Ruby on Rails

Знакомство с каркасом для гибкой разработки web-приложений с базами данных

Шохирев Михаил
библиотекарь Клуба программистов «Caiman»
mikhail@shokhirev.com

Ruby и Rails

Ruby — динамический объектно-ориентированный язык программирования

Rails — каркас (framework) для быстрой и гибкой разработки web-приложений

Создатели

Ruby (1993..1995->)
Yukihiro Matsumoto («Matz»)
(Япония)

Ruby on Rails (2003..2005->)
David Heinemeier Hansson
(DHH) *(Дания)*

Язык Ruby

Ruby – это тщательно сбалансированный язык. В нём гармонично соединились конструкции любимых языков (Perl, Smalltalk, Eiffel, Ada и Lisp) его создателя, чтобы сформировать новый язык, в котором функциональное программирование уравновешено императивным программированием на основе объектно-ориентированного подхода.

Ruby's architect, Yukihiro Matsumoto

“Мне был нужен скриптовый язык, который был бы более мощным, чем Perl, и более объектно-ориентированным, чем Python.”

-- Matz

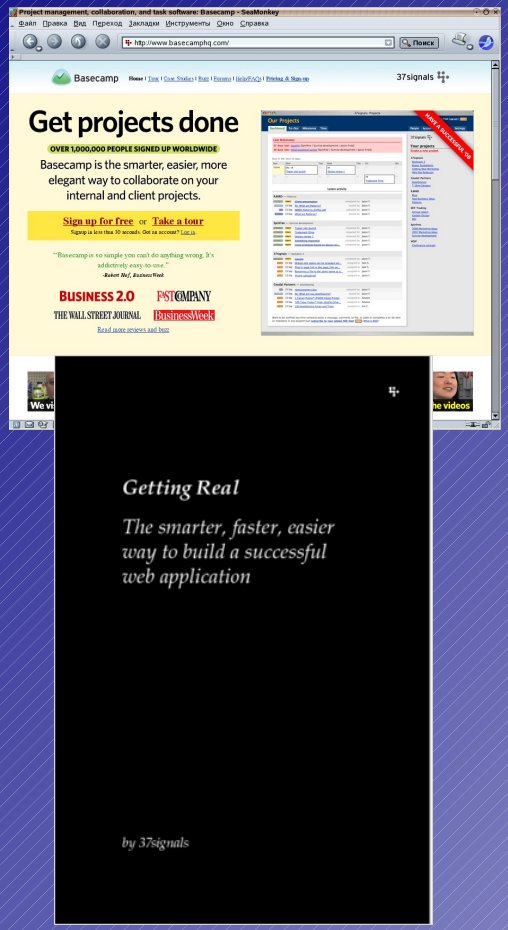
Особенности Ruby

```
"строка".class                                  # => String
"строка".class.superclass                       # => Object
25.class.superclass                             # => Integer
25.class.ancestors-25.class.included_modules    # => [Fixnum, Integer, Numeric, Object]

['это', 'одномерный', 'массив'].length         # => 3
{'это'=>'хэш', 'ассоциативный'=>'массив'}.size  # => 2
hash = { }                                      # пустой ассощиативный массив
hash[:key] = 'value'                            # :key - литерал класса Symbol
                                                # итераторы
3.times { print "Rails!" }                      # => Rails!Rails!Rails!
['это','массив'].each { |element| print element } # => этомассив
```

* Всё является объектом
* Можно расширять встроенные классы (Array, String, ...)
* Наглядный и удобный синтаксис
* Мощные и выразительные средства: итераторы, блоки, ...
* Развитые средства интроспекции и метапрограммирования
* **()** при вызове методов необязательны
* **:key => 'value'** для имитации именованных параметров
* хорошо подходит для создания **DSL** (Domain-Specific Lanuage)

Ruby on Rails



Ruby on Rails (RoR / Rails) создан молодым датским программистом Дэвидом Хайнемейером Ханссоном в 2003 году в ходе разработки проекта BaseCamp для компании 37signals. В 2004 году был выделен из работающего приложения в виде средства разработки с открытыми кодами, распространяемого по лицензии MIT.

Версия 1.0 вышла в 2005 году, 2.0 - в 2007 году.

http://gettingreal.37signals.com/GR_rus.php

Предыстория RoR

«У меня был опыт работы с PHP и Java, именно с этой областью и связана моя основная профессиональная подготовка. ... Существовало две силы, которые продолжали наращивать свое влияние. С помощью Ruby On Rails я попытался объединить лучшие черты обоих миров, с тем чтобы добиться такой же быстроты, как у PHP, и такой же прочности, как у Java.»

Давид Хайнемейер Ханссон

http://www.osp.ru/cw/2007/39/4442414/

Характеристики Rails

- Самодостаточный каркас (full-stack framework)
- Высокопроизводительный инструмент
- Кросс-платформенный (*BSD, Linux, MacOS, Solaris, UNIX, MS Windows*)
- Неазвисмый от СУБД (databas-agnostic)
- Легко расширяемый (plug-ins)
- Свободно распространяемый (open source)

Поддерживаемые СУБД

Имеются много адаптеров для подключения ко всем распространённым СУБД, включая: Cache, IBM DB2, Firebird / Interbase, FrontBase, Informix, Ingres, MS SQL Server, MySQL, OpenBase, Oracle, PostgreSQL, SQLAnywhere, SQLite, Sybase ASA.

Несложно разработать новый адаптер для конкретной СУБД и другого источника данных (например, есть ODBC, ActiveLDAP, ActiveRDF).

