



Spring Boot

# Introduction to Spring Boot for Microservices

Create all the needed java based web components with  
little to no effort

# Introduction

- 
- Spring Boot is an open source Java-based framework used to create a Micro Service. It is developed by Pivotal Team. It is easy to create a stand-alone and production ready spring applications using Spring Boot. Spring Boot contains a comprehensive infrastructure support for developing a micro service and enables you to develop enterprise-ready applications.
  - Spring Boot offers a fast way to build applications. It looks at your classpath and at the beans you have configured, makes reasonable assumptions about what you are missing, and adds those items. With Spring Boot, you can focus more on business features and less on infrastructure.
  - Spring Boot is a microservice-based framework and making a production-ready application in it takes very less time.

# Goals

- 
- Provide a radically faster and widely accessible getting started experience
  - Be opinionated out of the box, but get out of the way quickly as requirements start to diverge from the defaults
  - Provide a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration)
  - Absolutely no code generation and no requirement for XML configuration

# Spring Boot Main Components

---

- Spring Boot Starters - combine a group of common or related dependencies into single dependency
- Spring Boot Auto Configurator - reduce the Spring Configuration
- Spring Boot CLI - run and test Spring Boot applications from command prompt
- Spring Boot Actuator – provides End Points and Metrics

# Spring Boot Features

---

## Spring Boot Starter Template

- Spring Boot starters are templates that contain a collection of all the relevant transitive dependencies that are needed to start a particular functionality.
- For example, If we want to create a Spring Web application then in a traditional setup, we would have included all required dependencies ourselves. It leaves the chances of version conflicts which ultimately result in more runtime exceptions.
- With Spring boot, to create such an application, all we need to import is spring-boot-starter-web dependency which internally imports all the required dependencies and adds to the project.

## Spring Boot Auto-Configuration

- Spring boot auto-configuration scans the classpath, finds the libraries in the classpath, and then attempts to guess the best configuration for them, and finally configure all such beans.
- Auto-configuration tries to be as intelligent as possible and will back away as we define more of our own configuration. Auto-configuration is enabled with `@EnableAutoConfiguration` annotation.

## Embedded Servers

- Spring boot applications always include tomcat as default embedded server start at default port **8080**. It enables us to run any Spring boot application from the command prompt without including any complex server infrastructure.
- We can exclude Tomcat and include any other embedded server if we want, e.g. Jetty Server. Or we can make exclude the server environment altogether. It's all configuration-based.

