Django v/s Ruby on Rails

A Newbie Web Developer's Perspective

Shreyank Gupta/shreyankg@fedoraproject.org

March 20, 2010

Fascination

From the Website

Ruby on Rails

Web Development that doesn't hurt

- Open-source web development framework that's optimised for programmer's happiness and sustainable productivity.
- Lets you write beautiful code by favoring convention over configuration.

Django

The Web framework for perfectionists (with deadlines).

- Django makes it easier to build better Web apps more quickly and with less code.
- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

From the Website

Ruby on Rails

Web Development that doesn't hurt

- Open-source web development framework that's optimised for programmer's happiness and sustainable productivity.
- Lets you write beautiful code by favoring convention over configuration.

Django

The Web framework for perfectionists (with deadlines).

- Django makes it easier to build better Web apps more quickly and with less code.
- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

From the Website

Ruby on Rails

Web Development that doesn't hurt

- Open-source web development framework that's optimised for programmer's happiness and sustainable productivity.
- Lets you write beautiful code by favoring convention over configuration.

Django

The Web framework for perfectionists (with deadlines).

- Django makes it easier to build better Web apps more quickly and with less code.
- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

From the Website

Ruby on Rails

Web Development that doesn't hurt

- Open-source web development framework that's optimised for programmer's happiness and sustainable productivity.
- Lets you write beautiful code by favoring convention over configuration.

Django

The Web framework for perfectionists (with deadlines).

- Django makes it easier to build better Web apps more quickly and with less code.
- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

From the Website

Ruby on Rails

Web Development that doesn't hurt

- Open-source web development framework that's optimised for programmer's happiness and sustainable productivity.
- Lets you write beautiful code by favoring convention over configuration.

Django

The Web framework for perfectionists (with deadlines).

- Django makes it easier to build better Web apps more quickly and with less code.
- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Documentation

Ruby on Rails

- Guides - `http://guides.rubyonrails.org/`
- RailsCasts - `http://railscasts.com/`

Django

- Docs - `http://docs.djangoproject.com/`
- DjangoSnippets - `http://www.djangosnippets.org/`

Documentation

Ruby on Rails

- Guides - `http://guides.rubyonrails.org/`
- RailsCasts - `http://railscasts.com/`

Django

- Docs - `http://docs.djangoproject.com/`
- DjangoSnippets - `http://www.djangosnippets.org/`

Documentation

Ruby on Rails

- Guides - `http://guides.rubyonrails.org/`
- RailsCasts - `http://railscasts.com/`

Django

- Docs - `http://docs.djangoproject.com/`
- DjangoSnippets - `http://www.djangosnippets.org/`

Discipline v/s Flexibility

The Rails Way

Rails is opinionated software. That is, it assumes that there is a best way to do things, and its designed to encourage that best way and in some cases to discourage alternatives. If you learn "The Rails Way", you'll probably discover a tremendous increase in productivity. If you persist in bringing old habits from other languages to your Rails development, and trying to use patterns you learned elsewhere, you may have a less happy experience.

– http://guides.rubyonrails.org/getting_started.html

The Rails Way

DRY - Dont Repeat Yourself

Writing the same code over and over again is a bad thing.

Convention Over Configuration

Rails makes assumptions about what you want to do and how you're going to do it, rather than letting you tweak every little thing through endless configuration files.

REST

Organizing your application around resources and standard HTTP verbs is the fastest way to go.

The Rails Way

DRY - Dont Repeat Yourself

Writing the same code over and over again is a bad thing.

Convention Over Configuration

Rails makes assumptions about what you want to do and how you're going to do it, rather than letting you tweak every little thing through endless configuration files.

REST

Organizing your application around resources and standard HTTP verbs is the fastest way to go.

The Rails Way

DRY - Dont Repeat Yourself

Writing the same code over and over again is a bad thing.

Convention Over Configuration

Rails makes assumptions about what you want to do and how you're going to do it, rather than letting you tweak every little thing through endless configuration files.

REST

Organizing your application around resources and standard HTTP verbs is the fastest way to go.

Balance

Architecture

Ruby on Rails

MVC : Model - Controller - View

versus

Django

MTV : Model - Template - View

Architecture

Ruby on Rails

MVC : Model - Controller - View

versus

Django

MTV : Model - Template - View

Architecture

Ruby on Rails

MVC : Model - Controller - View

versus

Django

MTV : Model - Template - View

Object Relational Mapping - ORM

Ruby on Rails

Active Records

versus

Django

Django Model Instance Reference

Object Relational Mapping - ORM

Ruby on Rails

Active Records

versus

Django

Django Model Instance Reference

Object Relational Mapping - ORM

Ruby on Rails

Active Records

versus

Django

Django Model Instance Reference

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
  - Prototype.js
  - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
  - Prototype.js
  - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
  - Prototype.js
  - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
  - Prototype.js
  - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
  - Prototype.js
  - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Bundled JavaScript

Ruby on Rails

- Comes with bundled copies of:
    - Prototype.js
    - script.aculo.us
- `JavaScriptHelper` part of the framework.