Принципы Rails

- **ASD** – гибкая разработка приложений
- **DRY** – минимизация повторений
- **CoC** – «соглашения вместо конфигурации»
- **Metaprogramming** – метапрограммирование
- **DSL** – предметно-ориентированные языки
- **OOP** – объектный подход повсюду
- **ORM** – классы-обёртки для таблиц БД
- **MVC** – модель-представление-контроллер
- **TDD** – разработка через тестирование

Принципы Rails

Agile Software Development (ASD)
«гибкая методология разработки» – принципы создания программ в условиях жёсткого графика и часто меняющихся требований, на основе которых созданы методики: *Extreme Programming (XP), Agile Modelling, Agile Unified Process (AUP), Essential UP (EssUP), Test-Driven Development (TDD), Behavior-Driven Development (BDD)* и т. п.

http://agilemanifesto.org/
http://ru.wikipedia.org/wiki/Гибкая_методология_разработки

Принципы Rails

"Don't Repeat Yourself!" (DRY)
(«Не повторяйся!»)
– постоянное стремление к минимизации дублирования любой информации в разрабатываемой системе.

"Каждый фрагмент знания должен иметь единственное, однозначное, надёжное представление в системе."
Эндрю Хант, Дэйв Томас, «Программист-прагматик», глава 2.

http://www.pragprog.com/the-pragmatic-programmer

Принципы Rails

Convention over Configuration (CoC) («соглашения вместо конфигурации») – тщательно продуманные соглашения об именовании минимизируют необходимость в конфигурационных файлах и параметрах, а содержащаяся в соглашениях по умолчанию метаинформация может эффективно использоваться для настройки приложения (статической и динамической).

http://en.wikipedia.org/wiki/Convention_over_Configuration

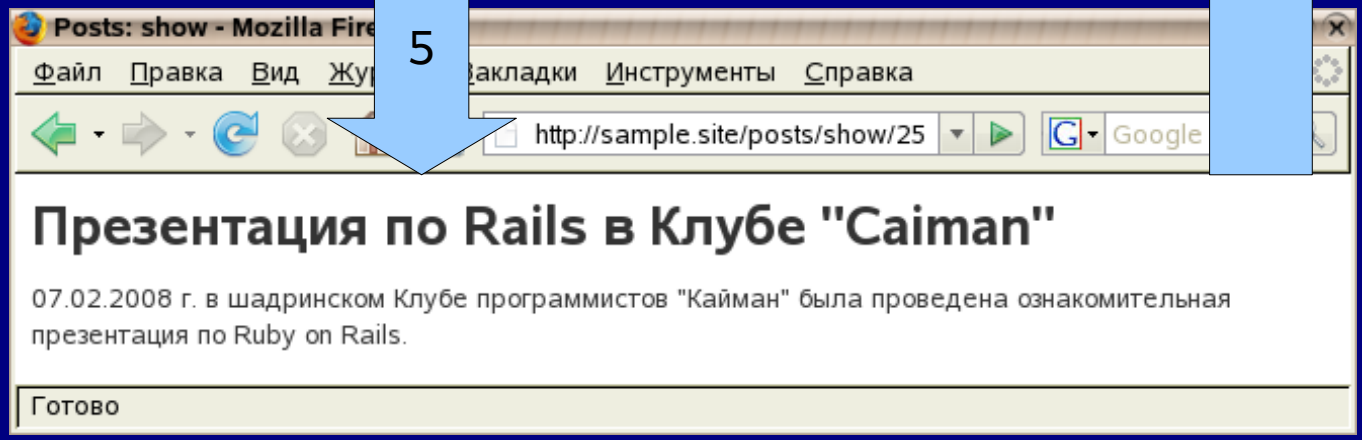
Пример соглашений в RoR

```
http://sample.org/posts/show/25
```

```
# app/controllers/posts_controller.rb
class PostsController < ApplicationController
  def show # действие
    @post = Post.find(params[:id]) # id = 25
  end
end
```

```
# app/models/post.rb
class Post < ActiveRecord::Base
end
```

```
# /app/views/posts/show.rhtml
<h1><%=h @post.title %></h1>
<p><%=h @post.body %></p>
```



1 2 3 4 5

Принципы Rails

Metaprogramming
(метапрограммирование) –
позволяет на основе метаданных
(из СУБД, соглашений, конфигурации)
генерировать статические части приложения
и динамически настраивать программы
(изменять и дополнять) во время их
выполнения.

http://ru.wikipedia.org/wiki/Метапрограммирование

Принципы Rails

Domain-Specific Language (DSL)
(язык предметной области) –
специализированный язык программирования для решения определённого круга задач в конкретной предметной области, который в сочетании с метапрограммированием используется для автоматизации разработки программного обеспечения.

http://ru.wikipedia.org/wiki/DSL
http://offline.computerra.ru/print/offline/2006/630/258015/

DSL в Ruby on Rails

```
                                # Изменение схемы данных
add_column     :people, :email, :text
change_column  :orders, :order_type, :string, :null=>false
remove_index   :people, :name
rename_table   :participants, :project_members
```

```
                                # Описание модели данных
class Person < ActiveRecord::Base
  belongs_to :parent, :class_name => "Person"
  has_many :children, :class_name => "Person",
                      :foreign_key => :parent_id
end
```

```
                                # Манипулирование данными
mother = Person.find_by_name 'Мария'
child = mother.children.create :name=>'Иван'
print child.parent.name # => Мария
```

DSL в Ruby on Rails

```
                                        # Представление данных
<% form_for :person,:url =>{:action=>:create} do |form| %>
  <p>Имя:
    <%= form.text_field :name, :size => 30 %></p>
  <p>Пол:
    <%= form.select :gender, %w{ мужской женский } %></p>
  <p>Заметки:
    <%= form.text_area :notes, :rows => 3 %></p>
  <p><%= submit_tag %></p>
<% end %>
```

Начать разрабатывать на Rails можно,
не зная языка Ruby,
поскольку в основном используются
несложные и наглядные DSL.

