Sinatra

Micro Web Framework

About

- “Sinatra is a DSL for quickly creating web applications in Ruby with minimal effort”

- Simply put Sinatra is a micro web framework, that allows rapid development

Unique Features

- Does not enforce MVC
- Does not have a default ORM
- Ability to encapsulate entire web apps in a single file

Uses

- Small to medium size web apps

- RESTful web services

Routes

- A route is an HTTP method and URL-matching pattern that are associated with a block

- Routes are matched in order; the first route that matches is invoked

Routes Example

```
get ‘/’ do
      “Hello!”
end

get ‘/Goodbye’ do
      “Goodbye!”
end
                                    #Routes can include named parameters
get ‘/:name’ do                     #matches get /Frank and sets params[:name]
      “Hello #{params[:name]}!”     #Hello Frank!
end
```

Wildcard Parameters

- Route patterns can include wildcard parameters

```
get '/Goodbye/*/*' do   #matches get /Goodbye/Frank/Sinatra
      params[:splat]    # => ["Frank", "Sinatra"]
end

get '/Goodbye/*.*' do   #matches get /Goodbye/path/to/file.xml
      params[:splat]    # => ["path/to/file", "xml"]
end
```

Query Parameters

- Route patterns can utilize query parameters

```
get ‘/Goodbye’ do                          #matches get /Goodbye?name=Frank
      “Goodbye #{params[:name]}!”  # Goodbye Frank!
end
```

Static Files

- Static files are served from the ./public directory

```
#Will check if a static file exists  before route matching
enable :static

#Set public folder to where static content lives
set :public_folder, ‘/static’
```

Other Features

- Conditions: :agent, :host_name, :provides
  - Routes may include a variety of matching conditions and can by user defined
- Custom Route Matchers
  - Sinatra provides string and regular expression
- Support for many views/template languages:
  - haml, erb, builder, etc
- Filters, Helpers, Error Handling, and Testing

Example

- Contacts RESTful service
  - Get all contacts
  - Add new contact
  - Update existing contact
  - Delete existing contact
- Uses MongoDB

Example get

```
get '/contacts' do
    content_type :json
    $coll.find.to_a.to_json   #$coll references DB collection
end
```

Example post

```
post '/contact' do
    content_type :json
    data = JSON.parse request.body.read
    id = $coll.insert data
    getContact(id)          #Helper that returns a contact
end                         #as json based on id
```

Example put

```
put '/contact/:id' do
    content_type :json
    data = JSON.parse request.body.read
    id = mongoId(params[:id])  #Helper to resolve Mongo ID
    $coll.update({:_id => id}, data)
    getContact(id)
end
```

Example delete

```
delete '/contact/:id' do
    content_type :json
    id = mongoId(params[:id])
    if $coll.find_one(id)
        $coll.remove(:_id => id)
        {:success => true}.to_json
    else
        {:success => false}.to_json
    end
end
```

Sources

- Sinatra
  - http://www.sinatrarb.com/intro.html
- Sinatra Recipes MongoDB
  - http://recipes.sinatrarb.com/p/databases/mongo
- About Tech
  - http://ruby.about.com/od/sinatra/a/sinatra1.htm
- Stack Overflow
  - http://stackoverflow.com/questions/812856/what-are-the-main-differences-between-sinatra-and-ramaze