TomEE Maven Plugin

TomEE Maven Plugin is a set of goals for the development and to prepare to go in production:

- `tomee:build`
- `tomee:exec`
- `tomee:configtest`
- `tomee:debug`
- `tomee:deploy`
- `tomee:exec`
- `tomee:list`
- `tomee:run`
- `tomee:start`
- `tomee:stop`
- `tomee:undeploy`

Run

The most commonly used goal, it allows to start a tomee with applications. Here is its configuration:

Name	Default	Description
synchronization	-	a synchronization (see after the table)
synchronizations	-	list of synchronizations
reloadOnUpdate	-	should the application be redeployed when a synchronization is triggered
skipCurrentProject	false	should current project not be considered as a deployable even if its packaging is compatible (war typically)
tomeeVersion	auto, plugin one	which version of TomEE to use
tomeeGroupId	org.apache.tomee	TomEE artifact groupId
tomeeArtifactId	apache-tomee	TomEE artifact artifactId
tomeeType	zip	the type of the TomEE artifact , only zip supported at the moment
tomeeClassifier	webprofile	which flavor of TomEE to use (classifier)
tomeeShutdownPort	read from server.xml	the shutdown port

Name	Default	Description
tomeeShutdownAttempts	60	how many times to wait for startup/shutdown (waits 1s in between)
tomeeShutdownCommand	SHUTDOWN	the shutdown command
tomeeAjpPort	read from the pom	the AJP port if needed
tomeeHttpsPort	read from the pom	the HTTPS port if needed
args	-	command line arguments (system properties, javaagent, JVM options ...)
debug	-	start and wait for a remote debugger to connect
debugPort	5005	used when debug to change the default port
simpleLog	false	use one line logs
extractWars	false	explode wars before starting
stripWarVersion	true	remove the version from the war name
stripVersion	false	remove the version from the artifact name whatever it is (even jar)
webappResources	${project.basedir}/src/main/webapp	where web resources are
webappClasses and classes	${project.build.outputDirectory}	where artifact binaries are
catalinaBase	${project.build.directory}/apache-tomee	where to create the tomee instance
context	-	name of the current artifact (rename the war from the maven name to this one)
webappDir	webapps	path to webapps folder from tomee.base
appDir	apps	path to apps folder from tomee.base
libDir	lib	where is lib folder
mainDir	${project.basedir}/src/main	used in openejb mode to change default config of conf/lib/bin folders to openejb instead of tomee

Name	Default	Description
config	${project.basedir}/src/main/tomee/conf	a conf folder synchronized with TomEE one
bin	${project.basedir}/src/main/tomee/bin	a bin folder synchronized with TomEE one
lib	${project.basedir}/src/main/tomee/lib	a lib folder synchronized with TomEE one
systemVariables	-	a map of system properties
classpaths	-	a list of additional entries for the startup classpath
customizers	-	a list of customizers
jsCustomizers	-	a list of js customizers (js scripts)
groovyCustomizers	-	a list of groovy customizers (groovy scripts)
webappDefaultConfig	false	auto config war oriented
quickSession	true	session generation will use `Random` instead of `SecureRandom` (for dev)
forceReloadable	false	ensure TomEE supports reloading/redeployment
forceJspDevelopment	true	JSP will be auto-recompiled on changes
libs	-	dependencies to add in lib, see after this table for advanced usage
endorsedLibs	-	dependencies to add in endorsed, see after this table for advanced usage
javaagents	-	javaagents to add on the JVM, supports maven coordinates
persistJavaagents	false	should javaagent be saved or just use for this plugin run
webapps	-	additional applicatinos to deploy
warFile	${project.build.directory}/${project.build.finalName}.${project.packaging}	the war to deploy
workWarFile	${project.build.directory}/${project.build.finalName}"	the exploded war to deploy

Name	Default	Description
removeDefaultWebapps	true	should default webapps (ROOT, docs, ...) be deleted
deployOpenEjbApplication	false	should openejb internal application be deployed
removeTomeeWebapp	true	should tomee webapp (with EJBd adapter) be deployed
tomeeAlreadyInstalled	false	skip all the setup configuration
ejbRemote	true	should EJBd be activated
checkStarted	false	should the plugin check the server is up (useful when used with `pre-integration` phase
useConsole	true	wait for the end of the execution reading inputs from the console (like `quit` command)
useOpenEJB	false	use openejb-standalone instead of tomee
inlinedServerXml	-	a server.xml content in pom.xml directly
inlinedTomEEXml	-	a tomee.xml content in pom.xml directly
overrideOnUnzip	true	if when unzipping tomee a file is already there should it be overriden
skipRootFolderOnUnzip	true	ignore root folder of the zip
keystore	-	path to keystore for HTTPS connector

Synchronization are blocks defining a source and target folder and both are synchronized. It typically copy `src/main/webapp` resources in `target/apache-tomee/webapps/myapp/`.

Customizers

Customizers are java classes loadable by the plugin and with a main or implementing `Runnable` and taking optionally as constructor parameter a `File` representing `tomee.base` or no arguments.

They are executed when creating the TomEE instance.

There are two scripting flavors of that: js and groovy. Both will have some contextual variables:

- catalinaBase: tomee base path
- resolver: a maven resolver to get a dependency using maven. For instance: `resolver.resolve('group', 'artfact', 'version', 'type')`

Dependencies (libs)

The format can be:

- a maven dependency:

```
groupId:artifactId:version
```

- a zip dependency and extracted in lib folder:

```
unzip:groupId:artifactId:version
```

- a matching prefix to remove:

```
remove:prefix
```

Example

```
<plugin>
  <groupId>org.apache.tomee.maven</groupId>
  <artifactId>tomee-maven-plugin</artifactId>
  <version>${tomee7.version}</version>
  <configuration>
    <tomeeClassifier>plus</tomeeClassifier>
    <debug>false</debug>
    <debugPort>5005</debugPort>
    <args>-Dfoo=bar</args>
    <config>${project.basedir}/src/test/tomee/conf</config>
    <libs>
      <lib>mysql:mysql-connector-java:5.1.20</lib>
    </libs>
    <webapps>
       <webapp>org.superbiz:myapp:4.3?name=ROOT</webapp>
       <webapp>org.superbiz:api:1.1</webapp>
    </webapps>
    <apps>
        <app>org.superbiz:mybugapp:3.2:ear</app>
    </apps>
    <libs>
        <lib>mysql:mysql-connector-java:5.1.21</lib>
        <lib>unzip:org.superbiz:hibernate-bundle:4.1.0.Final:zip</lib>
        <lib>remove:openjpa-</lib>
    </libs>
  </configuration>
</plugin>
```

Build

Excepted synchronization, build plugin inherit from `run` Mojo its configuration. It just adds the following:

Name	Default	Description
formats	-	map of configuration, keys are format (zip, tar.gz) and value the target location
zip	true	create a zip from the configured instance
attach	true	attach created artifacts
skipArchiveRootFolder	false	don't add a root folder in the zip

Tomcat like goals

`configtest`, `start` and `stop` just execute these commands on the server (like on `catalina.sh`).