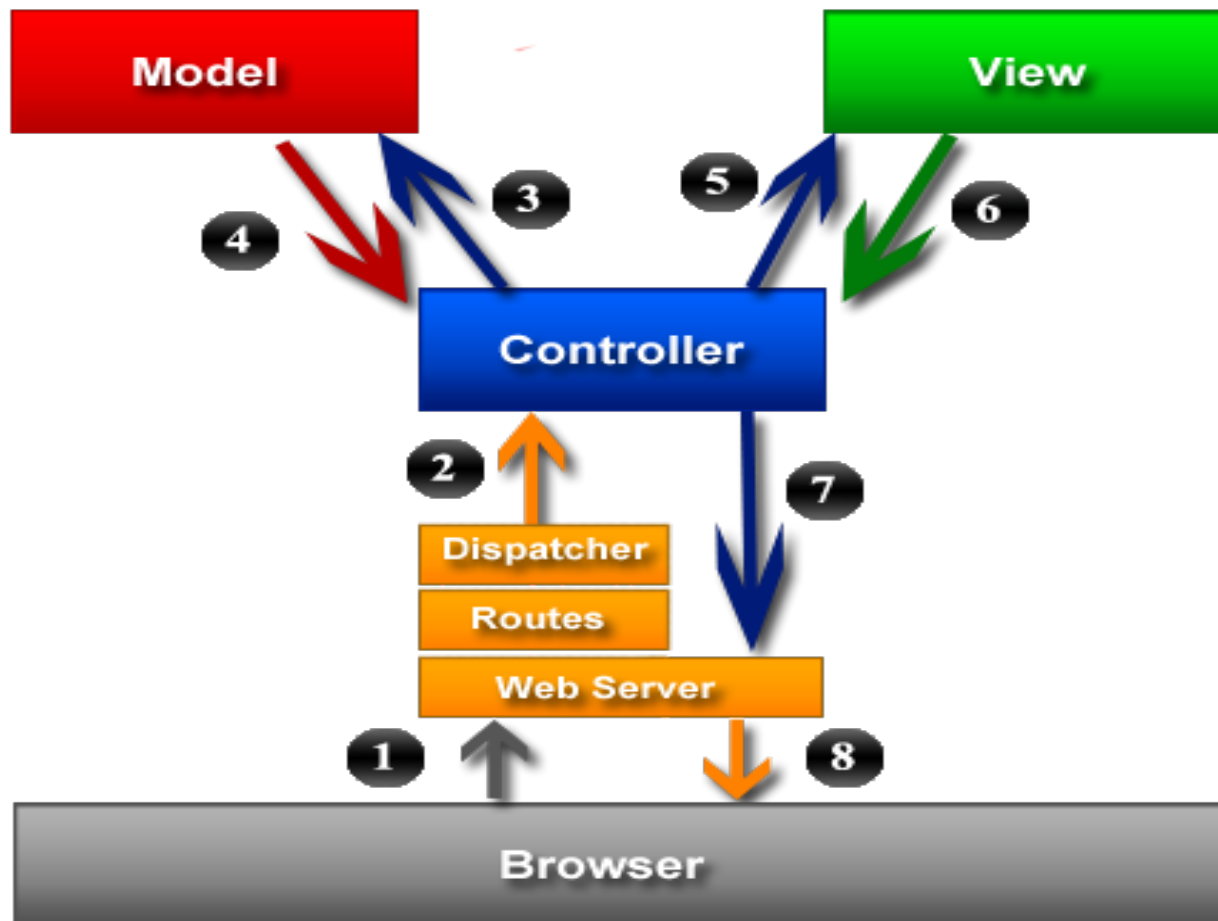
