Open Source Frameworks for Rapid Application Development

Marek Krętowski
Krzysztof Bandurski, Tomasz Łukaszuk, Tomasz Rybak

Software Departament
Faculty of Computer Science
Bialystok University of Technology

m.kretowski@pb.edu.pl
k.bandurski@pb.edu.pl, t.lukaszuk@pb.edu.pl, t.rybak@pb.edu.pl

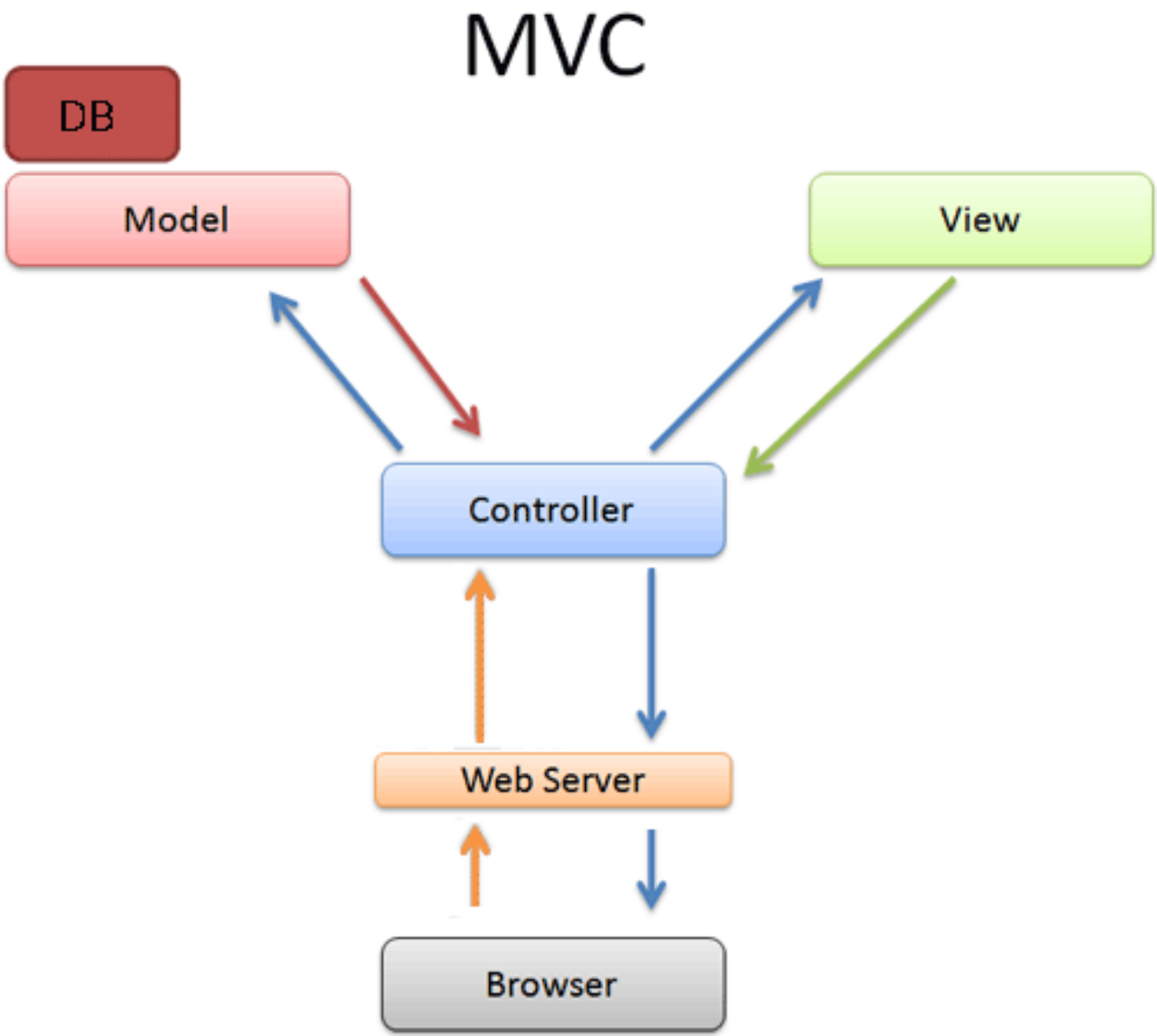
Lecture topic

Views in Ruby on Rails

Views in Ruby on Rails: Table of content

Introduction

View in MVC pattern



Views in RoR

- ActionController::Base#render method
- ActionView module
- app/views/controller_name folder
- action_name.html.erb files
- app/views/layouts folder

