Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili

[DΣIM]

Multi-Agent Systems

Jordi Pascual – jordi.pascual@urv.cat

Maven, Git and OSM

MESIIA – Master's Degree in Computer Security Engineering and Artificial Intelligence
MAI - Master's Degree in Artificial Intelligence

Outline

1. Maven
2. Teamwork
3. OpenStreetMaps
4. More resources

1. Maven

- JAVA tool to automate
  - Managing of dependencies
  - Compiling the source code
  - Packaging the code into deployable artifacts
  - Executing artifacts

1. Maven: POM

- XML file
- Several sections to configure it
  - Repositories
  - Dependencies
  - Build
  - Profiles
  - …

```xml
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <!-- The Basics -->
  <groupId>...</groupId>
  <artifactId>...</artifactId>
  <version>...</version>
  <packaging>...</packaging>
  <dependencies>...</dependencies>
  <parent>...</parent>
  <dependencyManagement>...</dependencyManagement>
  <modules>...</modules>
  <properties>...</properties>

  <!-- Build Settings -->
  <build>...</build>
  <reporting>...</reporting>

  <!-- More Project Information -->
  <name>...</name>
  <description>...</description>
  <url>...</url>
  <inceptionYear>...</inceptionYear>
  <licenses>...</licenses>
  <organization>...</organization>
  <developers>...</developers>
  <contributors>...</contributors>

  <!-- Environment Settings -->
  <issueManagement>...</issueManagement>
  <ciManagement>...</ciManagement>
  <mailingLists>...</mailingLists>
  <scm>...</scm>
  <prerequisites>...</prerequisites>
  <repositories>...</repositories>
  <pluginRepositories>...</pluginRepositories>
  <distributionManagement>...</distributionManagement>
  <profiles>...</profiles>
</project>
```

1. Maven: POM Profiles

- Profiles are used to define build configurations
- Profile template:

```
<profiles>
        <profile>
                <id>identifier</id>    <!— Profile identifier —>
                <build><plugins><plugin>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>exec-maven-plugin</artifactId>
                        <version>3.1.0</version>
                        <configuration>
                                <mainClass>path.to.Class</mainClass>          <!—   Path to main class —>
                                <arguments>                                   <!— Program parameters —>
                                        <argument>parameter1</argument>
                                        <argument>parameter2</argument>
                                </arguments>
                        </configuration>
                </plugin></plugins></build>
        </profile>
</profiles>
```

- Main class arguments must be introduced in new *argument* lines. **Blank spaces are not allowed in the arguments**

1. Maven: Goals

- Maven has a build **lifecycle** consisting on several **phases**
- Each Maven **phase** represents a stage in the Maven build **lifecycle**
- Each **phase** is a sequence of **goals**, and each **goal** is responsible for a specific **task**
- Some examples:
  - **clean**: lifecycle that removes all previously generated build files
  - **install**: phase to install the package to a local repository
  - **exec:java**: goal to execute an application

1. Maven: Goals

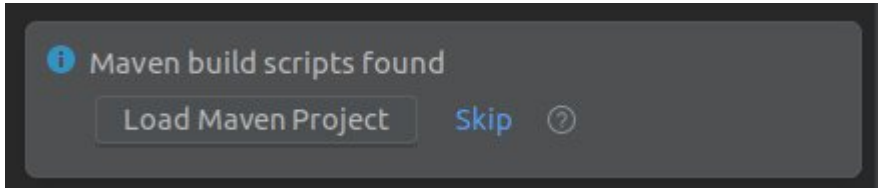
Useful Maven Goals:

- Clean: **mvn clean**
- Build: **mvn install**
- Run profile: **mvn -P profile-name exec:java**
- Build and run profile: **mvn install –P profile-name exec:java**

- **Every time** the source code is modified, the project must be build using **mvn install**

1. Maven: First steps

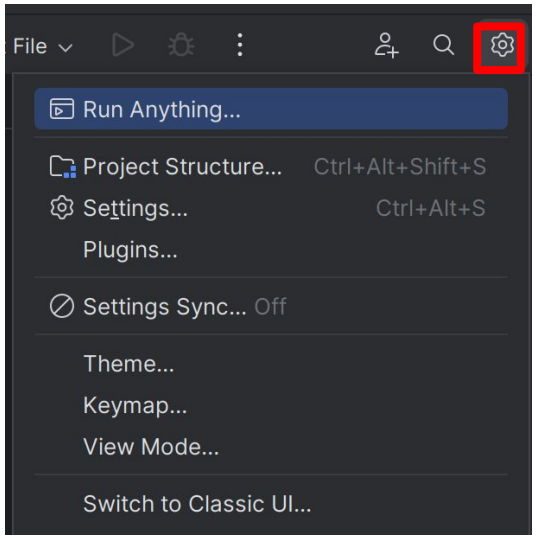
1. Download the *JadeExample.zip* file from the **URV Virtual Campus**
2. Extract the contents of the zip file
3. Open the project using IntelliJ. Set up the JDK if needed
4. When prompted, load it as a Maven project



