redhat

# Newcastle JBug

- https://github.com/mtaylor/jbug
- rvm use jruby@torquebox

# Ruby on Rails on TorqueBox

Martyn Taylor,
Software Engineer, RedHat

redhat

# Overview

- Background
- A look inside
  - Make up
  - Model
  - View
  - Controllers
- TorqueBox
- Running Practical Session

redhat

# What is RoR?


- Ruby Web Framework
- Focus on Convention over Configuration
- MVC
- Open Source (MIT License)

- David Heinemeir Hansson
- Released as Open Source 2004
- > 2100 Contributers
- Version 3.2

redhat

# The Good Stuff

- Rapid Development
- Interpreted*
  - Removes Compile, Build, Deploy Cycle
- Shallow Learning Curve
- Large Ruby Community
  - (> 40000 Gems since 2009)
- Matured Significantly - v3.0 Release
- Applications: Twitter, GitHub, Groupon...
  - (www.rubyonrails.org/applications)

redhat

# The Not so Good Stuff

- Convention only covers 95% Use Case
- Version upgrades often not backward compatible
- Maturity brings Complexity
- Fanatical Community
- No Decent IDE Support
  - Emacs
  - Aptana (eclipse)

Disclaimer:

- The stuff in Red is opinion only; but based on solid facts, for which there are no references or evidence.

redhat

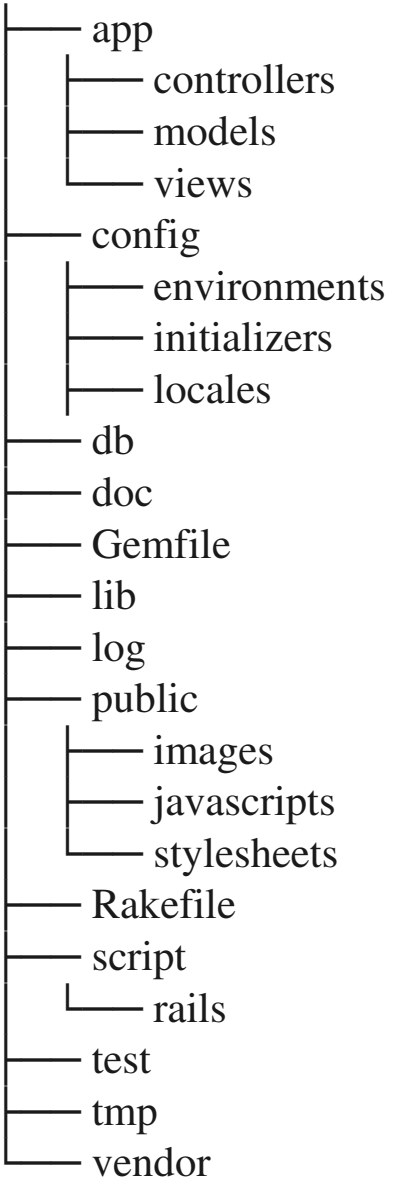
# RVM

- Command Line Tool
- Manages multiple versions of ruby
- Allows creation of library environments:
  - gemsets
- https://rvm.io//

redhat

# Rails application

$ rails new <name>



redhat

# Bundler and Gemfile

- Gemfile
  - Specifies dependencies
  - Gemfile.lock

- Bundler => Rubygem
  - Manages Application Depedencies
  - The de facto standard for rails
  - Determines dependencies from Gemfile
  - Generates Gemfile.lock
    - Locks dependencies and versions

```
$_bundle install
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
```

redhat

# Gemfile Example

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

gem "rails", "3.2.2"
gem "rack-cache"
gem "nokogiri", "~> 1.4.2"

group :development do
  gem "sqlite3"
end

group :production do
  gem "pg"
end
```

RAILS_ENV=production

redhat

# Configuring DB

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

redhat

# Model

- Migrations
- Validation
- Associations
- Querying

redhat

# Migrations

- Database Mappings
- Incremental Changes
- Timestamped
  - Allows easy upgrade
  - Prevents Database Clashes
- Transactional
- Naming Convention
  - YYYYMMDDHHMMSS_<Class.name.underscore>.rb
- Creating Database
  - $ rake db:create:all
  - $ rake db:migrate

redhat

# Migrations

