

SPRING BOOT - MAVEN



UP ASI
Bureau E204

Plan du Cours

- Définition Spring Boot
- Avantages Spring Boot
- Définition Maven
- Etapes de construction de projet
- Arborescence du projet
- Gestion et portée des dépendances
- TP Spring Boot

SPRING BOOT

- **Spring Boot** est le projet principal du Spring Framework.
- Il simplifie le démarrage et le développement de nouvelles applications Spring.
- Il diminue énormément le temps de développement et augmente la productivité.
- Il est très facile d'intégrer des applications Spring Boot avec ses écosystème de Spring (projets spring) comme Spring MVC, Spring Data, Spring Security etc...
- Avec Spring Boot, les configurations de Spring sont diminuées.

SPRING BOOT

- Spring Boot soutient des conteneurs embarqués (embedded containers).
- Cela permet à des applications web de s'exécuter sans déploiement sur un Web Server.
- Il suit l'approche "Configuration par défaut" afin de diminuer le temps et l'effort de développement.
- Spring Boot favorise le travail avec les microservices.

AVANTAGES SPRING BOOT

- Spring Boot offre trois avantages incontournables :
 - La gestion des configurations
 - Le serveur est embarqué
 - La gestion des dépendances

AVANTAGES SPRING BOOT - Gestion des configurations

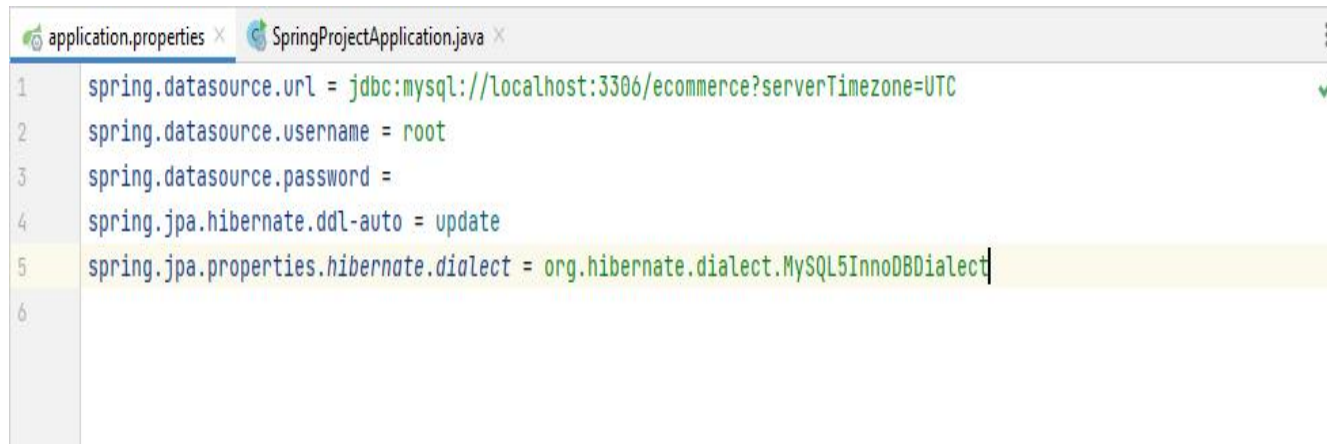
- Spring Boot facilite **la gestion des configurations** en centralisant les configurations dans un seul fichier. Ainsi en se focalisant sur le métier au lieu de la configuration, le développeur devient beaucoup plus productif.

Avant Spring Boot

web.xml
persistence.xml
dispatcher.xml
log4J.xml
.....

Avec Spring Boot

Un seul fichier application.properties

A screenshot of an IDE window showing two tabs: 'application.properties' and 'SpringProjectApplication.java'. The 'application.properties' tab is active and displays the following configuration lines:

```
1 spring.datasource.url = jdbc:mysql://localhost:3306/ecommerce?serverTimezone=UTC ✓
2 spring.datasource.username = root
3 spring.datasource.password =
4 spring.jpa.hibernate.ddl-auto = update
5 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect|
6
```

AVANTAGES SPRING BOOT - Serveur Embarquée

- **Spring Boot** fournit des serveurs intégrés (Embedded HTTP servers) comme Tomcat, Jetty afin de développer et de tester des applications web facilement.
- En lançant le projet, le jar du Tomcat dézippe et se lance.

The screenshot displays an IDE window titled 'springBootProject/pom.xml'. The main content area is divided into two panels: 'Dependency Hierarchy [test]' and 'Resolved Dependencies'. The 'Dependency Hierarchy' panel shows a tree structure of dependencies, with 'tomcat-embed-core : 9.0.38 (omitted for conflict with 9.0.3)' highlighted. The 'Resolved Dependencies' panel lists various dependencies, including 'mysql-connector-java : 8.0.21 [runtime]' and 'spring-boot-starter-web : 2.3.4.RELEASE [compile]'. A filter box at the top right of the 'Resolved Dependencies' panel contains the text 'tomcat|'. The bottom of the IDE window shows a tabbed interface with 'Overview', 'Dependencies', 'Dependency Hierarchy', 'Effective POM', and 'pom.xml' tabs, with 'pom.xml' currently selected.

springBootProject/pom.xml

Dependency Hierarchy [test] Filter: tomcat|

Dependency Hierarchy

- spring-boot-starter-web : 2.3.4.RELEASE [compile]
 - spring-boot-starter-tomcat : 2.3.4.RELEASE [compile]
 - tomcat-embed-core : 9.0.38 [compile]
 - tomcat-embed-websocket : 9.0.38 [compile]
 - tomcat-embed-core : 9.0.38 (omitted for conflict with 9.0.3)

Resolved Dependencies

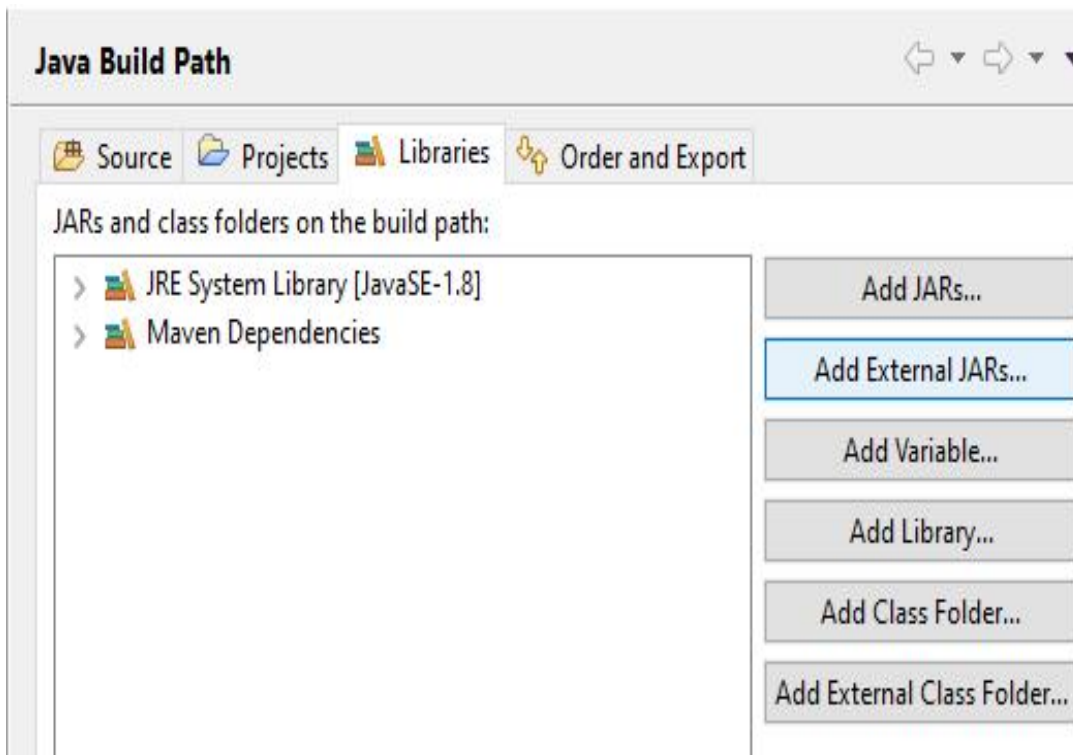
- mysql-connector-java : 8.0.21 [runtime]
- objenesis : 2.6 [test]
- opentest4j : 1.2.0 [test]
- slf4j-api : 1.7.30 [compile]
- snakeyaml : 1.26 [compile]
- spring-aop : 5.2.9.RELEASE [compile]
- spring-aspects : 5.2.9.RELEASE [compile]
- spring-beans : 5.2.9.RELEASE [compile]
- spring-boot : 2.3.4.RELEASE [compile]
- spring-boot-autoconfigure : 2.3.4.RELEASE [compile]
- spring-boot-starter : 2.3.4.RELEASE [compile]
- spring-boot-starter-aop : 2.3.4.RELEASE [compile]
- spring-boot-starter-data-jpa : 2.3.4.RELEASE [compile]
- spring-boot-starter-jdbc : 2.3.4.RELEASE [compile]
- spring-boot-starter-test : 2.3.4.RELEASE [compile]

