

---

# Spring Boot - Creating web application using Spring MVC

---

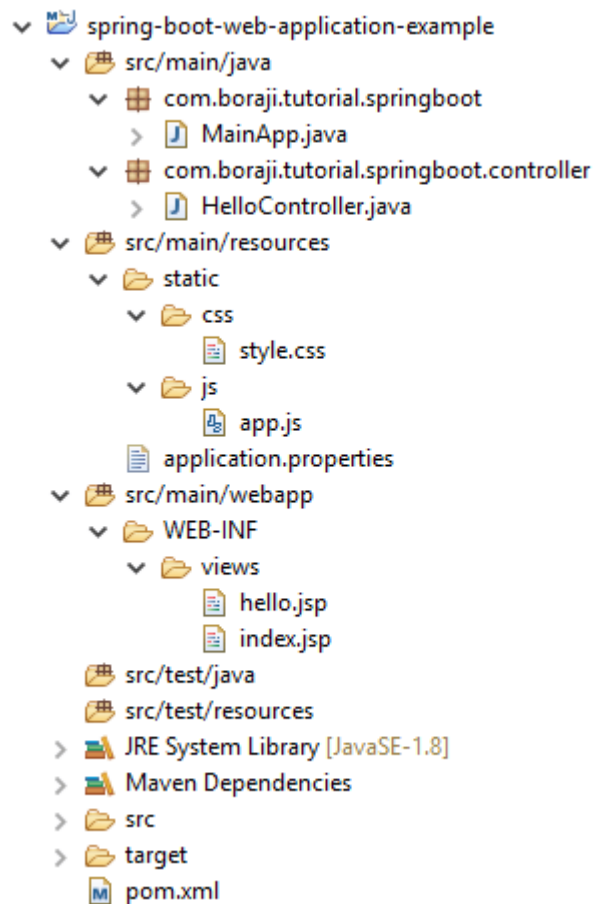
In this post, I will show you how to create and bootstrap a Spring MVC web application using the Spring Boot framework. We will use the JSP as view for our web application.

Tools and technologies used for this application are-

- Spring Boot 1.5.4.RELEASE
- Spring WebMVC 4.3.9.RELEASE
- Tomcat Embedded 8.5
- JavaSE 1.8
- Maven 3.3.9
- Eclipse Neon.3

## Project Structure

Review the spring boot project structure.



Related - How to create a maven project in eclipse IDE (/how-to-create-a-maven-project-in-eclipse).

## Jar dependencies

To create and run a Spring MVC web application in spring boot, you need to add the `spring-boot-starter` dependency in your `pom.xml` file.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Here is the complete `pom.xml` file.

**pom.xml**

```

<project xmlns="http://maven.apache.org/POM/4.0.0 (http://maven.apache.org/POM/4.0.0)" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/2001/XMLSchema-instance)" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 (http://maven.apache.org/POM/4.0.0) http://maven.apache.org/xsd/maven-4.0.0.xsd (http://maven.apache.org/xsd/maven-4.0.0.xsd)">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.tutorial.springboot</groupId>
  <artifactId>spring-boot-web-application-example</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.4.RELEASE</version>
  </parent>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- JSTL tag lib -->
    <dependency>
      <groupId>javax.servlet.jsp.jstl</groupId>
      <artifactId>javax.servlet.jsp.jstl-api</artifactId>
      <version>1.2.1</version>
    </dependency>

    <dependency>
      <groupId>taglibs</groupId>
      <artifactId>standard</artifactId>
      <version>1.1.2</version>
    </dependency>

    <!-- Tomcat for JSP rendering -->
    <dependency>
      <groupId>org.apache.tomcat.embed</groupId>
      <artifactId>tomcat-embed-jasper</artifactId>
      <scope>provided</scope>
    </dependency>
  </dependencies>

```

```
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
<packaging>war</packaging>
</project>
```

## Controller class

Create a `HelloController` class under `com.tutorial.springboot.controller` package and write the following code in it.

**HelloController.java**

```
package com.tutorial.springboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class HelloController {

    @RequestMapping("/")
    public String index() {
        return "index";
    }

    @PostMapping("/hello")
    public String sayHello(@RequestParam("name") String name, Model model) {
        model.addAttribute("name", name);
        return "hello";
    }
}
```

## JSP Views

Create `index.jsp` and `hello.jsp` files under `src/main/webapp/WEB-INF/views` folder as shown in the project structure.

