

# Wykład nr 7

## Trasowanie i warstwa kontrolera – `routes.rb` i ActionController

Aleksander Smywiński-Pohl

# Plan prezentacji

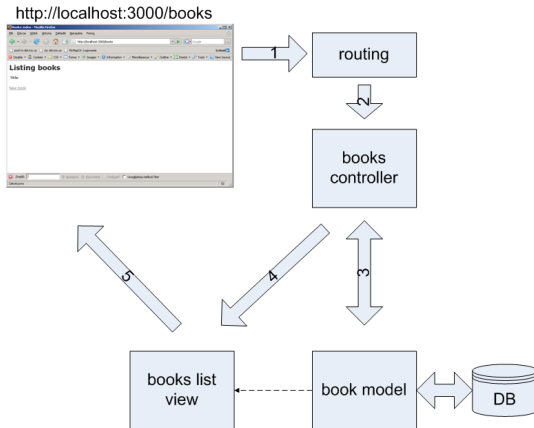
Wprowadzenie

Trasowanie

Warstwa kontrolera

Obsługa stanu

# MVC w RoR



## Trasowanie – config/routes.rb

- ▶ zastępuje rozwiązania typu `mod_rewrite`

## Trasowanie – config/routes.rb

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

## Trasowanie – config/routes.rb

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

## Trasowanie – config/routes.rb

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

## Trasowanie – config/routes.rb

- ▶ 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`



## Trasowanie – config/routes.rb

- ▶ 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`
- ▶ zbudowane w oparciu o **REST**

## Trasowanie – config/routes.rb

- ▶ 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`
- ▶ zbudowane w oparciu o **REST**
- ▶ pozwala na automatyczne **przekierowywanie** żądań

# 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**
- ▶ 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**
- ▶ każda akcja domyślnie posiada odpowiadający jej **widok**
- ▶ **filtry** pozwalają na łatwe dodanie zadań do wybranych akcji, np. autoryzacja, kompresja



# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB

# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB
- ▶ wykorzystuje dobrze znane cechy protokołu HTTP
  - ▶ adresy URL

# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB
- ▶ wykorzystuje dobrze znane cechy protokołu HTTP
  - ▶ adresy URL
  - ▶ czasowniki: GET, POST, PUT, DELETE

# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB
- ▶ wykorzystuje dobrze znane cechy protokołu HTTP
  - ▶ adresy URL
  - ▶ czasowniki: GET, POST, PUT, DELETE
  - ▶ formaty: HTML, JSON, XML

# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB
- ▶ wykorzystuje dobrze znane cechy protokołu HTTP
  - ▶ adresy URL
  - ▶ czasowniki: GET, POST, PUT, DELETE
  - ▶ formaty: HTML, JSON, XML
  - ▶ pozostałe: proxy, firewall, caching, mime, etc.

# REST – Representational State Transfer

- ▶ architektura zaprojektowana dla rozproszonych serwisów webowych, np.:
  - ▶ Amazon S3
  - ▶ CKAN
  - ▶ CouchDB
- ▶ wykorzystuje dobrze znane cechy protokołu HTTP
  - ▶ adresy URL
  - ▶ czasowniki: GET, POST, PUT, DELETE
  - ▶ formaty: HTML, JSON, XML
  - ▶ pozostałe: proxy, firewall, caching, mime, etc.
- ▶ centralne pojęcie: **zasób**

# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL

# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL
  - ▶ `http://example.com/cars` – kolekcja zasobów



# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL
  - ▶ `http://example.com/cars` – kolekcja zasobów
  - ▶ `http://example.com/cars/1` – pojedynczy zasób

# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL
  - ▶ `http://example.com/cars` – kolekcja zasobów
  - ▶ `http://example.com/cars/1` – pojedynczy zasób
- ▶ odczytywanie/modyfikowanie zasobu z wykorzystaniem dokumentów opisujących stan

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <author-id type="integer">1</author-id>
  <id type="integer">1</id>
  <title>Alef</title>
</book>
```

# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL
  - ▶ `http://example.com/cars` – kolekcja zasobów
  - ▶ `http://example.com/cars/1` – pojedynczy zasób
- ▶ odczytywanie/modyfikowanie zasobu z wykorzystaniem dokumentów opisujących stan

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <author-id type="integer">1</author-id>
  <id type="integer">1</id>
  <title>Alef</title>
</book>
```

