

/*****

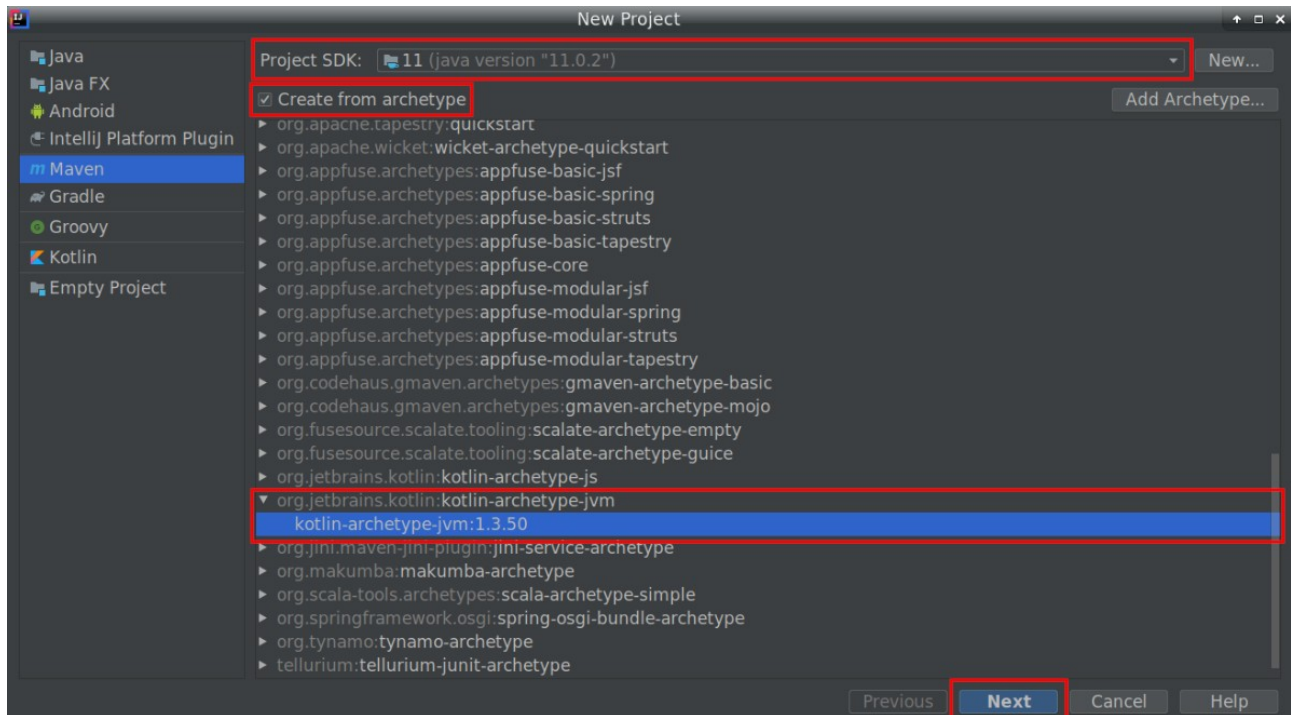
* Setari pom pentru maven si gradle

* Setari necesare pentru a starta un proiect kotlin spring boot

*****/

-> Setari pentru MAVEN

- Alegeți ca tip de proiect „Maven”, selectați versiunea 11 de Java SDK, apoi bifați „Create from archetype”. Din lista de mai jos, selectați „org.jetbrains.kotlin:kotlin-archetype-jvm” → „kotlin-archetype-jvm:<versiune_Kotlin>”, apoi apăsați pe „Next”.



