**Spring Core:**

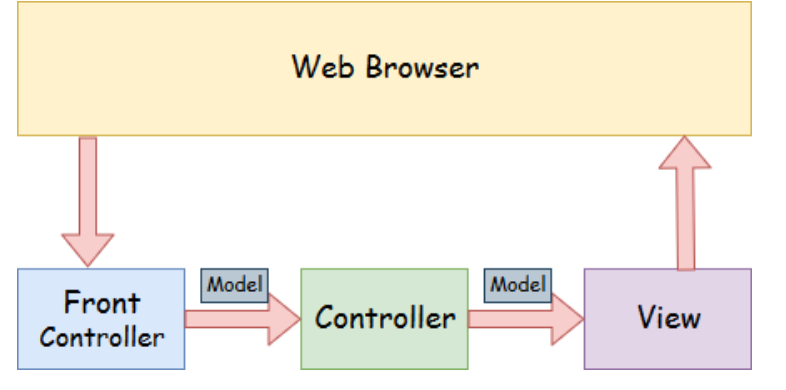
Spring Core Introduction / Overview, Spring Container, Dependency Injection, Metadata / Configuration. Spring MVC: Introduction / Developing Web applications with Spring MVC, Advanced Techniques, Spring Controllers, RESTful Web Services. Spring Boot: SPRING B00T Introduction, Using Spring Boot, Spring Boot Essentials. Spring Data JPA, Spring Data REST.

Spring MVC Tutorial

A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

## **Spring Web Model-View-Controller**

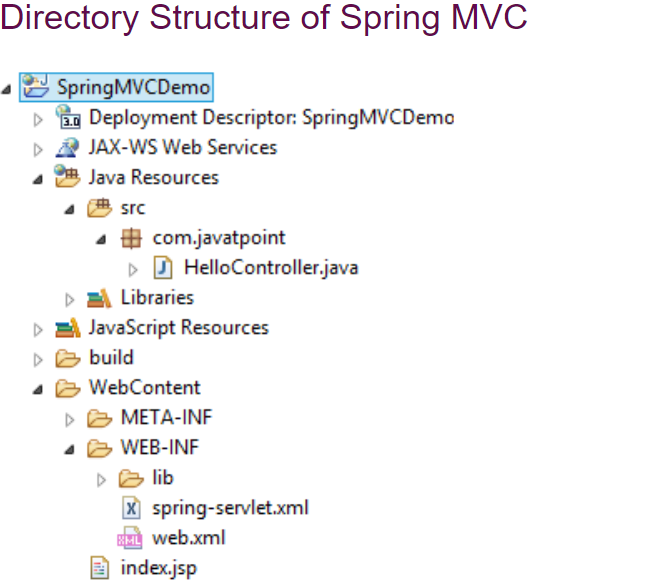


* **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.
* **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
* **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
* **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

## **Spring Web MVC Framework Example**

Let's see the simple example of a Spring Web MVC framework. The steps are as follows:

* Load the spring jar files or add dependencies in the case of Maven
* Create the controller class
* Provide the entry of controller in the web.xml file
* Define the bean in the separate XML file
* Display the message in the JSP page
* Start the server and deploy the project



**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.javatpoint</groupId>

  <artifactId>SpringMVC</artifactId>

  <packaging>war</packaging>

  <version>0.0.1-SNAPSHOT</version>

  <name>SpringMVC Maven Webapp</name>

  <url>http://maven.apache.org</url>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>3.8.1</version>

      <scope>test</scope>

    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-webmvc</artifactId>

    <version>5.1.1.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

    <groupId>javax.servlet</groupId>

    <artifactId>servlet-api</artifactId>

    <version>3.0-alpha-1</version>

</dependency>

  </dependencies>

  <build>

    <finalName>SpringMVC</finalName>

  </build>

</project>

## **What is Spring Boot**

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

It is a Spring module that provides the **RAD (Rapid Application Development)** feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.



In short, Spring Boot is the combination of **Spring Framework** and **Embedded Servers**.

In Spring Boot, there is no requirement for XML configuration (deployment descriptor). It uses convention over configuration software design paradigm that means it decreases the effort of the developer.

We can use Spring **STS IDE** or **Spring Initializr** to develop Spring Boot Java applications.

**Why should we use Spring Boot Framework?**

We should use Spring Boot Framework because:

* The dependency injection approach is used in Spring Boot.
* It contains powerful database transaction management capabilities.
* It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.
* It reduces the cost and development time of the application.

Along with the Spring Boot Framework, many other Spring sister projects help to build applications addressing modern business needs. There are the following Spring sister projects are as follows:

* **Spring Data:** It simplifies data access from the relational and **NoSQL** databases.
* **Spring Batch:** It provides powerful **batch** processing.
* **Spring Security:** It is a security framework that provides robust **security** to applications.
* **Spring Social:** It supports integration with **social networking** like LinkedIn.
* **Spring Integration:** It is an implementation of Enterprise Integration Patterns. It facilitates integration with other **enterprise applications** using lightweight messaging and declarative adapters.

## **Advantages of Spring Boot**

* It creates **stand-alone** Spring applications that can be started using Java **-jar**.
* It tests web applications easily with the help of different **Embedded** HTTP servers such as **Tomcat, Jetty,** etc. We don't need to deploy WAR files.
* It provides opinionated '**starter**' POMs to simplify our Maven configuration.
* It provides **production-ready** features such as **metrics, health checks,** and **externalized configuration**.
* There is no requirement for **XML** configuration.
* It offers a **CLI** tool for developing and testing the Spring Boot application.
* It offers the number of **plug-ins**.
* It also minimizes writing multiple **boilerplate codes** (the code that has to be included in many places with little or no alteration), XML configuration, and annotations.
* It **increases productivity** and reduces development time.

## **Limitations of Spring Boot**

Spring Boot can use dependencies that are not going to be used in the application. These dependencies increase the size of the application.

## **Goals of Spring Boot**

The main goal of Spring Boot is to reduce **development, unit test,** and **integration test** time.

* Provides Opinionated Development approach
* Avoids defining more Annotation Configuration
* Avoids writing lots of import statements
* Avoids XML Configuration.

By providing or avoiding the above points, Spring Boot Framework reduces **Development time, Developer Effort,** and **increases productivity**.