- ▶ CRUD – czasowniki HTTP: GET, POST, PUT, DELETE

# REST – zasady działania

- ▶ identyfikowanie zasobów poprzez adres URL
  - ▶ `http://example.com/cars` – kolekcja zasobów
  - ▶ `http://example.com/cars/1` – pojedynczy zasób
- ▶ odczytywanie/modyfikowanie zasobu z wykorzystaniem dokumentów opisujących stan

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <author-id type="integer">1</author-id>
  <id type="integer">1</id>
  <title>Alef</title>
</book>
```

- ▶ CRUD – czasowniki HTTP: GET, POST, PUT, DELETE
- ▶ linki w dokumencie identyfikują inne możliwe akcje do wykonania

# Plan prezentacji

Wprowadzenie

Trasowanie

Warstwa kontrolera

Obsługa stanu

# Definicja trasowania – plik config/routes.rb

```
Library::Application.routes.draw do
  resources :books           # wiele zasobów
  resources :authors do     # zasoby zagnieżdżone
    resources :books
  end
  resource :user_session    # jeden zasób

  root :to => 'books#index' # strona startowa

  post 'login' => "user_sessions#new"
                                # określony kontroler i akcja
                                # oraz zmiana adresu URL
  post 'logout' => "user_sessions#destroy", :as => :logout
                                # zmiana adresu oraz
                                # utworzenie helpera 'logout_path'
end
```

## Użycie trasowania – helper `link_to`

```

<%= link_to "książki", :controller => "books", :action => "index" %>
  <a href="/books">książki</a>

<%= link_to "książki", books_path %>
  <a href="/books">książki</a>

<%= link_to "nowa książka", new_book_path %>
  <a href="/books/new">nowa książka</a>

<%= link_to @book.title, @book %>
  <a href="/books/1">Mały Książę</a>

<%= link_to @author.books.first.title, [@author, @author.books.first] %>
  <a href="/authors/2/books/1"/>Mały Książę</a>

<%= link_to "start", root_url %>
  <a href="/">start</a>

<%= link_to "wyloguj", logout_path %>
  <a href="/logout">wyloguj</a>

```

# Wyświetlenie dostępnych tras

## ▷ rake routes

```
books GET /books(.:format)
      {:action=>"index", :controller=>"books"}
books POST /books(.:format)
      {:action=>"create", :controller=>"books"}
new_book GET /books/new(.:format)
      {:action=>"new", :controller=>"books"}
edit_book GET /books/:id/edit(.:format)
      {:action=>"edit", :controller=>"books"}
book GET /books/:id(.:format)
     PUT /books/:id(.:format)
      {:action=>"update", :controller=>"books"}
DELETE /books/:id(.:format)
      {:action=>"destroy", :controller=>"books"}
```



# Kolekcja zasobów

resources :books

| HTTP   | URL             | akcja   | przeznaczenie            |
|--------|-----------------|---------|--------------------------|
| GET    | /books          | index   | lista książek            |
| GET    | /books/new      | new     | formularz nowej książki  |
| POST   | /books          | create  | utworzenie nowej książki |
| GET    | /books/:id      | show    | wyświetlenie książki     |
| GET    | /books/:id/edit | edit    | formularz edycji książki |
| PUT    | /books/:id      | update  | zmodyfikowanie książki   |
| DELETE | /books/:id      | destroy | usunięcie książki        |

## Kolekcja zasobów – helper

```
resources :books
```

| HTTP   | URL             | akcja   | helper         |
|--------|-----------------|---------|----------------|
| GET    | /books          | index   | books_path     |
| GET    | /books/new      | new     | new_book_path  |
| POST   | /books          | create  |                |
| GET    | /books/:id      | show    | book_path(id)  |
| GET    | /books/:id/edit | edit    | edit_book_path |
| PUT    | /books/:id      | update  |                |
| DELETE | /books/:id      | destroy |                |

# Pojedynczy zasób

```
resource :user_session
```

| HTTP   | URL                | akcja   | przeznaczenie          |
|--------|--------------------|---------|------------------------|
| GET    | /user_session/new  | new     | formularz nowej sesji  |
| POST   | /user_session      | create  | utworzenie nowej sesji |
| GET    | /user_session      | show    | wyświetlenie sesji     |
| GET    | /user_session/edit | edit    | formularz edycji sesji |
| PUT    | /user_session      | update  | zmodyfikowanie sesji   |
| DELETE | /user_session      | destroy | usunięcie sesji        |

# Zagnieżdżone zasoby