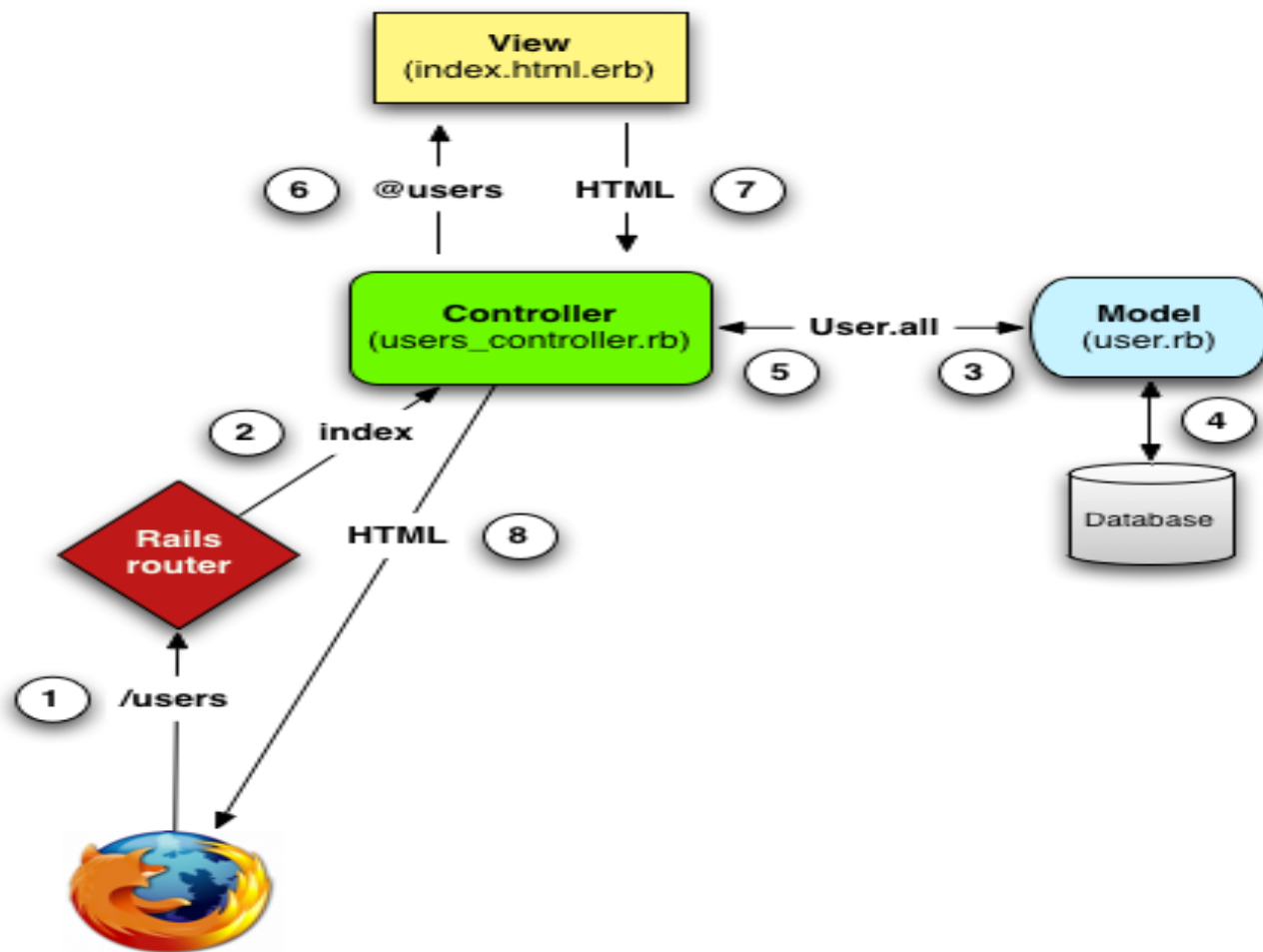


