

EPI: Interfejs Graficzny

Wykład nr 4

Podstawy frameworku Rails

Aleksander Smywiński-Pohl

Plan prezentacji

Framework Rails

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem
- ▶ **Wzorzec MVC**:

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem
- ▶ **Wzorzec MVC**:
 - ▶ model: mapowanie obiektowo-relacyjne, migracje, zunifikowana obsługa wielu RDMS-ów, etc.

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem
- ▶ **Wzorzec MVC**:
 - ▶ model: mapowanie obiektowo-relacyjne, migracje, zunifikowana obsługa wielu RDMS-ów, etc.
 - ▶ widok: szablony, layouty, formularze, AJAX

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem
- ▶ **Wzorzec MVC**:
 - ▶ model: mapowanie obiektowo-relacyjne, migracje, zunifikowana obsługa wielu RDMS-ów, etc.
 - ▶ widok: szablony, layouty, formularze, AJAX
 - ▶ kontroler: obsługa żądań, mapowanie adresów URL, cache'owanie, etc.

Ruby on Rails – podstawowe założenia

- ▶ **DRY**: nie powtarzaj się
- ▶ **Konwencja ponad konfiguracją**: domyślna struktura katalogów, STI, 3 środowiska pracy, jedna baza danych, ...
- ▶ **REST**: podstawowy sposób organizacji zasobów i interakcji ze światem
- ▶ **Wzorzec MVC**:
 - ▶ model: mapowanie obiektowo-relacyjne, migracje, zunifikowana obsługa wielu RDMS-ów, etc.
 - ▶ widok: szablony, layouty, formularze, AJAX
 - ▶ kontroler: obsługa żądań, mapowanie adresów URL, cache'owanie, etc.
- ▶ **Generatory** – szablony dla modeli, rusztowań, migracji bazy danych, itp.

Ruby on Rails – cd.

▶ konsola

Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail

Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji

Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji
- ▶ wsparcie dla umiędzynarodowienia aplikacji

Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji
- ▶ wsparcie dla umiędzynarodowienia aplikacji
- ▶ mechanizm kompozycji aplikacji

Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji
- ▶ wsparcie dla umiędzynarodowienia aplikacji
- ▶ mechanizm kompozycji aplikacji
- ▶ kompresja CSS-ów i Javascriptu

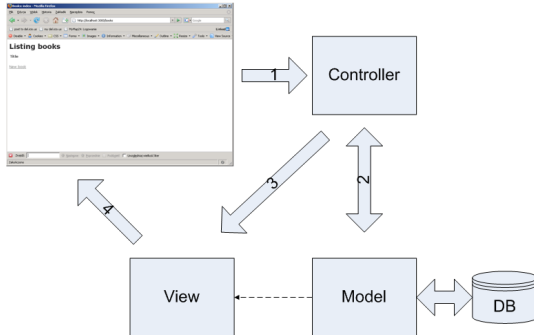
Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji
- ▶ wsparcie dla umiędzynarodowienia aplikacji
- ▶ mechanizm kompozycji aplikacji
- ▶ kompresja CSS-ów i Javascriptu
- ▶ wsparcie dla CoffeeScriptu

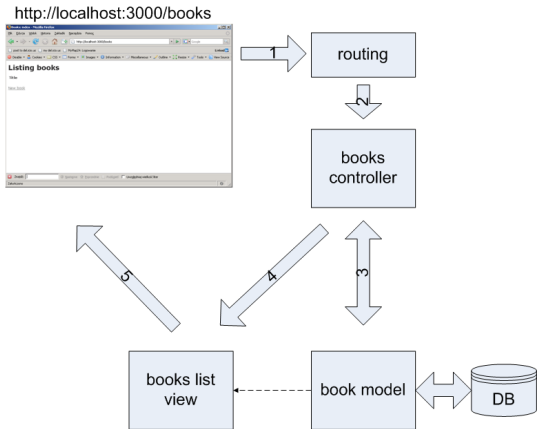
Ruby on Rails – cd.

- ▶ konsola
- ▶ zintegrowany mechanizm obsługi wiadomości e-mail
- ▶ zintegrowany mechanizm testowania aplikacji
- ▶ wsparcie dla umiędzynarodowienia aplikacji
- ▶ mechanizm kompozycji aplikacji
- ▶ kompresja CSS-ów i Javascriptu
- ▶ wsparcie dla CoffeeScriptu
- ▶ **tysiące** plug-inów

MVC – Model Widok Kontroler



MVC w RoR



Konwencje nazewnnicze

- ▶ **model** pisany w notacji wielbłędziej w liczbie pojedynczej
np. `class Book`

Konwencje nazewnictwa

- ▶ **model** pisany w notacji wielbłądziej w liczbie pojedynczej
np. `class Book`
- ▶ **kontroler** w notacji wielbłądziej w liczbie mnogiej
np. `BooksController`

Konwencje nazewnnicze

- ▶ **model** pisany w notacji wielbłądziej w liczbie pojedynczej
np. `class Book`
- ▶ **kontroler** w notacji wielbłądziej w liczbie mnogiej
np. `BooksController`
- ▶ **widok** domyślnie ma taką samą nazwę jak akcja w kontrolerze + format (HTML, XML, JSON) + język szablonów (erb,haml)
np. `BooksController#index` → `index.html.erb`

Konwencje nazewnnicze

- ▶ **model** pisany w notacji wielbłądziej w liczbie pojedynczej
np. `class Book`
- ▶ **kontroler** w notacji wielbłądziej w liczbie mnogiej
np. `BooksController`
- ▶ **widok** domyślnie ma taką samą nazwę jak akcja w kontrolerze + format (HTML, XML, JSON) + język szablonów (erb,haml)
np. `BooksController#index` → `index.html.erb`
- ▶ **pojedynczy zasób** pisany małą literą w liczbie pojedynczej
np. `resource :book`

Konwencje nazewnnicze

- ▶ **model** pisany w notacji wielbłądziej w liczbie pojedynczej
np. `class Book`
- ▶ **kontroler** w notacji wielbłądziej w liczbie mnogiej
np. `BooksController`
- ▶ **widok** domyślnie ma taką samą nazwę jak akcja w kontrolerze + format (HTML, XML, JSON) + język szablonów (erb,haml)
np. `BooksController#index` → `index.html.erb`
- ▶ **pojedynczy zasób** pisany małą literą w liczbie pojedynczej
np. `resource :book`
- ▶ **wielokrotny zasób** pisany małą literą w liczbie mnogiej
np. `resources :books`

Warstwa modelu – ActiveRecord

- ▶ obiektowy dostęp do bazy danych

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`,
`book.title`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`, `book.title`
 - ▶ **update** – metody `update_attribute`, `update_attributes`, np. `book.update_attribute(:title, 'Dziady')`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`, `book.title`
 - ▶ **update** – metody `update_attribute`, `update_attributes`, np. `book.update_attribute(:title, 'Dziady')`
 - ▶ **delete** – metoda `destroy`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`, `book.title`
 - ▶ **update** – metody `update_attribute`, `update_attributes`, np. `book.update_attribute(:title, 'Dziady')`
 - ▶ **delete** – metoda `destroy`
- ▶ **związki** – wysokopoziomowe określanie zależności pomiędzy klasami, np. `Book has_one :author → book.author`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`, `book.title`
 - ▶ **update** – metody `update_attribute`, `update_attributes`, np. `book.update_attribute(:title, 'Dziady')`
 - ▶ **delete** – metoda `destroy`
- ▶ **związki** – wysokopoziomowe określanie zależności pomiędzy klasami, np. `Book has_one :author → book.author`
- ▶ **walidacje** – weryfikacja występowania wartości atrybutu, jego formatu, itp., np. `validates_presence_of :name`

Warstwa modelu – ActiveRecord

- ▶ **obiektowy** dostęp do bazy danych
- ▶ implementacja operacji **CRUD**
 - ▶ **create** – konstruktor klasy, np. `Book.new`
 - ▶ **read** – metoda `find` oraz akcesory pól, np. `Book.find(1)`, `book.title`
 - ▶ **update** – metody `update_attribute`, `update_attributes`, np. `book.update_attribute(:title, 'Dziady')`
 - ▶ **delete** – metoda `destroy`
- ▶ **związki** – wysokopoziomowe określanie zależności pomiędzy klasami, np. `Book has_one :author → book.author`
- ▶ **walidacje** – weryfikacja występowania wartości atrybutu, jego formatu, itp., np. `validates_presence_of :name`
- ▶ **migracje** – pozwalają na inkrementalne określanie i modyfikowanie schematu bazy danych

