Project Dependencies & Tools

🧠 IntelliJ IDEA (JetBrains) | Visual Studio Code (Microsoft)

IntelliJ IDEA, and Visual Studio Code were the two Integrated Development Environments used by the developers in our team.

Instructions relevant to installation of IntelliJ IDEA can be accessed at : https://www.jetbrains.com/help/idea/installation-guide.html#standalone

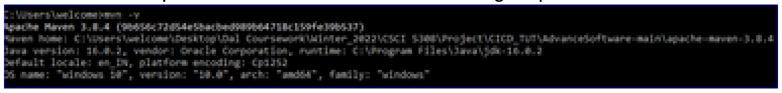
Installation and set-up guides for Visual Studio Code can be found at : Visual Studio Code Documentation

🛠️ Maven

1. Download the binary distribution archive for Maven from: https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/
2. Extract the distribution archive in a directory
3. Add the path of the “bin” directory in the extracted maven directory to the “PATH” environment variable in your system.
4. Verify that Maven is installed appropriately by executing the following command in a new shell window to check the version of Maven installed:

```
mvn -v
```

The resultant output should be similar to the following snip:



```
C:\Users\welcome>mvn -v
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: C:\Users\welcome\Desktop\Dal Coursework\Winter_2022\CSCI 5308\Project\CICD_TUT\AdvanceSoftware-main\apache-maven-3.8.4
Java version: 16.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-16.0.2
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

🦋 Maven Dependencies

Note:

The pair of group ID, artifact ID, and version is called the **Group, Artifact, and Version (GAV)** of a project and uniquely identifies the project:

- **Group**: The namespace of the organization that owns the projects
- **Artifact**: The name of the specific project
- **Version**: The version of the project

> *`spring-boot-starter-parent`: The spring-boot-starter-parent project is a special starter project – that provides default configurations for our application and a complete dependency tree to quickly build our Spring Boot project. Beyond that, it also inherits dependency management from spring-boot-dependencies which is the parent to the spring-boot-starter-parent.*

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.6.3</version>
  <relativePath/>
</parent>
```

> *`mysql-connector-java`: To connect to a MySQL database server from a Java app, we need a JDBC driver. The MySQL Connector/J product is one such JDBC driver. A JDBC driver is a set of Java classes that implement the JDBC interfaces, targeting a specific database. The MySQL driver is used in Java application to MySQL database using JDBC API.*

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

> *`springboot-framework-parent`: Springboot-framework-parent project for Spring Boot.*

```
<dependency>
        <groupId>com.codingapi.springboot</groupId>
        <artifactId>springboot-framework-parent</artifactId>
        <version>0.0.14</version>
        <type>pom</type>
</dependency>
```

> *`spring-boot-starter-thymeleaf`: Thymeleaf is a Java template engine for processing and creating HTML, XML, JavaScript, CSS and text. In web applications Thymeleaf aims to be a complete substitute for JavaServer Pages (JSP), and implements the concept of Natural Templates: template files that can be directly opened in browsers and that still display correctly as web pages. Thymeleaf is open-source software, licensed under the Apache License 2.0.*

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

> *`spring-boot-devtools`: Spring Boot DevTools provides the following features:*

1. Property Defaults
2. Automatic Restart
3. LiveReload
4. Remote Debug Tunneling
5. Remote Update and Restart

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
</dependency>
```

> *`spring-boot-starter-test`: The spring-boot-starter-test is the primary dependency for the test. It is a “starter module” for testing Spring Boot applications with libraries including JUnit Jupiter, Hamcrest and Mockito. It is useful to be able to perform integration testing without requiring deployment of the application or needing to connect to other infrastructure.*

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

> *`spring-boot-starter-data-jpa`: The Spring Java Persistent API (JPA) specification and Spring Data JPA unify and provide accessibility to the different kinds of persistence stores, both relational database systems, and NoSQL data stores. Spring Data JPA adds a layer on the top of JPA. Spring Data JPA uses all features defined by JPA specification, including the entity, association mappings, and JPA's query capabilities.*

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

> *`junit-jupiter`: Dependency for a unit testing framework in Java. The platform is responsible for launching testing frameworks on the JVM. The JUnit platform easily integrates clients with JUnit to discover and execute tests. It also defines the TestEngine API for developing a testing framework that runs on the JUnit platform.*

```
<dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.8.0</version>
        <scope>test</scope>
</dependency>
```

> *`mockito-all`: The main purpose of using the Mockito framework is to simplify the development of a test by mocking external dependencies and use them in the test code. As a result, it provides a simpler test code that is easier to read, understand, and modify. This dependency bundles Mockito as well as its required dependencies.*

mockito-all

|-- org

| |-- hamcrest

| |-- mockito

| |-- objenesis

```
<dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-all</artifactId>
        <version>1.10.19</version>
        <scope>test</scope>
</dependency>
```

> `junit`: *This is the dependency artifact for the unit testing framework utilized in our application for writing repeatable tests.*

```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>RELEASE</version>
        <scope>test</scope>
</dependency>
```

> `junit-jupiter-engine`: *JUnit Jupiter is the combination of the programming model and extension model for writing tests and extensions in JUnit 5. The Jupiter sub-project provides a Test Engine for running Jupiter based tests on the platform.*

```
<dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.5.2</version>
</dependency>
```

Maven Plugins

> `maven-surefire-plugin`: *The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application. It generates reports in two different file formats: Plain text files (*.txt) & XML files (*.xml).*

By default, these files are generated in **${base_directory}/target/surefire-reports/TEST-*.xml**.

```
<plugin>
                                <groupId>org.springframework.boot<
/groupId>
                                <artifactId>spring-boot-maven-plugin<
/artifactId>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins<
/groupId>
                                <artifactId>maven-surefire-plugin<
/artifactId>
                                <version>2.22.2</version>
                                <configuration>
                                        <testFailureIgnore>true<
/testFailureIgnore>
                                </configuration>
                        </plugin>
```