

Sumar

Lista de Figuri.....	1
Lista de Code Snippets	1
• Creare proiect Maven în IntelliJ IDEA.....	1
• Adăugarea dependențelor JUnit și rulare multi-thread (<i>failsafe</i>).....	3
• Rularea aplicației într-un proiect Maven	7

Lista de Figuri

Figure 1 Creare proiect Maven.....	2
Figure 2 Completare <i>Name</i> , <i>GroupId</i> și <i>ArtifactID</i> pentru proiectul Maven	2
Figure 3 Configurări proiect Maven	3
Figure 4. Alegerea opțiunii <i>Enable Auto Import</i> pentru proiectul Maven creat	3
Figure 5. Configurarea proiectului Maven (fișierul pom.xml) pentru utilizarea JUnit 5.x.....	6
Figure 6. Fereastra Maven - Comenzi, Plugins, Dependențe, Configurații de rulare.....	7
Figure 7. Fereastra de rulare a comenzilor Maven	7
Figure 8. Comenzile asociate plugin-ului de execuție a proiectului Maven.....	8
Figure 9. Crearea unei configurații de rulare pentru un proiect Maven	8

Lista de Code Snippets

Snippet 1. Dependențe și plugins pentru proiectul Maven	6
---	---

Tutorialul pentru crearea unui proiect Maven in IntelliJ IDEA poate conține anumiți pași care pot fi omiși.

- **Creare proiect Maven în IntelliJ IDEA**

1. în meniul **File** ---> **New** ---> **Project**;
2. se selectează din lista tipurilor de proiecte **Maven**;
3. se bifează **Create from archetype**;
4. se alege din lista de tipuri de proiecte **maven-archetype-quickstart** (vezi Figure 1), apoi **Next**;

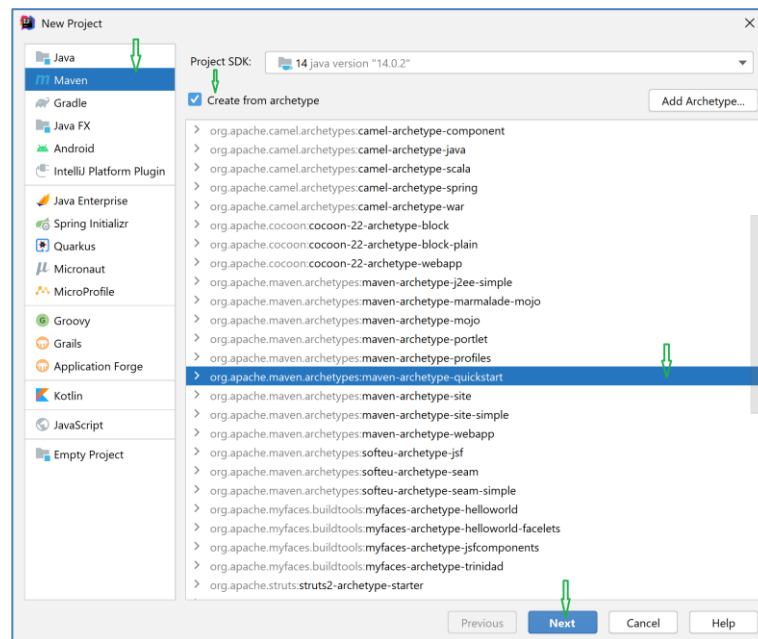


Figure 1 Creare proiect Maven

5. se completează numele proiectului **Name** (se va folosi id-ului userului de pe domeniul SCS) (vezi Figure 2);
 - e.g., userul cu adresa xyir1234@scs.ubbcluj.ro, va avea **Name xyir1234**;
6. se completează **Group Id**, **numele pachetului root** (se va folosi id-ului userului de pe domeniul SCS) (vezi Figure 2);
 - e.g., userul cu adresa xyir1234@scs.ubbcluj.ro, va avea **Group Id xyir1234MV**;
7. se completează **Artifact Id** (se va folosi id-ului userului de pe domeniul SCS) (vezi Figure 2);
 - e.g., userul cu adresa xyir1234@scs.ubbcluj.ro, va avea **Artifact Id xyir1234**;
8. apoi **Next**;

