Maven Repository 설정 방법

Sonya Project

2008-05-28

Sonya Project

YoungGue Bae

Copyright © 2008 Sonya. All rights reserved

목차

[사내 Maven Repository의 필요성 사내 Maven Repository의 필요성 3](#_Toc199768853)

[Artifactory를 이용한 사내 Maven Repository 설치하기 5](#_Toc199768854)

[Artifactory 사용하기 7](#_Toc199768855)

[Artifactory Config 설정 7](#_Toc199768856)

[Maven Setting 13](#_Toc199768857)

[POM 설정 16](#_Toc199768858)

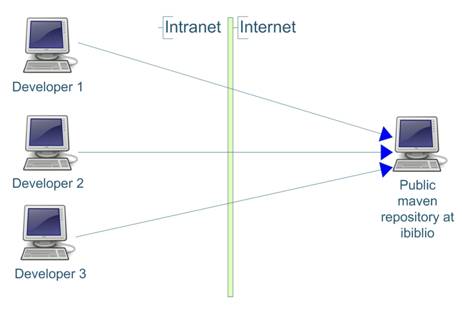
[Artifactory에 라이브러리 배포하기 18](#_Toc199768859)

# 사내 Maven Repository의 필요성 사내 Maven Repository의 필요성

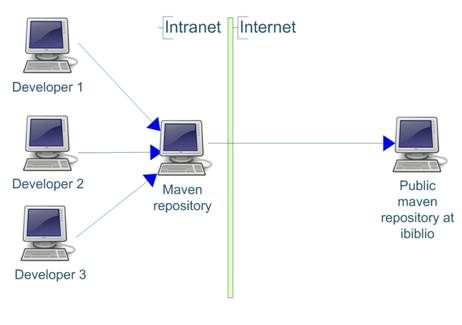
Maven을 기반으로한 라이브러리 관리 방식은 Maven Repository에서 라이브러리를 관리하고 이를 Local Repository에 다운로드하는 방식이다. Maven Repository에서 관리하고 있는 라이브러리의 수는 너무 많아서 그 수를 헤아리기조차 힘들다. 그러나 Maven Repository가 우리가 필요로 하는 모든 라이브러리를 모두 관리할 수는 없는 것이다.

예를 들어 사내에서 개발하여 배포되는 라이브러리는 범용적으로 사용되는 라이브러리가 아니기 때문에 Maven Repository에 관리하기 힘들다. 이외에도 사내 Maven Repository를 가질 경우 많은 잇점이 있다.

사내 Repository를 가질 때와 그렇지 않을 때의 개발 환경을 살펴보면 다음과 같다.



위 그림은 사내 Maven Repository를 가지지 않을 때의 개발 환경이다.



위 그림은 사내 Maven Repository를 가질 때의 개발 환경이다.

# Artifactory를 이용한 사내 Maven Repository 설치하기

Artifactory는 오픈 소스로 사내 Maven Repository로 설치해서 사용하기 좋다. Setting Up a Maven Repository문서를 보면 사내 Maven Repository로 사용할 수 있는 툴들을 비교한 자료를 볼 수 있다.

Artifactory 설치에 대한 이 강좌의 모든 내용 또한 Setting Up a Maven Repository문서를 참고해서 작성했다.

* http://www.jfrog.org/sites/artifactory/latest/ 에서 Artifactory 최신 버전을 다운 받는다.
* JDK 5.x 이상을 설치한다.
* Tomcat 5.5 이상을 설치한다.

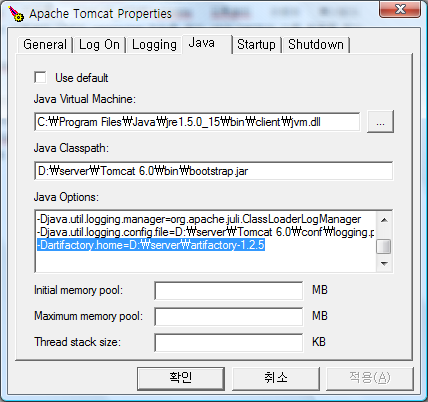
Artifactory는 웹 애플리케이션으로 구현되어 있다. 다운 받은 Artifactory 최신 버전의 압축을 풀어보면 webapps 디렉토리 아래에 artifactory.war 파일을 찾을 수 있다.

