VELA SYSTEMS

The Leader in Field Software for the AECO Industry™

Ruby on Rails

Frank Tino

Some Definitions

- **Ruby**
  - Programming language
  - Object-oriented
  - Interpreted

- **Rails**
  - A framework written in Ruby
  - Designed for writing RIA’s
  - Enforces MVC architecture
  - Extensible via gems

Some Basic Ruby

- **Highly readable (no semicolons, untyped)**

```
def say_goodnight (name)
    result = “Good night, #{name.capitalize}”
    return result
end
# comment
puts say_goodnight(“gracie”)
```

- **Supported Types:**
  - Numbers, Strings, Ranges, Regular Expressions

- **Heavy reliance on containers**
  - Arrays

```
a = [‘dog’, ‘cat’]
a[0] -> ‘dog’
```

  - Hashes

```
animals = {:dog => ‘canine’, :cat => ‘feline’}
animals[:dog] -> ‘canine’
```

More Basic Ruby

- **Intuitive looping**
  - 3.times {print “x ”}
  - 1.upto(5) {|i| print i, “ “}
  - until a < 100 {a -= 10}
  - for song in songlist {song.play}

- **Standard but flexible control structures**
  - if…elsif…else…end
  - puts “Danger, Will Robinson” if radiation > 3000
  - unless a < 0 then puts a end

- **Inheritance**

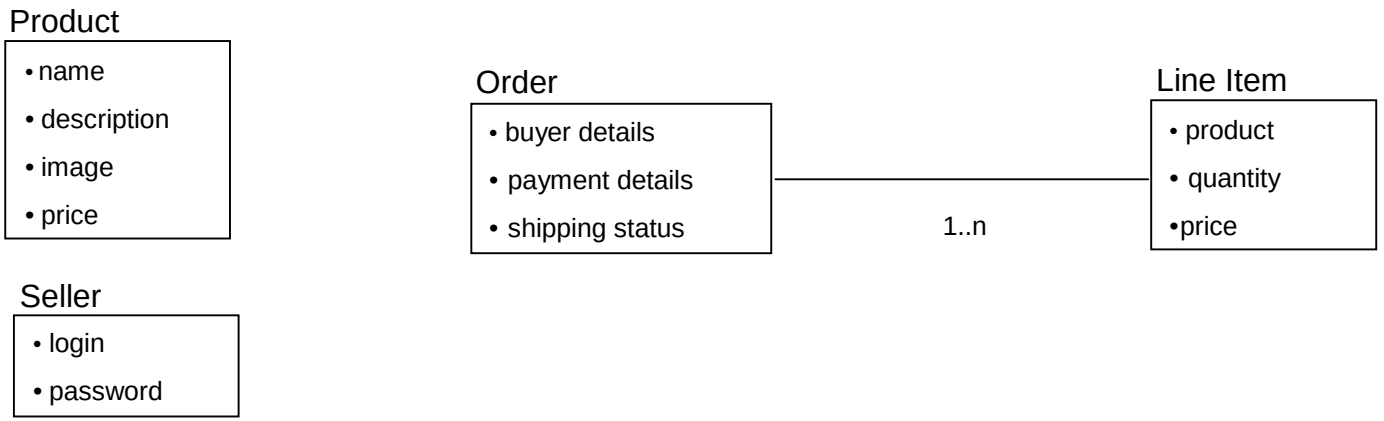
```
class KaraokeSong < Song
    def initialize(name, artist, lyrics)
      super(name, artist)
      @lyrics = lyrics
    end
end
```

  - No need for multiple inheritance due to module inclusion

Rails – Let's Build an App!

- **Depot Application**
  - Allows sellers to post products to sell
  - Allows buyers peruse a product listing and select items to purchase

- **Database Structure**



Rails – Setting up the model

- **Need to configure the adapter to talk to the database**
  - Simple yml file
  - Supports development, test, and production configurations

- **“ruby script/generate model product”**

- **Generates migration file**

```
class CreateProducts < ActiveRecord::Migration
    def self.up
        create_table :products do |t|
            t.column :title, :string
            t.column :description, :text
            t.column :image_url, :string
            t_column :price, :decimal, :precision => 8, :scale => 2, :default => 0
        end
    def self.down
        drop_table :products
    end
end
```

- **Generates model file**

```
class Product < ActiveRecord::Base
    validates_presence_of :title, :description, :image_url
    validates_numericality_of :price
end
```

Rails – Setting up the controller

- **Could use the built-in rails scaffold to get the app up and running**

```
class AdminController < ApplicationController
    scaffold :product
end
```

  - http://localhost:3000/admin will display a listing products and generate the forms to add new ones

- **"ruby script/generate controller store list"**

- **Generates controller file**

```
Class StoreController < ApplicationController
    def list
        @products = Product.find(:all, :order => "title")
    end
end
```

  - http://localhost:3000/store/list will now hit our controller method
  - However this will fail because we haven't set up the view yet

Rails – Setting up the view

- **Unless told otherwise the controller will try to render the view named after the method (list in this case)**

- **list.rhml generated by the controller generate command**

```
<h1>Product Catalog</h1>
<% for product in @products %>
    <div class="entry">
        <%= image_tag(product.image_url) %>
        <h3><%= product.title %></h3>
        <%= product.description %>
        <span class="price"><%= number_to_currency(product.price) %></span>
        <%= button_to "Add to Cart", :action => :add_to_cart, :id => product.id %>
    </div>
<% end %>
```

- **We can now add a method to the store controller to handle the Add to Cart**

```
def add_to_cart
    session[:cart] ||= Cart.new
    product = Product.find(params[:id])
    session[:cart].add_product(product)
end
```

Rails – Other Amazing Things

- **Relationships between models**

```
class Order < ActiveRecord::Base
    has_many :line_items
end
class LineItem < ActiveRecord::Base
    belongs_to :order
end
```

- **Allows you to do cool things like:**

```
li = LineItem.find(…)
puts “This line item was bought by #{li.order.name}”

Order = Order.find(…)
puts “This order has #{order.line_items.size} line items”
```

Rails – Other Amazing Things

- **Form Processing**
  - Order Controller

```
def edit
  @order = Order.find(param[:id])
end
```

  - Order View

```
<% form_for :order :action => :save_order do |form| %>
  <label>Name</label>
  <%= form.text_field :name, :size => 40 %>
<% end %>
```

  - Rails will create an HTML form and automatically populate it with the elements from the database!

  - Saving back to the database after the form submit is as simple as:

```
def save_order
  order = Order.find(params[:id])
  order.update_attributes(params[:order])
end
```

Getting Started

- **Pieces**
  - Ruby interpreter
  - Ruby on Rails
  - Database
- **Windows**
  - InstantRails (www.**instantrails**.rubyforge.org/)
- **Mac OS X**
  - Ruby already included!
  - gem install rails
  - Install mysql or your db of choice
- **Development Environment**
  - Most developers just use their favorite text editor
  - Good IDEs
    - RubyMine (www.jetbrains.com)
    - Aptana Studio (www.aptana.com)

Rails – Great References

- **“Programming Ruby” by Dave Thomas**
- **“Agile Web Development with Rails” by Dave Thomas & David Heinemeier Hanson**
- **“The Rails Way” by Obie Fernandez**
- **http://www.rubyonrails.org**