- Modificari pom.xml pentru Spring Boot:

a) In tagul <project> se adauga:

```
<parent>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>2.1.9.RELEASE</version>
```

```
    <relativePath/> <!-- lookup parent from repository -->
```

```
</parent>
```

```
m pom.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>com.sd.laborator</groupId>
7     <artifactId>exempluMaven</artifactId>
8     <version>1.0.0</version>
9     <packaging>jar</packaging>
10
11     <parent>
12         <groupId>org.springframework.boot</groupId>
13         <artifactId>spring-boot-starter-parent</artifactId>
14         <version>2.1.9.RELEASE</version>
15         <relativePath/> <!-- lookup parent from repository -->
16     </parent>
17
18     <name>com.sd.laborator exempluMaven</name>
19
20     <properties>
21         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22         <kotlin.version>1.3.60</kotlin.version>
23         <kotlin.code.style>official</kotlin.code.style>
24         <junit.version>4.12</junit.version>
25     </properties>
26
```

project › parent

b) In tagul <plugins> se adauga:

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

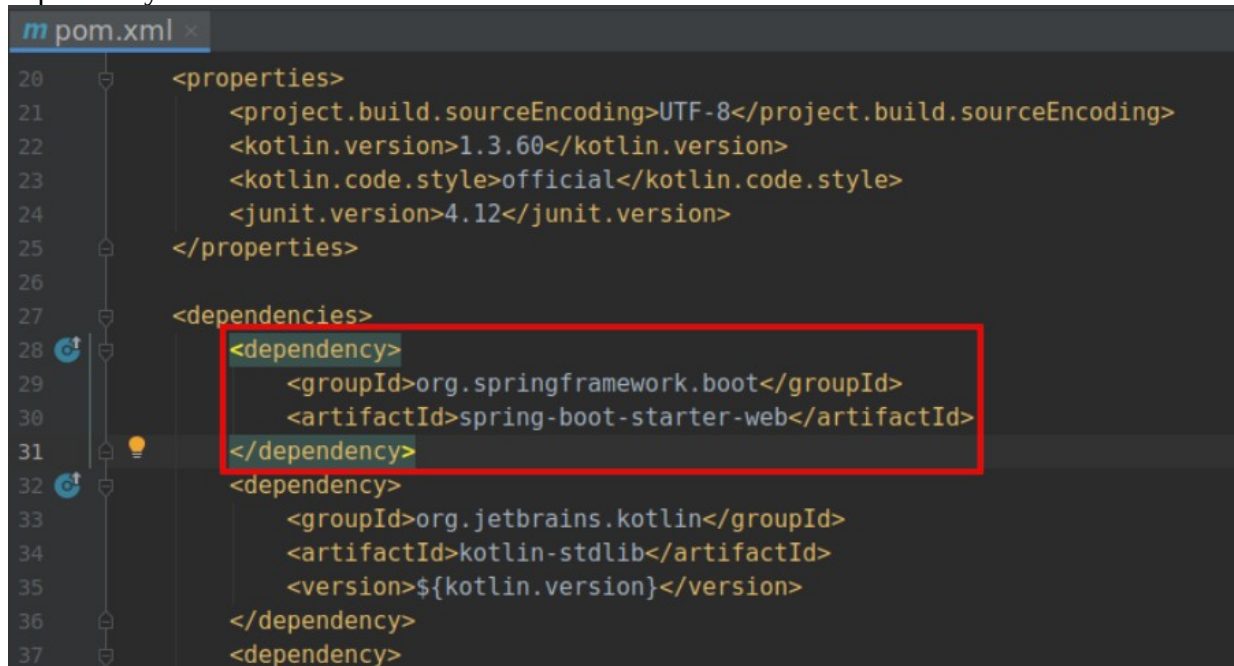
</plugin>

```
m pom.xml x
46
47 <build>
48     <sourceDirectory>src/main/kotlin</sourceDirectory>
49     <testSourceDirectory>src/test/kotlin</testSourceDirectory>
50
51     <plugins>
52         <plugin>
53             <groupId>org.springframework.boot</groupId>
54             <artifactId>spring-boot-maven-plugin</artifactId>
55         </plugin>
56         <plugin>
57             <groupId>org.jetbrains.kotlin</groupId>
58             <artifactId>kotlin-maven-plugin</artifactId>
59             <version>${kotlin.version}</version>
60             <executions>
61                 <execution>
62                     <id>compile</id>
63                     <phase>compile</phase>
64                     <goals>
65                         <goal>compile</goal>

