Лекция 10
View & Model

План

- Шаблонизаторы
- Active Record
- CRUD в Active Record
- Ассоциации

Шаблонизаторы

HTML

```
<!DOCTYPE html>
<html>
    <body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

    </body>
</html>
```

Динамическая генерация

В основе – работа со строками

Хочется выполнять код на языке бэкенда

Быстро (вроде бы)

ERB (erubi, embedded ruby)

```
<h1>
  Hello <%= name %>!
</h1>
```

if - else

```
<% if @favorite_food == "chocolate" %>
   Here are some chocolate bars!
<% else %>
  Here are our top 10 snacks
<% end %>
```

Циклы

```
<% @books.each do |book| %>
  <%= book.title %>
  <%= book.author %>
  <br>
<% end %>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>App Title</title>
    <%= stylesheet_link_tag    'application’%>
    <%= javascript_include_tag 'application’%>
  </head>

  <body>
    <%= yield %>
  </body>
</html>
```

Формы

```
<%= form_with(url: "/search", method: "get") do %>
  <%= label_tag(:q, "Search for:") %>
  <%= text_field_tag(:q) %>
  <%= submit_tag("Search") %>
<% end %>
```

Helpers - Чекбокс

```
<%= check_box_tag(:pet_dog) %>
<%= label_tag(:pet_dog, "I own a dog") %>
<%= check_box_tag(:pet_cat) %>
<%= label_tag(:pet_cat, "I own a cat") %>
```

Helpers - Чекбокс

```
<%= radio_button_tag(:age, "child") %>
<%= label_tag(:age_child, "I am younger than 21") %>
<%= radio_button_tag(:age, "adult") %>
<%= label_tag(:age_adult, "I am over 21") %>
```

```
<%= text_area_tag(:message, "Hi, nice site", size: "24x6") %>
<%= password_field_tag(:password) %>
<%= hidden_field_tag(:parent_id, "5") %>
<%= search_field(:user, :name) %>
<%= telephone_field(:user, :phone) %>
<%= date_field(:user, :born_on) %>
<%= datetime_local_field(:user, :graduation_day) %>
<%= month_field(:user, :birthday_month) %>
<%= week_field(:user, :birthday_week) %>
<%= url_field(:user, :homepage) %>
<%= email_field(:user, :address) %>
<%= color_field(:user, :favorite_color) %>
<%= time_field(:task, :started_at) %>
<%= number_field(:product, :price, in: 1.0..20.0, step: 0.5) %>
<%= range_field(:product, :discount, in: 1..100) %>
```

Вне Rails

```ruby
require 'erb'
Book     = Struct.new(:title, :author)
template = ERB.new(File.read('template.erb'))
template.result_with_hash(books: [Book.new("test"), Book.new("abc")])
```

HAML

```
%head
  %title Title
  = 123 * 2
  %body
    - @collection.each do |x|
      %p
        = x
```

Slim

```
.col-lg-4
  .features-icons-item.mx-auto.mb-5.mb-lg-0.mb-lg-3
    .features-icons-icon.d-flex
      i.icon-calendar.m-auto.theme-red
    h3
      ' Статистика с
      =@data[:min_year]
      '  по
      =@data[:max_year]
      '  год
    p.lead.mb-0
```

Что выбрать?

- ERB быстрее остальных
- HAML и Slim лучше читаются, легче поддерживаются
- View не содержит логики
- JS работает с готовой страницей
- Мешать логику JS и шаблонизаторов не стоит
- Мешать фреймворк и шаблонизаторы не стоит

ORM

Что нужно делать?

- Описывать модели и данные, которые в них хранятся
- Представлять связи между моделями
- Преобразовывать наследование в термины реляционной БД
- Проводить валидацию моделей
- Выполнять операции с моделями как с объектами

Подходы к проектированию

- Code First
- Database First

Соглашение об именовании

Model / Class	Table / Schema
Article	articles
LineItem	line_items
Deer	deers
Mouse	mice
Person	people

Первичный ключ: id

Внешний ключ:

model_name_id

Зарезервированные имена столбцов

```
created_at
updated_at
lock_version
type
association_name_type
table_name_count
```

Со стороны Object

```ruby
class Product < ApplicationRecord
end
```

Со стороны Relational

```sql
CREATE TABLE products (
    id int(11) NOT NULL auto_increment,
    name varchar(255),
    PRIMARY KEY  (id)
);
```

Методы класса

```ruby
p = Product.new
p.name = "Some Book"
puts p.name # "Some Book"
```

Нарушение конвенции

```
class Product < ApplicationRecord
  self.table_name = "my_products"
end
```

```
class Product < ApplicationRecord
  self.primary_key = "product_id"
end
```

Создание новых записей

```
user = User.create(name: "David",
                   occupation: "Code Artist")


user = User.new
user.name = "David"
user.occupation = "Code Artist«
user.save
```