Принципы Rails

"Less code"
(«Короткие исходники»)
– программисту даны мощные средства (DSL, metaprogramming, helpers, классы RoR и средства Ruby), помогающие сократить размер исходных текстов программ.

Короткие программы лучше понимать, проще изменять, в них меньше ошибок, их легче находить и исправлять.

Принципы Rails

Object-Oriented Programming (OOP)
(объектно ориентированное программирование)
– все компоненты каркаса и приложения представлены в виде классов, включая инструменты изменения схемы данных, средства доступа к БД и описания связей, генераторы шаблонов, тесты и т. д.

Всё разрабатывается на одном языке, **Ruby**, без привязки к платформе или СУБД.

Принципы Rails

Object-Relational Mapping (ORM)
(отображение реляционной модели на объектно-ориентированную) –
модели ActiveRecord, предоставляющие классы-обёртки для работы с таблицами БД, средства работы с записями, как с объектами, поисковые методы, объектные средства управления связями (N:1, 1:1, 1:N, N:M) между моделями.

http://en.wikipedia.org/wiki/Active_Record
http://martinfowler.com/eaaCatalog/activeRecord.html

ORM в ActiveRecord

```ruby
# app/models/country.rb
class Country < ActiveRecord::Base # Страна
  has_many :localities # содержит много населённых пунктов
end

# app/models/locality.rb
class Locality < ActiveRecord::Base # Населённый пункт
  belongs_to :country # принадлежит стране
end
```

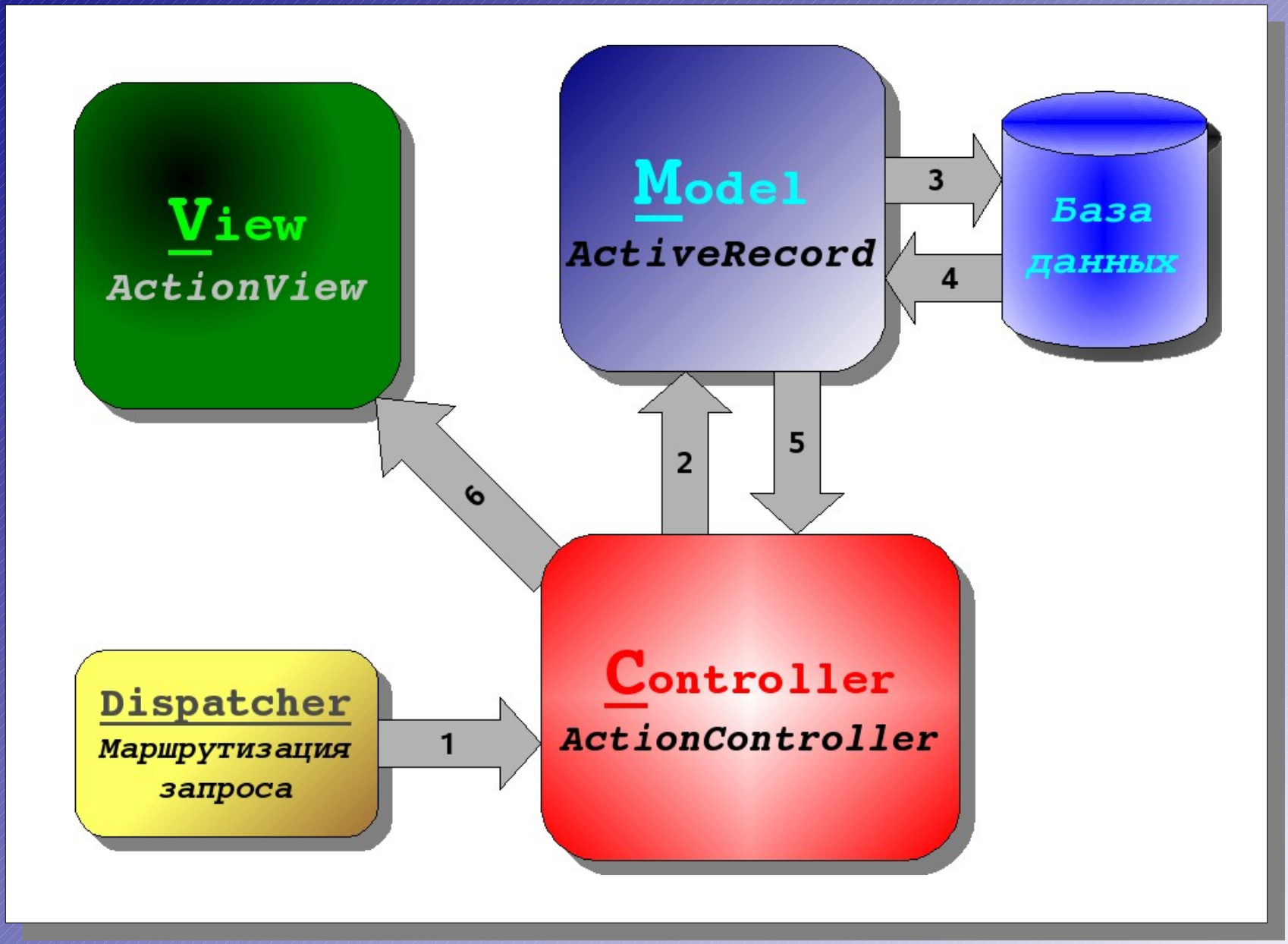
```ruby
# ruby script/console
r = Country.create(:name => 'Россия')
t = Locality.create(:name => 'Шадринск', :country => r)
e = Locality.create(:name => 'Екатеринбург')
r.localities << e                    # добавить город к стране
t.country.name                       # => Россия
r.localities.size                    # => 2
```

Принципы Rails

Model-View-Controller (MVC)
("модель-представление-контроллер") –
архитектурный образец проектирования и программирования,
когда система подразделяется на уровни,
изолирующие в себе особенности
доступа к данным (Model),
предствления информации (View) и
управления логикой приложения (Controller).

http://ru.wikipedia.org/wiki/MVC

Архитектура Rails



Принципы Rails

Test-Driven Development (TDD)
(«разработка через тестирование») –
в Rails предусмотрены удобные средства для организации постоянного и всестороннего пакетного тестирования всех компонентов web-приложения, включая модели доступа к данным (unit tests), маршрутизацию запросов и их обработку в контроллерах (functional tests), взаимодействие компонентов (integration tests).

http://ru.wikipedia.org/wiki/Разработка_через_тестирование

Компоненты Rails

- **ActiveRecord** – объектные модели для БД
- **ActionView** – генерация представлений
- **ActionController** – выполнение действий
- **ActiveResource** – доступ к ресурсам по REST
- **ActionMailer** – работа с электронной почтой
- **ActiveSupport** – вспомогательные средства
- **ActionWebService*** – доступ через Web-сервисы (SOAP, XML-RPC)

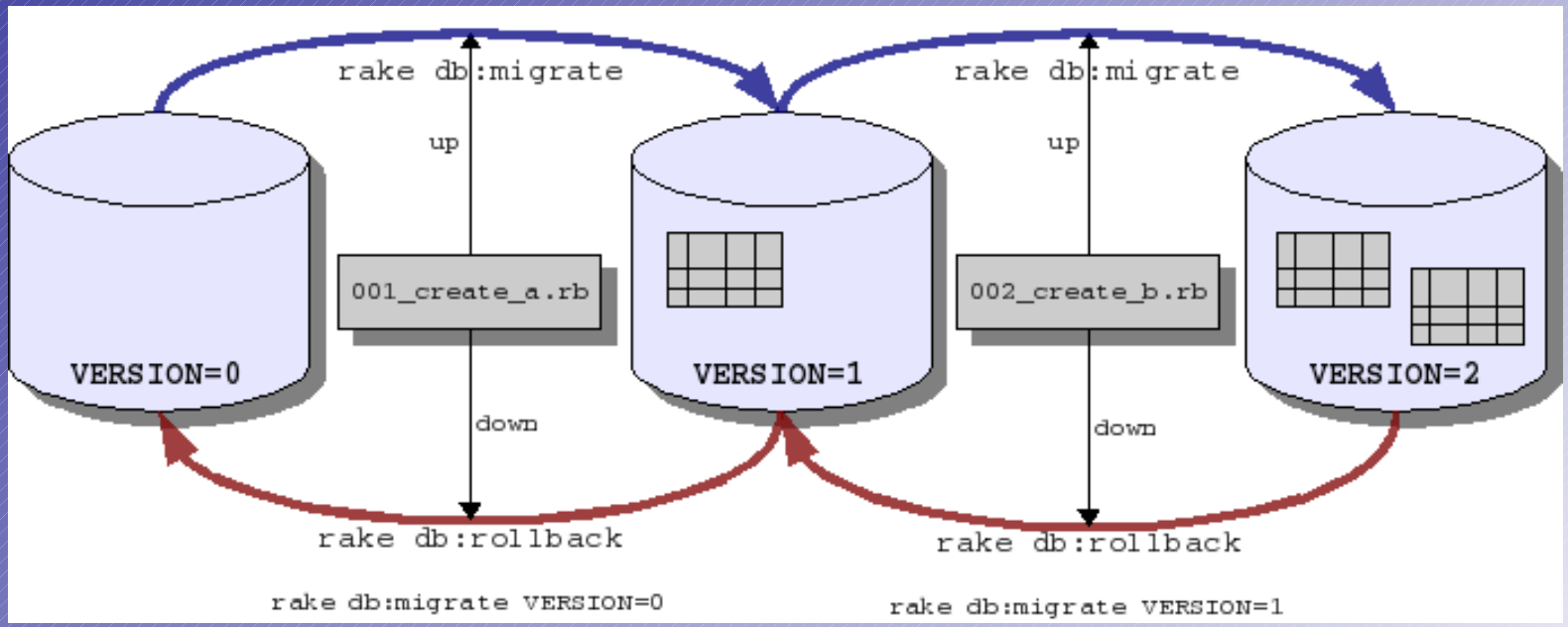
* дополнительный компонент, начиная с версии 2.0.

ActiveRecord

- Соединения с СУБД (adapters)
- Изменение схемы данных (Migrations)
- Object-Relational Mapping (**ORM**)
- Манипулирование данными (**CRUD**)
- Поисковые методы (finders)
- Управление связями (model relationships)
- Проверки данных (validation)
- Триггеры (callbacks): before_xxx, after_xxx

ActiveRecord::Migration

«*Миграция*» – класс на Ruby, где в независимой от СУБД форме описывается **задание на переход схемы базы данных из одного состояния в другое**, во время которого выполняется создание и удаление таблиц, индексов, изменение структуры данных (добавление, удаление или изменение колонок и их типов).



Формат миграции

```ruby
# db/migrate/001_create_people.rb

class CreatePeople < ActiveRecord::Migration
  def self.up # выполнение перехода на 1 версию вперёд
    create_table :people do |t|
      t.string     :name       # имя
      t.string     :email      # адрес э-почты
      t.date       :birthday   # дата рождения
      t.timestamps             # время создания и изменения
    end
    add_index :people, :name
  end

  def self.down # откат изменений на 1 версию назад
    remove_index :people, :name
    drop_table :people
  end
end
```

ActionView

- Генерация многих представлений (multi-view)
- Шаблоны (templates, partials, layouts)
- **Erb** (Embedded Ruby) для (X)HTML
- **Builder** для XML / RSS / Atom
- Помощники (helpers)
- Работа со ссылками, ресурсами, формами, сообщениями об ошибках
- Визуальные эффекты (*Script.aculo.us*)

ActionController

- Маршрутизация запросов (**routing**)
- Выполнение методов действий (**action**)
- Обмен данными между действиями (*flash*)
- Взаимодействие с **AR** и **AV**
- Рендеринг и переадресация запросов
- Сеансы (sessions & cookies)
- Фильтры (before_filter, after_filter, around_filter)
- Проверка условий выполнения (verification)

ActiveSupport

- Вспомогательные классы и модули
- Дополнения к встроенным классам Ruby:
  - Array, Hash, Enumeration
  - String, Integer, Numeric, Float
  - Time, Date, DateTime
  - Symbol, Range
- Поддержка Unicode (**UTF-8**)
- Сериализация и преобразования данных
- logger, JSON, testing, ...

Средства на Ruby

- **Ruby** – мощный универсальный язык
- **RubyGems** – система управления пакетами
- **Rake** – менеджер заданий (~= make / Ant)
- **WEBrick** – встроенный Web-сервер
- **generators** – генераторы частей приложения
- **scaffold** – «лесá, подпорки» в начале работы
- **plug-ins** – подключаемые модули

Технологии и инструменты

- **Web-Services** – поддержка web-служб
- **REST** – удалённый HTTP-доступ к ресурсам
- **AJAX** – Asynchronous JavaScript And XML
- **XML**, **RSS** – удобные средства генерации
- **YAML** = YAML Ain't Markup Language
- **Prototype** – JavaScript: DOM, OOP, AJAX
- **Script.aculo.us** – JS: widgets, visual effects
- **Subversion** – интеграция с VCS

Web-сервисы на REST

REST (Representational State Transfer) – принципы удалённого манипулирования (CRUD = Create, Read, Update, Delete) сетевыми ресурсами с использованием имеющихся общедоступных средств:

- стандартные команды HTTP (POST, GET, PUT, DELETE) для операций CRUD;
- URI для адресации ресурсов – http://site/category/resource/id/subresource/sub_id;
- текстовый формат (plain text, XML, YAML, JSON) для обмена данными.

ActiveResource

Появившийся в версии 2.0 компонент **ActiveResource** предоставляет удобные объектно-ориентированные средства для манипулирования (CRUD) сетевыми ресурсами удалённого приложения (не обязательно Rails) по технологии **REST**: работа с моделями ресурсов, расположенными на удалённом сервере, аналогична работе с моделями ActiveRecord на сервере баз данных.

http://api.rubyonrails.org/classes/ActiveResource.html

Модели в AR и ARes

```
# site1/application1/app/models/post.rb
 class Post < ActiveRecord::Base # обращается к СУБД
   has_many :comments
 end
```

```
# site2/application2/app/models/post.rb
 class Post < ActiveResource::Base # обращается к сайту
   self.site = 'http://login:password@remote.site:3000'
 end

# site2/application2/app/models/comment.rb
 class Comment < ActiveResource::Base # обращается к сайту
   self.site = 'http://login:password@remote.site:3000'
   self.prefix = '/posts/:post_id/'
 end
```

```
site2/application2> ruby script/console

post = Post.create :title => "Добавлен удалённо"
all_posts = Post.find :all # найти все сообщения
post = Post.find 25        # найти сообщение 25
comments = Comment.find(:all, :params => {:post_id => post.id}
```

AJAX

Asynchronous JavaScript And XML – поддерживается в Rails с помощью библиотеки **Prototype** и вспомогательных модулей (helpers) для удобной *генерации программного кода* на JavaScript, чтобы организовать асинхронное взаимодействие между клиентской и серверной частями приложения посредством XMLHttpRequest (**XHR**).

http://ru.wikipedia.org/wiki/AJAX

plug-ins

Подключаемые модули для добавления и расширения функциональности (около 80 репозитариев с сотнями различных подключаемых модулей):

ruby script/plugin discover

ruby script/plugin install http://plugin/address/

* classic_pagination
* will_paginate
* acts_as_list
* acts_as_tree
* acts_as_nested_set
* acts_as_state_machine
* Streamlined
* ActiveScaffold
* Rails PDF Plugin
* Simple LDAP Authenticator
* Simple Sidebar
* Globalize

http://wiki.rubyonrails.com/rails/pages/Plugins

Почему мне нравится RoR

- Обозримый, понятный и лёгкий в освоении
- Правильно и удачно спроектированный
- Чётко структурированный и модульный
- Направляет разработку в правильное русло
- Даёт удобные высокоуровневые инструменты
- Автоматизирует многие скучные действия
- Помогает тщательно и легко тестировать
- Предоставляет ООП для работы с БД

Разработчики о Rails

«Невозможно не замечать Ruby on Rails. Он произвел огромный эффект как в сообществе Ruby, так и вне его. Rails стал стандартом, с которым сравнивают себя даже прочно утвердившиеся инструменты.»

***Мартин Фаулер**,*
известный разработчик и автор многих популярных книг,
один из основоположников движения
«Agile Software Development»

«Ruby on Rails — это прорыв в снижении барьера при переходе к программированию. Мощные web-приложения, которые раньше разрабатывались за недели или месяцы, теперь могут быть сделаны за считанные дни.»

***Тим О'Рейли**,*
основатель издательства O'Reilly Media,
активист движений Open Source и Free Culture

Разработчики о Rails

«Rails — это самый продуманный каркас для web-приложений, какой я когда-либо использовал. И это после десятилетия зарабатывания на жизнь разработкой web-приложений. Я создавал свои собственные каркасы, помогал разрабатывать Servlet API и сделал с нуля несколько web-серверов. Но ничего подобного прежде не делал никто.»

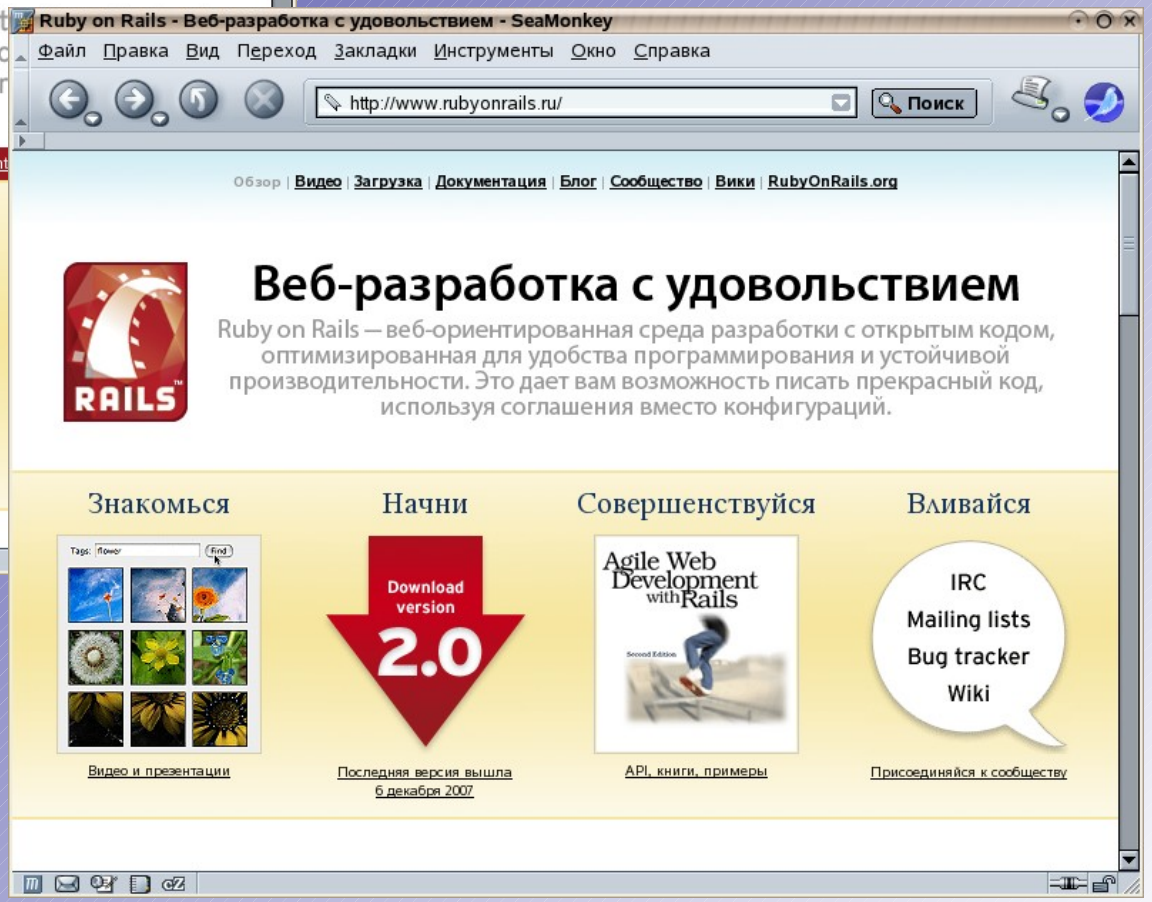
Джеймс Дункан Дэвидсон,
создатель сервера Tomcat и утилиты Ant

«Как хороший рецепт, Rails помогает вам достичь нового уровня производительности за счёт сочетания правильных составляющих в нужном количестве.»

Керт Хиббс,
разработчик пакета InstantRails,
соавтор книги «Ruby on Rails: Up & Running»

Основные сайты

http://www.rubyonrails.org/



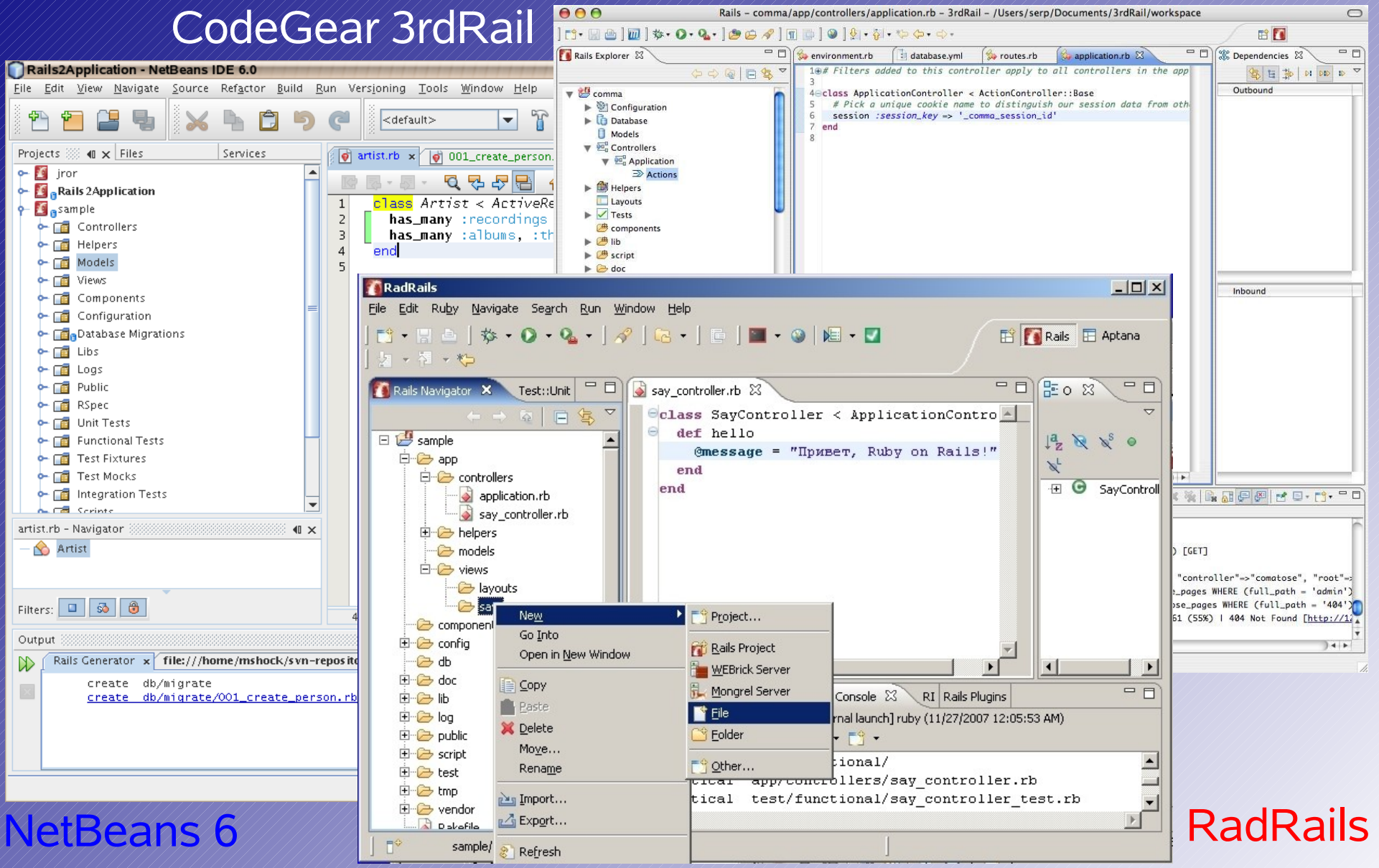
http://www.rubyonrails.ru/

Комплексный установщик

- Ruby, RubyGems
- **Ruby on Rails**
- Apache, Mongrel
- OpenSSL
- MySQL, SQLite
- phpMyAdmin
- ImageMagick
- zlib, libiconv
- Subversion