**index.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<!-- Static content -->
<link rel="stylesheet" href="/resources/css/style.css">
<script type="text/javascript" src="/resources/js/app.js"></script>

<title>Spring Boot</title>
</head>
<body>
  <h1>Spring Boot - MVC web application example</h1>
  <hr>

  <div class="form">
    <form action="hello" method="post" onsubmit="return validate()">
      <table>
        <tr>
          <td>Enter Your name</td>
          <td><input id="name" name="name"></td>
          <td><input type="submit" value="Submit"></td>
        </tr>
      </table>
    </form>
  </div>

</body>
</html>
```

**hello.jsp**

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" (http://java.sun.com/jsp/jstl/core)" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Spring Boot</title>
</head>
<body>
    <h1>Spring Boot - MVC web application example</h1>
    <hr>

    <h2>Your name is ${name}</h2>

</body>
</html>

```

## Static web resources

By default Spring Boot serves the static web content from the `/static` or `/public` or `/resources` or `/META-INF/resources` folder in a classpath. In this example, we will put all static resources such as stylesheet, JavaScript files under `/static` folder.

Create a stylesheet file under `src/main/resources/static/css` folder and write the following code in it.

### style.css

```

.form {
    background-color: #efefef;
    width: 400px;
    height: 50px;
    border-radius: 7px;
    padding: 20px;
}

```

Now, create a javascript file under `src/main/resources/static/js` folder and write the following code in it.

**app.js**

```
function validate() {
    var name = document.getElementById("name").value;
    if (name == '') {
        alert('Please enter a valid name.');
```

```
        return false;
    } else {
        return true;
    }
}
```

## Application properties

Spring boot load the properties from `application.properties` and add them to the Spring Environment . You can set the properties related to the Spring MVC or static web content in `application.properties` file.

Create an `application.properties` file under `src/main/resources` source folder and write the following properties in it.

**application.properties**

```
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

**Note -** `spring.mvc.static-path-pattern=/resources/**` will map the `classpath:/static/css/style.css` to `/resources/css/style.css` . Similarly, the `classpath:/static/js/app.js` will be mapped to `/resources/js/app.js` . You can use these static resources in jsp as follows.

```
<link rel="stylesheet" href="/resources/css/style.css">
<script type="text/javascript" src="/resources/js/app.js"></script>
```

## Main class

To bootstrap Spring MVC application using Spring Boot, create a main class annotated with `@SpringBootApplication` annotation as follows.

**MainApp.java**



```
package com.tutorial.springboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