이 artifactory.war 파일을 TOMCAT\_HOME/webapps 디렉토리 아래에 복사한다. TOMCAT\_HOME/bin 디렉토리 아래에 catalina.bat 파일을 열어 JAVA\_OPTS에 다음 항목을 추가한다.

JAVA\_OPTS = -Dartifactory.home= D:\server\artifactory-1.2.5

TOMCAT에 추가하지 않고 시스템 환경 변수에 추가해도 된다. artifactory.home에 설정해야 하는 디렉토리는 Artifactory 최신 버전의 압축을 푼 디렉토리이다. Artifactory는 Artifactory 설치 디렉토리 하위의 data 디렉토리를 이용하기 때문에 위 항목을 반드시 추가해야 한다.

Artifactory 설치는 위 과정만 거치면 된다. Artifactory는 간단하게 설치할 수 있다.



# Artifactory 사용하기

## Artifactory Config 설정

라이브러리를 분리해서 관리하기 위하여 Artifactory를 다음과 같이 설정하는 것이 가능하다. <ARTIFACTORY\_INSTALLATION\_FOLDER>/etc/artifactory.config.xml 파일을 만든 후 다음과 같이 설정할 수 있다.

|  |
| --- |
| <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://artifactory.jfrog.org/xsd/1.1.0"  xsi:schemaLocation="http://artifactory.jfrog.org/xsd/1.1.0  http://www.jfrog.org/xsd/artifactory-v1\_1\_0.xsd">  <!-- Backup every 12 hours -->  <!--<backupCronExp>0 0 /12 \* \* ?</backupCronExp>-->  <localRepositories>  <localRepository>  <key>private-internal-repository</key>  <description>Private internal repository</description>  <handleReleases>true</handleReleases>  <handleSnapshots>true</handleSnapshots>  </localRepository>  <localRepository>  <key>third-party-repository</key>  <description>3rd party jars added manually</description>  <handleReleases>true</handleReleases>  <handleSnapshots>false</handleSnapshots>  </localRepository>  </localRepositories>  <remoteRepositories>  <remoteRepository>  <key>repo1</key>  <handleReleases>true</handleReleases>  <handleSnapshots>false</handleSnapshots>  <excludesPattern>  org/artifactory/\*\*,org/jfrog/\*\*  </excludesPattern>  <url>http://repo1.maven.org/maven2</url>  </remoteRepository>  <remoteRepository>  <key>java.net.maven1</key>  <description>java.net</description>  <handleReleases>true</handleReleases>  <handleSnapshots>true</handleSnapshots>  <type>maven1</type> <!-- To proxy Maven 1 repositories, simply add the <type>maven1</type> element to a remote repository definition (the default is type is maven2, and does not need to be explicitly specified). -->  <url>http://download.java.net/maven/1</url>  </remoteRepository>  <remoteRepository>  <key>appfuse</key>  <description>appfuse</description>  <handleReleases>true</handleReleases>  <handleSnapshots>true</handleSnapshots>  <url>http://static.appfuse.org/repository</url>  </remoteRepository>  <remoteRepository>  <key>jboss</key>  <description>jboss</description>  <handleReleases>true</handleReleases>  <handleSnapshots>false</handleSnapshots>  <url>http://repository.jboss.com/maven2</url>  </remoteRepository>  </remoteRepositories>  </config> |

위 설정은 라이브러리를 3가지 용도로 나뉘어 관리하겠다는 것이다. 중앙 Maven Repository의 Mirror 역할을 하게 될 Repository(repo1, java.net.maven1, appfuse), 3rd party 라이브러리를 관리하기 위한 Repository(third-party-repository), 사내에서 개발되는 라이브러리를 관리하기 위한 Repository(private-internal-repository)로 나누고 있다. 물론 사내 정책에 따라 더 많은 Repository를 관리하는 것 또한 가능하다.

