

# EPI: Interfejs Graficzny

## Wykład nr 8

### Warstwa widoku

dr inż. Aleksander Smywiński-Pohl

Elektroniczne Przetwarzanie Informacji  
Konsultacje: czw. 14.00-15.30, pokój 3.211

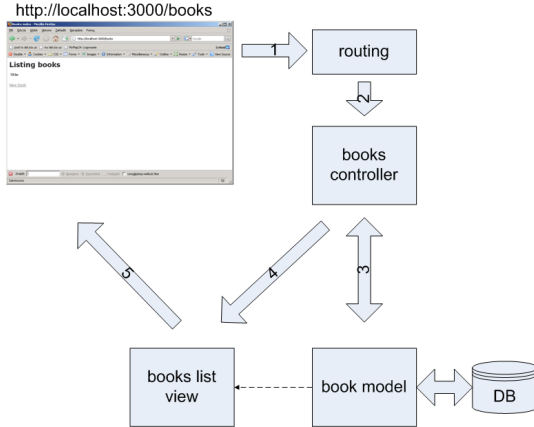
# Plan prezentacji

Wprowadzenie

Helpery

Fragmenty widoków

# MVC w RoR



## erb

```
<%= some_statement %>
```

ewaluuje wyrażenie, zamienia je na łańcuch znaków i wyświetla w miejscu wystąpienia

# erb

```
<%= some_statement %>
```

ewaluuje wyrażenie, zamienia je na łańcuch znaków i wyświetla w miejscu wystąpienia

```
<% some_statement %>
```

ewaluuje wyrażanie, ale nie wyświetla wyniku

## erb

```
<%= some_statement %>
```

ewaluuje wyrażenie, zamienia je na łańcuch znaków i wyświetla w miejscu wystąpienia

```
<% some_statement %>
```

ewaluuje wyrażenie, ale nie wyświetla wyniku

Przykład:

```
<b>Names of all the people</b>  
<% for person in @people %>  
  Name: <%= person.name %><br/>  
<% end %>
```

# haml

**haml** jest alternatywą dla erb. Pozwala na tworzenie znacznie bardziej zwięzłego kodu:

```
#people
  %b Names of all the people
  - for person in @people
    .name
      Name:
      = person.name
```

# haml

haml jest alternatywą dla erb. Pozwala na tworzenie znacznie bardziej zwięzłego kodu:

```
#people
  %b Names of all the people
  - for person in @people
    .name
      Name:
      = person.name
```

jest zamieniane na:

```
<div id="people">
  <b>Names of all the people</b>
  <div class="name">
    Name: John Brown
  <div>
  <div class="name">
    Name: Ann Black
  <div>
</div>
```



# Układ strony (layout)

app/views/layouts/library.html.erb:

```
<html>
<head>
  <title>Library</title>
  <%= stylesheet_link_tag "scaffold" %>
</head>
<body>
  <%= link_to "Books", books_path %>
  <%= link_to "Authors", authors_path %>
  <p style="color: green"><%= flash[:notice] %></p>

  <%= yield %>

</body>
</html>
```

# Układ strony (layout)

app/views/layouts/library.html.erb:

```
<html>
<head>
  <title>Library</title>
  <%= stylesheet_link_tag "scaffold" %>
</head>
<body>
  <%= link_to "Books", books_path %>
  <%= link_to "Authors", authors_path %>
  <p style="color: green"><%= flash[:notice] %></p>

  <%= yield %>

</body>
</html>

class AuthorsController < ApplicationController
  layout "library"
  ...
end
```

# Plan prezentacji

Wprowadzenie

Helpery

Fragmenty widoków

# Helpery

- ▶ pozwalają na zminimalizowanie ilości kodu Rubiego w widoku

# Helpery

- ▶ pozwalają na zminimalizowanie ilości kodu Rubiego w widoku
- ▶ są modułami (a nie klasami) Rubiego

# Helpery

- ▶ pozwalają na zminimalizowanie ilości kodu Rubiego w widoku
- ▶ są modułami (a nie klasami) Rubiego
- ▶ Railsy posiadają wiele helperów wbudowanych (np. do tworzenia linków, załączania obrazów, etc.)

