Spring Boot Maven Plugin

Pour utiliser le `spring-boot-maven-plugin` dans un projet Spring Boot basé sur Maven, suivez ces étapes pour configurer et utiliser le plugin pour construire, exécuter et gérer votre application. Voici un guide complet :

---

1. Assurez-vous que votre projet est un projet Spring Boot

Avant d'utiliser le `spring-boot-maven-plugin`, confirmez que votre projet est configuré comme un projet Spring Boot. Cela implique généralement :

- **Héritage de `spring-boot-starter-parent` (Recommandé)** :
  - Dans votre `pom.xml`, définissez `spring-boot-starter-parent` comme parent pour gérer les dépendances et les versions des plugins Spring Boot.
  - Exemple :

```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version> <!-- Remplacez par votre version de Spring Boot -->
    <relativePath/> <!-- Recherche du parent dans le dépôt -->
</parent>
```

- **Alternativement, Utilisation de `spring-boot-dependencies` BOM (Bill of Materials)** :
  - Si vous ne pouvez pas utiliser `spring-boot-starter-parent`, importez le `spring-boot-dependencies` BOM dans la section `dependencyManagement`.
  - Exemple :

```xml
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>2.5.4</version> <!-- Remplacez par votre version de Spring Boot -->
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

L'utilisation de `spring-boot-starter-parent` est recommandée pour sa simplicité, car elle gère automatiquement les versions des plugins.

---

2. Ajoutez le `spring-boot-maven-plugin` à votre `pom.xml`

Pour utiliser le plugin, vous devez le déclarer dans la section `<build><plugins>` de votre `pom.xml`.

- **Si vous utilisez `spring-boot-starter-parent` :**
  - Ajoutez le plugin sans spécifier la version, car elle est gérée par le parent.
  - Exemple :

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

- **Si vous n'utilisez pas `spring-boot-starter-parent` :**
  - Spécifiez la version explicitement, en la faisant correspondre à la version de Spring Boot utilisée.
  - Exemple :

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>2.5.4</version> <!-- Remplacez par votre version de Spring Boot -->
        </plugin>
    </plugins>
</build>
```

---

3. Utilisez les objectifs du plugin

Le `spring-boot-maven-plugin` fournit plusieurs objectifs pour aider à construire, exécuter et gérer votre application Spring Boot. Voici les objectifs les plus couramment utilisés :

- `spring-boot:run`
  - Exécute votre application Spring Boot directement à partir de Maven en utilisant un serveur web intégré (par exemple, Tomcat).
  - Utile pour le développement et les tests.
  - Commande :

    ```
    mvn spring-boot:run
    ```

- `spring-boot:repackage`
  - Reconditionne le fichier JAR ou WAR généré par `mvn package` en un "fat JAR"ou WAR exécutable qui inclut toutes les dépendances.
  - Cet objectif est automatiquement exécuté pendant la phase `package` si le plugin est configuré.
  - Commande :

    ```
    mvn package
    ```

  - Après l'exécution, vous pouvez démarrer l'application avec :

    ```
    java -jar target/myapp.jar
    ```

- `spring-boot:start` **et** `spring-boot:stop`
  - Utilisés pour les tests d'intégration pour démarrer et arrêter l'application pendant les phases `pre-integration-test` et `post-integration-test`, respectivement.
  - Exemple :

    ```
    mvn spring-boot:start
    mvn spring-boot:stop
    ```

- `spring-boot:build-info`
  - Génère un fichier `build-info.properties` contenant des informations de build (par exemple, heure de build, version).
  - Ces informations peuvent être accessibles dans votre application en utilisant le bean `BuildProperties` de Spring Boot ou les annotations `@Value`.
  - Commande :

    ```
    mvn spring-boot:build-info
    ```

---

4. Personnalisez la configuration du plugin (Optionnel)

Vous pouvez personnaliser le comportement du `spring-boot-maven-plugin` en ajoutant des options de configuration dans le `pom.xml`. Voici quelques personnalisations courantes :

- **Spécifiez la classe principale** :
  - Si le plugin ne peut pas détecter automatiquement la classe principale, spécifiez-la manuellement.
  - Exemple :

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <mainClass>com.example.MyApplication</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>
```

- **Excluez les dépendances du fat JAR** :
  - Excluez les dépendances fournies par l'environnement d'exécution (par exemple, un conteneur de servlets externe).
  - Exemple :

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>com.example</groupId>
                        <artifactId>some-dependency</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

- **Définissez les arguments de l'application** :
  - Configurez les arguments à passer à l'application lors de l'exécution avec `spring-boot:run`.
  - Exemple dans `pom.xml` :

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <arguments>
                    <argument>--server.port=8081</argument>
                </arguments>
            </configuration>
        </plugin>
    </plugins>
</build>
```

  - Alternativement, passez les arguments via la ligne de commande :

```
mvn spring-boot:run -Dspring-boot.run.arguments=--server.port=8081
```

- **Construction de fichiers WAR** :
  - Si vous construisez un fichier WAR pour le déploiement sur un conteneur de servlets externe, assurez-vous que le packaging de votre projet est défini sur `war` dans `pom.xml` :

```
<packaging>war</packaging>
```

  - Vous devrez peut-être également exclure les conteneurs de servlets intégrés (par exemple, Tomcat) s'ils sont fournis par l'environnement. Ajoutez la dépendance suivante comme `provided` :

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

---

5. Notes clés

- **Comportement par défaut** :

  - Lorsque vous exécutez `mvn package`, le plugin reconditionne automatiquement le JAR/WAR pendant la phase `package` pour le rendre exécutable.
  - L'artefact reconditionné est l'artefact principal, et le JAR/WAR original est attaché avec un classificateur (par exemple, `original`).

- **Flux de travail de développement** :
  - Utilisez `mvn spring-boot:run` pour un développement et des tests rapides.
  - Utilisez `mvn package` pour construire un JAR ou WAR exécutable déployable.

- **Tests d'intégration** :
  - Utilisez `spring-boot:start` et `spring-boot:stop` pour gérer le cycle de vie de l'application pendant les tests d'intégration.

Résumé

Pour utiliser le `spring-boot-maven-plugin` : 1. Assurez-vous que votre projet est un projet Spring Boot, généralement en héritant de `spring-boot-starter-parent` ou en utilisant le BOM `spring-boot-dependencies`. 2. Ajoutez le plugin à la section `<build><plugins>` de votre `pom.xml`. 3. Utilisez les commandes Maven comme : - `mvn spring-boot:run` pour exécuter l'application. - `mvn package` pour construire un JAR/WAR exécutable. - D'autres objectifs comme `spring-boot:start`, `spring-boot:stop`, et `spring-boot:build-info` selon les besoins. 4. Optionnellement, personnalisez la configuration du plugin dans `pom.xml` pour des cas d'utilisation avancés.

Cette configuration vous permet de construire, exécuter et gérer efficacement les applications Spring Boot en utilisant Maven.