# Spring Boot Features contd...

---

## Bootstrapping the Application

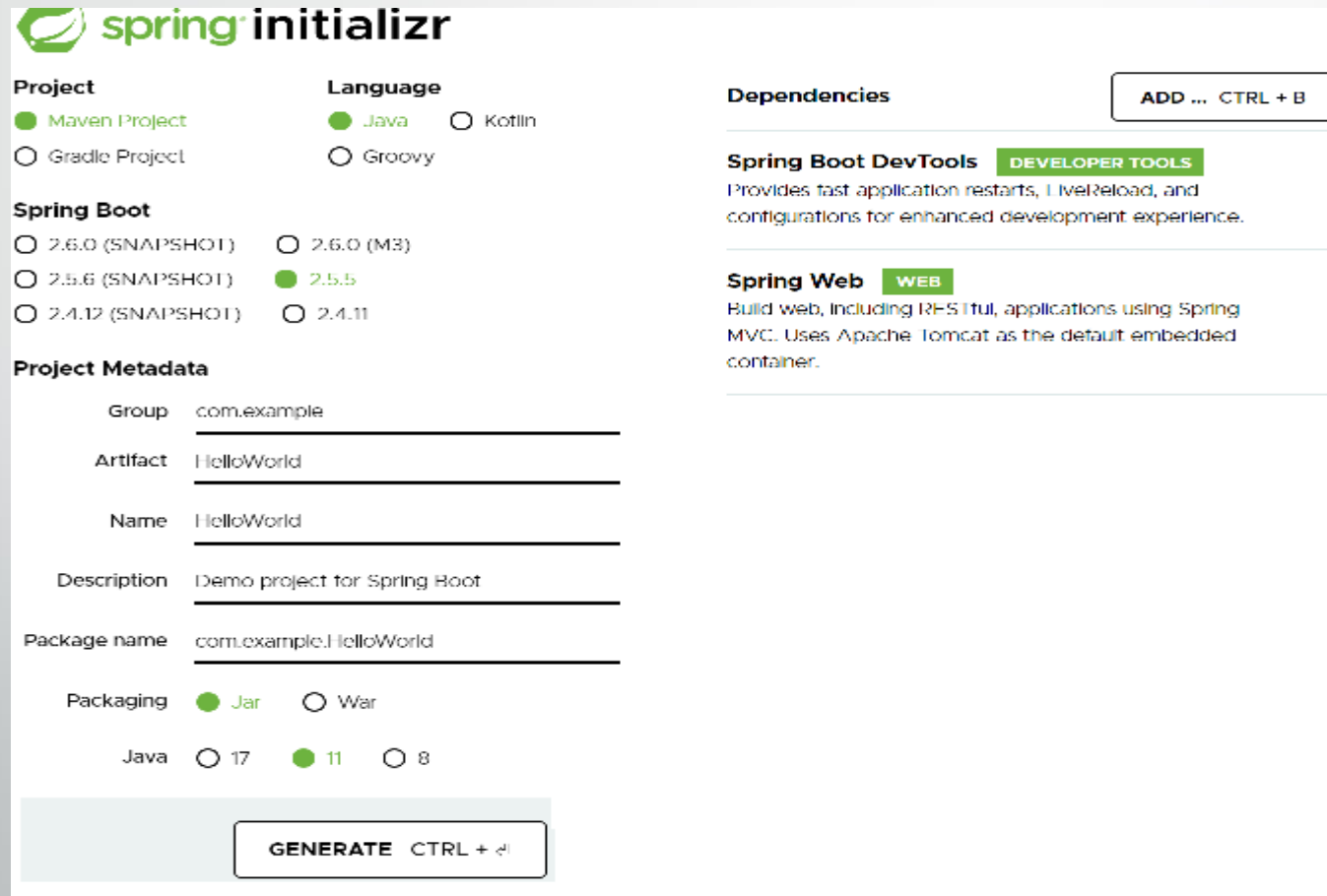
- To run spring boot application, we need to use [@SpringBootApplication](#) annotation. That's equivalent to [@Configuration](#), [@EnableAutoConfiguration](#), and [@ComponentScan](#) together.
- [@SpringBootApplication](#) enables the scanning of config classes, files and load them into spring context. In the following example, the program execution start with [main\(\)](#) method.
- When we run the application, auto-configuration feature starts loading all the configuration files, configure them and bootstrap the application based on [application properties](#) in application.properties file in [/resources](#) folder.

## Advantages of Spring Boot

- Spring boot helps in resolving dependency conflict. It identifies required dependencies and import them for you.
- Spring boot has information of compatible version for all dependencies. It minimizes the runtime class loader issues.
- Spring boot's "opinionated defaults configuration" approach helps in configuring most important pieces behind the scene. Override them only when you need. Otherwise everything just works, perfectly. It helps in avoiding boilerplate code, annotations and XML configurations.
- Spring boot provides embedded HTTP server Tomcat so that you can develop and test quickly.
- Spring boot has excellent integration with IDEs like eclipse and IntelliJ IDEA.

# Setup

- Let's use [Spring Initializr](#) to generate the base for our project.

The image shows the Spring Initializr web interface. It is a form for generating a new Spring project. The form is divided into several sections: Project, Language, Spring Boot, Project Metadata, Dependencies, and a bottom bar with a GENERATE button. The Project section has radio buttons for Maven Project (selected) and Gradle Project. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 2.6.0 (SNAPSHOT), 2.6.0 (M3), 2.5.6 (SNAPSHOT), 2.5.5 (selected), and 2.4.12 (SNAPSHOT), 2.4.11. The Project Metadata section has input fields for Group (com.example), Artifact (HelloWorld), Name (HelloWorld), Description (Demo project for Spring Boot), and Package name (com.example.HelloWorld). The Packaging section has radio buttons for Jar (selected) and War. The Java section has radio buttons for 17, 11 (selected), and 8. The Dependencies section has a button to ADD ... (CTRL + B) and two sections: Spring Boot DevTools (DEVELOPER TOOLS) and Spring Web (WEB).