# Helpers

- ▶ pozwalają na zminimalizowanie ilości kodu Rubiego w widoku
- ▶ są modułami (a nie klasami) Rubiego
- ▶ Railsy posiadają wiele helperów wbudowanych (np. do tworzenia linków, załączania obrazów, etc.)
- ▶ użytkownik może definiować własne helpery w katalogu `app/helpers`

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```



# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML
- ▶ `form_for` – szkielet formularza

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML
- ▶ `form_for` – szkielet formularza
- ▶ `text_field`, `check_box`, `label`, `password_field`, `radio_button`, `text_area`, `select` – pola formularza

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML
- ▶ `form_for` – szkielet formularza
- ▶ `text_field`, `check_box`, `label`, `password_field`, `radio_button`, `text_area`, `select` – pola formularza
- ▶ `auto_link` – automatycznie tworzy linki do adresów wykrytych w tekście

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML
- ▶ `form_for` – szkielet formularza
- ▶ `text_field`, `check_box`, `label`, `password_field`, `radio_button`, `text_area`, `select` – pola formularza
- ▶ `auto_link` – automatycznie tworzy linki do adresów wykrytych w tekście
- ▶ `simple_format` – proste formatowanie tekstu

# Wbudowane helpery Rails

Wbudowane helpery Rails, np. `link_to`:

```
<%= link_to 'Back', books_path %>
```

Wynik:

```
<a href="/books/list">Back</a>
```

Inne:

- ▶ `h` – escapowanie tagów HTML (obecnie domyślnie włączony)
- ▶ `raw` – brak escapowania tagów HTML
- ▶ `form_for` – szkielet formularza
- ▶ `text_field`, `check_box`, `label`, `password_field`, `radio_button`, `text_area`, `select` – pola formularza
- ▶ `auto_link` – automatycznie tworzy linki do adresów wykrytych w tekście
- ▶ `simple_format` – proste formatowanie tekstu
- ▶ `ActionView::Helpers` – pozostałe helpery

# Formularze

- ▶ helper `form_for` pozwala w łatwy sposób tworzyć formularz dla określonego obiektu



# Formularze

- ▶ helper `form_for` pozwala w łatwy sposób tworzyć formularz dla określonego obiektu
- ▶ ma on postać bloku z jednym argumentem

# Formularze

- ▶ helper `form_for` pozwala w łatwy sposób tworzyć formularz dla określonego obiektu
- ▶ ma on postać bloku z jednym argumentem
- ▶ argument bloku pozwala na tworzenie pól formularza powiązanych z odpowiednimi atrybutami obiektu

```
<%= form_for @book do |f| %>
  <%= f.text_field :title %>
  <%= f.select :author_id, @authors %>
  <%= f.submit "Create" %>
<% end %>
```

# Formularze – przykład

```
<%= form_for @book do |f| %>
  <%= f.text_field :title %>
  <%= f.select :author_id, @authors %>
  <%= f.submit "Create" %>
<% end %>
```

# Formularze – przykład

```
<%= form_for @book do |f| %>
  <%= f.text_field :title %>
  <%= f.select :author_id, @authors %>
  <%= f.submit "Create" %>
<% end %>

<form action="/books/create" method="post">
  <input name="book[title]" size="30" type="text"/>
  <select name="book[author_id]">
    <option value="1">Alan Alexander Milne</option><!-- ... -->
  </select>
  <input name="commit" type="submit" value="Create"/>
</form>
```

## Formularze – przykład

```

<%= form_for @book do |f| %>
  <%= f.text_field :title %>
  <%= f.select :author_id, @authors %>
  <%= f.submit "Create" %>
<% end %>

<form action="/books/create" method="post">
  <input name="book[title]" size="30" type="text"/>
  <select name="book[author_id]">
    <option value="1">Alan Alexander Milne</option><!-- ... -->
  </select>
  <input name="commit" type="submit" value="Create"/>
</form>

{
  "utf8"=>"[tick]",
  "authenticity_token"=>"lV/YiZzgseVuXjsoVW5JX8lyk4egMSpz2IXTJ5fHhgE=",
  "book"=>{"title"=>"Harry Potter", "author_id"=>"2"},
  "commit"=>"Create Book", "action"=>"create", "controller"=>"books"
}
params[:book][:title]

```

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:



# Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:

```
class Book < ActiveRecord::Base
  belongs_to :author
  accepts_nested_attributes_for :author, reject_if: :all_blank
end
```

# Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:

```
class Book < ActiveRecord::Base
  belongs_to :author
  accepts_nested_attributes_for :author, reject_if: :all_blank
end
```

- ▶ w formularzu korzystamy z helpera `fields_for`

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:

```
class Book < ActiveRecord::Base
  belongs_to :author
  accepts_nested_attributes_for :author, reject_if: :all_blank
end
```