<주의>

**Proxying Maven 1 repositories**

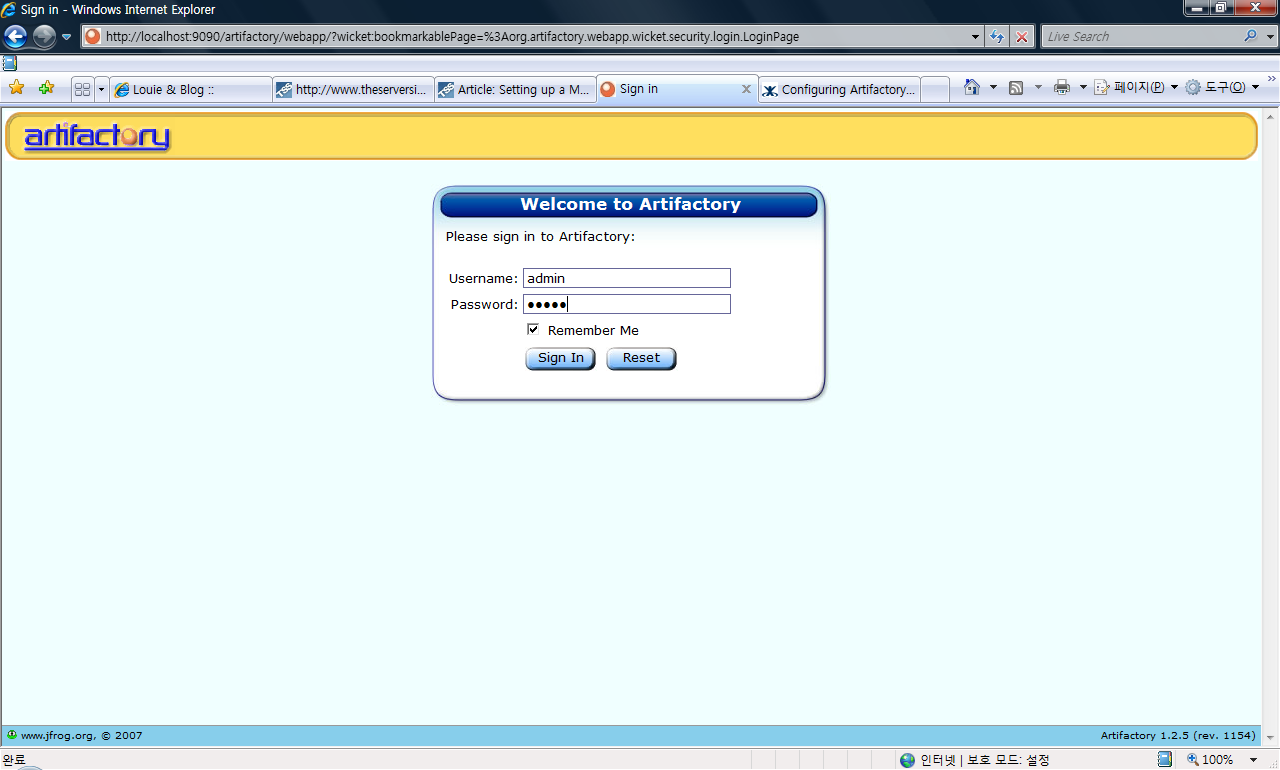
To proxy Maven 1 repositories, simply add the <type>maven1</type> element to a remote repository definition (the default is type is maven2, and does not need to be explicitly specified).

For example, to proxy the java.net Maven 1 repo, include the following in your artifactory.config.xml:

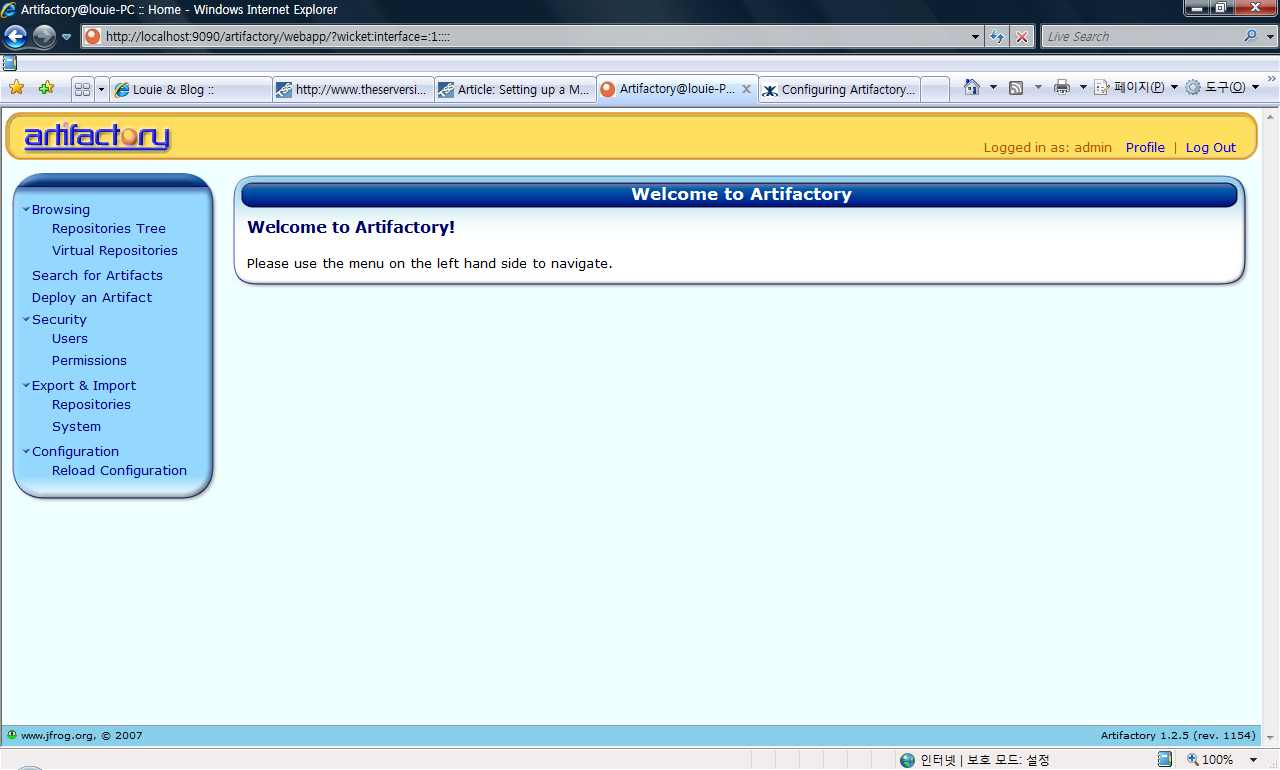
|  |
| --- |
| <remoteRepository>  <key>java.net.maven1</key>  <description>java.net</description>  <handleReleases>true</handleReleases>  <handleSnapshots>true</handleSnapshots>  <type>maven1</type>  <url>http://download.java.net/maven/1</url>  </remoteRepository> |

위와 같이 설정한 다음 Tomcat을 시작한 다음 Artifactory에 접근할 수 있다.

http://localhost:8080/artifactory로 접근하면 Artifactory 로그인 화면에 접근할 수 있다.

