



Spring Boot Web App

Agenda

1

Spring Boot Web App.

2

Spring Boot Web App Using Properties File

3

Spring Boot Web App Accepting Client Data

4

Spring Boot MVC

5

Spring Boot MVC Using Model Object

Objectives

At the end of this module, you will be able to:

- Understand, How to create a web app in spring boot.
- Understand, How to create a web app using properties file.
- Understand, How the spring boot accepts client data.
- Understand, How to create MVC in spring boot.
- Understand, How to create MVC with model object.

Spring Boot Web App



Spring Boot Web App.

- If we want to develop a web application, we need to add the following dependency in pom.xml file:

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

- There are two important features of spring-boot-starter-web:
 - It is compatible for web development
 - Auto configuration
- Starter of Spring web uses Spring MVC, REST and Tomcat as a default embedded server.

Spring Boot Web App.

- While creating a spring boot web project, you can add web dependency as like below.

Project
☒ Maven Project ☐ Gradle Project

Language
☒ Java ☐ Kotlin ☐ Groovy

Spring Boot
☐ 2.4.0 (SNAPSHOT) ☐ 2.4.0 (M1) ☐ 2.3.3 (SNAPSHOT) ☒ 2.3.2
☐ 2.2.10 (SNAPSHOT) ☐ 2.2.9 ☐ 2.1.17 (SNAPSHOT) ☐ 2.1.16

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☐ Jar ☒ War

Java ☐ 14 ☐ 11 ☒ 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot Web App.

- Since we are going to use JSP in our example, we need to add the below dependency in our pom.xml as per our embedded tomcat version.
- Because mainly Spring Boot is concentrating on REST and Micro Services.

```
<dependency>  
    <groupId>org.apache.tomcat</groupId>  
    <artifactId>tomcat-jasper</artifactId>  
    <version>9.0.33</version>  
</dependency>
```

- Here, we are going to use two annotations @Controller and @RequestMapping.

How to find out embedded tomcat version

While running the previous example, We can find the embedded tomcat version from the console like below.

```
... Starting DemoApplication on ... with PID ...
: No active profile set, falling back to default profile
Server : Tomcat initialized with port(s): 9022 (http)
Service : Starting service [Tomcat]
Engine : Starting Servlet engine: [Apache Tomcat/9.0.37]
... : At least one JAR was scanned for TLDs yet contained no
... : Initializing Spring embedded WebApplicationContext
Context : Root WebApplicationContext: initialization completed i
```

We can get the dependency from the below link

<https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper>

Spring Boot Web App - Example



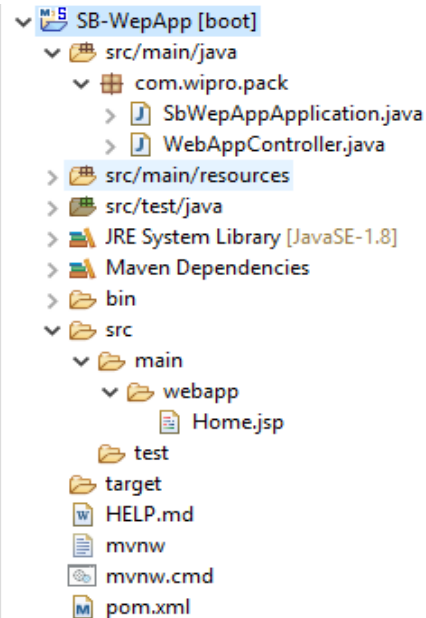
Spring Boot Web App.

```
package com.wipro.pack;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class WebAppController {

    @RequestMapping("/home")
    public String home() {
        return "Home.jsp";
    }

}
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<font color=green><b>Welcome to Home Page</b></font>
</body>
</html>
```



← → ↻ ⓘ localhost:8123/home

Welcome to Home Page

SB Web App Using Properties File



Spring Boot Web App Using Properties File.

- In our previous example, We have created Home.jsp file inside “webapp” folder. Because by default the spring boot will check this folder to find the view pages.
- Also in our “WebAppController”, we have hard coded the extensions of view page as “.jsp”.
- In case, If I want to store all my view pages in a separate folder and I want change the view page extension based on customer requirement with out touching my Controller then the properties file will come to the picture.

SB Web App Using Properties File - Example



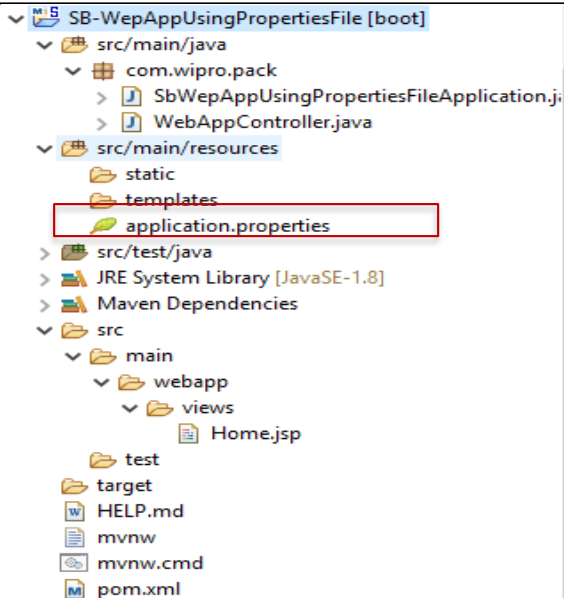
Spring Boot Web App Using Properties File.

```
package com.wipro.pack;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class WebAppController {

    @RequestMapping("/home")
    public String home() {
        return "Home";
    }
}
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<font color=green><b>Welcome to Home Page</b></font>
</body>
</html>
```

```
server.port=8123
spring.mvc.view.prefix=/views/
spring.mvc.view.suffix=.jsp
```



← → ↻ ⓘ localhost:8123/home

Welcome to Home Page

SB Web App Accepting Client Data



SB Web App Accepting Client Data.

```
package com.wipro.pack;
import javax.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class WebAppController {
    @RequestMapping("/home")
    public String home() {
        return "Home";
    }
    @RequestMapping("/welcome")
    public String welcome(HttpServletRequest req) {
        String name = req.getParameter("name");
        req.setAttribute("name", name);
        return "Welcome";
    }
}
```

Home.jsp

```
<form action="welcome"><font color=green><b>
Enter Your Name : <input type="text"
name="name"/><br>
<input type="submit" value="Submit"/>
</form></b></font>
```

Welcome.jsp

```
<body>
<font color=green><b>Welcome
${name}</b></font>
</body>
```

← → ↻ ⓘ localhost:8123/home

Enter Your Name : Valan

Submit

← → ↻ ⓘ localhost:8123/welcome?name=Valan

Welcome Valan

Spring Boot MVC



Spring Boot MVC.

```
package com.wipro.pack;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
@Controller
public class WebAppController {
    @RequestMapping("/home")
    public String home() {
        return "Home";
    }
    @RequestMapping("/welcome")
    public ModelAndView welcome(@RequestParam("name")
String name) {
        ModelAndView mv = new ModelAndView();
        mv.addObject("name", name);
        mv.setViewName("Welcome");
        return mv;
    }
}
```

Home.jsp

```
<form action="welcome"><font color=green><b>
Enter Your Name : <input type="text"
name="name"/><br>
<input type="submit" value="Submit"/>
</form></b></font>
```

Welcome.jsp

```
<body>
<font color=green><b>Welcome
${name}</b></font>
</body>
```

← → ↻ ⓘ localhost:8123/home

Enter Your Name : Valan

Submit

← → ↻ ⓘ localhost:8123/welcome?name=Valan

Welcome Valan

SB MVC Using Employee Model Object



SB MVC Using Employee Model Object.

Home.jsp

```
package com.wipro.pack;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;
@Controller
public class WebAppController {
    @RequestMapping("/home")
    public String home() {
        return "Home";
    }
    @RequestMapping("/welcome")
    public ModelAndView welcome(Employee emp) {
        ModelAndView mv = new ModelAndView();
        mv.addObject("emp", emp);
        mv.setViewName("Welcome");
        return mv;
    }
}
```

```
<form action="welcome"><font color=green><b>
Enter Your Id : <input type="text" name="id"/><br>
Enter Your Name : <input type="text"
name="name"/><br>
Enter Your Salary : <input type="text"
name="salary"/><br>
<input type="submit" value="Submit"/>
</form></b></font>
```

Welcome.jsp

```
<body>
<font color=green><b>Welcome ${emp.id} ,
${emp.name}, ${emp.salary}</b></font>
</body>
```

SB MVC Using Employee Model Object.

```
package com.wipro.pack;
public class Employee {
    private int id;
    private String name;
    private int salary;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
}
```

← → ↻ ⓘ localhost:8123/home

Enter Your Id :

Enter Your Name :

Enter Your Salary :

← → ↻ ⓘ localhost:8123/welcome?id=101&name=Valan&salary=2000

Welcome 101 , Valan, 2000

Summary

In this session, you have learned about:

- What is web app in spring boot.
- How to create a web app using properties file.
- How the spring boot accepts client data.
- How to create MVC in spring boot.
- How to create MVC with Employee model object.



Thank you