Ruby On Rails: Web-разработка по-другому!

- Несколько простых причин перейти на Ruby (and) Rails.
- Проблематика web-разработки.
- Разработка web-проектов всех типов: к чему нужно стремиться.
- Этапы разработки проекта.
- Заключение.

Простые причины перейти на Ruby

- Ruby — полностью объектный язык. Все переменные — объекты. Все операторы — методы.
- Ruby динамичен! Изменяйте класс on-the-fly или используйте модули.
- Синтаксис Ruby во многом похож на Smalltalk. It's really flexible (smile).
- В Ruby встроены средства контроля ошибок.

Примеры кода на Ruby

```
class Numeric
	def feet
		self*3.2808399
	end
end
```

```
self.say_hello if self.has_name?
```

```
3.times do
	puts «ANYTHING».lowercase
end
```

```
5.days.ago
```

Переменные и объекты в Ruby

- Все переменные являются объектами.
- Все операторы являются методами.
- Назначение переменной объекта — вызов метода.
- Нет множественного наследования, есть модули.
- Класс и/или объекты класса могут быть изменены in runtime, два объекта одного класса могут вести себя по-разному.
- Не нужно объявлять переменные.

Веб-разработка сегодня.

- Простая классификация проектов:
  - Промо-сайты.
  - Интернет-магазины, сайты-визитки, прочие стандартные проекты.
  - Нестандартные проекты, Saas веб-проекты, прочие проекты среднего размера с нестандартной логикой.
  - Крупные, очень крупные проекты. Корпоративные сайты в компаниях с 500+ сотрудников, Amazon.com, etc.

К чему стоит стремиться в проктах разных типов?

- Малые проекты
  - Сокращение сроков разработки.
  - Упрощение доработки функционала.
- Нестандартные / средние проекты
  - Масштабируемость.
  - Наличие плагинов.
- Крупные проекты
  - Стабильность.

К чему мы вообще стремимся в web-разработке?

- Уменьшение количества необходимого кода.
- Уменьшение сроков разработки.
- Уменьшение количества багов.
- Улучшение производительности и качества кода.

Парадигмы разработки. Требования к платформе.

- MVC
  - Структура приложения.
    - app/models
    - app/controllers
    - app/views
- Возможность расширения приложения за счет плагинов.
- BDD
  - Хорошее покрытие кода спецификациями.

Test-first, behavior-driven разработка.

- Разработка с применением Rspec.
- Установка Rspec в виде плагинов.
- Rspec для моделей.
- Rspec для контроллеров.
- И даже для представлений!
- User Stories scenario
  - As (actor), when i do (action), then i should get (result)

Rails-приложение. Основные этапы разработки.

- Дизайн и верстка + выбор визуальных эффектов.
- Модели данных.
- Контроллеры.
- Представления и «косметические» детали.
- Установка на сервер.

Модели данных

- Создание моделей
- Миграции
- Валидация
- Отношения
- Обратные вызовы (callbacks)

Генерация моделей

- script/generate
  - rspec_model / rspec_scaffold
  - Автоматическая генерация spec файлов.

DB Migrations

- Файлы в папке db/migrate
- Версионность базы данных
  - Development, test, production базы данных в проекте.
  - Таблица schema_info
  - Rake db:migrate VERSION=N task. Изменение версии базы.
- Синтаксис создания таблиц.

Валидация и отношения.

- Макро-подобные методы в Rails
  - Валидация моделей:
    - validates_presence_of :attribute
    - validates_uniqueness_of
    - validates_length_of :attr, 3..10
    - Etc
  - Отношения моделей
    - has_one :model_name
    - has_many (:through)
    - belongs_to
    - has_and_belongs_to_many

Контроллеры

- Генераторы
- Actions и маршрутизация
- REST

Actions и маршрутизация

- Контроллер объединяет несколько действия по работе с объектами одного типа.
- Одно действие — одна страница, это одна public функция контроллера. (conroller::action => /controller/action url)
- Настройки маршрутизации хранятся в /config/routes.rb

Правила маршрутизации

- Создавайте правила для url определенного вида (/book/:id/read/:page_number)
- Создавайте набор правил одной командой! (map.resources :posts)
- Используйте вложенные блоки!
  map.resources :posts, :has_many => :comments

What is REST?

- Rails 2.0 — поворот в сторону REST.
- Ресурсы. Метод Resources.
- Выгоды этого подхода.

Представления

- Представления в Rails.
- Шаблоны.
- Автоматическая генерация для scaffold.
- Функции-помощники.
- Части шаблонов. Partials.
- Возможность генерировать XML без шаблонов.
- RJS.

Шаблоны

- Html шаблоны со сниппетами ruby кода.
- Применение нескольких парсеров шаблонов в зависимости от расширения файла шаблона. (*.html.erb — «стандартный» вариант)
- Возможно парсить разные по своему типу шаблоны в зависимости от формата действия. (rjs для запросов page.js и rhtml для запроса page.html)

Функции — помощники

- Функции — помощники Rails.
  - link_to, form_for, javasctipt_include_tag, etc
- Ваши собственные функции-помощники.
  - Если вы используете какую-то часть логики неоднократно. (DRY)
  - Если в коде шаблона слишком много логики (вызов функции с большим числом параметров на несколько строк кода).

Partials & layouts

- Страница обрамлена своим layout файлом — шаблоном особого вида.
- Несколько контентных областей (слотов).
- Если какой-то блок html вы используете неоднократно — вынесите его в отдельный шаблон.
- Вставка шаблона render :partial => *.
- Поддержка тестирования на уровне тагов и партиалов. Поддержка тестирования каждого партиала отдельно.

RJS

- Генерируйте JavaScript в ответ на ajax запросы.
- Ответ сервера в виде скрипта vs обновление одного элемента. Подход Rails.
  - Вы можете применять классический подход.
  - Вы можете работать с RJS.
- Технология RJS. Генерируйте JS для DOM вашей страницы. Объект page.
- Аксессор page[element_id]

Заключение

- Приложение — демонстрация.
- Учебные материалы по Rails.
  - http://railscasts.com
  - http://wiki.rubyonrails.com
  - http://agilewebdevelopment.com