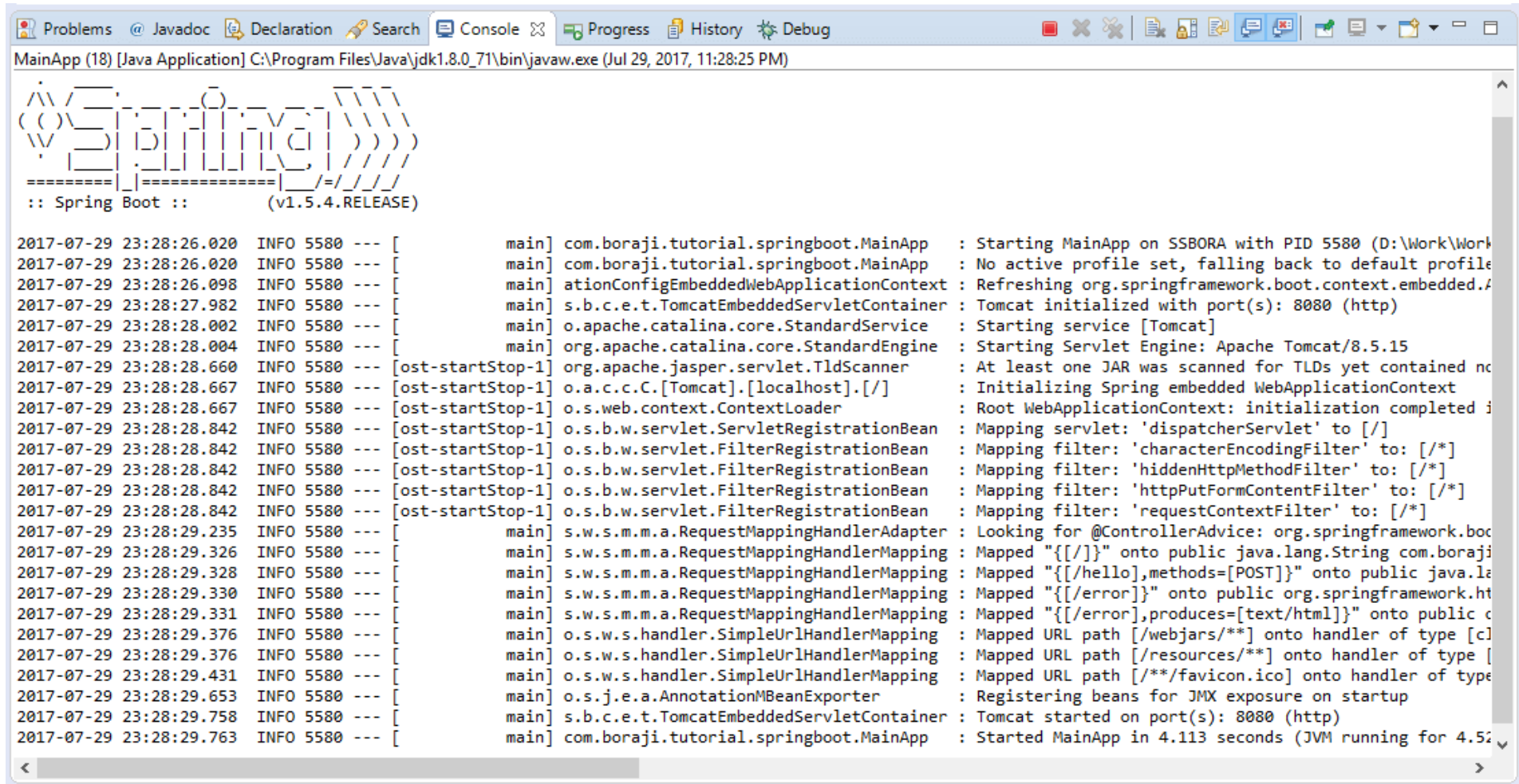
@SpringBootApplication
public class MainApp {
    public static void main(String[] args) {
        SpringApplication.run(MainApp.class, args);
    }
}
```

## Run application

Run the `MainApp.java` class as Java application i.e. go to Run → Run as → Java Application

You can use the `mvn spring-boot:run` command to run the spring boot application.

After executing the main class, the console output will look like as follows.