Django

- Ships JQuery as a part of the Admin Interface.
- Not a part of the framework.

Disbalance

Views/Templates

Ruby on Rails

Rails Rendering/Layout

versus

Django

Django Templating

Views/Templates

Ruby on Rails

Rails Rendering/Layout

versus

Django

Django Templating

Views/Templates

Ruby on Rails

Rails Rendering/Layout

versus

Django

Django Templating

Directory Structure

Ruby on Rails

```
$ rails newrails
```

Django

```
$ django-admin startproject newdjango
$ cd newdjango
$ python manage.py startapp myapp
```

Now let's have a look at the tree.

Directory Structure

Ruby on Rails

```
$ rails newrails
```

Django

```
$ django-admin startproject newdjango
$ cd newdjango
$ python manage.py startapp myapp
```

Now let's have a look at the tree.

Directory Structure

Ruby on Rails

```
$ rails newrails
```

Django

```
$ django-admin startproject newdjango
$ cd newdjango
$ python manage.py startapp myapp
```

Now let's have a look at the tree.

Authentication Modules

Ruby on Rails

- No built-in authentication framework.
- Third party authentication plugins available:
  - Restful Authentication.
  - Authlogic

Django

- Authentication part of the framework.

Authentication Modules

Ruby on Rails

- No built-in authentication framework.
- Third party authentication plugins available:
  - Restful Authentication.
  - Authlogic

Django

- Authentication part of the framework.

Authentication Modules

Ruby on Rails

- No built-in authentication framework.
- Third party authentication plugins available:
  - Restful Authentication.
  - Authlogic

Django

- Authentication part of the framework.

Authentication Modules

Ruby on Rails

- No built-in authentication framework.
- Third party authentication plugins available:
  - Restful Authentication.
  - Authlogic

Django

- Authentication part of the framework.

Authentication Modules

Ruby on Rails

- No built-in authentication framework.
- Third party authentication plugins available:
    - Restful Authentication.
    - Authlogic

Django

- Authentication part of the framework.

Learning curve

Ruby on Rails

- Steep Learning Curve
- Lots of framework specific stuff to learn

Django

- Slight learning curve if you are already familier with Python.
- The Django template language ;-)

Learning curve

Ruby on Rails

- Steep Learning Curve
- Lots of framework specific stuff to learn

Django

- Slight learning curve if you are already familier with Python.
- The Django template language ;-)

Learning curve

Ruby on Rails

- Steep Learning Curve
- Lots of framework specific stuff to learn

Django

- Slight learning curve if you are already familier with Python.
- The Django template language ;-)

Learning curve

Ruby on Rails

- Steep Learning Curve
- Lots of framework specific stuff to learn

Django

- Slight learning curve if you are already familier with Python.
- The Django template language ;-)

Database Schema Management

Ruby on Rails

```
$ rake db:migrate
```

versus

Django

```
$ python manage.py syncdb
```

Database Schema Management

Ruby on Rails

```
$ rake db:migrate
```

versus

Django

```
$ python manage.py syncdb
```

Database Schema Management

Ruby on Rails

```
$ rake db:migrate
```

versus

Django

```
$ python manage.py syncdb
```

Unique (Selling|Breaking) Points

REST - Representational State Transfer

- Using resource identifiers (URLs) to represent resources
- Transferring representations of the state of that resource between system components.

Example

to a Rails application a request such as:
`DELETE /photos/17`
would be understood to refer to a photo resource with the ID of 17, and to indicate a desired action — deleting that resource. REST is a natural style for the architecture of web applications, and Rails makes it even more natural by using conventions to shield you from some of the RESTful complexities and browser quirks.

REST - Representational State Transfer

- Using resource identifiers (URLs) to represent resources
- Transferring representations of the state of that resource between system components.

Example

to a Rails application a request such as:
`DELETE /photos/17`
would be understood to refer to a photo resource with the ID of 17, and to indicate a desired action — deleting that resource. REST is a natural style for the architecture of web applications, and Rails makes it even more natural by using conventions to shield you from some of the RESTful complexities and browser quirks.

Migrations

Definition

Migrations are a convenient way for you to alter your database in a structured and organised manner. You could edit fragments of SQL by hand but you would then be responsible for telling other developers that they need to go and run it. You'd also have to keep track of which changes need to be run against the production machines next time you deploy.

