IzPack Maven Plugin Reference

Lifecycle Mappings

The lifecycle mapping to use in a POM launching the IzPack Maven plugin depends on your needs.

In common, there are the following mappings useful with izpack-maven-plugin:

- **pom**
  This is a standard lifecycle mapping in Maven, intended for just grouping several submodules, but almost not doing any job. The izpack-maven-plugin must be explicitely launched in a phase of your choice.
- **izpack-jar**
  This is a new mapping coming with IzPack 5.0, reducing the complexity of a POM launching the izpack-maven-plugin.
  It acts similar than the jar lifecycle mapping in Maven except it replaces launching the maven-jar-plugin in the phase package by launching izpack-maven-plugin, instead. This can be used as a snap-in for building installer jars instead of compiled jars from Java code, especially combined with the features for attaching an IzPack installer a direct or classified Maven artifact to the project.
  The izpack-jar packaging type cannot be used in a module compiling Java source code into a Jar file, but is intended to be used in a standalone module just doing the job of assembling and building an installer. This increases the transparency of your project and using izpack-jar decreases the complexity of the POM.

For bigger projects with many modules it is a good practise to separate launching of the IzPack Maven Plugin in a separate modules or even more, having one modul per application installer to be built.

> **Note**
>
> For the **izpack-jar** mapping is important to add the **<extensions>true</extensions>**. If you don't, Maven will not be able to see this as a valid packaging. See the example usage below.

Configuration

Configuration Attribute	Value Range (Default)	Default Value	Description
baseDir	string	`${project.build.directory}/staging`	Base directory of compilation process
installFile	string	`${basedir}/src/main/izpack/install.xml`	Target location of the compiled installer jar

comprFormat	`default | bzip2`	default	Target compression format of the compiled installer jar
comprLevel	integer -1, 0..9	`-1`	Compression level of the compiled installer jar. Deactivated by default (-1)
outputDirectory		`${project.build.directory}`	Target directory containing the compiled installer jar
mkdirs	`true | false`	`false`	Whether to automatically create parent directories of the output file
finalName			Name of the compiled installer jar
classifier		Not set	Classifier to add to the artifact generated. If given, the artifact is attachable. Furthermore, the output file name gets -*classifier* as suffix. If this is not given,it will merely be written to the output directory according to the finalName.
enableAttachArtifact	`true | false`	`true`	Whether to attach the generated installer jar to the project as artifact if a classifier is specified. This has no effect if no classifier was specified.
enableOverrideArtifact	`true | false`	`false`	Whether to override the artifact file by the generated installer jar, if no classifier is specified. This will set the artifact file to the given name based on *outputDirectory* + *final Name* or on *output*. This has no effect if a classifier is specified.
autoIncludeUrl	`true | false`	`false`	Whether to automatically include project.url from Maven into the IzPack info header

autoIncludeDevelopers	`true | false`	`false`	Whether to automatically include developer list from Maven into IzPack info header
kind	`standard | web`	`standard`	Resulting installer type

Example Usage

Example usage of IzPack Maven Plugin

```
<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/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  ...


  <!-- Launch IzPack automatically -->
  <packaging>izpack-jar</packaging>

  <properties>
    <!-- Installer variables -->

<staging.dir>${project.build.directory}/stag
ing</staging.dir>
```

```
    <info.appName>My Killer
Application</info.appName>

<info.appsubpath>my-killer-app/standard</inf
o.appsubpath>

<izpack.dir.app>${basedir}/src/main/izpack</
izpack.dir.app>

<staging.dir.app>${staging.dir}/appfiles</st
aging.dir.app>
  </properties>



  ...

  <plugin>
    <groupId>org.codehaus.izpack</groupId>

<artifactId>izpack-maven-plugin</artifactId>
    <extensions>true</extensions>
    <configuration>
      <baseDir>${staging.dir.app}</baseDir>

<installFile>${izpack.dir.app}/install.xml</
installFile>

<outputDirectory>${project.build.directory}<
/outputDirectory>

<finalName>${project.build.finalName}</final
```

```
Name>

<enableOverrideArtifact>true</enableOverride
Artifact>
      <mkdirs>true</mkdirs>
      <autoIncludeUrl>false</autoIncludeUrl>

<autoIncludeDevelopers>false</autoIncludeDev
elopers>
    </configuration>
  </plugin>
```

```
  ...

</project>
```