Конспект RubySchool.us [3]

Урок 20

app.rb

```
require "sinatra"

get '/' do
  "Hello Foo Bar"
end

get '/contacts' do
  @title = "Contacts"
  @message = "Hello all! Phone me now."
  erb :message
end

get '/faq' do
  @title = "FAQ about our products"
  @message = "It`s FAQ"
  erb :message
end

get '/none-page' do
  under_construction
end

def under_construction
  @title = "Under Construction"
  @message = "This page is under construction."
  erb :message
end
```

views/message.erb

```
<h1><%= @title %></h1>

<p><%= @message %></p>
```

Я пользуюсь git в командной строке, но можно использовать ungit (npm-пакет).

Babershop.rb

Listing babershop.rb

```
require 'sinatra'

get '/' do
  erb :index
end

# спросим Имя, номер телефона и дату, когда придёт клиент.
post '/' do
  # user_name, phone, date_time
  @user_name = params[:user_name]
  @phone = params[:phone]
  @date_time = params[:date_time]

  @title = "Thank you!"
  @message = "Уважаемый #{@user_name}, мы ждём вас #{@date_time}"

  # запишем в файл то, что ввёл клиент
  f = File.open 'users.txt', 'a'
  f.write "User: #{@user_name}, phone: #{@phone}, date and time: #{@date_time}.\n"
  f.close

  erb :message
end
```

Listing views/index.erb

```
<h1>Babershop</h1>

<form action="/" method="POST">
  Your name:<br> <input type="text" name="user_name"><br>
  Your phone:<br> <input type="text" name="phone"><br>
  Date and time:<br> <input type="text" name="date_time"><br>

  <input type="submit">
</form>
```

Listing views/message.erb

```
<h1><%= @title %></h1>

<p><%= @message %></p>
```

Задание:

В программу babershop.rb добавить зону /admin где по паролю будет выдаваться список тех, кто записался (из users.txt)

Подсказка: ищи sinatra text file на stackoverflow

Варианты чтения из файла в синатру:

1:

```
get '/result' do
  send_file 'users.txt'
end
```

2:

```
get '/result' do
  @file = File.open("users.txt","r")
  erb :result
  # @file.close - с этим у меня не работает, перенёс в erb
end
```

Listing views/result.erb:

```
<h2>RESULT:</h2>
<% @file.each_line do |line| %>
  <%= line %>
<% end %>

<%= @file.close %>
```

или:

```
<h2>RESULT:</h2>
<%= @file.read %>
<%= @file.close %>
```

Решение:

babershop-2.rb

```
require 'sinatra'

get '/' do
  erb :index
end

# спросим Имя, номер телефона и дату, когда придёт клиент.
post '/' do
  # user_name, phone, date_time
  @user_name = params[:user_name]
  @phone = params[:phone]
  @date_time = params[:date_time]

  @title = "Thank you!"
  @message = "Уважаемый #{@user_name}, мы ждём вас #{@date_time}"

  # запишем в файл то, что ввёл клиент
  f = File.open 'users.txt', 'a'
  f.write "User: #{@user_name}, phone: #{@phone}, date and time: #{@date_time}.\n"
  f.close

  erb :message
end

# Добавить зону /admin где по паролю будет выдаваться список тех, кто записался (из users.txt)

# sinatra text file sso
get '/admin' do
  erb :admin
end

post '/admin' do
  @login = params[:login]
  @password = params[:password]

  # проверим логин и пароль, и пускаем внутрь или нет:
  if @login == 'admin' && @password == 'krdprog'
    @file = File.open("./users.txt","r")
    erb :watch_result
    # @file.close - должно быть, но тогда не работает. указал в erb
  else
    @report = '<p>Доступ запрещён! Неправильный логин или пароль.</p>'
    erb :admin
  end
end
```

views/index.erb

```
<h1>Babershop 2</h1>

<form action="/" method="POST">
  Your name:<br> <input type="text" name="user_name"><br>
  Your phone:<br> <input type="text" name="phone"><br>
  Date and time:<br> <input type="text" name="date_time"><br>

  <input type="submit">
</form>
```

views/admin.erb

```
<h1>Admin Panel</h1>

<form action="/admin" method="POST">
  Login:<br> <input type="text" name="login"><br>
  Password:<br> <input type="password" name="password"><br>

  <input type="submit">
</form>

<%= @report %>
```

views/message.erb

```
<h1><%= @title %></h1>

<p><%= @message %></p>
```

views/watch_result.erb

```
<h1>Admin Panel (Watch Result)</h1>

<table border="1">
<% @file.each_line do |line| %>
  <tr><td><%= line %></td></tr>
<% end %>
</table>

<%= @file.close %>
```

Урок 21

> Посмотри ссылку про git: http://think-like-a-git.net/

В Sinatra можно создать каталог **public** и всё, что в нём будет размещено, будет доступно из веба.

style.css - надо размещать в каталог /public

> Sinatra Bootstrap: https://github.com/bootstrap-ruby/sinatra-bootstrap

Bash:

```
git clone https://github.com/bootstrap-ruby/sinatra-bootstrap.git
mv sinatra-bootstrap/ app
cd app
bundle install
bundle exec ruby app.rb
```

Результат в браузере: http://localhost:4567

Урок 22

Решение, чтобы не перезапускать Sinatra

- https://rubygems.org/gems/sinatra-reloader - gem sinatra reloader

- https://rubygems.org/gems/sinatra-contrib - gem sinatra contrib (в его состав входит sinatra-reloader, возможно в некоторых случаях надо попробовать поставить его).