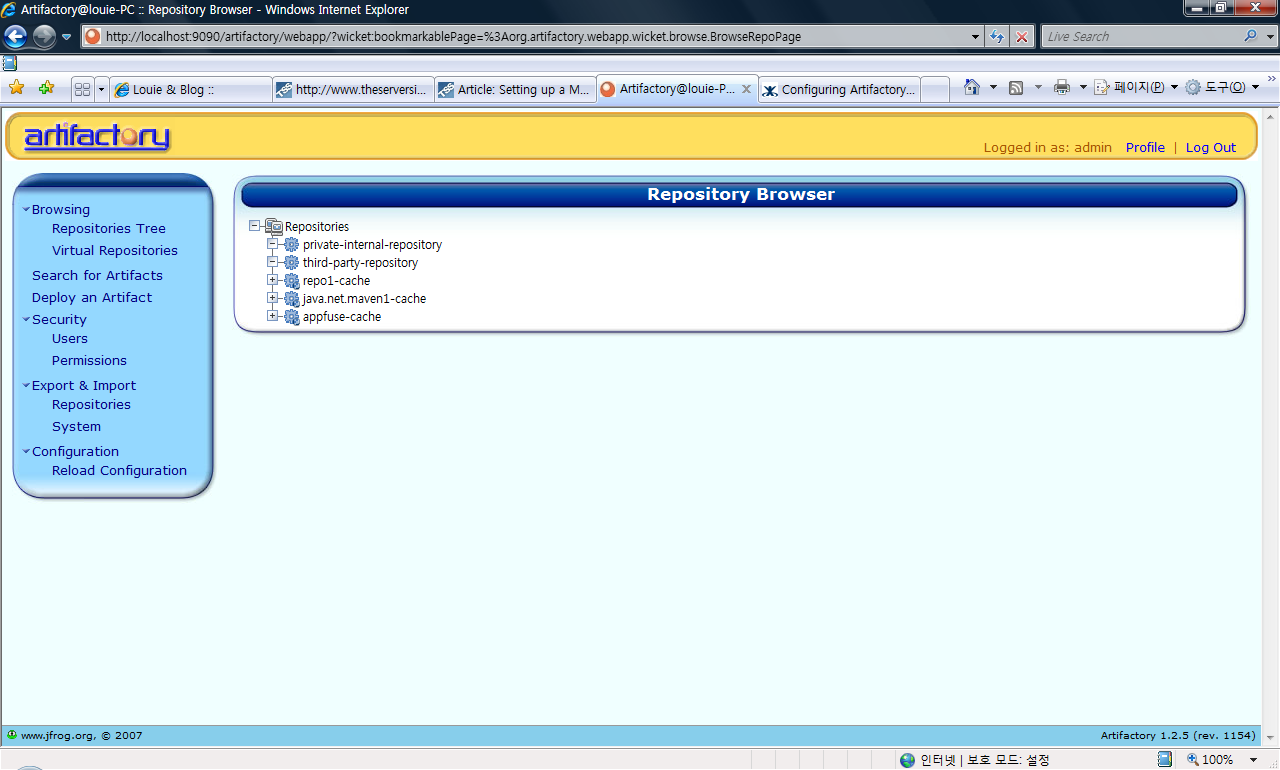


Artifactory의 디폴트 사용자는 UserName : admin, Password : password 를 이용하여 로그인할 수 있다.



Artifactory 메인 페이지는 위와 같다.

좌측 메뉴에서 Browse the repository를 선택하면 다음 화면과 같다. 앞에서 설정한 Repository가 생성되어 있는 것을 볼 수 있다.