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**
 - ▶ czasowniki HTTP: GET, POST, PUT, DELETE

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**
 - ▶ czasowniki HTTP: GET, POST, PUT, DELETE
 - ▶ dostępność treści w różnych formatach – np. HTML, JSON

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**
 - ▶ czasowniki HTTP: GET, POST, PUT, DELETE
 - ▶ dostępność treści w różnych formatach – np. HTML, JSON
 - ▶ predefiniowane akcje: `index`, `show`, `new`, `create`, `edit`, `update`, `destroy`

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**
 - ▶ czasowniki HTTP: GET, POST, PUT, DELETE
 - ▶ dostępność treści w różnych formatach – np. HTML, JSON
 - ▶ predefiniowane akcje: `index`, `show`, `new`, `create`, `edit`, `update`, `destroy`
- ▶ każda akcja domyślnie posiada odpowiadający jej **widok**

Warstwa kontrolera – ActionController

- ▶ **reaguje** na żądania przeglądarki
- ▶ **łączy** warstwę modelu z warstwą widoku
- ▶ definiuje **akcje** (jako metody Rubiego)
- ▶ przyjmuje podstawowe założenia koncepcji **REST**
 - ▶ czasowniki HTTP: GET, POST, PUT, DELETE
 - ▶ dostępność treści w różnych formatach – np. HTML, JSON
 - ▶ predefiniowane akcje: `index`, `show`, `new`, `create`, `edit`, `update`, `destroy`
- ▶ każda akcja domyślnie posiada odpowiadający jej **widok**
- ▶ **filtry** pozwalają na łatwe dodanie zadań do wybranych akcji, np. autoryzacja, kompresja

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)
- ▶ może definiować **wiele sposobów** prezentacji danych

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)
- ▶ może definiować **wiele sposobów** prezentacji danych
- ▶ oparta jest na wybranym **języku szablonów** (erb, haml, itp.)

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)
- ▶ może definiować **wiele sposobów** prezentacji danych
- ▶ oparta jest na wybranym **języku szablonów** (erb, haml, itp.)
- ▶ definiuje **funkcje pomocnicze** (*helpers*), do łatwego tworzenia tagów HTML:
 - ▶ linków
 - ▶ obrazków
 - ▶ formularzy

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)
- ▶ może definiować **wiele sposobów** prezentacji danych
- ▶ oparta jest na wybranym **języku szablonów** (erb, haml, itp.)
- ▶ definiuje **funkcje pomocnicze** (*helpery*), do łatwego tworzenia tagów HTML:
 - ▶ linków
 - ▶ obrazków
 - ▶ formularzy
- ▶ szablony mogą składać się z wielu **pod-szablonów** (*partials*)

Warstwa widoku – ActionView

- ▶ **oddziela** dane od metody ich prezentowania (HTML, JSON, itp.)
- ▶ może definiować **wiele sposobów** prezentacji danych
- ▶ oparta jest na wybranym **języku szablonów** (erb, haml, itp.)
- ▶ definiuje **funkcje pomocnicze** (*helpery*), do łatwego tworzenia tagów HTML:
 - ▶ linków
 - ▶ obrazków
 - ▶ formularzy
- ▶ szablony mogą składać się z wielu **pod-szablonów** (*partials*)
- ▶ aplikacja może definiować jeden/wiele **layoutów**

Warstwa trasowania

- ▶ zastępuje rozwiązania typu `mod_rewrite`

Warstwa trasowania

- ▶ zastępuje rozwiązania typu `mod_rewrite`
- ▶ interpretuje żądania przychodzące do aplikacji

Warstwa trasowania

- ▶ zastępuje rozwiązania typu `mod_rewrite`
- ▶ interpretuje żądania przychodzące do aplikacji
- ▶ przekazuje żądania do kontrolera