Надо установить гем sinatra-reloader:

```
gem install sinatra-reloader
```

или установить sinatra-contrib:

```
gem install sinatra-contrib
```

или так:

```
sudo apt install ruby-sinatra-contrib
```

и, добавляем в свои app.rb строчку:

```
require 'sinatra/reloader'
```

Про yield в views/layout.erb

app.rb:

```
require 'sinatra'

get '/' do
  erb "Hello all Ruby-programmers!"
end
```

views/layout.erb:

```
<h1>Some text</h1>
<%= yield %>
```

Информация из app.rb будет помещена в yield в views/layout.erb

В файле views/layout.erb можно сделать основной каркас страницы, разместить yield и уже из других erb подгружать информацию.

app.rb

```
get '/foo' do
  erb :foo
end
```

views/layout.erb

```
<p>This block for all site</p>
<p>Block navigation</p>
<%= yield %>
```

views/foo.erb

```
<h1>Foo Header</h1>
<p>Text for Foo.</p>
```

Напомним, как работает yield:

```
def show_me_text
  puts "<body>"
  yield
  puts "</body>"
end

show_me_text { puts "Foo!" }

# OR:

show_me_text do
  puts "Foo!"
end

# puts "Foo!" вставится в место, где указан yield
```

Задание:

В Babershop добавить возможность выбора парикмахера (через select).

Решение:

В views/index.erb добавить часть кода:

```
  <label for="baber">Выбор парикмахера</label>
  <select name="baber">
    <option value="none" selected>Выберите парикмахера...</option>
    <option value="Петрович">Петрович</option>
    <option value="Макарыч">Макарыч</option>
    <option value="Федорыч">Федорыч</option>
  </select>
```

А, в babershop-2.rb добавить и исправить код:

```
post '/' do
  # user_name, phone, date_time
  @user_name = params[:user_name]
  @phone = params[:phone]
  @date_time = params[:date_time]
  @baber = params[:baber]

  @title = "Thank you!"
  @message = "Уважаемый #{@user_name}, мы ждём вас #{@date_time} у выбранного парикмахера #{@baber}."

  # запишем в файл то, что ввёл клиент
  f = File.open 'users.txt', 'a'
  f.write "User: #{@user_name}, phone: #{@phone}, date and time: #{@date_time}. Baber: #{@baber}.\n"
  f.close

  erb :message
end
```

Здесь мы добавили код:

```
@baber = params[:baber]
```

плюс добавили в @message и в код записи в файл f.write

Урок 23

```
<script src="script.js"></script>
```

jQuery:

```
$('#aaa').css('background-color','yellow');
$('.ccc').css('background-color','red');
```

Код в { ... } выполнится после того, как загрузится вся страница:

```
<body>
  <script>
    $(function() {
      ...
    });
  </script>
</body>
```

Урок 24

Валидация - проверка параметров

Вариант 1:

app.rb

```
   if @user_name == ''
     @error = 'Ошибка: Введите имя!'
     return erb :index
   end
```

views/index.erb

```
<p><strong><%= @error %></strong></p>
```

> Атомарность коммитов.

Возврат view вынесем отдельно:

```
if @error != ''
  return erb :index
end
```

Для проверки валидации можно использовать хеши:

```
# хеш для валидации параметров
hh = { :user_name => 'Введите имя',
  :phone => 'Введите телефон',
  :date_time => 'Введите дату и время' }

# для каждой пары ключ-значение
hh.each do |key, value|
  # если параметр пустой
  if params[key] == ''
    # переменной @error присвоить value из хеша hh
    @error = hh[key]

    # вернуть представление
    return erb :index
  end
end
```

Как сделать, чтобы введённое значение оставалось в поле после вывода ошибки валидации:

Добавим атрибут value в файле views/index.erb:

```
Your name:<br> <input type="text" name="user_name" value="<%= @user_name %>">
```

В итоге получим улучшенную версию views/index.erb

```
<h1>Babershop</h1>

<p><strong><%= @error %></strong></p>

<form action="/" method="POST">
  Your name:<br> <input type="text" name="user_name" value="<%= @user_name %>"><br>
  Your phone:<br> <input type="text" name="phone" value="<%= @phone %>"><br>
  Date and time:<br> <input type="text" name="date_time" value="<%= @date_time %>"><br>

  <br><br>
  <label for="baber">Выбор парикмахера</label>
  <select name="baber">
    <option value="none" selected>Выберите парикмахера...</option>
    <option value="Петрович">Петрович</option>
    <option value="Макарыч">Макарыч</option>
    <option value="Федорыч">Федорыч</option>
  </select>
  <br><br>

  <input type="submit">
</form>
```

Ещё можно сделать так (будет выводить через запятую пустые поля):

Edit babershop-2.rb:

```
# хеш для валидации параметров
hh = { :user_name => 'Введите имя',
    :phone => 'Введите телефон',
    :date_time => 'Введите дату и время' }

@error = hh.select {|key,_| params[key] == ''}.values.join(", ")

if @error != ''
  return erb :index
end
```

Вынесем это решение в отдельный метод (сделать самостоятельно).

Домашнее задание:

Сделать отправку данных, которые введены в форму на электронную почту. Через гем Pony. Описание тут: https://stackoverflow.com/questions/2068148/contact-form-in-ruby-sinatra-and-haml

Главное, чтобы на gmail или другом почтовом сервере был открыт 587 порт.

> gem pony: https://github.com/benprew/pony/ Документация gem Pony: https://www.rubydoc.info/gems/pony/1.12

Урок 25

Базы данных - gem sqlite3

bash:

```
gem install sqlite3
```

> sqlite - https://sqlite.org/index.html

NOTE: При первой установке и настройке, помимо установки гема, возможно потребуется дополнительно установить libsqlite3-dev по следующей команде (перед установкой гема):

```
sudo apt-get install libsqlite3-dev
```

Язык запросов SQL

```
sqlite3 --version
```

Создать базу данных в терминале:

```
sqlite3 my_base.sqlite
```

Немного SQL:

> Пройти упражнения на тренажёре SQL is Hard: http://www.sqlishard.com/

Выбрать все колонки для таблицы:

```
SELECT * FROM Customers
```

Выбрать определённые колонки:

```
SELECT Id, FirstName FROM Customers
```

или в другом порядке:

```
SELECT FirstName, Id FROM Customers
```

WHERE:

```
SELECT * FROM Customers WHERE id = 5
```

WHERE conditions:

```
SELECT * FROM Orders WHERE OrderTotal BETWEEN 100 AND 200
```

```
SELECT * FROM Orders WHERE DeliveryTime < '2013-06-20'
```

LIKE (поиск):

```
SELECT * FROM Customers WHERE LastName LIKE 'A%'
```

OR:

```
продолжение http://www.sqlishard.com/Exercise#/exercises/SELECT/S2.4
```

Инструмент:

> SQLite Browser https://sqlitebrowser.org/dl/

- создадим базу данных test.sqlite
- создадим таблицу Cars со столбцами: -- Id (INTEGER) - первичный ключ, автоинкремент: INTEGER PRIMARY KEY AUTOINCREMENT; -- Name (VARCHAR) -- Price (INTEGER)

```
CREATE TABLE "Cars" ("Id" INTEGER PRIMARY KEY AUTOINCREMENT, "Name" VARCHAR, "Price" INTEGER)
```

Пример запроса:

```
SELECT * FROM Cars WHERE Price > 1000
```

Создание - INSERT:

```
INSERT INTO Cars (Id, Name, Price) VALUES (1, 'BMW', 10000)
```

А, с автоинкрементом:

```
INSERT INTO Cars (Name, Price) VALUES ('Audi', 3000)
```

Откроем базу через терминал:

```
sqlite3 test.sqlite
```

Список таблиц:

```
.tables
```

Сделаем запрос (в конце запроса точка с запятой, иначе на другую строку переходит вод запроса):

```
SELECT * FROM Cars
```

Фишка .mode column

```
.mode column
```

Включить заголовки:

```
.headers on
```

Добавить:

```
INSERT INTO Cars (Name, Price) VALUES ('Foo', 6743)
```

Выйти:

```
.exit
```

Как обращаться к базе данных из программы:

app.rb

```
require 'sqlite3'

# подключим базу данных (которую делали выше)
db = SQLite3::Database.new 'test.sqlite'

# добавим новую информацию в базу данных
db.execute "INSERT INTO Cars (Name, Price) VALUES ('Jaguar', 7000000)"

# прочитаем данные из базы
db.execute "SELECT * FROM Cars" do |car|
  puts car
  puts "======"
end

db.close
```

Обратить внимание на тему SQL Injection

```
'DROP TABLE Cars --
```

Это нужно фильтровать. Как? Будет ниже.

Домашнее задание:

1. Создать для BaberShop бд с таблицами:

Users:

Id - идентификатор, primary key, автоинкремент, тип INTEGER

Name - Varchar

Phone - Varchar

DateStamp - Varchar

Baber - Varchar

Color - Varchar

Contacts:

Id - идентификатор, primary key, автоинкремент, тип INTEGER

Email - Varchar

Message - Varchar

2. **Добавить через терминал и sqlite3 несколько записей в таблицу Users и Contacts**

Продолжение конспекта: Урок 26-30 - https://github.com/krdprog/rubyschool-notes/blob/master/rubyschool-notes-04.md

Содержание конспекта: