Игорь Борисов

Ruby on Rails

Кто хочет, тот ищет возможность, кто не хочет — ищет причину

---

http://igor-borisov.ru

Темы курса

- Погружение в World Wide Web
- Создание CGI-приложений на Ruby
- Создание приложений на Ruby on Rails

Получаем Ruby и Rails

- http://railsinstaller.org/ru-RU

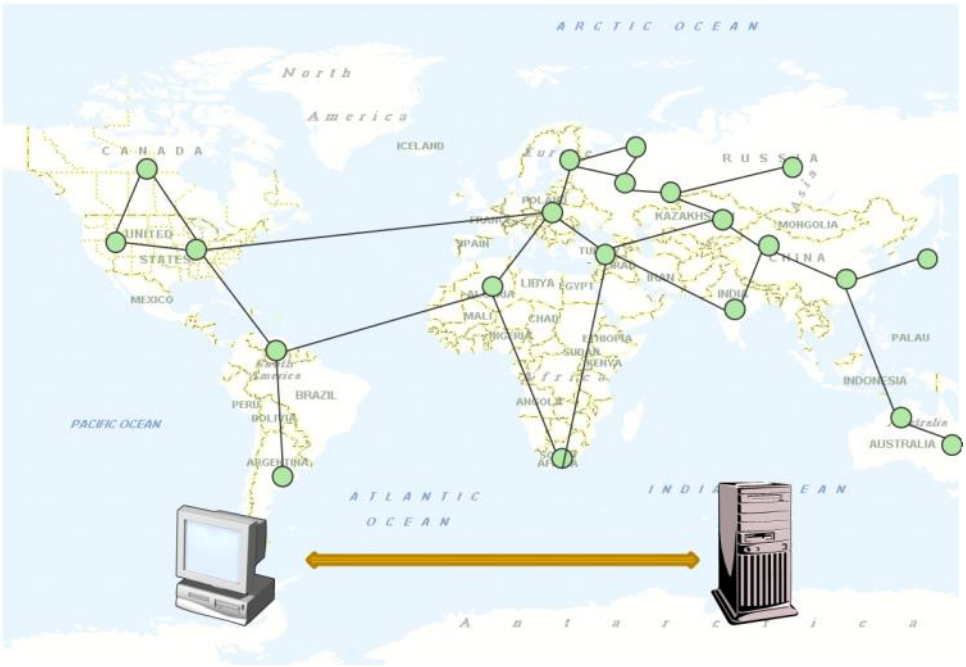
Модуль 1

Ruby on Rails
Погружение в World Wide Web

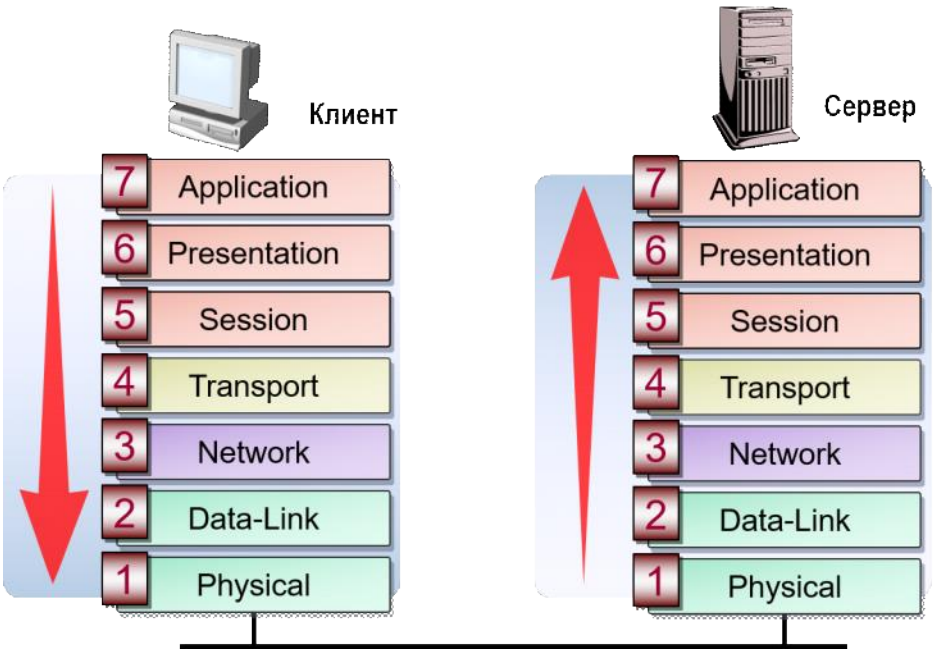
Темы модуля

- Коммуникация в Интернете
- Службы Интернета
- HTTP
- Терминология
- Методы и заголовки запроса клиента
- Статусы и заголовки ответа сервера
- Запуск CGI-приложений

World Wide Web



Сетевые коммуникации



Адресация в сетях

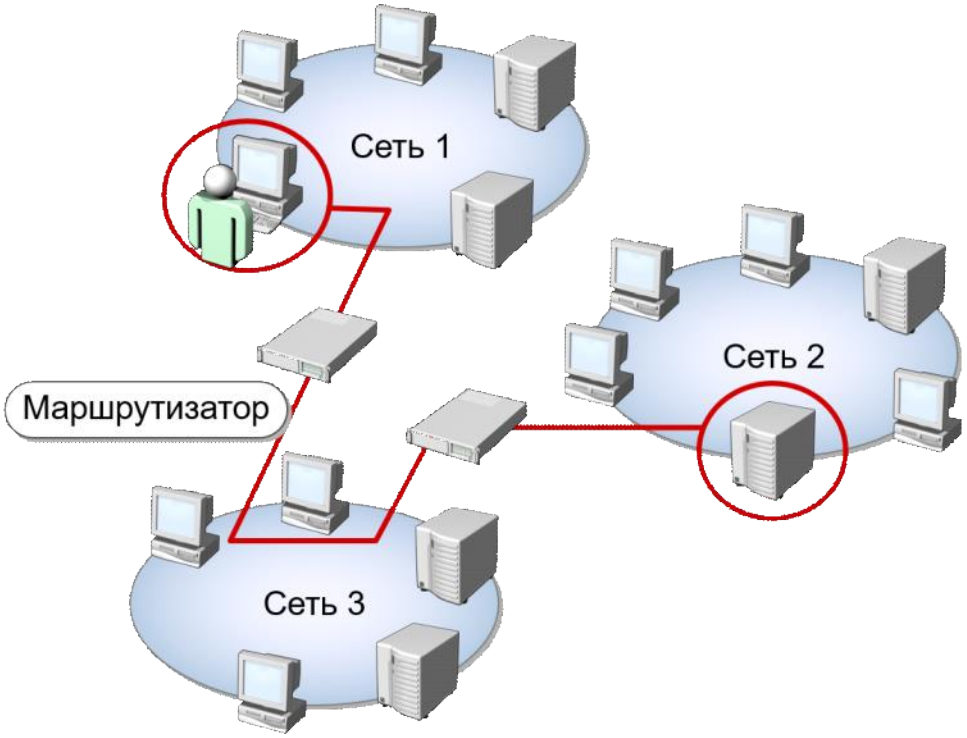
IPv4
213.180.204.8
11010101 10110100 11001100 00001000

Адреса для построения локальных сетей
10.x.x.x
172.16.x.x - 172.31.x.x
192.168.x.x

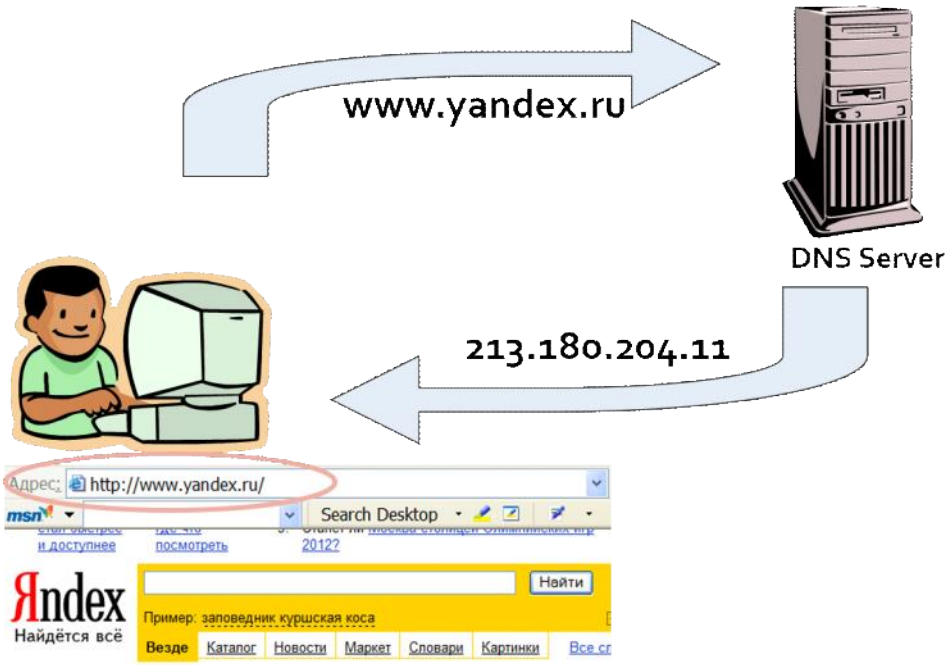
Адреса для внутреннего использования
127.0.0.x

Принцип соединения клиента и сервера
192.168.0.1**:1252** <=> 213.180.204.3**:80**

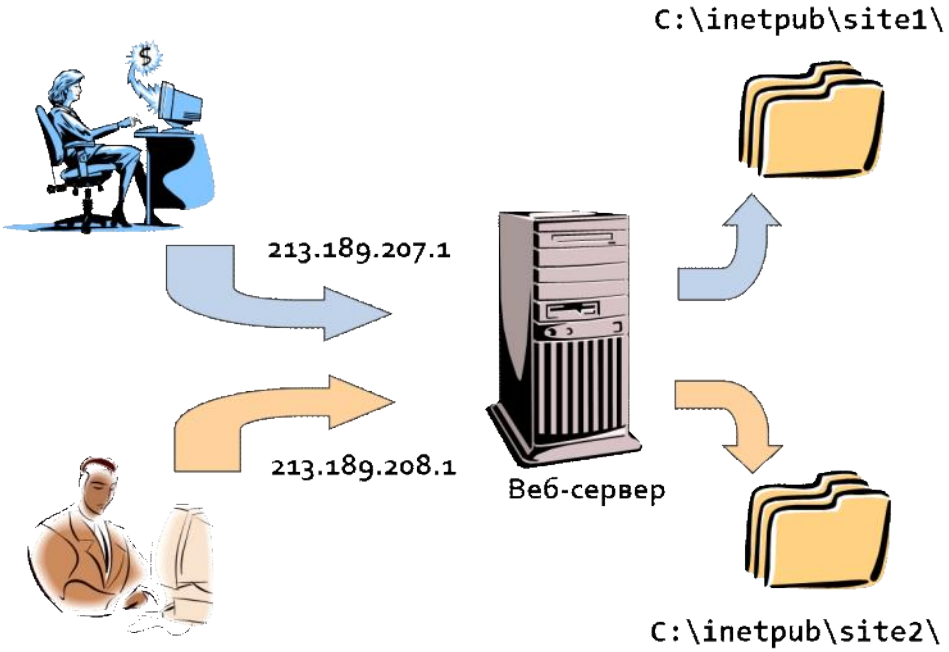
Маршрутизация в сети



DNS


www.yandex.ru
DNS Server
213.180.204.11
Адрес: http://www.yandex.ru/
Search Desktop
Найти
Пример: заповедник куршская коса
Везде
Каталог
Новости
Маркет
Словари
Картинки

Виртуальный хост



HyperText Transfer Protocol

```
GET /folder/index.html HTTP/1.1↵
Host: mysite.ru↵
User-Agent: Mozilla/5.0 ...↵
Accept: */*↵
Accept-Language: ru,en-us↵
Keep-Alive: 300↵
Connection: keep-alive↵
↵

HTTP/1.1 200 OK↵
Date: Sun, 17 Aug 2008 07:47:24 GMT↵
Server: Microsoft-IIS/6.0↵
Last-Modified: Mon, 07 Jul 2008 11:07:04 GMT↵
Content-Length: 34234↵
Content-Type: text/html↵
↵
<html>. . .</html>
```

Статусы ответа сервера

- 1xx – Информационное сообщение
- 2xx – Успешное выполнение команды
  - 200 OK
- 3xx – Переадресация
  - 301 Moved Permanently
  - 302 Found
  - 304 Not Modified
- 4xx – Ошибка на стороне клиента
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
  - 410 Gone
- 5xx – Ошибка на стороне сервера
  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Service Unavailable
  - 504 Gateway Timeout

Запрос с валидатором

```
HTTP/1.1 200 OK⏎
...⏎
Last-Modified: Thu, 24 Jul 2008 13:56:14⏎

GET /folder/index.html HTTP/1.1⏎
...⏎
If-Modified-Since: Thu, 24 Jul 2008 13:56:14⏎
⏎

HTTP/1.1 304 Not Modified⏎
...⏎
```

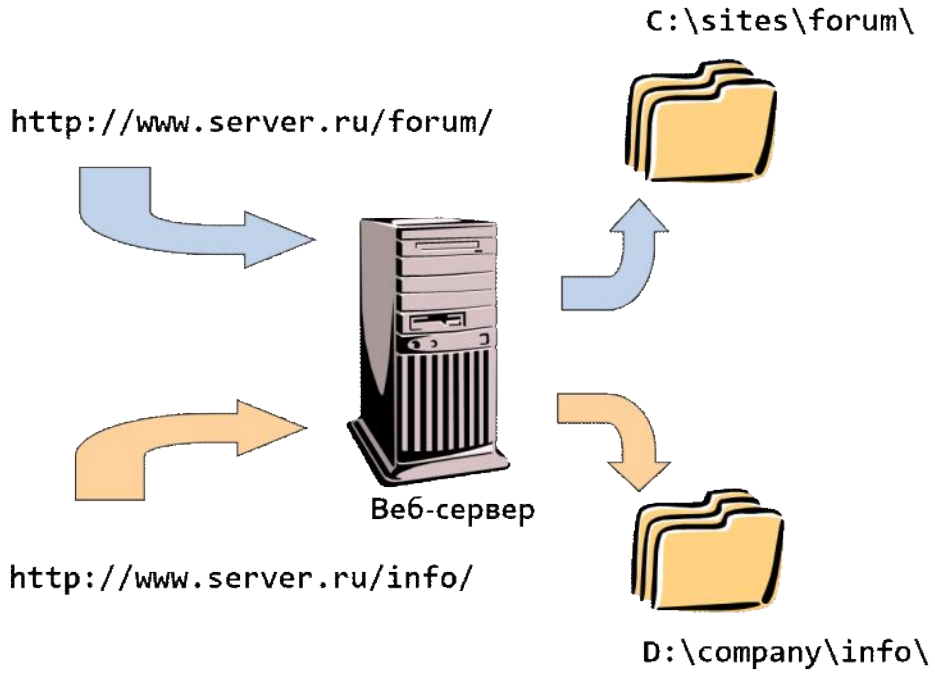
Веб-сервер Apache и терминал

```
# Запуск/остановка
$ httpd
$ Ctrl + C
$ Ctrl + Break

# conf\httpd.conf
# Подключенние модуля заголовков
LoadModule headers_module modules/mod_headers.so

# .htaccess
Options Indexes
DirectoryIndex index.html
```

Виртуальная директория



Запуск CGI приложений

```
# CGI-файл
#!ruby
puts "Content-Type: text/html"
puts ""
puts "<h1>Hello, world</h1>"

# Конфигурация сервера
ScriptAlias /cgi-bin/ "c:/Apache24/cgi-bin"

<Directory "c:/Apache24/cgi-bin">
    AllowOverride All
    Options Indexes
    Require all granted
</Directory>

AddHandler cgi-script .cgi .rb

# Убираем проблемы с кодировкой на сервере
puts "Content-Type: text/html;charset=utf-8"

# Убираем кракозябры в браузере (.htaccess)
AddDefaultCharset utf-8

# Доступ к переменным окружения сервера в CGI скрипте
puts ENV["HTTP_USER_AGENT"]
```

Передача параметров методом GET

```
<form action="handler.rb" method="GET">
   Имя: <input name="name" type="text">
   Возраст: <input name="age" type="text">
   <input type="submit">
</form>
```

```
GET /handler.rb?name=Vasya&age=25 HTTP/1.1
```

```
# Простой доступ к параметрам в CGI-скрипте
require "cgi"
c = CGI.new
name = c["name"]
age = c["age"]
puts "<p>Ваше имя: #{name}<br>"
puts "Ваш возраст: #{age}</p>"
```

```
# Перебор параметров в CGI-скрипте
require "cgi"
c = CGI.new
name = c["name"]
age = c["age"]
method = c.request_method

c.params.each do |key, value|
  puts "#{key} : #{value[0]}<br>"
end
```

Передача параметров методом POST

```
<form action="handler.rb" method="POST">
   Имя: <input name="name" type="text">
   Возраст: <input name="age" type="text">
   <input type="submit">
</form>
```

```
POST /handler.rb HTTP/1.1↵
...↵
Referer: http://mysite.ru/↵
Content-Type: application/x-www-form-
urlencoded↵
Content-Length: 17↵
↵
name=Vasya&age=25↵
↵
```

```
# Простой доступ к параметрам в CGI-скрипте
require "cgi"
c = CGI.new
name = c["name"]
age = c["age"]
puts "<p>Ваше имя: #{name}<br>"
puts "Ваш возраст: #{age}</p>"
```

```
# Перебор параметров в CGI-скрипте
require "cgi"
c = CGI.new
name = c["name"]
age = c["age"]
method = c.request_method

c.params.each do |key, value|
  puts "#{key} : #{value[0]}<br>"
```

```
end
```

Cookies

```
GET /folder/handler.rb?name=Vasya HTTP/1.1⏎
...⏎
⏎

HTTP/1.1 200 OK⏎
Server: Microsoft IIS 6⏎
...⏎
Set-Cookie: userName=Vasya;path=/⏎
⏎

GET /folder/info.html HTTP/1.1⏎
...⏎
Cookie: userName=Vasya⏎
⏎
```

```ruby
# Ставим
puts "Set-Cookie: name=#{name};path=/"
# Читаем
puts ENV["HTTP_COOKIE"]
```

Что мы изучили?

- Уяснили принципы WWW
- Познакомились с HTTP - основным протоколом WWW
- Познакомились с базовым функционалом веб-серверов
- Узнали, как гибко управлять браузером, посылая нужные заголовки ответа
- Принимать и обрабатывать данные переданные через адресную строку
- Принимать и обрабатывать данные из веб-формы
- Познакомились с cookie

Модуль 2

Ruby on Rails
Знакомство с фреймворком Rails

Темы модуля

- Знакомимся с Rails
- Установка Rails
- Создание rails-приложения
- Использование Rails-сервера
- Принципы Ruby on Rails
- Архитектура Rails
- Структура rails-приложения
- Маршрутизация
- Создание контроллера
- Изменение представления
- Использование rails-консоли

Установка Rails

- Windows OS
  - http://railsinstaller.org/ru-RU
- Другие OS
  - `$ gem install rails`
- `$ rails -v`

Создание и запуск rails-приложения

- Создание приложения
  - `$ rails new rails-app`

- Запуск сервера
  - `$ bin\rails server`
  - `$ bin\rails s`
  - `$ bin\rails s -b "127.0.0.1" -p "80"`

Лабораторная работа 2.1

Создание приложения и запуск сервера

1. Проверьте версии установленных программ последовательно набрав в терминале следующие команды:
   ```
   $ ruby -v
   $ rails -v
   ```
2. Создайте новое Rails приложение **blog** выполнив в терминале команду:
   ```
   $ rails new blog
   ```
3. Перейдите в директорию **blog\bin** выполнив в терминале команду:
   ```
   $ cd blog\bin
   ```

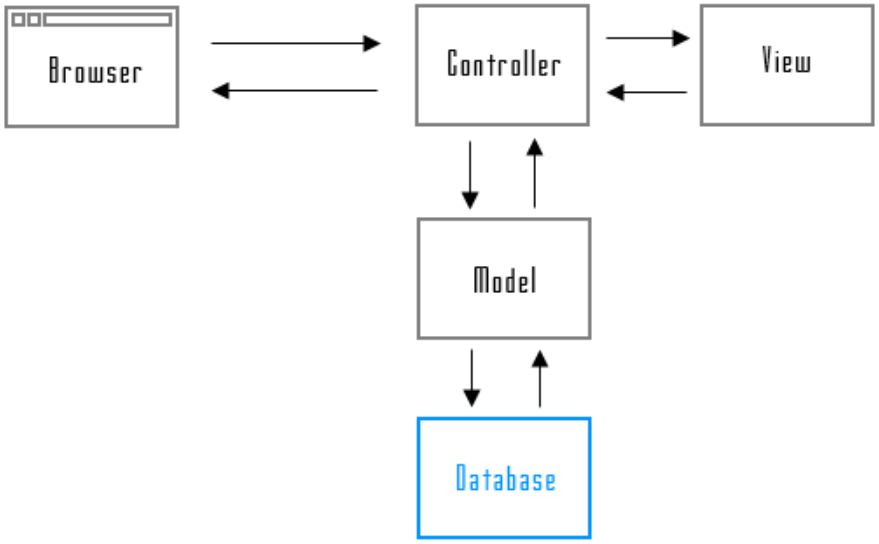
   ВНИМАНИЕ! С этого момента на протяжении всего курса все команды в терминале выполняются в директории **blog**, если прямо не указана какая-либо другая директория.
4. Запустите rails сервер выполнив в терминале команду:
   ```
   $ rails s -p "80"
   ```
5. Откройте браузер н наберите в адресной строке адрес вашего сайта: http://localhost. Вы должны увидеть страницу приветствия Ruby on Rails.
6. Вы можете завершить работу сервера сочетанием клавиш **Ctrl+C (Ctrl+Break)**

Принципы Ruby on Rails

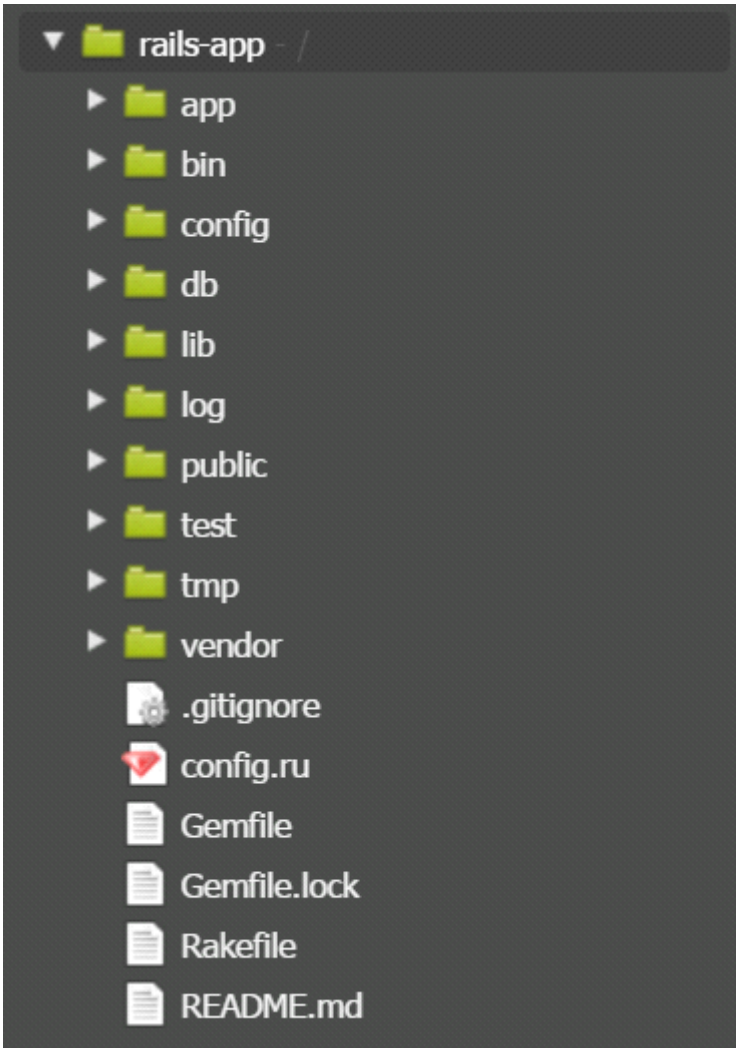
- Convention over Configuration
- Don't Repeat Yourself (DRY)

Архитектура Rails

- **Model-View-Controller**



Структура rails-приложения



Команды Rails

- gem
- bundle
  - $ bin\bundle list
- rake
  - $ bin\rake --tasks
- rails
  - $ bin\rails

Первые шаги

- Смотрим маршрутизацию
  - $ bin\rake routes
  - $ config\routes.rb
    - root "learning#index"
- Создаём контроллер
  - $ bin\rails g **controller** learning index
- Изменяем представление
  - app/views/learning/index.html.rb
- Используем cURL
  - $ curl -v http://...
- Где находится контроллер?
  - app/controllers/learning_controller.rb
- Используем rails-консоль
  - $ bin\rails c
    - > app.get "/"

Лабораторная работа 2.2

Создание домашней страницы приложения

ВНИМАНИЕ! С этого момента на протяжении всего курса все команды типа
`$ папка\команда`
означают, что надо войти в директорию "папка" и в ней выполнить команду "команда".

1. Набертие в терминале следующую команду:
   `$ bin\rake routes`

   Убедитесь, что список маршрутов пуст
2. Откройте файл **config/routes.rb** и впишите в блок **do**:
   `root "welcome#index"`
3. Набертие в терминале:
   `$ bin\rake routes`

   Убедитесь, что появился новый маршрут
4. Перегрузите страницу в браузере. Вы должны получить страницу ошибки "Routing Error"
5. Создайте необходимый контроллер выполнив в терминале команду:
   `$ bin\rails g controller welcome index`
6. Перегрузите страницу в браузере. Вы должны увидеть обновлённую страницу
7. Откройте файл **app/views/welcome/index.html.rb** и поменяейте его содержимое на:
   `<h1>Привет всем!</h1>`

   Не забудьте преобразовать кодировку файла в UTF-8 без BOM
8. Перегрузите страницу в браузере. Вы должны увидеть страницу с новым содержимым
9. Откройте новое окно терминала и выполните команду:
   $ curl -v https://localhost

   Изучите полученный ответ сервера

10. Откройте файл **app/controllers/welcome_controller.rb** и найдите метод **index**

11. В терминале, где запущен сервер, найдите запрос к корню сайта и изучите его

12. Откройте новое окно терминала и запустите rails консоль:
    ```
    $ bin\rails c
    ```

13. Сэмулируйте запрос браузера к серверу набрав:
    ```
    > app.get "/"
    ```

14. В браузере перейдите по адресу **/welcome/index**. Объясните, почему вы видите тоже содержимое, что и при обращении к корню сайта?

15. Откройте файл **config/routes.rb** и удалите строку:
    ```
    get "welcome/index"
    ```

16. Перегрузите страницу в браузере. Вы должны получить страницу ошибки "Routing Error"

Что мы изучили?

- Установили Rails
- Создали простое rails-приложение
- Запустили Rails-сервер
- Познакомились с основными концепциями Ruby on Rails
- Создали контроллер
- Увидели, как взаимодействуют контроллер и представление при обработке запроса

Модуль 3

Ruby on Rails MVC

Темы модуля

- Модель
- Взаимодействие компонентов MVC
- Взаимодействие представления и модели

Модель

- Добавляем ресурсы
  - $ config\routes.rb
    - resourses :courses
    - resourse :course

- Изучаем маршрутизацию
  - $ bin\rake routes
    - create - создание нового ресурса
    - update - изменение ресурса
    - delete - удаление ресурса
    - show - отображение ресурса
    - index - отображение списка ресурсов

- Создание модели
  - $ bin\rails g **model** course title:string summary:string

- Используем миграцию
  - db/migrate/..._create_courses.rb
  - $ bin\rake db:migrate

- Используем rails-консоль базы данных
  - $ bin\rails db
    - > select * from courses;
    - > select count(*) from courses;

- Используем rails-консоль
  - $ bin\rails c
    - > Course.count
    - > Course.create(title: "...", summary: "...")
    - > course = Course.new(title: "...", summary: "...")
    - > course.save

Лабораторная работа 3.1

Создание модели

1. Откройте файл **config/routes.rb** и впишите в блок **do**:
   ```
   resourses :articles
   ```
2. Наберите в терминале:
   ```
   $ bin\rake routes
   ```
   Изучите таблицу маршрутов
3. Создайте модель **article** выполнив в терминале команду:
   ```
   $ bin\rails g model article title:string description:string
   ```
4. Откройте файл **db/migrate/..._create_articles.rb** и изучите его
5. Примените миграцию набрав в терминале:
   ```
   $ bin\rake db:migrate
   ```
   Эта команда должна создать таблицу **articles** в базе данных
6. Откройте новое окно терминала и выполните команду:
   ```
   $ bin\rails db
   ```
   Вы должны попасть в интерфейс консоли базы данных sqlite
7. В консоли базы данных наберите:
   ```
   > select * from articles
   > select count(*) from articles
   ```
   Убедитесь, что записи в таблице отсутствуют
8. В rails консоли (`$ bin\rails c`) наберите:
   ```
   > Article.count
   ```
   Вы должны получить число 0 в виде ответа
9. Создайте первую запись в базе данных набрав в rails консоли:
   ```
   > Article.create(title: 'Первая статья', description: 'описание статьи')
   ```
10. Проверьте количество записей в таблице articles в консоли базы данных и в rails консоли
11. Создайте другую запись в базе данных набрав в rails консоли:
    ```
    > article = Article.new(title: 'Вторая статья', description: 'описание статьи')
    ```
12. Проверьте количество записей в таблице articles в консоли базы данных и в rails консоли. Сколько записей вы видите? Почему?
13. Сохраните новую запись в базе данных набрав в rails консоли:
    ```
    > article.save
    ```
14. Проверьте количество записей в таблице articles в консоли базы данных и в rails консоли.

Взаимодействие компонентов MVC

- Добавляем ссылки в представление
  - `<%= link_to "Относительная ссылка", courses_path %>`
  - `<%= link_to "Абсолютная ссылка", courses_url %>`

- Выбираем записи из базы данных в контроллере
  - ```
    def index
    end
    ```
  - `@courses = Course.all`

- Изменяем представление
  - ```
    <% @courses.each do |course| %>
       <%= course.title %>
    <% end %>
    ```

Лабораторная работа 3.2

Взаимодействие компонентов MVC

1. Откройте файл **app/views/welcome/index.html.rb** и добавьте строку:
   `<%= link_to 'Мой блог', ... %>`
2. Наберите в терминале:
   `$ bin\rake routes`

   Найдите маршрут с префиксом **articles** и посмотрите, что указано в колонке **Controller#Action**
3. В файле **app/views/welcome/index.html.rb** в ссылке вместо многоточия напишите:
   `articles_path`
4. Откройте в браузере свой сайт. Вы должны увидеть страницу со ссылкой 'Мой блог'
5. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированную ссылку
6. В **index.html.rb** поменяйте **`articles_path`** на **`articles_url`**, перегрузите страницу в браузере и вновь найдите ссылку в html-коде страницы. Уясните разницу.
7. В браузере нажмите ссылку 'Мой блог'. Вы должны получить страницу ошибки "Routing Error"
8. Создайте контроллер **articles**:
   `$ bin\rails g controller articles index`
9. В браузере вернитесь назад на главную страницу и ещё раз нажмите ссылку 'Мой блог'. Вы должны увидеть страницу "Articles#index"
10. Вам надо на эту страницу вывести список статей из базы данных. Откройте файл **app/controllers/articles_controller.rb** и опишите содержимое метода **index** как:
    `@articles = Article.all`
11. Откройте файл **app/views/articles/index.html.rb** и опишите его как:
    ```
    <h1>Мои статьи</h1>
    <% @articles.each do |article| %>
    <%= article.title %>: <%= article.description %>
    <hr>
    <% end %>
    ```
12. Загрузите в браузере страницу по адресу **/articles**. Вы должны увидеть список своих статей
13. Посмотрите в терминале лог сервера. Какой http-метод был использован браузером при запросе? Какой ресурс был отправлен сервером браузеру?

От представления к модели

- Ссылка на создание ресурса
  - `<%= link_to "Создать", new_course_path %>`

- Представление для создания ресурса
  - app/views/courses/new.html.rb
  - ```
    <%= form_for @course do |f| %>
       <%= f.label :title %>
       <%= f.text_field :title %>
       <%= f.submit %>
    <% end %>
    ```
- Создаём метод **new** в контроллере
  - ```
    def new
    end
    ```
  - `@course = Course.new`

- Создаём метод **create** в контроллере
  - ```
    def create
    end
    ```
  - ~~`Course.create(params[:course])`~~
  - `fields = params.require(:course).permit(:title, :summary)`
  - `Course.create(fields)`

- Переадресация запроса
  - `redirect_to courses_index_path`
    ```

Лабораторная работа 3.3

Получение данных от пользователя и их сохранение в базе данных

1. Откройте файл **app/views/articles/index.html.rb** и добавьте строку:
   `<%= link_to 'Создать статью', ... %>`
2. Наберите в терминале:
   `$ bin\rake routes`

   Найдите маршрут с префиксом **new_articles** и посмотрите, что указано в колонке **Controller#Action**
3. В файле **app/views/articles/index.html.rb** в ссылке вместо многоточия напишите:
   `new_article_path`
4. Загрузите в браузере страницу по адресу **/articles**. На странице вы должны увидеть ссылку "Создать статью"
5. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированную ссылку
6. В браузере перейдите по ссылке "Создать статью". Вы должны получить страницу ошибки "Unknown action"
7. Откройте файл **app/controllers/articles_controller.rb** и создайте метод **new** как:
   def new
   end
8. В браузере перезагрузите страницу. Вы должны получить страницу ошибки "Template is missing"
9. В директории **app/views/articles** создайте файл по имени **new.html.rb** и опишите его как:
   ```
   <h1>Создать статью</h1>
   <%= form_for @article do |article| %>
     <div>
       <%= article.label :title %><br>
       <%= article.text_field :title %>
     </div>
     <div>
       <%= article.label :description %><br>
       <%= article.text_area :description %>
     </div>
     <div><%= article.submit %></div>
   <% end %>
   ```
10. В браузере перезагрузите страницу. Вы должны получить страницу ошибки "ArgumentError"
11. Впишите в контроллере в метод **new**:
    `@article = Article.new`
12. В браузере перезагрузите страницу. Вы должны увидеть страницу с формой добавления новой статьи
13. Нажмите сочетание клавиш **Ctrl+U** и изучите код сгенерированной веб-формы. Обратите внимание на http-метод, которым будет отправлена форма. Какой метод какого контроллера будет исполняться при пересылке данных из формы?
14. Наберите в терминале:
    `$ bin\rake routes`

    Найдите нужный маршрут

15. Заполните веб-форму и отправьте её. Вы должны получить страницу ошибки "Unknown action"
16. В файле **app/controllers/articles_controller.rb** создайте метод **create** как:
    def create
    end
17. Ещё раз заполните веб-форму и отправьте её. Вы должны получить страницу ошибки "Template is missing"
18. Впишите в контроллере в метод **create**:
    `Article.create(params[:article])`
19. Ещё раз заполните веб-форму и отправьте её. Вы должны получить страницу ошибки "ForbiddenAttributesError"
20. В файле **app/controllers/articles_controller.rb** измените метод **create** на:
    ```
    fields = params.require(:article).permit(:title, :description)
    Article.create(fields)
    ```
21. Ещё раз заполните веб-форму и отправьте её. Посмотрите в лог сервера. Операция прошла успешно? Какой статус ответа прислал сервер? Почему?
22. В файле **app/controllers/articles_controller.rb** в методе **create** допишите:
    `redirect_to articles_index_path`

    Почему мы использовали именно этот аргумент метода redirect_to? Откуда это понятно?
23. Ещё раз заполните веб-форму и отправьте её. Вы должны быть перенаправлены на страницу со списком статей
24. Проверьте количество записей в таблице articles в консоли базы данных или в rails консоли. Попробуйте эти записи просмотреть

Что мы изучили?

- Рассмотрели архитектуру MVC в действии
  - Создали модель
  - Применили миграцию
  - Экспериментировали с консолью базы данных
  - Проследили цепочку взаимодействия:
    - View --> Controller
    - Controller --> Model
    - Controller --> View
- Увидели, как пользователь посылает данные в базу данных
- Отследили взаимодействие View --> Controller --> Model
- Уяснили, как контроллер выбирает нужное представление
- Увидели для чего используются HTTP методы
- Научились добавлять запись в базу данных
- Научились извлекать записи из базы данных

Модуль 4

Ruby on Rails
Работа с ресурсами

Темы модуля

- Обновление ресурса
- Отображение ресурса
- Удаление ресурса
- Дополнительные элементы представлений

Обновление ресурса

- Ссылка на изменение ресурса
  - `<%= link_to "Изменить", edit_course_path(course) %>`

- Создаём метод **edit** в контроллере
  - `def edit`
    `end`
  - `@course = Course.find(params[:id])`

- Создаём метод **update** в контроллере
  - `def update`
    `end`
  - `@course = Course.find(params[:id])`
  - ~~`@course.update_attributes(params[:course])`~~
  - `fields = params.require(:course).permit(:title, :summary))`
  - `@course.update_attributes(fields)`

- Представление для создания ресурса **edit.html.rb**
  - ```
    <%= form_for @course do |f| %>
        <%= f.label :title %>
        <%= f.text_field :title %>
        <%= f.submit %>
    <% end %>
    ```
    ```

