Server-side Web Development & Ruby on Rails

John Wu, Jan Stelovsky
Server-Side Environment

- **PHP**
  - the granddaddy
  - wealth or resources (Worldpress)
  - cumbersome to use

- **Ruby on Rails (RoR, Rails)**
  - active community
  - increasingly popular

- **jNode**
  - based on JavaScript and CoffeeScript
  - active community
  - not as structured as Rails

- **Django**
  - Python based
  - increasingly popular
Ruby

• Yukihiro Matsumoto, 1995
• An interpreted programming language
  – Not compiled
• Loosely typed (weak typing)
• Between functional & object-oriented
  – Everything is an object
  – Functional features
    • 1st class citizens, closures, yield
• Gem – a self-contained ruby code library
  – rubygems.org Many useful codes
Rails

• David Heinemeier Hansson, 2003
• A web application development framework
• Model-View-Controller design (MVC)
• REpresentational State Transfer (REST) architecture
• “Allows to write less code while accomplishing more than many other languages and frameworks”
  guides.rubyonrails.org/getting_started.html
Rails

• Many conventions
  – if you obey conventions your life will be easy
• Scaffolding
• Assets pipeline
• CoffeeScript
• LESS (cascading style sheets)
• Testing
Resources

• Books
  • Ruby on Rails™ 3 Tutorial by Michael Hartl
  • Everyday Scripting with Ruby by Brian Marick
  • Metaprogramming Ruby by Paolo Perrotta
• Web
  • guides.rubyonrails.org/
  • rubygems.org
  • www.google.com
  • www.stackoverflow.com forum
# Routing

- `/config/routes.rb`
- RESTful routes
- `resources :user`

<table>
<thead>
<tr>
<th>Verb</th>
<th>Path</th>
<th>action</th>
<th>used for</th>
<th>Path Helper</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td><code>/users</code></td>
<td>index</td>
<td>display a list of all users</td>
<td><code>users_path</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/users/new</code></td>
<td>new</td>
<td>return an HTML form for creating a new user</td>
<td><code>new_users_path</code></td>
</tr>
<tr>
<td>POST</td>
<td><code>/users</code></td>
<td>create</td>
<td>create a new user</td>
<td><code>users_path</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/users/:id</code></td>
<td>show</td>
<td>display a specific user</td>
<td><code>user_path(:id)</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/users/:id/edit</code></td>
<td>edit</td>
<td>return an HTML form for editing a user</td>
<td><code>edit_user_path(:id)</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/users/:id</code></td>
<td>update</td>
<td>update a specific user</td>
<td><code>user_path(:id)</code></td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/users/:id</code></td>
<td>destroy</td>
<td>delete a specific user</td>
<td><code>user_path(:id)</code></td>
</tr>
</tbody>
</table>
Database

• Migrations
  – Describe the creation and modification of the database schema
  – `bundle exec rake db:migrate`
    Executes the migrations
  – Rollback
  – MySQL, PostgreSQL, SQLite, SQL Server, Sybase, and Oracle
    • all common databases except DB2
Model

• ActiveRecord
  – An object-relational mapping system for database access
  – Super class, does all the work
• We define the relationships (RDB) and validations in the class
  – class User < ActiveRecord::Base
    has_many :videos
    validates :name,
      :presence => true,
      :length => { :maximum => 60 }
  – class Video < ActiveRecord::Base
    belongs_to :user
Model: ActiveRecord

• `has_many automatically make Video.user_id the foreign key for User.id`
  – foreign key is the lowercase class name, i.e. `user` followed by `_id`

• A layer on top of SQL
  – `User.all`
    • `SELECT * FROM users`
  – `@user.videos`
    • `SELECT * FROM videos WHERE videos.user_id = @user.id`
Controller

- Actions (functions)
- Prepare model data for use by the view
- Instance variables, @user, is accessible in subsequent view/partials
- Navigation
  - render(view/path, options_hash)
    - Can also render JSON, XML, JS, etc.
  - redirect_to(view/path, options_hash)
View

• erb (Embedded RuBy) files
  – HTML files with Ruby code inserted
  – Will be evaluated by Rails to render a final HTML page
• File structure organized by controllers
• Rendering: layout => view => partials
View

- Partials
  - Not for controllers. Just erb files in views.
  - `<%= render 'index' %>` ➔ `_index.html.erb`
  - Break the rendering of a single page to multiple files
  - More manageable, reusable
    - Sort of like subroutines
    - Can pass in variables
      - `<%= render 'index', :name => 'John' %>`
Model: Mass-Assignment

• Model can be created by a request
  – A form:

```ruby
<%= form_for @user do |f| %>
  <%= f.text_field :name %>
  <%= f.submit "Set Name" %>
<% end %>
```
  – In controller:

```ruby
@user.update_attributes(params[:user])
```

• Mass-assignment. Insecure!
Model: Mass-Assignment

- `attr_accessor :name` ➔ Ruby’s getter/setter
- `attr_accessible :name`
  - Allow mass-assignment only on specified attributes
  - Advised to do this for all model classes
  - The following won’t work:
    ```ruby
    @user.update_attributes(:pwd => "abcd")
    ```
  - Because `pwd` is not included in `attr_accessible`