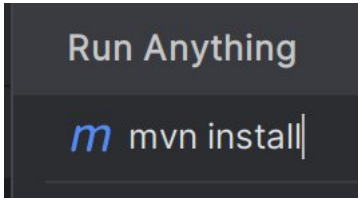
5. Take a look at the POM file

1. Maven: First steps

6. Open the **Run Anything** command line to execute Maven Goals. Double-tap the *Ctrl* key or



7. Execute **mvn install** to install the needed dependencies and build the project

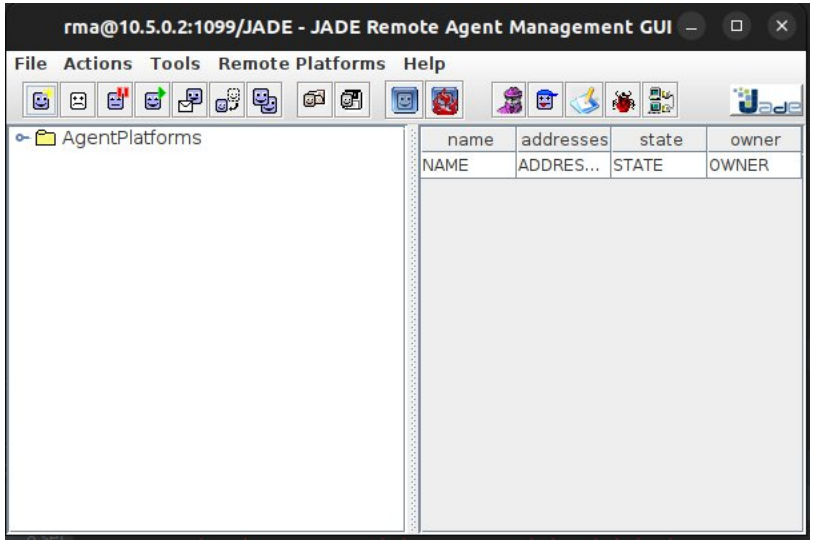


1. Maven: First steps

8. Execute the **jade-gui** profile using one of the two available methods. This profile starts the JADE GUI

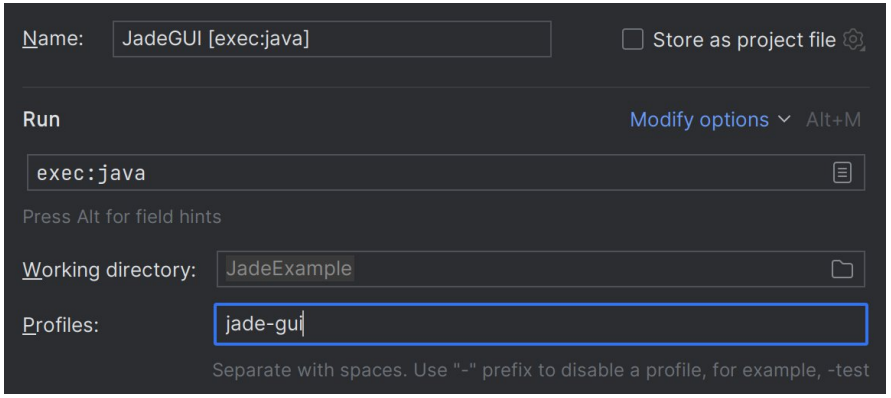
Maven Goals

9. Execute the command **mvn install -P jade-gui exec:java** in the Run Anything command line



Run Configuration

9. Run -> Edit configurations
10. Add new **Maven** configuration
11. Set it up as in the image below
12. Run -> Run...



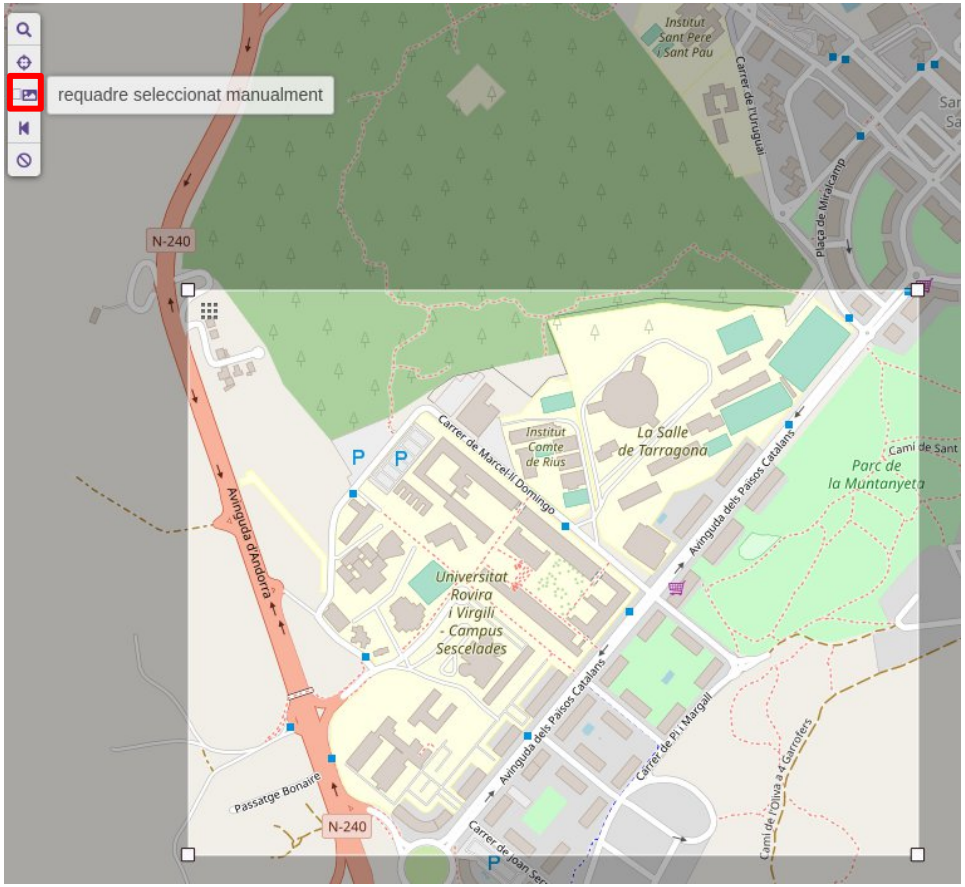
2. Teamwork

- To evaluate the teamwork in the practical work, you will have to use a **Git** based source control system to **document all changes** on the practical work code
- You can use:
  - GitLab: https://about.gitlab.com/
  - GitHub: https://github.com/
  - Bitbucket: https://bitbucket.org/
- The teacher must have **read-only** access to the repository
- Use the following email to add him to the Git repository: jordi.pascual@urv.cat

3. OpenStreetMaps (OSM)

We will use maps extracted from OSM in the practical work. To obtain them in the needed format, follow the next steps:

1. Go to https://overpass-turbo.eu/
2. Select a rather small area of any city
3. Press on **Assistant**