```

c) Se adauga urmatoarele dependente in tagul <dependencies>:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-reflect</artifactId>
    <version>${kotlin.version}</version>
</dependency>
```



d) Adaugarea in elementul <plugin> (Clasele în Kotlin sunt, în mod implicit, marcate ca și final, deci nu se pot moșteni decât dacă dezvoltatorul le marchează explicit ca open (de exemplu, open class MyClass ...). Spring necesită ca acele clase ce vor primi anumite tipuri de adnotări (cum ar fi @Component sau @Service) să fie moștenibile, adică marcate cu open.):

```
<dependencies>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-maven-allopen</artifactId>
        <version>${kotlin.version}</version>
    </dependency>
</dependencies>
```

```

65     <plugins>
66       <plugin>
67         <groupId>org.jetbrains.kotlin</groupId>
68         <artifactId>kotlin-maven-plugin</artifactId>
69         <version>${kotlin.version}</version>
70         <executions>
71           <execution>
72             <id>compile</id>
73             <phase>compile</phase>
74             <goals>
75               <goal>compile</goal>
76             </goals>
77           </execution>
78           <execution>
79             <id>test-compile</id>
80             <phase>test-compile</phase>
81             <goals>
82               <goal>test-compile</goal>
83             </goals>
84           </execution>
85         </executions>
86         <dependencies>
87           <dependency>
88             <groupId>org.jetbrains.kotlin</groupId>
89             <artifactId>kotlin-maven-allopen</artifactId>
90             <version>${kotlin.version}</version>
91           </dependency>
92         </dependencies>
93       </plugin>

```

e) Adaugarea pluginului spring ca dependenta la compilare in tagul **<execution>**:

```

<configuration>
  <compilerPlugins>
    <plugin>spring</plugin>
  </compilerPlugins>
</configuration>

```

```

65      <plugins>
66          <plugin>
67              <groupId>org.jetbrains.kotlin</groupId>
68              <artifactId>kotlin-maven-plugin</artifactId>
69              <version>${kotlin.version}</version>
70              <executions>
71                  <execution>
72                      <id>compile</id>
73                      <phase>compile</phase>
74                      <goals>
75                          <goal>compile</goal>
76                      </goals>
77                      <configuration>
78                          <compilerPlugins>
79                              <plugin>spring</plugin>
80                          </compilerPlugins>
81                      </configuration>
82                  </execution>
83              </executions>

```

f) Pentru productivitate sporita, adaugarea dependentei:

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <version>${project.parent.version}</version>
</dependency>

```

Adăugând această dependență, plugin-ul Spring Boot va reîncărca automat artefactele rezultate din compilarea surselor, atunci când acestea sunt modificate de utilizator. Deci, odată pornite aplicația și server-ul Tomcat (folosind goal-ul `spring-boot:run`), dacă modificați sursele și vreți să vedeți rezultatul modificărilor, doar le compilați cu lifecycle-ul Maven `compile` și atât. Aplicația vi se reîncarcă automat (dacă nu sunt erori!).

În acest moment, s-a obținut un proiect Maven creat folosind un șablon (archetype) pentru Kotlin, în care ați actualizat fișierul POM pentru a include ca și dependențe „Spring Boot”, „Spring Boot Developer Tools” și „Spring Boot Starter Web”. Cea din urmă este necesară pentru construirea de aplicații web, folosind Spring MVC.

*Dependente suplimentare

1) SQLite

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.28.0</version>
</dependency>
```

2) Starter amqp(advanced message queue protocol) - RabbitMq

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

3) RxKotlin

a) File → Project Structure... → Libraries → click pe pictograma „+“ → From Maven.
Introduceți următoarele în căsuța de căutare și apăsați „Ok”:
io.reactivex.rxjava3:rxkotlin:3.0.0

b)

```
<dependency>
    <groupId>io.reactivex.rxjava3</groupId>
    <artifactId>rxkotlin</artifactId>
    <version>3.0.0</version>
</dependency>
```

4) Corutine

```
<dependency>
    <groupId>org.jetbrains.kotlinx</groupId>
    <artifactId>kotlinx-coroutines-core</artifactId>
    <version>1.3.3</version>
</dependency>
<dependency>
    <groupId>org.jetbrains.kotlinx</groupId>
    <artifactId>kotlinx-coroutines-debug</artifactId>
    <version>1.3.1</version>
</dependency>
```


5) Kotlin serialization