```

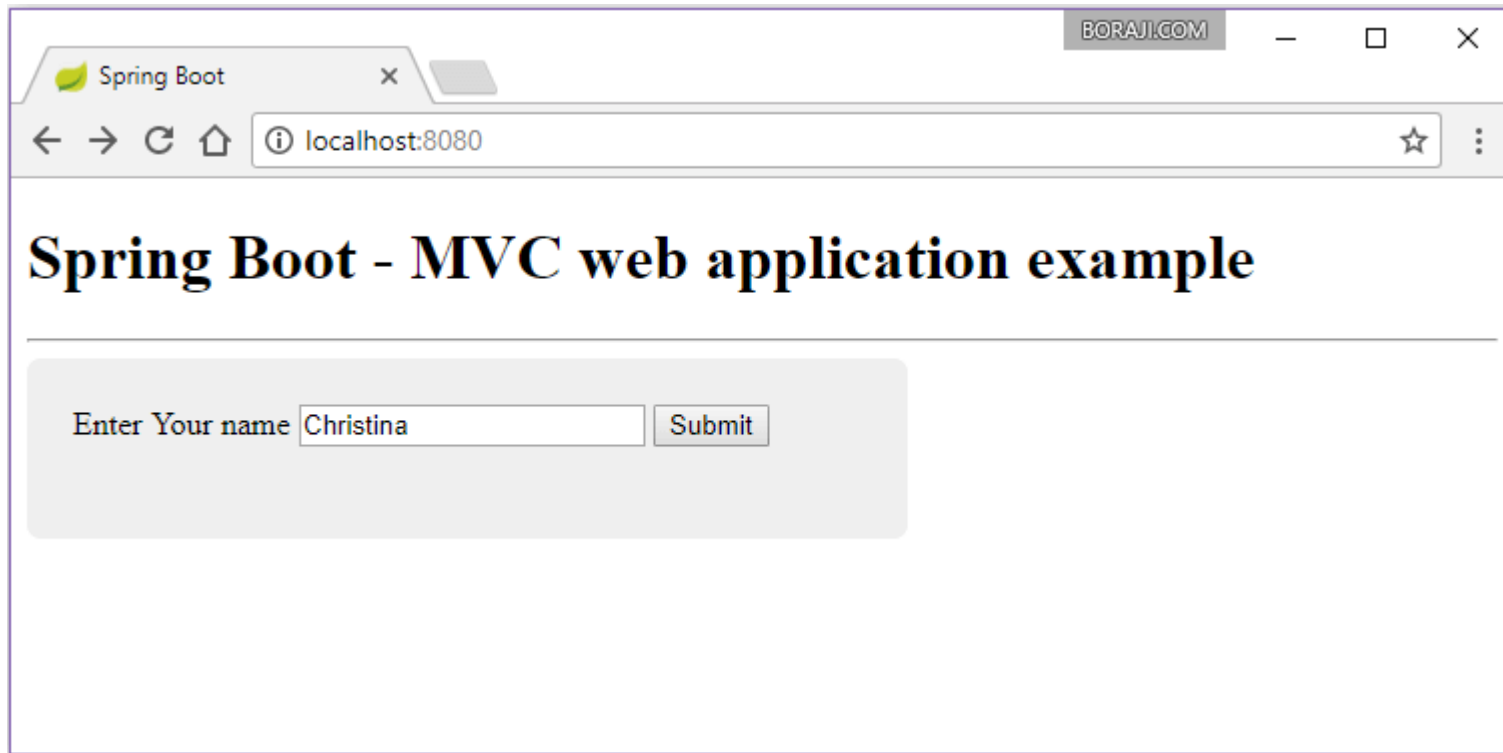
MainApp (18) [Java Application] C:\Program Files\Java\jdk1.8.0_71\bin\javaw.exe (Jul 29, 2017, 11:28:25 PM)

:: Spring Boot :: (v1.5.4.RELEASE)