**spring initializr**

**Project**

☒ Maven Project ☐ Gradle Project

**Language**

☒ Java ☐ Kotlin ☐ Groovy

**Spring Boot**

☐ 2.6.0 (SNAPSHOT) ☐ 2.6.0 (M3) ☐ 2.5.6 (SNAPSHOT) ☒ 2.5.5 ☐ 2.4.12 (SNAPSHOT) ☐ 2.4.11

**Project Metadata**

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 17 ☒ 11 ☐ 8

**Dependencies** ADD ... CTRL + B

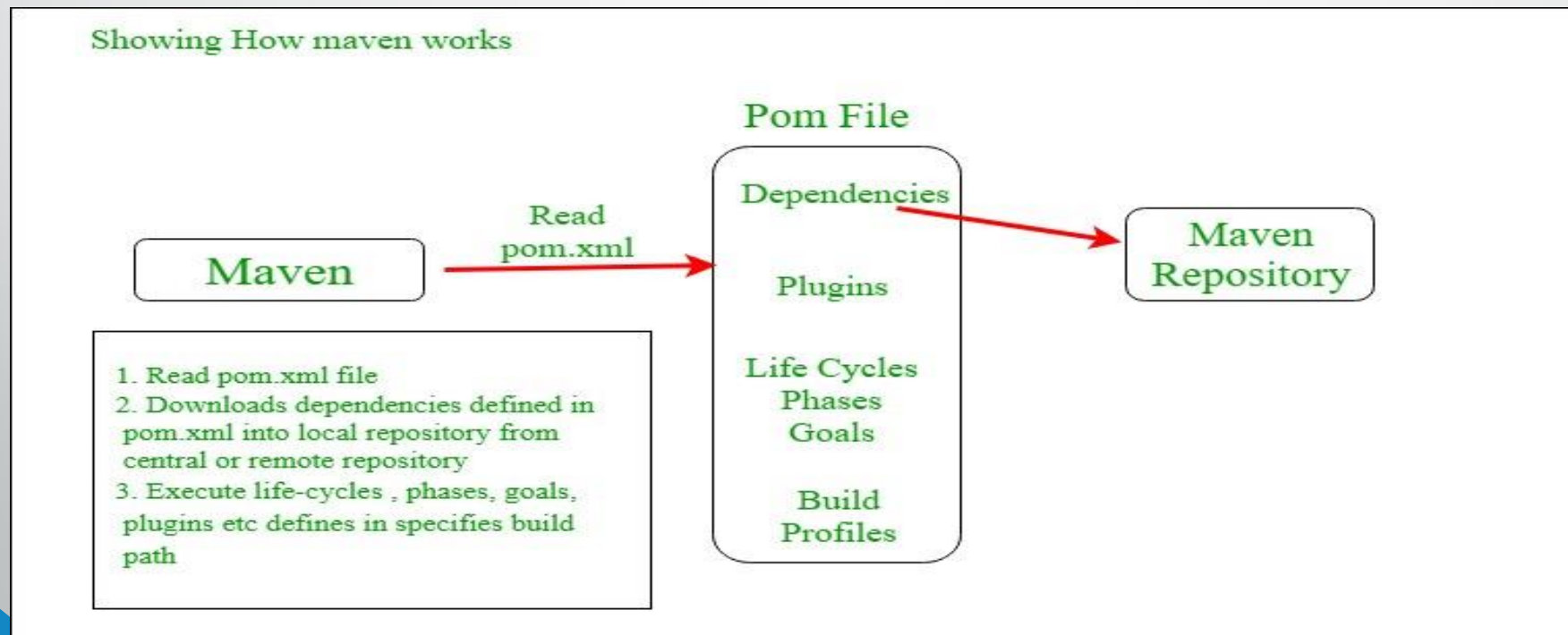
**Spring Boot DevTools** DEVELOPER TOOLS  
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

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

**GENERATE** CTRL + G

# What is Maven?

- Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process and it is a tool that can be used for building and managing any Java-based project.

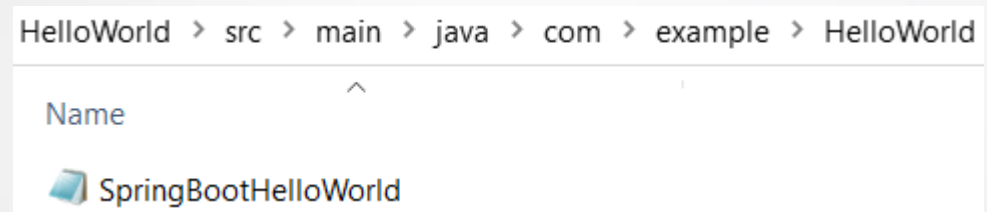
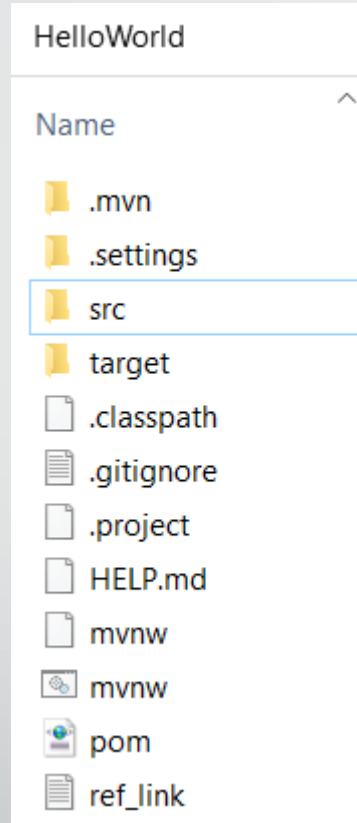




# What maven does?

- We can easily build a project using maven.
- We can add jars and other dependencies of the project easily using the help of maven.
- Maven provides project information (log document, dependency list, unit test reports etc.)
- Maven is very helpful for a project while updating central repository of JARs and other dependencies.
- With the help of Maven we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.
- Using maven we can easily integrate our project with source control system (such as Subversion or Git).

# Directory Structure



# What is POM.XML

---

- A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains **information about the project and configuration details used by Maven to build the project**. It contains default values for most projects

# pom.xml

- The generated project relies on the Boot parent:

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.5</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>HelloWorld</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>HelloWorld</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

# Application Configuration

- Next, we'll configure a simple *main* class for our application

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

- Notice how we're using **@SpringBootApplication** as our primary application configuration class.
- Change default server port from properties file
- *server.port* changes the server port from the default 8080 to 9000; there are of course many more [Spring Boot properties available](#).

application.properties

server.port=9000

application.yml

server:  
 port : 9000

# Building REST(REpresentational State Transfer) services with Spring

- REST has quickly become the de-facto standard for building web services on the web because they're easy to build and easy to consume.
- We use the REST application for developing and designing networked applications. It generates the HTTP request that performs CRUD operations on the data. Usually, it returns data in JSON or [XML](#) format.
- In Spring's approach to building RESTful web services, HTTP requests are handled by a controller. These components are identified by the `@RestController` annotation, and the **SpringBootHelloWorld** shown in the example. GET requests for / by returning a new instance of the **SpringBootHelloWorld** class
- This code uses Spring `@RestController` annotation, which marks the class as a controller where every method returns a domain object instead of a view. It is shorthand for including both `@Controller` and `@ResponseBody`.
- Every request handling method of the controller class automatically serializes return objects into *HttpResponse*.

# HelloWorld Example

- We'll define a simple [controller](#) and a basic home page with a welcome message

```
package com.example.HelloWorld;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.web.bind.annotation.*;

@RestController
@EnableAutoConfiguration
public class SpringBootHelloWorld {

    @RequestMapping("/")
    String home() {
        return "Hello Spring Boot!!!";
    }

    public static void main(String[] args) throws Exception {
        SpringApplication.run(SpringBootHelloWorld.class, args);
    }
}
```

# Run the application

- Go to command prompt and change directory(cd) to root folder HelloWorld
- mvnw spring-boot:run (in windows)
- ./mvnw (in Ubuntu)

```
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd E:\SPRING_BOOT_2021\HelloWorld

C:\Users\Admin>e:

E:\SPRING_BOOT_2021\HelloWorld>mvnw spring-boot:run
```

```
[INFO]
[INFO]
[INFO] --- spring-boot-maven-plugin:2.5.5:run (default-cli) @ HelloWorld ---
[INFO] Attaching agents: []

  ____ _
 / ___ \| | | |
/ /   \| |_| |
\ \   /| | | |
 \___/\_|_|_|_|
       | |
       |_|

:: Spring Boot ::
              (v2.5.5)

2021-10-08 12:58:51.215 INFO 12124 --- [main] c.e.HelloWorld.SpringBootHelloWorld : Starting SpringBoot
HelloWorld using Java 11.0.11 on cseadmin with PID 12124 (E:\SPRING_BOOT_2021\HelloWorld\target\classes started by Admin
in E:\SPRING_BOOT_2021\HelloWorld)
2021-10-08 12:58:51.218 INFO 12124 --- [main] c.e.HelloWorld.SpringBootHelloWorld : No active profile s
et, falling back to default profiles: default
2021-10-08 12:58:54.787 INFO 12124 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
with port(s): 8082 (http)
2021-10-08 12:58:54.837 INFO 12124 --- [main] o.apache.catalina.core.StandardService : Starting service [T
omcat]
2021-10-08 12:58:54.838 INFO 12124 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet en
gine: [Apache Tomcat/9.0.53]
2021-10-08 12:58:54.990 INFO 12124 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring
embedded WebApplicationContext
2021-10-08 12:58:54.991 INFO 12124 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplication
Context: initialization completed in 3560 ms
2021-10-08 12:58:55.674 INFO 12124 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on p
ort(s): 8082 (http) with context path: /
```

← → ↻ ⓘ http://localhost:8082

Hello Spring Boot!!!



# Exercise

---

- Develop a web application which reads employee information and print on the screen.
- Develop a web application which reads students details using @RequestParam and display the information.