- ▶ w formularzu korzystamy z helpera **fields\_for**
- ▶ jego argumentem może być powiązany obiekt (np. `@book.author`) lub symbol (np. `:author`)

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:

```
class Book < ActiveRecord::Base
  belongs_to :author
  accepts_nested_attributes_for :author, reject_if: :all_blank
end
```

- ▶ w formularzu korzystamy z helpera **fields\_for**
- ▶ jego argumentem może być powiązany obiekt (np. `@book.author`) lub symbol (np. `:author`)
  - ▶ w przypadku obiektu – musi on istnieć

## Formularze dla powiązanych obiektów

- ▶ możliwe jest łatwe tworzenie formularzy dla obiektów powiązanych relacjami jeden-do-jeden lub wiele-do-wiele
- ▶ np. jeśli chcemy jednocześnie utworzyć książkę i jej autora
- ▶ książka musi akceptować zagnieżdżone parametry dla autora:

```
class Book < ActiveRecord::Base
  belongs_to :author
  accepts_nested_attributes_for :author, reject_if: :all_blank
end
```

- ▶ w formularzu korzystamy z helpera **fields\_for**
- ▶ jego argumentem może być powiązany obiekt (np. `@book.author`) lub symbol (np. `:author`)
  - ▶ w przypadku obiektu – musi on istnieć
  - ▶ w przypadku symbolu – zostanie utworzony nowy obiekt, jeśli powiązany obiekt nie istnieje

## Formularze dla powiązanych obiektów – cd.

zmodyfikowany formularz pozwala na wybranie istniejącego lub utworzenie nowego autora

```
<%= form_for(@book) do |f| %>
  <!-- pola książki ... -->
  <div>
    <%= f.fields_for :author do |author_f| %>
      <p>
        <%= author_f.label :first_name %>
        <%= author_f.text_field :first_name %>
      </p>
      <p>
        <%= author_f.label :last_name %>
        <%= author_f.text_field :last_name %>
      </p>
    <% end %>
  </div>
  <!-- submit ... -->
<% end %>
```

# Helpery użytkownika

- ▶ dla widoków określonego kontrolera, ładowane są następujące helpery użytkownika:

# Helpery użytkownika

- ▶ dla widoków określonego kontrolera, ładowane są następujące helpery użytkownika:
  - ▶ ApplicationHelper **app/helpers/application\_helper.rb** zawierający metody pomocnicze wykorzystywane we wszystkich kontrolerach



# Helpery użytkownika

- ▶ dla widoków określonego kontrolera, ładowane są następujące helpery użytkownika:
  - ▶ ApplicationHelper **app/helpers/application\_helper.rb** zawierający metody pomocnicze wykorzystywane we wszystkich kontrolerach
  - ▶ NazwaKontroleraHelper np. dla  
app/controllers/**authors\_controller.rb**  
app/helpers/**authors\_helper.rb** zawierający metody pomocnicze wykorzystywane tylko w widokach tego kontrolera

# Helpers użytkownika

- ▶ dla widoków określonego kontrolera, ładowane są następujące helpers użytkownika:
  - ▶ ApplicationHelper **app/helpers/application\_helper.rb** zawierający metody pomocnicze wykorzystywane we wszystkich kontrolerach
  - ▶ NazwaKontroleraHelper np. dla  
app/controllers/**authors\_controller.rb**  
app/helpers/**authors\_helper.rb** zawierający metody pomocnicze wykorzystywane tylko w widokach tego kontrolera
- ▶ dodatkowe helpers można aktywować w kontrolerze:

```
class AuthorsController < ApplicationController
  helper BooksHelper
end
```

# Prosty przykład helpera

Definicja w `helpers/books_helper.rb`:

```
module BooksHelper
  def book_fancy_title(book)
    raw("<b>***#{h(book.title)}***</b>")
  end
end
```

# Prosty przykład helpera

Definicja w `helpers/books_helper.rb`:

```
module BooksHelper
  def book_fancy_title(book)
    raw("<b>***#{h(book.title)}***</b>")
  end
end
```

Użycie w `views/books/show.html.erb`:

```
<%= book_fancy_title(@book) %>
```

# Prosty przykład helpera

Definicja w `helpers/books_helper.rb`:

```
module BooksHelper
  def book_fancy_title(book)
    raw("<b>***#{h(book.title)}***</b>")
  end
end
```