```
<properties>
  <serialization.version>0.20.0</serialization.version>
</properties>
<dependency>
  <groupId>org.jetbrains.kotlin</groupId>
  <artifactId>kotlinx-serialization-runtime</artifactId>
  <version>${serialization.version}</version>
</dependency>
<plugin>
  <groupId>org.jetbrains.kotlin</groupId>
  <artifactId>kotlin-maven-plugin</artifactId>
  <version>${kotlin.version}</version>
  <executions>
    <execution>
      <id>compile</id>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <compilerPlugins><plugin>kotlinx-serialization</plugin>
  </compilerPlugins>
</configuration>
<dependencies>
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-maven-serialization</artifactId>
    <version>${kotlin.version}</version>
  </dependency>
</dependencies>
</plugin>
```

6) Plugin pentru jar with dependencies

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>2.6</version>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <archive>
      <manifest>
        <mainClass><NUME_CLASA(EX:pachet.clasa)></mainClass>
      </manifest>
    </archive>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
</plugin>
```

```
<sourceDirectory>src/main/kotlin</sourceDirectory>
<testSourceDirectory>src/test/kotlin</testSourceDirectory>

<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>2.6</version>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
        <configuration>
          <archive>
            <manifest>
              <mainClass>org.example.HelloKt</mainClass>
            </manifest>
          </archive>
          <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
          </descriptorRefs>
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
```


7) Spring Kafka

```
<dependency>
  <groupId>org.springframework.kafka</groupId>
  <artifactId>spring-kafka</artifactId>
  <version>2.2.9.RELEASE</version>
</dependency>
```

```
<dependencies>
  <!-- Dependenta specifica Kafka pentru proiecte spring. -->
  <dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
    <version>2.2.9.RELEASE</version>
  </dependency>

  <!-- Dependente aduse de managerul maven. -->
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-stdlib</artifactId>
    <version>${kotlin.version}</version>
  </dependency>
  <dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-test-junit</artifactId>
```

8) Log pentru micronaut serverless

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.13.2</version>
  <scope>runtime</scope>
</dependency>
```

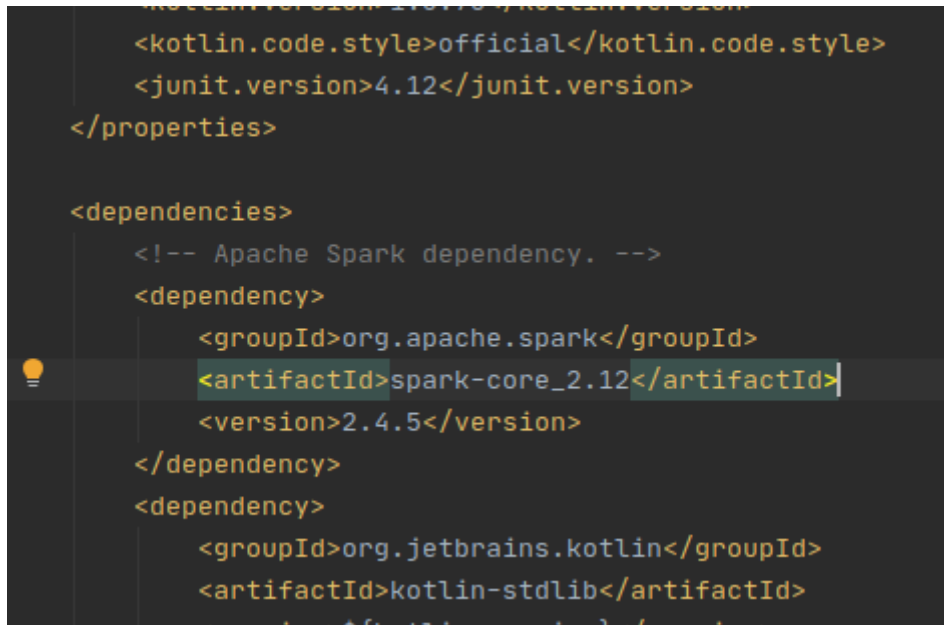
```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>