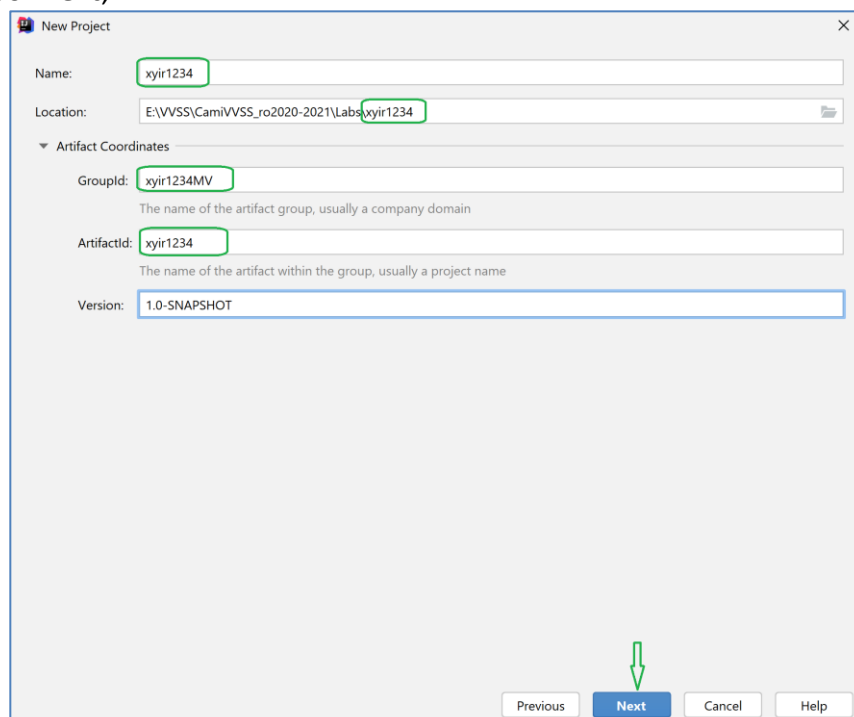


Figure 2 Completare Name, GroupID și ArtifactID pentru proiectul Maven

9. se poate configura repository-ul implicit și se pot vizualiza configurările realizate (vezi Figure 3), apoi **Finish** pentru a finaliza crearea proiectului Maven;