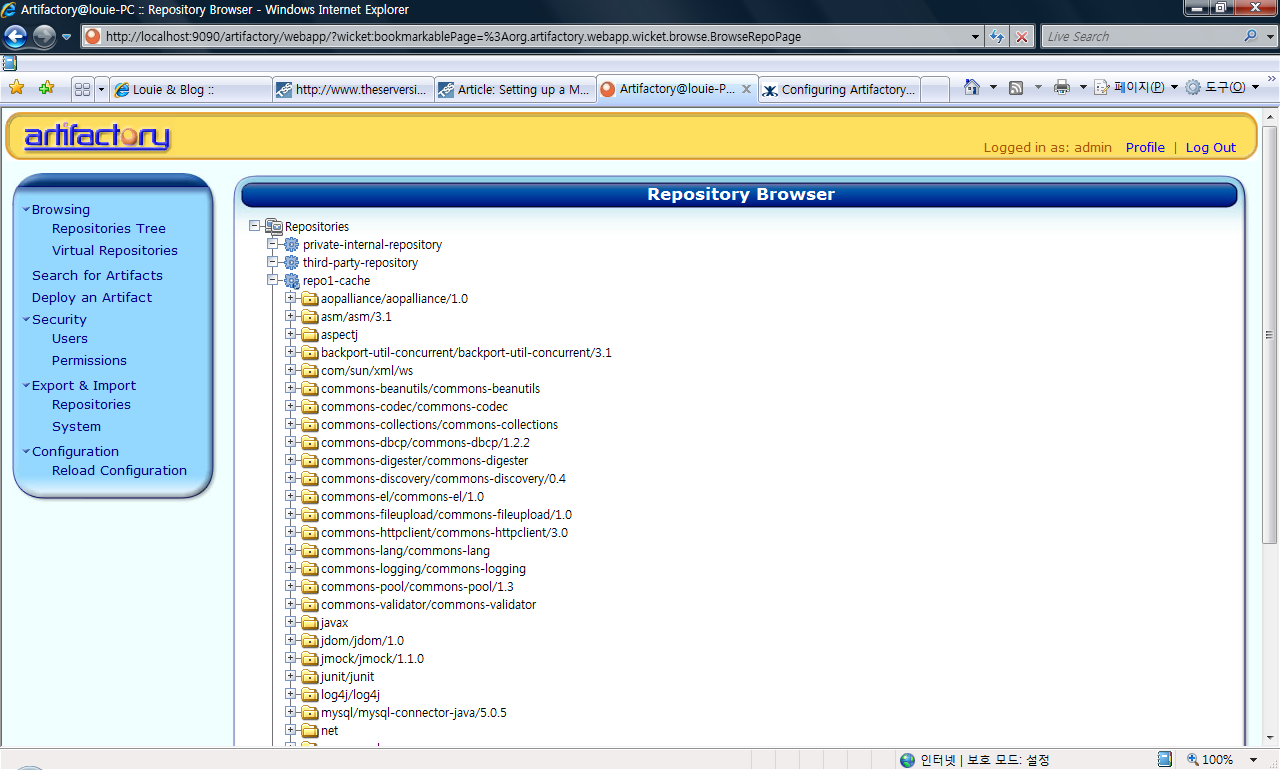
## Maven Setting

다음으로 설정할 부분은 Maven 빌드툴이 새로 설치한 Artifactory를 사용할 수 있도록 설정하는 것이다. 사내 Maven Repository를 사용하도록 설정하는 방법은 다음과 같다.

USER\_HOME/.m2/settings.xml 파일을 다음과 같이 설정한다.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <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">  <servers>  <server>  <id>organisation-internal</id>  <username>admin</username>  <password>admin</password>  </server>  </servers>  <profiles>  <profile>  <id>dev</id>  <properties>  <tomcat6x.home>D:\server\Tomcat 6.0</tomcat6x.home>  </properties>  <repositories>  <repository>  <id>central</id>  <url>  http://localhost:9090/artifactory/repo  </url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  <repository>  <id>snapshots</id>  <url>  http://localhost:9090/artifactory/repo  </url>  <releases>  <enabled>false</enabled>  </releases>  </repository>  </repositories>  <pluginRepositories>  <pluginRepository>  <id>central</id>  <url>  http://localhost:9090/artifactory/repo  </url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </pluginRepository>  <pluginRepository>  <id>snapshots</id>  <url>  http://localhost:9090/artifactory/repo  </url>  <releases>  <enabled>false</enabled>  </releases>  </pluginRepository>  </pluginRepositories>  </profile>  </profiles>  </settings> |

위와 같이 설정한 다음 Maven 빌드툴을 실행해보자. Maven 중앙 Repository에서 다운되는 모든 라이브러리가 Artifactory Repository에 다음과 같이 저장된다.



# POM 설정

The repository settings can also be done in the project pom.xml. A simple ‘pom.xml’ is shown below:

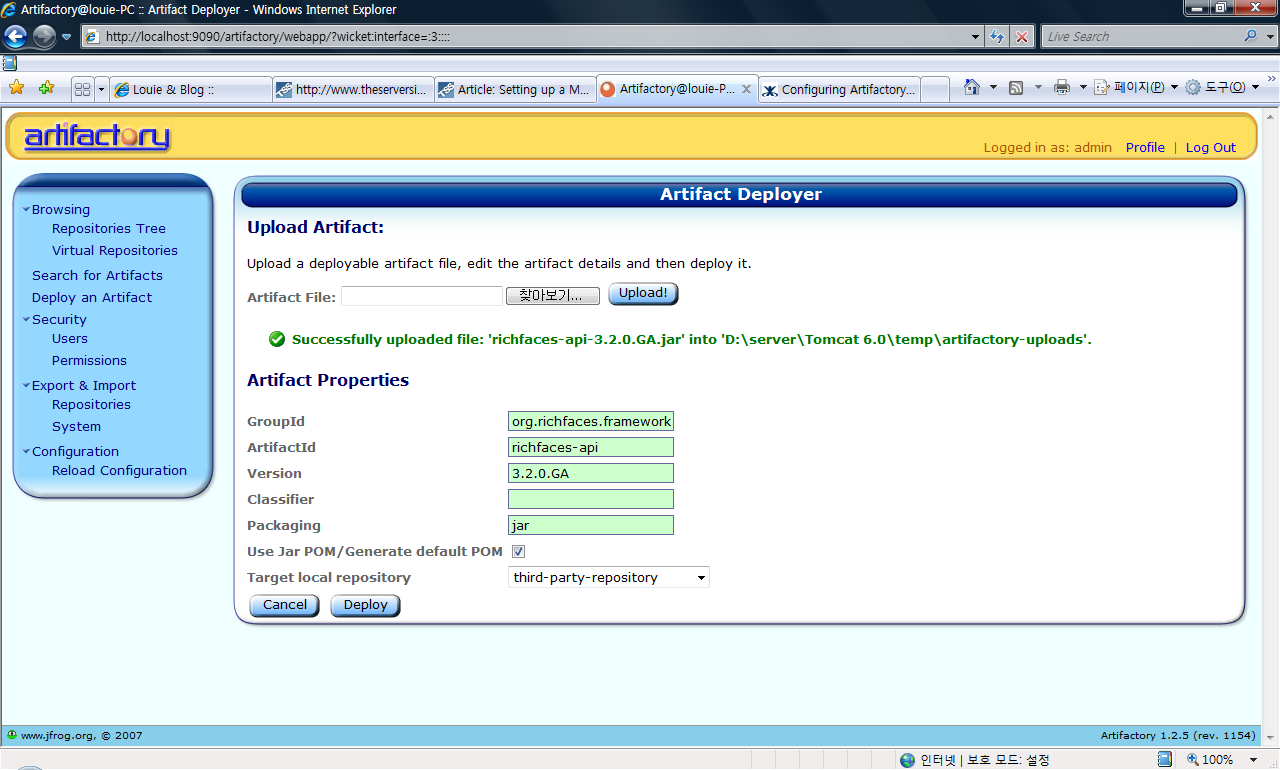
|  |
| --- |
| <repositories>  <repository>  <id>central</id>  <url>http://localhost:9090/artifactory/repo</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  <repository>  <id>snapshots</id>  <url>http://localhost:9090/artifactory/repo</url>  <releases>  <enabled>false</enabled>  </releases>  </repository>  </repositories>  <!--  <pluginRepositories>  <pluginRepository>  <id>central</id>  <url>http://localhost:9090/artifactory/repo</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </pluginRepository>  <pluginRepository>  <id>snapshots</id>  <url>http://localhost:9090/artifactory/repo</url>  <releases>  <enabled>false</enabled>  </releases>  </pluginRepository>  </pluginRepositories>  --> |

