Using the Plugin

Maven users can inherit from the `spring-boot-starter-parent` project to obtain sensible defaults. The parent project provides the following features:

Edit this Page
GitHub Project
Stack Overflow

- UTF-8 source encoding.
- Compilation with `-parameters`.
- A dependency management section, inherited from the `spring-boot-dependencies` POM, that manages the versions of common dependencies. This dependency management lets you omit `<version>` tags for those dependencies when used in your own POM.
- An execution of the `repackage` goal with a `repackage` execution id.
- A `native` profile that configures the build to be able to generate a Native image.
- Sensible resource filtering.
- Sensible plugin configuration (Git commit ID, and shade).
- Sensible resource filtering for `application.properties` and `application.yml` including profile-specific files (for example, `application-dev.properties` and `application-dev.yml`)

NOTE

Since the `application.properties` and `application.yml` files accept Spring style placeholders (`${…}`), the Maven filtering is changed to use `@..@` placeholders. (You can override that by setting a Maven property called `resource.delimiter`.)

NOTE

The `spring-boot-starter-parent` sets the `maven.compiler.release` property, which restricts the `--add-exports`, `--add-reads`, and `--patch-module` options if they modify system modules. In case you need to use those options, unset `maven.compiler.release`:

```
<maven.compiler.release></maven.compiler.release>
```

and then configure the source and the target options instead:

```
<maven.compiler.source>${java.version}</maven.compiler.source>
<maven.compiler.target>${java.version}</maven.compiler.target>
```

Inheriting the Starter Parent POM

To configure your project to inherit from the `spring-boot-starter-parent`, set the `parent` as follows:

```xml
<!-- Inherit defaults from Spring Boot -->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.0</version>
</parent>
```

NOTE

You should need to specify only the Spring Boot version number on this dependency. If you import additional starters, you can safely omit the version number.

With that setup, you can also override individual dependencies by overriding a property in your own project. For instance, to use a different version of the SLF4J library and the Spring Data release train, you would add the following to your `pom.xml`:

```xml
<properties>
    <slf4j.version>1.7.30</slf4j.version>
    <spring-data-releasetrain.version>Moore-SR6</spring-data-releasetrain
</properties>
```

Browse the Dependency Versions Properties section in the Spring Boot reference for a com-

plete list of dependency version properties.

Using Spring Boot without the Parent POM

There may be reasons for you not to inherit from the `spring-boot-starter-parent` POM. You may have your own corporate standard parent that you need to use or you may prefer to explicitly declare all your Maven configuration.

If you do not want to use the `spring-boot-starter-parent`, you can still keep the benefit of the dependency management (but not the plugin management) by using an `import` scoped dependency, as follows:

```xml
<dependencyManagement>
    <dependencies>
        <dependency>
            <!-- Import dependency management from Spring Boot -->
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>3.5.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

The preceding sample setup does not let you override individual dependencies by using properties, as explained above. To achieve the same result, you need to add entries in the `dependencyManagement` section of your project **before** the `spring-boot-dependencies` entry. For instance, to use a different version of the SLF4J library and the Spring Data release train, you could add the following elements to your `pom.xml`:

```xml
<dependencyManagement>
    <dependencies>
        <!-- Override SLF4J provided by Spring Boot -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.30</version>
        </dependency>
        <!-- Override Spring Data release train provided by Spring Boot --
        <dependency>
            <groupId>org.springframework.data</groupId>
            <artifactId>spring-data-releasetrain</artifactId>
            <version>2020.0.0-SR1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>3.5.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Overriding Settings on the Command Line

The plugin offers a number of user properties, starting with `spring-boot`, to let you customize the configuration from the command line.

For instance, you could tune the profiles to enable when running the application as follows:

```shell
$ mvn spring-boot:run -Dspring-boot.run.profiles=dev,local
```

If you want to both have a default while allowing it to be overridden on the command line, you should use a combination of a user-provided project property and MOJO configuration.

```xml
<project>
    <properties>
        <app.profiles>local,dev</app.profiles>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
```

```
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <profiles>${app.profiles}</profiles>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

The above makes sure that `local` and `dev` are enabled by default. Now a dedicated property has been exposed, this can be overridden on the command line as well:

SHELL

```
$ mvn spring-boot:run -Dapp.profiles=test
```

Prev
Getting Started

Next
Goals