Java SE & Spring

Module 2: Spring

04.Introduction to Spring Boot

What is Spring Boot?

Spring Boot is a Framework from “The Spring Team” to ease the bootstrapping and development of new Spring Applications.

It provides defaults for code and annotation configuration to quick start new Spring projects within no time.

It follows “Opinionated Defaults Configuration” Approach to avoid lot of boilerplate code and configuration to improve Development, Unit Test and Integration Test Process.

The main goal of Spring Boot Framework is to reduce Development, Unit Test and Integration Test time and to ease the development of Production ready web applications very easily compared to existing Spring Framework, which really takes more time.

Further reading: https://docs.spring.io/spring-boot/docs/current/reference/html/

Advantages of Spring Boot

- Easy to develop Spring Based applications with Java or Groovy.
- Reduces lots of development time and increases productivity.
- Provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- Avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- Provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle

Main Goal of the Spring Boot

The main goal of Spring Boot Framework is to reduce Development, Unit Test and Integration Test time and to ease the development of Production ready web applications very easily compared to existing Spring Framework, which really takes more time.

- To avoid XML Configuration completely
- To avoid defining more Annotation Configuration(It combined some existing Spring Framework Annotations to a simple and single Annotation)
- To avoid writing lots of import statements
- To provide some defaults to quick start new projects within no time.
- To provide Opinionated Development approach.

Key Components of Spring Boot

Spring Boot Framework has mainly four major Components:

- Spring Boot Starters
- Spring Boot AutoConfigurator
- Spring Boot CLI
- Spring Boot Actuator

Spring Boot Starters

Spring Boot Starters is one of the major key features or components of Spring Boot Framework. The main responsibility of Spring Boot Starter is to combine a group of common or related dependencies into single dependencies. We will explore this statement in detail with one example.

For instance, we would like to develop a Spring WebApplication with Tomcat WebServer. Then we need to add the following minimal jar dependencies in our Maven's pom.xml

- Spring core Jar file(spring-core-xx.jar)
- Spring Web Jar file(spring-web-xx.jar)
- Spring Web MVC Jar file(spring-webmvc-xx.jar)
- Servlet Jar file(servlet-xx.jar)

Spring Boot Starters

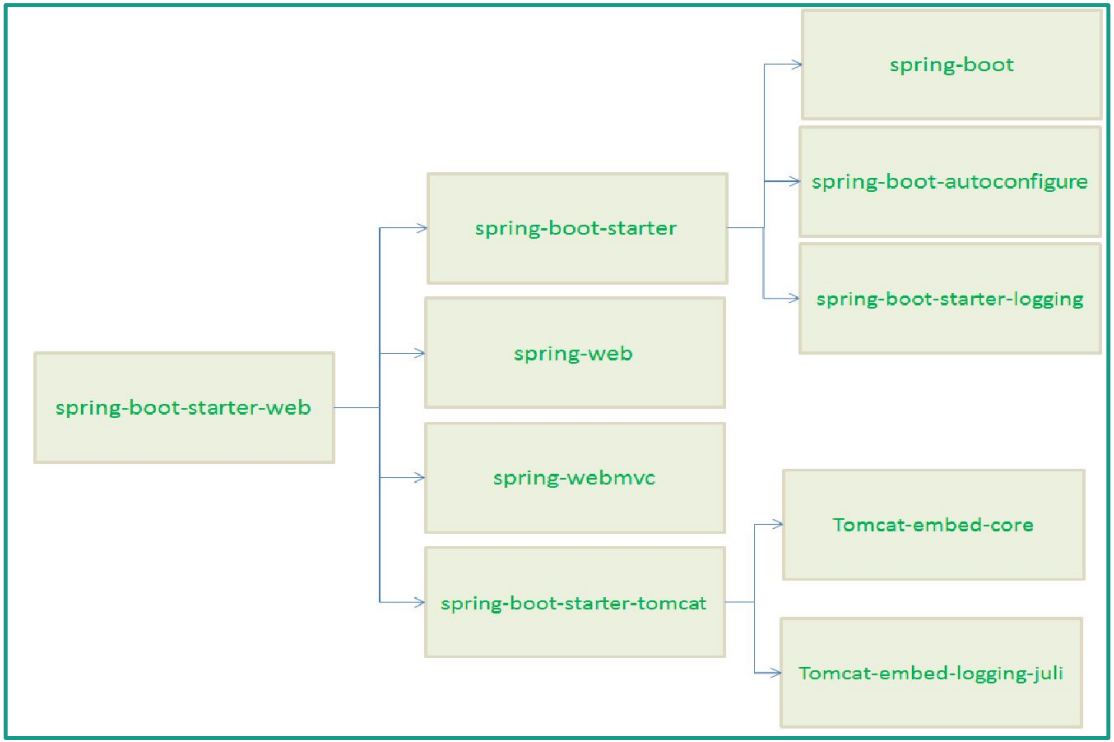
We need to define lot of dependencies in our build files. It is very tedious and cumbersome tasks for a Developer. And also it increases our build file size.

What is the solution to avoid this much dependencies definitions in our build files? The solution is Spring Boot Starter component. Spring Boot Starter component combines all related jars into single jar file so that we can add only jar file dependency to our build files. Instead of adding above 4 jars files to our build file, we need to add one and only one jar file: “spring-boot-starter-web” jar file.

When we add “spring-boot-starter-web” jar file dependency to our build file, then Spring Boot Framework will automatically download all required jars and add to our project classpath.

Further reading: https://www.journaldev.com/7989/key-components-and-internals-of-spring-boot-framework

Spring Boot Starters



Further reading: https://www.journaldev.com/7989/key-components-and-internals-of-spring-boot-framework

Spring Boot Starters

Major Advantages of Spring Boot Starter

Spring Boot Starter reduces defining many dependencies simplify project build dependencies. Spring Boot Starter simplifies project build dependencies.