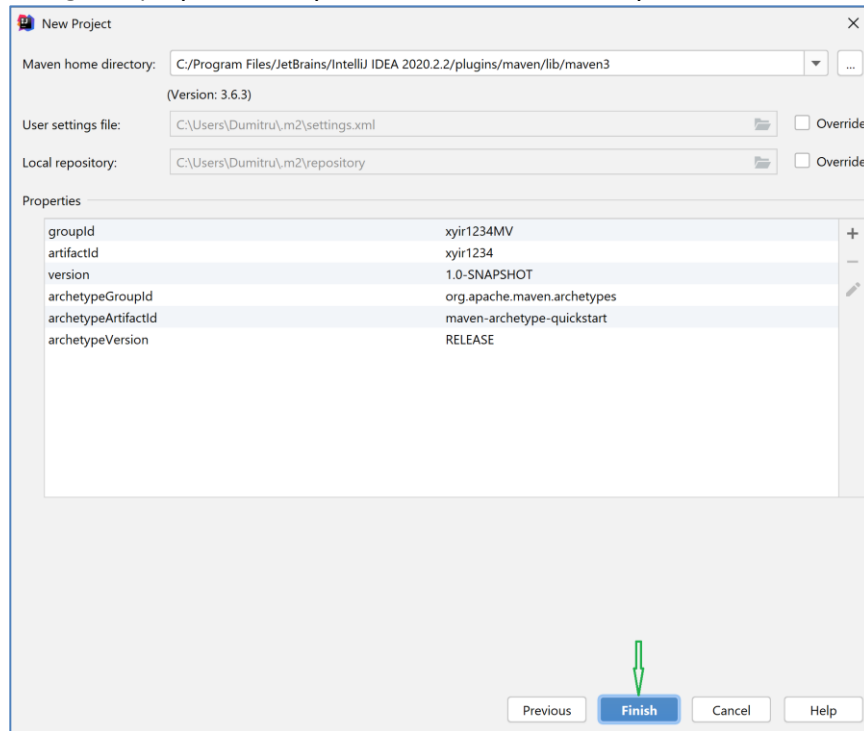


Figure 3 Configurări proiect Maven

10. Din fereastra pop-up care apare în colțul din dreapta jos se alege opțiunea **Enable Auto Import** (vezi Figure 4).

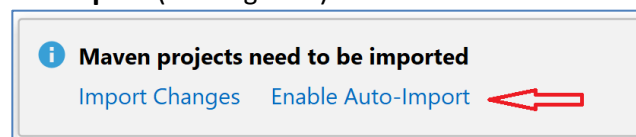


Figure 4. Alegerea opțiunii *Enable Auto Import* pentru proiectul Maven creat

- **Adăugarea dependențelor JUnit și rulare multi-thread (*failsafe*)**

Pentru proiectul utilizat în cadrul activităților de laborator sunt necesare includerea dependențelor pentru **JUnit 5.x (api, engine și params)** și **JavaFX**, cât și plugins pentru Maven surefire și Maven failsafe. Acestea se adaugă în fișierul **pom.xml**.

În funcție de proiectul Maven propriu-zis, pot fi incluse în fișierul **pom.xml** și **alte dependențe sau plugins**. De exemplu, pentru proiectul **Tasks**, fișierul **pom.xml** are conținutul din Snippet 1.

```
<?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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>tasks</groupId>
  <artifactId>Tasks</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>Tasks</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>
```

```

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <junit-plaform.version>5.6.0</junit-plaform.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.6.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
    <version>5.6.0</version>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-base</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-graphics</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-controls</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-fxml</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-media</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-swing</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-web</artifactId>
    <version>14-ea+4</version>
  </dependency>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
  </dependency>
</dependencies>

```

```

<groupId>org.controlsfx</groupId>
<artifactId>controlsfx</artifactId>
<version>11.0.1</version>
</dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <artifactId>maven-clean-plugin</artifactId>
      <version>3.1.0</version>
    </plugin>
    <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-
core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.1</version>
      <configuration>
        <!-- <failIfNoTests>false</failIfNoTests> -->
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-failsafe-plugin</artifactId>
      <version>2.22.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-install-plugin</artifactId>
      <version>2.5.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-deploy-plugin</artifactId>
      <version>2.8.2</version>
    </plugin>
    <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-
core/lifecycles.html#site_Lifecycle -->
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.7.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>3.0.0</version>
    </plugin>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>1.2.1</version>
      <executions>
        <execution>
          <goals>
            <goal>java</goal>
          </goals>
        </execution>
      </executions>
      <configuration>

```

```

        <mainClass>tasks.view.Main</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>

<pluginRepositories>
  <pluginRepository>
    <id>central</id>
    <name>Central Repository</name>
    <url>https://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
    <releases>
      <updatePolicy>never</updatePolicy>
    </releases>
  </pluginRepository>
</pluginRepositories>
<repositories>
  <repository>
    <id>central</id>
    <name>Central Repository</name>
    <url>https://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
</project>

```

Snippet 1. Dependențe și plugins pentru proiectul Maven

Figure 5 prezintă configurarea proiectului Maven pentru utilizarea JUnit 5.x, inclusă în fișierul **pom.xml** prezentat mai sus.



Figure 5. Configurarea proiectului Maven (fișierul pom.xml) pentru utilizarea JUnit 5.x

În general, pentru adăugarea în proiectul Maven a unei noi dependențe se urmează una dintre variantele de mai jos:

(1) se realizează click dreapta în fereastra fișierului **pom.xml** ---> **Generate...** ---> **Dependency Template** și se va genera o nouă dependență pentru care se va completa manual **groupId**, **artifactId**, **version**, etc;

(2) se realizează click dreapta în fereastra fișierului **pom.xml** ---> **Generate...** ---> **Dependency** și se va deschide fereastra care permite căutarea după **nume** și **versiune** a dependenței ce va fi inclusă în proiectul Maven.

