Usage

Introduction

The Deploy Plugin has two basic functions. In most project builds, the `deploy` phase of the build lifecycle is implemented using the `deploy:deploy` mojo. Also, artifacts which are not built using Maven can be added to any remote repository using the `deploy:deploy-file` mojo.

The `deploy:deploy` Mojo

In most cases, this mojo is invoked when you call the `deploy` phase of the default build lifecycle.

To enable this mojo to function, you must include a valid `<distributionManagement/>` section POM, which at the minimum provides a `<repository/>` defining the remote repository location for your artifact. To separate snapshot artifacts from release artifacts, you can also specify a `<snapshotRepository/>` location. Finally, to deploy a project website, you must specify a `<site/>` section here as well. It's also important to note that this section can be inherited, allowing you to specify the deployment location one time for a set of related projects.

If your repository is secured, you may also want to configure your `settings.xml` file to define corresponding `<server/>` entries which provides authentication information. Server entries are matched to the different parts of the distributionManagement using their `<id/>` elements. For example, your project may have a distributionManagement section similar to the following:

```
1. [...]
2.   <distributionManagement>
3.     <repository>
4.       <id>internal.repo</id>
5.       <name>MyCo Internal Repository</name>
6.       <url>Host to Company Repository</url>
7.     </repository>
8.   </distributionManagement>
9. [...]
```

In this case, you can specify a server definition in your `settings.xml` to provide authentication information for both of these repositories at once. Your server section might look like this:

```
1. [...]
2.     <server>
3.       <id>internal.repo</id>
4.       <username>maven</username>
5.       <password>foobar</password>
6.     </server>
7. [...]
```

Please see the article about Password Encryption (https://maven.apache.org/guides/mini/guide-encryption.html) for instructions on how to avoid clear text passwords in the `settings.xml`.

Once you've configured your repository deployment information correctly deploying your project's artifact will only require invocation of the `deploy` phase of the build:

```
mvn deploy
```

The `deploy:deploy-file` Mojo

The `deploy:deploy-file` mojo is used primarily for deploying artifacts, which were not built by Maven. The project's development team may or may not provide a POM for the artifact, and in some cases you may want to deploy the artifact to an internal remote repository. The deploy-file mojo provides functionality covering all of these use cases, and offers a wide range of configurability for generating a POM on-the-fly. Additionally, you can specify what layout your repository uses. The full usage statement of the deploy-file mojo can be described as:

```
mvn deploy:deploy-file -Durl=file://C:\m2-repo \
                       -DrepositoryId=some.id \
                       -Dfile=your-artifact-1.0.jar \
                       [-DpomFile=your-pom.xml] \
                       [-DgroupId=org.some.group] \
                       [-DartifactId=your-artifact] \
                       [-Dversion=1.0] \
                       [-Dpackaging=jar] \
                       [-Dclassifier=test] \
                       [-DgeneratePom=true] \
                       [-DgeneratePom.description="My Project Description"] \
                       [-DrepositoryLayout=legacy]
```

If the following required information is not specified in some way, the goal will fail:

- the artifact file to deploy
- the group, artifact, version and packaging of the file to deploy. These can be taken from the specified pomFile, and overriden or specified using the command line. When the pomFile contains a *parent* section, the parent's groupId can be considered if the groupId is not specified further for the current project or on the command line.
- the repository information: the url to deploy to and the repositoryId mapping to a server section in the settings.xml file. If you don't specify a repositoryId, Maven will try to extract authentication information using the id `'remote-repository'`.
