http://sparkjava.com/

Java Spark - A micro framework for creating web applications in Kotlin and Java 8 with minimal effort

Inspired by Ruby Sinatra(created 2007) Version1 was created in 2011(no longer maintained Supports java7 )

Version2 rewritten in 2014 supports Java8

Spark uses Java 8's lambda expressions extensively, which makes Spark applications a lot less verbose. In contrast to other Java web frameworks, Spark does not use heavily XML files or annotations.

The JVM offers one of the biggest programming ecosystems in the world. It has a lot of Java web frameworks, but pure Java web development has traditionally been very cumbersome. If you love the JVM, but hate verbose code and frameworks, Spark is the web framework for you. It will have you up and running in minutes, and you can even use it with Groovy or Kotlin or whatever you want. Spark is an expressive, lightweight and unopinionated pure Java (and Kotlin) web framework .

Microservices work best with micro frameworks, and Spark has your REST API ready to serve JSON in less than ten lines of code. Spark is mainly used for creating REST API’s, but it also supports a multitude of template engines.

Supports all HTTP versbs

Request access and Resonse configurations

Cookies

Sessions

Aop like Filters

Redirects

Exception Mappings

Views and Templates

Spark has community-provided wrappers for a lot of popular template engines:

Velocity (very mature, feature rich, great IDE support)

Freemarker (very mature, feature rich, great IDE support)

Mustache (mature, decent IDE support)

Handlebars (mature, decent IDE support)

Jade (mature, decent IDE support)

Thymeleaf (mature, feature rich, decent IDE support)

Pebble (we know very little about this)

Water (we know very little about this)

jTwig (we know very little about this)

Jinjava (we know very little about this)

Jetbrick (we know very little about this)

Embedded web server

Standalone Spark runs on an embedded Jetty web server Runs on default port 4567

To run Spark on another web server (instead of the embedded jetty server), an implementation of the interface spark.servlet.SparkApplication is needed. You have to initialize your routes in the init() method, and the following filter might have to be configured in your web.xml:

maven project and add the dependency

<dependency>

<groupId>com.sparkjava</groupId>

<artifactId>spark-core</artifactId>

<version>2.7.2</version>

</dependency>

Rest Service

import static spark.Spark.\*;

public class HelloWorld {

public static void main(String[] args) {

get("/hello", (req, res) -> "Hello World");

}

}

<http://localhost:4567/hello>

The server is automatically started when you do something that requires the server to be started (i.e. declaring a route or setting the port).  
You can also manually start the server by calling init().

The main building block of a Spark application is a set of routes. A route is made up of three simple pieces:

* A **verb** (get, post, put, delete, head, trace, connect, options)
* A **path** (/hello, /users/:name)
* A **callback** (request, response) -> { }

oute patterns can include named parameters, accessible via the params()

Route patterns can also include splat (or wildcard) parameters. These parameters can be accessed by using the splat()

get("/hello/:name", (request, response) -> {

return "Hello: " + request.params(":name");

});

get("/say/\*/to/\*", (request, response) -> {

return "Number of splat parameters: " + request.splat().length;

});

Before-filters are evaluated **before each request**, and can read the request and read/modify the response.  
To stop execution, use halt():

before((request, response) -> {

boolean authenticated;

// ... check if authenticated

if (!authenticated) {

halt(401, "You are not welcome here");

}

});

Copy

After-filters are evaluated **after each request**, and can read the request and read/modify the response:

after((request, response) -> {

response.header("foo", "set by after filter");

});

Copy

ResponseTransformer

Mapped routes that transform the output from the handle method. This is done by extending the ResponseTransformer object and passing it to the mapping method

get("/hello", "application/json", (request, response) -> {

return new MyMessage("Hello World");

}, new JsonTransformer());

<https://sparktutorials.github.io/2016/06/10/spark-basic-structure.html>

<http://zetcode.com/java/spark/>

<http://www.baeldung.com/spark-framework-rest-api>