Spring Boot Starter Web as Single Dependency

In this lesson, we'll look at how the Spring Boot starter web can serve as a single web dependency.

WE'LL COVER THE FOLLOWING

- Spring cloud
- The Maven plugin

The application has a dependency on the library `spring-boot-starter-web`. This dependency integrates the Spring framework, the Spring web framework, and an environment for the processing of HTTP requests.

The **default** for the processing of the HTTP requests is a **Tomcat server** which runs *embedded as part of the application*.

> Thus, the dependency on `spring-boot-starter-web` would be enough as a **sole dependency** for the application!

The dependency on `spring-boot-starter-test` is necessary for **tests**.

> **Note:** The code for the test is not part of this course.

Spring cloud

Spring Cloud is a collection of extensions for Spring Boot which are useful for **cloud applications** and for **microservices**.

Spring Cloud contains **additional starters**. To be able to use the Spring Cloud starters, an entry has to be inserted into the dependency-management section in the `pom.xml` for importing the information about the Spring Cloud starter.

The `pom.xml` files in the examples already contain the required import for this.

The Maven plugin

The Maven plugin `spring-boot-maven-plugin` is necessary to build a **Java JAR** that starts an environment with the Tomcat server and the application.

```
mvn clean package
```

The above command deletes the old build results and builds a new JAR.

> **JAR** is a Java file format which contains **all the code** for an application.

Maven gives this JAR file a name that is derived from the project name. It can be started with:

```
java -jar simplest-spring-boot-0.0.1-SNAPSHOT.jar
```

Spring Boot can also generate **WARs** (web archives) which can be deployed on a Java web server like Tomcat or a Java application server.

QUIZ

1

Why is **Tomcat server** NOT considered to be a dependency of our application?

COMPLETED 0%

1 of 3

In the *next lesson*, we’ll look at how Spring Boot fulfills the communication requirement.