Конспект RubySchool.us [4]

Урок 26

Как сохранять введённые данные в поле selected (html):

```
<option value="Опция 1"<%= @option == 'Опция 1' ? ' selected' %>>Опция 1</option>
```

app.rb

```
require 'sqlite3'

db = SQLite3::Database.new 'base.sqlite'
```

В SQLite использовать тип TEXT, поправка к предыдущему уроку.

SQL:

Такой способ записи будем применять для создания базы данных и таблиц в ней при инициализации приложения:

```
CREATE TABLE IF NOT EXISTS "Users" ("Id" INTEGER PRIMARY KEY AUTOINCREMENT, "Username" TEXT, "Phone" TEXT);
```

Удалить таблицу:

```
DROP TABLE Users;
```

> Ссылка: Шпаргалка по SQLite3 (русская версия) https://github.com/krdprog/sqlite-3-rus-howto
>
> Базу данных не принято коммитить в репозиторий и мешать с исходным кодом. Базу данных надо добавлять в .gitignore

В Sinatra есть специальная команда, которая запускается при инициализации приложения:

```
configure do
  # some code
end
```

Создадим базу данных при инициализации приложения:

При этом у нас подключены гемы:

```
require 'sinatra'
require 'sqlite3'
```

```
configure do
  @db = SQLite3::Database.new 'base.db'

  @db.execute 'CREATE TABLE IF NOT EXISTS "Messages"
    (
      "id" INTEGER PRIMARY KEY AUTOINCREMENT,
      "username" TEXT,
      "phone" TEXT,
      "email" TEXT,
      "option" TEXT,
      "comment" TEXT
    )'
  # db.close (надо?)
end
```

или в одну строку, но менее читаемо:

```
configure do
  @db = SQLite3::Database.new 'base.db'

  @db.execute 'CREATE TABLE IF NOT EXISTS "Messages" ("id" INTEGER PRIMARY KEY AUTOINCREMENT, "username" TEXT, "phone" TEXT, "email"
  # @db.close (надо?)
end
```

> Ссылка: Escaping Strings For Ruby SQLite Insert https://stackoverflow.com/questions/9614236/escaping-strings-for-ruby-sqlite-insert
>
> Prepare: https://www.rubydoc.info/github/luislavena/sqlite3-ruby/SQLite3/Database#prepare-instance_method
>
> Про вставку данных: https://stackoverflow.com/questions/13462112/inserting-ruby-string-into-sqlite
>
> Документация: https://www.rubydoc.info/github/luislavena/sqlite3-ruby
>
> https://www.rubydoc.info/github/luislavena/sqlite3-ruby/SQLite3/Database

Вставка данных в БД в Sinatra app:

Переменная @db не сработала, сделали через db

```
require 'sinatra'
require 'sqlite3'

def get_db
  return SQLite3::Database.new 'base.db'
end

configure do
  db = get_db
  db.execute 'CREATE TABLE IF NOT EXISTS "Messages"
    (
      "id" INTEGER PRIMARY KEY AUTOINCREMENT,
      "username" TEXT,
      "phone" TEXT,
      "email" TEXT,
      "option" TEXT,
      "comment" TEXT
    )'
  db.close
end

def save_form_data_to_database
  db = get_db
  db.execute 'INSERT INTO Messages (username, phone, email, option, comment)
  VALUES (?, ?, ?, ?, ?)', [@username, @phone, @email, @option, @comment]
  db.close
end

get '/' do
  @title = "Форма заявки для Sinatra (Ruby)"
  erb :index
end

post '/' do
  @username = params[:username]
  @phone = params[:phone]
  @email = params[:email]
  @option = params[:option]
  @comment = params[:comment]

  save_form_data_to_database

  @title = "Спасибо, ваше сообщение отправлено"
  erb :sent
end
```

> Полная работающая версия формы со всеми файлами тут: https://github.com/krdprog/contact-form-for-sinatra-ruby-rus

К полю выбора даты можно подключить плагин:

> Плагин для выбора даты (js) https://github.com/xdan/datetimepicker

Вывод результатов (данных) из базы данных:

Чтобы выводить результаты в виде хеша, надо добавить:

```
db.results_as_hash = true
```

Добавим в ранее созданный метод и переделаем app.rb:

```
def get_db
  db = SQLite3::Database.new 'base.db'
  db.results_as_hash = true
  return db
end
```

Домашнее задание:

1. сделать страницу показа результата из базы данных используя запрос

```
SELECT * FROM Users ORDER BY id DESC
```

> Ссылка SQLite ORDER BY: http://www.tutorialspoint.com/sqlite/sqlite_order_by.htm

2. В configure сделать дополнительную таблицу Barbers со списком парикмахеров. Загружать список парикмахеров в configure (вставка в таблицу 1 раз).
3. Загрузить данные из таблицы Barbers в форму в выпадающий список select.

Урок 27

> Отделить логику от представления.

Синтаксис в erb, будет выполнено, но ничего не будет выводиться:

```
<% %>
```

Будет выводиться в месте, где стоит:

```
<%= %>
```

Отображение на веб-странице данных из базы данных:

Добавить в app.rb

```
# Method connect with database
def get_db
  @db = SQLite3::Database.new 'base.db'
  @db.results_as_hash = true
  return @db
end

# Show result in admin panel
get '/admin/show' do
  get_db

  @results = @db.execute 'SELECT * FROM Messages ORDER BY id DESC'
  @db.close

  erb :show
end
```

views/show.erb

```
<table border="1">
  <tr>
    <th>Имя</th>
    <th>Телефон</th>
    <th>E-mail</th>
    <th>Опция</th>
    <th>Комментарий</th>
  </tr>

<% @results.each do |row| %>

  <tr>
    <td><%= row['username'] %></td>
    <td><%= row['phone'] %></td>
    <td><%= row['email'] %></td>
    <td><%= row['option'] %></td>
    <td><%= row['comment'] %></td>
  </tr>

<% end %>

</table>
```

Покрасим в красный цвет первую запись:

```
<%= "style='background-color: red; color: white;'" if row == @results.first %>
```

erb:

```
<% @results.each do |row| %>

  <tr <%= "style='background-color: red; color: white;'" if row == @results.first %>>
    <td><%= row['username'] %></td>
    <td><%= row['phone'] %></td>
    <td><%= row['email'] %></td>
    <td><%= row['option'] %></td>
    <td><%= row['comment'] %></td>
  </tr>

<% end %>
```

Решение задания:

> В configure сделать дополнительную таблицу Barbers со списком парикмахеров. Загружать список парикмахеров в configure (вставка в таблицу 1 раз) - сделаю для контактной формы для поля Options:

```
# Method validation data Options table in database
def is_option_exists? base, param
  base.execute('SELECT * FROM Options WHERE option=?', [param]).length > 0
end

# Method add data to table in database
def seed_db base, options
  options.each do |option|
    if !is_option_exists? @db, option
      base.execute 'INSERT INTO Options (option) VALUES (?)', [option]
    end
  end
end
```

```
# Configure application
configure do
  get_db
  # create table Messages in database
  @db.execute 'CREATE TABLE IF NOT EXISTS "Messages"
    (
      "id" INTEGER PRIMARY KEY AUTOINCREMENT,
      "username" TEXT,
      "phone" TEXT,
      "email" TEXT,
      "option" TEXT,
      "comment" TEXT
    )'

  # create table Options in database
  @db.execute 'CREATE TABLE IF NOT EXISTS "Options"
    (
      "id" INTEGER PRIMARY KEY AUTOINCREMENT,
      "option" TEXT
    )'

  # add data to table
  seed_db @db, ['Foo', 'Faa', 'Moo', 'Zoo', 'Faz', 'Maz', 'Kraz']

  @db.close
end
```

seed_db - seed устоявшееся выражение - наполнить

Понятие миграция (база данных).