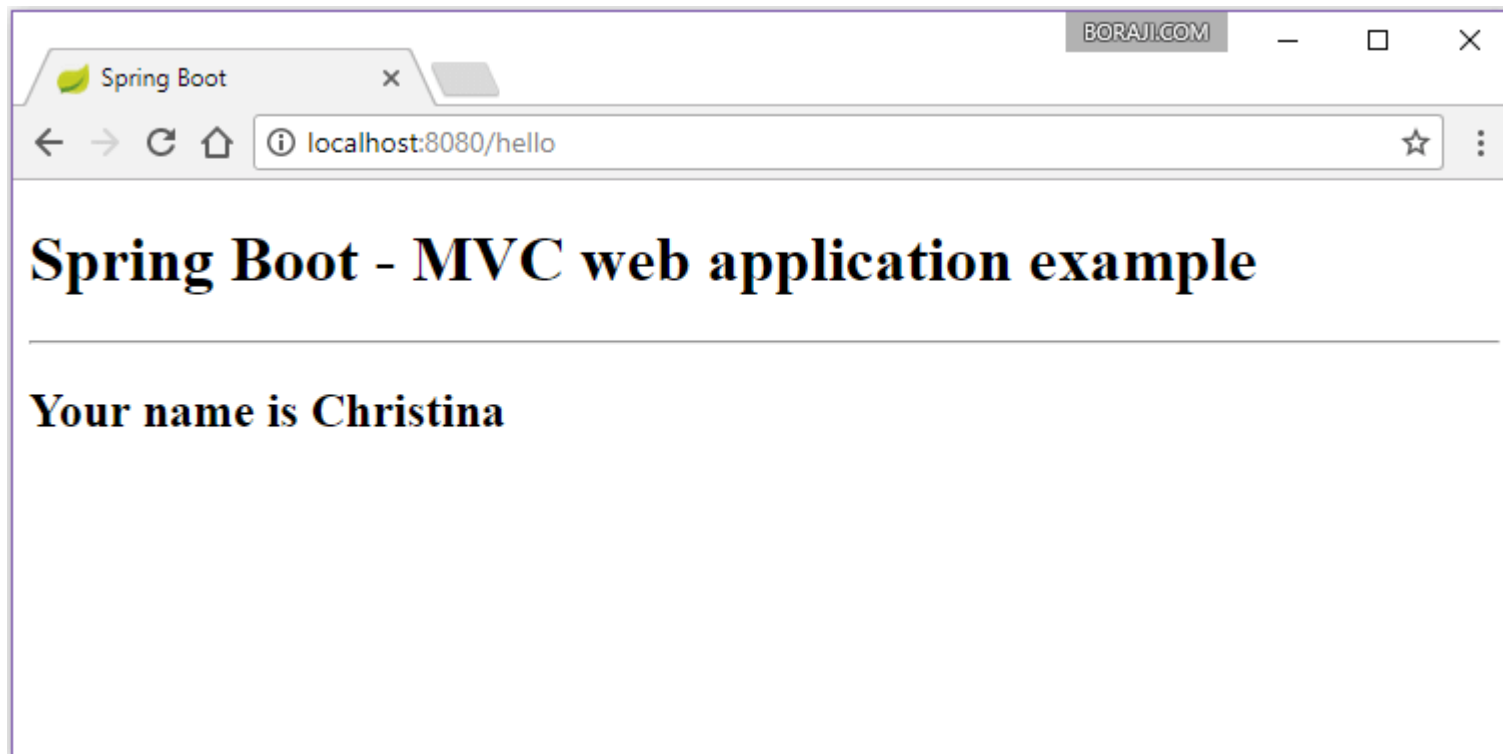
2017-07-29 23:28:26.020 INFO 5580 --- [main] com.boraji.tutorial.springboot.MainApp : Starting MainApp on SSBORA with PID 5580 (D:\Work\Work
2017-07-29 23:28:26.020 INFO 5580 --- [main] com.boraji.tutorial.springboot.MainApp : No active profile set, falling back to default profile
2017-07-29 23:28:26.098 INFO 5580 --- [main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.
2017-07-29 23:28:27.982 INFO 5580 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2017-07-29 23:28:28.002 INFO 5580 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2017-07-29 23:28:28.004 INFO 5580 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.15
2017-07-29 23:28:28.660 INFO 5580 --- [ost-startStop-1] org.apache.jasper.servlet.TldScanner : At least one JAR was scanned for TLDs yet contained no
2017-07-29 23:28:28.667 INFO 5580 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2017-07-29 23:28:28.667 INFO 5580 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed i
2017-07-29 23:28:28.842 INFO 5580 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2017-07-29 23:28:28.842 INFO 5580 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/]
2017-07-29 23:28:28.842 INFO 5580 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/]
2017-07-29 23:28:28.842 INFO 5580 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/]
2017-07-29 23:28:28.842 INFO 5580 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/]
2017-07-29 23:28:29.235 INFO 5580 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boc
2017-07-29 23:28:29.326 INFO 5580 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/]}" onto public java.lang.String com.boraji
2017-07-29 23:28:29.328 INFO 5580 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hello],methods=[POST]}" onto public java.la
2017-07-29 23:28:29.330 INFO 5580 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.ht
2017-07-29 23:28:29.331 INFO 5580 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public c
2017-07-29 23:28:29.376 INFO 5580 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [c]
2017-07-29 23:28:29.376 INFO 5580 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/resources/**] onto handler of type [
2017-07-29 23:28:29.431 INFO 5580 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type
2017-07-29 23:28:29.653 INFO 5580 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2017-07-29 23:28:29.758 INFO 5580 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2017-07-29 23:28:29.763 INFO 5580 --- [main] com.boraji.tutorial.springboot.MainApp : Started MainApp in 4.113 seconds (JVM running for 4.5

```

Now, enter the `http://localhost:8080` in browser's address and enter the your name in input field.



Click on submit button and see the result page.



## Packaging Spring Boot application as an executable war

You need to package the Spring boot application as war instead of jar, if you are using an embedded servlet container. There are some limitations in the JSP support. Read more.. (<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-jsp-limitations>)

You can use the following maven command to package your spring boot application as war.

```
mvn clean package
```

## Running executable war from command line

Use the following command to run the war from the CMD.

```
>java -jar spring-boot-web-application-example-0.0.1-SNAPSHOT.war
```

### Download Sources: