MVC Frameworks

MVC Frameworks

- What is MVC?
  - Model – View – Controller
  - Architectural and Design Pattern
  - Described in 1979 by Trygve Reenskaug who was working on SmallTalk at Xerox PARC
- MVC – Then and Now
  - “Rediscovered” for web app development

Quotes from Trygve Reenskaug

- “MVC was conceived as a general solution to the problem of users controlling a large and complex data set.”
- “The hardest part was to hit upon good names for the different architectural components. Model-View-Editor was the first set.”

MVC Architecture – Model Layer

- Corresponds to the database – some form of data persistence
- Can be a real database like MySQL, PostgreSQL, etc.
- Can alternatively be an XML file, flat files, etc.

MVC Architecture – Model Layer (2)

- Decouple the data storage and retrieval from the other aspects such as the UI
- UI does not change depending on whether the data comes from an XML file or from an Oracle DB
- Central place to do all the validations such as integrity constraints and null checks

MVC Architecture – View Layer

- Corresponds to the User Interface
- For web apps, this is typically a web page
- The web page designer need not be concerned about things like business logic
- Programmers typically use tools like Eclipse and emacs; Web page designers use different tools like Adobe Dreamweaver
- Allow the web page designers to use whatever they are comfortable with

MVC Architecture – Controller Layer

- Corresponds to the “business logic”
- Theoretically lets the programmers use any language they are comfortable with
  - There are no dependencies with the View or the Model Layers
- In practice, this is not true as picking an MVC framework forces you to use a fixed programming language


TURBO
GEARS

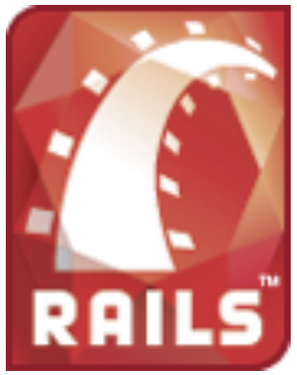

tapestry


symfony


CakePHP


Stripes


Struts


RAILS


pylons
POWERED


Spring


merb


seaside


django


catalyst
Web Framework


JBoss
a division of Red Hat


Cappuccino


nitro

Ruby on Rails

- Web Application Framework created by David Heinemeier Hansson (DHH) at 37signals
- Extracted from real-world web application called Basecamp and made open source in 2004
- Some 37signals applications
  - Basecamp (project management)
  - Ta-Da List (personal todo list)
  - Campfire (business oriented online chat service)

Ruby on Rails

- Uses Ruby
- Ruby is a dynamic, object-oriented programming language
- Created by Yukihiro Matsumoto (Matz) in 1995
- Based on Perl, Smalltalk, Eiffel, Ada, and Lisp
- Supports multiple programming paradigms – functional, OO, imperative, etc.
- Strong support for reflection and Metaprogramming

Design Philosophy of Ruby

- “I wanted a language more powerful than Perl and more object-oriented than Python. Then, I remembered my old dream and decided to design my own language.” – Matz
- Principle of Least Surprise
- Make programming fun!

Sample Ruby Code

factorial.rb

New Open Recent Save Undo Redo Cut Copy Paste Help

```
def factorial(n)
  if (n <= 1) then 1
  else n * factorial(n-1)
  end
end
```

-:** factorial.rb All (6,0) (Ruby)

Beginning of buffer

Design Philosophy of Ruby on Rails

- Don't Repeat Yourself (DRY)
  - Very Little Duplication
  - "Every piece of knowledge in a system should be expressed in just one place"
- Convention over Configuration
  - Sensible Defaults for Everything
  - "Follow the conventions and you can write a Rails application using less code than a typical Java web application uses in XML configuration"

Design Philosophy of Ruby on Rails

- Inspired other MVC frameworks
- Most notable ones include
  - Symfony
  - CakePHP
  - PHP on TRAX
  - Merb

Ruby on Rails – Model Layer

- Active Record is the default Model Component in Rails and is the Base Class for all models
- Provides Object-Relational Mapping (ORM)
  - Mapping between tables in the database and the classes in the application
  - Classes correspond to Tables
  - Attributes correspond to columns of the table
  - Objects correspond to rows of the table
- Provides database independence, basic CRUD functionality, advanced finding capabilities, etc.

Ruby on Rails – Model Layer (2)

```ruby
# == Schema Information
#
# Table name: albums
#
#  id         :integer          not null, primary key
#  name       :string(255)
#  price      :float
#  release    :date
#  created_at :datetime         not null
#  updated_at :datetime         not null
#

class Album < ActiveRecord::Base

  validates :price, :presence => true
  validates :name, :length => {:minimum => 2}

  has_many :songs
end
```

Ruby on Rails – View Layer

- Action View manages the views in Rails applications
- Can create both HTML and JSON output by default
- Manages rendering templates, including nested and partial templates, and includes built-in AJAX support
- Can embed Ruby code in HTML for the View Layer (similar to JSPs, etc.)

Ruby on Rails – View Layer (2)

```
1   <h1>Listing playlists</h1>
2
3   <table>
4     <tr>
5       <th>Title</th>
6       <th>Description</th>
7       <th></th>
8       <th></th>
9       <th></th>
10    </tr>
11
12  <% @playlists.each do |playlist| %>
13    <tr>
14      <td><%= playlist.title %></td>
15      <td><%= playlist.description %></td>
16      <td><%= link_to 'Show', playlist %></td>
17      <td><%= link_to 'Edit', edit_playlist_path(playlist) %></td>
18      <td><%= link_to 'Destroy', playlist, method: :delete, data: {
    confirm: 'Are you sure?' } %></td>
19    </tr>
20  <% end %>
21  </table>
22
23  <br />
24
25  <%= link_to 'New Playlist', new_playlist_path %>
26
```

Ruby on Rails – Controller Layer

- Action Controller manages the controllers in a Rails application
- The Action Controller framework processes incoming requests to a Rails application, extracts parameters, and dispatches them to the intended action
- Services provided by Action Controller include session management, template rendering, and redirect management.

Ruby on Rails – Controller Layer (2)

```ruby
class PlaylistsController < ApplicationController
  # GET /playlists
  # GET /playlists.json
  def index
    @playlists = Playlist.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @playlists }
    end
  end
end
```