• Rularea aplicației într-un proiect Maven

La nivelul proiectului Maven există mai multe comenzi utile, disponibile în meniul Maven (**View**---> **Tool Windows**---> **Maven**, vezi Figure 6). Rularea unei comenzi Maven se face alegând opțiunea **Execute Maven Goal** din meniul Maven (vezi Figure 7).

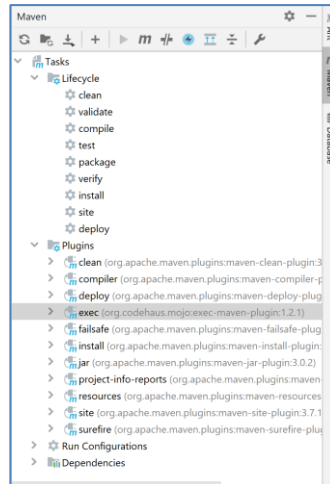


Figure 6. Fereastra Maven - Comenzi, Plugins, Dependențe, Configurații de rulare

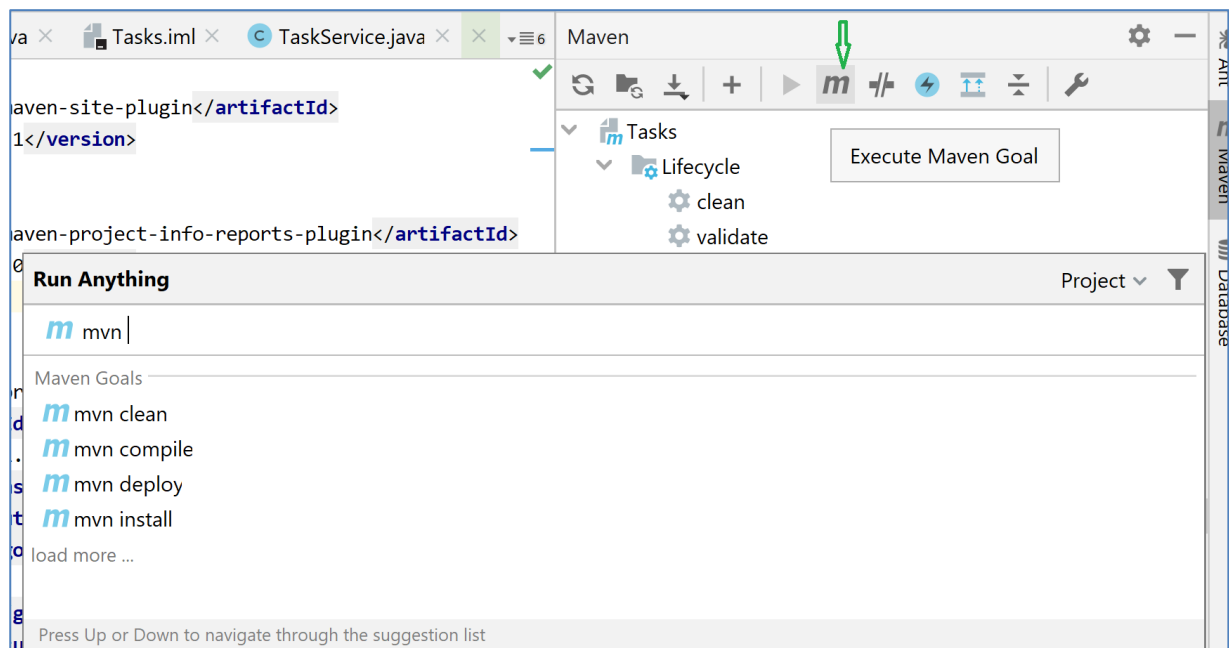


Figure 7. Fereastra de rulare a comenzilor Maven

Pentru rularea aplicației se creează o configurație de rulare, folosind plugin-ul de execuție a proiectului Maven, inclus în fișierul **pom.xml**:

- în fereastra de comenzi Maven, click dreapta pe comanda **exec:java** a plugin-ului **exec**;
- se alege opțiunea **Create "<PrjMaven> exec:java"** (vezi Figure 8);
- în fereastra de creare a configurării de rulare se poate modifica comanda existentă astfel încât să se realizeze compilarea înainte de execuția propriu-zisă (vezi Figure 9), apoi **OK**;
- configurația de rulare nou creată este adăugată în fereastra de comenzi Maven, secțiunea **Run Configurations**, de unde se și rulează (dublu-click pe configurația de rulare);

- numele clasei main asociată configurației de rulare este precizată în fișierul **pom.xml**, în descrierea plugin-ului de execuție, e.g., pentru proiectul **Tasks**, *main class*-ul este **tasks.view.Main** (vezi Snippet 1, descrierea plugin-ului **exec-maven-plugin**, elementul **<configuration>**).

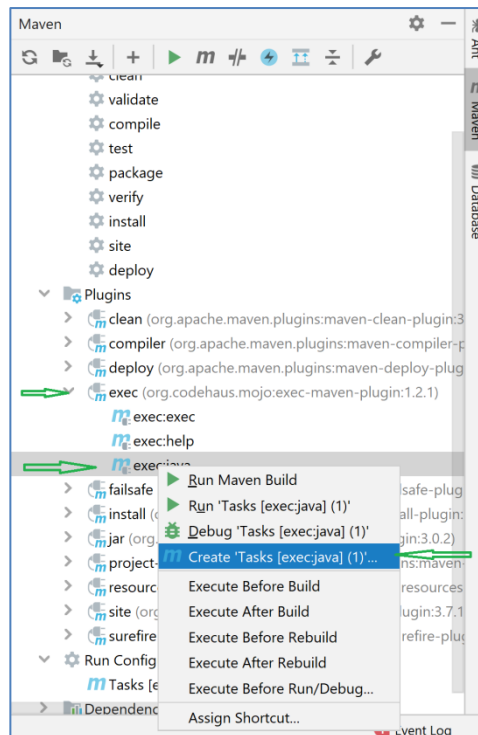


Figure 8. Comenzile asociate plugin-ului de execuție a proiectului Maven

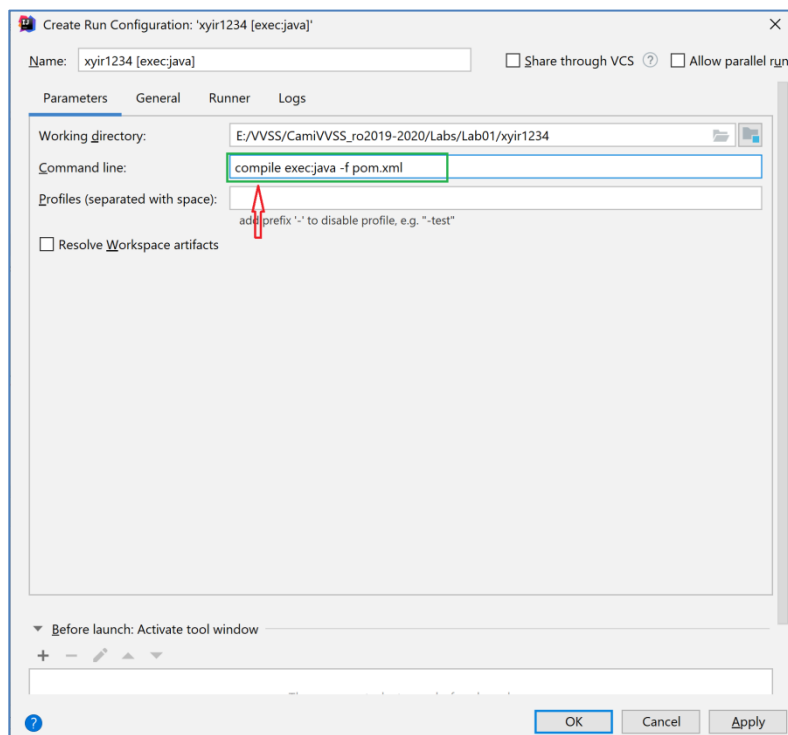


Figure 9. Crearea unei configurații de rulare pentru un proiect Maven