Rendering

Creating responses I

- three ways to create an HTTP response: render, redirect_to, head
- by default, controllers in Rails automatically render views with names that correspond to actions

```
1  class BooksController < ApplicationController
2     def show
3        @book = Book.find(params[:id])
4        #automatically render app/views/books/show.html.erb
5     end
6  end
```

- render method cases

```
1  render :nothing => true
2  render "name"     #action (template) in the same controller
3  render :template => 'controller_name/template_name'
4  render :file => "/path/to/file/filename"
5  render :text => "ok"
6  render :xml => @product
```

Creating responses II

- options for render:
  :content_type
  :layout
  :status
  :location

```
1  render :file => filename, :content_type => 'application/rss'
2  render :layout => 'special_layout'
3  render :status => 500
4  render :xml => photo, :location => photo_url(photo)
```

Avoiding double render errors

- error message ”Can only render or redirect once per action”
- example of error

```
1  def show
2     @book = Book.find(params[:id])
3     if @book.special?
4        render :action => "special_show"
5     end
6  end
```

- using and return

```
1  def show
2     @book = Book.find(params[:id])
3     if @book.special?
4        render :action => "special_show" and return
5     end
6  end
```

Passing parameters from controller to template

- instance variables (begins with @) of controller are available in the rendered template

Listing 1: app/controllers/books_controller.rb

```
1  class BooksController < ApplicationController
2     def index
3        @library = "University Library"
4        @books = Book.all
5     end
6  end
```

Listing 2: app/views/books/index.html.erb

```
1  <%= @library %><br />
2  <% @books.each do |book| %>
3     <%= h book.author %>: <%= h book.title %><br />
4  <% end %>
```

Views files

View file content

- HTML with embedded Ruby code
- special tags in HTML
  <% ruby code %>
  <%= ruby expression %>
  <%# comment %>

```
1  <%# view file example %>
2  <h1><%= @name %> (<%= @code %>)</h1>
3  <p><%= @desc %></p>
4
5  <ul>
6     <% @features.each do |f| %>
7        <li><b><%= f %></b></li>
8     <% end %>
9  </ul>
```

Layouts

What is layout?

- layout - defines the surroundings of an HTML page
- layout - the place to define common look and feel of your final output
- layout files reside in app/views/layouts

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html;charset=UTF-8" />
  <title>Books: <%= controller.action_name %></title>
  <%= stylesheet_link_tag 'scaffold' %>
</head>
<body>

<p style="color: green"><%= flash[:notice] %></p>

<%= yield %>

</body>
</html>
```

Finding layouts

```
1  class BooksController < ApplicationController
2     def show
3        @book = Book.find(params[:id])
4     end
5  end
```

- Rails looks for a file in app/views/layouts with the same base name as the controller /app/views/layouts/books.html.erb
- if there is no such controller-specific layout, Rails will use /app/views/layouts/application.html.erb

Assigning specific layouts I

- specifying layouts on a per-controller basis

```
1  class BooksController < ApplicationController
2     layout "spec_layout"
3     #...
4  end
```

- choosing layouts at runtime

```
1  class ProductsController < ApplicationController
2     layout :products_layout
3
4     def show
5        @product = Product.find(params[:id])
6     end
7
8    private
9      def products_layout
10        @current_user.special? ? "special" : "products"
11     end
12 end
```

Assigning specific layouts II

- conditional layouts

```
1  class ProductsController < ApplicationController
2     layout "inventory", :only => :index
3     layout "product", :except => [:index, :rss]
4     #...
5  end
```

- specifying layout for action