MVC



MVC w Railsach

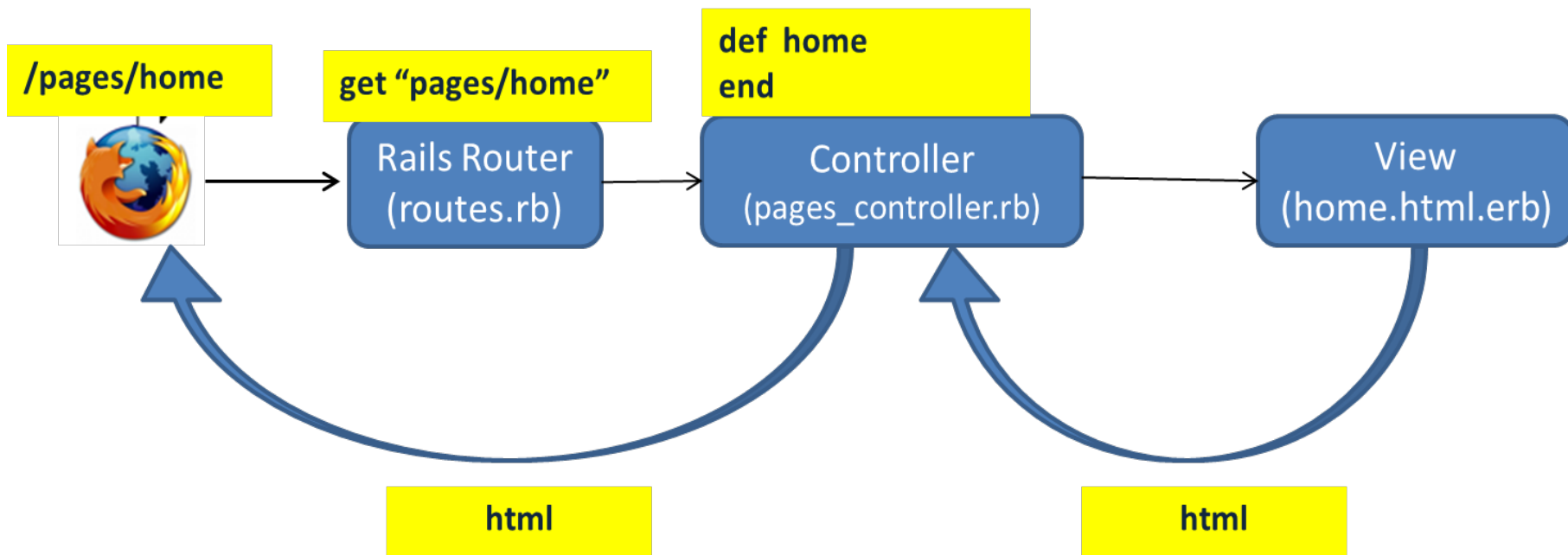


MVC



- na początku użytkownik wpisuje adres URL.
- adres url sprawdzany jest w routes.rb
- „routes.rb” oddaje odpowiedzialność kontrolerowi.
- następnie wyświetlany jest widok html.

Jak działa MVC



Kontroler



- do wygenerowania kontrolera służy polecenie **rails g controller nazwa_kontrolera**
- kontroler odpowiedzialny jest za komunikację pomiędzy widokiem, a modelem.
- warstwa widoku musi być „odizolowana” od logiki
- osoby, które „piszą” w HTML, nie necessarily muszą znać Ruby.

Przykład generowania kontrolera



```
MacBook-Pro-Micha:new michalos$ rails g controller ads index
  create  app/controllers/ads_controller.rb
  route   get "ads/index"
  invoke  erb
  create  app/views/ads
  create  app/views/ads/index.html.erb
  invoke  test_unit
  create  test/controllers/ads_controller_test.rb
  invoke  helper
  create  app/helpers/ads_helper.rb
  invoke  test_unit
  create  test/helpers/ads_helper_test.rb
  invoke  assets
  invoke  coffee
  create  app/assets/javascripts/ads.js.coffee
  invoke  scss
  create  app/assets/stylesheets/ads.css.scss
```

Bez kontrolera...



`<%= User.last.name %>`

Akcja



- akcja to metoda kontrolera

Before_action



- before_action wywoływana jest metoda przed wykonaniem akcji kontrolera
- w Rails 3 było to before_filter
- before_action wywołujemy kiedy np. Mamy powtarzające się operacje dla wielu akcji kontrolera
- przykładowo jeśli chcemy przed wywołaniem kontrolera autentykować użytkownika piszemy before_action

<http://railscasts.com/episodes/400-what-s-new-in-rails-4>

Before_action - przykład



```
class ApplicationController < ActionController::Base

  before_action :require_login

  private

  def require_login

    unless logged_in?

      flash[:error] = "You must be logged in to access this section"

      redirect_to new_login_url # halts request cycle

    end

  end

end
```

Protokół HTTP



GET ([http://pl.wikipedia.org/wiki/GET_\(metoda\)\)](http://pl.wikipedia.org/wiki/GET_(metoda))))

POST ([http://pl.wikipedia.org/wiki/POST_\(metoda\)\)](http://pl.wikipedia.org/wiki/POST_(metoda))))

- w Railsach obecnie musimy wskazać w routes przy jakiej metodzie będzie możliwe wykonanie żądania HTTP.