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

AVANTAGES SPRING BOOT - Gestion des dépendances

- Spring Boot facilite **la gestion des dépendances** pour commencer un projet Spring.

Avant Spring et Spring Boot



Avec Spring Boot

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


AVANTAGES SPRING BOOT - Gestion des dépendances

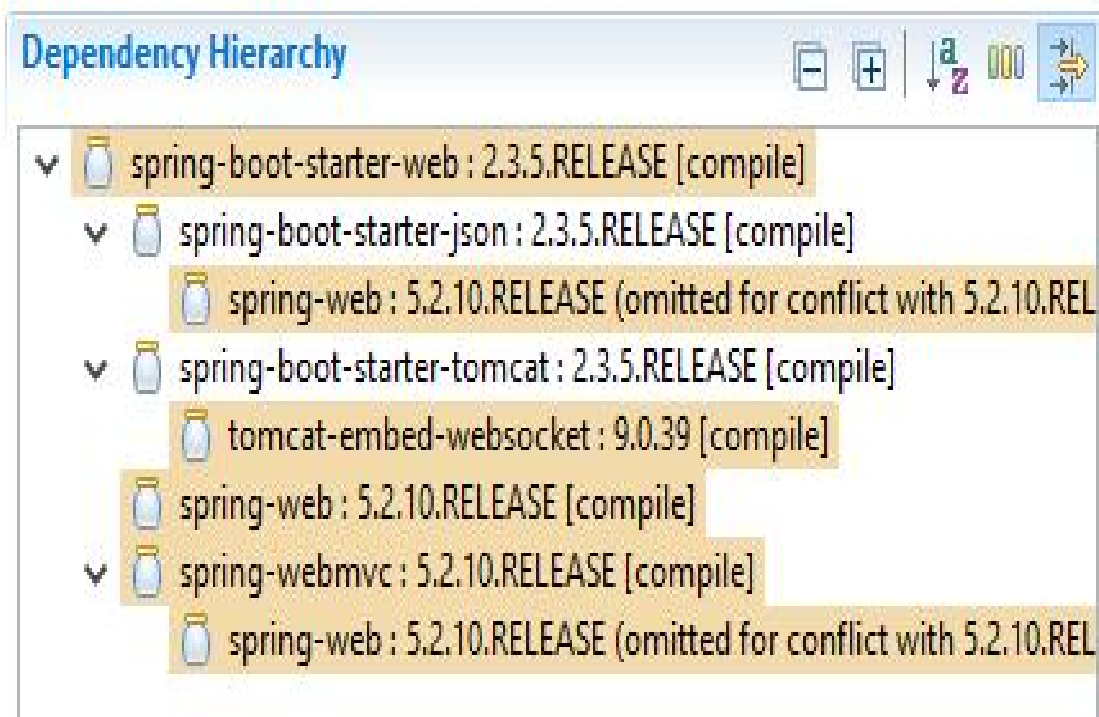
- Spring Boot facilite **la gestion des dépendances** grâce notamment à l'utilisation des **starters**.

