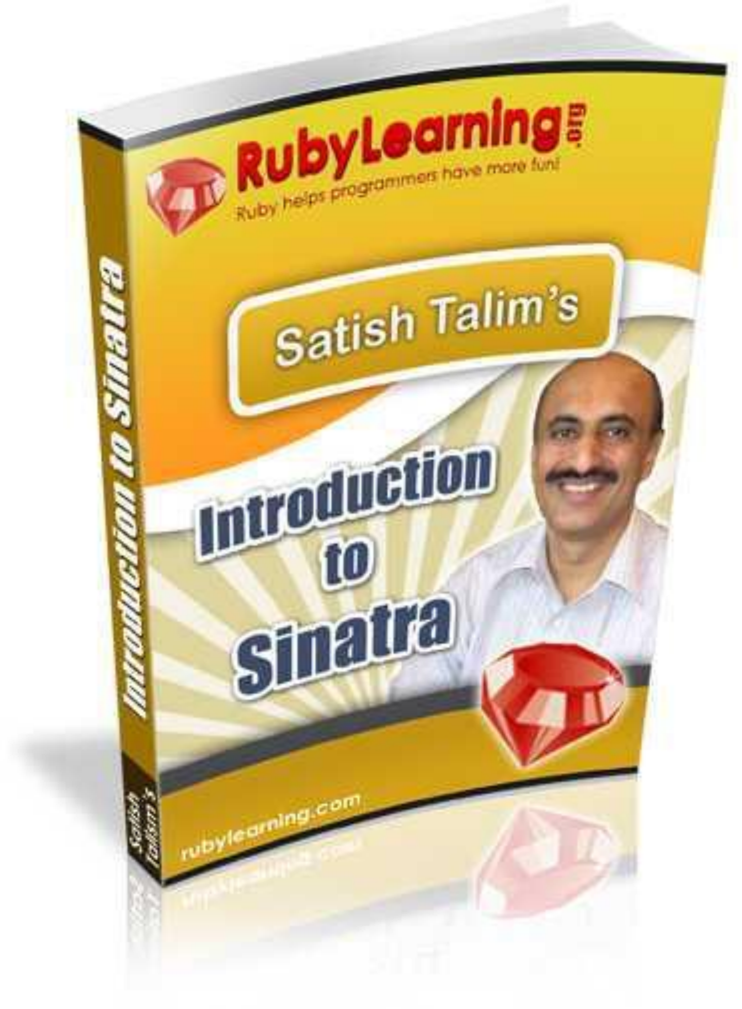

*This course is based on RubyLearning's "**[Introduction to Sinatra](#)**" book. Do consider buying this eBook to support the time and energy put into the various courses by RubyLearning's mentors and to help RubyLearning bring to you relevant eBooks and courses related to the Ruby programming language.*

With Sinatra installed, let us quickly write and run a "Hello Rubyists" Sinatra app.

For now, open your favorite text editor and type the following:

```
# day2_1_e1.rb

require 'sinatra'

get '/' do

  ‘Hello Rubyists’

end
```

Note: If you are using a version of Ruby which is less than 1.9, then please add the following line as the first line to every Sinatra program you write:

```
require 'rubygems'
```

Save the file into your Sinatra programs folder say `sinatrafb` as `day2_1_e1.rb`.

On Windows, open a command window and run this application by typing:

```
ruby day2_1_e1.rb
```

Next load `http://localhost:4567` in your browser. You should see:

Hello Rubyists

in the browser window.

This happens because when you use the above URL, Sinatra runs the `get` block and will return whatever you put inside it, to the web browser.

Now, let us familiarize ourselves with Sinatra's features.

Routes

The main feature of Sinatra is defining 'routes' as an HTTP verb for a path that executes an arbitrary block of ruby code. Something like:

```
verb 'path' do
 ... # return/render something
end
```

Sinatra's routes are designed to respond to the HTTP request methods (**GET**, **POST**, **PUT**,**DELETE**).

In Sinatra, a route is an HTTP method paired with an URL matching pattern.

These URL handlers (also called "routing") can be used to match anything from a static string (such as `/hello`) to a string with parameters (`/hello/:name`) or anything you can imagine using wildcards and regular expressions.

Each route is associated with a block:

Let us look at an example:

```
get '/' do
  .. show something ..
end


get '/hello/:name' do
  # The /hello portion matches that portion of the URL from the
  # request you made, and :name will absorb any other text you
  # give it and put it in the params hash under the key :name
end


post '/' do
  .. create something ..
end


put '/' do
  .. update something ..
```

```
end


delete '/' do

  .. delete something ..

end
```

Routes are matched in the order they are defined. The first route that matches the request is invoked.

When a new request comes in, the first route that matches the request is invoked i.e. the handler (the code block) attached to that route gets executed. For this reason, you should put your most specific handlers on top, and your most vague handlers on the bottom.

To access POSTed parameters, use `params[:xxx]` where xxx is the name of the form element that was posted (more on this later).

Links to the courseware

- Back to the **Beginning of the course**
- Back to **Day 1**
- Forward to **Day 2.2**

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