Definicje routes



`get "people/index" => 'people#index'` mówi nam „jeśli ktoś wpisze ścieżkę `localhost:3000/people/index` to przejdź do kontrolera `people` i akcji `index`”

Linijka `get 'people/index'` robi dokładnie to samo.

- w Railsach obecnie musimy wskazać w routes przy jakiej metodzie będzie możliwe wykonanie żądania HTTP.

Formularze HTML



- służą do przekazywania informacji od użytkownika
- np. możemy za pomocą nich wysłać mail do użytkownika strony
- zalogować się do serwisu
- robić zakupy w sklepie internetowym

Nowa definicja routes



Definicja w routes:

```
get 'products/:id' => 'products#show'
```

Mówi nam, że po ścieżce products będziemy wprowadzać symbol i następnie mamy „oddać działanie” akcji show kontrolera products.

Ścieżka np. Products/1 , ale także ścieżka products/ads

Hash params



- params służy do pobierania wartości wprowadzonych przez użytkownika
- jest to hash
- params należy do klasy ActionController:Parameters
- dzięki niej możemy otrzymać wartości przekazane HTTP GET bądź HTTP POST.

<http://api.rubyonrails.org/classes/ActionController/Parameters.html>

Protokół HTTP



- params służy do pobierania wartości wprowadzonych przez użytkownika
- jest to hash
- params należy do klasy ActionController:Parameters
- dzięki niej możemy otrzymać wartości przekazane HTTP GET bądź HTTP POST.

Aby wyświetlić przykładowy produkt



W kontrolerze:

```
def show
```

```
  @product = Product.find(params[:id])
```

```
end
```

W widoku:

```
<%= @product.name %>
```

Ważna jest kolejność – ze względu na symbol



Definicja w routes:

```
get 'products/add'
```

```
get 'products/choice'
```

```
get 'products/search'
```

```
get 'products/search_results'
```

```
get 'products/:id' => 'products#show'
```

Formularze HTML podstawy



- Formularz rozpoczyna się i kończy znacznikiem form

`<form>`

`</form>`

Akcja formularza



- akcja formularza mówi nam gdzie po kliknięciu na submit mamy zostać przekierunkowani.
- method określa jakiej formy HTTP używamy do przekazania danych

```
<form action="ktos@gdzies.com.pl" method="get" enctype="text/plain">
```

```
<div>
```

Tutaj zawartość formularza

```
</div>
```

```
</form>
```

Kontrolki formularza



- pole tekstowe `<input type="text" name="imie">`
- text area `<textarea name="tresc" rows="5" cols="50">Fragment tekstu</textarea>`
- pole jednokrotnego wyboru

`<input type="radio" name="odpowiedz" value="tak">Tak
`

`<input type="radio" name="odpowiedz" value="nie">Nie
`

`<input type="radio" name="odpowiedz" value="niewiem">Nie wiem`

Kontrolki formularza



- pole listy `<select name="jezyk" size="1">`

`<option>Polski</option>`

`<option>Angielski</option>`

`<option>Niemiecki</option>`

`<option>Francuski</option>`

`</select>`

- pole ukryte `<input type="hidden">`

- submit `<input type="submit">`

Więcej o formularzach



- www.kurshtml.edu.pl
- <http://www.poradnik-webmastera.com/kursy/html/formularze.php>