Użycie w `views/books/show.html.erb`:

```
<%= book_fancy_title(@book) %>
```

Wynik:

```
<b>***Mały Książę***</b>
```

# Rozbudowany przykład helpera

Definicja w `app/helpers/application_helper.rb`:

```
module ApplicationHelper
  def close()
    raw link_to(image_tag("cancel.png", title: "zamknij",
                        class: "close_button"), "#")
  end

  def panel(&block)
    content_tag(:div,
      content_tag(:div, class: "content",
                  style: "width:95%", &block) +
      content_tag(:div, close(), class: "actions") +
      content_tag(:div, tag(:span), class: "clear") +
      '</div>'.html_safe, class: "relations")
  end
end
```

# Rozbudowany przykład helpera

Użycie:

```
<%= panel do %>  
  <p> Jakiś tekst </p>  
<% end %>
```

# Rozbudowany przykład helpera

Użycie:

```
<%= panel do %>
  <p> Jakiś tekst </p>
<% end %>
```

Wynik:

```
<div class="relations"><div class="content" style="width:95%">
  <p> Jakiś tekst </p>
</div>
<div class="actions">
  <a href="#">
    
  </a>
</div>
<div class="clear"><span /></div></div></div>
```



# Rozbudowany przykład helpera – efekt

[symbole](#) | [przykłady](#) | [leksemy](#) | [nowy leksem](#) | [grupy przykładów](#) | [wzorce](#) | [umbel](#) | [użytkownicy](#) | [szukaj](#) | [wyloguj](#)

Szukaj Resetuj ✕

CLP  SSJP (n)  Cyc (name)  Cyc (strict)  bigramy  typy  definicje  SPARQL  en-pl  pl-en

Cyc symbol (@localhost)  
'dog' was not found in cyc

✕

# Plan prezentacji

Wprowadzenie

Helpery

Fragmenty widoków

## Fragmenty widoków (partials)

- ▶ ich nazwa rozpoczyna się od znaku podkreślenia

## Fragmenty widoków (partials)

- ▶ ich nazwa rozpoczyna się od znaku podkreślenia
- ▶ pozwalają na reużytkowanie fragmentów widoku w wielu widokach.

# Fragmenty widoków (partials)

- ▶ ich nazwa rozpoczyna się od znaku podkreślenia
- ▶ pozwalają na reużytkowanie fragmentów widoku w wielu widokach.

## new.html.erb

```
<h1>New book</h1>
<% form_for(@book) do |f| %>
  <%= render partial: 'form',
    object: f %>
<p>
  <%= f.submit "Create" %>
</p>
<% end %>
<%= link_to 'Back', books_path %>
```

# Fragmenty widoków (partials)

- ▶ ich nazwa rozpoczyna się od znaku podkreślenia
- ▶ pozwalają na reużytkowanie fragmentów widoku w wielu widokach.

## new.html.erb

```
<h1>New book</h1>
<% form_for(@book) do |f| %>
  <%= render partial: 'form',
    object: f %>
<p>
  <%= f.submit "Create" %>
</p>
<% end %>
<%= link_to 'Back', books_path %>
```

## edit.html.erb

```
<h1>Editing book</h1>
<% form_for(@book) do |f| %>
  <%= render partial: 'form',
    object: f %>
<p>
  <%= f.submit "Update" %>
</p>
<% end %>
<%= link_to 'Show', @book %> |
<%= link_to 'Back', books_path %>
```

# Fragmenty widoków cd.

## ▶ \_form.html.erb

```
<p>
  <%= form.label :title %><br />
  <%= form.text_field :title %>
</p>
<p>
  <%= form.label :author %><br />
  <%= form.select :author_id, @authors %>
</p>
```

## Fragmenty widoków cd.

- ▶ `_form.html.erb`

```
<p>
  <%= form.label :title %><br />
  <%= form.text_field :title %>
</p>
<p>
  <%= form.label :author %><br />
  <%= form.select :author_id, @authors %>
</p>
```

- ▶ domyślnie fragment jest szukany w katalogu widoków danego kontrolera, czyli np. dla BooksController w `app/views/books`.



## Fragmenty widoków cd.

### ▶ `_form.html.erb`

```
<p>
  <%= form.label :title %><br />
  <%= form.text_field :title %>
</p>
<p>
  <%= form.label :author %><br />
  <%= form.select :author_id, @authors %>
</p>
```

- ▶ domyślnie fragment jest szukany w katalogu widoków danego kontrolera, czyli np. dla BooksController w `app/views/books`.
- ▶ jeśli chcemy skorzystać z fragmentu innego kontrolera, musimy poprzedzić jego nazwę nazwą kontrolera, np.

```
<%= render :partial => "books/form" %>
```

# Pytania

PYTANIA?