```
class CreateBaseImages < ActiveRecord::Migration
  def self.up
    create_table :base_images do |t|
      t.integer :id
      t.string :name
      t.string :description
      t.integer :template_id
      t.timestamps
    end
  end

  def self.down
    drop_table :base_images
  end
end
```

Called when migrations are run

Called on Rollback

redhat

# Migration Generators

**$ rails generate model User first_name:string dob:date**

app/models/user.rb

```
class User < ActiveRecord::Base
end
```

db/migrate/20120626135824_create_users.rb

```
class CreateUsers < ActiveRecord::Migration
  def self.up
    create_table :users do |t|
      t.string :first_name
      t.date :dob

      t.timestamps
    end
  end

  def self.down
    drop_table :users
  end
end
```

redhat

# Migration Generators

**$ rails generate migration AddEmailToUser**

db/migrate/20120626140751_add_email_to_user.rb

```
class AddEmailToUser < ActiveRecord::Migration
  def self.up
  end

  def self.down
  end
end
```

redhat

# Schema

```
ActiveRecord::Schema.define(:version => 20120626140751) do
  create_table "users", :force => true do |t|
    t.string   "first_name"
    t.string   "last_name"
    t.date     "dob"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
end
```

Matches Migration timestamp

redhat

# Validations

- When is validation performed?
  - Before model is persisted
    - create, create!
    - save, save!
    - update, update!
    - update_attributes
- Raises ActiveRecord::RecordInvalid Exception
- Checking a model is valid
  - valid?
  - Invalid?

redhat

# Validation Example

```
class User < ActiveRecord::Base
  validates :first_name, :presence => true
  validates :postcode,
            :format => { :with => /^([A-Z][A-Z][0-9]? [0-9][A-Z]{2})$/,
                         :message => "must be a valid UK Postcode" }
end
```

>> User.create(:first_name => “Martyn”, :postcode => “Bad Post Code”)

=> ActiveRecord::RecordInvalid: Validation failed: must be a valid UK Postcode

>> User.create(:postcode => “NE1 1RU”)

=> ActiveRecord::RecordInvalid: Validation failed: FirstName can't be blank

redhat

# Conditional Validations

```
class Order < ActiveRecord::Base
  validates :card_number, :presence => true, :if => :paid_with_card?

  def paid_with_card?
    payment_type == "card"
  end
end
```

redhat

# Custom Methods

```
class Person < ActiveRecord::Base
  validate :must_be_british

  def must_be_british
    [“England”, “Scotland”, “Ireland”, “Wales”].include?(record)
  end

end
```

redhat

# Associations

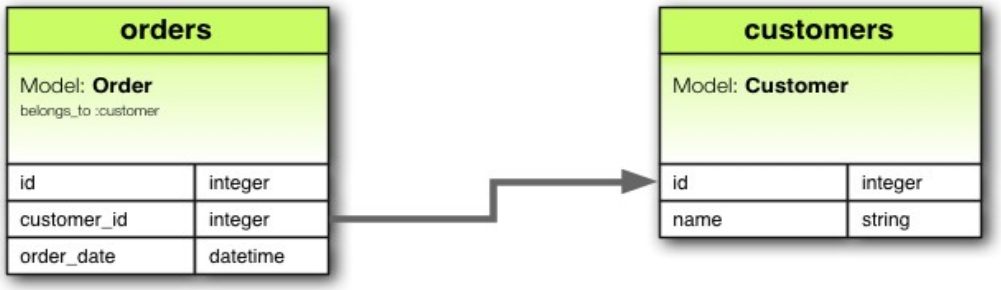
- Model Relationships
  - belongs_to
  - has_one
  - has_many
  - has_many :through
  - has_one :through
  - has_and_belongs_to_many

redhat

# has_one and belongs_to

- 1 to 1 relationships
- Can specify 1-way or bi-directional relationship

- if table contains foreign_key
  use has_one
  else
  use belongs_to



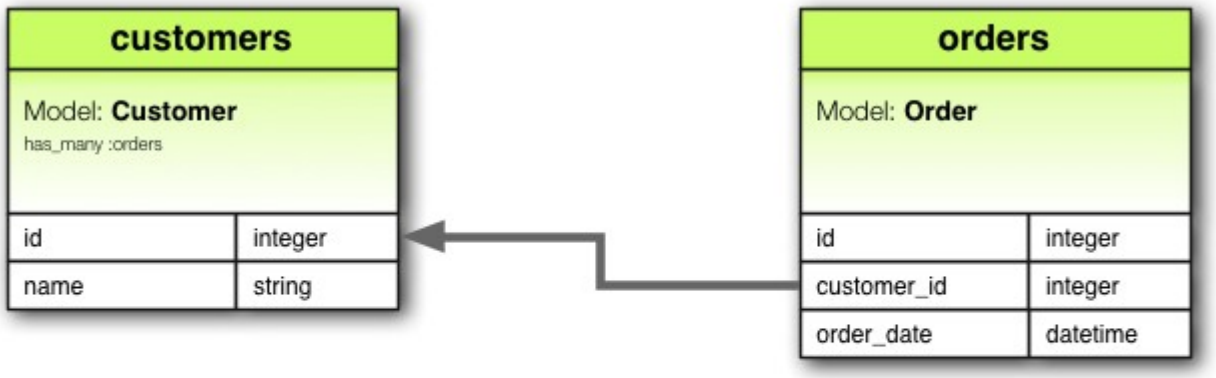

```ruby
class Order
  belongs_to :customer
end
```

```ruby
class Customer
  has_one :order
end
```

redhat

# has_many

- Specify 1 -> many




```
class Customer
  has_many :orders
end
```

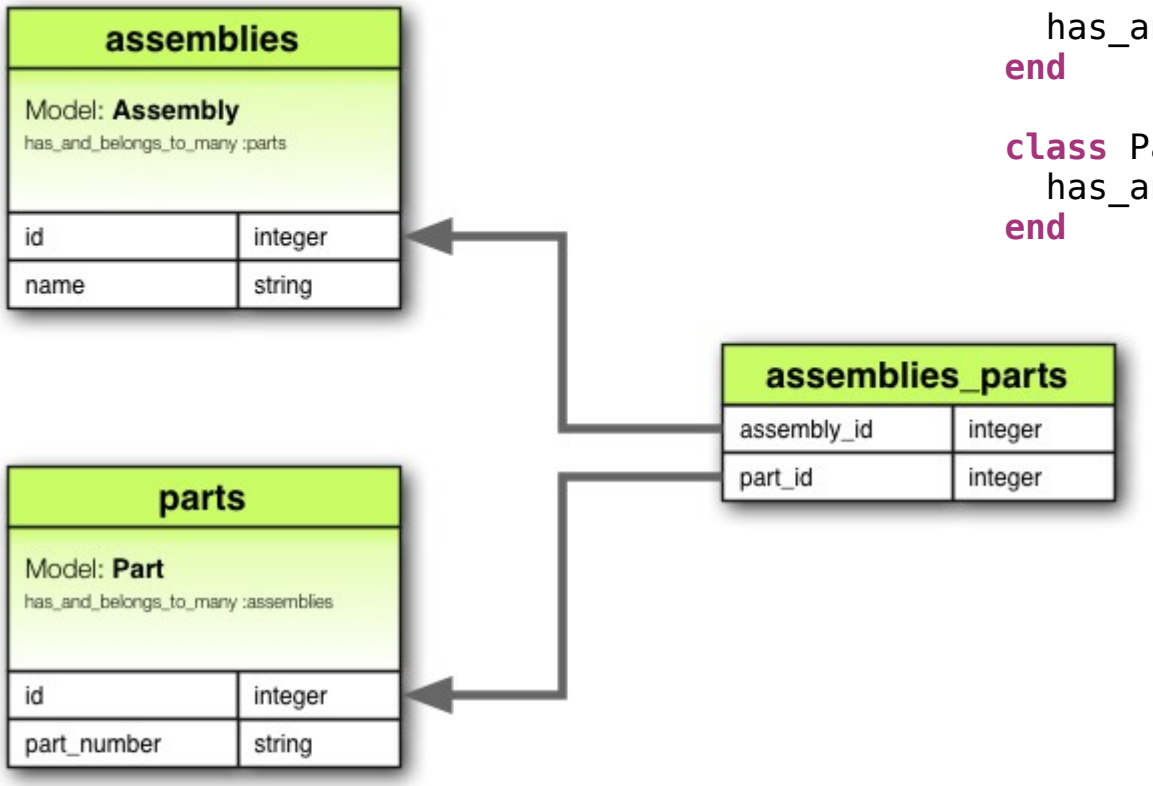
```
class Order
  belongs_to :customer
end
```

Notice the plural

redhat

# has_and_belongs_to_many

- Many to many relationships



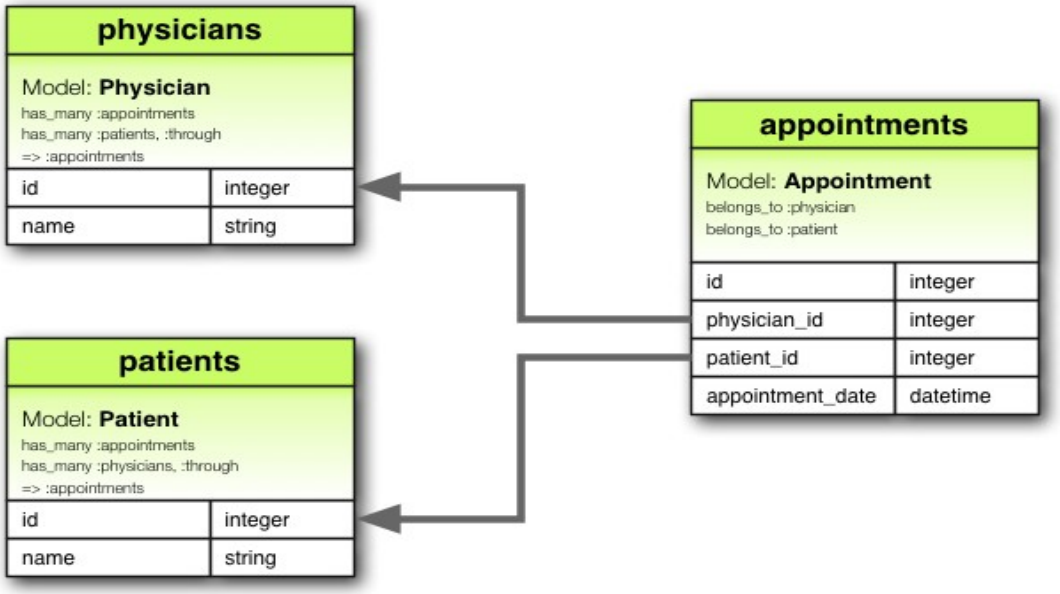