С использованием блока

```
user = User.new do |u|
  u.name = "David"
  u.occupation = "Code Artist"
end
```

Местный select

```ruby
users = User.all
```

```sql
select * from users;
```

```ruby
david = User.find_by(name: 'David')
```

```sql
select first(1) *
from users
where name='David';
```

find

```ruby
client = Client.find(10)
                    #ActiveRecord::RecordNotFound

clients = Client.find([1, 10])

          Client.find(1, 10)
```

take

```ruby
client = Client.take

clients = Client.take(2)

clients = Client.take!(2)
                        #ActiveRecord::RecordNotFound
```

first (last)

```ruby
client  = Client.first
clients = Client.first!(3)
client  = Client.order(:first_name).first
```

```sql
SELECT * FROM clients ORDER BY clients.id ASC LIMIT 3
```

```sql
SELECT *
FROM clients
ORDER BY clients.first_name ASC LIMIT 1
```

find_by

Не выбрасывает ошибку, позволяет выбрать поле

```
Client.find_by first_name: 'Jon'
```

```
Client.find_by_first_name: 'Jon'
```

Выбрасывает ошибку

```
Client.find_by! first_name: 'Jon'
```

all

```
User.all.each do |user|
  NewsMailer.weekly(user).deliver_now
end
```

find_each

```ruby
User.find_each do |user|
  NewsMailer.weekly(user).deliver_now
end
```

```ruby
User.find_each(batch_size: 5000) do |user|
  NewsMailer.weekly(user).deliver_now
end
```

find_each + условия

```ruby
User.find_each(start: 2000) do |user|
  NewsMailer.weekly(user).deliver_now
end
```

```ruby
User.find_each(start: 2000, finish: 10000) do |user|
  NewsMailer.weekly(user).deliver_now
end
```

find_in_batches

```ruby
Invoice.find_in_batches do |invoices|
  export.add_invoices(invoices)
end
```

```
where

Client.where("orders_count = '2'")

Client.where("first_name LIKE '%#{params[:first_name]}%'")

Client.where("orders_count = ?",
                                params[:orders])
```

```ruby
Client.where("orders_count = ? AND locked = ?",
                                params[:orders], false)
```

```ruby
Client.where("created_at >= :start_date
                  AND created_at <= :end_date",
                 {start_date: params[:start_date],
                  end_date: params[:end_date]
                 })
```

Условия в хэше

```
Client.where(locked: true)

Client.where('locked' => true)

Article.where(author: author)

Client.where(created_at:
(Time.now.midnight - 1.day)..Time.now.midnight)
```

Условия в хэше

```
Client.where(orders_count: [1,3,5])
```

```
SELECT *
FROM clients
WHERE clients.orders_count IN (1,3,5)
```

НЕ, ИЛИ

```
Client.where.not(locked: true)
```

```
Client.where(locked: true)
      .or(Client.where(orders_count: [1,3,5]))
```

Сортировка

```
Client.order(:created_at)
Client.order("created_at")
Client.order(created_at: :desc)

Client.order(orders_count: :asc,
             created_at: :desc)

Client.order("orders_count ASC")
      .order("created_at DESC")
```

select

```
Client.select(:viewable_by, :locked)
Client.select("viewable_by, locked")

                ActiveModel::MissingAttributeError

Client.select(:name).distinct
```

Limit и Offset

```
Client.limit(5)
Client.limit(5).offset(30)
```

Group By

```
Order.select("date(created_at) as ordered_date,
              sum(price) as total_price")
     .group("date(created_at)")
```

```
Order.group(:status).count
```

```
# => { 'awaiting_approval' => 7, 'paid' => 12 }
```

having

```
Order.select("date(created_at) as ordered_date,
              sum(price) as total_price")
.group("date(created_at)")
.having("sum(price) > ?", 100)
```

unscope

```
query = Article.where('id > 10’)
                .limit(20)
                .order('id asc')

query.unscope(:order)
```

```sql
SELECT *
FROM articles
WHERE id > 10
ORDER BY id asc
LIMIT 20
```

```sql
SELECT *
FROM articles
WHERE id > 10
LIMIT 20
```

unscope отдельных условий

```
Article.where(id: 10, trashed: false)
       .unscope(where: :id)
```

Переопределения

only
reselect
reorder
reverse_order
rewhere

NULL relation

```ruby
Article.none

@articles = current_user.visible_articles
                        .where(name: params[:name])
```

```ruby
def visible_articles
  case @role
  when 'Country Manager'
    Article.where(country: country)
  when 'Reviewer'
    Article.published
  when 'Bad User'
    Article.none
  end
end
```

Доступ на чтение

```ruby
client = Client.readonly.first
client.visits += 1
client.save

# ActiveRecord::ReadOnlyRecord
```

Ссылки

https://www.rubyguides.com/2018/11/ruby-erb-haml-slim/

https://guides.rubyonrails.org/active_record_querying.html

https://guides.rubyonrails.org/association_basics.html