Sumar

Lista de Figuri

Lista de Code Snippets

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. din lista tipurilor de proiecte (***Generators***) se selectează **Maven Archetype**;
3. se completează câmpul **Name** cu numele proiectului; se poate folosi id-ul userului din domeniul SCS (vezi Figure 1):
    - e.g., userul cu adresa xyir1234@scs.ubbcluj.ro, va avea ca ***Name*** **xyir1234**;
4. se completează câmpul **Location** cu numele directorului în care se va salva proiectul;
5. se bifează opțiunea **Create Git repository**;
6. se selectează din lista **Catalog** sursa tipului de proiect Maven: *Maven Central*;
7. se completează în câmpul **Archetype** tipul de proiect Maven (vezi Figure 1):
    - **org.apache.maven.archetypes:maven-archetype-quickstart**
8. *Dacă tipul de proiect maven-archetype-quickstart nu este disponibil, se va adauga în lista folosind opțiunea* ***Add…*** (vezi Figure 2):
    - *Groupid:* **org.apache.maven.archetypes**
    - *Artifactid:* **maven-archetype-quickstart**
    - *Version:* **1.5**, apoi **Add;**
9. în secțiunea **Advanced Settings** se completează:
    - numele pachetului root **GroupId**: *tasks*
    - numele proiectului **Artifact**: *xyir1234*
10. se finalizează crearea proiectului prin **Create.**

