

# Tecnologie informatiche per il Web

## IntelliJ Guide

### Contents

<b>1 Preliminaries</b>	<b>2</b>
<b>2 IntelliJ Idea configuration</b>	<b>3</b>
2.1 Database configuration . . . . .	5
2.2 Configure MariaDB connection . . . . .	7
2.3 Convert Eclipse projects . . . . .	8
<b>3 Datagrip configuration</b>	<b>11</b>

# 1 Preliminaries

The tech stack is as follows:

- JetBrains IntelliJ Idea Ultimate as Java IDE and Maven as build system
  - Optionally Datagrip as SQL IDE
- MariaDB, a compatible fork of MySQL
- Tomcat as application server

## OS compatibility

Although the guide has been verified on Arch Linux, since all of JetBrains' IDEs are cross-platform, apart from the installation process, commands and paths the configuration will be the same on *every* operating system.

Programs and dependencies needed – commands for Arch Linux and derivatives:

```
# from official repositories
```

```
sudo pacman -S jdk21-openjdk mariadb tomcat10 maven
```

```
# from AUR
```

```
yay -S intellij-idea-ultimate-edition mariadb-jdbc
```

To install yay check the [official repository](#).

All of Datagrip functionalities are integrated in every JetBrains' IDE, so its not stricly needed – however, if you want install Datagrip as a standalone application:

```
# from AUR
```

```
yay -S datagrip datagrip-jre
```

## 2 IntelliJ Idea configuration

1. Create a new project with Jakarta EE as generator:
  - Template: web application
  - JDK: OpenJDK 21 (path: /usr/lib/jvm/java-21-openjdk)
  - Build system: Maven<sup>1</sup>

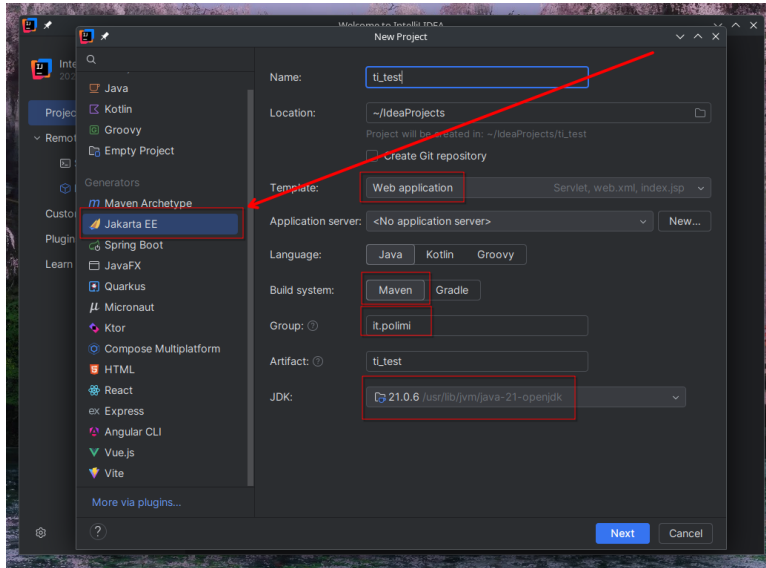


Figure 1: IntelliJ project configuration.

- Application server: Tomcat (path: /usr/share/tomcat10)

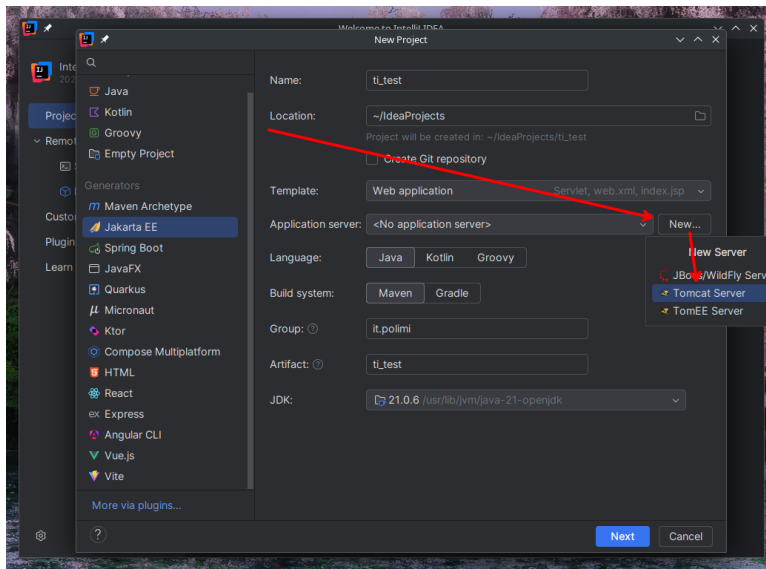


Figure 2: Tomcat configuration (1/2).

---

<sup>1</sup>In accordance with the Software Engineering course.

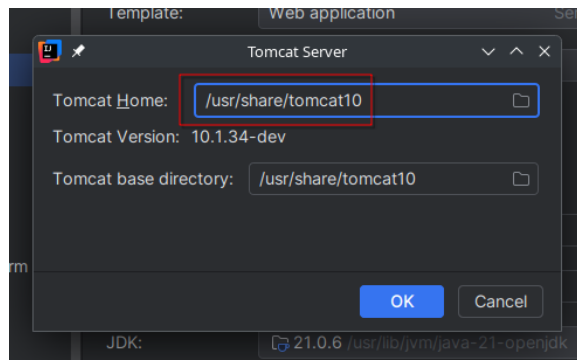
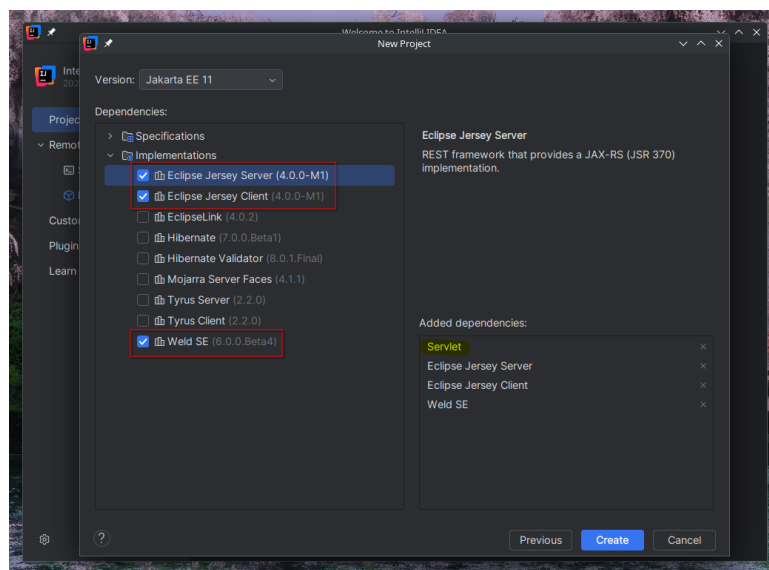


Figure 3: Tomcat configuration (2/2).

## 2. Check Eclipse server and client, Weld as implementations



Note that Servlet is already added as dependency.

### Permissions error

After a test, IntelliJ could report an error stating it cannot copy `/usr/share/tomcat10/conf` – this maybe caused by permissions:

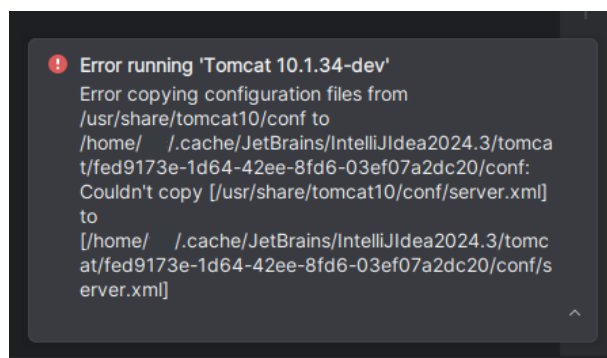


Figure 4: IntelliJ error.

to fix it run:

```
sudo chmod -R 777 /usr/share/tomcat10/conf
```

## 2.1 Database configuration

### 1. Configure MariaDB

```
mariadb-install-db --user=mysql --basedir=/usr --datadir=/var/lib/mysql  
mariadb-secure-installation
```

and then start it:

```
sudo systemctl start mariadb
```

If you want to start the database server at every boot type:

```
sudo systemctl enable mariadb
```

### 2. Create the user and grant *all* permissions on *all* databases:

```
sudo mariadb  
MariaDB [(none)]> CREATE USER 'name'@'localhost' IDENTIFIED BY 'password';  
MariaDB [(none)]> GRANT PRIVILEGES ON *.* TO 'name'@'localhost';  
MariaDB [(none)]> quit;
```

this is needed since in order to create a database *you need permission* to do so. If you want to verify:

```
MariaDB [(none)]> SHOW ALL PRIVILEGES FOR 'name'@'localhost';
```

### 3. Open the database configuration from IntelliJ (above right)



Figure 5: Database configuration in IntelliJ.

### 4. To import a MySQL dump execute the following command:

```
mariadb --user name --password < dump.sql
```

where *name* and *password* reference step 2.

5. Add the data source from **Figure 5**:

- Select MariaDB

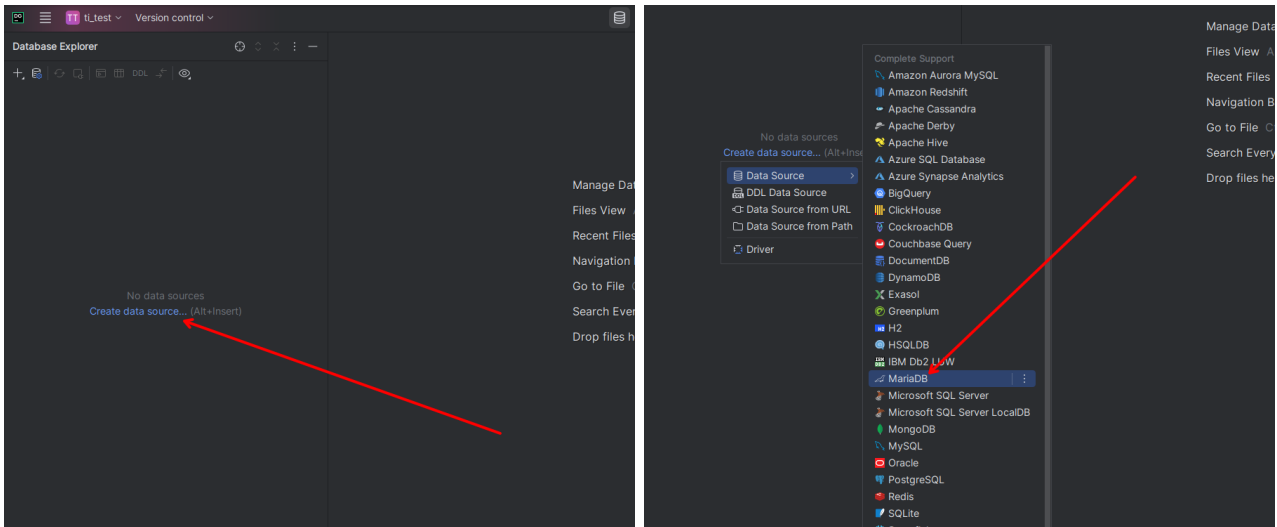


Figure 6: Selecting MariaDB as source.

- user, password from step 2
- Name of the database from step 5 – to check available databases:

```
sudo mariadb
MariaDB [(none)]> SHOW DATABASES;
MariaDB [(none)]> quit;
```

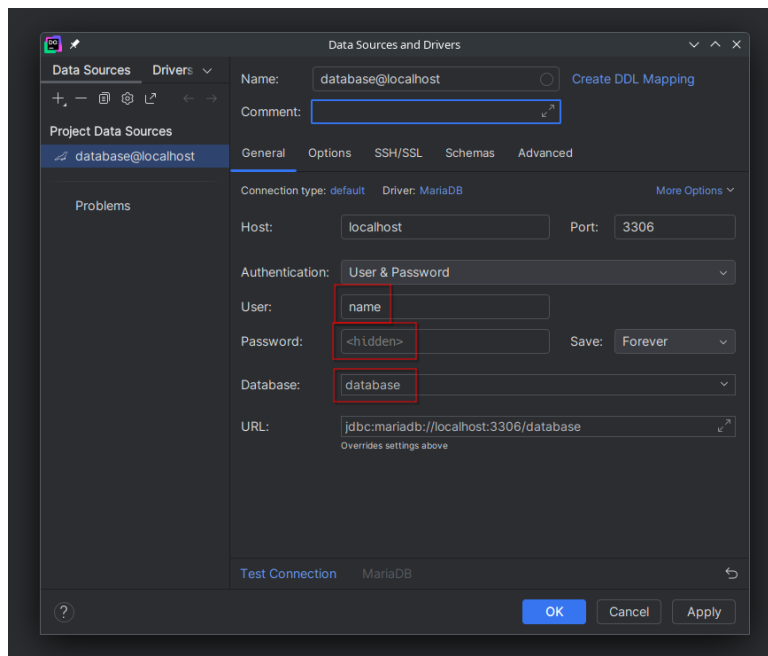


Figure 7: Adding the database.

Repeat step 4 and 5 for each dump.

## 2.2 Configure MariaDB connection

Add the following to pom.xml:

```
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>3.4.1</version>
</dependency>
```

and synchronize Maven, which then downloads all the necessary files. Last but not least, verify the connection by creating the `ConnectionTester` class:

```
import java.sql.*;

public class ConnectionTester {
    public static void main(String[] args) throws SQLException,
        ClassNotFoundException {
        final String DATABASE = "database";
        final String USER = "name";
        final String PASSWORD = "password";
        Connection connection = null;

        // Load the JDBC driver
        try {
            Class.forName("org.mariadb.jdbc.Driver");
            System.out.println("Driver loaded");
        } catch (ClassNotFoundException e) {
            System.err.println("Driver not found");
            e.printStackTrace();
        }
        try {
            connection = DriverManager.getConnection
                ("jdbc:mariadb://localhost:3306/" + DATABASE, USER, PASSWORD);
            System.out.println("Database connection successful");
            connection.close();
        } catch (Exception e) {
            System.err.println("Connection failed");
            e.printStackTrace();
        }
    }
}
```

by editing DATABASE, USER and PASSWORD accordingly.

## 2.3 Convert Eclipse projects

Briefly the steps are:

1. Figure out the dependencies and their versions
  - This applies both to libraries (such as thymeleaf) and Java JDK
2. Paste the /src directory from the original project ZIP
3. Add Tomcat run configuration

Start by adding the dependencies to the pom.xml:

```
<dependencies>
  <!-- Jakarta EE servlet (jsp, jstl) -->
  <dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>2.0.0</version>
  </dependency>
  <dependency>
    <groupId>jakarta.servlet.jsp.jstl</groupId>
    <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
    <version>2.0.0</version>
  </dependency>
  <!-- Thymeleaf -->
  <dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
    <version>3.1.3.RELEASE</version>
  </dependency>
  <!-- Apache Commons -->
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.17.0</version>
  </dependency>
</dependencies>
```

### Pick the correct versions

**Be sure** to check for the correct versions. To do so, open the Eclipse's file .classpath and search in it. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
  <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug
    .ui.launcher.StandardVMType/JavaSE-21">
    <attributes>
      <attribute name="module" value="true"/>
    </attributes>
  </classpathentry>
  <classpathentry kind="src" path="src/main/java"/>
  <classpathentry kind="con" path="org.eclipse.jst.j2ee.internal.web.container"/>
  <classpathentry kind="con" path="org.eclipse.jst.j2ee.internal.module.container"/>
  <classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/apache-commons-lang.jar"/>
  <classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/mysql-connector-j-8.0.32.jar"/>
  <classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/attoparser-2.0.7.RELEASE.jar"/>
  <classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/javassist-3.29.0-GA.jar"/>
  <classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/ognl-3.3.4.jar"/>
```



```

<classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/slf4j-api-2.0.16.jar"/>
<classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/thymeleaf-3.1.3.RELEASE.jar"/>
<classpathentry kind="lib" path="src/main/webapp/WEB-INF/lib/unescape-1.1.6.RELEASE.jar"/>
<classpathentry kind="con" path="org.eclipse.jst.server.core.container/org.eclipse.jst.server.tomcat
.runtimeTarget/apache Tomcat v10.1">
  <attributes>
    <attribute name="owner.project.facets" value="jst.web"/>
  </attributes>
</classpathentry>
<classpathentry kind="output" path="build/classes"/>
</classpath>

```

If you look closely, you'll see that the dependencies of this project are:

- Java 21
- mysql-connector-j-8.0.3
- attoparser-2.0.7.RELEASE
- javassist-3.29.0-GA
- ognl-3.3.4
- slf4j-api-2.0.1
- thymeleaf 3.1.3.RELEASE
- unescape-1.1.6.RELEASE
- Tomcat v10.1

And to set the JDK version:

```

<properties>
  <!-- Properties set for Java 21 -->
  <maven.compiler.source>21</maven.compiler.source>
  <maven.compiler.target>21</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

```

Search them on the Maven repository with the following URL scheme:

<https://mvnrepository.com/artifact/groupId/artifactId>.

To build the application and deploy it you also need:

```

<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.13.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.2.3</version>
    </plugin>
  </plugins>
</build>

```

Finally paste the /src directory from the desired project and configure Tomcat. An example can be seen in the tomcat\_config.xml file, which will have to be pasted into .idea/run-Configurations folder.

### Optimize the conversion

Since these steps are the same for every project, I'd suggest to create a **template** folder which houses the **pom.xml** file along with the Tomcat configuration. This way, to convert a project you will only have to create a new directory and then paste in it the **src/** folder and the contents of **template**.

### Follow the standard structure

Finally, the directory tree will have to look like:

```
src/
|-- main/
|   |-- java/
|   |   |-- it.polimi.tiw
|   |-- resources
|   |-- webapp
|-- test/
|   |-- java/
|   |   |-- it.polimi.tiw
|   |-- resources
|   |-- webapp
LICENSE.txt
README.md
pom.xml
```

In accordance with the [Maven standard directory layout](#).

This is **not optional**: for instance, in some projects there's a **resources** folder which is NOT located in the correct path; once the project will be deployed, Java will look for the **src/main/resources** folder and will not find it. IntelliJ won't throw an error.

### 3 Datagrip configuration

1. Follow step 1, 2 from [subsection 2.1](#)
2. Create a new project in Datagrip

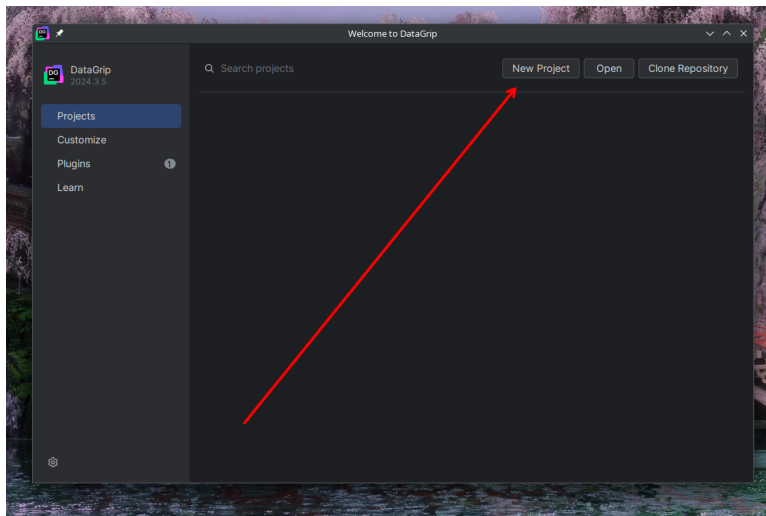


Figure 8: Creating a new project in Datagrip.

3. Follow the remaining steps from [subsection 2.1](#)