spark-notes

September 3, 2018

Contents

Part I

Project Setup

Chapter 1

Dependencies

1.1 Maven

```
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.7.2</version>
</dependency>
```

Part II

Server Operations

Chapter 2

init()

Default behaviour if it fails:

```
private Consumer<Exception> initExceptionHandler = (e) -> {
    LOG.error("ignite failed", e);
    System.exit(100);
};
```

The behaviour can be customised by invoking

```
initExceptionHandler(e -> ());
```

Chapter 3

stop()

Shuts down the server.

Part III

Routes

Chapter 4

Overview

- they are made of three peces:
  - **a verb** - *get, post put, delete, head, trace, connect, options*
  - **a path** *(URL)*
  - **a callback** - *(request, response) ->()*
- they are matched in the order they are defined
- **static import is recommended** for improved readability

get() - shows something.

post() - creates something

put() - updates something

delete() - self-explanatory

options() - 'appease' something

Chapter 5

Named Paramaters

- preceded by a colon
- accessed by ***params*** method on ***request*** object.

```
get("/path/:param", (req, res) -> {
    return "Hello " + req.params(":param");
});
```

5.1 splat()

It works like a combination of wildcard and vararg. ***splat()*** return an arrays of those parameters.

```
get("/path/*", (req, res) -> {
    return "Number of splat parameters is " + req.splat().length;
})
```

Chapter 6

Grouping Routes

We need to call a ***path(..)*** method:

```
path("/api", () -> {
    before("/*", (q, a) -> log.info("Received api call"));
    path("/email", () -> {
        post("/add",       EmailApi.addEmail);
        put("/change",     EmailApi.changeEmail);
        delete("/remove",  EmailApi.deleteEmail);
    });

    path("/username", () -> {
        post("/add",       UserApi.addUsername);
        put("/change",     UserApi.changeUsername);
        delete("/remove",  UserApi.deleteUsername);
    });
});
```

Part IV

Request

Chapter 7

List of Methods

request.attributes();	the attributes list
request.attribute("foo");	value of foo attribute
request.attribute("A", "V");	sets value of attribute A to V
request.body();	request body sent by the client
request.bodyAsBytes();	request body as bytes
request.contentLength();	length of request body
request.contentType();	content type of request.body
request.contextPath();	the context path, e.g. "/hello"
request.cookies();	request cookies sent by the client
request.headers();	the HTTP header list
request.headers("BAR"	value of BAR header
request.host	the host, e.g. "example.com
request.	client IP address
request.params("foo"	value of foo path parameter
request.params();	map with all parameters
request.pathInfo();	the path info
request.port();	the server port
request.protocol();	the protocol, e.g. HTTP/1.1
request.queryMap();	the query map
request.queryMap("foo");	query map for a certain parameter
request.queryParams();	the query param list
request.queryParams("FOO");	value of FOO query param
request.queryParamsValues("FOO")	all values of FOO query param
request.raw();	raw request handed in by Jetty
request.requestMethod();	The HTTP method (GET, ..etc)
request.scheme();	"http"
request.servletPath();	the servlet path, e.g. /result.jsp
request.session();	session management
request.splat();	splat (*) parameters
request.uri();	the uri, e.g. "http://example.com/foo"
request.url();	the url. e.g. "http://example.com/foo"
request.userAgent();	user agent

Part V

Respond

Chapter 8

List of Methods

response.body();	get response content
response.body("Hello");	sets content to Hello
response.header("FOO", "bar");	sets header FOO with value bar
response.raw();	raw response handed in by Jetty response.redirect("/example");
browser redirect to /example	
response.status();	get the response status
response.status(401);	set status code to 401
response.type();	get the content type
response.type("text/xml");	set content type to text/xml

Chapter 9

Querry Maps, Cookies, Sessions, Halting

http://sparkjava.com/documentation#sessions

9.1 Query Maps

```
request.queryMap().get("user", "name").value();
request.queryMap().get("user").get("name").value();
request.queryMap("user").get("age").integerValue();
request.queryMap("user").toMap();
```

9.2 Cookies

```
request.cookies();              // get map of all request cookies
request.cookie("foo"            // access request cookie by name
response.cookie("foo", "bar");        // set cookie with a value
response.cookie("foo", "bar", 3600);  // set cookie with a max-age
response.cookie("foo", "bar", 3600, true); // secure cookie
response.removeCookie("foo");              // remove cookie
```

9.3 Sessions

```
// create and return session
request.session(true);

 // Get session attribute 'user'
request.session().attribute("user");

// Set session attribute 'user'
```

```
request.session().attribute("user","foo");

// Remove session attribute 'user'
request.session().removeAttribute("user");
request.session().attributes();      // Get all session attributes
request.session().id();              // Get session id
request.session().isNew();           // Check if session is new
request.session().raw();             // Return servlet object
```

9.4 Halting

Stops immediately a request within a filter or route.

```
halt();                // halt
halt(401);             // halt with status
halt("Body Message");  // halt with message
halt(401, "Go away!"); // halt with status and message
```

Chapter 10

Filters

10.1 Before

They are evaluated **before each request**:

```
before((request, response) -> {
    boolean authenticated;

    // ... check if authenticated

    if (!authenticated) {
        halt(401, "You are not welcome here");
    }
});
```

10.2 After

They are evaluated **after each request**:

```
after((request, response) -> {
    response.header("foo", "set by after filter");
});
```

10.3 afterAfter

Works similar to 'finally' blocks:

```
afterAfter((request, response) -> {
    response.header("foo", "set by afterAfter filter");
});
```

10.4 Filters Taking Patterns

They are evaluated **only if request path matches the pattern**:

```
before("/protected/*", (request, response) -> {
    // ... check if authenticated

    halt(401, "Go Away!");
});
```

Part VI

TBC

http://sparkjava.com/documentation#response-transformer