  <!-- Dependenta inclusa pentru logging. -->
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.13.2</version>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.jetbrains.k
    <artifactId>kotlin-stdlib-jdk8</artifactId>
    <version>${kotlinVersion}</version>
    <scope>compile</scope>
  </dependency>
```

Tag name: version
Description: The version of the
versions.
Version: 3.0.0+

9) Apache Spark RDD

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.12</artifactId>
  <version>2.4.5</version>
</dependency>
```

A screenshot of a code editor with a dark background. It shows an XML configuration for dependencies. The root element is <dependencies>. Inside, there is a comment <!-- Apache Spark dependency. --> followed by a <dependency> block. This block contains <groupId>org.apache.spark</groupId>, <artifactId>spark-core_2.12</artifactId>, and <version>2.4.5</version>. Below this, there is another <dependency> block for org.jetbrains.kotlin:kotlin-stdlib. The <artifactId>spark-core_2.12</artifactId> line is highlighted with a green selection bar. A lightbulb icon is visible on the left margin next to the first dependency block.

```
    <kotlin.code.style>official</kotlin.code.style>
    <junit.version>4.12</junit.version>
  </properties>

  <dependencies>
    <!-- Apache Spark dependency. -->
    <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.12</artifactId>
      <version>2.4.5</version>
    </dependency>
    <dependency>
      <groupId>org.jetbrains.kotlin</groupId>
      <artifactId>kotlin-stdlib</artifactId>
      <version>1.3.61</version>
    </dependency>
  </dependencies>
```

10) Apache Spark SQL

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.12</artifactId>
  <version>2.4.5</version>
</dependency>
```

11) Apache Spark Streaming

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming_2.12</artifactId>
  <version>2.4.5</version>
</dependency>
```

12) MySQL interface

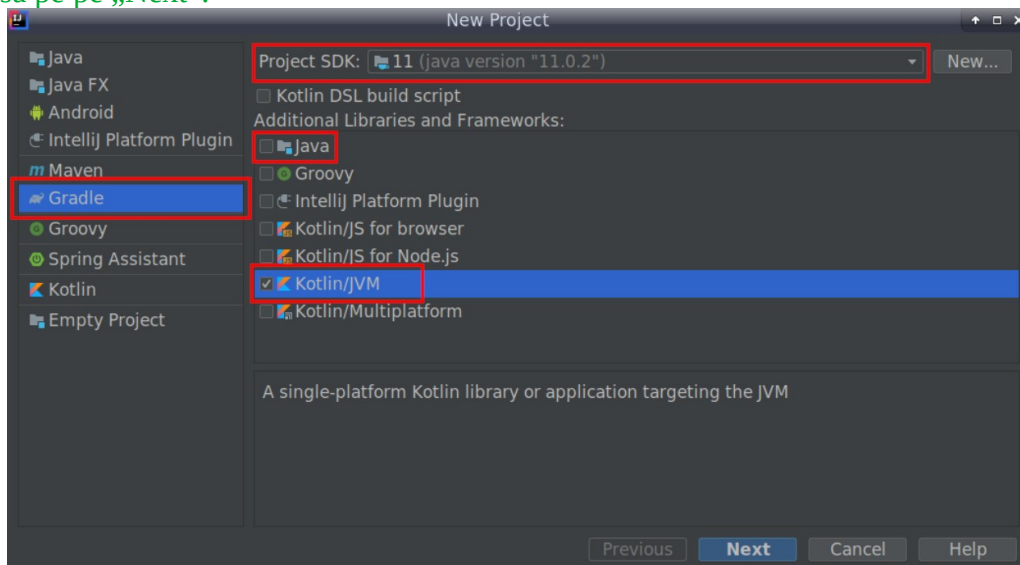
```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.19</version>
</dependency>
```

13) KHTTP

```
<repository>
  <id>jitpack.io</id>
  <url>https://jitpack.io</url>
</repository>
<dependency>
  <groupId>com.github.jkcclemens</groupId>
  <artifactId>khttp</artifactId>
  <version>-SNAPSHOT</version>
</dependency>
```

-> Setari pentru GRADLE

-Se va alege ca tip de proiect „Gradle”, selectați versiunea 11 de Java SDK, apoi, în secțiunea „Additional Libraries and Frameworks”, debifați „Java” și bifați „Kotlin/JVM”. Se va apăsa pe pe „Next”.