Лабораторная работа 4.1

Обновление ресурса

1. Откройте файл **app/views/articles/index.html.rb** и добавьте в цикл строку:
   `<%= link_to 'Редактировать', ... %>`
2. Наберите в терминале:
   `$ bin\rake routes`

   Найдите нужный вам маршрут
3. В файле **app/views/articles/index.html.rb** в ссылке вместо многоточия напишите:
   `edit_article_path()`
4. В rails консоли напишите:
   > app.edit_article_path()

   Вы должны получить ошибку "No route matches"
5. Ещё раз просмотрите маршрут с префиксом edit_article. Обратите внимание на колонку "URI Pattern"
6. В файле **index.html.rb** добавьте в метод `edit_article_path()` параметр **`article`**
7. В rails консоли напишите:
   > app.edit_article_path(1)

   Вы должны увидеть сгенерированный путь для первой статьи
8. Загрузите в браузере страницу по адресу **/articles**. На странице вы должны увидеть ссылки "Редактировать"
9. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированные ссылки
10. В браузере нажмите по любой ссылке "Редактировать". Вы должны получить страницу ошибки "Unknown action"
11. Откройте файл **app/controllers/articles_controller.rb** и создайте метод **edit** как:
    ```
    def edit
    end
    ```
12. В браузере перезагрузите страницу. Вы должны получить страницу ошибки "Template is missing"
13. В директории **app/views/articles** создайте файл по имени **edit.html.rb** и скопируйте в него содержимое файла **app/views/articles/new.html.rb**. Не забудьте поменять заголовок страницы
14. В браузере перезагрузите страницу. Вы должны получить страницу ошибки "ArgumentError"
15. Посмотрите в лог сервера. Вы увидите, что Rails автоматически сгенерировал в качестве параметра хэш с идентификатором статьи
16. Впишите в контроллере в метод **edit**:
    `@article = Article.find(params[:id])`
17. В браузере перезагрузите страницу. Вы должны увидеть страницу с формой редактирования статьи
18. Нажмите сочетание клавиш **Ctrl+U** и изучите код сгенерированной веб-формы. Обратите внимание на http-метод, которым будет отправлена форма. Какой метод какого контроллера будет исполняться при

пересылке данных из формы? Обратите внимание на скрытое поле формы по имени _method. Какой http-метод будет исполняться? На какой http-метод будет реагировать Rails?

19. Наберите в терминале:
```
$ bin\rake routes

Найдите нужный маршрут
```

20. Заполните веб-форму и отправьте её. Вы должны получить страницу ошибки "Unknown action"

21. В файле **app/controllers/articles_controller.rb** создайте метод **create** как:
```
def update
  @article = Article.find(params[:id])
  fields = params.require(:article).permit(:title, :description)
  @article.update_attributes(fields)
end
```

22. Ещё раз заполните веб-форму и отправьте её. Посмотрите в лог сервера. Запись должна обновиться (обратите внимание на статус ответа сервера), но в браузере мы остались на той же странице

23. Допишите в контроллере внизу метода **update**:
```
redirect_to articles_path
```

24. В браузере вернитесь назад, ещё раз заполните веб-форму и отправьте её. Вы должны вернуться на страницу со списком статей

25. Откройте файл **config/routes.rb** и удалите строку `get "articles/index"`:

26. В файле **app/controllers/articles_controller.rb** внизу метода **create** допишите:
```
redirect_to articles_path
```

Отображение ресурса

- Ссылка на ресурс
  - `<%= link_to "Показать", course_path(course) %>`

- Создаём метод **show** в контроллере
  - `def show`
    `end`
  - `@course = Course.find(params[:id])`

- Создаём представление **show.html.rb**
  - `<p><%= @course.title %></p>`
  - `<p><%= @course.summary %></p>`

Лабораторная работа 4.2

Отображение ресурса

1. Откройте файл **app/views/articles/index.html.rb** и добавьте в цикл строку:
   `<%= link_to 'Показать', ... %>`
2. Наберите в терминале:
   `$ bin\rake routes`

   Найдите нужный вам маршрут
3. В файле **app/views/articles/index.html.rb** в ссылке вместо многоточия напишите:
   `article_path()`
4. В rails консоли напишите:
   > app.`article_path`

   Вы должны получить ошибку "No route matches"
5. Ещё раз просмотрите маршрут с префиксом article. Обратите внимание на колонку "URI Pattern"
6. В файле **index.html.rb** добавьте в метод `article_path()` параметр **`article.id`**
7. В rails консоли напишите:
   > app.`article_path(article.first.id)`
   > app.`article_path(Article.first)`

   В обоих случаях вы должны увидеть сгенерированный путь для первой статьи
8. В файле **index.html.rb** поменяйте параметр метода `article_path()` на **`article`**
9. Загрузите в браузере страницу по адресу **/articles**. На странице вы должны увидеть ссылки "Показать"
10. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированные ссылки
11. В браузере нажмите по любой ссылке "Показать". Вы должны получить страницу ошибки "Unknown action"
12. Откройте файл **app/controllers/articles_controller.rb** и создайте метод **show** как:
```
def show
  @article = Article.find(params[:id])
end
```
13. В браузере перезагрузите страницу. Вы должны получить страницу ошибки "Template is missing"
14. В директории **app/views/articles** создайте файл по имени **show.html.rb** и опишите его как:
```
<p><%= @article.title %></p>
<p><%= @article.description %></p>
```
15. В браузере перезагрузите страницу. Вы должны описание статьи

Удаление ресурса

- Ссылка на удаление ресурса
  - `<%= link_to "Удалить", course_path(course), method: :delete, data: {confirm: "Вы уверены?"} %>`
- Создаём метод **destroy** в контроллере
  - `def destroy`
    `end`
  - `@course = Course.find(params[:id])`
  - `@course.destroy`
  - `redirect_to course_path, notice: "Удалено!"`
- Добавляем вывод сообщения
  - app/views/**layouts**/application.html.rb
  - `<% flash.each do |type, message| %>`
    `<%= content_tag :div, message, class: type %>`
    `<% end %>`

Лабораторная работа 4.3

Удаление ресурса

1. Наберите в терминале:
   ```
   $ bin\rake routes
   ```

   Найдите нужный вам маршрут

2. Откройте файл **app/views/articles/index.html.rb** и добавьте в цикл строку:
   ```
   <%= link_to 'Удалить', article_path(article), method: :delete,
                                                 data: {confirm: 'Вы уверены?'} %>
   ```

3. Загрузите в браузере страницу по адресу **/articles**. На странице вы должны увидеть ссылки "Удалить"

4. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированные ссылки. Обратите внимание на атрибут data-method. Какой http-метод будет исполняться? На какой http-метод будет реагировать Rails?

5. В браузере нажмите по любой ссылке "Удалить". Должно выскочить окошко с вашим вопросом. Нажмите кнопку "Cancel" ("Отменить")

6. Откройте файл **app/controllers/articles_controller.rb** и создайте метод **destroy** как:
   ```
   def destroy
     @article = Article.find(params[:id])
     @article.destroy
     redirect_to articles_path
   end
   ```

7. В браузере выполните удаление статьи. Посмотрите в лог сервера. Запись должна удалиться

8. Допишите в контроллере в методе **destroy** второй параметр для метода redirect_to:
   notice: "Статья удалена"

9. Откройте файл **app/views/layouts/application.html.rb** и добавьте после открывающего тега **body**:
   ```
   <% flash.each do |type, msg| %>
     <%= content_tag :div, msg, class: type %>
   <% end %>
   ```

10. В директории **app/views/articles** создайте файл по имени **edit.html.rb** и скопируйте в него содержимое файла **app/views/articles/new.html.rb**. Не забудьте поменять заголовок страницы

11. В браузере выполните удаление статьи. Если статей не хватает, то добавьте несколько новых. После удаления статьи вы должны увидеть в верхней части страницы текст "Статья удалена"

12. Попробуйте создать css-класс **notice**, чтобы как-либо оформить сообщение об успешном удалении. Стили добавляются в файле **app/assets/stylesheets/application.css**

Улучшаем шаблоны

- Создаём частичный шаблон
  - app/views/courses/_form.html.rb
- Используем частичный шаблон в представлении
  - app/views/courses/_form.html.rb
  - <%= render "form" %>

Лабораторная работа 4.4

Удаление дубликатов в представлении

1. Откройте файлы **app/views/articles/new.html.rb** и **app/views/articles/edit.html.rb**
2. Сравните содержимое файлов. Что бросается в глаза? Какие можно сделать выводы?
3. В директории **app/views/articles** создайте файл по имени **_form.html.rb** и скопируйте в него код генерации веб-формы из файла **app/views/articles/new.html.rb**
4. В файле **app/views/articles/new.html.rb** удалите код генерации веб-формы и на его месте напишите: `<%= render 'form' %>`
5. В браузере загрузите страницу по адресу **/articles**. Нажмите на ссылки "Создать статью" и "Редактировать". Убедитесь, что веб-формы отображаются корректно

Что мы изучили?

- Полностью рассмотрели CRUD-функционал для работы с ресурсами:
  - Научились обновлять запись в базе данных
  - Научились обновлять запись в базе данных
  - Научились показывать запись из базы данных
  - Научились удалять запись из базы данных
- Научились отправлять дополнительные сообщения для обратной связи с пользователем
- Научились разбивать шаблоны на подшаблоны с целью избежания дублирования кода в представлениях

Модуль 5

Ruby on Rails
Связи между моделями

Темы модуля

- Использование нескольких моделей в приложении
- Создание связей между моделями

Создание связей между моделями

- Создание модели
  - `$ bin\rails g model file name:string description:string`
- Связи
  - app/models/file.rb
    - `belongs_to: :course`
  - app/models/course.rb
    - `has_many: :files`
- Добавляем ресурсы
  - config/routes.rb
    - ```
      resourses :courses do
        resourses :files
      end
      ```
- Модификация представления для формы
  - `<%= form_for ([@course, @course.files.build]) do |f| %>`
- Создаём метод **create** в контроллере
  - ```
    def create
    end
    ```
  - `@course = Course.find(params[:course_id])`
  - `fields = params[:file].permit(:name, :description))`
  - `@file = course.files.create(fields)`
- Модификация представления для показа
  - `<%= @course.files.each do |file| %>`

Лабораторная работа 5.1

Использоание связей между моделями

1. Создайте модель **comment** выполнив в терминале команду:
   ```
   $ bin\rails g model comment author:string message:string
   ```
2. Откройте последний по дате файл **db/migrate/..._create_articles.rb** и изучите его
3. Примените миграцию набрав в терминале:
   ```
   $ bin\rake db:migrate
   ```
   Эта команда должна создать таблицу **comments** в базе данных
4. Откройте новое окно терминала и выполните команду:
   $ bin\rails db

   Вы должны попасть в интерфейс консоли базы данных sqlite
5. В консоли базы данных наберите:
   ```
   > .tables
   ```
   Убедитесь, что таблица создана
6. Откройте файл **app/models/comment.rb**. Вы должны увидеть строку:
   ```
   belongs_to :articles
   ```
7. Откройте файл **app/models/articles.rb**. Добавьте связь с моделью комментариев:
   ```
   has_many :comments
   ```
8. Откройте файл **config/routes.rb** и перепишите указание использование ресурсов как:
   ```
   resourses :articles do
     resourses :comments
   end
   ```
9. Создайте контроллер **comments** выполнив в терминале команду:
   ```
   $ bin\rails g controller comments
   ```
10. Откройте файл **app/views/articles/show.html.rb** и добавьте в него следующее содержимое:
    ```
    <h2>Комментировать</h2>
    <%= form_for([@article, @article.comments.build]) do |f| %>
      <div>
        <%= f.label :author %><br>
        <%= f.text_field :author %>
      </div>
      <div>
        <%= f.label :message %><br>
        <%= f.text_area :message %>
      </div>
      <div><%= f.submit %></div>
    <% end %>
    ```
11. В браузере загрузите страницу по адресу **/articles** и нажмите на ссылку "Показать". Вы должны увидеть сгенерированную веб-форму для добавления комментариев
12. Нажмите сочетание клавиш **Ctrl+U** и найдите в коде страницы сгенерированные ссылки

13. Наберите в терминале:
```
$ bin\rake routes
```

    Посмотрите на новые маршруты. Найдите маршрут, который используется при создании комментария

14. В браузере заполните веб-форму комментария и отправьте её. Вы должны получить страницу ошибки "Unknown action"

15. В файле **app/controllers/comments_controller.rb** создайте метод **create** как:
```
def create
end
```

16. Ещё раз отправьте веб-форму. Посмотрите в лог сервера. Вы должны увидеть сформированные данные для записи

17. В файле **app/controllers/comments_controller.rb** опишите метод **create** как:
```
@article = Article.find(params[:id])
fields = params[:comment].permit(:author, :message)
@comment = @article.comments.create(fields)
redirect_to article_path(@article)
```

18. В файле **app/views/articles/show.html.rb** добавьте следующее содержимое:
```
<h2>Комментарии</h2>
<%= @article.comments.each do |comment| %>
  <div>
    <%= comment.author %>: 
    <%= comment.message %>
  </div>
<% end %>
```

19. В браузере заполните веб-форму комментария и отправьте её. Вы должны на страницу статьи и увидеть опубликованный коменнтарий

Удаление связанных записей

- Ссылка на удаление ресурса
  - `<%= link_to "Удалить", course_file_path(@course, file), method: :delete, data: {confirm: "Вы уверены?"} %>`

- Создаём метод **destroy** в контроллере
  - `def desytoy`
    `end`
  - `@course = Course.find(params[:course_id])`
  - `@file = @course.files.find(params[:id])`
  - `@file.destroy`

- Жёсткая связь при удалении записей
  - app/models/course.rb
    - `has_many: :files, dependent: :destroy`

Лабораторная работа 5.2

Удаление связанных записей

1. Откройте файл **app/views/articles/show.html.rb** и ссылки на удаление комментария
2. Наберите в терминале:
   ```
   $ bin\rake routes | grep comments
   ```
   Найдите маршрут, который используется при удалении комментария
3. Укажите в качестве второго параметра метода **link_to**:
   ```
   article_comment_path(@article, comment)
   ```
4. Укажите в качестве третьего параметра метода **link_to**:
   ```
   method: :delete
   ```
5. Укажите в качестве четвёртого параметра метода **link_to** указание на показ окна предупреждения (см. Лабораторная работа 4.3 пункт 2)
6. В файле **app/controllers/comments_controller.rb** создайте метод **destroy** как:
   ```
   def destroy
     @article = Article.find(params[:article.id])
     @comment = @article.comments.find(params[:id])
     @comment.destroy
     redirect_to article_path(@article)
   end
   ```
7. В браузере удалите какой-либо комментарий какой-либо статьи. Убедитесь, что при этом не происходит никаких ошибок. Просмотрите таблицу comments в консоли базы данных и убедитесь, что удалённая запись в таблице отсутствует
8. В браузере создайте несколько комментариев в какой-либо статье. Удалите *эту статью*.
9. Просмотрите таблицу comments в консоли базы данных. Что произошло?
10. Откройте файл **модели Article** и допишите второй параметр метода **has_many** как:
    ```
    dependent: :destroy
    ```
11. В браузере ещё раз создайте несколько комментариев в какой-либо статье. Удалите *эту статью*.
12. Просмотрите таблицу comments в консоли базы данных. Убедитесь, что при удалении статьи удаляются все её комментарии

Что мы изучили?

- Познали зависимости между моделями
- Научились создавать связи между моделями

Модуль 6

Ruby on Rails
Улучшаем веб-приложение

Темы модуля

- Валидация данных
- Фильтры контроллера
- Множественное представление ресурсов
- Использование базовой аутентификации
- Локализация приложения

Валидация данных

- Добавляем ограничения в модель
  - `validates :title, presence: true`

- Изменяем представление