```
1  class BooksController < ApplicationController
2     def show
3        @book = Book.find(params[:id])
4        render :layout => "special"
5     end
6  end
```

- layout inheritance

Structuring layouts

- asset tags
- yield and content_for
- partials

Asset tags

- provide methods for generating HTML that links views to assets like images, javascript, stylesheets, and feeds
  - auto_discovery_link_tag
  - javascript_include_tag
  - stylesheet_link_tag
  - image_tag
  - vidio_tag
  - audio_tag
- you can use these tags in layouts or other views
- the asset tags do not verify the existence of the assets at the specified locations

javascript_include_tag

```
1  <%= javascript_include_tag "main" %>
2  #public/javascripts/main.js
3
4  <%= javascript_include_tag "main", "/photos/columns" %>
5  #public/javascripts/main.js and public/photos/columns.js
6
7  <%= javascript_include_tag "http://example.com/main.js" %>
8  #http://example.com/main.js
9
10 <%= javascript_include_tag :defaults %>
11 #Prototype and Scriptaculous libraries
12
13 <%= javascript_include_tag :all %>
14 #every javascript file in public/javascripts
```

stylesheet_link_tag

```
1  <%= stylesheet_link_tag "main" %>
2  #public/stylesheets/main.css
3
4  <%= stylesheet_link_tag "main", "/photos/columns" %>
5  #public/stylesheets/main.css and public/photos/columns.css
6
7  <%= stylesheet_link_tag "http://example.com/main.css" %>
8  #http://example.com/main.css
9
10 <%= stylesheet_link_tag :all %>
11 #every css file in public/stylesheets
```

image_tag

```
1   <%= image_tag "header" %>
2   #public/images/header.png
3
4   <%= image_tag "icons/delete.gif" %>
5
6   <%= image_tag "icons/delete.gif", {:height => 45} %>
7
8   <%= image_tag "home.gif", :onmouseover => "menu/home_highlight.gif"  %>
9   <%= image_tag "home.gif",
10      :alt => "Go Home", :id => "HomeImage", :class => 'nav_bar' %>
```

yield and content_for I

- yield identifies a section where content from the view should be inserted

```
1  <html>
2    <head>
3    </head>
4    <body>
5          <%= yield %>
6    <hbody>
7  </html>
```

```
1  <html>
2    <head>
3          <%= yield :head %>
4    </head>
5    <body>
6          <%= yield %>
7    <hbody>
8  </html>
```

yield and content_for II

- the main body of the view will always render into the unnamed yield, to render content into a named yield, you use the content_for method

```
1  <% content_for :head do %>
2    <title>A simple page</title>
3  <% end %>
4
5  <p>Hello, Rails!</p>
```

```
1  <html>
2    <head>
3          <title>A simple page</title>
4    </head>
5    <body>
6          <p>Hello, Rails!</p>
7    <hbody>
8  </html>
```

Partials

- Partial templates - another device for breaking apart the rendering process into more manageable chunks
- render a partial as part of a view

```
1  <%= render :partial => "menu" %>
2  #render a file named _menu.html.erb
3
4  <%= render "shared/menu" %>
5  #render app/views/shared/_menu.html.erb
```

- using partials to simplify views

```
1  <%= render :partial => "shared/ad_banner" %>
2
3  <h1>Products</h1>
4
5  <p>Here are a few of our fine products:</p>
6  ...
7
8  <%= render :partial => "shared/footer" %>
```

Nested layouts I

Listing 3: app/views/layouts/application.erb

```
1  <html>
2  <head>
3    <title><%= @page_title %><title>
4    <% stylesheet_tag 'layout' %>
5    <style type="text/css"><%= yield :stylesheets %></style>
6  <head>
7  <body>
8    <div id="top_menu">Top menu items here</div>
9    <div id="menu">Menu items here</div>
10   <div id="content"><%= yield :content %></div>
11   <div id="main"><%= yield %></div>
12 </body>
13 </html>
```

Nested layouts II

Listing 4: app/views/layouts/news.erb

```
1  <% content_for :stylesheets do %>
2    #top_menu {display: none}
3    #right_menu {float: right; background-color: yellow; color: black}
4  <% end %>
5  <% content_for :content %>
6    <div id="right_menu">Right menu items here</div>
7  <% end %>
8  <% render :file => 'layouts/application' %>
```

References

- Ruby on Rails guides - http://guides.rubyonrails.org/
- Przewodniki po Ruby on Rails - http://apohllo.pl/guides/index.html