```
class Assembly < ActiveRecord::Base
  has_and_belongs_to_many :parts
end

class Part < ActiveRecord::Base
  has_and_belongs_to_many :assemblies
end
```

redhat

# => through

- Specifies relationships through 3rd model
- has_one, has_many




```ruby
class Physicians
  has_many :appointments
  has_many :patients,
           :through => :appointments
end

class Patients
  has_many :appointments
  has_many :physicians,
           :through => :appointments
end

class Appointment
  belongs_to :physician
  belongs_to :patient
end
```

redhat

# Active Record Queries

```
product = Product.find(10)
```

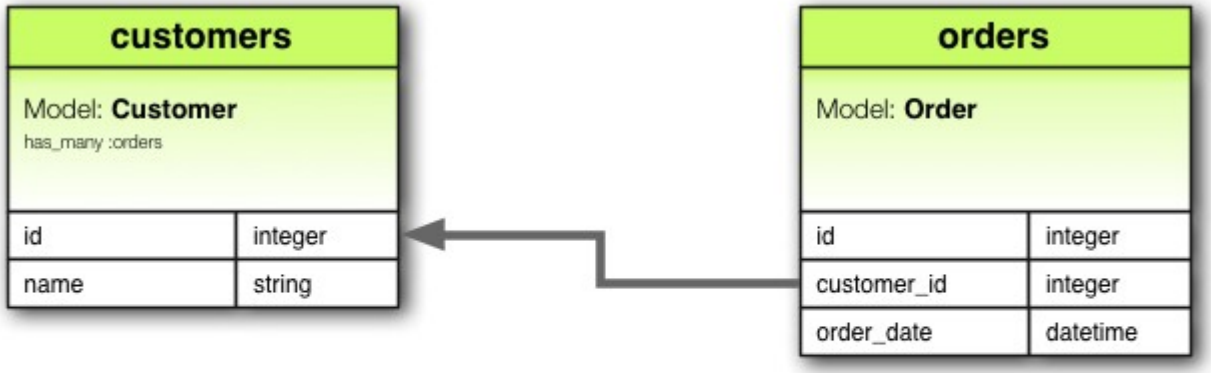
```
SELECT * FROM products WHERE (products.id = 10) LIMIT 1
```

redhat

# Dynamic Finder Methods

- Creates finder methods based on class attributes

- find_by_
- find_first_by_
- find_last_by_
- find_all_by_