Further reading: https://www.journaldev.com/7989/key-components-and-internals-of-spring-boot-framework

Spring Boot Auto Configuration

Another important key component of Spring Boot Framework is Spring Boot AutoConfigurator. Most of the Spring IO Platform (Spring Framework) Critics opinion is that “To develop a Spring-based application requires lot of configuration (Either XML Configuration of Annotation Configuration). Then how to solve this problem.

The solution to this problem is Spring Boot AutoConfigurator. The main responsibility of Spring Boot AutoConfigurator is to reduce the Spring Configuration. If we develop Spring applications in Spring Boot,then We dont need to define single XML configuration and almost no or minimal Annotation configuration. Spring Boot AutoConfigurator component will take care of providing those information.

Further reading: https://www.journaldev.com/7989/key-components-and-internals-of-spring-boot-framework

Spring Boot Auto Configuration

For instance, if we want to declare a Spring MVC application using Spring IO Platform, then we need to define lot of XML Configuration like views, view resolvers etc. But if we use Spring Boot Framework, then we dont need to define those XML Configuration. Spring Boot AutoConfigurator will take of this.

If we use “spring-boot-starter-web” jar file in our project build file, then Spring Boot AutoConfigurator will resolve views, view resolvers etc. automatically.

Further reading: https://www.journaldev.com/7989/key-components-and-internals-of-spring-boot-framework

Spring Initializr

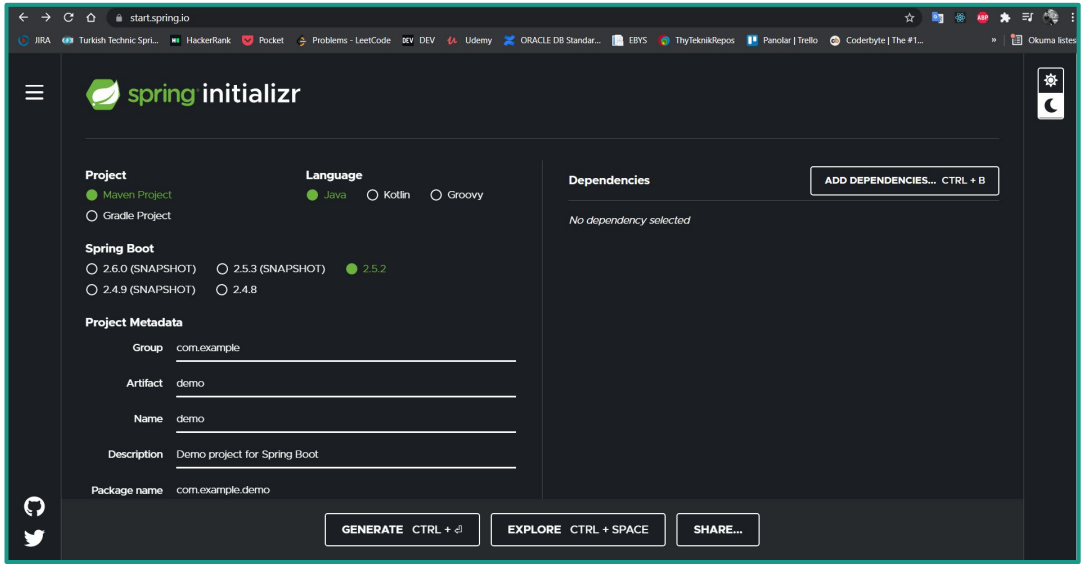
Starting from Spring Boot 4 Components, Spring Boot Framework has another important key component that is: **Spring Boot Initializr**.

The Spring Team has provided a Web Interface for Spring Boot Initializr at “https://start.spring.io/” to quick start the Development of Spring Boot Applications using Maven/Gradle build tool very easily.

The main aim of Spring Boot Initializr is to start new project development within no time. It provides all required project structure and base build script files to quick start Spring Boot application without wasting time. It reduces development time.

Spring Web Initializr

The Spring Team has provided a Web Interface for Spring Boot Initializr at [Spring Initializr](). We can use it to create our new Project's base structure for Maven/Gradle build tools. It not only generates Project Base structure but also provides all required Spring Boot Jar files.



Spring and Spring Boot Annotations

Core Spring

- **@Bean** - Annotated method produces a bean managed by the Spring IoC container. Components (**@Component, @Service, @Repository, @Controller** etc.) are automatically registered as Spring Beans.

Stereotype annotations

- **@Component** - Marks annotated class as a bean found by the component-scanning and loaded into the application context
- **@Controller** - Marks annotated class as a bean for Spring MVC containing request handler

---

Further reading: https://dzone.com/articles/frequently-used-annotations-in-spring-boot-applica

Spring and Spring Boot Annotations

- **@RestController** - Marks annotated class as a **@Controller** bean and adds **@ResponseBody** to serialize returned results as messages
- **@Configuration** - Marks annotated class as a Java configuration defining beans
- **@Service** - Marks annotated class as a bean (as convention usually containing business logic)
- **@Repository** - Marks annotated class as a bean (as convention usually providing data access) and adds auto-translation from **SQLException** to **DataAccessExceptions**

Spring Boot Annotations

- **@SpringBootApplication** - Combination of *@SpringBootConfiguration*, *@EnableAutoConfiguration*, *@ConfigurationPropertiesScan* and *@ComponentScan*

Further reading: https://dzone.com/articles/frequently-used-annotations-in-spring-boot-applica

Questions ?

That’s all for this course.
Thank you for your attendance!