Tecnologie informatiche per il Web
IntelliJ Guide

Contents

1 Preliminaries

The tech stack is as follows:

- IntelliJ Idea Ultimate as Java IDE and Maven as build system
  - Datagrip as SQL IDE
- MariaDB, a retrocompatible 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:

```
# from ufficial 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.

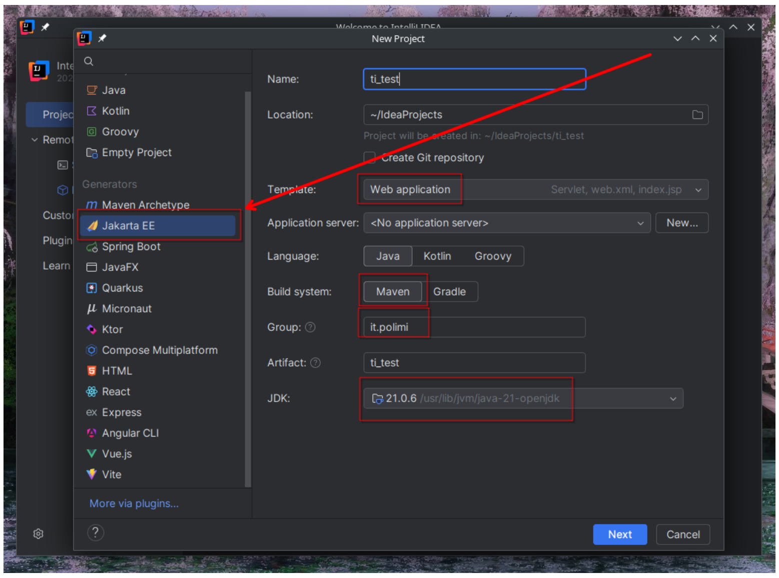
All of Datagrip functionalities are integrated in every JetBrains' IDE, so its not stricly needed – however, if you want install Datagrip as a standalone application:

```
# from AUR
yay -S datagrip datagrip-jre
```

2 IntelliJ Idea configuration

1. Create a new project with `Jakarta EE` as generator:

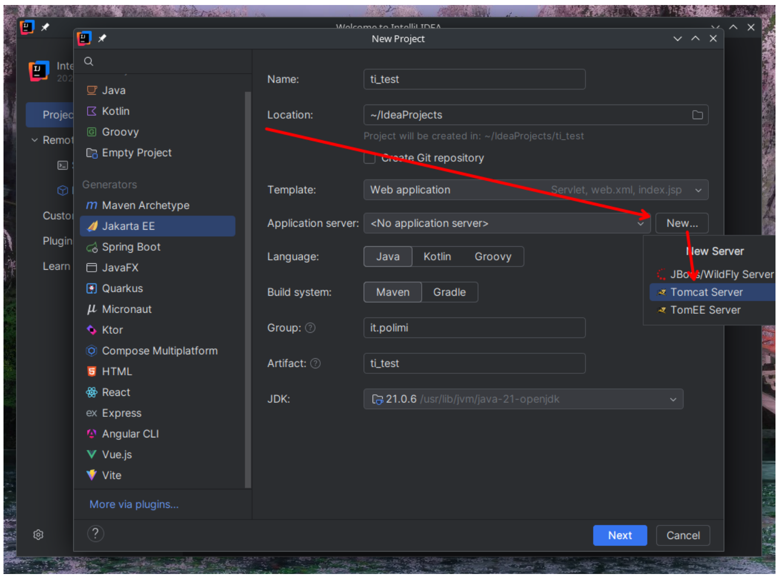
   - Template: web application
   - JDK: OpenJDK 21 (path: `/usr/lib/jvm/java-21-openjdk`)
   - Build system: Maven[1]



Name: ti_test
Location: ~/IdeaProjects
Project will be created in: ~/IdeaProjects/ti_test
Create Git repository
Template: Web application — Servlet, web.xml, index.jsp
Application server: <No application server> New...
Language: Java | Kotlin | Groovy
Build system: Maven | Gradle
Group: it.polimi
Artifact: ti_test
JDK: 21.0.6 /usr/lib/jvm/java-21-openjdk

Figure 1: IntelliJ project configuration.

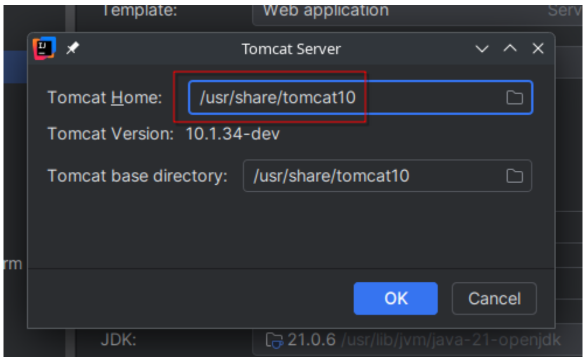
   - Application server: Tomcat (path: `/usr/share/tomcat10`)



Name: ti_test
Location: ~/IdeaProjects
Project will be created in: ~/IdeaProjects/ti_test
Create Git repository
Template: Web application — Servlet, web.xml, index.jsp
Application server: <No application server> New...
Language: Java | Kotlin | Groovy
Build system: Maven | Gradle
Group: it.polimi
Artifact: ti_test
JDK: 21.0.6 /usr/lib/jvm/java-21-openjdk

Figure 2: Tomcat configuration (1/2).

[1] In accordance with the Software Engineering course.



Template: Web application

Tomcat Server

Tomcat Home: /usr/share/tomcat10

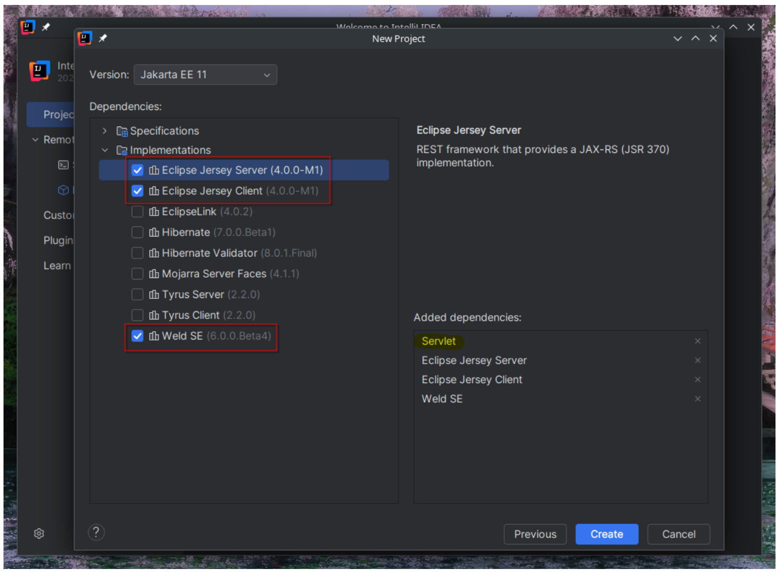
Tomcat Version: 10.1.34-dev

Tomcat base directory: /usr/share/tomcat10

JDK: 21.0.6 /usr/lib/jvm/java-21-openjdk

Figure 3: Tomcat configuration (2/2).

2. Check Eclipse server and client, Welde as implementations



New Project

Version: Jakarta EE 11

Dependencies:

- Specifications
- Implementations
  - [✓] Eclipse Jersey Server (4.0.0-M1)
  - [✓] Eclipse Jersey Client (4.0.0-M1)
  - [ ] EclipseLink (4.0.2)
  - [ ] Hibernate (7.0.0.Beta1)
  - [ ] Hibernate Validator (8.0.1.Final)
  - [ ] Mojarra Server Faces (4.1.1)
  - [ ] Tyrus Server (2.2.0)
  - [ ] Tyrus Client (2.2.0)
  - [✓] Weld SE (6.0.0.Beta4)

Eclipse Jersey Server

REST framework that provides a JAX-RS (JSR 370) implementation.

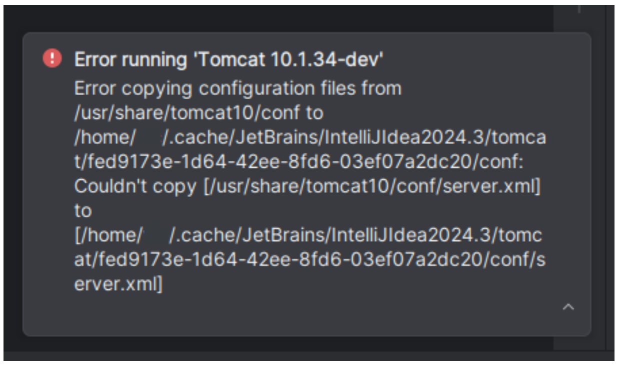
Added dependencies:

- Servlet
- Eclipse Jersey Server
- Eclipse Jersey Client
- Weld SE

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:



```
Error running 'Tomcat 10.1.34-dev'
Error copying configuration files from
/usr/share/tomcat10/conf to
/home/     /.cache/JetBrains/IntelliJIdea2024.3/tomca
t/fed9173e-1d64-42ee-8fd6-03ef07a2dc20/conf:
Couldn't copy [/usr/share/tomcat10/conf/server.xml]
to
[/home/     /.cache/JetBrains/IntelliJIdea2024.3/tomc
at/fed9173e-1d64-42ee-8fd6-03ef07a2dc20/conf/s
erver.xml]
```

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 check:

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

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

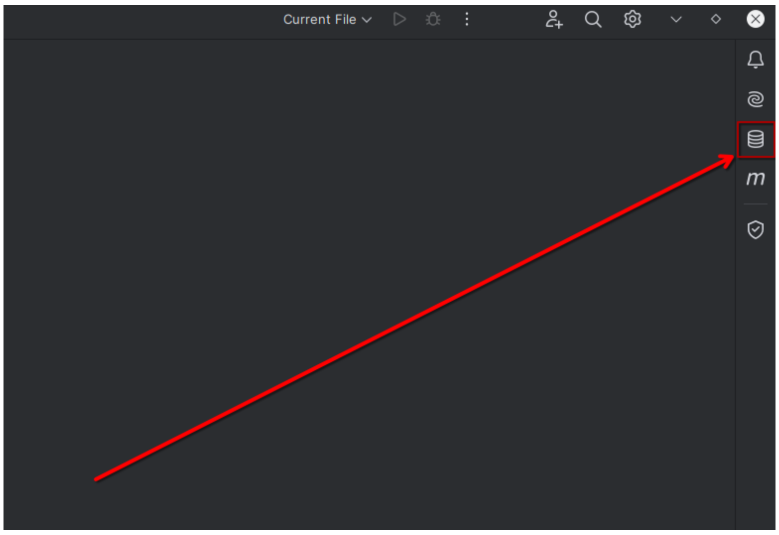


Figure 5: Database configuration in IntelliJ.

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

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

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

5. From the above left menu add the data source:
   - Select MariaDB

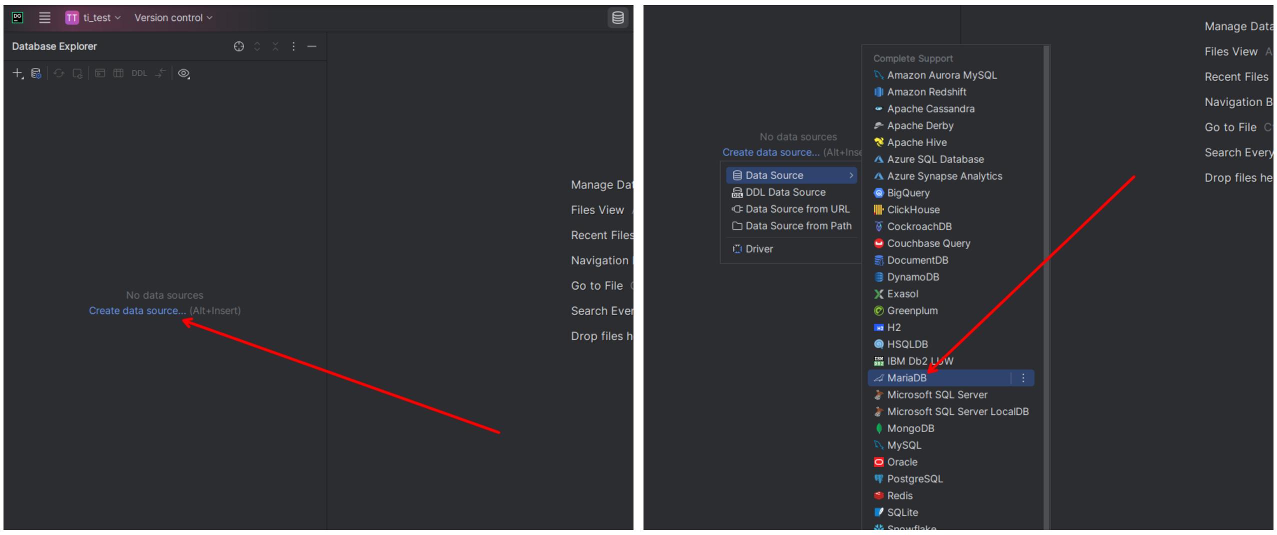
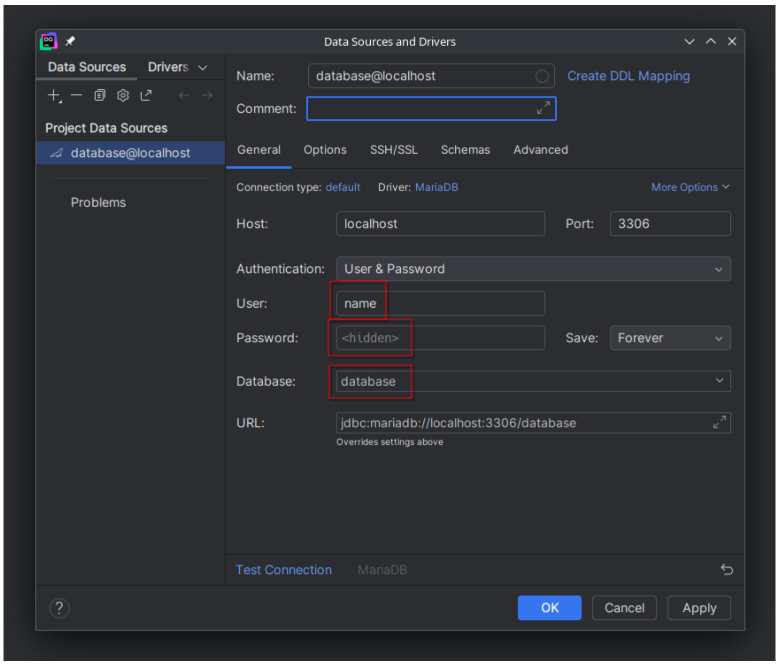


Figure 6: Selecting MariaDB as source.

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

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



Data Sources and Drivers

Name: database@localhost

Comment:

Connection type: default Driver: MariaDB

Host: localhost Port: 3306

Authentication: User & Password

User: name

Password: <hidden> Save: Forever

Database: database

URL: jdbc:mariadb://localhost:3306/database

Overrides settings above

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 adds all the necessary drivers. 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 Configure Maven dependencies

In order to make the current Eclipse projects to work, the pom.xml file needs some other dependencies:

```
<dependencies>
    <!-- Jakarta EE servlet (jsp, jstl) -->
    <dependency>
        <groupId>org.glassfish.web</groupId>
```

```xml
        <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>
</dependencies>
```

Be sure to use the correct versions. Search them on the Maven repository with the following URL scheme: `https://mvnrepository.com/artifact/groupId/artifactId`.

Also, you might need to set the JDK version:

```xml
<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>
```

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

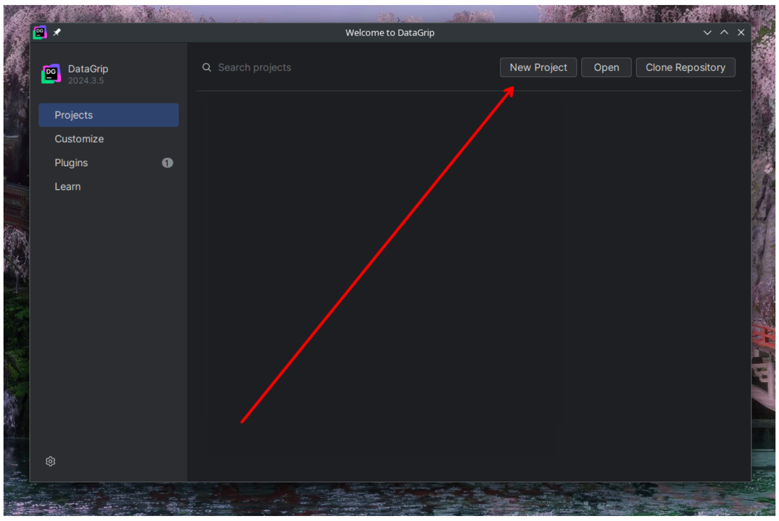


Figure 8: Creating a new project in Datagrip.

3. Follow the remaining steps from subsection 2.1