This course is based on RubyLearning's "Introduction to Sinatra" book. Do consider buying this eBook to support the time and energy put into the various courses by RubyLearning's mentors and to help RubyLearning bring to you relevant eBooks and courses related to the Ruby programming language.

Error Handling

A 404 page

When someone comes to a page on your domain that is no longer there (either because it's been deleted, because they've typed something in wrong or because the link that they followed was wrong) they are shown the dreaded 404 'page not found' error page.

This error simply means that the person was able to communicate with your server but that the server couldn't find the page that they were after.

404 errors should not be confused with "server not found" or similar errors, in which a connection to the destination server could not be made at all.

When a **Sinatra::NotFound** exception is raised, or the response's status code is 404, the**not_found** handler is invoked:

```
not_found do
  "I'm sorry but the Page You're Looking for Is Not Here"
end
```

Therefore the application **day5_2_p1.rb** from Day 5.2 can be re-written as:

```
require 'sinatra'

get '/' do
  erb :main
end


not_found do
  "I'm sorry but the Page you're Looking for Is Not Here"
end
```

We could also have a say **404.erb** file in the *views* folder which could render your 404 page. For example, the above **day5_2_p1.rb** can be re-written as:

```
require 'sinatra'

get '/' do
  erb :main
end


not_found do
  erb :'404'
end
```

Note that I had to surround the templates in single quotes. This is because Ruby syntax doesn't let symbol's first character be a number. By quoting it, it gets around that issue.

The **404.erb** file contents could be:

"I'm sorry but the Page you're Looking for Is Not Here"

A 500 page

A 500 error page will be thrown to the client when the Web server (running the Web Site) encounters an unexpected condition (usually something has gone wrong with your app) that prevents it from fulfilling the request by the client (e.g. your Web browser) for access to the requested URL.

By default **error** will catch **Sinatra::ServerError**. Sinatra will pass you the error via the 'sinatra.error' in **request.env**.

We can have an application file as follows:

```
# Render views/500.erb

error do

  erb :'500'

end


get '/' do

  raise "Error happened!"

end
```

The **500.erb** file contents could be:

```
request.env['sinatra_error']
```

Helpers

Helpers are methods available to routes, before filters, and views. They are useful for cutting down on any repetitive tasks that you might want to do. So they could be used in the erb files but are not necessarily exclusive to them.

Creating helpers in Sinatra is really easy, all you do is place them inside the helpers block. This basically means that with the **helpers** method I can define say methods that would be available in my blocks in **get** or my views etc.

Let us write a program **day6_1_p1.rb** as follows:

```
require 'sinatra'

helpers do

  def human_date(datetime)

      datetime.strftime('%d|%m|%Y').gsub(/ 0(\d{1})/, ' \1')

  end

end


get '/' do

  erb :time1

end
```

In the **views** folder, I have the file **time1.erb**:

```
<h1>Looking at Helpers</h1>
<p>The current date is: <%= human_date(Time.now) %></p>
```

Deploy this app to Heroku (remember to change **config.ru**) and check out the results. **Please post the URL of your app as comments to this page.**

Some of the other built-in helper methods available in Sinatra are: **error, not_found, redirect, request, response, session, status, throw**.

Let's wrap up this discussion with a simple program using the **request** helper method. Type in the following program **day6_1_p2.rb** deploy it (remember to change **config.ru**), run it and check your browser window. You should see that the class name is:

```
Sinatra::Request
```

This is basically a Rack request.

Here's the program:

```
require 'sinatra'

get '/' do

  request.class.inspect

end
```

Please post the URL of your app as comments to this page.

Links to the courseware

- Back to the **Beginning of the course**
- Back to **Day 5.2**
- Forward to **Day 6.2**

Please click on the **Like** button of this page. **Comments** appreciated.