

Programowanie w Ruby

Wykład 12

Marcin Młotkowski

9 stycznia 2013

Plan wykładu

Żądania protokołu HTTP

Budowa serwisu

- Przeglądanie elementów modelu

- Usuwanie elementów

- Nowy element

- Aktualizacja danych

- Funkcje pomocnicze

Sesje

Standardowa postać żądania

<metoda> <url> HTTP/1.1

...

...

Standardowa postać żądania

<metoda> <url> HTTP/1.1

...

...

Rodzaje metod:

- GET – pobranie dokumentu
- POST – wysłanie danych do serwera
- PUT – umieszczenie na serwerze
- DELETE – usunięcie zasobu z serwera

Mapowanie żądań

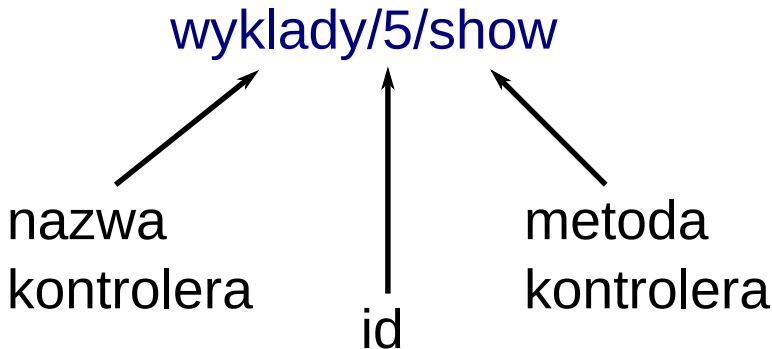
Mapowanie określa, które metody są odpowiedzialne za obsługę odpowiednich żądań.

Mapowanie żądań

Mapowanie określa, które metody są odpowiedzialne za obsługę odpowiednich żądań.

- ▶ Mapowania są opisane w pliku `config.routes`
- ▶ `GET /model` oznacza wywołanie metody `kontroler.index`
- ▶ `GET /model/:id` oznacza `kontroler.view(:id)`

Przykład mapowania



Plan wykładu

Żądania protokołu HTTP

Budowa serwisu

- Przeglądanie elementów modelu

- Usuwanie elementów

- Nowy element

- Aktualizacja danych

- Funkcje pomocnicze

Sesje

Kilka założeń o serwisie Zapisy

- ▶ Główną stroną jest lista wykładów
- ▶ Przy każdym elemencie są odnośniki do edycji, usunięcia i przeglądania

Wykłady

- ▶ Model `Wyklad` w pliku `app/models/wyklad.rb`
- ▶ Kontroler `WykladsController` w pliku `app/controllers/wyklads_controller.rb`
 - ▶ `index`
 - ▶ `edit`
 - ▶ `new`
 - ▶ `show`

Widoki

- ▶ index.html.erb (lub index.rhtml)
- ▶ edit.html.erb (lub edit.rhtml)
- ▶ new.html.erb (lub new.rhtml)
- ▶ show.html.erb (lub show.rhtml)

Index: kontroler

```
def index  
  @wyklads = Wyklad.find(:all)  
end
```

Index: widok

```
<% for wyklad in @wyklads %>
<%=h wyklad.tytul %>
<%=h wyklad.opis %>
<%= link_to 'Show', wyklad %>
<%= link_to 'Edit', edit_wyklad_path(wyklad) %>
<%= link_to 'Destroy', wyklad,
      :confirm => 'Are you sure?', :method => :delete %>
<% end %>
```

Index: widok

```
<% for wyklad in @wyklads %>
<%=h wyklad.tytul %>
<%=h wyklad.opis %>
<%= link_to 'Show', wyklad %>
<%= link_to 'Edit', edit_wyklad_path(wyklad) %>
<%= link_to 'Destroy', wyklad,
      :confirm => 'Are you sure?', :method => :delete %>
<% end %>
```

Show

Kontroler

```
def show  
  @wyklad = Wyklad.find(params[:id])  
end
```

Show

Kontroler

```
def show  
  @wyklad = Wyklad.find(params[:id])  
end
```

Widok

```
<%=h @wyklad.tytul %>  
<%=h @wyklad.opis %>  
<%=h @wyklad.punkty %>
```

Usuwanie

Nawigacja

```
<%= link_to 'Usunięcie', wyklad,  
  :confirm => 'Jesteś pewien?',  
  :method => :delete %>
```

Odnosnik na stronie

```
<a href="/wyklads/1" onclick=".....">
```

Kontroler

Kontroler

```
def destroy
  @wyklad = Wyklad.find(params[:id])
  @wyklad.destroy
  redirect_to('wyklads')
end
```

Kontroler

Kontroler

```
def destroy
  @wyklad = Wyklad.find(params[:id])
  @wyklad.destroy
  redirect_to('wyklads')
end
```