```
resources :authors do
  resources :books
end
```

| HTTP   | URL                         | akcja   | przeznaczenie        |
|--------|-----------------------------|---------|----------------------|
| GET    | /authors/:id/books          | index   | lista książek autora |
| GET    | /authors/:id/books/new      | new     | formularz n. książki |
| POST   | /authors/:id/books          | create  | utworzenie książki   |
| GET    | /authors/:id/books/:id      | show    | wyświetlenie książki |
| GET    | /authors/:id/books/:id/edit | edit    | formularz e. książki |
| PUT    | /authors/:id/books/:id      | update  | zmodyf. książki      |
| DELETE | /authors/:id/books/:id      | destroy | usunięcie książki    |

## Dodatkowe akcje

```
resources :books do
  get 'abstract', :on => :member
  get 'recent', :on => :collection
end
```

## Dodatkowe akcje

```
resources :books do
  get 'abstract', :on => :member
  get 'recent', :on => :collection
end
```

- ▶ poza predefiniowanymi akcjami można dodawać własne

# Dodatkowe akcje

```
resources :books do
  get 'abstract', :on => :member
  get 'recent', :on => :collection
end
```

- ▶ poza predefiniowanymi akcjami można dodawać własne
- ▶ należy zdefiniować czy akcja definiowana jest dla
  - ▶ pojedynczego zasoby – member
  - ▶ grupy zasobów – collection

## Dodatkowe akcje

```
resources :books do
  get 'abstract', :on => :member
  get 'recent', :on => :collection
end
```

- ▶ poza predefiniowanymi akcjami można dodawać własne
- ▶ należy zdefiniować czy akcja definiowana jest dla
  - ▶ pojedynczego zasoby – `member`
  - ▶ grupy zasobów – `collection`
- ▶ jeśli tworzymy wiele akcji tego typu, należy zastanowić się czy nie powinniśmy dodać nowego zasobu (zagnieżdżonego)



# Helpery

```
# zasoby zwykłe
link_to @book.title, book_path(@book)
link_to @book.title, @book

link_to @book.title, abstract_book_path(@book)

link_to "Najnowsze książki", recent_books_path
```

# Helpery

```
# zasoby zwykłe
link_to @book.title, book_path(@book)
link_to @book.title, @book

link_to @book.title, abstract_book_path(@book)

link_to "Najnowsze książki", recent_books_path

# zasoby zagnieżdżone
link_to @author.books.first.title,
  author_book_path(@author,@author.books.first)

link_to @author.books.first.title, [@author,@author.books.first]
```

# Dodatkowe możliwości trasowania

▶ strona startowa

```
root :to => "books#index"
```

# Dodatkowe możliwości trasowania

- ▶ strona startowa

```
root :to => "books#index"
```

- ▶ obsługa adresu spoza REST

```
match 'contact' => 'posts#show', :name => 'contact'
```

# Dodatkowe możliwości trasowania

- ▶ strona startowa

```
root :to => "books#index"
```

- ▶ obsługa adresu spoza REST

```
match 'contact' => 'posts#show', :name => 'contact'
```

- ▶ segmenty opcjonalne w adresie

```
match ':controller(/:action(/:id))'
```

# Dodatkowe możliwości trasowania

- ▶ strona startowa

```
root :to => "books#index"
```

- ▶ obsługa adresu spoza REST

```
match 'contact' => 'posts#show', :name => 'contact'
```

- ▶ segmenty opcjonalne w adresie

```
match ':controller(/:action(/:id))'
```

- ▶ ograniczenia dla segmentów

```
match 'posts/:name' => "posts#show", :constraints => {:name => /\w+/}
```

# Dodatkowe możliwości trasowania

- ▶ strona startowa

```
root :to => "books#index"
```

- ▶ obsługa adresu spoza REST

```
match 'contact' => 'posts#show', :name => 'contact'
```

- ▶ segmenty opcjonalne w adresie

```
match ':controller(/:action(/:id))'
```

- ▶ ograniczenia dla segmentów

```
match 'posts/:name' => "posts#show", :constraints => { :name => /\w+/ }
```

- ▶ ograniczenia dla akcji

```
get 'posts/:name' => "posts#show"
```

# Plan prezentacji

Wprowadzenie

Trasowanie

Warstwa kontrolera

Obsługa stanu



# Generowanie kontrolera

▷ rails generate controller messages hello greeting

```
create  app/controllers/messages_controller.rb
route  get "messages/greeting"
route  get "messages/hello"
invoke erb
create  app/views/messages
create  app/views/messages/hello.html.erb
create  app/views/messages/greeting.html.erb
invoke test_unit
create  test/functional/messages_controller_test.rb
invoke helper
create  app/helpers/messages_helper.rb
...
```

# Przykładowy kontroler

```
app/controllers/messages_controller.rb
```

```
class MessagesController < ApplicationController
  def hello
    @message = "Witaj EPI!"
  end
  def greeting
    @message = "Witaj #{params[:name]}"
    render :action => "hello"
  end
end
```

# Przykładowy kontroler

## app/controllers/messages\_controller.rb

```
class MessagesController < ApplicationController
  def hello
    @message = "Witaj EPI!"
  end
  def greeting
    @message = "Witaj #{params[:name]}"
    render :action => "hello"
  end
end
```

## config/routes.rb

```
Simple::Application.routes.draw do
  get "messages/hello"
  post "messages/greeting"
end
```

# Przykładowy kontroler

## app/controllers/messages\_controller.rb

```
class MessagesController < ApplicationController
  def hello
    @message = "Witaj EPI!"
  end
  def greeting
    @message = "Witaj #{params[:name]}"
    render :action => "hello"
  end
end
```

## config/routes.rb

```
Simple::Application.routes.draw do
  get "messages/hello"
  post "messages/greeting"
end
```

## app/views/messages/hello.html.erb

```
<h1> Komunikat systemu </h1>
<%= @message %>
```

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu Controller

## Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu Controller
  - ▶ akcje są tożsame z wszystkimi *publicznymi* metodami tej klasy



## Zasady działania kontrolera

- ▶ dziedziczy z klasy `ApplicationController`
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu `Controller`
  - ▶ akcje są tożsame z wszystkimi *publicznymi* metodami tej klasy
- ▶ domyślnie każdej akcji odpowiada odrębny widok o tej samej nazwie co akcja

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu Controller
  - ▶ akcje są tożsame z wszystkimi *publicznymi* metodami tej klasy
- ▶ domyślnie każdej akcji odpowiada odrębny widok o tej samej nazwie co akcja
- ▶ widoki kontrolera znajdują się w katalogu `app/views/nazwa_kontrolera`

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu Controller
  - ▶ akcje są tożsame z wszystkimi *publicznymi* metodami tej klasy
- ▶ domyślnie każdej akcji odpowiada odrębny widok o tej samej nazwie co akcja
- ▶ widoki kontrolera znajdują się w katalogu **app/views/*nazwa\_kontrolera***
  - ▶ app/views/**messages**  
widoki kontrolera MessagesController

# Zasady działania kontrolera

- ▶ dziedziczy z klasy ApplicationController
- ▶ kontroler posiada **nazwę** oraz definiuje **akcje**
  - ▶ nazwa jest tożsama z nazwą klasy pisaną w notacji wielbłądziej i pozbawioną sufiksu Controller
  - ▶ akcje są tożsame z wszystkimi *publicznymi* metodami tej klasy
- ▶ domyślnie każdej akcji odpowiada odrębny widok o tej samej nazwie co akcja
- ▶ widoki kontrolera znajdują się w katalogu **app/views/*nazwa\_kontrolera***
  - ▶ app/views/**messages**  
widoki kontrolera MessagesController
  - ▶ app/views/messages/**hello.html.erb**  
widok odpowiadający akcji hello

## Przekazywanie danych do widoku

- ▶ dane pomiędzy kontrolerem a widokiem przekazywane są za pośrednictwem *zmiennych instancyjnych*

# Przekazywanie danych do widoku

- ▶ dane pomiędzy kontrolerem a widokiem przekazywane są za pośrednictwem *zmiennych instancyjnych*  
**akcja hello**

```
def hello  
  @message = "Witaj EPI!"  
end
```

# Przekazywanie danych do widoku

- ▶ dane pomiędzy kontrolerem a widokiem przekazywane są za pośrednictwem *zmiennych instancyjnych*  
**akcja hello**

```
def hello  
  @message = "Witaj EPI!"  
end
```

**widok hello**

```
<%= @message %>
```

# Przekazywanie danych do widoku

- ▶ dane pomiędzy kontrolerem a widokiem przekazywane są za pośrednictwem *zmiennych instancyjnych*

## akcja hello

```
def hello
  @message = "Witaj EPI!"
end
```

## widok hello

```
<%= @message %>
```

- ▶ akcja może wykorzystać inny widok niż domyślnie przypisany, o ile zainicjuje odpowiednie zmienne instancyjne



# Przekazywanie danych do widoku

- ▶ dane pomiędzy kontrolerem a widokiem przekazywane są za pośrednictwem *zmiennych instancyjnych*

## akcja hello

```
def hello
  @message = "Witaj EPI!"
end
```

## widok hello

```
<%= @message %>
```

- ▶ akcja może wykorzystać inny widok niż domyślnie przypisany, o ile zainicjuje odpowiednie zmienne instancyjne

## akcja greeting

```
def greeting
  @message = "Witaj #{params[:name]}"
  render :action => "hello"
end
```

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST formularz

```
<% form_tag :controller => "messages", :action => "greeting" do %>  
<%= text_field_tag :name %>  
<% end %>
```

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST formularz

```
<% form_tag :controller => "messages", :action => "greeting" do %>
  <%= text_field_tag :name %>
<% end %>
```

## akcja greeting

```
def greeting
  @message = "Witaj #{params[:name]}"
  render :action => "hello"
end
```

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST formularz

```
<% form_tag :controller => "messages", :action => "greeting" do %>
  <%= text_field_tag :name %>
<% end %>
```

## akcja greeting

```
def greeting
  @message = "Witaj #{params[:name]}"
  render :action => "hello"
end
```

- ▶ niezależnie od źródłowej postaci (query string, POST body, XML, JSON), dane dostępne są za pomocą zmiennej `params`

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST formularz

```
<% form_tag :controller => "messages", :action => "greeting" do %>
  <%= text_field_tag :name %>
<% end %>
```

## akcja greeting

```
def greeting
  @message = "Witaj #{params[:name]}"
  render :action => "hello"
end
```

- ▶ niezależnie od źródłowej postaci (query string, POST body, XML, JSON), dane dostępne są za pomocą zmiennej `params`
  - ▶ parametry są konwertowane do postaci tablicy asocjacyjnej

# Przekazywanie danych do kontrolera

- ▶ dane pomiędzy widokiem a kontrolerem przekazywane są pośrednictwem *parametrów żądania HTTP*: GET oraz POST formularz

```
<% form_tag :controller => "messages", :action => "greeting" do %>
  <%= text_field_tag :name %>
<% end %>
```

## akcja greeting

```
def greeting
  @message = "Witaj #{params[:name]}"
  render :action => "hello"
end
```

- ▶ niezależnie od źródłowej postaci (query string, POST body, XML, JSON), dane dostępne są za pomocą zmiennej `params`
  - ▶ parametry są konwertowane do postaci tablicy asocjacyjnej
  - ▶ kluczami tej tablicy są symbole, np. `:name`

# Zmiana przepływu sterowania

```
class MessagesController < ApplicationController
  def hello
    @message = "Witaj EPI!"
  end

  def greeting
    if params[:name].empty?
      redirect_to :action => "hello"
    else
      @message = "Witaj #{params[:name]}"
      render :action => "hello"
    end
  end
end
```



# Renderowanie vs. przekierowanie

renderowanie akcji:

# Renderowanie vs. przekierowanie

renderowanie akcji:

- ▶ przesłanie treści do użytkownika

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

## przekierowanie do akcji:

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

## przekierowanie do akcji:

- ▶ podmiana akcji

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

## przekierowanie do akcji:

- ▶ podmiana akcji
- ▶ wszystkie zmienne instancyjne oraz parametry żądania ustawione w pierwszej akcji są tracone po przekierowaniu

# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

## przekierowanie do akcji:

- ▶ podmiana akcji
- ▶ wszystkie zmienne instancyjne oraz parametry żądania ustawione w pierwszej akcji są tracone po przekierowaniu
- ▶ tylko zmienne instancyjne ustawione w drugiej akcji są dostępne



# Renderowanie vs. przekierowanie

## renderowanie akcji:

- ▶ przesłanie treści do użytkownika
- ▶ wszystkie zmienne instancyjne ustawione w pierwszej akcji są dostępne w widoku
- ▶ żadne dodatkowe zmienne instancyjne nie są ustawiane

## przekierowanie do akcji:

- ▶ podmiana akcji
- ▶ wszystkie zmienne instancyjne oraz parametry żądania ustawione w pierwszej akcji są tracone po przekierowaniu
- ▶ tylko zmienne instancyjne ustawione w drugiej akcji są dostępne
- ▶ zawartość obiektu `flash` nie ulega jednak zmianie

# DoubleRenderError

Wyjątek `DoubleRenderError` zostanie rzucony jeśli aplikacja opuszcza akcję, w której wielokrotnie zażądano renderowania.

# DoubleRenderError

Wyjątek `DoubleRenderError` zostanie rzucony jeśli aplikacja opuszcza akcję, w której wielokrotnie zażądano renderowania.

Przykład:

```
def greeting
  redirect_to :action => "hello"  #pierwsze renderowanie
  render :action => "hello"      #drugie renderowanie
end
```

# DoubleRenderError

Wyjątek `DoubleRenderError` zostanie rzucony jeśli aplikacja opuszcza akcję, w której wielokrotnie zażądano renderowania.

Przykład:

```
def greeting
  redirect_to :action => "hello"  #pierwsze renderowanie
  render :action => "hello"      #drugie renderowanie
end
```

Jeśli trzeba renderować odmienne widoki zależnie od jakiegoś warunku, należy użyć instrukcji warunkowej lub wyrażenia **and return** po `render/redirect`:

# DoubleRenderError

Wyjątek `DoubleRenderError` zostanie rzucony jeśli aplikacja opuszcza akcję, w której wielokrotnie zażądano renderowania.

Przykład:

```
def greeting
  redirect_to :action => "hello"  #pierwsze renderowanie
  render :action => "hello"      #drugie renderowanie
end
```

Jeśli trzeba renderować odmienne widoki zależnie od jakiegoś warunku, należy użyć instrukcji warunkowej lub wyrażenia **and return** po `render/redirect`:

```
def do_something
  if params[:name].empty?
    redirect_to(:action => "hello") and return
  end
  render :action => "hello"
end
```

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:



## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych
  - ▶ określenie dodatkowych nagłówek

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych
  - ▶ określenie dodatkowych nagłówek
- ▶ rodzaje filtrów:

## Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych
  - ▶ określenie dodatkowych nagłówek
- ▶ rodzaje filtrów:
  - ▶ `before_filter` – filtr wykonywany przed akcją główną

# Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych
  - ▶ określenie dodatkowych nagłówek
- ▶ rodzaje filtrów:
  - ▶ `before_filter` – filtr wykonywany przed akcją główną
  - ▶ `after_filter` – filtr wykonywany po akcji głównej

# Filtrowanie akcji

- ▶ pozwala na deklaratywne wywołanie metod, które mają się wykonać przed, pod lub przed i po akcji głównej
- ▶ mogą przerwać wykonywanie określonej akcji np. przekierowując do innej akcji
- ▶ zastosowania:
  - ▶ wymuszenie uwierzytelniania
  - ▶ weryfikacji dostępu do określonych akcji
  - ▶ kompresja danych wynikowych
  - ▶ określenie dodatkowych nagłówek
- ▶ rodzaje filtrów:
  - ▶ `before_filter` – filtr wykonywany przed akcją główną
  - ▶ `after_filter` – filtr wykonywany po akcji głównej
  - ▶ `around_filter` – filtr okalający – część wykonywana przed, a część po akcji głównej



# Przykład filtrowania akcji

```
class ApplicationController < ActionController::Base
  before_filter :require_login

  protected
  def require_login
    unless logged_in?
      flash[:error] = 'Trzeba być zalogowanym aby wykonać tę akcję!'
      redirect_to root_url
    end
  end

  def logged_in?
    !session[:user_id].nil?
  end
end
```

## Negocjacja treści

Kontroler może użyć różnej reprezentacji danych (HTML, XML, JSON, itp.), w zależności od wystosowanego żądania

```
app/controllers/books_controller.rb
```

# Negocjacja treści

Kontroler może użyć różnej reprezentacji danych (HTML, XML, JSON, itp.), w zależności od wystosowanego żądania

## app/controllers/books\_controller.rb

```
class BooksController < ApplicationController
  respond_to :html, :json
  # GET /books/1
  # GET /books/1.xml
  def show
    @book = Book.find(params[:id])
    respond_with @book
  end
end
```

# Negocjacja treści

Kontroler może użyć różnej reprezentacji danych (HTML, XML, JSON, itp.), w zależności od wystosowanego żądania

## app/controllers/books\_controller.rb

```
class BooksController < ApplicationController
  respond_to :html, :json
  # GET /books/1
  # GET /books/1.xml
  def show
    @book = Book.find(params[:id])
    respond_with @book
  end
end
```

Dla każdego formatu używany jest domyślnie widok o nazwie:

**nazwa\_akcji.nazwa\_formatu.nazwa\_renderera**

np. show.html.erb, show.xml.haml

## Negocjacja treści – przykład

▶ `/books/1`

## Negocjacja treści – przykład

▶ /books/1

```
<p>
```

```
  <b>Title:</b>
```

```
  Alef
```

```
</p>
```

```
<p>
```

```
  <b>Author:</b>
```

```
  Jorge Luis Borges
```

```
</p>
```

## Negocjacja treści – przykład

- ▶ /books/1

```
<p>
  <b>Title:</b>
  Alef
</p>
<p>
  <b>Author:</b>
  Jorge Luis Borges
</p>
```
- ▶ /books/1.xml lub /books/1 Accept:application/xml

## Negocjacja treści – przykład

▶ /books/1

```
<p>
  <b>Title:</b>
  Alef
</p>
<p>
  <b>Author:</b>
  Jorge Luis Borges
</p>
```

▶ /books/1.xml lub /books/1 Accept:application/xml

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <author-id type="integer">1</author-id>
  <id type="integer">1</id>
  <title>Alef</title>
</book>
```



# Plan prezentacji

Wprowadzenie

Trasowanie

Warstwa kontrolera

Obsługa stanu

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu  
np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)



# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)
- ▶ metody dostępne w kontrolerze do obsługi stanu:

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)
- ▶ metody dostępne w kontrolerze do obsługi stanu:
  - ▶ `session`

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)
- ▶ metody dostępne w kontrolerze do obsługi stanu:
  - ▶ session
  - ▶ cookies

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)
- ▶ metody dostępne w kontrolerze do obsługi stanu:
  - ▶ session
  - ▶ cookies
  - ▶ flash

# Obsługa stanu

- ▶ protokół HTTP jest bezstanowy
- ▶ potrzebny jest dodatkowy mechanizm do obsługi stanu np. do przechowywania informacji o zalogowanym użytkowniku, zawartości jego koszyka, itp.
- ▶ możliwe sposoby obsługi stanu
  - ▶ zagnieżdżanie identyfikatora sesji w URL-u
  - ▶ zagnieżdżanie w niewidocznych polach formularza
  - ▶ użycie ciasteczek (cookies)
- ▶ metody dostępne w kontrolerze do obsługi stanu:
  - ▶ session
  - ▶ cookies
  - ▶ flash
- ▶ powyższe metody posiadają strukturę tablicy asocjacyjnej

# Sesja – session

```
def login
  user = User.authenticate(params[:login],params[:password])
  session[:user_id] = user.id unless user.nil?
end
def find_user
  @user = User.find(session[:user_id])
end
```

- ▶ przechowuje dane pomiędzy kolejnymi żądaniami

# Sesja – session

```
def login
  user = User.authenticate(params[:login],params[:password])
  session[:user_id] = user.id unless user.nil?
end
def find_user
  @user = User.find(session[:user_id])
end
```

- ▶ przechowuje dane pomiędzy kolejnymi żądaniami
- ▶ może przechowywać obiekt dowolnego typu

# Sesja – session

```
def login
  user = User.authenticate(params[:login],params[:password])
  session[:user_id] = user.id unless user.nil?
end
def find_user
  @user = User.find(session[:user_id])
end
```

- ▶ przechowuje dane pomiędzy kolejnymi żądaniami
- ▶ może przechowywać obiekt dowolnego typu
- ▶ identyfikator sesji jest przechowywany w cookie i wysyłany przy każdym żądaniu



## Sesja – session

```
def login
  user = User.authenticate(params[:login],params[:password])
  session[:user_id] = user.id unless user.nil?
end
def find_user
  @user = User.find(session[:user_id])
end
```

- ▶ przechowuje dane pomiędzy kolejnymi żądaniami
- ▶ może przechowywać obiekt dowolnego typu
- ▶ identyfikator sesji jest przechowywany w cookie i wysyłany przy każdym żądaniu
- ▶ różnorodne opcje przechowywania danych sesji (plik, baza danych, cookie)

# Sesja – session

```
def login
  user = User.authenticate(params[:login],params[:password])
  session[:user_id] = user.id unless user.nil?
end
def find_user
  @user = User.find(session[:user_id])
end
```

- ▶ przechowuje dane pomiędzy kolejnymi żądaniami
- ▶ może przechowywać obiekt dowolnego typu
- ▶ identyfikator sesji jest przechowywany w cookie i wysyłany przy każdym żądaniu
- ▶ różnorodne opcje przechowywania danych sesji (plik, baza danych, cookie)
- ▶ domyślnie – zaszyfrowane w cookie (nie trzeba usuwać „umarłych” sesji)

# Ciasteczka – cookies

```
def remember_me
  cookies[:auth_token] = {
    :value => @session[:user].remember_token ,
    :expires => @session[:user].remember_token_expires
  }
end

def find_user
  @user = User.find_by_remember_token(cookies[:auth_token])
end
```

- ▶ przechowywane w przeglądarce użytkownika

# Ciasteczka – cookies

```
def remember_me
  cookies[:auth_token] = {
    :value => @session[:user].remember_token ,
    :expires => @session[:user].remember_token_expires
  }
end

def find_user
  @user = User.find_by_remember_token(cookies[:auth_token])
end
```

- ▶ przechowywane w przeglądarce użytkownika
- ▶ przesyłane z każdym żądaniem

## Ciasteczka – cookies

```
def remember_me
  cookies[:auth_token] = {
    :value => @session[:user].remember_token ,
    :expires => @session[:user].remember_token_expires
  }
end

def find_user
  @user = User.find_by_remember_token(cookies[:auth_token])
end
```

- ▶ przechowywane w przeglądarce użytkownika
- ▶ przesyłane z każdym żądaniem
- ▶ mogą przechowywać wyłącznie łańcuchy znaków

## Komunikaty jednorazowe – flash

```
flash[:notice] = 'Logowanie do systemu przebiegło pomyślnie'  
flash[:error] = 'Logowanie nie powiodło się!'
```

- ▶ pozwala na przekazywane tymczasowych obiektów pomiędzy akcjami

## Komunikaty jednorazowe – flash

```
flash[:notice] = 'Logowanie do systemu przebiegło pomyślnie'  
flash[:error] = 'Logowanie nie powiodło się!'
```

- ▶ pozwala na przekazywane tymczasowych obiektów pomiędzy akcjami
- ▶ wszystko co zostanie umieszczone w flashu, zostanie udostępnione następnej akcji a później wyczyszczone

## Komunikaty jednorazowe – flash

```
flash[:notice] = 'Logowanie do systemu przebiegło pomyślnie'  
flash[:error] = 'Logowanie nie powiodło się!'
```

- ▶ pozwala na przekazywane tymczasowych obiektów pomiędzy akcjami
- ▶ wszystko co zostanie umieszczone w flashu, zostanie udostępnione następnej akcji a później wyczyszczone
- ▶ najczęściej wykorzystywany do ustawiania komunikatów w akcjach, po których następuje przekierowanie do innej akcji



# Przykład wykorzystania komunikatu

## Kontroler

```
class PostsController < ApplicationController
  def create
    post = Post.new(params[:post])
    if post.save
      flash[:notice] = "Post został utworzony"
      redirect_to :action => "show", :params => {:id => post.id}
    end
  end

  def show
    @post = Post.find(params[:id])
  end
end
```

# Przykład wykorzystania komunikatu

## Kontroler

```
class PostsController < ApplicationController
  def create
    post = Post.new(params[:post])
    if post.save
      flash[:notice] = "Post został utworzony"
      redirect_to :action => "show", :params => {:id => post.id}
    end
  end

  def show
    @post = Post.find(params[:id])
  end
end
```

## Widok

```
<% if flash[:notice] %>
  <div class="notice"><%= flash[:notice] %></div>
<% end %>
```



# Materiały

- ▶ [guides.rubyonrails.org/index.html](https://guides.rubyonrails.org/index.html)  
**podstawowe informacje o warstwie kontrolera**
- ▶ [guides.rubyonrails.org/routing.html](https://guides.rubyonrails.org/routing.html)  
**trasowanie**
- ▶ [guides.rubyonrails.org/action\\_controller\\_overview.html](https://guides.rubyonrails.org/action_controller_overview.html)  
**warstwa kontrolera**
- ▶ [github.com/sferik/rails\\_admin](https://github.com/sferik/rails_admin)  
**Rails admin**
- ▶ [github.com/activescaffold/active\\_scaffold](https://github.com/activescaffold/active_scaffold)  
**ActiveScaffold**

# Pytania

PYTANIA?