Spring Boot Maven Plugin

Um das `spring-boot-maven-plugin` in einem Maven-basierten Spring Boot-Projekt zu verwenden, befolgen Sie diese Schritte, um das Plugin für das Erstellen, Ausführen und Verwalten Ihrer Anwendung zu konfigurieren und zu nutzen. Hier ist eine umfassende Anleitung:

---

1. Stellen Sie sicher, dass Ihr Projekt ein Spring Boot-Projekt ist

Bevor Sie das `spring-boot-maven-plugin` verwenden, stellen Sie sicher, dass Ihr Projekt als Spring Boot-Projekt eingerichtet ist. Dies umfasst in der Regel:

- **Erweiterung von `spring-boot-starter-parent` (Empfohlen)**:
  - In Ihrem `pom.xml` setzen Sie `spring-boot-starter-parent` als Eltern, um Spring Boot-Abhängigkeiten und Plugin-Versionen zu verwalten.
  - Beispiel:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version> <!-- Ersetzen Sie durch Ihre Spring Boot-Version -->
    <relativePath/> <!-- Eltern aus dem Repository suchen -->
</parent>
```

- **Alternativ, Verwenden von `spring-boot-dependencies` BOM (Bill of Materials)**:
  - Wenn Sie `spring-boot-starter-parent` nicht verwenden können, importieren Sie das `spring-boot-dependenci` BOM im `dependencyManagement`-Bereich.
  - Beispiel:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>2.5.4</version> <!-- Ersetzen Sie durch Ihre Spring Boot-Version -->
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Die Verwendung von `spring-boot-starter-parent` wird empfohlen, da sie die Plugin-Versionen automatisch verwaltet.

---

2. Fügen Sie das `spring-boot-maven-plugin` zu Ihrem `pom.xml` hinzu

Um das Plugin zu verwenden, müssen Sie es im `<build><plugins>`-Bereich Ihres `pom.xml` deklarieren.

- **Wenn `spring-boot-starter-parent` verwendet wird**:
  - Fügen Sie das Plugin ohne Angabe der Version hinzu, da es vom Eltern verwaltet wird.
  - Beispiel:

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

- **Wenn `spring-boot-starter-parent` nicht verwendet wird**:
  - Geben Sie die Version explizit an, passend zur verwendeten Spring Boot-Version.
  - Beispiel:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>2.5.4</version> <!-- Ersetzen Sie durch Ihre Spring Boot-Version -->
        </plugin>
    </plugins>
</build>
```

---

3. Verwenden Sie Plugin-Ziele

Das `spring-boot-maven-plugin` bietet mehrere Ziele, um das Erstellen, Ausführen und Verwalten Ihrer Spring Boot-Anwendung zu unterstützen. Hier sind die am häufigsten verwendeten Ziele:

- `spring-boot:run`
  - Führt Ihre Spring Boot-Anwendung direkt aus Maven mit einem eingebetteten Webserver (z.B. Tomcat) aus.
  - Nützlich für Entwicklung und Test.
  - Befehl:

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

- `spring-boot:repackage`
  - Verpackt die JAR- oder WAR-Datei, die von `mvn package` erzeugt wurde, in eine ausführbare "fat JAR"oder WAR, die alle Abhängigkeiten enthält.
  - Dieses Ziel wird während der `package`-Phase automatisch ausgeführt, wenn das Plugin konfiguriert ist.
  - Befehl:

    ```
    mvn package
    ```

  - Nach dem Ausführen können Sie die Anwendung mit starten:

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

- `spring-boot:start` **und** `spring-boot:stop`
  - Wird für Integrationstests verwendet, um die Anwendung während der `pre-integration-test` und `post-integration-test`-Phasen zu starten und zu stoppen.
  - Beispiel:

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

- `spring-boot:build-info`
  - Erstellt eine `build-info.properties`-Datei, die Build-Informationen (z.B. Build-Zeit, Version) enthält.
  - Diese Informationen können in Ihrer Anwendung mit dem `BuildProperties`-Bean oder `@Value`-Annotations von Spring Boot abgerufen werden.
  - Befehl:

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

---

4. Anpassen der Plugin-Konfiguration (Optional)

Sie können das Verhalten des `spring-boot-maven-plugin` anpassen, indem Sie Konfigurationsoptionen im `pom.xml` hinzufügen. Hier sind einige gängige Anpassungen:

- **Hauptklasse angeben**:
  - Wenn das Plugin die Hauptklasse nicht automatisch erkennen kann, geben Sie sie manuell an.
  - Beispiel:

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

- **Abhängigkeiten von der Fat JAR ausschließen**:
  - Schließen Sie Abhängigkeiten aus, die von der Laufzeitumgebung bereitgestellt werden (z.B. ein externer Servlet-Container).
  - Beispiel:

```
<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>
```

- **Anwendungsargumente festlegen**:
  - Konfigurieren Sie Argumente, die an die Anwendung beim Ausführen mit `spring-boot:run` übergeben werden.
  - Beispiel in `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>
```

  - Alternativ können Sie Argumente über die Befehlszeile übergeben:

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

- **Erstellen von WAR-Dateien**:
  - Wenn Sie eine WAR-Datei für die Bereitstellung in einem externen Servlet-Container erstellen, stellen Sie sicher, dass Ihr Projekt-Packaging in `pom.xml` auf `war` gesetzt ist:

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

  - Sie müssen möglicherweise eingebettete Servlet-Container (z.B. Tomcat) ausschließen, wenn sie von der Umgebung bereitgestellt werden. Fügen Sie die folgende Abhängigkeit als `provided` hinzu:

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

---

5. Wichtige Hinweise

- **Standardverhalten**:

  - Wenn Sie `mvn package` ausführen, verpackt das Plugin die JAR/WAR während der `package`-Phase automatisch neu, um sie ausführbar zu machen.
  - Das neu verpackte Artefakt ist das Hauptartefakt, und die ursprüngliche JAR/WAR wird mit einem Klassifikator (z.B. `original`) angehängt.

- **Entwicklungsworkflow**:
  - Verwenden Sie `mvn spring-boot:run` für schnelle Entwicklung und Tests.
  - Verwenden Sie `mvn package`, um eine ausführbare JAR- oder WAR-Datei zu erstellen.

- **Integrationstests**:
  - Verwenden Sie `spring-boot:start` und `spring-boot:stop`, um den Anwendungslebenszyklus während der Integrationstests zu verwalten.

Zusammenfassung

Um das `spring-boot-maven-plugin` zu verwenden: 1. Stellen Sie sicher, dass Ihr Projekt ein Spring Boot-Projekt ist, in der Regel durch Erweiterung von `spring-boot-starter-parent` oder Verwendung des `spring-boot-dependencies` BOM. 2. Fügen Sie das Plugin dem `<build><plugins>`-Bereich Ihres `pom.xml` hinzu. 3. Verwenden Sie Maven-Befehle wie: - `mvn spring-boot:run`, um die Anwendung auszuführen. - `mvn package`, um eine ausführbare JAR/WAR zu erstellen. - Weitere Ziele wie `spring-boot:start`, `spring-boot:stop` und `spring-boot:build-info` nach Bedarf. 4. Optional können Sie die Plugin-Konfiguration im `pom.xml` für fortgeschrittene Anwendungsfälle anpassen.

Diese Einrichtung ermöglicht es Ihnen, Spring Boot-Anwendungen effizient mit Maven zu erstellen, auszuführen und zu verwalten.