Warstwa trasowania

- ▶ **zastępuje** rozwiązania typu `mod_rewrite`
- ▶ **interpretuje** żądania przychodzące do aplikacji
- ▶ **przekazuje** żądania do kontrolera
- ▶ pozwala na całkowite **wyabstrahowanie** mechanizmu tworzenia linków wewnątrz aplikacji

Warstwa trasowania

- ▶ zastępuje rozwiązania typu `mod_rewrite`
- ▶ interpretuje żądania przychodzące do aplikacji
- ▶ przekazuje żądania do kontrolera
- ▶ pozwala na całkowite **wyabstrahowanie** mechanizmu tworzenia linków wewnątrz aplikacji
- ▶ pozwala na tworzenie **przyjaznych adresów URL**
 - ▶ `/ksiazki/1-dziady-cz-IV`
 - ▶ `/2011/11/11`

Warstwa trasowania

- ▶ zastępuje rozwiązania typu `mod_rewrite`
- ▶ interpretuje żądania przychodzące do aplikacji
- ▶ przekazuje żądania do kontrolera
- ▶ pozwala na całkowite **wyabstrahowanie** mechanizmu tworzenia linków wewnątrz aplikacji
- ▶ pozwala na tworzenie **przyjaznych adresów URL**
 - ▶ `/ksiazki/1-dziady-cz-IV`
 - ▶ `/2011/11/11`
- ▶ koncentruje się wokół koncepcji **REST**
 - ▶ `resource :book`
 - ▶ `get 'book/latest'`
 - ▶ `post 'book/:id/review'`

Warstwa trasowania

- ▶ **zastępuje** rozwiązania typu `mod_rewrite`
- ▶ **interpretuje** żądania przychodzące do aplikacji
- ▶ **przekazuje** żądania do kontrolera
- ▶ pozwala na całkowite **wyabstrahowanie** mechanizmu tworzenia linków wewnątrz aplikacji
- ▶ pozwala na tworzenie **przyjaznych adresów URL**
 - ▶ `/ksiazki/1-dziady-cz-IV`
 - ▶ `/2011/11/11`
- ▶ koncentruje się wokół koncepcji **REST**
 - ▶ `resource :book`
 - ▶ `get 'book/latest'`
 - ▶ `post 'book/:id/review'`
- ▶ pozwala na automatyczne **przekierowywanie** żądań

Materiały

- ▶ nie ma sensu kupować książek na temat Rails!

Materiały

- ▶ **nie ma sensu kupować książek na temat Rails!**
- ▶ podstawowe źródło informacji – przewodniki
`guides.rubyonrails.org`

Materiały

- ▶ **nie ma sensu kupować książek na temat Rails!**
- ▶ podstawowe źródło informacji – przewodniki
`guides.rubyonrails.org`
- ▶ wersja polska przewodników – trochę przestarzała
`apohllo.pl/guides/index.html`

Materiały

- ▶ **nie ma sensu kupować książek na temat Rails!**
- ▶ podstawowe źródło informacji – przewodniki
`guides.rubyonrails.org`
- ▶ wersja polska przewodników – trochę przestarzała
`apohllo.pl/guides/index.html`
- ▶ interaktywny tutorial – Rails for Zombies
`railsforzombies.org`

Materiały

- ▶ **nie ma sensu kupować książek na temat Rails!**
- ▶ podstawowe źródło informacji – przewodniki
`guides.rubyonrails.org`
- ▶ wersja polska przewodników – trochę przestarzała
`apohllo.pl/guides/index.html`
- ▶ interaktywny tutorial – Rails for Zombies
`railsforzombies.org`
- ▶ railscasty Rayana Batesa
`railscasts.com`

Materiały

- ▶ **nie ma sensu kupować książek na temat Rails!**
- ▶ podstawowe źródło informacji – przewodniki
guides.rubyonrails.org
- ▶ wersja polska przewodników – trochę przestarzała
apohllo.pl/guides/index.html
- ▶ interaktywny tutorial – Rails for Zombies
railsforzombies.org
- ▶ railscasty Rayana Batesa
railscasts.com
- ▶ RubyToolbox – mnóstwo przydatnych plug-inów, informacje na temat popularności, sposobu instalacji, itp.
www.ruby-toolbox.com