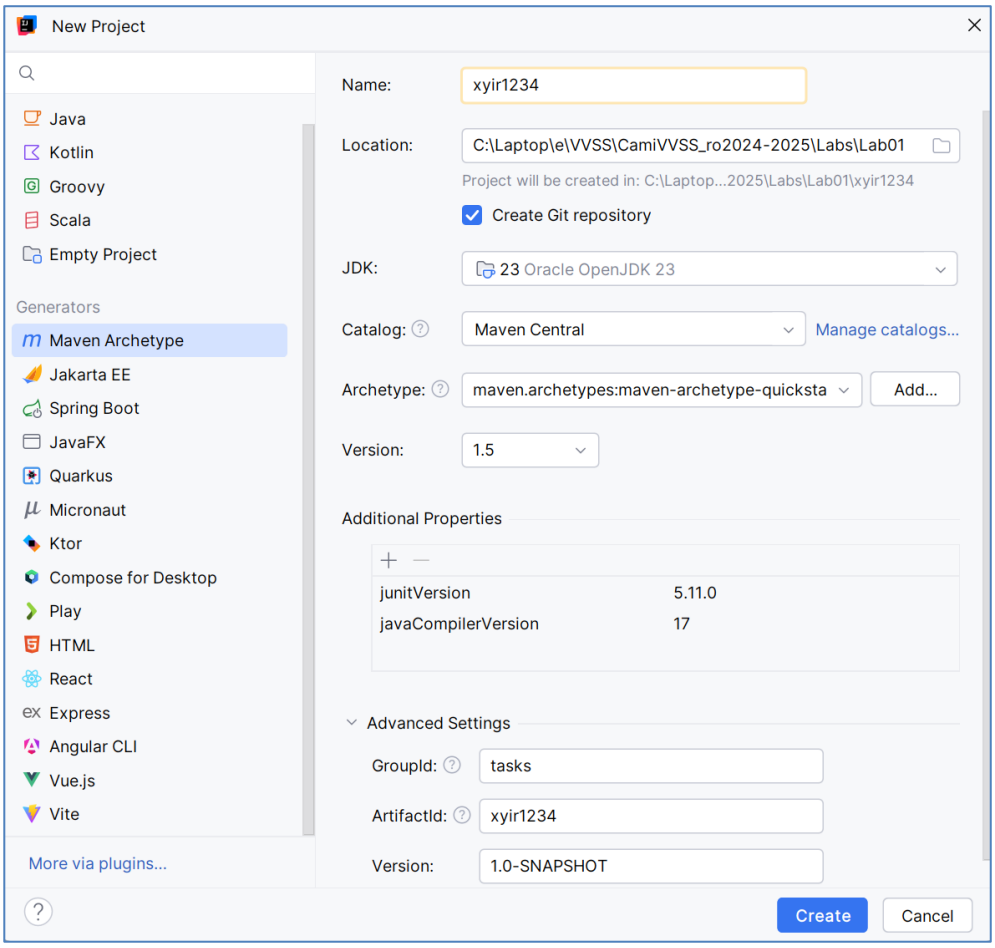

Figure 1. Creare proiect Maven

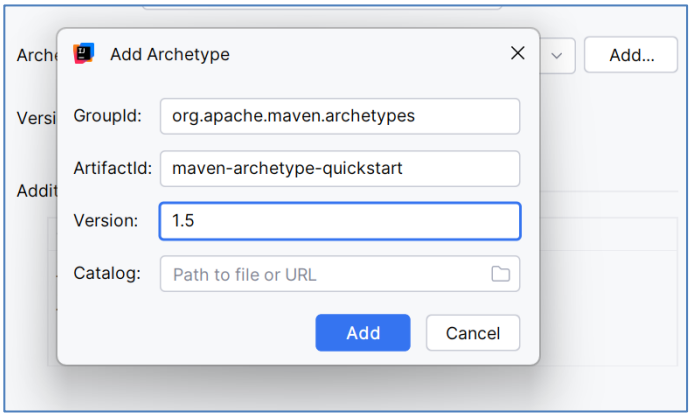

Figure 2. Adăugare Archetype Maven

- ## Adăugarea dependenţelor

Pentru proiectul utilizat în cadrul activităţilor de laborator sunt necesare setări pentru compilatorul Java[1], includerea unor dependenţe pentru JavaFX și log4j (log4j doar pentru unele proiecte), cât și un plugin pentru Maven failsafe. Acestea se adaugă în fișierul ***pom.xml***. De exemplu, pentru proiectul **Tasks**, fişierul ***pom.xml*** are conţinutul din Snippet 1.

[1] https://maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html

```xml
<?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>xzir1234</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>xzir1234</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.release>17</maven.compiler.release>-->
    <maven.compiler.source>9</maven.compiler.source>
    <maven.compiler.target>9</maven.compiler.target>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.junit</groupId>
        <artifactId>junit-bom</artifactId>
        <version>5.11.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <scope>test</scope>
    </dependency>
    <!-- Optionally: parameterized tests support -->
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-params</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-base</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-graphics</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
```

```
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-fxml</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-media</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-swing</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-web</artifactId>
      <version>17.0.2</version>
    </dependency>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>
    <dependency>
      <groupId>org.controlsfx</groupId>
      <artifactId>controlsfx</artifactId>
      <version>11.1.1</version>
    </dependency>

  </dependencies>

  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven
defaults (may be moved to parent pom) -->
      <plugins>
        <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-
core/lifecycles.html#clean_Lifecycle -->
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.4.0</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>3.3.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.13.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>3.3.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.4.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>3.1.2</version>
```

```
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>3.1.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-
core/lifecycles.html#site_Lifecycle -->
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.12.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.6.1</version>
        </plugin>

        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-failsafe-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>


      </plugins>
    </pluginManagement>
    <plugins> <!-- se va alege fie proprietatea de la inceputul fisierului, fie
plugin-ul de mai jos-->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>9</source>
          <target>9</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Snippet 1. Dependenţe şi plugins pentru proiectul Maven

- ## Codul sursă aferent proiectului Maven

În proiectul Maven creat se suprascrie folder-ul `src/main` cu folder-ul `src/main` din arhiva proiectului atribuit, i.e, Tasks, Inventory sau Pizza. Se copiază în proiect si folder-ul aferent datelor, i.e., **data**. Apoi se rulează aplicația.