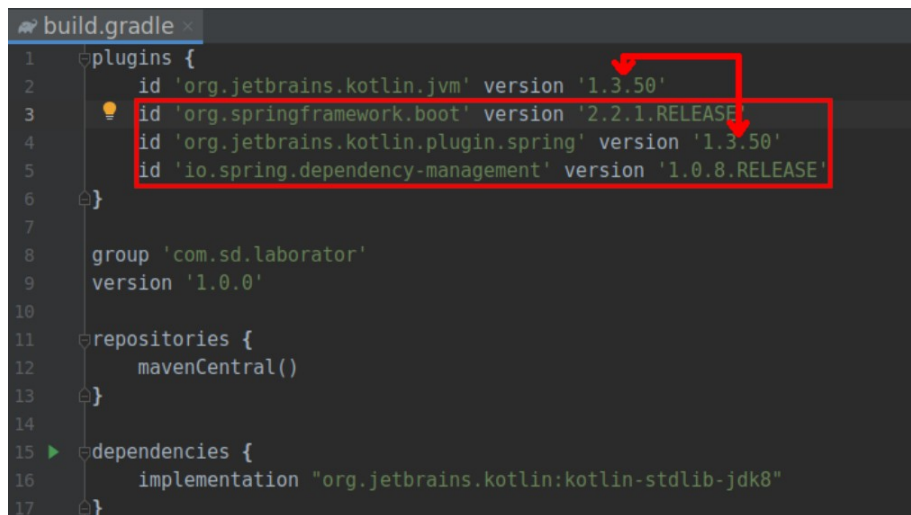


-Modificarea fisierului build.gradle

a) Adaugarea de pluginuri ca subordonati in sectiunea plugins:

```
id 'org.springframework.boot' version '2.2.1.RELEASE'
id 'org.jetbrains.kotlin.plugin.spring' version '<VERSIUNE_KOTLIN>'
id 'io.spring.dependency-management' version '1.0.8.RELEASE'
```

Înlocuiți <VERSIUNE_KOTLIN> cu versiunea de Kotlin existentă în repositories la momentul creării proiectului. Aceasta apare pe linia de mai sus, adăugată automat de IntelliJ.



```

1  plugins {
2      id 'org.jetbrains.kotlin.jvm' version '1.3.50'
3      id 'org.springframework.boot' version '2.2.1.RELEASE'
4      id 'org.jetbrains.kotlin.plugin.spring' version '1.3.50'
5      id 'io.spring.dependency-management' version '1.0.8.RELEASE'
6  }
7
8  group 'com.sd.laborator'
9  version '1.0.0'
10
11 repositories {
12     mavenCentral()
13 }
14
15 dependencies {
16     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
17 }

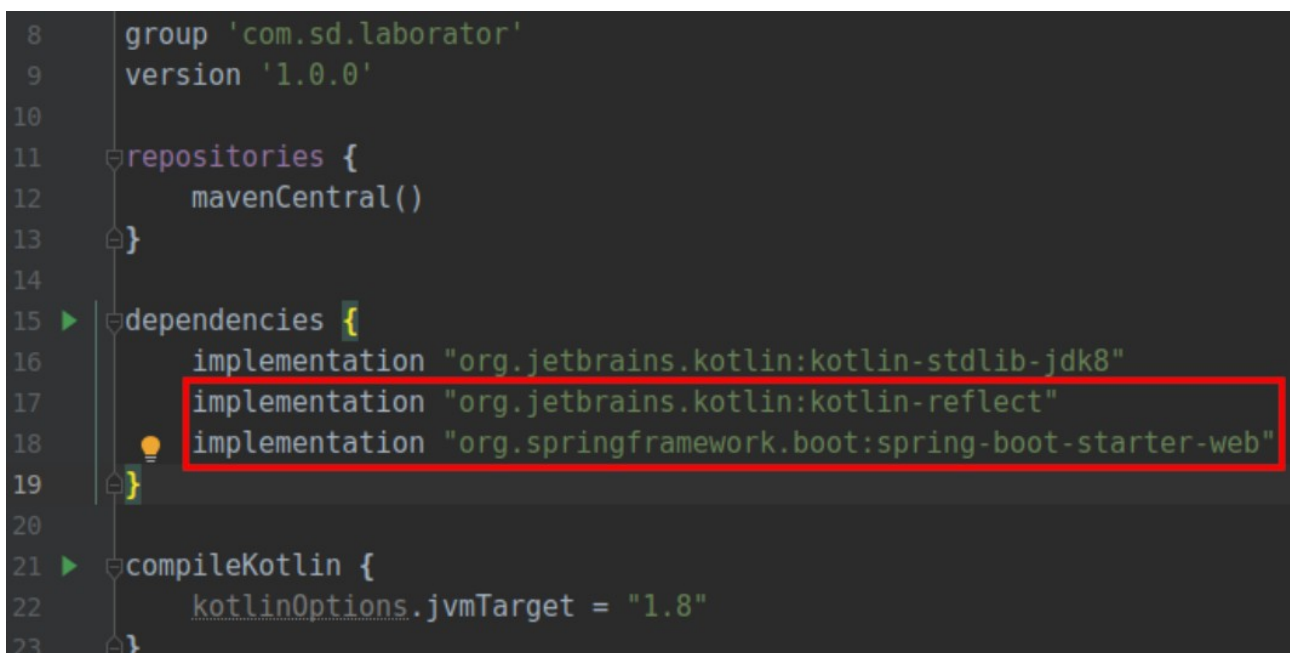
```