# Artifactory에 라이브러리 배포하기

개발하고 있는 라이브러리를 사내 Maven Repository에 배포하여 관리할 필요가 있다. 프로젝트간의 Dependency를 관리하기 위하여 개발 라이브러리를 사내 Maven Repository에 배포하고 사용할 수 있도록 개발 환경을 설정하는 것이 중요하다.

Artifactory에 라이브러리를 배포하는 방법은 Artifactory 관리툴을 이용하는 방법과 Maven 빌드툴을 이용하는 방법이 있다.

Artifactory 관리툴을 이용하는 방법은 간단하다. Artifactory 관리툴의 Deploy an artifact를 이용하여 새로운 라이브러리를 배포할 수 있다. 배포할 라이브러리를 업로드하면 다음과 같이 라이브러리에 대한 정보를 입력하는 것이 가능하다.



이 방법은 3rd party 라이브러리를 관리할 때 유용하게 사용할 수 있는 방법이다. 개발하고 있는 라이브러리를 매번 이 방식으로 배포하기에는 적합하지 않다. 개발을 진행하는 중에는 상당히 자주 배포할 필요가 있기 때문에 개발 환경에 바로 배포할 수 있어야 한다. Maven 빌드툴은 deploy goal을 통하여 사내 Repository에 배포하는 것이 가능하도록 지원한다.

로컬에 설치한 Artifactory에 개발 라이브러리를 배포하기 위해서 먼저 USER\_HOME/.m2/settings.xml 파일에 다음과 같이 Artifactory 인증에 필요한 정보를 추가해야 한다.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <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">  <servers>  <server>  <id>organisation-internal</id>  <username>admin</username>  <password>admin</password>  </server>  </servers>  <profiles>  </profiles>  </settings |

Artifactory에 로그인할 때 사용했던 username과 password를 이용하여 인증이 가능하도록 설정한다.

다음은 개발 프로젝트의 pom.xml 파일에 다음과 같이 배포할 서버를 지정해야 한다.

|  |
| --- |
| <distributionManagement>  <repository>  <id>organisation-internal</id>  <name>Sonya Repository</name>  <url>  http://localhost:9090/artifactory/private-internal-repository  </url>  </repository>  </distributionManagement> |

위와 같이 설정한 다음 "mvn deploy"를 실행하면 로컬에 설치한 Artifactory에 개발 라이브러리를 배포하는 것이 가능하다.

개발 라이브러리르 배포하면 다음과 같이 private-internal-repository에 배포되게 된다.