```
<% if @course.errors.any? %>
   <ul>
     <%= @course.errors.full_messages do |msg| %>
       <%= msg %>
     <% end %>
   </ul>
<% end %>
```

- Изменяем метод контроллера
  -
```
if @course.errors.any?
  render :new
else
  redirect_to courses_path
end
```

Лабораторная работа 6.1

Валидация данных

1. В браузере создайте новую статью оставив поля веб-формы пустыми. Вы должны увидеть, что статья успешно была добавлена в базу данных
2. Откройте файл **модели Article** и добавьте следующий код:
```
validates :title, presence: true
validates :description, presence: true
```
3. В браузере ещё раз создайте новую статью оставив поля веб-формы пустыми. Вы должны увидеть в базу данных ничего не добавилось
4. Откройте файл **app/views/articles/_form.html.rb** и внутри блока form_for допишите:
```
<% if @article.errors.any? %>
  <% @articles.errors.full_messages.each do |msg| %>
    <li><%= msg %></li>
  <% end %>
<% end %>
```
5. Откройте файл **контроллера ArticlesController** и измените метод **create** так, чтобы он выглядел как:
```
fields = params.require(:article).permit(:title, :description)
@article = Article.create(fields)
if @article.error.any?
  render :new
else
  redirect_to articles_path
end
```
6. В браузере ещё раз создайте новую статью оставив поля веб-формы пустыми. Вы должны увидеть сообщения о том, что поля необходимо заполнить

Фильтры контроллера

- Описываем фильтр
  - ```
    def get_course
      @course = Course.find(params[:id])
    end
    ```
- Добавляем фильтр
  - ```
    before_action :get_course, except[:new, :create, :index]
    ```

Лабораторная работа 6.2

Избежание дублирования кода с помощью фильтров

1. Откройте файл **контроллера ArticlesController** и добавьте метод **get_article** со следующим содержимым: @article = `Article.find(params[:id])`
2. В этом же файле созданный метод в качестве фильтра для всех action-методов как: `before_action :get_article, except: [:new, :create, :index]`
3. Во методах контроллера **edit**, **update**, **show** и **destroy** *удалите* дублирующуюся строку @article = `Article.find(params[:id])`
4. Сделайте метод **get_article** закрытым (private)
5. В браузере отредактируйте или удалите какую-либо статью. Вы должны увидеть, что код работает без изменений

Множественное представление ресурсов

- Gemfile
  - gem 'responders'
  - bundle install (bundle update)
-
- Изменение контроллера
  - respond_to :json, :html
- Изменение метода **index**
  - respond_with(@courses)
- Запрос ресурса
  - http://.../courses.**xml**
  - http://.../courses.**html**

Лабораторная работа 6.3

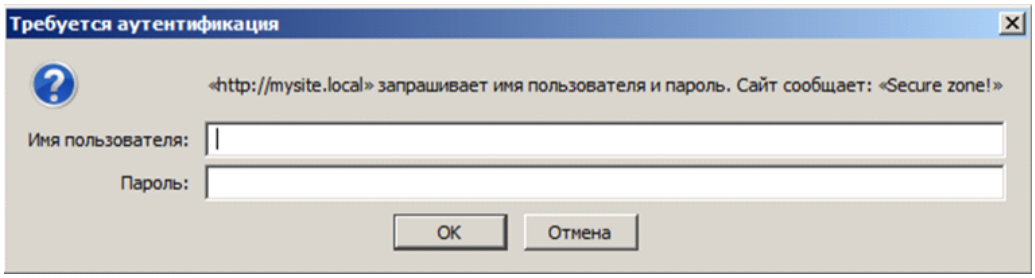
Множественное представление ресурсов

1. Откройте файл Gemfile и добавьте в него строку:
   gem 'responders'
2. Наберите в терминале:
   bundle update
3. Откройте файл **контроллера ArticlesController** и добавьте следующую строку:
   respond_to :json
4. В метод контроллера **index** в самом низу допишите:
   respond_with(@articles)
5. В браузере загрузите страницу **/articles.json**. Вы должны увидеть содержимое страницы в xml-формате
6. В браузере загрузите страницу **/articles**. Вы должны получить ошибку "UnknownFormat" - Rails больше не распознаёт этот запрос
7. Наберите в терминале:
   ```
   $ bin\rake routes

   Внимательно посмотрите на колонку URI Pattern. Обратите внимание на шаблон запроса:
   /articles(.:format)
   ```
8. Добавьте в контроллере второй параметр **:html** для метода **respond_to**
9. В браузере загрузите страницу **/articles.html**. Вы должны увидеть содержимое страницы в html-формате
10. Попробуйте добавить вывод данных в **json**-формате. Используйте для этого символ **:json** в качестве параметра метода **respond_to**

Базовая аутентификация

- Принцип работы
  - HTTP/1.x 401 Unauthorized
    ...
    WWW-Authenticate: Basic realm="www.mysite.local"



  - GET / HTTP/1.1
    ...
    Authorization: Basic QWRtaW5pc3RyYXRvcjpwYXNzQHdvcmQx

Administrator:pass@word1

- Определение аутентификации в контроллере
  - http_basic_authenticate_with name: "admin",
    password: "12345",
    except: [:index, :show]

Лабораторная работа 6.4

Использование базовой аутентификации для доступа к ресурсам

1. Откройте файл **контроллера ArticlesController** и добавьте следующий код:
```
http_basic_authenticate_with name: "admin",
password: "pass@word1",
except: [:index, :show]
```

2. В браузере попробуйте отредактировать или удалить какую-либо статью. Вы должны получить окно с предложением ввести логин и пароль для исполнения данной операции
3. Введите неправильную пару логин/пароль. Что произошло?
4. Нажмите кнопку отмены. Что произошло? Посмотрите ответ сервера. Какой статус ответа возвращается?
5. Перегрузите страницу. Введите правильную пару логин/пароль. Ваша операция должна успешно исполниться

Локализация приложения

- https://github.com/svenfuchs/rails-i18n/blob/master/rails/locale/ru.yml
- Добавление пакета локализации в Gemfile
  - `gem "russian" "~> 0.6.0"`
- Обновление пакетов
  - `bundle update`
- Изменение config/application.rb
  - `config.i18n.default_locale = :ru`
- Добавление параметров в файл config/**locales**/ru.yml

```
ru:
  form_header_for_new_course: Новый курс
  activerecord:
    models:
      course: Курс
    attributes:
      course:
        title: "Название"
        description: Описание
```

- Использование словаря в представлении
  - `<%= t("description") %>`

Что мы изучили?

- Узнали, как валидировать приходящие данные
- Использовали фильтры для избежания повторяющегося кода
- Добавили аутентификацию пользователя
- Научились создавать представления в различных форматах
- Локализовали наше приложение под русский язык

Что почитать?

- Ruby on Rails API

Что дальше?

- JavaScript. Уровень 1. Основы веб-программирования
- JavaScript. Уровень 2. Расширенные возможности