Widok

```
brak
```

Nowy element

```
<% for wyklad in @wyklads %>
```

```
<%=h wyklad.tytul %>
```

```
<%=h wyklad.opis %>
```

```
<% end %>
```

```
<%= link_to 'New wyklad', new_wyklad_path %>
```

```
<a href="/wyklads/new">Nowy wykład</a>
```

New: kontroler

```
def new  
  @wyklad = Wyklad.new  
end
```

Widok

- ▶ W widoku musi być edycja
- ▶ Po zatwierdzeniu edycji wysyłana jest metoda POST z url wykłads i danymi
- ▶ Żądanie **POST wykłads** obsługuje metoda kontrolera **create**

Edycja danych

W html'u robimy to tak:

```
<form action="akcja" method="post" >
<input id="wyklad_tytul" name="wyklad[tytul]"
      size="30" type="text" value="Kurs Ruby" />
<input id="wyklad_punkty"
      name="wyklad[punkty]" size="30" type="text"
      value="12" />
</form>
```

Edycja danych

A w Rails'ach robimy to tak:

```
<% form_for(@wyklad) do | f | %>
<%= f.text_field :tytul %>
<%= f.text_area :opis %>
<%= f.text_field :punkty %>
<%= f.submit "Update" %>
<% end %>
```

Edycja danych

A w Rails'ach robimy to tak:

```
<% form_for(@wyklad) do | f | %>
<%= f.text_field :tytul %>
<%= f.text_area :opis %>
<%= f.text_field :punkty %>
<%= f.submit "Update" %>
<% end %>
```

Zadanie `form_for` <model> &blok:

- ▶ Funkcja pomocnicza do tworzenia formularza do wczytywania danych
- ▶ Uzupełnia formularz o już wcześniej wpisane wartości

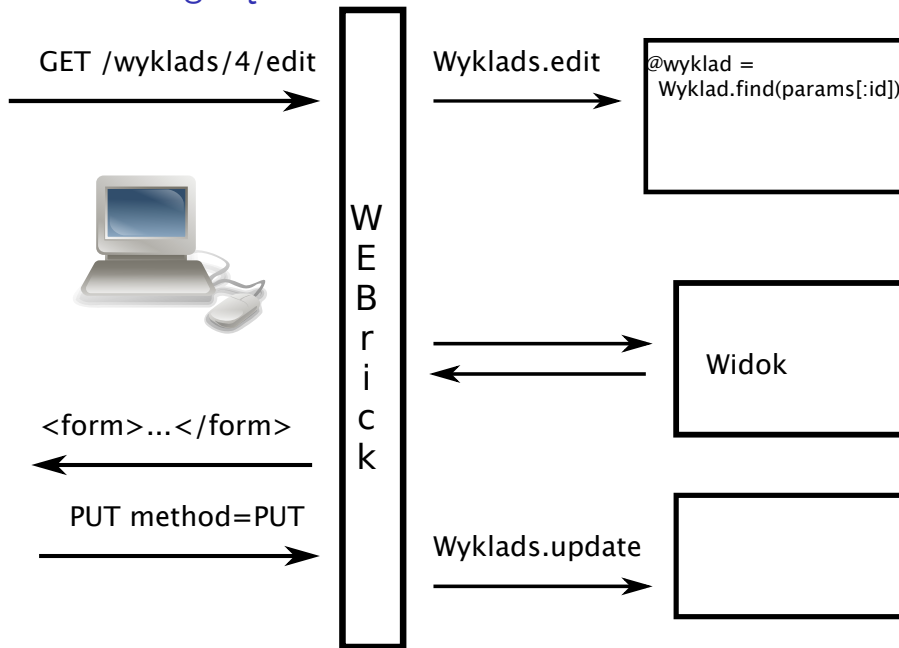
Zapis stanu

- ▶ Formularze są przesyłane za pomocą metody **POST**
- ▶ Wywoływana jest metoda **create**

Metoda create

```
def create
  @wyklad = Wyklad.new(params[:wyklad])
  if @wyklad.save
    flash[:notice] = 'utworzone'
    redirect_to(@wyklad)
  else
    format.render :action => "new"
  end
end
```

Schemat obsługi żądań



Edycja elementu

- ▶ Edycja wywoływana jest żądaniem
`<a href="/wyklads/4/edit" >`
- ▶ Widok wygląda (prawie) identycznie jak widok dla nowego elementu
- ▶ Ciało kontrolera
`@wyklad = Wyklad.find(params[:id])`
- ▶ Po edycji generowane jest żądanie `PUT /wyklads/4` obsługiwane przez metodę kontrolera `update`

Metoda update

```
@wyklad = Wyklad.find(params[:id])  
if @wyklad.update_attributes(params[:wyklad])  
  flash[:notice] = 'Wykład zaktualizowany.'  
  redirect_to(@wyklad)  
else  
  render :action => "edit"  
end
```

Funkcja pomocnicza: link_to

Składnia

```
link_to(opis, opcje={ }, opcje_html={ })
```

opcje_html

```
{ :id => "link-to", :class => "szary" }
```

Opis

Opis linka

Opcje link_to

- ▶ :controller => 'Wyklad'
- ▶ :action => 'show'
- ▶ :confirm => 'Na pewno?'
- ▶ :method => <żądanie HTTP>

Definiowanie linków (1)

- ▶ `action => 'akcja'`
- ▶ `:controller => 'kontroler', :action => 'akcja'`
- ▶ `:controller => 'kontroler'`

Definiowanie linków (2)

```
$ rake routes
```

```
...
```

```
new_wyklady GET /wykladies/new
```

```
  {:controller => "wykladies", :action => "new" }
```

```
edit_wyklady GET /wykladies/:id/edit
```

```
  {:controller => "wykladies", :action => "edit" }
```

```
...
```

```
new_wyklady_path
```

```
edit_wyklady_path(obj)
```

Zastosowanie

link_to "Nowy element", new_wyklady_path

link_to "Edycja", edit_wyklady_path(obj)

Inne wersje link_to

`link_to_if(warunek, nazwa, opcje, opcje_html)`

gdy warunek jest nieprawdziwy, to zwracana jest `nazwa`

`link_to_unless(warunek, nazwa, opcje, opcje_html)`

mail_to

Składnia

mail_to <adres>, <nazwa>, <opcje>

Opcje

- ▶ :encode => 'javascript'
- ▶ :subject => 'temat maila'
- ▶ :cc => kopia do
- ▶ :body => <treść maila>

button_to

Składnia

```
button_to(nazwa, opcje=, opcje_html=)
```

Opis

Definiuje formularz z przyciskiem o nazwie `<nazwa>` o parametrach:

- ▶ `:action =>` `<akcja>`
- ▶ `:confirm =>` `'Na pewno?'`

Metody pomocnicze

`redirect_to :akcja`

przekierowanie do akcji (metody kontrolera i widoku)

`render :action => 'akcja'`

wyświetlenie widoku związanego z akcją

Plan wykładu

Żądania protokołu HTTP

Budowa serwisu

- Przeglądanie elementów modelu

- Usuwanie elementów

- Nowy element

- Aktualizacja danych

- Funkcje pomocnicze

Sesje

Problem

Protokół http jest bezstanowy.

Problem

Protokół http jest bezstanowy.

Rozwiązanie: SID (identyfikator sesji)

`http://.../akcja/argument?SID=453454564`

Ruby on Rails

ciasteczka

Zastosowanie sesji

- ▶ Personalizowanie stron, np. tworzenie koszyków z zakupami
- ▶ Tworzenie własnego programu studiów

Obsługa sesji w RoR

RoR definiuje tablicę asocjacyjną [session](#).

Dla każdej sesji (rozdzielanej przez ciasteczko) istnieje inna tablica asocjacyjna

Przykład: plan zajęć

- ▶ Zdefiniujemy klasę Zajecia przechowującą informacje o wybranych przez użytkownika (studenta) przedmiotach
- ▶ Uzupełnimy akcje o dodawanie i usuwanie przedmiotów

Implementacja klasy Wyklad

- ▶ Trzeba utworzyć koszyk z wykładami, tj. klasę Zajecia
- ▶ W ogólnym widoku (liście przedmiotów) trzeba dodać przycisk/link dodania 'do koszyka'
- ▶ Uzupełnić kontroler Wyklad o obsługę tej akcji
- ▶ Akcja niech się nazywa dodaj

Dodanie linku do akcji

views/wyklads/index.rhtml

```
<%=h wyklad.opis %>
```

```
<%=h wyklad.punkty %>
```

```
<%= link_to 'Dodaj wyklad',  
           { :action => 'dodaj', :id => wyklad } %>
```

Dodanie linku do akcji

views/wyklads/index.rhtml

```
<%=h wyklad.opis %>
```

```
<%=h wyklad.punkty %>
```

```
<%= link_to 'Dodaj wyklad',  
           { :action => 'dodaj', :id => wyklad } %>
```

Obsługa akcji w kontrolerze

```
def dodaj  
  session[:zajecia] ||= Zajecia.new  
  wyklad = Wyklad.find(params[:id])  
  session[:zajecia].add(wyklad)  
  redirect_to :action => 'index'  
end
```

Dygresja

Definicja operatora `||=`:

```
session[:zajecia] ||= Zajecia.new
```

jest równoważne

```
unless session[:zajecia]
  session[:zajecia] = Zajecia.new
end
session[:zajecia]
```

Dalsza implementacja

Obsługa (w formie odrębnego kontrolera) wybranych wykładów:

- ▶ Przeglądanie
- ▶ Usuwanie
- ▶ Trwały zapis (wymaga systemu użytkowników)