b) Adaugarea Kotlin reflect si Spring boot web application ca si dependente(elemente subordonate ale sectiunii dependencies):

```

implementation "org.jetbrains.kotlin:kotlin-reflect"
implementation "org.springframework.boot:spring-boot-starter-web"

```



```

8  group 'com.sd.laborator'
9  version '1.0.0'
10
11 repositories {
12     mavenCentral()
13 }
14
15 dependencies {
16     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
17     implementation "org.jetbrains.kotlin:kotlin-reflect"
18     implementation "org.springframework.boot:spring-boot-starter-web"
19 }
20
21 compileKotlin {
22     kotlinOptions.jvmTarget = "1.8"
23 }

```

c) Pentru productivitate sporita:

```

compileOnly "org.springframework.boot:spring-boot-devtools"

```

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    implementation "org.jetbrains.kotlin:kotlin-reflect"
    implementation "org.springframework.boot:spring-boot-starter-web"
    compileOnly "org.springframework.boot:spring-boot-devtools"
```

d) Adaugarea unei sectiuni noi in fisier:

```
configurations {
    compileOnly
    runtimeClasspath {
        extendsFrom compileOnly
    }
}
```

```
15 configurations {
16     compileOnly
17     runtimeClasspath {
18         extendsFrom compileOnly
19     }
20 }
21
22 dependencies {
23     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
24     implementation "org.jetbrains.kotlin:kotlin-reflect"
25     implementation "org.springframework.boot:spring-boot-starter-web"
26
27     compileOnly "org.springframework.boot:spring-boot-devtools"
28 }
```

Adăugând această dependență, plugin-ul Spring Boot va reîncărca automat artefactele rezultate din compilarea surselor, atunci când acestea sunt modificate de utilizator. Deci, odată pornite aplicația și server-ul Tomcat (folosind task-ul application/bootRun), dacă modificați sursele și vreți să vedeți rezultatul modificărilor, doar le compilați cu task-ul Gradle build/build și atât. Aplicația vi se reîncarcă automat (dacă nu sunt erori!).

În acest moment, aveți un proiect Gradle, la care s-a adăugat librăria adițională Kotlin/JVM, proiect în care ați actualizat fișierul build.gradle pentru a include ca și dependențe „Spring Boot” și „Spring Boot Starter Web”. Cea din urmă este necesară pentru construirea de aplicații web, folosind Spring MVC.