wstęp	7
Część I	Ruby	9
Rozdział 1.	Wprowadzenie do Ruby	11
Rozdział 2.	Instalacja interpretera	15
	Windows	15
	Linux	16
	Podsumowanie	18
Rozdział 3.	System pomocy — RI	19
	Podsumowanie	20
Rozdział 4.	Podstawy składni	21
	Standardy nazewnictwa	21
	Tablice	25
	Zakresy	28
	Iteratory	29
	Bloki kodu	30
	BEGIN oraz END	31
	Dołączanie kodu źródłowego	32
	Zmienne predefiniowane	33
	Słowa kluczowe	34
	Istotne uwagi	34
	Podsumowanie	35

Rozdział 5. Struktury kontrolne	37
Warunek if	38
Warunek unless	40
Warunek case — wielokrotny wybór	40
Pętla for	41
Pętla loop	43
Pętla while	43
Pętla until	44
break, redo, next i retry	45
each	45
Podsumowanie	46
Rozdział 6. Programowanie zorientowane obiektowo	47
Klasy i obiekty	47
Metody	48
Metody klas	50
Moduły	52
Metody modułów	53
Rozszerzanie obiektów	53
Dziedziczenie	54
Singletony	55
Kontrola dostępu	56
Podsumowanie	58
Rozdział 7. Wyjątki	59
rescue	59
raise	60
ensure	60
catch i throw	61
Podsumowanie	62
Rozdział 8. Wyrażenia regularne	63
Pierwszy kontakt	64
Dopasowanie	64
Klasy znaków	67
Zamiana	68
Podejście obiektowe	69
Podsumowanie	70
Rozdział 9. Operacje wejścia i wyjścia	71
Operacje podstawowe	71
Pliki	73
Podsumowanie	74

Rozdział 10. Usuwanie błędów	75
Podsumowanie	76
Rozdział 11. Biblioteka klas	77
Array	77
Object	80
File	81
Podsumowanie	82
Część II Rails	9
Rozdział 12. Wprowadzenie do Rails	85
Rozdział 13. Rails — instalacja	87
Rails	87
XAMPP	88
Podsumowanie	92
Rozdział 14. Pierwsza aplikacja	93
Przypomnienie	93
Drzewo projektu	94
WEBrick	96
Witaj świecie	97
Znaczniki	99
Podsumowanie	102
Rozdział 15. Active Record	103
Przygotowanie środowiska	103
Przepisy kulinarne	106
Zmiana wyglądu	108
Kategorie	111
Podsumowanie	114
Rozdział 16. Action Controller	115
Routing	115
Generowanie URL	118
Metody akcji	120
Środowisko kontrolera	121
Generowanie szablonów	122
Wysyłanie danych	122
Ciasteczka	123
Flash — komunikacja pomiędzy metodami	124
Filtry	125
Podsumowanie	126

Rozdział 17. Action View	127
Szablony RHTML	127
Helpers — wsparcie pomocników	130
Formatowanie	133
Odnosińniki	136
Stronicowanie	137
Elementy formularzy	138
Podsumowanie	140
Rozdział 18. Action Mailer	141
Wysyłanie poczty	141
Odbieranie poczty	147
Podsumowanie	149
Rozdział 19. Web 2.0	151
link_to_remote	151
observe_field	154
periodically_call_remote	155
Podsumowanie	157
Rozdział 20. Bezpieczeństwo	159
SQL Injection	159
CSS/XSS	161
Formularze	162
Publiczne metody kontrolerów	163
Upload plików	164
Podsumowanie	164
Rozdział 21. Zakończenie	165
Część III Dodatki	167
Dodatek A Projekt — TimeTracker	169
Przygotowanie	169
Wersja SQLite	170
Wersja MySQL	172
Opis działania	174
Opis budowy	178
Podsumowanie	186



Pierwsza aplikacja

Przypomnienie



Zazwyczaj rozpoczynając przygodę z jakimkolwiek językiem programowania, pierwszą aplikacją, która wychodzi spod naszych rąk, jest znane wszystkim „Witaj świecie”. Tak też będzie w tym przypadku.