4. Write **highway=* and type:way** and Execute



3. OpenStreetMaps (OSM)

5. Click **Export**, and download it as a *GeoJSON*
6. Open the *geojson* file using **JOSM**. This program can be used to modify the map if needed
7. Save a copy of the map in *osm* format
8. Clone the *gs-geography* repository from **GitLab** https://gitlab.com/dedale/gs-geography
9. Open the *gs-geography* project on IntelliJ
10. Replace **http** for **https** in the following repositories (lines 236 and 240)

```
<repositories>
        <repository>
                <id>de.topobyte</id>
                <url>https://mvn.topobyte.de</url>
        </repository>
        <repository>
                <id>com.slimjars</id>
                <url>https://mvn.slimjars.com</url>
        </repository>
</repositories>
```

3. OpenStreetMaps (OSM)

11. Create a new Maven profile named **create-map**. Its main class must be *Test_OSM_RoadNetwork2*
12. Copy your *osm* map into the data folder of the *gs-geography* project
13. Replace the default map on the *Test_OSM_RoadNetwork2* file by your *osm* map

```
GeoSourceOSM_RoadNetwork.generateGsFromOSM(ls, System.getProperty("user.dir")+"/data/myMap.osm");
```

14. Build and execute the **create-map** profile using **mvn install -P create-map exec:java**
15. A new *dgs* map should be on the data folder. We will use the map later on with Dedale

4. More resources

- Maven in 5 minutes
- Maven introduction to the Build Lifecycle
- Maven in IntelliJ
- Git documentation
- Dedale gs-geography