Средства разработки

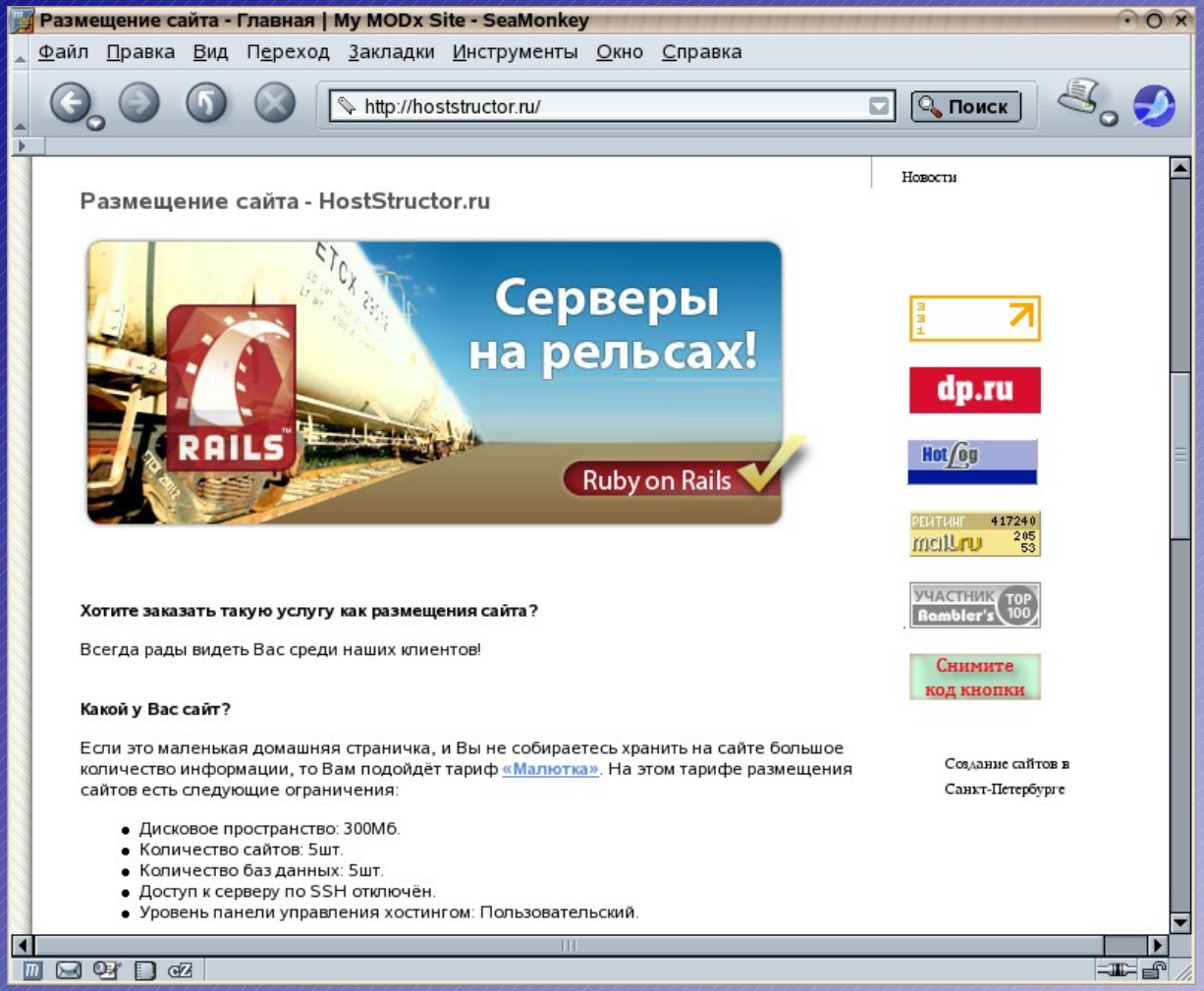
CodeGear 3rdRail



NetBeans 6

RadRails

А как насчёт хостинга?

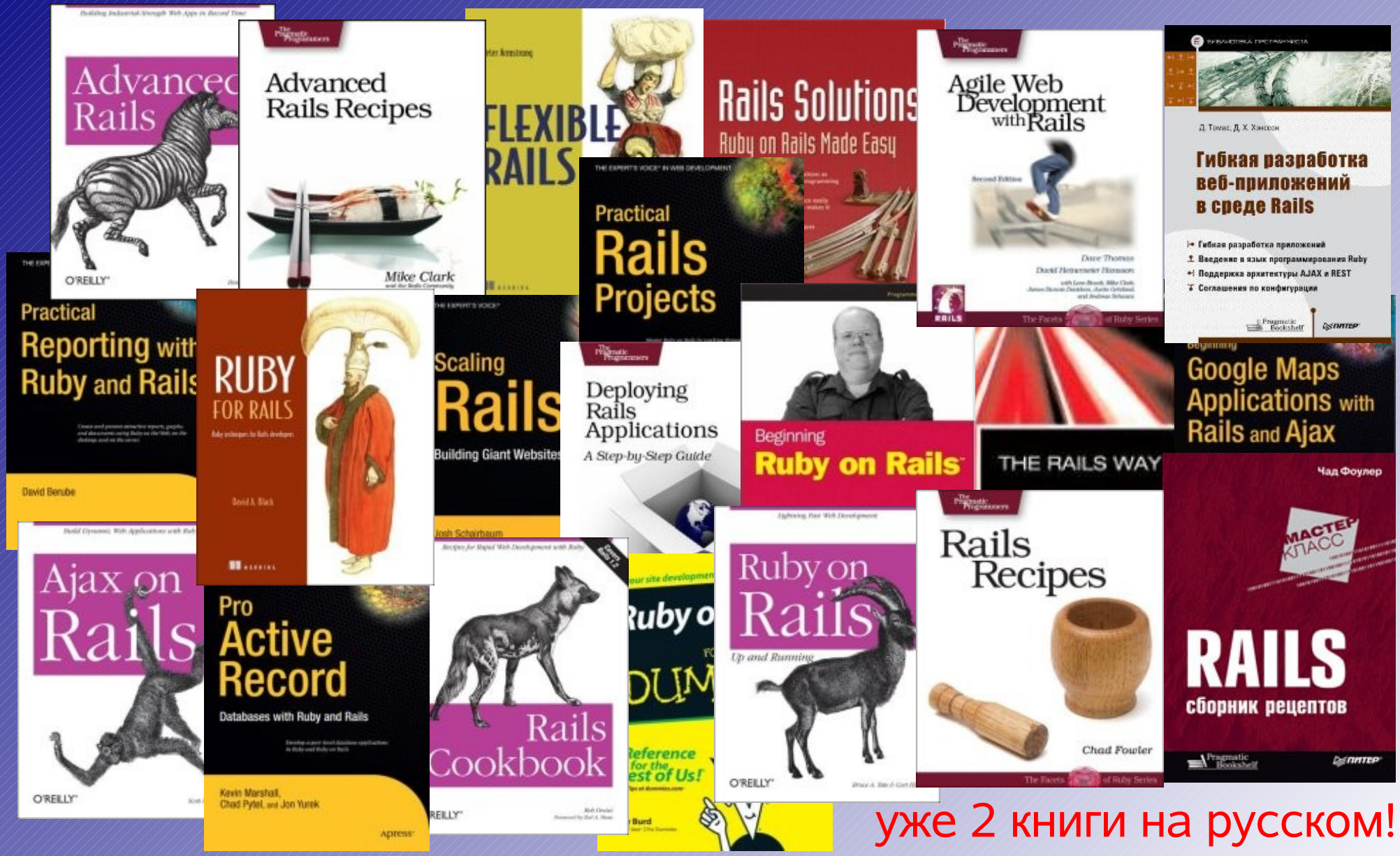


Хостинг есть!

В том числе в России.

http://www.railshosting.org/
http://www.hostingrails.com/
http://www.rubyonrailswebhost.com
http://wiki.rubyonrails.org/rails/pages/RailsWebHosts

Литература



уже 2 книги на русском!

Ruby on Rails

Вопросы?

Мнения?

Демонстрация!

Презентация для членов Клуба «Caiman» (г. Шадринск, Россия).
версия 1.5 (2008.02.07..14)