Routes

In Sinatra, a route is an HTTP method paired with a URL-matching pattern. Each route is associated with a block:

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

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

put '/' do
  .. replace something ..
end

patch '/' do
  .. modify something ..
end

delete '/' do
  .. annihilate something ..
end

options '/' do
  .. appease something ..
end

link '/' do
  .. affiliate something ..
end

unlink '/' do
  .. separate something ..
end
```

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

Route patterns may include named parameters, accessible via the params hash:

```
get '/hello/:name' do
  # matches "GET /hello/foo" and "GET /hello/bar"
  # params['name'] is 'foo' or 'bar'
  "Hello #{params['name']}!"
end
```

You can also access named parameters via block parameters:

```
get '/hello/:name' do |n|
  # matches "GET /hello/foo" and "GET /hello/bar"
  # params['name'] is 'foo' or 'bar'
  # n stores params['name']
  "Hello #{n}!"
end
```

Route patterns may also include splat (or wildcard) parameters, accessible via the params['splat'] array:

```
get '/say/*/to/*' do
  # matches /say/hello/to/world
  params['splat'] # => ["hello", "world"]
end

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

Or with block parameters:

```
get '/download/*.*' do |path, ext|
  [path, ext] # => ["path/to/file", "xml"]
end
```

Route matching with Regular Expressions:

```
get /\A\/hello\/([\w]+)\z/ do
  "Hello, #{params['captures'].first}!"
end
```

Or with a block parameter:

```
get %r{/hello/([\w]+)} do |c|
  # Matches "GET /meta/hello/world", "GET /hello/world/1234" etc.
  "Hello, #{c}!"
end
```

Route patterns may have optional parameters:

```
get '/posts.?:format?' do
  # matches "GET /posts" and any extension "GET /posts.json", "GET /posts.xml" etc.
```

```
end
```

Routes may also utilize query parameters:

```
get '/posts' do
  # matches "GET /posts?title=foo&author=bar"
  title = params['title']
  author = params['author']
  # uses title and author variables; query is optional to the /posts route
end
```

By the way, unless you disable the path traversal attack protection (see below), the request path might be modified before matching against your routes.

Passing

A route can punt processing to the next matching route using pass:

```
get '/guess/:who' do
  pass unless params['who'] == 'Frank'
  'You got me!'
end

get '/guess/*' do
  'You missed!'
end
```

The route block is immediately exited and control continues with the next matching route. If no matching route is found, a 404 is returned.

Triggering Another Route

Sometimes pass is not what you want, instead you would like to get the result of calling another route. Simply use call to achieve this:

```
get '/foo' do
  status, headers, body = call env.merge("PATH_INFO" => '/bar')
  [status, headers, body.map(&:upcase)]
end

get '/bar' do
  "bar"
end
```

Note that in the example above, you would ease testing and increase performance by simply moving "bar" into a helper used by both /foo and /bar.

If you want the request to be sent to the same application instance rather than a duplicate, use call! instead of call.

Check out the Rack specification if you want to learn more about call.