```
customer = Customer.find_by_name("Martyn")
```

redhat

# Finder Conditions

```
customer = Customer.find_by_name(“Martyn”)
```

```
customer = Customer.where(“name = ? AND age = ?”, “Martyn”, 27)

customer = Customer.where(:name => “Martyn”, :age => 27)

customer = Customer.where(“age < ?”,  27)
```

redhat.

# Face Blog

redhat

# Rails Routing

- Matches URL -> Controller Actions
- Generates Paths and URLs
- Specify CRUD operations
- Nested Resources
- Redirection
- Constraints

redhat

# Rails Routing

```
MyApplication::Application.routes.draw do
  root :to => 'pages#main'

  resources :products

  resources :order do
    resources :products
  end

  match "/sale_items" => redirect("/products")

  resources :users, :constraints => {:id => /[A-Z][A-Z][0-9]+/}

end
```

redhat

# Rake Routes

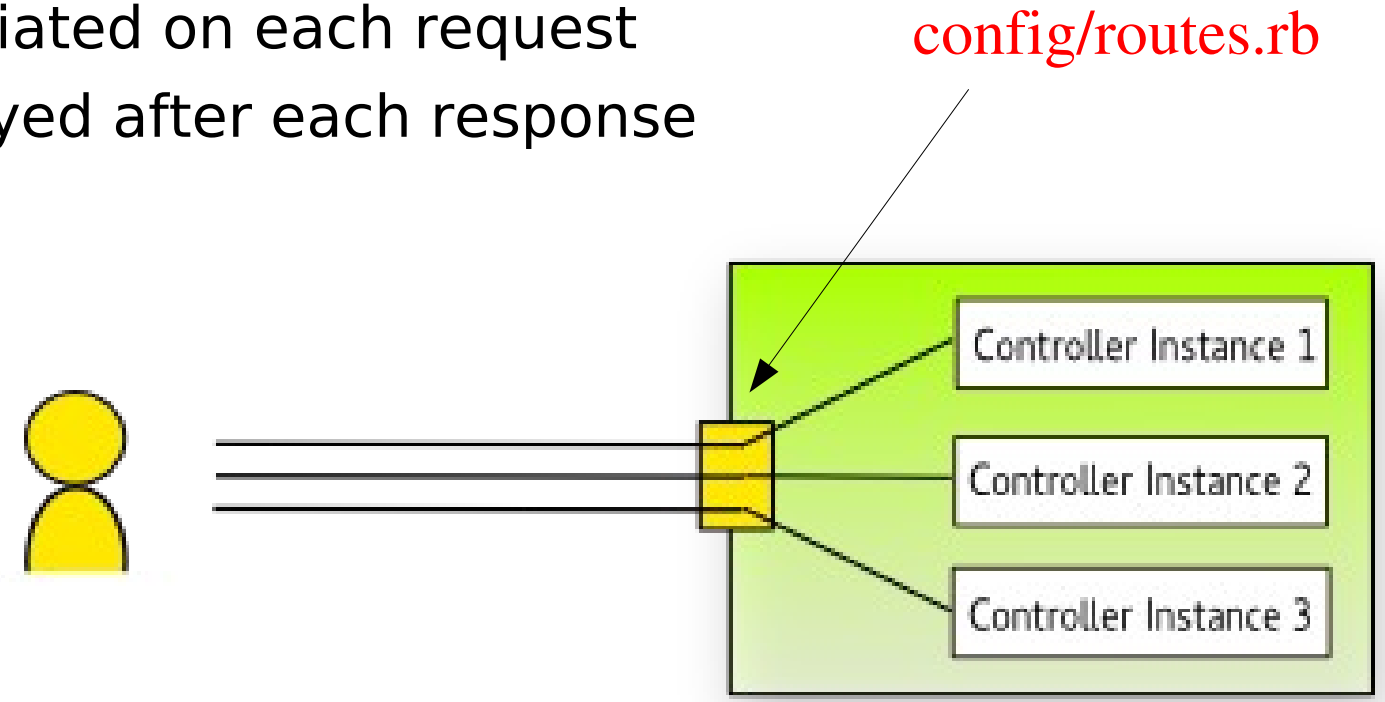
$ rake routes

```
          root        /(.:format)                                {:controller=>"pages", :action=>"main"}
      products GET    /products(.:format)                        {:controller=>"products", :action=>"index"}
               POST   /products(.:format)                        {:controller=>"products", :action=>"create"}
   new_product GET    /products/new(.:format)                    {:controller=>"products", :action=>"new"}
  edit_product GET    /products/:id/edit(.:format)               {:controller=>"products", :action=>"edit"}
       product GET    /products/:id(.:format)                    {:controller=>"products", :action=>"show"}
               PUT    /products/:id(.:format)                    {:controller=>"products", :action=>"update"}
               DELETE /products/:id(.:format)                    {:controller=>"products", :action=>"destroy"}
order_products GET    /order/:order_id/products(.:format)        {:controller=>"products", :action=>"index"}
```

redhat

# Controller

- Extends ApplicationController
- Methods mapped to routes
- Convention to use RESTful Routes
- Instantiated on each request
- Destroyed after each response



redhat

# Retreving Params

- params() available in each controller
- params() => returns hash of data
- Automatically converts content using content-type header
- Query params and url params are treated the same

```xml
<order>
  <product id='1' />
  <quantity>2</quantity>
  <cost>
    <amount>25</amount>
    <currency>GBP</currency>
  </cost>
</order>
```

```
{
  :order => {
    :product => {
      :id => '1'
    },
    :quantity => 2,
    :cost {
      :amount => 25,
      :currency => "GBP"
    }
  }
  # Extra Meta Data Here
}
```

redhat

```ruby
class OrdersController < ApplicationController

  def index
    @orders = Order.all

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @orders }
    end
  end

  def show
    @order = Order.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @order }
    end
  end
end
```

redhat

# Sessions

- Available through, *session()* instance method
- Key/value store
- Lazy Loaded
- Scoped across controllers and views
- To create => call session
- To destroy => set to nil

```
# First time session is called. (A new session is created here)
session[:user_id] = authenticate_user().id

current_user = User.find(session[:user_id])

# Destroyed
session = nil
```

redhat

# eRuby

- Embeds Ruby code inside text documents
- Equivilent to JSP, ASP etc...
- Default in Rails
- Accesses global variables set in controller

redhat

# Controller Triggers

- render
  - Renders
    - View
    - Text
    - xml, json
- redirect_to
  - Sends redirect to browers
- head
  - Returns only headers

redhat

controller

```
class OrdersController < ApplicationController

  def index
    @orders = Order.all

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @orders }
    end
  end
```

view

```
<h1>Listing orders</h1>

<% @orders.each do |order| %>
  <tr>
    <td><%= order.name %></td>
    <td><%= link_to 'Show', order %></td>
```

redhat

# Partials

- Partials can be rendered inside views
- Partials start with underscore
  - _order_partial.html.erb
- Parameters can be passed into partials from views

```
<h1>Listing orders</h1>

<table>
  <tr>
    <th>Name</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @orders.each do |order| %>
  <%= render :partial => "order.rb", :locals => { :order => order } %>
```

redhat

# Layouts

- Can be used to structure views
- Views are rendered and inserted into layouts
- Default site layout: layouts/application.html.erb
- Yield renders view

redhat

# Layout Example

layout

```
<html>
  <head>
  </head>
  <body>
    <%= yield %>
  </body>
</html
```

view

```
<title>A simple page</title>

<p>Hello, Rails!</p>
```

resulting html

```
<html>
  <head>
  </head>
  <body>
    <title>A simple page</title>

    <p>Hello, Rails!</p>
  </body>
</html
```

redhat

# Html Helpers

```
<%= form_tag("/search", :method => "get") do %>
  <%= label_tag(:q, "Search for:") %>
  <%= text_field_tag(:q) %>
  <%= submit_tag("Search") %>
<% end %>
```

```
<form accept-charset="UTF-8" action="/search" method="get">
  <label for="q">Search for:</label>
  <input id="q" name="q" type="text" />
  <input name="commit" type="submit" value="Search" />
</form>
```

redhat

# HTML Helpers

```
<%= form_for @article, :url => { :action => "create" },
                       :html => {:class => "nifty_form"} do |f| %>
  <%= f.text_field :title %>
  <%= f.text_area :body, :size => "60x12" %>
  <%= f.submit "Create" %>
<% end %>
```

```
<form accept-charset="UTF-8" action="/articles/create" method="post">
  <input id="article_title" name="article[title]" size="30" type="text" />
  <textarea id="article_body" name="article[body]" cols="60" rows="12"></textarea>
  <input name="commit" type="submit" value="Create" />
</form>
```

redhat.

# Face Blog 2

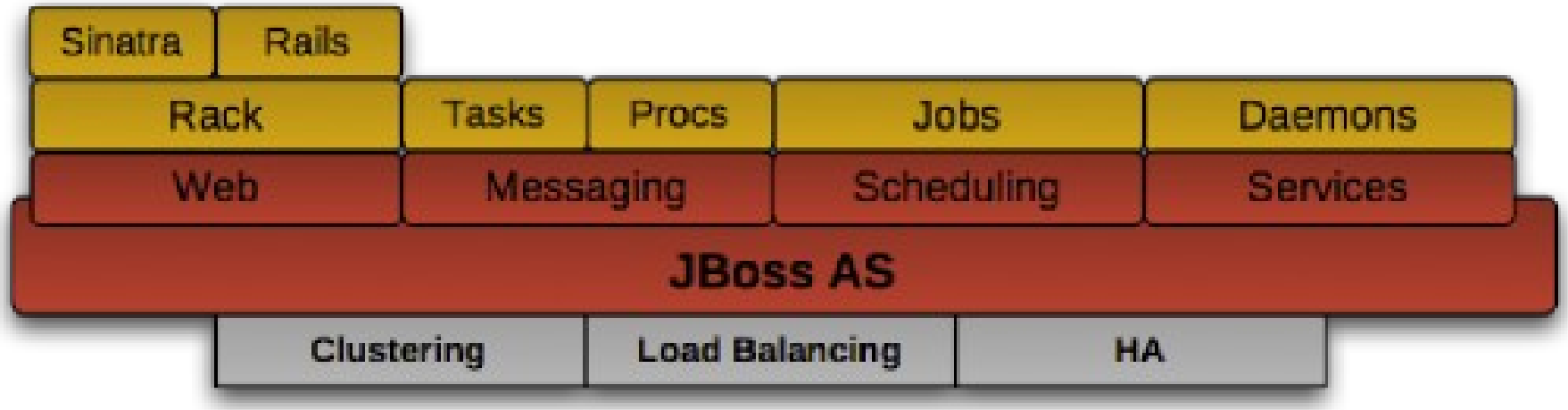
redhat

# TorqueBox

- Ruby Application Server
  - Sinatra, Rack, RoR
- Built on top of JBossAS
  - “Ninja Grade”
  - Clustering
  - High Performance
- Extra functionality
  - Messaging
  - XA Transactions

redhat

# TorqueBox



redhat

# TorqueBox Messaging

- Deploying Destinations

  torquebox.yml

```
application:
  ..
queues:
  /queues/my_app_queue:
    durable: false
topics:
  /queues/my_app_topic:
```

redhat

# TorqueBox Messaging

## Publish

```
@queue = inject '/queues/foo'
@queue.publish "A text message"

@topic = inject '/topics/foo'
@topic.publish "A text message"
```

## Receive/Subscribe

```
@queue = TorqueBox::Messaging::Queue.new('/queues/foo')
@message = queue.receive

@topic = TorqueBox::Messaging::Topic.new('/topics/foo')
@message = topic.receive
```

redhat

# Message Processor

- Ruby MDB

```
class MyConsumer < TorqueBox::Messaging::MessageProcessor
  def on_message(body)
  end

  def on_error(exception)
  end
end
```

application:
  ..
queues:
  /queues/my_app_queue:  MyConsumer

redhat.

# Face Blog 3