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

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
```

AVANTAGES SPRING BOOT - Gestion des dépendances

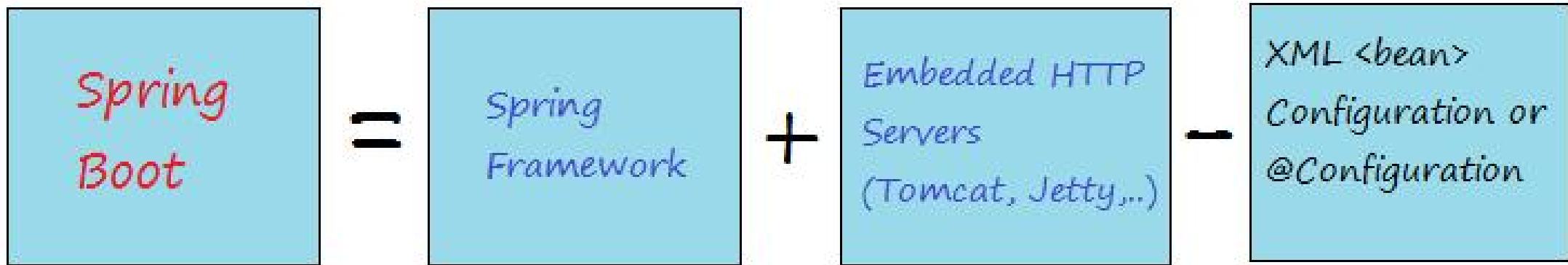
- Un **starter** va apporter à votre projet un ensemble de dépendances, communément utilisées pour un type de projet donné.
- Les starters facilitent la **gestion des versions**. Plus besoin de chercher quelles versions sont compatibles afin de les ajouter une à une dans le pom.xml.



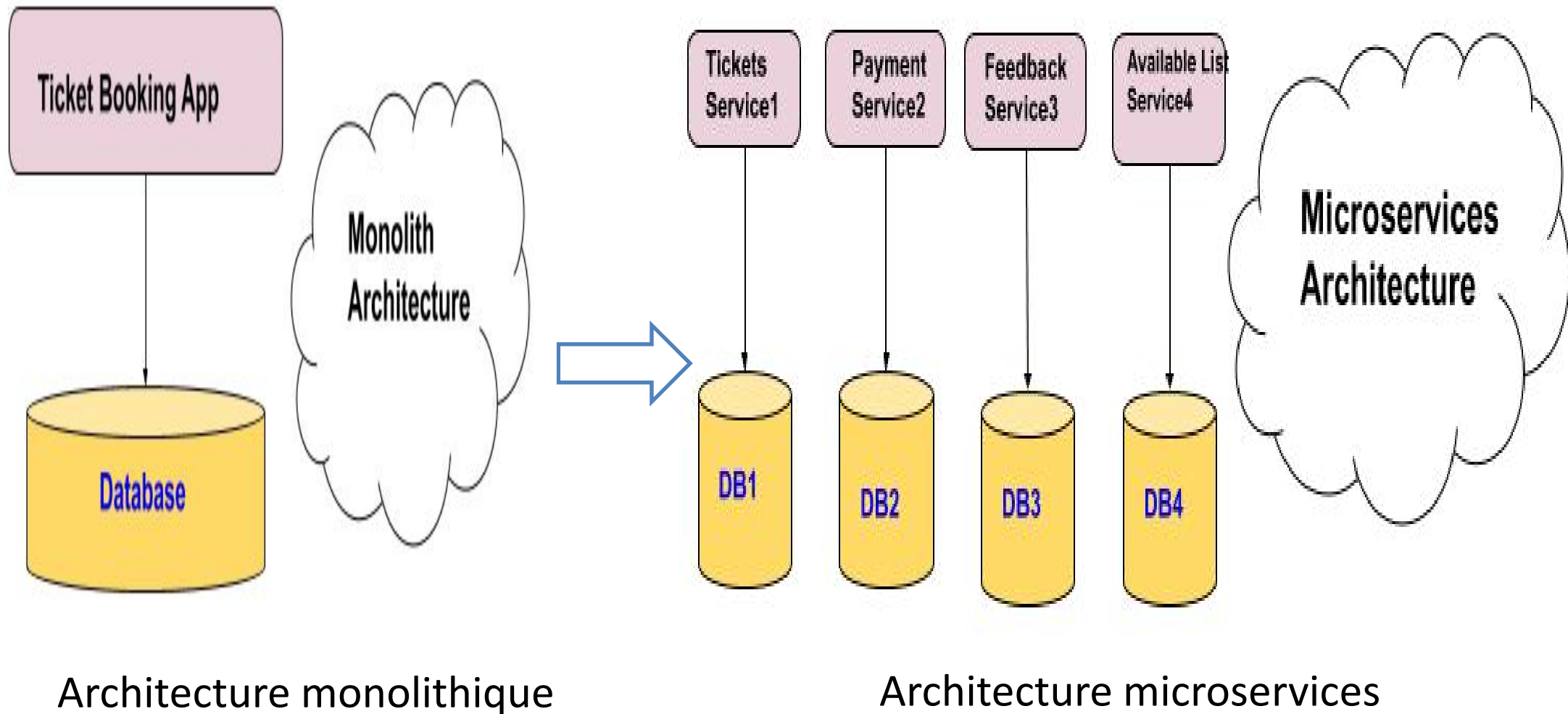
```
<parent>
<groupId>
org.springframework.boot
</groupId>
<artifactId>
spring-boot-starter-parent</artifactId>
<version>2.7.3.RELEASE</version>
<relativePath/>
</parent>
```

AVANTAGES SPRING BOOT

Spring Boot peut s'expliquer simplement par l'illustration ci-dessous:

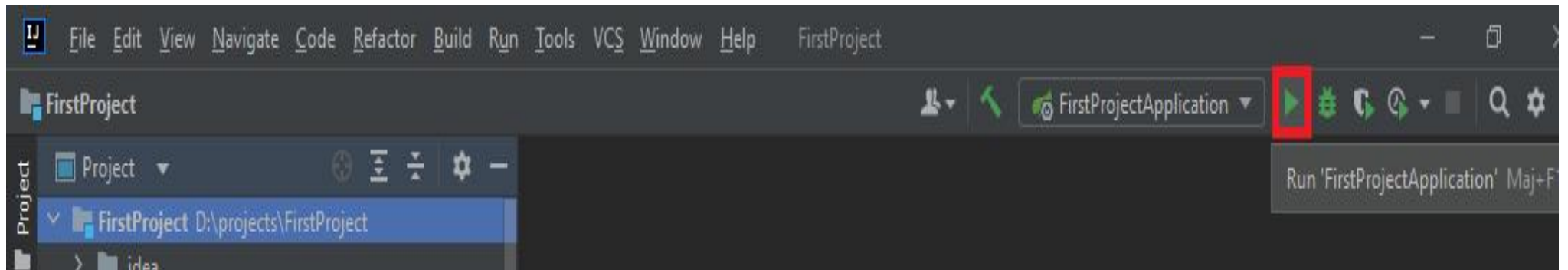


Spring Boot et les microservices (optionnelle)



SPRING BOOT

- ***Comment démarrer un projet Spring Boot :***



- Spring Boot fournit beaucoup de plugins afin de développer et tester des applications Spring Boot rapidement en utilisant les outils de Build comme Maven et Gradle.

DEFINITION MAVEN

Maven est un outil pour le management et l'**automatisation** de production des projets (**construction** des projets) développé par la fondation Apache permettant :

- L'intégration continue
- La gestion des dépendances locales et distantes dans le Modèle de projet basé sur des conventions (POM)
- Automatiser la gestion des builds et la génération des livrables
- Automatisation de tâches récurrentes
- Le lancement des tests



INSTALLATION DE MAVEN

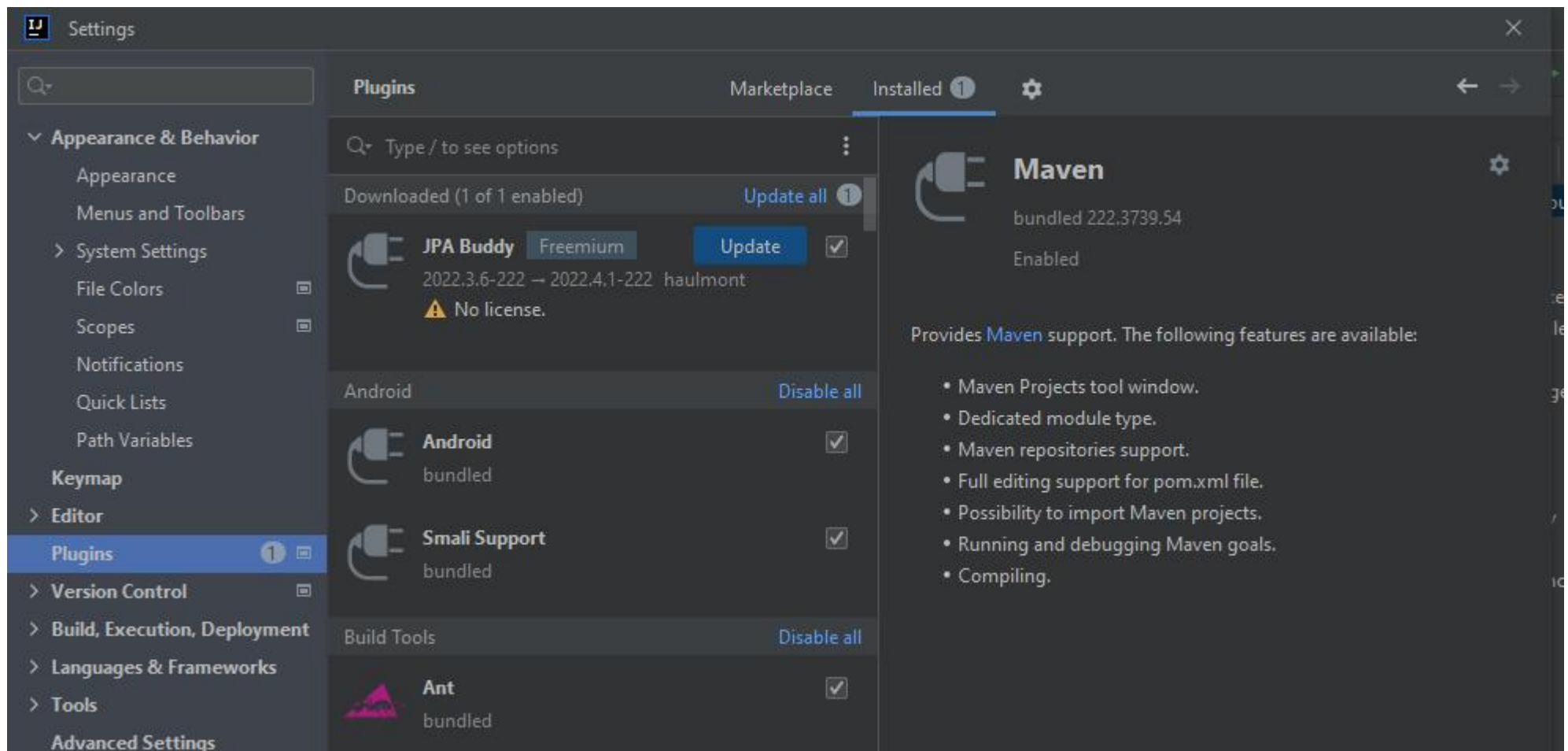
- Maven peut être installé :
 - En mode standalone
Exemple: Utiliser Maven dans l'intégration continue (domaine DevOps)
 - **En tant que plugin**
Par défaut, IntelliJ intègre un plugin Maven

Disque local (C:) > products > apache-tomcat-8.5.28 >

Nom	Modifié le	Type	Taille
backup	10/10/2020 18:15	Dossier de fichiers	
bin	27/09/2020 12:05	Dossier de fichiers	
conf	10/10/2020 18:15	Dossier de fichiers	
lib	27/09/2020 12:05	Dossier de fichiers	
logs	22/01/2021 15:19	Dossier de fichiers	
temp	22/01/2021 15:19	Dossier de fichiers	
webapps	27/09/2020 12:05	Dossier de fichiers	
work	10/10/2020 18:15	Dossier de fichiers	
wtpwebapps	10/10/2020 18:25	Dossier de fichiers	
LICENSE	06/02/2018 22:10	Fichier	57 Ko
NOTICE	06/02/2018 22:10	Fichier	2 Ko
RELEASE-NOTES	06/02/2018 22:10	Fichier	8 Ko
RUNNING	06/02/2018 22:10	Fichier TXT	17 Ko

INSTALLATION DE MAVEN

- En mode plugin :
- Cliquer sur File => Settings => plugins



PREMIER PROJET MAVEN

Un projet est caractérisé par :

- **project** : Balise racine de tous les fichiers pom.xml.
- **modelVersion** : Version de POM utilisée.
- **groupId** : Identifier un groupe qui a créé le projet. Ex: org.apache.
- **artifactId** : Nom unique utilisé pour nommer l'artifact à construire.
- **packaging** : Type de packaging du projet (ex. : JAR, WAR, EAR...).
- **archetype** : Template de Projet.
- **name** : Nom du projet.
- **description** : Description du projet.

Gestion des versions du projet

Exercice

Soit les versions du projet suivantes :

1.0.0

1.1.0

2.0.0

2.0.1

3.0.0

A quoi correspond chaque numérotation ?

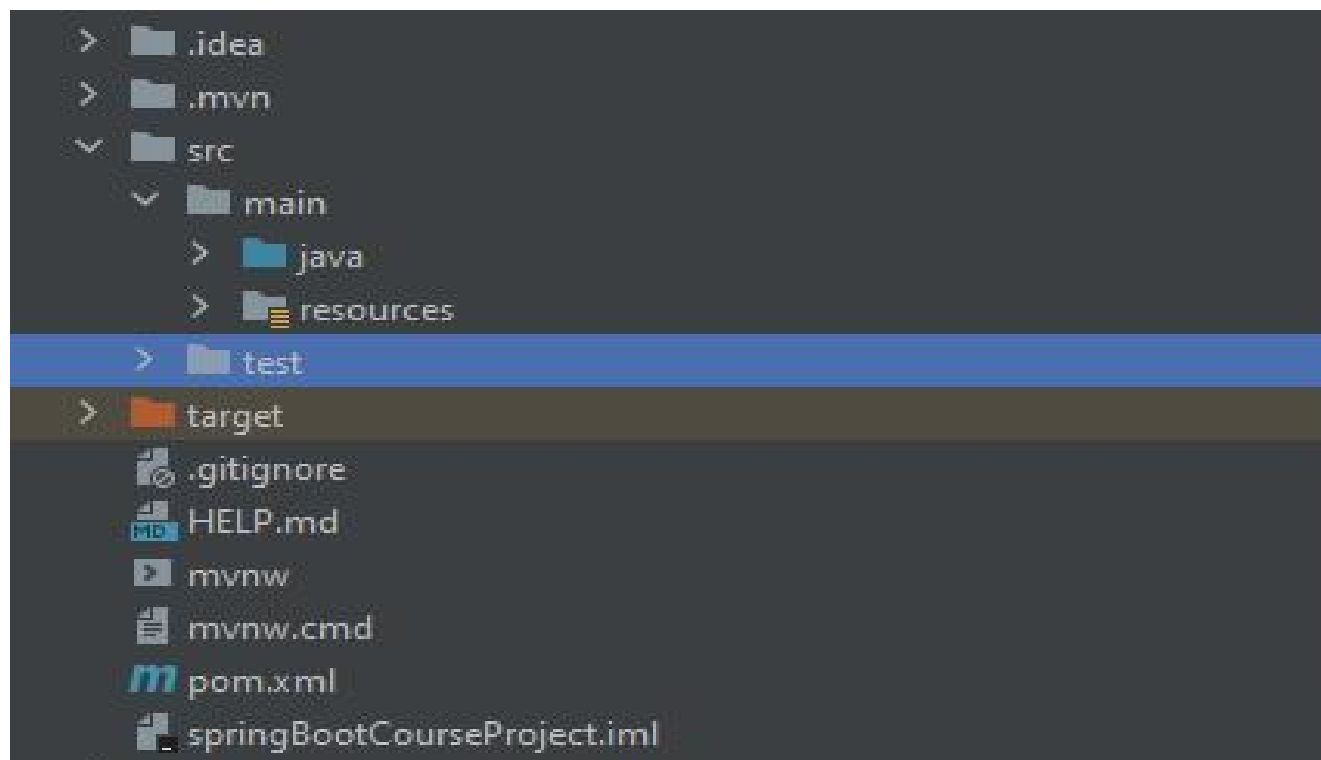
Release majeure

Release mineure

Patch (Correctif)

ARBORESCENCE DU PROJET

- Maven est basé sur la notion de **convention over configuration (arborescence prédéfinie)** ce qui le différencie par rapport à ses concurrents (Gradle, Ant) où une configuration supplémentaire est requise.



ARBORESCENCE DU PROJET

- **pom.xml** : le fichier de configuration du projet
- **/src** : code source et fichiers source principaux
- **/src/main/java** : code source java
- **/src/main/resources** : fichiers de ressources (images, fichiers config...)
- **/src/main/webapp** : webapp du projet
- **/src/test** : fichiers de test
- **/src/test/java** : code source Java de test
- **/src/test/resources** : fichiers de ressources de test
- **/target** : fichiers résultat, les binaires (du code et des tests), les packages générés et les résultats des tests

Fichier POM.XML

- Permet de spécifier les dépendances dont le projet a besoin.
- Spécifie l'emplacement de l'artefact du projet (groupId, ArtifactId, version)

