

# Using Maven To Build a Library Jar

## Table of contents

1 Overview.....	2
2 Prerequisites:.....	2
3 How the maven-ctl-plugin works.....	2
4 Plugin Details.....	2
5 Creating a new Project.....	3
5.1 1. archetype:create.....	3
5.2 2. ProjectBuilder Register.....	3
5.3 3. Build the library.....	3
6 Using existing sources.....	4
6.1 1. ProjectBuilder Register.....	4
6.2 2. Archetype Create.....	4
7 Notes.....	5
7.1 Repository Access.....	5
7.2 Installing the ctier-library-jar archetype.....	6

## 1. Overview

This document describes how to use the CTL maven plugin and the ControlTier Library Jar maven archetype to build a ControlTier library jar using maven.

## 2. Prerequisites:

- ControlTier server and CTL client.  
The CTL base directory should be set as an environment variable named \$CTL\_BASE. (if not, see [Notes](#))
- A project should be defined in the ControlTier server. The name of this project is referred to as the \$DEP
- You should have a name for your ControlTier Library, referred to as \$NAME.
- If you have existing library sources, you can skip to the [Using Existing Sources](#) section.

## 3. How the maven-ctl-plugin works

The maven-ctl-plugin allows you to invoke any CTL command from any maven build phase. The plugin must be configured in the pom.xml to attach to a build phase, and to specify the required parameters for executing an CTL command. The parameters required are: ctlBase directory, depot name, type name, object name, command name, and any option command arguments.

See the Document [Maven Integration](#) for more detail about the maven-ctl-plugin.

Using the maven-ctl-plugin, we can use ProjectBuilder's build-library command to package the modules located in a source directory into their respective jar files, and then we can use Maven's built-in jar plugin create the final library jar with the necessary Manifest components.

The pom.xml necessary for doing this can be created for you using the Maven Archetype plugin, and the ctier-library-jar archetype. Then the only necessary actions are to using the archetype:create goal of the Archetype plugin, and to register a ProjectBuilder object within ControlTier to build the module jars.

## 4. Plugin Details

groupId	artifactId	latest version
com.controltier	ctier-library-jar	1.0-SNAPSHOT

See [Maven Repository](#) for the ControlTier maven repository URL.

## 5. Creating a new Project

### 5.1. 1. archetype:create

Run the maven archetype:create command to create a project and pom.xml you can use to build your library jar:

```
set GROUPID=com.mycompany
set VERSION=1.0-SNAPSHOT
mvn archetype:create -DarchetypeGroupId=com.controltier
-DarchetypeArtifactId=ctier-library-jar \
-DarchetypeVersion=1.0-SNAPSHOT -DgroupId=$GROUPID -DartifactId=$NAME
\
-Dversion=$VERSION
-DremoteRepositories=http://open.controltier.org/repo
```

The usual initial version is "1.0-SNAPSHOT". Specify your groupId similar to a Java package, e.g. "com.mycompany".

This command will create a directory named \$NAME, and a file \$NAME/pom.xml file for you.

**Note:**

the "-DremoteRepositories=..." is needed to install the ctier-library-jar archetype from the ControlTier maven repository the first time. Subsequent invocations will not need the "-DremoteRepositories" section.

### 5.2. 2. ProjectBuilder Register

Create a ProjectBuilder object to manage your module code:

```
cd $NAME
set BASE=`pwd`
ad -p $DEPOT -t ProjectBuilder -o $NAME -c Register -- -basedir $BASE/src
-installroot $BASE/target -install
```

This will create the ProjectBuilder object, and deploy it locally.

Create the directory for your modules:

```
mkdir $BASE/src/modules
```

Now you can develop modules under the structure  
\$BASE/src/modules/<modulename>/type.xml.

### 5.3. 3. Build the library

To build the library jar, first edit the pom.xml.

Replace the "\${context.depot}" with \$DEPOT defined above. (If necessary, replace the <name>\${project.artifactId}</name> parameter in the configuration of the maven-ctl-plugin to correspond to \$NAME, if it is not equivalent to your maven project's artifactId.).

Execute the package command:

```
mvn package
```

**Note:**

You can also define the context.depot variable on the commandline, by executing "mvn package -Dcontext.depot=\$DEPOT".

## 6. Using existing sources

This example will show how to use existing module sources in a library with a maven build process. Note: if you already have a ProjectBuilder object used to build your library jar, you can skip the ProjectBuilder Register command section. Just follow the Archetype Create steps, and configure the maven-ctl-plugin section of your pom.xml to use the correct object.

### 6.1. 1. ProjectBuilder Register

First of all, make sure your sources exist in a directory named the same as the the maven artifactId you are about to generate. For example, if you will call your library "mysystem", put the sources under a directory named "mysystem", as so:

```
mysystem/src/modules/<moduleName>/type.xml
```

Create a ProjectBuilder object to manage the library:

```
cd $NAME
ad -p $DEPOT -t ProjectBuilder -o $NAME -c Register -- -basedir `pwd`/src
-installroot `pwd`/target -install
```

The -basedir is important, since it specifies which directory contains the "modules" directory. Make sure this is correct. You can use any name for "-o" (object name), however it is convenient if the object name is the same as the maven artifactId.

### 6.2. 2. Archetype Create

Change to the directory that contains your library project.

```
cd ..
```

Run the archetype:create command, and specify a groupId, artifactId, and version:

```
set GROUPID=com.mycompany
set VERSION=1.0-SNAPSHOT
mvn archetype:create -DarchetypeGroupId=com.controltier
-DarchetypeArtifactId=ctier-library-jar \
  -DarchetypeVersion=1.0-SNAPSHOT -DgroupId=$GROUPID -DartifactId=$NAME
-Dversion=$VERSION \
  -DremoteRepositories=http://open.controltier.org/repo
```

This will create "\$NAME/pom.xml".

The pom.xml will now need to be edited:

Replace the "\${context.depot}" with \$DEPOT defined above. (If necessary, replace the `<name>${project.artifactId}</name>` parameter in the configuration of the maven-ctl-plugin to correspond to \$NAME, if it is not equivalent to your maven project's artifactId.)

you can now execute the following to create the library jar file:

```
mvn package
```

## 7. Notes

### 7.1. Repository Access

To use the controltier repository, add a "pluginRepository" that has the URL <http://open.controltier.org/repo> to your settings.xml document (~/.m2/settings.xml).

```
<settings 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/settings-1.0.0.xsd">
  <profiles>
    <profile>
      <id>default</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <pluginRepositories>
        <pluginRepository>
          <id>opentier</id>
          <name>ControlTier Repository</name>
          <releases>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>fail</checksumPolicy>
          </releases>
          <snapshots>
            <enabled>true</enabled>
```

```
        <updatePolicy>daily</updatePolicy>
        <checksumPolicy>warn</checksumPolicy>
    </snapshots>
    <url>http://open.controltier.org/repo</url>
    <layout>default</layout>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>
</settings>
```

## 7.2. Installing the ctier-library-jar archetype

The first time you want to use the archetype, you must specify a remote repository to download it from, by adding "-DremoteRepositories=http://open.controltier.org/repo" to the "mvn archetype:create" command.