Proof.

```
# This file is auto-generated from the current state of the
database.  Instead of editing this file,
# please use the migrations feature of Active Record to
incrementally modify your database, and
# then regenerate this schema definition.
```

□

Migrations

Definition

Migrations are a convenient way for you to alter your database in a structured and organised manner. You could edit fragments of SQL by hand but you would then be responsible for telling other developers that they need to go and run it. You'd also have to keep track of which changes need to be run against the production machines next time you deploy.

Proof.

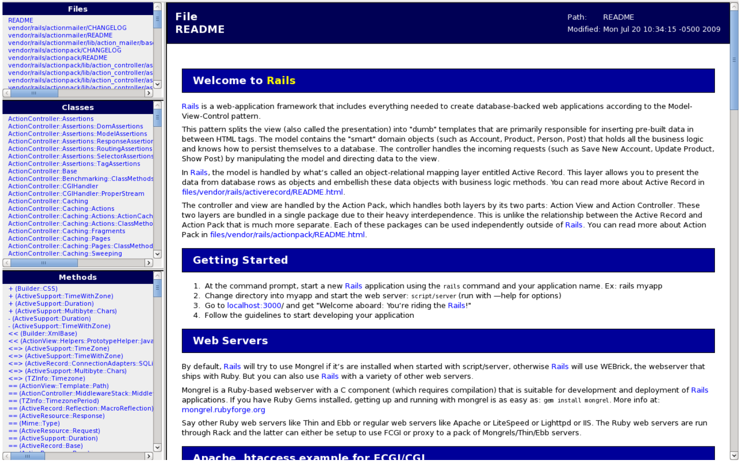
```
# This file is auto-generated from the current state of the
database.  Instead of editing this file,
# please use the migrations feature of Active Record to
incrementally modify your database, and
# then regenerate this schema definition.
```
□

Rails Scaffold

Definition

Scaffolds an entire resource, from model and migration to controller and views, along with a full test suite. The resource is ready to use as a starting point for your RESTful, resource-oriented application.

Cross-Reference API



Files

README
vendor/rails/actionmailer/CHANGELOG
vendor/rails/actionmailer/README
vendor/rails/actionmailer/lib/action_mailer/bas
vendor/rails/actionpack/CHANGELOG
vendor/rails/actionpack/README
vendor/rails/actionpack/lib/action_controller/as
vendor/rails/actionpack/lib/action_controller/as
vendor/rails/actionpack/lib/action_controller/as

Classes

ActionController::Assertions
ActionController::Assertions::DomAssertions
ActionController::Assertions::ModelAssertions
ActionController::Assertions::ResponseAssertior
ActionController::Assertions::RoutingAssertions
ActionController::Assertions::SelectorAssertions
ActionController::Assertions::TagAssertions
ActionController::Base
ActionController::Benchmarking::ClassMethods
ActionController::CGIHandler
ActionController::CGIHandler::ProperStream
ActionController::Caching
ActionController::Caching::Actions
ActionController::Caching::Actions::ActionCach
ActionController::Caching::Actions::ClassMetho
ActionController::Caching::Fragments
ActionController::Caching::Pages
ActionController::Caching::Pages::ClassMethod
ActionController::Caching::Sweeping

Methods

+ (Builder::CSS)
+ (ActiveSupport::TimeWithZone)
+ (ActiveSupport::Duration)
+ (ActiveSupport::Multibyte::Chars)
- (ActiveSupport::Duration)
- (ActiveSupport::TimeWithZone)
<< (Builder::XmlBase)
<< (ActionView::Helpers::PrototypeHelper::Java
<=> (ActiveSupport::TimeZone)
<=> (ActiveSupport::TimeWithZone)
<=> (ActiveRecord::ConnectionAdapters::SQLi
<=> (ActiveSupport::Multibyte::Chars)
<=> (TZInfo::Timezone)
== (ActionView::Template::Path)
== (ActionController::MiddlewareStack::Middle
== (TZInfo::TimezonePeriod)
== (ActiveRecord::Reflection::MacroReflection)
== (ActiveResource::Response)
== (Mime::Type)
== (ActiveResource::Request)
== (ActiveSupport::Duration)
== (ActiveRecord::Base)

File
README

Path: README
Modified: Mon Jul 20 10:34:15 -0500 2009

Welcome to Rails

Rails is a web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Control pattern.

This pattern splits the view (also called the presentation) into "dumb" templates that are primarily responsible for inserting pre-built data in between HTML tags. The model contains the "smart" domain objects (such as Account, Product, Person, Post) that holds all the business logic and knows how to persist themselves to a database. The controller handles the incoming requests (such as Save New Account, Update Product, Show Post) by manipulating the model and directing data to the view.

In Rails, the model is handled by what's called an object-relational mapping layer entitled Active Record. This layer allows you to present the data from database rows as objects and embellish these data objects with business logic methods. You can read more about Active Record in files/vendor/rails/activerecord/README.html.

The controller and view are handled by the Action Pack, which handles both layers by its two parts: Action View and Action Controller. These two layers are bundled in a single package due to their heavy interdependence. This is unlike the relationship between the Active Record and Action Pack that is much more separate. Each of these packages can be used independently outside of Rails. You can read more about Action Pack in files/vendor/rails/actionpack/README.html.

Getting Started

1. At the command prompt, start a new Rails application using the `rails` command and your application name. Ex: rails myapp
2. Change directory into myapp and start the web server: `script/server` (run with —help for options)
3. Go to localhost:3000/ and get "Welcome aboard: You're riding the Rails!"
4. Follow the guidelines to start developing your application

Web Servers

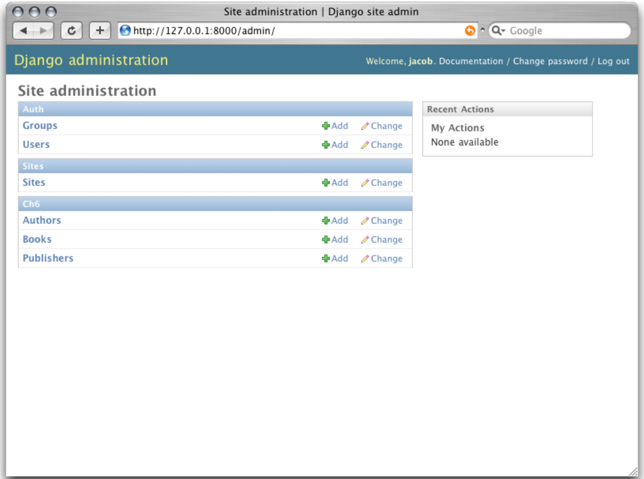
By default, Rails will try to use Mongrel if it's are installed when started with script/server, otherwise Rails will use WEBrick, the webserver that ships with Ruby. But you can also use Rails with a variety of other web servers.

Mongrel is a Ruby-based webserver with a C component (which requires compilation) that is suitable for development and deployment of Rails applications. If you have Ruby Gems installed, getting up and running with mongrel is as easy as: `gem install mongrel`. More info at: mongrel.rubyforge.org

Say other Ruby web servers like Thin and Ebb or regular web servers like Apache or LiteSpeed or Lighttpd or IIS. The Ruby web servers are run through Rack and the latter can either be setup to use FCGI or proxy to a pack of Mongrels/Thin/Ebb servers.

Apache .htaccess example for FCGI/CGI

Django Admin Interface


Site administration | Django site admin
http://127.0.0.1:8000/admin/

Django administration

Welcome, **jacob**. Documentation / Change password / Log out

Site administration

Auth		
Groups	Add	Change
Users	Add	Change
Sites		
Sites	Add	Change
Ch6		
Authors	Add	Change
Books	Add	Change
Publishers	Add	Change

Recent Actions

My Actions

None available

Python APIs

Django has the advantage of:

- High quality Python APIs available for a lot of services.
- Equivalent Ruby APIs not as good quality.

Example

- Mark Pilgrim's Universal Feed Parser v/s rFeedParser
- Ruby parser for kickstart issue

Python APIs

Django has the advantage of:

- High quality Python APIs available for a lot of services.
- Equivalent Ruby APIs not as good quality.

Example

- Mark Pilgrim's Universal Feed Parser v/s rFeedParser
- Ruby parser for kickstart issue

Python APIs

Django has the advantage of:

- High quality Python APIs available for a lot of services.
- Equivalent Ruby APIs not as good quality.

Example

- Mark Pilgrim's Universal Feed Parser v/s rFeedParser
- Ruby parser for kickstart issue

Python APIs

Django has the advantage of:

- High quality Python APIs available for a lot of services.
- Equivalent Ruby APIs not as good quality.

Example

- Mark Pilgrim's Universal Feed Parser v/s rFeedParser
- Ruby parser for kickstart issue

Python APIs

Django has the advantage of:

- High quality Python APIs available for a lot of services.
- Equivalent Ruby APIs not as good quality.

Example

- Mark Pilgrim's Universal Feed Parser v/s rFeedParser
- Ruby parser for kickstart issue

The syncdb Drawback

Syncdb will not alter existing tables

syncdb will only create tables for models which have not yet been installed. It will never issue ALTER TABLE statements to match changes made to a model class after installation. Changes to model classes and database schemas often involve some form of ambiguity and, in those cases, Django would have to guess at the correct changes to make. There is a risk that critical data would be lost in the process.
If you have made changes to a model and wish to alter the database tables to match, use the sql command to display the new SQL structure and compare that to your existing table schema to work out the changes.

– `http://docs.djangoproject.com/en/dev/ref/django-admin`

Alternatives

!?!