```
<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>com.esprit.spring</groupId>
<artifactId>springBootCourseProject</artifactId>
<version>1.0.0</version>
<name>springBootCourseProject</name>
<description>springBootCourseProject</description>
<dependencies><dependency>
<groupId>org.apache.logging.log4j</groupId>
<artifactId>log4j-core</artifactId>
<version>2.14.0</version>
</dependency></dependencies>
</project>
```







Etapes de construction du projet

- **mvn compile** : Créer les .class
- **mvn test** : Jouer les tests unitaires
- **mvn package** : Création du livrable dans target.
- **mvn install** : Copie du livrable dans le Repository local :
~\.m2\repository\...
- **mvn deploy** : Copie du livrable sur le repository distant
- **mvn clean** : Supprime le contenu du dossier target.

Etapes de construction du projet

- Emplacement du livrable :
 {emplacement Repository}/groupId/artifactId/version
- Nom du package (jar en général) : {artifactId}-{version}.{package}

Ce PC > Disque local (C:) > Utilisateurs > ASUS > .m2 > repository > com > esprit > spring > springBootCourseProject > 1.0.0

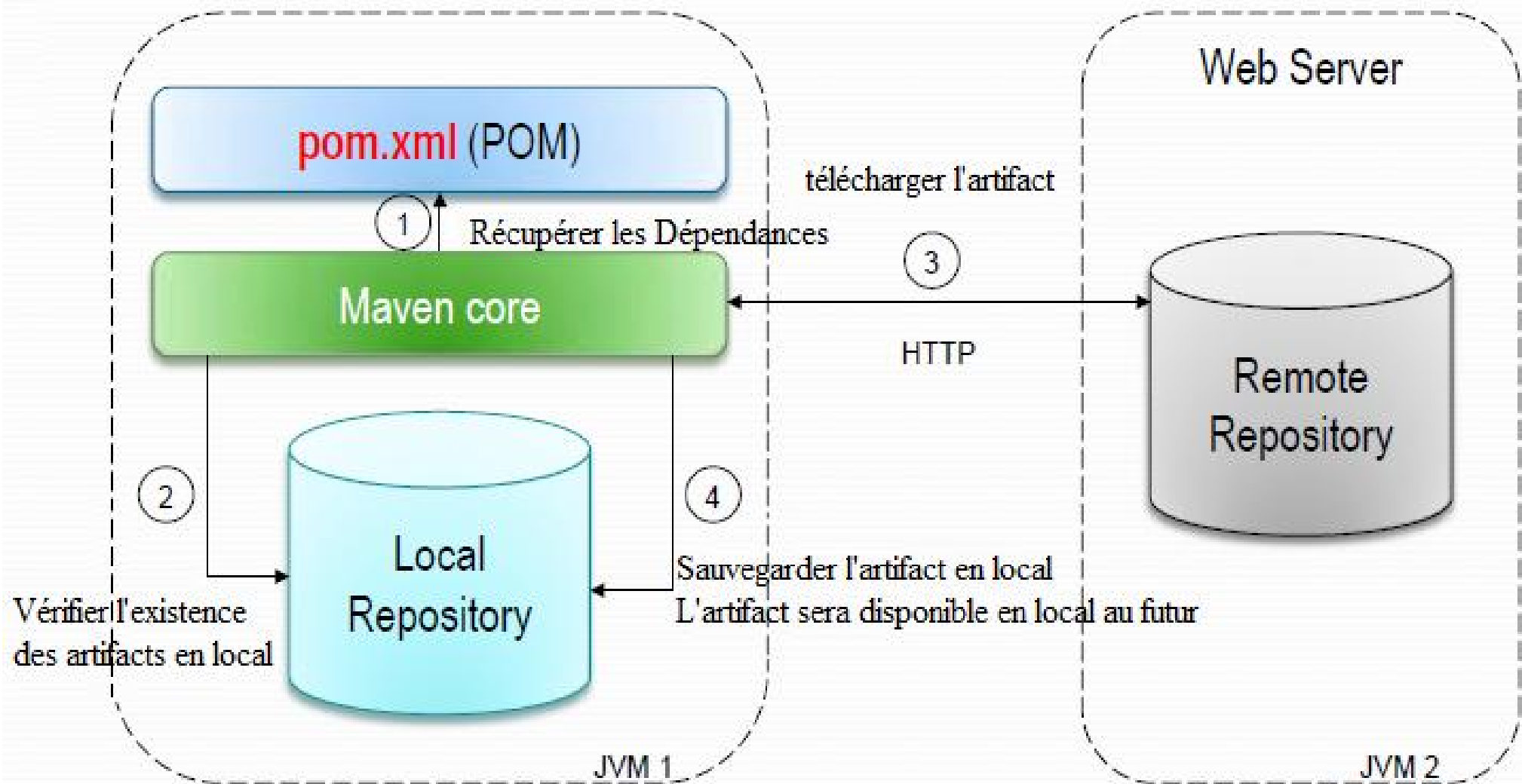
	Nom	Modifié le	Type	Taille
	 _remote.repositories	14/09/2022 07:29	Fichier REPOSITOR...	1 Ko
	 springBootCourseProject-1.0.0	14/09/2022 07:29	Executable Jar File	38 016 Ko
	 springBootCourseProject-1.0.0.pom	14/09/2022 07:29	Fichier POM	3 Ko
				

Gestion des dépendances

Pour ajouter une dépendance, il suffit de chercher la dépendance en question dans le **mvnRepository** (<https://mvnrepository.com/>) et l'inclure dans le pom.xml sous la balise **<dependencies>** comme suit :

```
<dependencies>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.23</version>
</dependency>
</dependencies>
```


Gestion des dépendances



Dépôts Maven

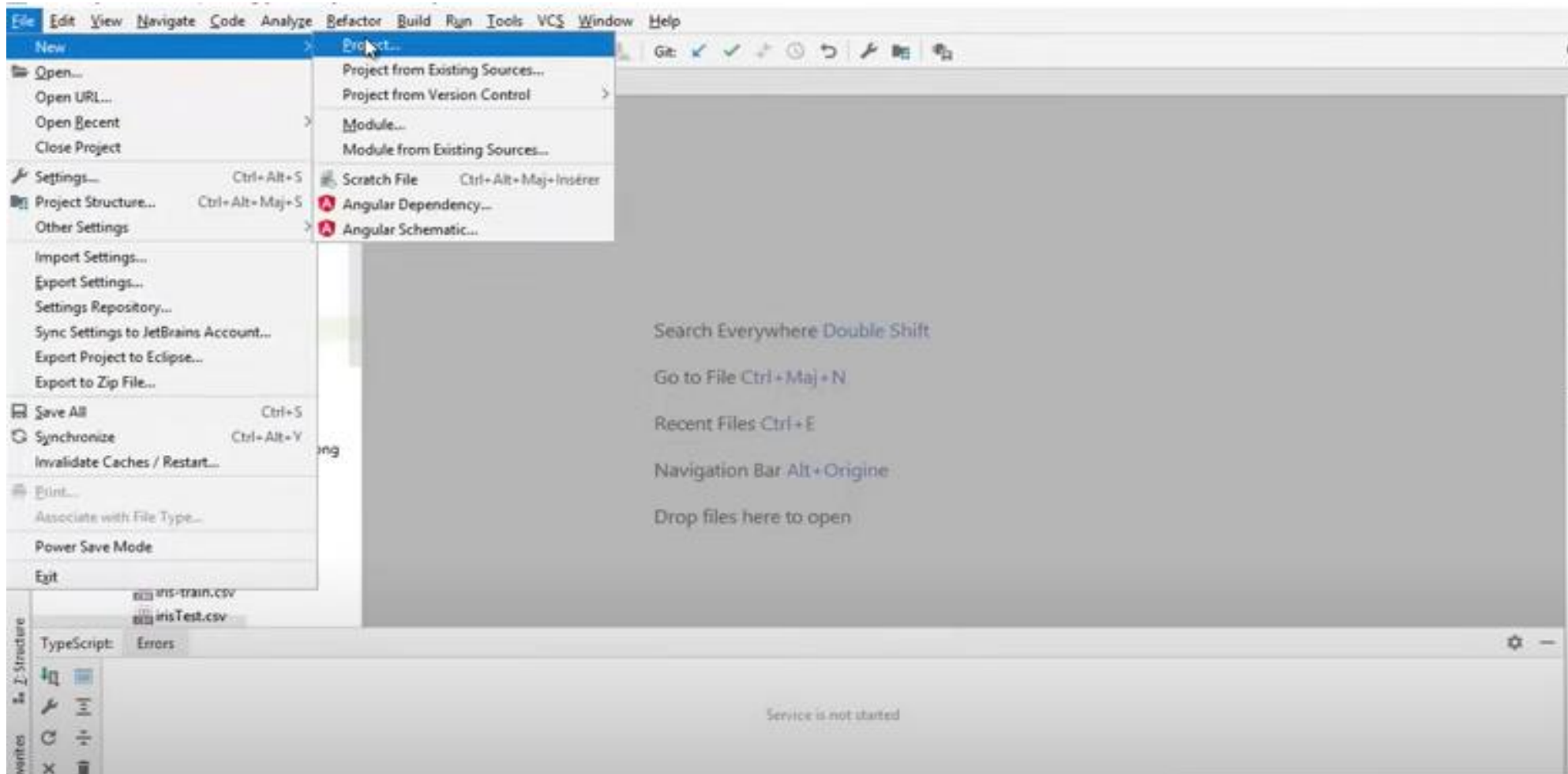
- Maven s'appuie sur les dépôts (repositories) pour stocker les jars de dépendances et des livrables.
- Il y a deux types de repositories :
 - Local : Le dépôt local se trouve par défaut sous l'arborescence `%path_dossier_M2%\repository`
 - Remote (Distant) :
 - Central : dépôt public Maven accessible via <https://mvnrepository.com/>
OU
 - Internal (Private) : dans les serveurs dédiés à l'entreprise (pour des raisons de sécurité)
- Les dépôts sont organisés en groupes, artefacts et versions

TP - Spring Boot- Maven

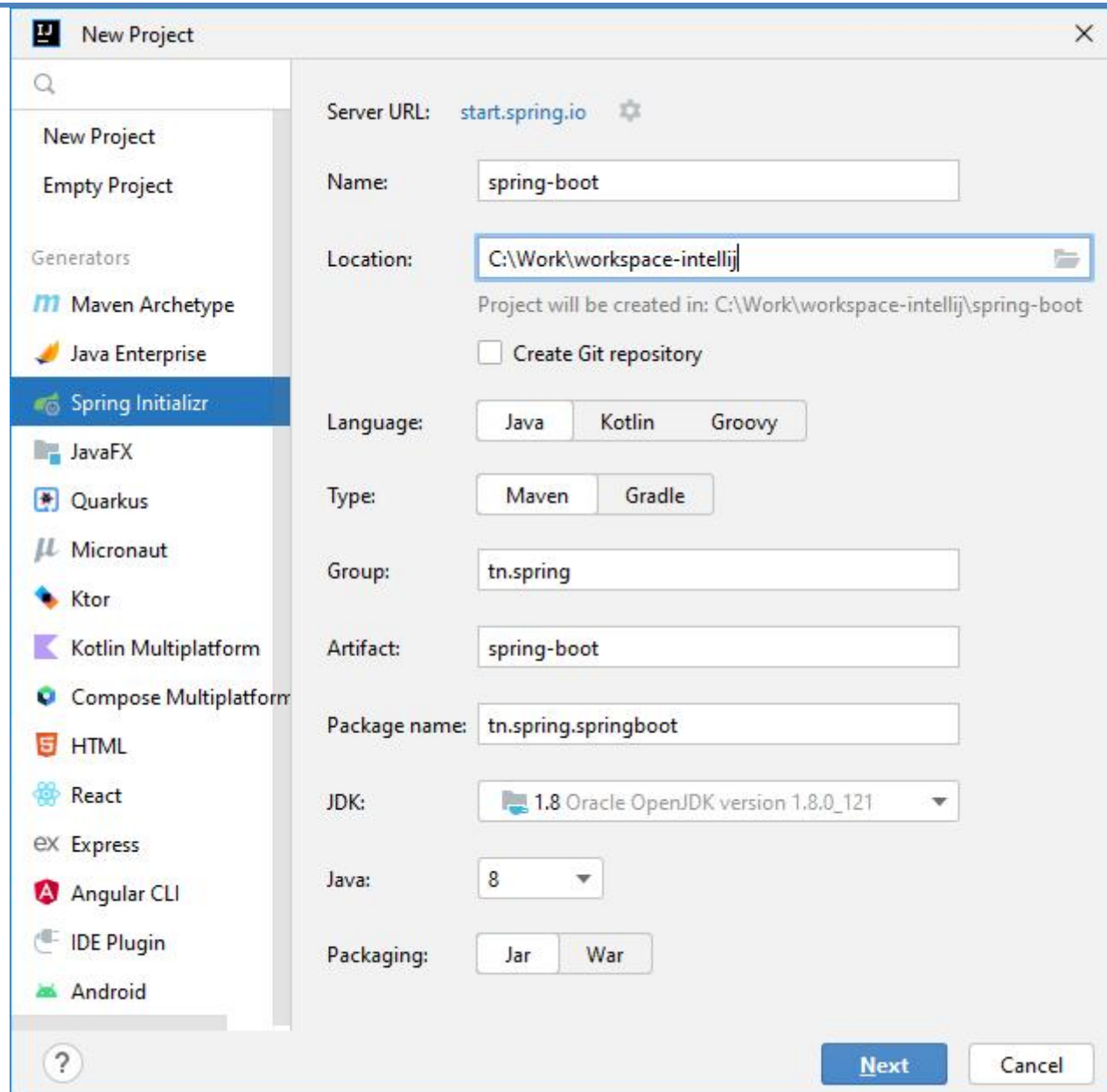
- Nous allons dans ce TP créer notre premier projet Spring Boot.
- Nous allons découvrir les différentes commandes liées au cycle de vie Maven et les appliquer sur le projet spring boot déjà créé
- Il sera utilisé dans la suite des cours (Spring Data JPA)
- Les étapes seront décrites dans les slides suivants :

TP - Spring Boot- Maven

- Création d'un projet spring Boot



TP - Spring Boot- Maven

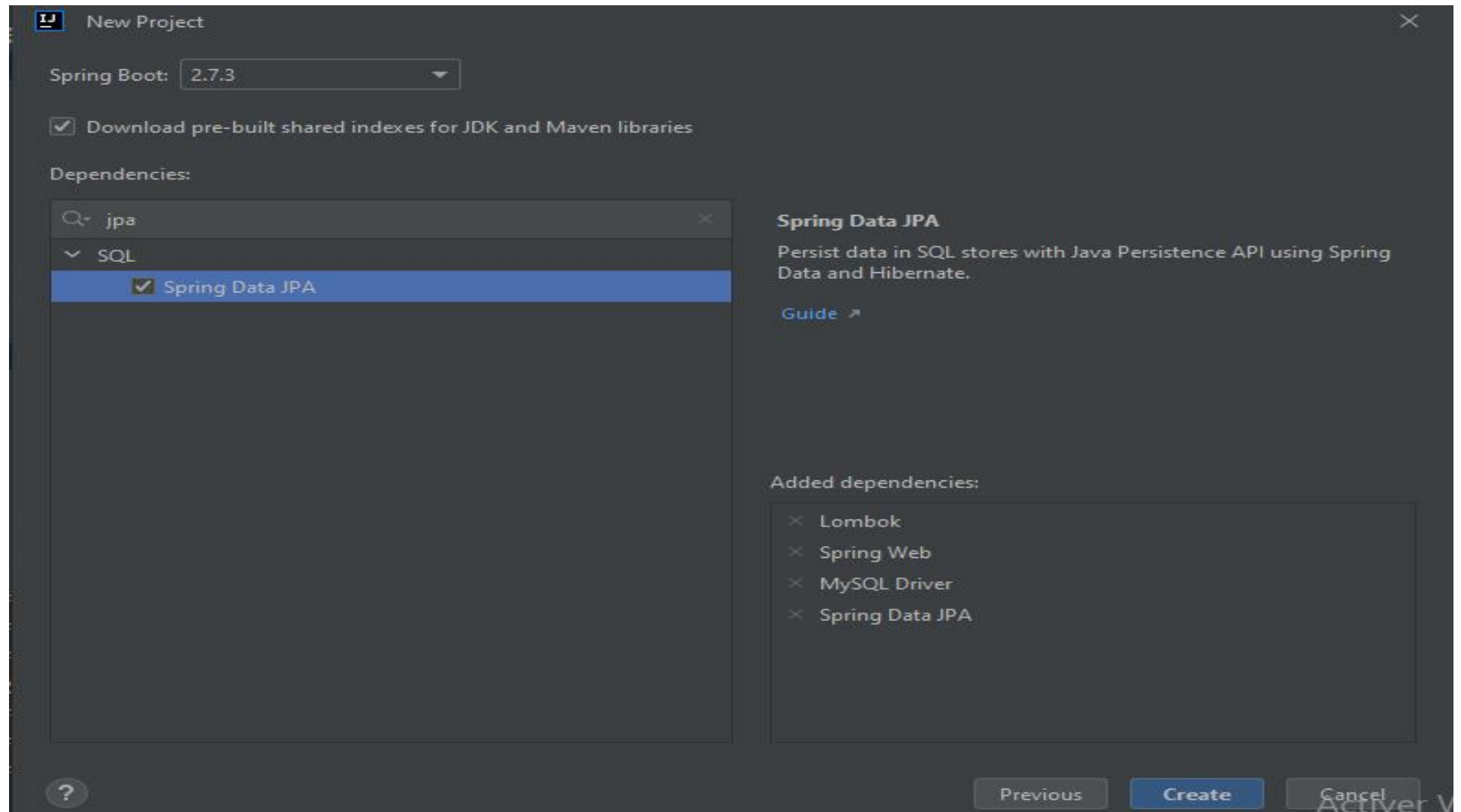


The image shows the 'New Project' dialog in IntelliJ IDEA. The 'Spring Initializr' generator is selected in the left sidebar. The main configuration area on the right contains the following fields and options:

- Server URL:** start.spring.io
- Name:**
- Location:** (with a folder icon on the right)
- Project will be created in:** C:\Work\workspace-intellij\spring-boot
- Create Git repository:** ☐
- Language:** ☒ Java ☐ Kotlin ☐ Groovy
- Type:** ☒ Maven ☐ Gradle
- Group:**
- Artifact:**
- Package name:**
- JDK:**
- Java:**
- Packaging:** ☒ Jar ☐ War

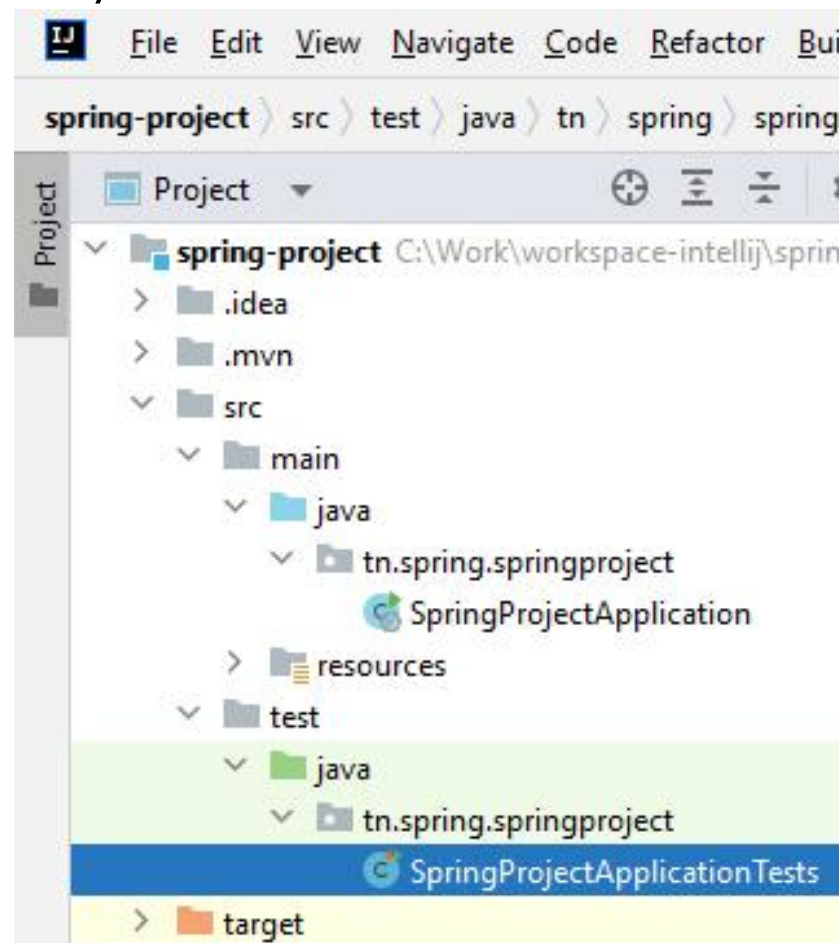
At the bottom right, there are 'Next' and 'Cancel' buttons. A help icon (?) is located at the bottom left of the dialog.

TP - Spring Boot- Maven



TP - Spring Boot- Maven

- Supprimer **la classe de test** pour éviter les erreurs lors de l'appel des commandes Maven (car « Maven install » par exemple essaiera de lancer les tests unitaires) :



TP - Spring Boot- Maven

- Ajouter les propriétés suivantes pour éviter les erreurs lors du lancement des commandes Maven (Comme il y a la dépendance Spring Data JPA, Maven vérifiera s'il y a une base de données de configurée) :

DATABASE

```
spring.datasource.url=jdbc:mysql://localhost:3306/springdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&createDatabaseIfNotExist=true&useLegacyDatetimeCode=false&serverTimezone=UTC
```

```
spring.datasource.username=root
```

```
spring.datasource.password=
```

JPA / HIBERNATE

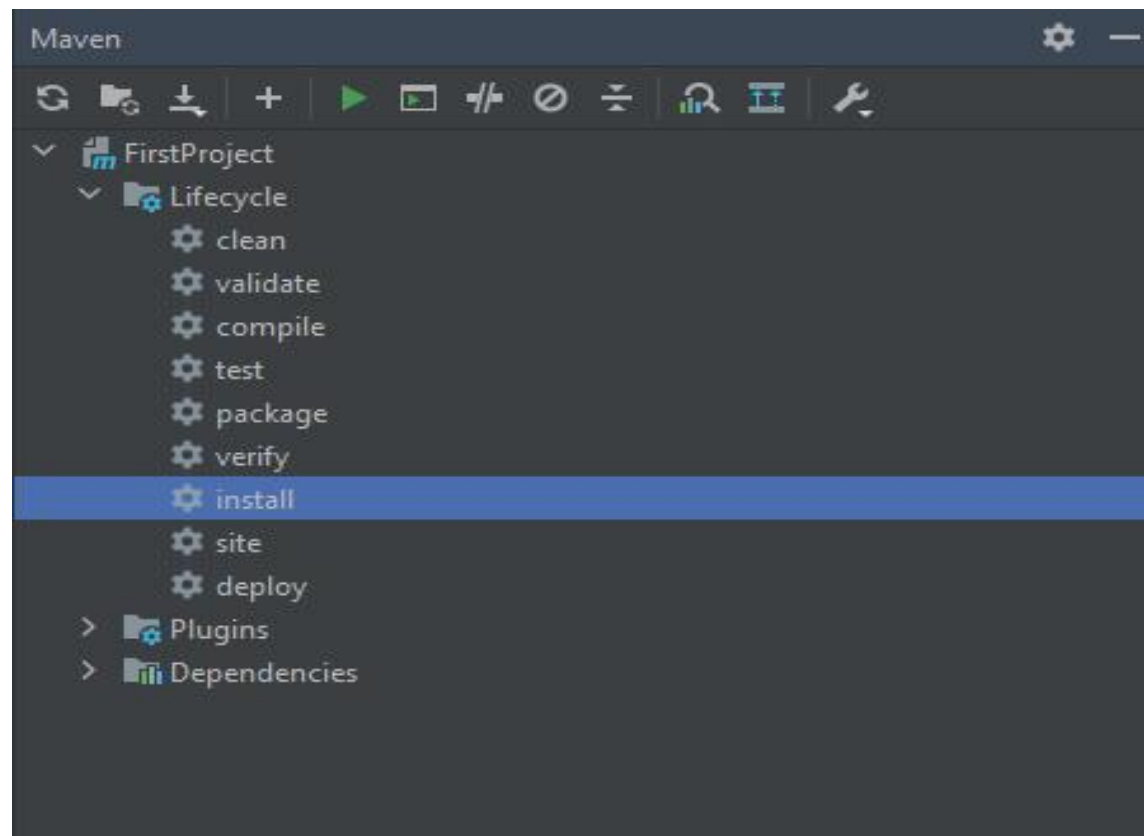
```
spring.jpa.show-sql=true
```

```
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

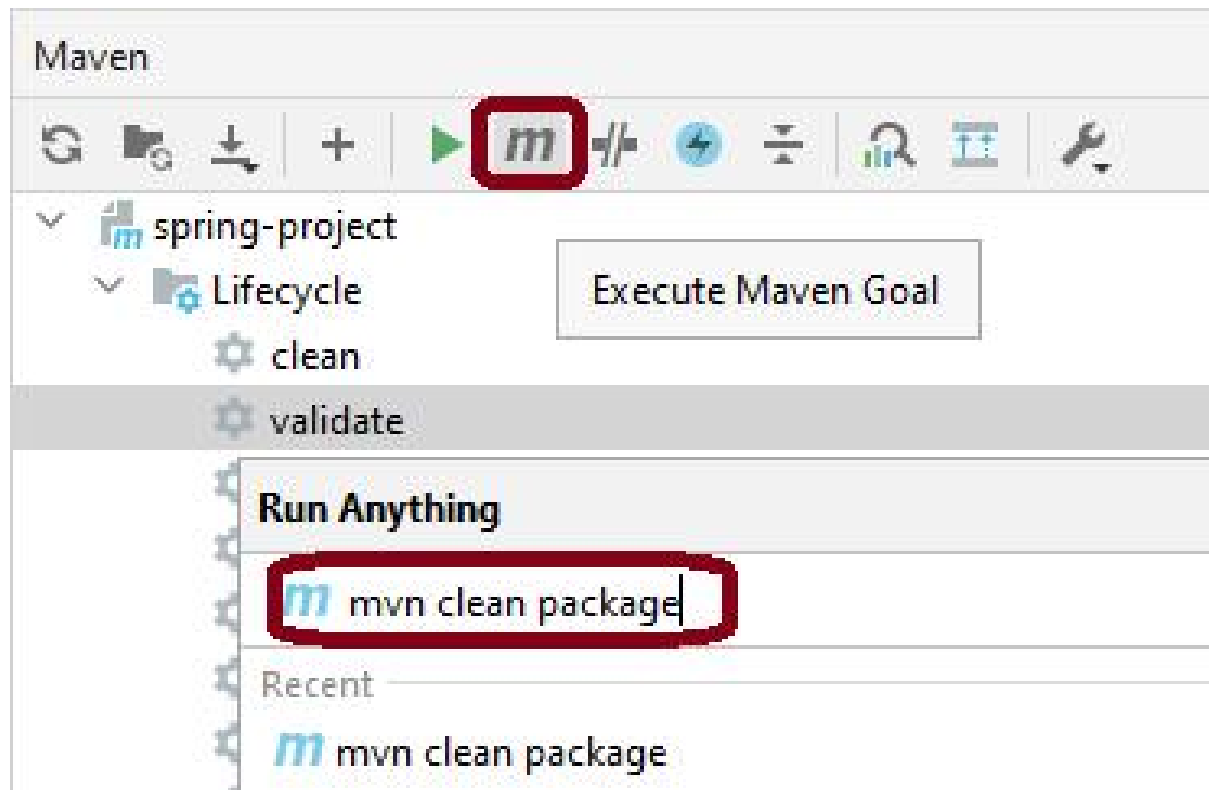

TP - Spring Boot- Maven

Tester les différents commandes maven



TP - Spring Boot- Maven

- On peut faire une commande Maven Customisé en utilisant Maven Goal : Cliquer sur le Symbole «m» et taper la commande voulue, exemple mvn clean package ou mvn clean install ou ... :



Erreur (si ancien MySQL)

- Pour information, nous avons ajouté les paramètres suivants « `?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC` » dans le fichier `application.properties` pour éviter l'erreur suivante, lors de l'exécution :

```
java.sql.SQLException: The server time zone value 'Paris, Madrid' is unrecognized or represents more than one time zone. You must configure either the server or JDBC driver (via the serverTimezone configuration property) to use a more specific time zone value if you want to utilize time zone support.
```
- Nous allons voir tout cela en détails par la suite. Spring Boot sera l'outil qui nous permettra de créer tous nos prochains projets.

SPRING BOOT-Maven

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP ASI
Bureau E204