Rake routes



```
MacBook-Pro-Micha:michal_routes michalos$ rake routes
```

	Prefix	Verb	URI Pattern	Controller#Action
	people	GET	/people(.:format)	people#index
		POST	/people(.:format)	people#create
	new_person	GET	/people/new(.:format)	people#new
	edit_person	GET	/people/:id/edit(.:format)	people#edit
	person	GET	/people/:id(.:format)	people#show
		PATCH	/people/:id(.:format)	people#update
		PUT	/people/:id(.:format)	people#update
		DELETE	/people/:id(.:format)	people#destroy
	products_add	GET	/products/add(.:format)	products#add
	products_choice	GET	/products/choice(.:format)	products#choice
		GET	/products/:id(.:format)	products#show

Rake routes



- rake routes wyświetla zdefiniowane routes, wraz z helperami, które możemy używać w widokach

Przykładowo `get 'people/index'` generuje nam ścieżkę `people_index_path`

Helpery widoków Rails



- helpery widoków to metody Ruby , które zamieniają kod Ruby na kod HTML
- mamy helpery także formularzy
- przykład helpera to np. `link_to` , który zamienia odpowiednio na znacznik `<a>` ``
- są po to żeby pisać jak najmniej kodu:)
- dzięki nim HTML nie musi być straszny:)

Rake routes – przykład użycia helpera



- rake routes wyświetla zdefiniowane routes, wraz z helperami, które możemy używać w widokach

```
<%= link_to "choice",products_choice_path %>
```

- używamy path i url

```
<%= link_to "choice",products_choice_url %>
```

Przykład helpera formularza



```
<%= form_tag("/search", method: "get") do %>
```

```
  <%= label_tag(:q, "Search for:") %>
```

```
  <%= text_field_tag(:q) %>
```

```
  <%= submit_tag("Search") %>
```

```
<% end %>
```

Formularz zamieniony jest na..



```
<form accept-charset="UTF-8" action="/search" method="get">  
  <label for="q">Search for:</label>  
  <input id="q" name="q" type="text" />  
  <input name="commit" type="submit" value="Search" />  
</form>
```

Formularz z path

```
<%= form_tag(people_path) do %>
```

```
  <%= label_tag(:q, "Search for:") %>
```

```
  <%= text_field_tag(:q) %>
```

```
  <%= submit_tag("Search") %>
```

```
<% end %>
```

- Gdy nie podamy z jakiej metody HTTP korzystamy to standardowo używana jest metoda GET.

Najczęstsze błędy z formularzami



- najważniejsze to czytamy opis błędu, bo zazwyczaj on prowadzi do rozwiązania
- częsty błąd to użycie metody POST zamiast GET, jeśli mamy inną metodę, a ta sama ścieżka to dla Railsów jest to inna ścieżka
- przesyłamy inne elementy niż odbieramy

Basic auth




- basic auth to prosta autoryzacja, która polega na podaniu loginu i hasła
- używa się ją często do serwisów w wersji testowych, tak aby z zewnątrz przy testach osoby nie miały dostępu

`http_basic_authenticate_with name: "dhh", password: "secret",
except: :index`

- metoda ta jest wywołana na kontrolerze

Basic Auth

Wiedza ▾ Popularne ▾



Aby zobaczyć tę stronę, należy zalogować się w obszarze „Application” na localhost:3000.

Hasło zostanie wysłane w postaci niezaszyfrowanej.

Nazwa:

Hasło:

☐ Pamiętaj to hasło w moim pęku kluczy

Anuluj

Render action



- jeśli w kontrolerze użyjemy na końcu naszej akcji `render :action` to w tym momencie przechodzimy do widoku `action`

```
def show
```

```
  @person = Person.find(2)
```

```
  render :action => 'status'
```

```
end
```

Match



```
match '/all' => 'tweets#index', :via => [:get], :as => 'all_tweets'
```

- działa podobnie jak get
- as dzięki niemu możemy korzystać z helpera `all_tweets_path`

```
match '/google' => redirect('http://www.google.com/'), :via => [:get]
```

Szablony częściowe



- szablony częściowe to pliki html.erb
- wykorzystujemy je wtedy, kiedy chcemy wykorzystać jeden plik w wielu miejscach
- bądź chcemy jeden plik html.erb rozdzielić na kilka częściowych
- szablony częściowe ich nazwa musi rozpoczynać się od _ np..
_form.html.erb
- aby z nich korzystać w widoku musimy zrobić `render 'form'`



Kurs programowania Ruby on Rails

Kontroler