Ć W I C Z E N I E

14.1 Pierwsza aplikacja

Aby stworzyć pierwszą aplikację:

1. Uruchom edytor tekstowy.
2. Zapisz poniższy kod źródłowy do nowego pliku.

```
class Osoba
  @@ilosc = 0
  def initialize(imie, nazwisko, wiek)
    @imie = imie
    @nazwisko = nazwisko
    @wiek = wiek
    @@ilosc += 1
  end

  def przywitaj_sie
    puts "Nazywam sie #{@imie} #{@nazwisko}."
    puts "Mam #{@wiek} lat."
  end
end
```

```
def Osoba.ile_osob
  puts "Utworzono #{@@ilosc} osob!"
end
end

pracownik_1 = Osoba.new("Michał", "Sobczak", 19)
pracownik_1.przywitalaj_sie
Osoba.ile_osob
```

3. Zapisz plik i wykonaj go w interpreterze języka Ruby. Wpisz w wierszu poleceń ruby <nazwa pliku>.

W powyższym przykładzie utworzyliśmy klasę `Osoba`. Ponadto zdefiniowaliśmy trzy metody. Pierwsza z nich to metoda inicjalizacji obiektu danej klasy. Druga to metoda `przywitalaj_sie`, która jest odpowiedzialna za wyświetlenie na ekranie podanego tekstu w postaci imienia, nazwiska oraz wieku. Ostatnia z nich wyświetla liczbę utworzonych do tej pory obiektów klasy `Osoba`. Za przechowywanie informacji o liczbie tych obiektów odpowiedzialna jest zmienna klasy nosząca nazwę `ilosc`.

Przykład ten miał na celu przypomnienie składni języka. Mając za sobą wprowadzenie do środowiska oraz ogólny przegląd jego możliwości w poprzednich rozdziałach, czas na zastosowanie naszej jak dotąd skromnej wiedzy w praktyce.

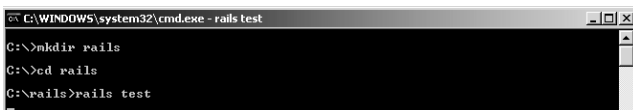
Drzewo projektu

Ć W I C Z E N I E

14.2 Tworzenie aplikacji wraz z drzewem projektu

Aby utworzyć nową aplikację, a wraz z nią drzewo projektu:

1. Uruchom wiersz poleceń.
2. Wpisz polecenie rails <nazwa aplikacji> (rysunek 14.1).



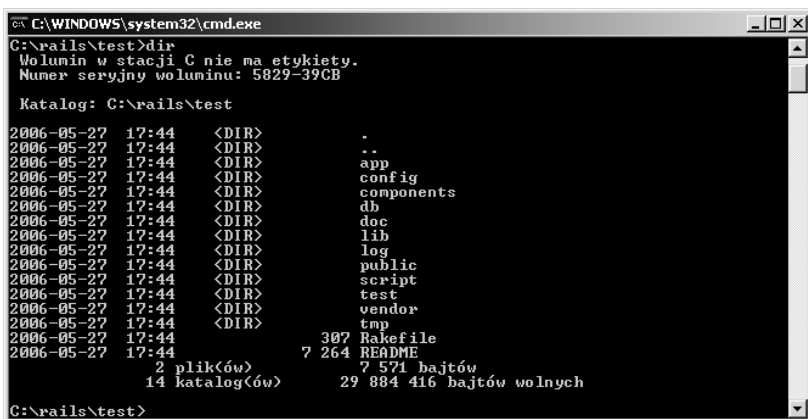
Rysunek 14.1. Tworzenie nowej aplikacji w Rails

Ć W I C Z E N I E

14.3 Przygotowanie środowiska pracy

Zaczynając pracę z Rails, niezmiernie istotne jest, aby wybrać katalog, w którym będziemy przechowywać nasze projekty. Ważne jest, żeby katalog ten znajdował się jak najwyżej w strukturze katalogów dysku twardego.

1. W wierszu poleceń wpisz `mkdir <nazwa katalogu>`, żeby utworzyć nowy katalog.
2. Do katalogu przechodzimy poleceniem `cd <nazwa katalogu>`.
3. Żeby wyświetlić wszystkie pliki oraz katalogi w katalogu, w którym znajdujemy się obecnie, używamy polecenia `dir` (rysunek 14.2).



```
C:\WINDOWS\system32\cmd.exe
C:\rails\test>dir
Wolumin w stacji C nie ma etykiety.
Numer seryjny woluminu: 5829-39CB

Katalog: C:\rails\test

2006-05-27 17:44 <DIR>      .
2006-05-27 17:44 <DIR>      ..
2006-05-27 17:44 <DIR>      app
2006-05-27 17:44 <DIR>      config
2006-05-27 17:44 <DIR>      components
2006-05-27 17:44 <DIR>      db
2006-05-27 17:44 <DIR>      doc
2006-05-27 17:44 <DIR>      lib
2006-05-27 17:44 <DIR>      log
2006-05-27 17:44 <DIR>      public
2006-05-27 17:44 <DIR>      script
2006-05-27 17:44 <DIR>      test
2006-05-27 17:44 <DIR>      vendor
2006-05-27 17:44 <DIR>      tmp
2006-05-27 17:44          397 Rakefile
2006-05-27 17:44          7 264 README
                2 plik(ów)          7 571 bajtów
                14 katalog(ów)      29 884 416 bajtów wolnych

C:\rails\test>
```

Rysunek 14.2. Drzewo projektu

4. Żeby wyświetlić to samo, ale rekursywnie, czyli wraz z zawartością podkatalogów, wpisujemy `dir /S`.

Ć W I C Z E N I E

14.4 Obsługa wbudowanego w wiersz poleceń edytora tekstu

Aby uruchomić wbudowany w wiersz poleceń edytor tekstu:

1. Uruchom wiersz poleceń.

2. Edytor uruchamiamy poleceniem `edit`. Używając edytora `edit`, możemy poruszać się pomiędzy poleceniami menu edytora na dwa sposoby — za pomocą myszy albo klawiatury.
3. Żeby dostać się do menu za pomocą klawiatury, używamy kombinacji klawiszy `Alt`+wiodąca litera menu.
4. Na przykład, aby dostać się do menu *Plik*, wybierzmy `Alt+p`. Kiedy naciśniemy klawisz `Alt`, podświetlone zostaną wiodące litery menu.

Jeśli chcesz zapoznać się ze znaczeniem poszczególnych katalogów z drzewie projektu, sięgnij do dokumentacji Ruby on Rails, która znajduje się pod adresem <http://www.rubyonrails.org/docs>.

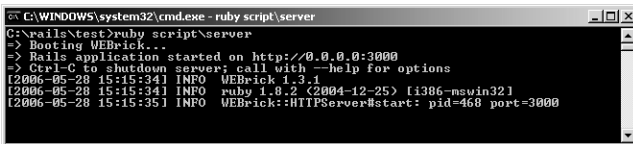
WEBrick

Ć W I C Z E N I E

14.5 Uruchomienie serwera WEBrick

Aby uruchomić serwer WEBrick:

1. Uruchom wiersz poleceń.
2. Aby uruchomić serwer WEBrick, wydajemy polecenie `ruby script/server` (rysunek 14.3).



```

C:\WINDOWS\system32\cmd.exe - ruby script/server
C:\rails>ruby script/server
-> Booting WEBrick...
-> Rails application started on http://0.0.0.0:3000
-> Ctrl-C to shutdown server; call with --help for options
[2006-05-28 15:15:34] INFO WEBrick 1.3.1
[2006-05-28 15:15:34] INFO ruby 1.8.2 (2004-12-25) [i386-mswin32]
[2006-05-28 15:15:35] INFO WEBrick::HTTPServer#start: pid=468 port=3000
```

Rysunek 14.3. Uruchomienie serwera WEBrick

Witaj świecie

Kilka chwil wcześniej stworzyliśmy nowy projekt poleceniem rails <nazwa projektu> i w związku z tym mamy już całe drzewo aplikacji, które będzie dla nas szablonem do tworzenia w późniejszym czasie bardziej zaawansowanych aplikacji niż obecne „Witaj świecie”.

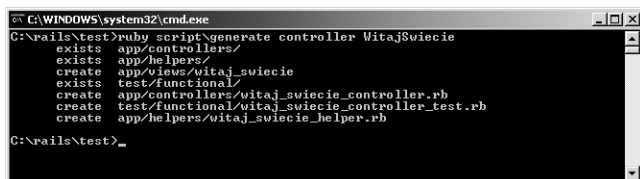
Stwórzmy zatem aplikację, która będzie składała się z jednego kontrolera oraz kilku metod. Do tej operacji będziemy potrzebowali podstawowej znajomości **HTML** i **CSS**¹.

Ć W I C Z E N I E

14.6 Tworzenie pierwszego kontrolera i widoku

Aby utworzyć kontroler:

1. Uruchom wiersz poleceń.
2. Przejdź do katalogu z projektem, na przykład `cd rails\test`.
3. Wykonaj polecenie `ruby script\generate controller WitajSwiecie`, aby stworzyć nowy kontroler (rysunek 14.4).



```
C:\WINDOWS\system32\cmd.exe
C:\rails\test>ruby script\generate controller WitajSwiecie
exists app\controllers/
exists app\helpers/
create app\views\witaj_swiecie
exists test\functional/
create app\controllers\witaj_swiecie_controller.rb
create test\functional\witaj_swiecie_controller_test.rb
create app\helpers\witaj_swiecie_helper.rb
C:\rails\test>_
```

Rysunek 14.4. Tworzenie kontrolera WitajSwiecie

4. Przejdź do katalogu `app\controllers` poleceniem `cd app\controllers`.
5. Wykonaj polecenie `dir` w celu przejrzania zawartości katalogu. W tym momencie widzimy dwa pliki: `application.rb` oraz `witaj_swiecie_controller.rb`. Są to kontrolery akcji. W tym przypadku najbardziej interesuje nas `witaj_swiecie_controller.rb`.
6. Edytuj plik `witaj_swiecie_controller.rb` (rysunek 14.5).

¹ Kaskadowe arkusze stylów (ang. *Cascading Style Sheets*, CSS) jest to język służący do opisu sposobu generowania stron WWW — *przyp. aut.*

Rysunek 14.5.

Edytor gVim
z otwartym
plikiem kontrolera
WitajSwiecie



7. Zapisz do pliku *witaj_swiecie_controller.rb* następujący kod:

```
class WitajSwiecieController < ApplicationController
  def witaj
  end
end
```

8. Otwórz przeglądarkę, na przykład Internet Explorer, i wpisz:
http://127.0.0.1:3000/witaj_swiecie (rysunek 14.6).

Rysunek 14.6.

Nieznamą akcję
(unknown action)



9. Po wpisaniu http://127.0.0.1:3000/witaj_swiecie/witaj otrzymamy błąd innego rodzaju. Poprzedni mówi nam, że wprawdzie mamy taki kontroler, ale nie mamy żadnej akcji z nim związanej. Z tego powodu musimy stworzyć widok dla kontrolera, żeby mógł się wyświetlić (rysunek 14.7).

Rysunek 14.7.

Brak szablonu
(template is
missing)



10. cd `..\views` oraz cd `witaj_swiecie`.
11. Stwórz plik szablonu o nazwie, jaką posiada metoda w kontrolerze `WitajSwiecie`, czyli *witaj.rhtml*:

```
<html>
  <head>
    <title>Witaj swiecie!</title>
    <style type="text/css">
      table.tabela
      {font-weight:bold;}
      div
      {text-align:center;}
    </style>
```

```
</head>
<body>
  <h1>Witaj świecie!</h1>
  <div>
    <table class="tabela">
      <tr><td>Witaj świecie!</td></tr>
    </table>
  </div>
</body>
</html>
```

12. Teraz pod adresem <http://127.0.0.1:3000/WitajSwiecie/witaj> możesz zobaczyć swoją pierwszą działającą aplikację. Gratulacje! To jednak nie wszystko.

13. Edytuj plik `witaj_swiecie_controller.rb`.

```
class WitajSwiecieController < ApplicationController
  def witaj
    5.times do |@i| @i+=1 end
  end
end
```

14. Edytuj plik `witaj.rhtml`.

```
<table class="tabela">
  <tr>
    <td>Witaj świecie!</td>
    <td><%= @i %>
  </tr>
</table>
```

Wynikiem powyższych działań jest wyświetlenie liczby 5. Jak widać, kontrolery oraz widoki współpracują ze sobą, przekazując sobie nazwajem wartości zmiennych.

Znaczniki

Znaczniki pozwalają na osadzanie instrukcji języka Ruby pomiędzy kodem w języku HTML.

14.7 Osadzanie znaczników Ruby w HTML

Aby osadzić znaczniki języka Ruby stronie HTML, podążaj za poniższym opisem:

1. Wartości wyrażeń w plikach *rhtml* zapisujemy pomiędzy znacznikami `<%=` oraz `%>`. Wszystko pomiędzy nimi jest zamieniane na łańcuch znaków i wyświetlane w przeglądarce. Załóżmy, że stworzymy taką konstrukcję:

```
<%= puts "Witaj!" %>
```

2. Zostanie to co prawda wyświetlone, ale nie tam, gdzie moglibyśmy się tego spodziewać. Pamiętamy okno wiersza poleceń, w którym uruchamialiśmy WEBrick? Jeśli użyjemy standardowego wyjścia dla danych, pojawią one się właśnie w konsoli z uruchomionym serwerem.

```
<%= 1+2 %>  
<%= "jeden" + "dwa" $>
```

3. Istnieje ponadto druga para znaczników `<%`, `%>`. Wszystko pomiędzy nim może być, tak jak w przypadku poprzedników, kodem źródłowych języka Ruby. Różnica polega na tym, że znacznik ten nie zamienia swojego wyjścia na łańcuch znaków i nie próbuje go wyświetlić. Znacznik ten jest najczęściej stosowany do iteracji oraz pętli i warunków wyświetlania kodu HTML.

```
<% 3.times do %>  
  Witaj!<br>  
<% end %>
```

4. Istnieje możliwość łączenia obu znaczników.

```
<% 4.times do |i| %>  
  <%= i %> <br>  
<% end %>
```

5. Na zakończenie czas stworzyć drugą metodę klasy kontrolera, stworzyć widok oraz połączyć obie strony odnośnikami. Edytuj plik *witaj_swiecie_controller.rb*.

```
class WitajSwiecieController < ApplicationController  
  def witaj  
    5.times do |@i| @i+=1 end  
  end  
end
```

```
def doZobaczenia
end
end
```

6. Edytuj plik widoku *doZobaczenia.rhtml*.

```
<html>
  <head>
    <title>Do Zobaczenia!</title>
  </head>
  <body>
    Do zobaczenia!
    <br>
    <%= link_to "Witaj!", :action => "witaj" %>
  </body>
</html>
```

7. Do pliku *witaj.rhtml* dodaj:

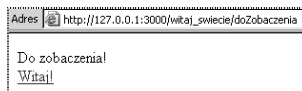
```
<%= link_to "Do Zobaczenia!", :action => "doZobaczenia" %>
```

8. Obejrzyj wynik tego ćwiczenia w przeglądarce internetowej pod adresem *http://127.0.0.1:3000/witaj_swiecie/witaj* (rysunki 14.8 i 14.9).

Rysunek 14.8.
Pierwsza aplikacja



Rysunek 14.9.
Działanie odnośnika



Podsumowanie

W rozdziale:

- ❑ Przypomnieliśmy sobie składnię języka Ruby.
- ❑ Utworzyliśmy szkielet aplikacji Rails.
- ❑ Uruchomiliśmy serwer WEBrick.
- ❑ Stworzyliśmy przykładowy kontroler oraz dwa widoki.
- ❑ Dowiedzieliśmy się, jak wykorzystywać znaczniki osadzania kodu źródłowego języka Ruby pomiędzy kodem HTML.
- ❑ Połączyliśmy dwie strony odsyłaczami.