Механизмы миграции, чтобы версия базы данных всегда совпадала с версией кода.

to keep up to date - держать в актуальном состоянии

Откат БД вместе с программным кодом.

Дальше будем проходить наполнение и миграции. База наполняется из програмного кода.

Синтаксис before (в sinatra) - исполняет код перед запросом - будет доступно во всех представлениях

```
before do
  # some code
end
```

Как загружать варианты в select формы из базы данных:

В app.rb добавим часть кода:

```
# Index page with form
get '/' do

  # Write to array data from database table Options
  get_db
  @options = @db.execute 'SELECT * FROM Options'
  @db.close

  @title = "Форма заявки для Sinatra (Ruby)"
  erb :index
end
```

в views/index.erb в форме заменим статические данные на :

```
<p>
  <select name="option">
    <option value="" selected>Выбрать опцию...</option>
    <% @options.each do |item| %>
    <option value="<%= item['option'] %>"><%= item['option'] %></option>
    <% end %>
  </select>
</p>
```

Вставка, чтобы сделать select - selected

```
<option <%= @barber == item['name'] ? 'selected' : '' %>><%= item['name'] %></option>
```

configure - при запуске программы before - при каждом обращении к программе

Урок 28

> Мой вариант Sinatra Blog: https://github.com/krdprog/sinatra-blog

Перенаправление:

```
post '/new'
  # some code

  redirect to '/'
end
```

```
  redirect to ('/post/' + @post_id)
```

id поста:

app.rb

```
get '/post/:post_id' do
  @post_id = params[:post_id]
  erb :post
end
```

views/post.erb

```
This is post: <%= @post_id %>
```

Домашнее задание:

1. Авторизация для автора
2. Валидация поля комментария

Ссылка на документацию Sinatra:

- sinatra/README.ru.md at master · sinatra/sinatra · GitHub

Урок 29 - введение в Active Record

Gemfile

```
source "https://rubygems.org"

gem "sinatra"
gem "sinatra-contrib"
gem "sqlite3"
gem "activerecord"
gem "sinatra-activerecord"

group :development do
  gem "tux"
end
```

Установка

```
bundle install
```

gem tux

deploy - развёртывание

Подключение базы и ActiveRecord

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

set :database, { adapter: 'sqlite3', database: 'my_database.db' }

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

Создадим сущности

```
class Client < ActiveRecord::Base
end
```

В терминале запустить гем tux

```
tux
```

Список всех сущностей в БД (набрать в tux):

```
Client.all
```

rake

Создать Rakefile

```
touch Rakefile
```

Содержимое Rakefile:

```
require "./app"
require "sinatra/activerecord/rake"
```

> У меня в Debian не заработало, выпадали ошибки. Удалил ruby и поставил всё через rvm - https://rvm.io/

Install RVM:

```
sudo apt-get install libgdbm-dev libncurses5-dev automake libtool bison libffi-dev
gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB
curl -sSL https://get.rvm.io | bash -s stable
source ~/.rvm/scripts/rvm
rvm install 2.5.3
rvm use 2.5.3 --default
ruby -v
```

> LINK command line ruby cheat sheets: http://cheat.errtheblog.com/s/rvm

Всё заработало, идём дальше...

Список параметров:

```
rake -T
```

NOTE: Работает только при наличии Rakefile:

```
# Rakefile

require "./app"
require "sinatra/activerecord/rake"
```

Rakefile произошёл от сишного Makefile

3 команды rake:

создаёт новую миграцию в db/migrate/:

```
rake db:create_migration NAME=name_of_migration
```

применяет (выполняет) созданную миграцию:

```
rake db:migrate
```

если команда отрабатывает без каких-либо сообщений и база данный пустая, выполнить следующие шаги:

```
- Добавить в Gemfile: gem "rake"
- Запустить bundle install
- В app.rb изменить строку set :database, "sqlite3:barbershop.db"  на set :database, {adapter: "sqlite3", database: "barbershop.db"}
```

возврат к предыдущей миграции:

```
rake db:rollback
```

Все команды rake:

```
rake db:create              # Creates the database from DATABASE_URL or con...
rake db:create_migration    # Create a migration (parameters: NAME, VERSION)
rake db:drop                # Drops the database from DATABASE_URL or confi...
rake db:environment:set     # Set the environment value for the database
rake db:fixtures:load       # Loads fixtures into the current environment's...
rake db:migrate             # Migrate the database (options: VERSION=x, VER...
rake db:migrate:status      # Display status of migrations
rake db:rollback            # Rolls the schema back to the previous version...
rake db:schema:cache:clear  # Clears a db/schema_cache.yml file
rake db:schema:cache:dump   # Creates a db/schema_cache.yml file
rake db:schema:dump         # Creates a db/schema.rb file that is portable ...
rake db:schema:load         # Loads a schema.rb file into the database
rake db:seed                # Loads the seed data from db/seeds.rb
rake db:setup               # Creates the database, loads the schema, and i...
rake db:structure:dump      # Dumps the database structure to db/structure.sql
rake db:structure:load      # Recreates the databases from the structure.sq...
rake db:version             # Retrieves the current schema version number
```

Миграция - это очередная версия нашей базы данных.

Создадим миграцию:

```
rake db:create_migration NAME=create_clients
```

Создаётся каталог db/migrate и миграция.

Открываем созданный файл и создаём таблицу в базе данных:

```
# File db/migrate/389173729_create_clients.rb

class CreateClients < ActiveRecord::Migration[5.2]
  def change
    create_table :clients do |t|
      t.text :name
      t.text :phone
      t.text :datestamp
      t.text :barber

        t.timestamps
    end
  end
end
```

> id primary key будет создан автоматически
>
> t.timestamps создаст 2 дополнительных столбца created_at и updated_at дата создания и обновления сущности

text -> TEXT

string -> VARCHAR(255)

Запустить миграцию:

```
rake db:migrate
```

Мы настроили mapping (ORM - Object Relational Mapper)

Связка ООП с реляционными БД

Добавим таблицу barbers:

```
# + in app.rb

class Barber < ActiveRecord::Base
end
```

Создаём миграцию (bash):

```
rake db:create_migration NAME=create_barbers
```

Правим миграцию (создаём таблицу и вносим данные):

```
# db/migrate/37837298_create_barbers.rb

class CreateBarbers < ActiveRecord::Migration[5.2]
  def change
    create_table :barbers do |t|
      t.text :name

      t.timestamps
    end

    Barber.create :name => "Joe Doe"
    Barber.create :name => "Elon Musk"
    Barber.create :name => "Alisha Moon"
    Barber.create :name => "Marie Fooo-bar"

  end
end
```

Запустим миграцию:

```
rake db:migrate
```

> Совет от @krdprog: чтобы в командной строке sqlite3 каждый раз не писать, показывать в столбец и с заголовками, создайте в домашней директории файл .sqliterc с содержимым:

```
.headers on
.mode column
```

Теперь, это настройки по-умолчанию.

Едем дальше...

Откроем консоль tux

```
tux
```

И введём:

```
Barber.count
```

Из этой консоли можно создать дополнительные сущности.

Замечание:

```
Barber.create # создаёт в БД

b = Barber.new # создаёт в памяти
b.save # после этого надо сделать
```

Создадим в консоли tux нового парикмахера:

```
Barber.create :name => 'Faz Maz'
```

или через .new

```
b = Barber.new :name => 'Vasya' # создадим, но не сохраним в базе
b.new_record? # покажет, новый ли это объект
b.save # сохраним в базе
```

Все записи Barber:

```
Barber.all
```

rake db:migrate надо запускать в каталоге приложения, где Rakefile

> **Изучи ссылку:** Active Record Query Interface — Ruby on Rails Guides; https://guides.rubyonrails.org/active_record_querying.html

Создадим вывод на страницу список наших парикмахеров:

```
# + in app.rb

get '/' do
  @barbers = Barber.all
  erb :index
end
```

```
# + in views/index.erb

<h2>Список парикмахеров:</h2>

<ul>
<% @barbers.each do |barber| %>
  <li><%= barber.name %></li>
<% end %>
</ul>
```

Сортировка .order:

```
@barbers = Barber.order "created_at DESC"
```

Домашнее задание: 1. сделать сохранение записи к парикмахеру в БД с помощью ActiveRecord 2. сделать сущность Contact и на странице /contacts сохранять в БД данные с помощью ActiveRecord

Урок 30

5 шагов по созданию миграции:

1. подключаем БД

```
set :database, "sqlite3.my_database.db"
```

2. создаём модель (класс)

```
class Client < ActiveRecord::Base
end
```

3. создаём миграцию

```
rake db:create_migration NAME=create_clients
```

4. редактируем миграцию

```
class CreateClients < ActiveRecord::Migration[5.2]
  def change
    create_table :clients do |t|
      t.text :name
      t.text :phone
      t.text :datestamp
      t.text :barber

      t.timestamps
    end
  end
end
```

5. делаем миграцию

```
rake db:migrate
```

Необходимо наличие всех гемов и Rakefile

Сохранение в БД через ActiveRecord

Ламерский способ:

```
# + to app.rb

post '/order' do
  @name = params[:name]
  @phone = params[:phone]
  @datestamp = params[:datestamp]
  @barber = params[:barber]

  c = Client.new
  c.name = @name
  c.phone = @phone
  c.datestamp = @datestamp
  c.barber = @barber
  c.save

  erb :sent
end
```

Способ лучше:

```
# + to app.rb
post '/order' do
  c = Client.new params[:client]
  c.save

  erb :sent
end
```

```
# + to views/order.erb

<form action="/order" method="POST">

  <p><label for="name">Ваше имя:</label></p>
  <p><input type="text" name="client[name]" value=""></p>

  <p><label for="phone">Телефон:</label></p>
  <p><input type="text" name="client[phone]" value=""></p>

  <p><label for="datestamp">Выберите дату:</label></p>
  <p><input type="text" name="client[datestamp]" value=""></p>

  <p><label for="barber">Парикмахер:</label>
    <select name="client[barber]">
      <option value="" selected>Выбрать парикмахера...</option>

      <% @barbers.each do |barber| %>
      <option value="<%= barber.name %>"><%= barber.name %></option>
      <% end %>

    </select>
  </p>

  <br>
  <p><input type="submit" value="Отправить заявку"></p>

</form>
```

Т.е. вместо создания переменных из params. мы сократили код до пары строк:

```
c = Contact.new params[:contact]
c.save
```

и в представлении:

```
<p><input type="text" name="contact[name]" value=""></p>
<p><textarea rows="10" cols="45" name="contact[comment]"></textarea></p>
```

> Важное замечание: метод save для новых записей проводит валидацию, если всё правильно, то возвращает true иначе false

Настройка валидации:

Можно проверить пустое - не пустое, можно проверить длину

```
# + to app.rb

class Client < ActiveRecord::Base
  validates :name, presence: true
  validates :phone, presence: true
  validates :datestamp, presence: true
  validates :barber, presence: true
end
```

Проверка в tux:

```
c = Client.new
c.valid?
c.errors.count
c.errors.messages
```

```
# + to app.rb

post '/order' do
  c = Client.new params[:client]
  c.save

  if c.save
    erb "<p>Thank you!</p>"
  else
    erb "<p>Error</p>"
  end
end
```

или:

```
# + to app.rb

post '/order' do
  c = Client.new params[:client]
  c.save

  if c.save
    erb "<p>Thank you!</p>"
  else
    @error = c.errors.full_messages.first
    erb :order
  end
end
```

```
# + views/order.erb

<p style="color: red"><%= @error %></p>
```

Чтобы сохранять данные в поле формы после перезагрузки страницы при неправильном заполнении, надо сделать так:

```
# + to app.rb

get '/order' do
  @c = Client.new

  erb :order
end

post '/order' do
  @c = Client.new params[:client]
  @c.save

  if @c.save
    erb "<p>Thank you!</p>"
  else
    @error = @c.errors.full_messages.first
    erb :order
  end
end
```

```
# + to views/order.erb

<p><input type="text" name="client[name]" value="<%= @c.name %>"></p>
<p><input type="text" name="client[phone]" value="<%= @c.phone %>"></p>
<p><input type="text" name="client[datestamp]" value="<%= @c.datestamp %>"></p>
```

Для textarea:

```
<p><textarea rows="10" cols="45" name="contact[comment]"><%= @c.comment %></textarea></p>
```

> Ссылка на полный проект Barbershop Sinatra with ActiveRecord: https://github.com/krdprog/barbershop-sinatra-with-activerecord

Популярные свойства валидации ActiveRecord:

> Ссылка: Active Record Validations — Ruby on Rails Guides https://guides.rubyonrails.org/active_record_validations.html

- length - length: { minimum: 3 } https://guides.rubyonrails.org/active_record_validations.html#length
- numericality: true - проверка, введены ли числа
- inclusion - https://guides.rubyonrails.org/active_record_validations.html#inclusion

> **Домашнее задание**: переписать блог с использованием ActiveRecord

Создадим отдельные страницы для каждого парикмахера:

```
# + to app.rb

get '/barber/:id' do
  @barber = Barber.find(params[:id])

  erb :barber
end
```

```
# + to views/index.erb

<h2>Список парикмахеров:</h2>

<ul>
<% @barbers.each do |barber| %>
  <li><a href="/barber/<%= barber.id %>"><%= barber.name %></a></li>
<% end %>
</ul>
```

```
# + to views/barber.erb

<h2>Barber page</h2>

<p>Name: <%= @barber.name %></p>
```

> Find в ActiveRecord: https://guides.rubyonrails.org/active_record_querying.html

Показать записавшихся в обратном порядке (кто свежий - наверху):

```
# + to app.rb

get '/clients' do
  @clients = Client.order('created_at DESC')

  erb :clients
end

get '/clients/:id' do
  @client = Client.find(params[:id])

  erb :client
end
```

```
# + to views/clients.erb

<h2>Список записавшихся</h2>

<table border="1" cellpadding="10" cellpadding="0">
  <tr>
    <th>Name</th>
    <th>Phone</th>
    <th>Date</th>
    <th>Barber</th>
  </tr>

<% @clients.each do |client| %>
  <tr>
    <td><a href="/clients/<%= client.id %>"><%= client.name %></a></td>
    <td><%= client.phone %></td>
    <td><%= client.datestamp %></td>
    <td><%= client.barber %></td>
  </tr>
<% end %>
</table>
```

```
# + to views/client.erb

<h2>Страница клиента:</h2>

<p><strong>Имя:</strong> <%= @client.name %></p>

<p><a href="/clients"><< назад к списку записавшихся</a></p>
```

В ActiveRecord есть штука, которая связывает автоматически посты и комментарии. Домашнее задание - найти это и написать движок блога.

> ActiveRecord "one-to-many" (отношение сущностей) https://guides.rubyonrails.org/association_basics.html

---

Продолжение конспекта: Урок 31-35 - https://github.com/krdprog/rubyschool-notes/blob/master/rubyschool-notes-05.md

Содержание конспекта: