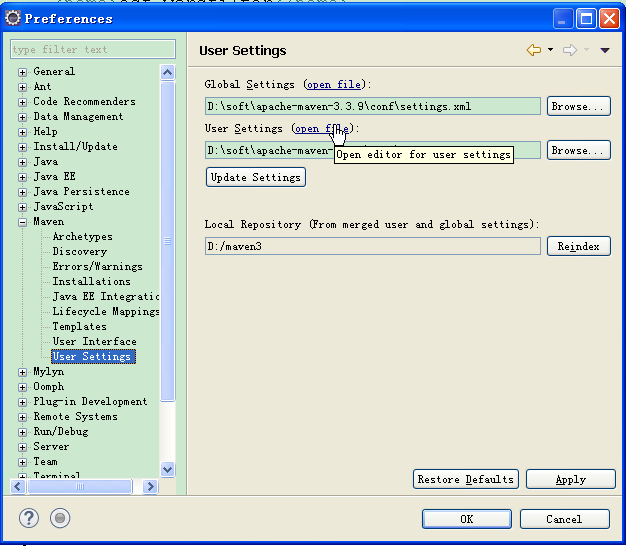
1. JDK安装
   1. 环境变量

|  |  |
| --- | --- |
| JAVA\_HOME | C:\Program Files\Java\jdk1.7.0\_21 |
| Path | .;% JAVA\_HOME%\bin |

1. Maven安装
   1. 环境变量

|  |  |
| --- | --- |
| MAVEN\_HOME | D:\soft\apache-maven-3.3.9 |
| Path | .;% MAVEN\_HOME %\bin |

1. Maven全局设置：settings.xml



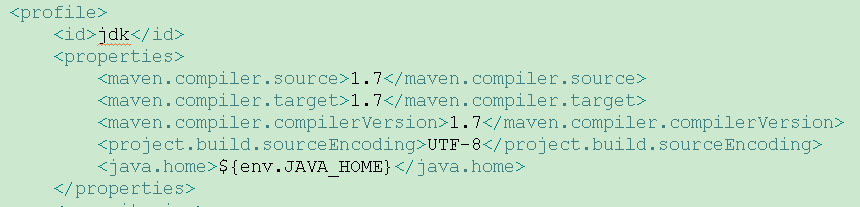
* 1. 属性

1. 引用Settings属性

${ settings.localRepository}

1. 引用系统环境变量

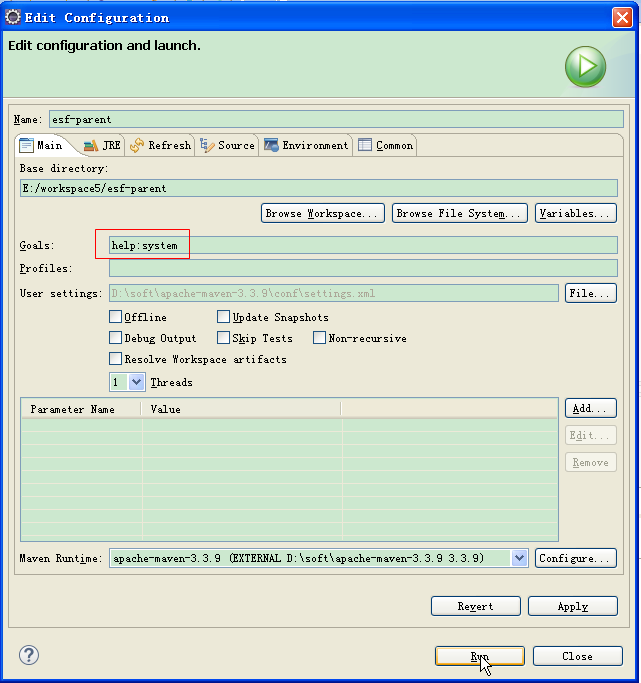
${env.JAVA\_HOME}

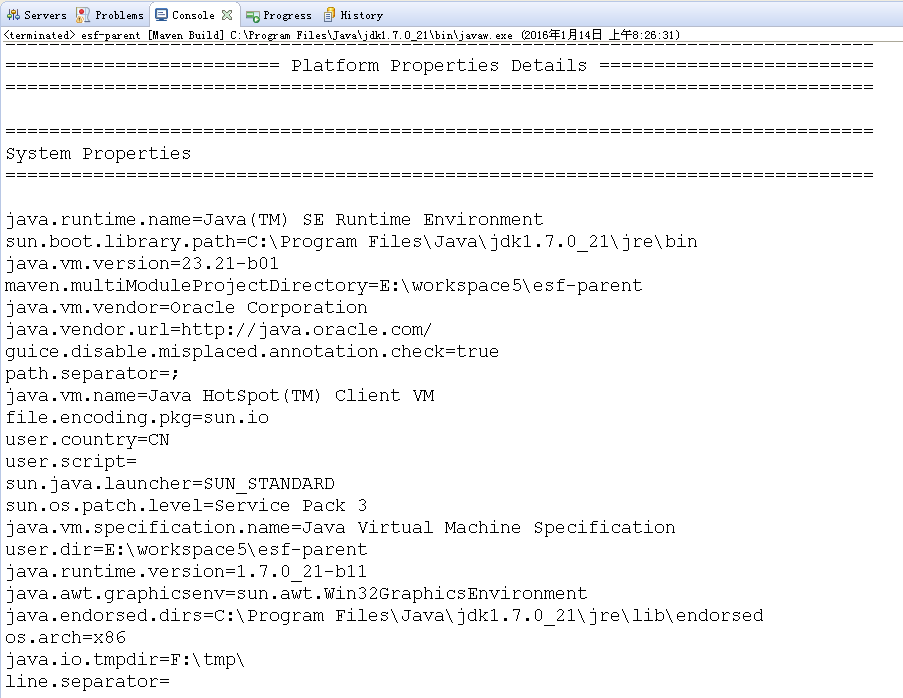


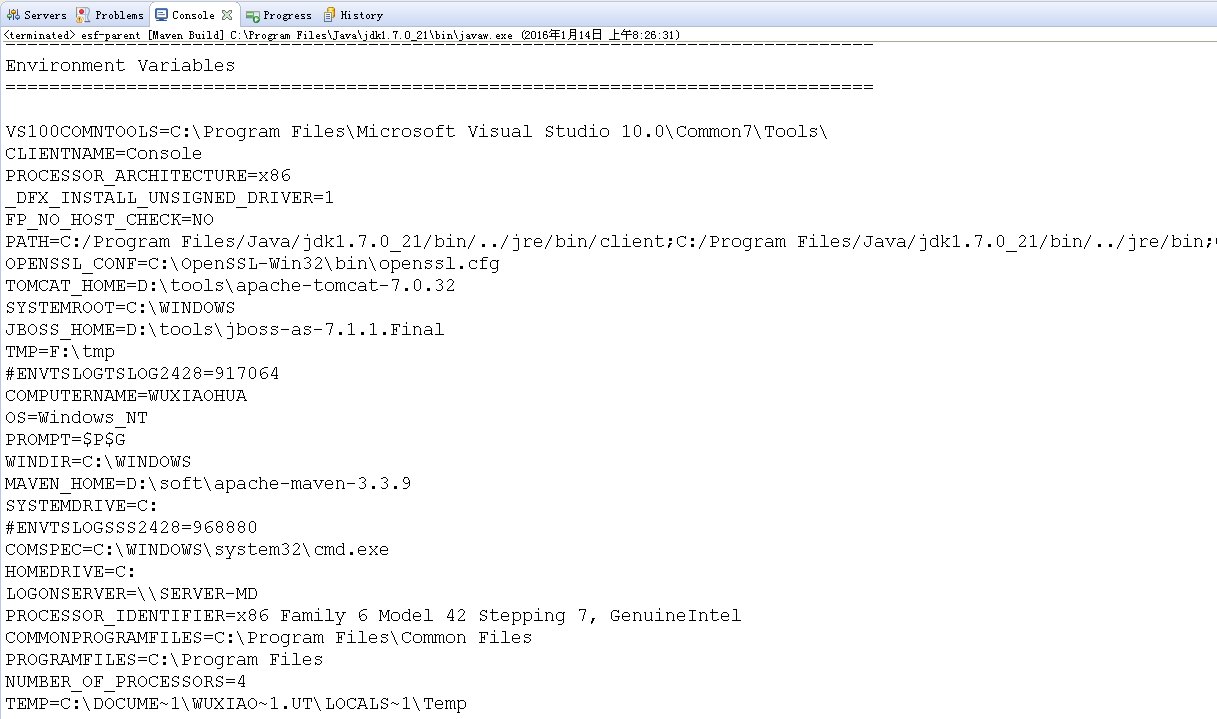
1. 引用系统属性

${ java.io.tmpdir }

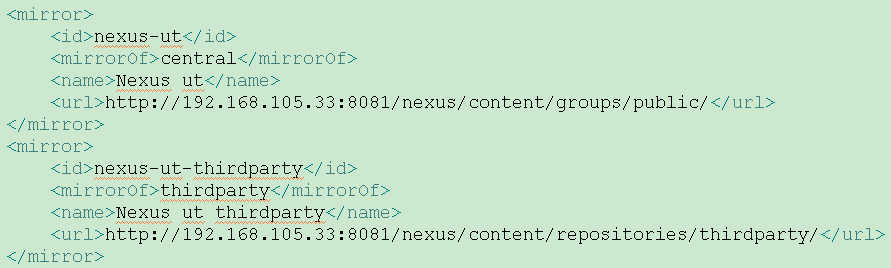
1. 查看系统环境变量或系统属性

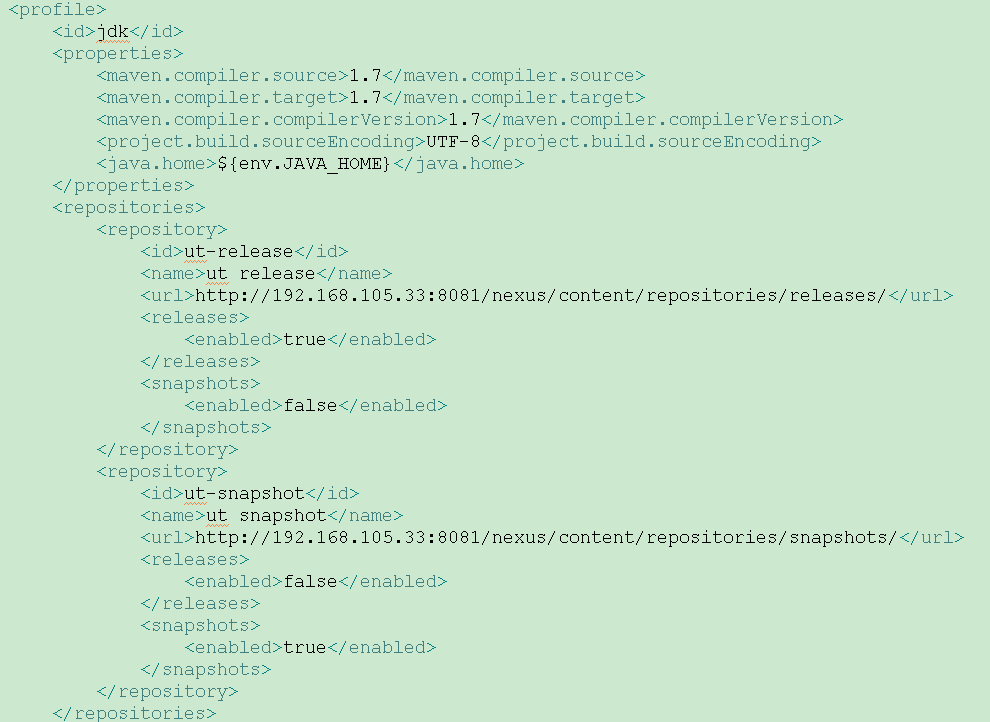


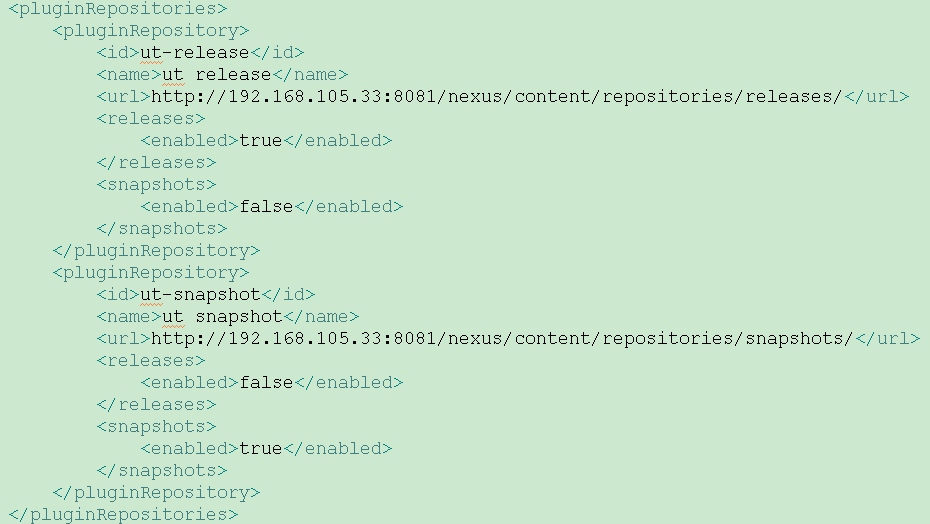




* 1. 镜像仓库

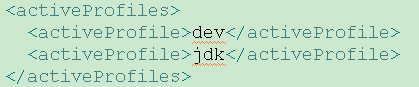






* 1. Profile



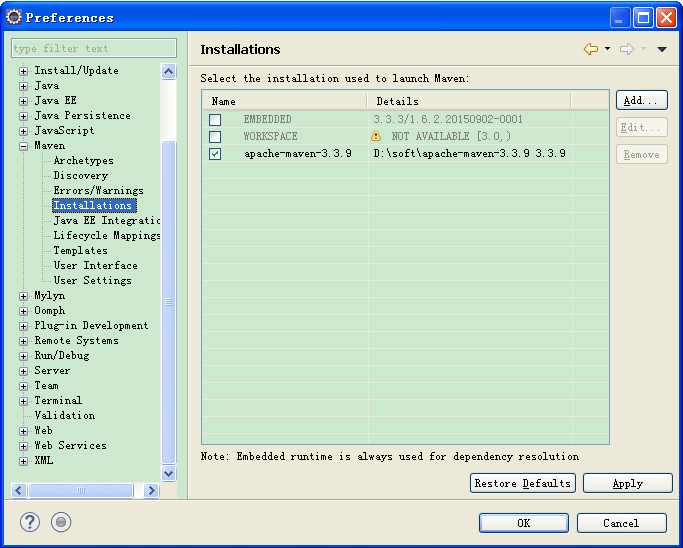


1. Eclipse
   1. 推荐版本

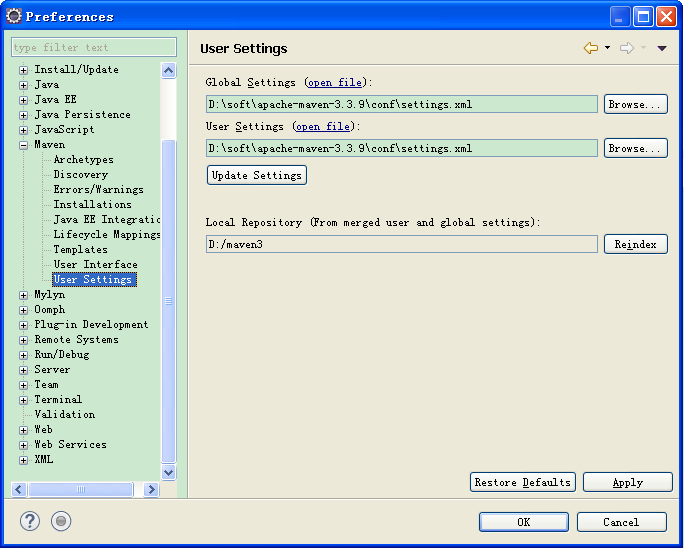
eclipse-jee-mars-1-win32.zip

eclipse-jee-mars-1-win32-x86\_64.zip

1. Eclipse Maven设置
   1. 使用外部Maven

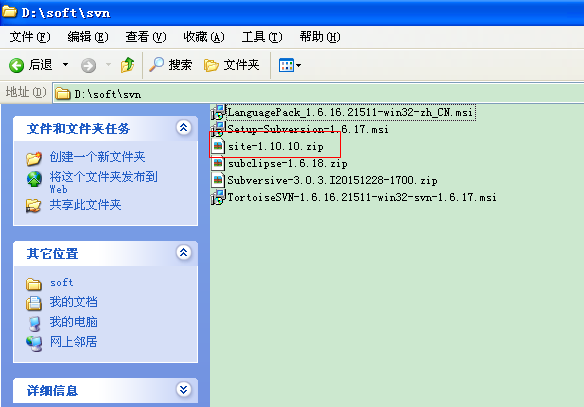


* 1. 设置Maven配置文件路径

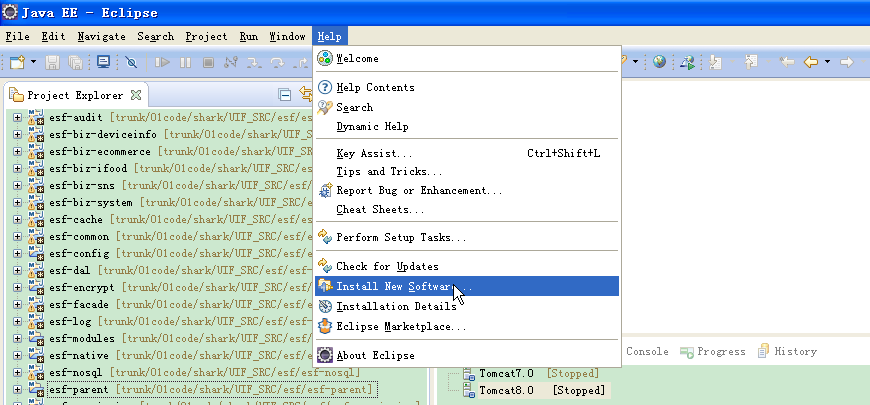


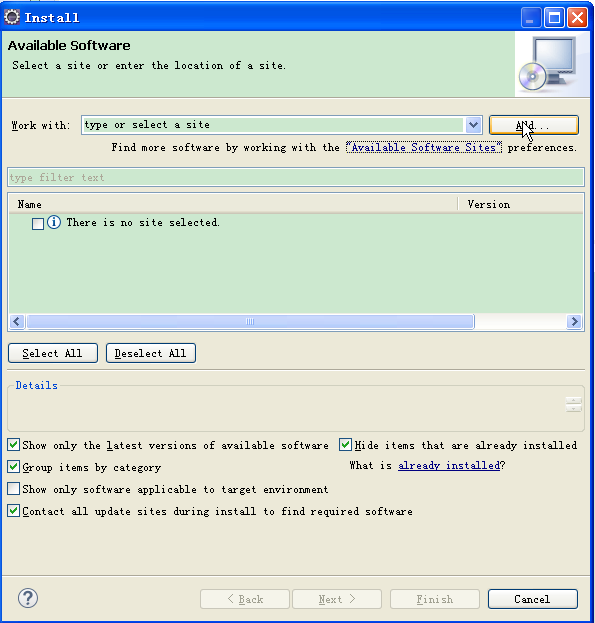
1. Eclipse SVN插件：Subclipse
   1. 版本

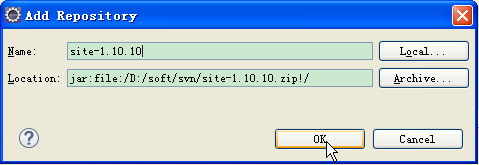
site-1.10.10.zip

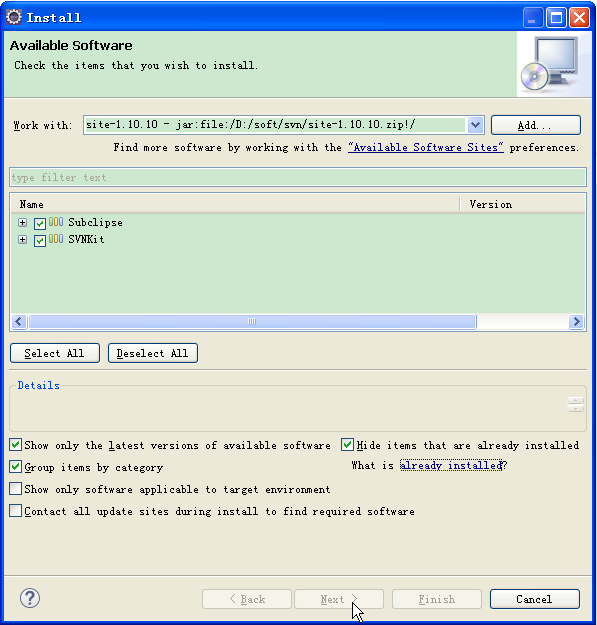


* 1. 安装离线包



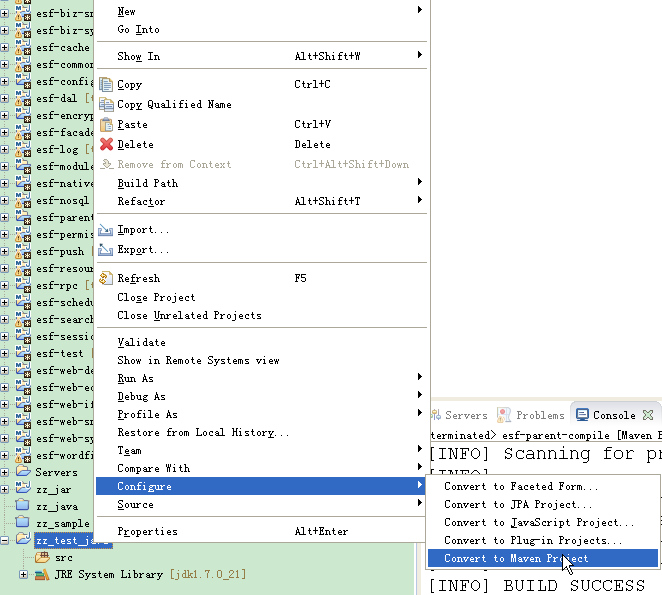


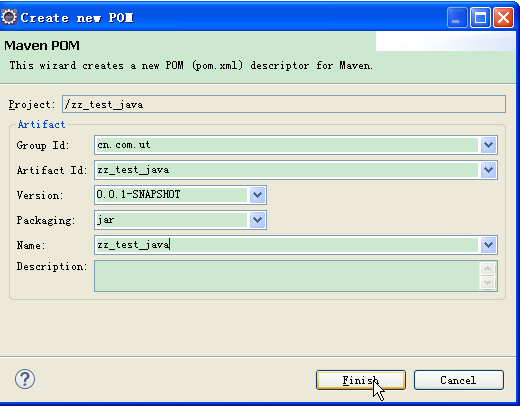


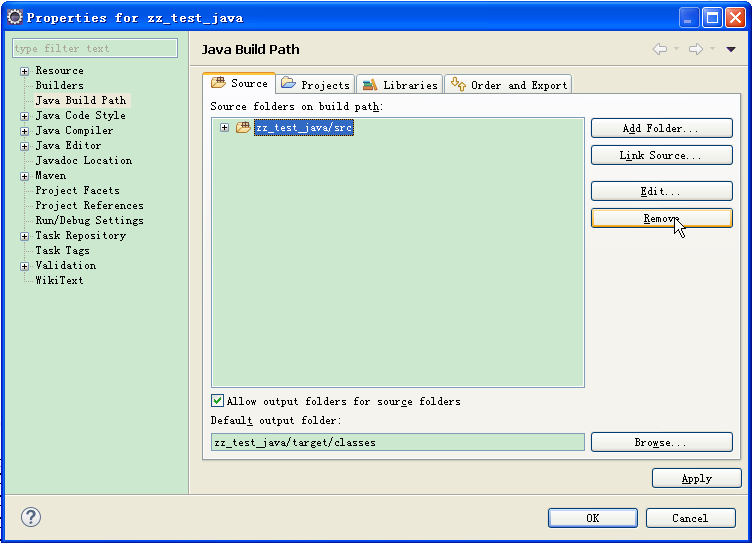




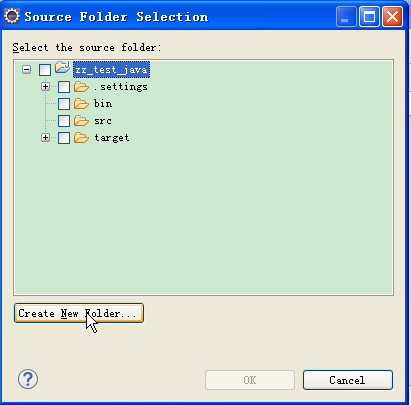
1. 迁移
   1. Java Project到Maven Project

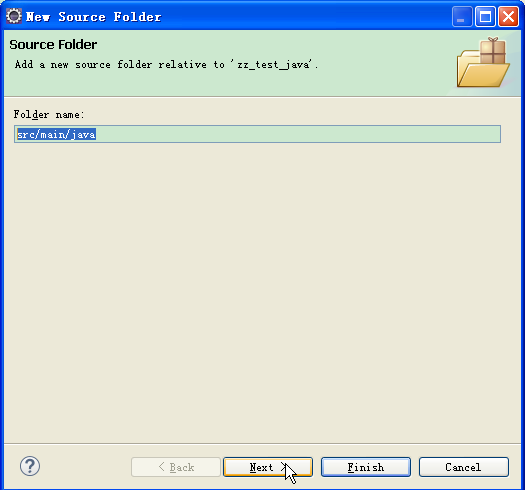


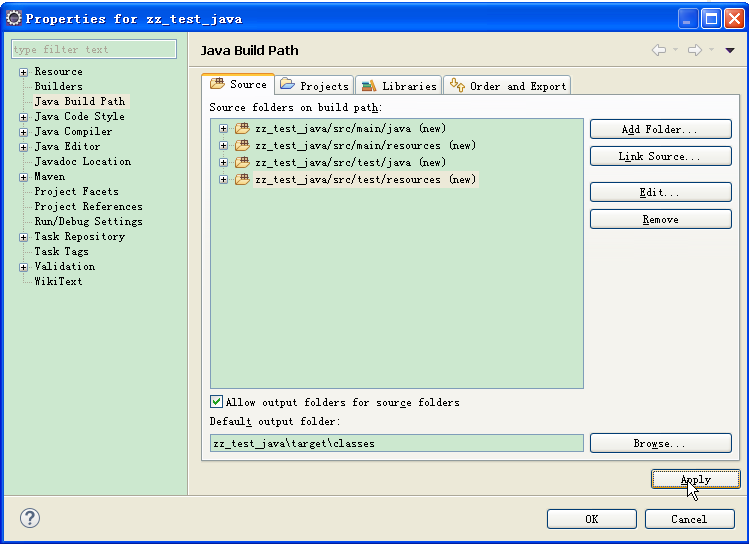




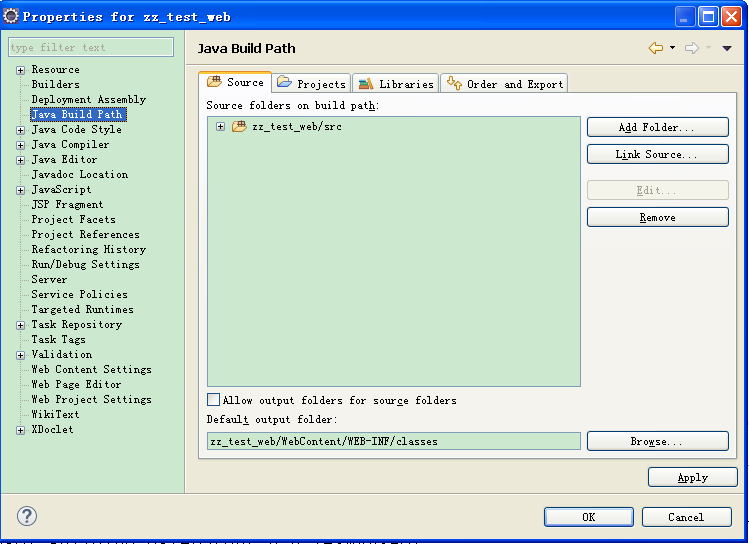


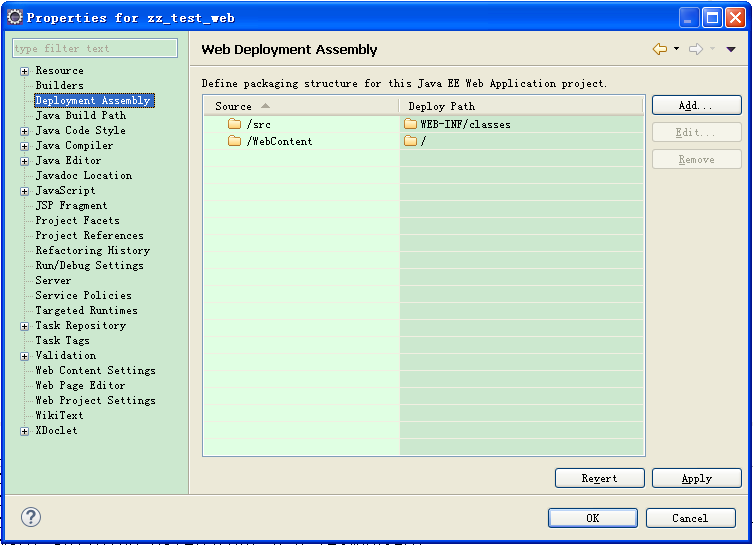


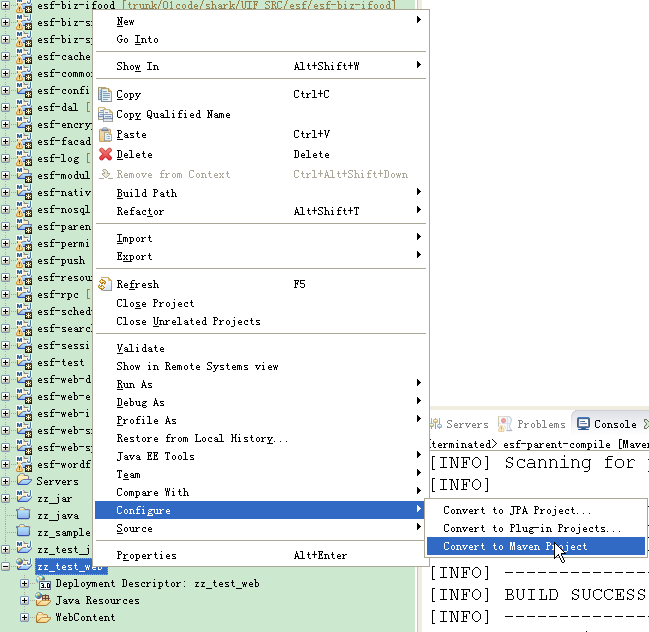


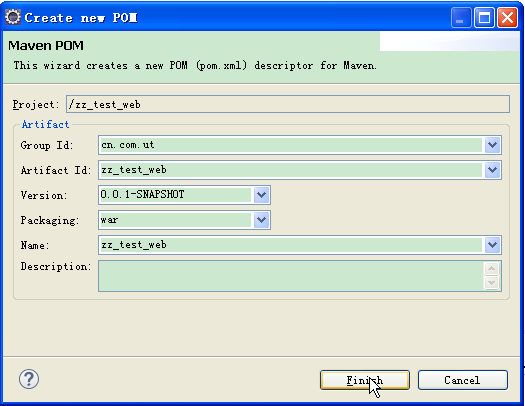


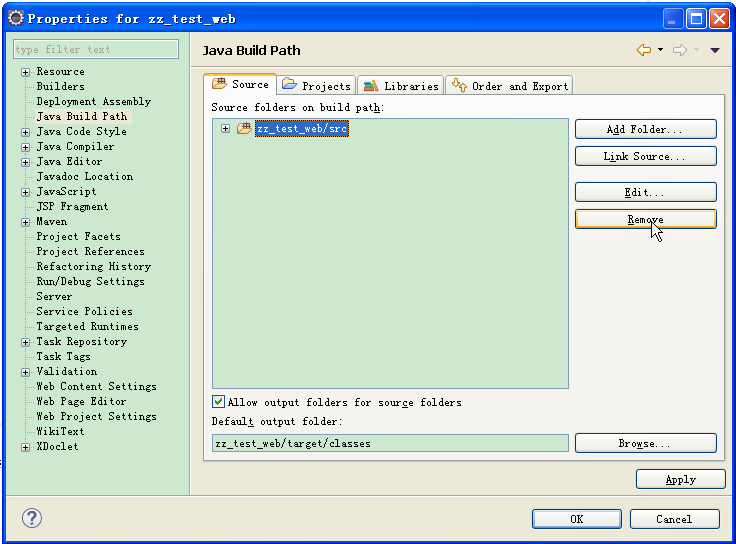
* 1. Web Project到Maven Project

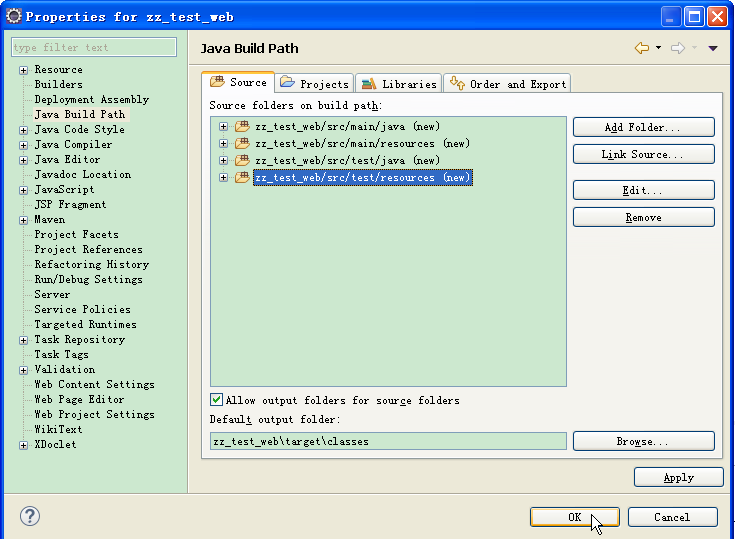


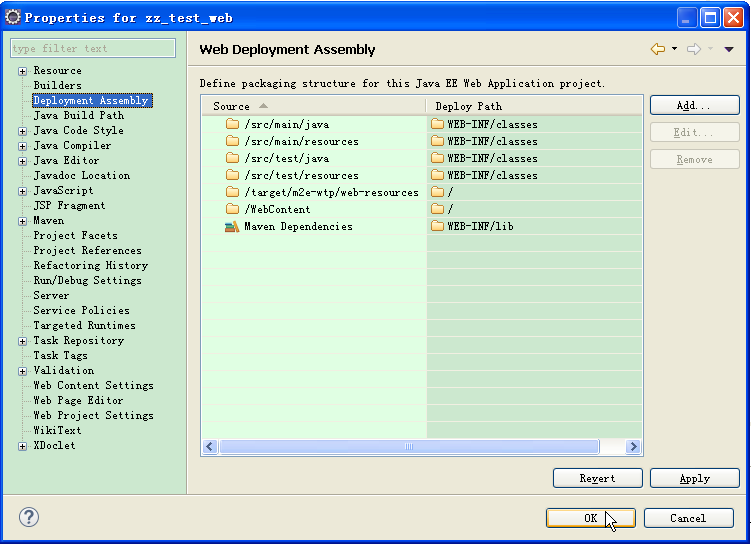






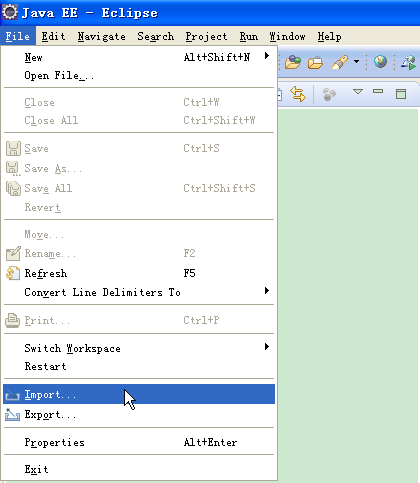


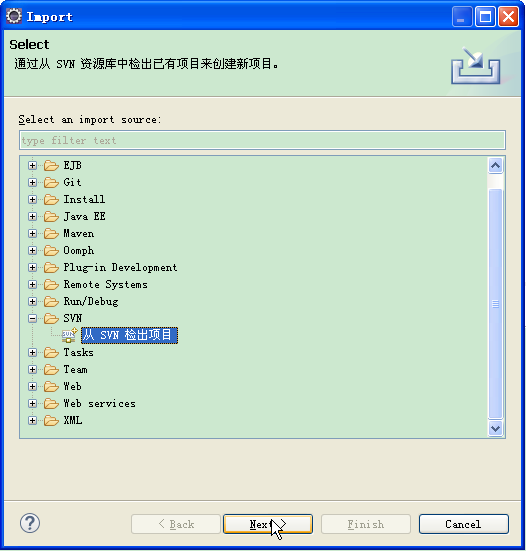




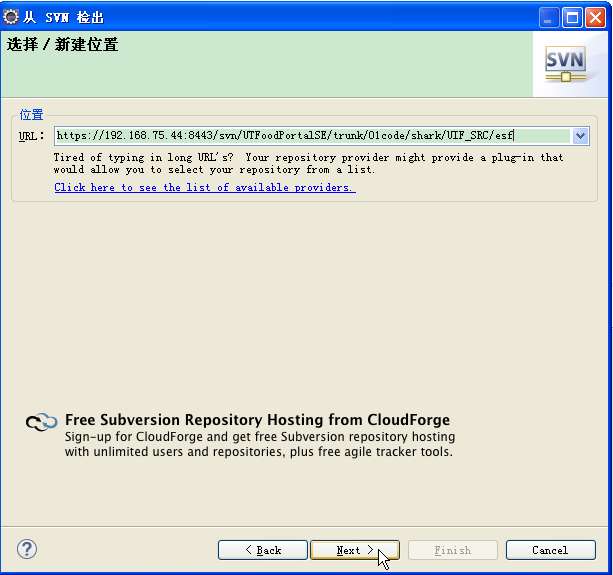
1. SVN检出project、Maven导入project
   1. 方式一

Import🡪SVN🡪从SVN检出项目

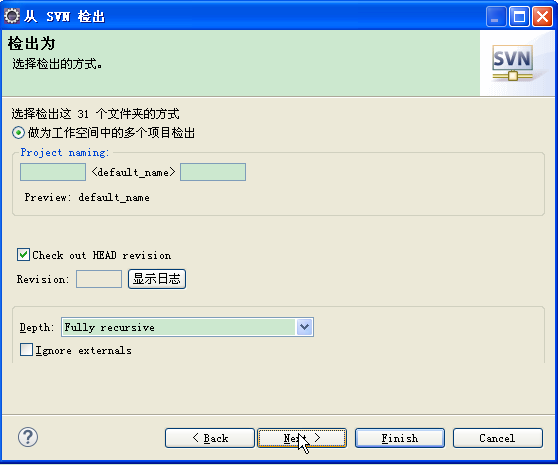


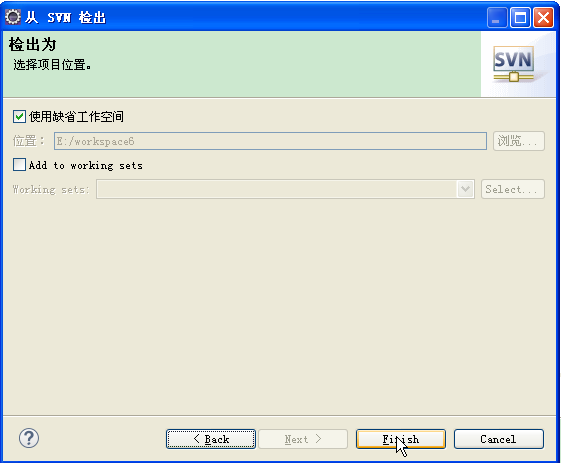


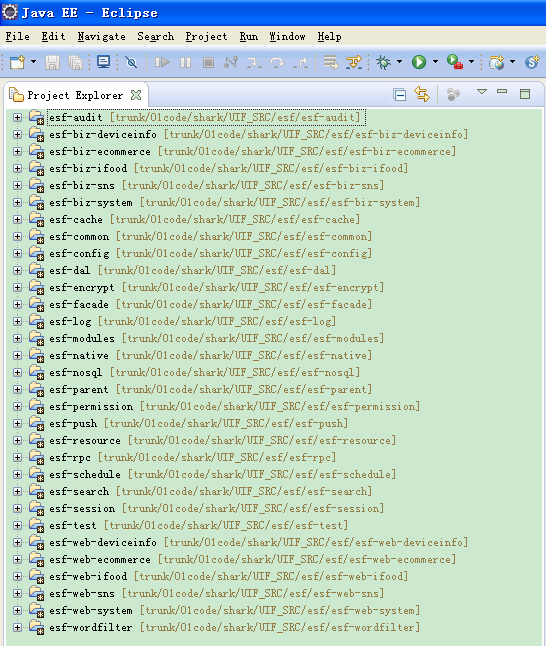




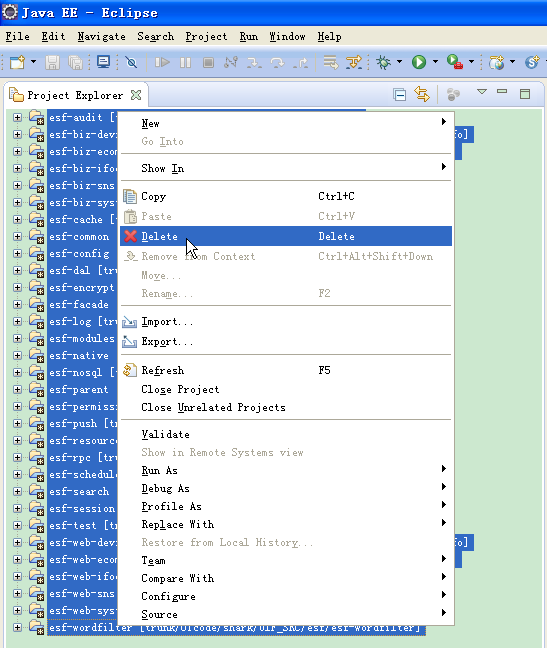


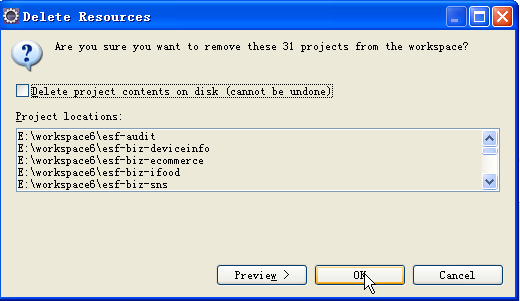




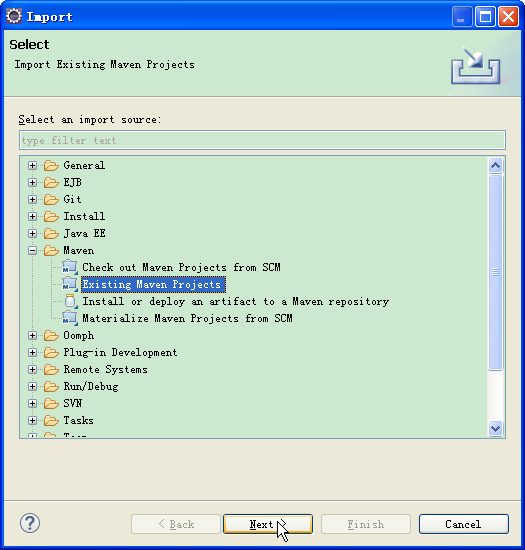


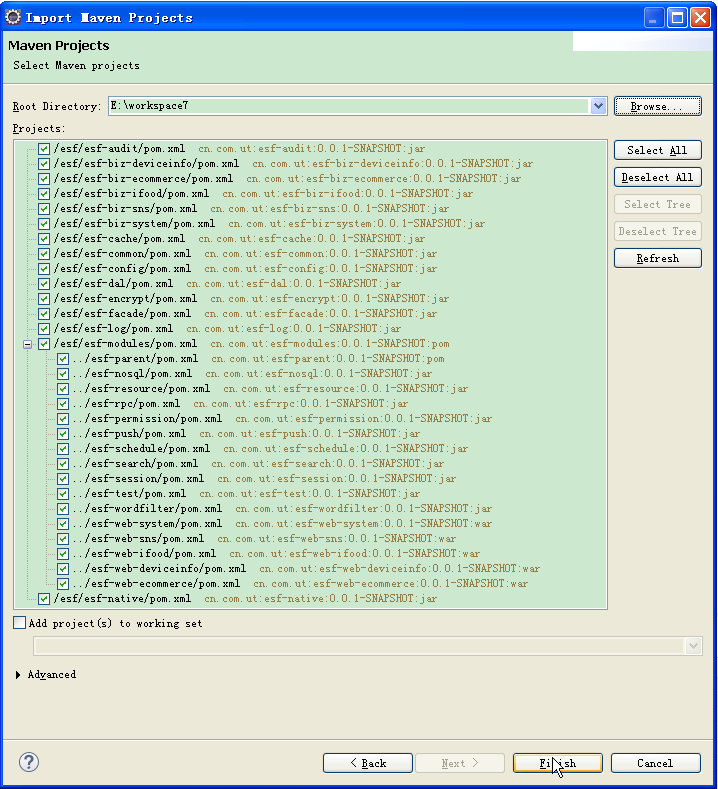
Delete（不删除项目内容）

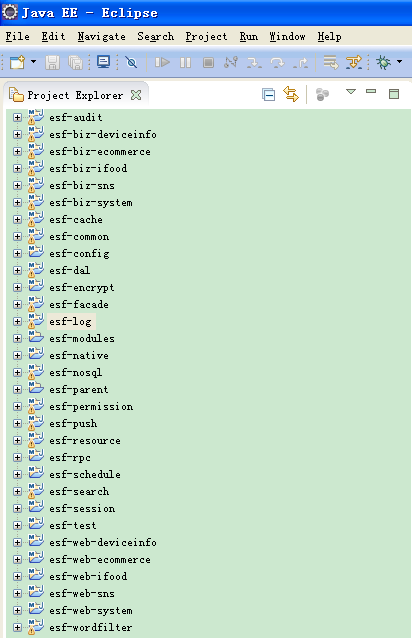




Import🡪Maven🡪 Import Existing Maven Projects

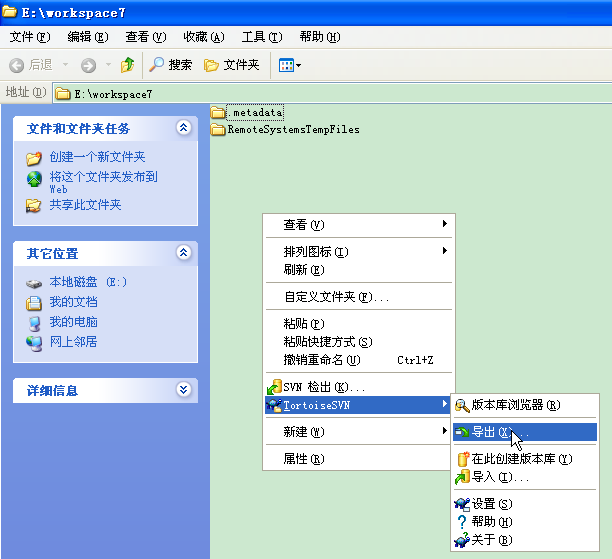


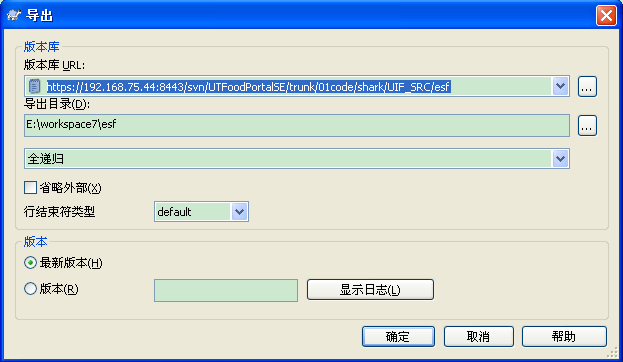




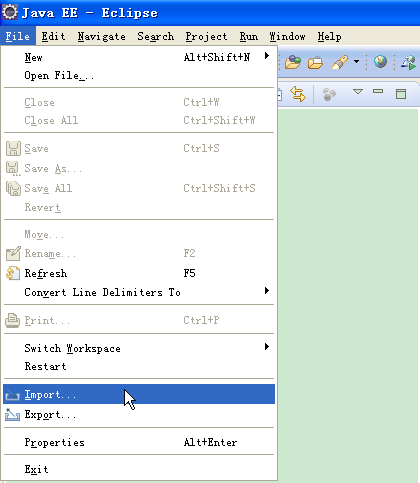
* 1. 方式二

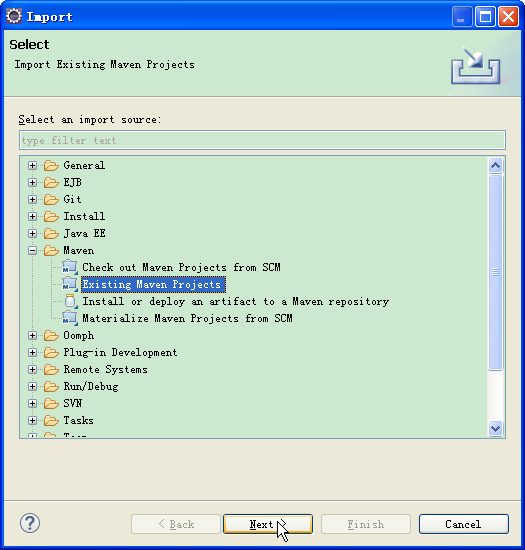
从SVN导出项目

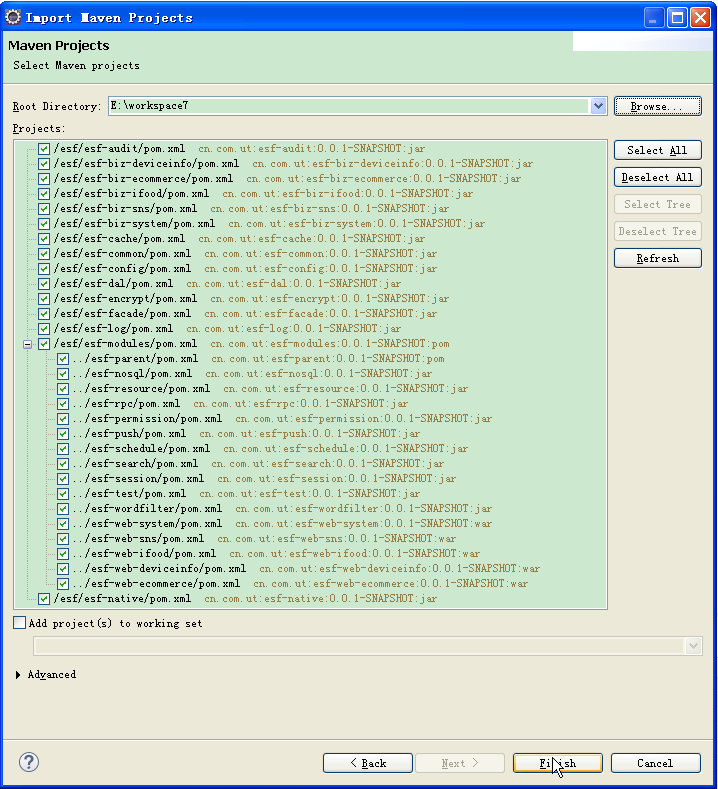


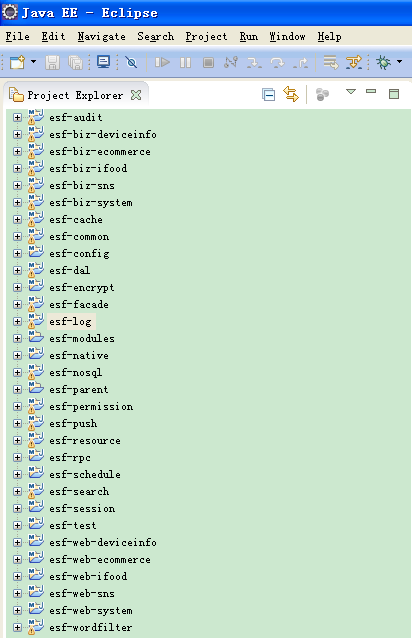


Import🡪Maven🡪 Import Existing Maven Projects



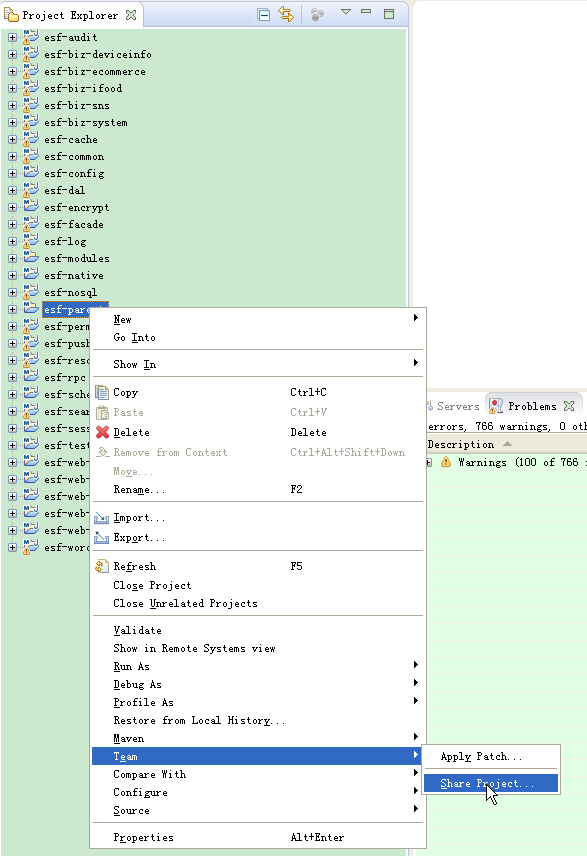


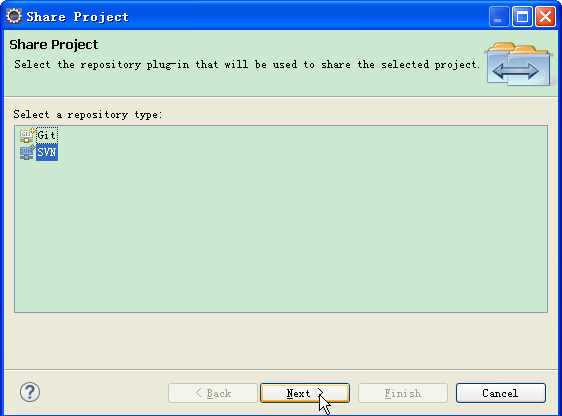


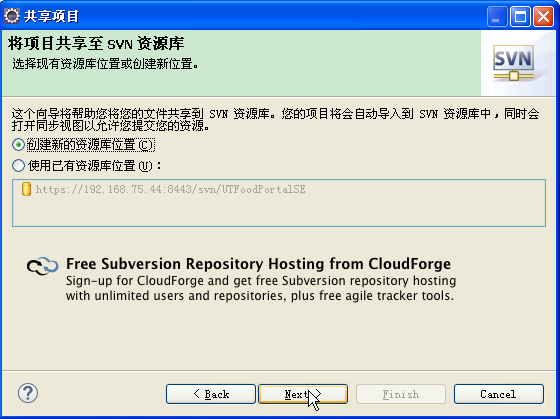


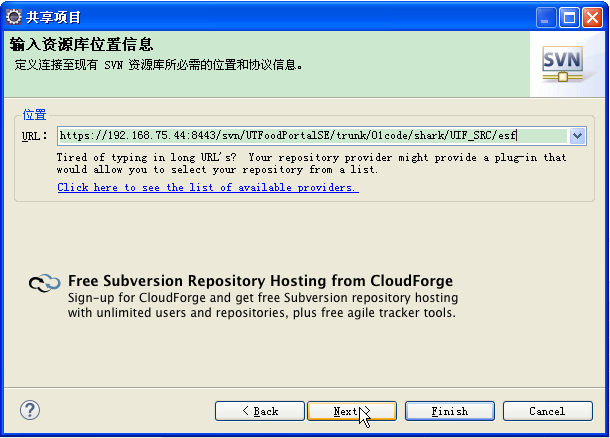
选中项目🡪点击右键🡪share project🡪新的SVN url

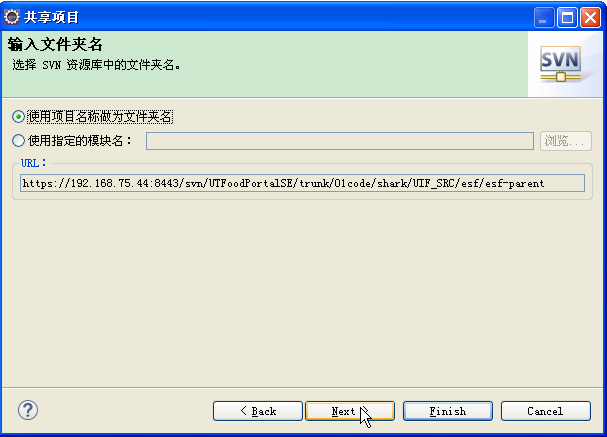
<https://192.168.75.44:8443/svn/UTFoodPortalSE/trunk/01code/shark/UIF_SRC/esf>

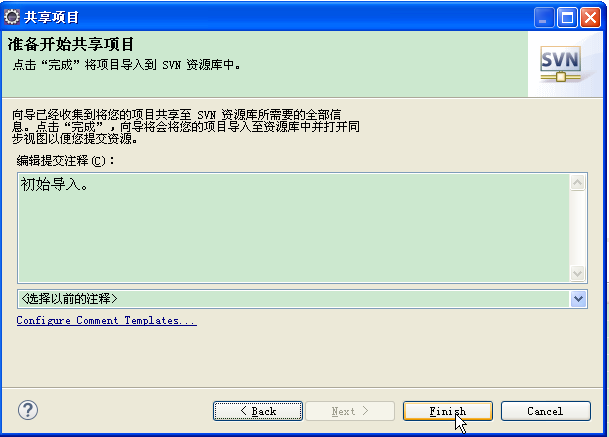


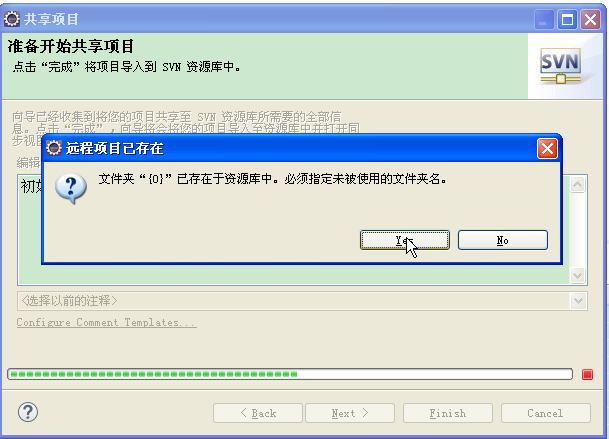


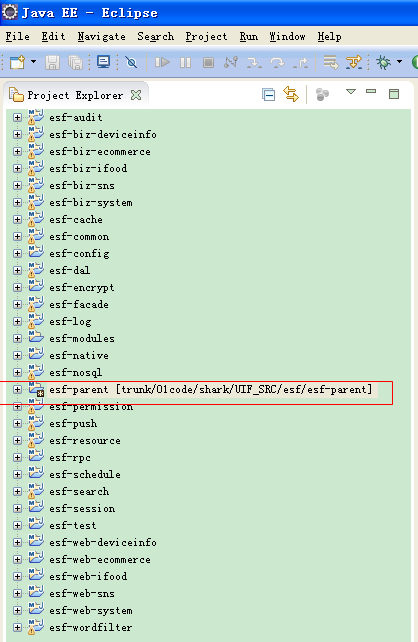






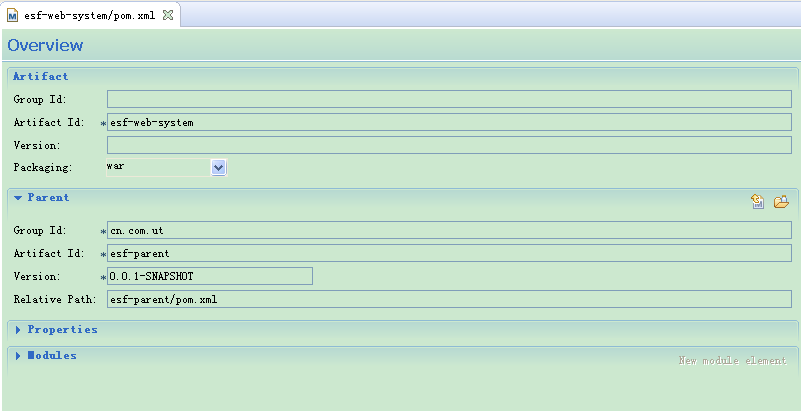




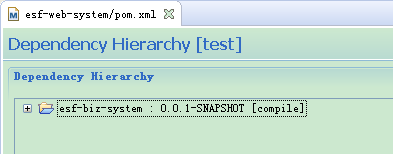


1. 项目结构
   1. Web Project：esf-web-system

pom.xml



直接依赖

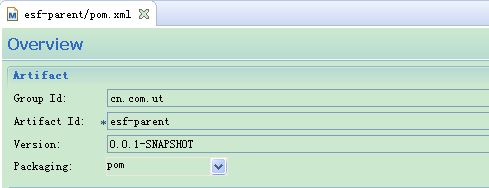


间接依赖

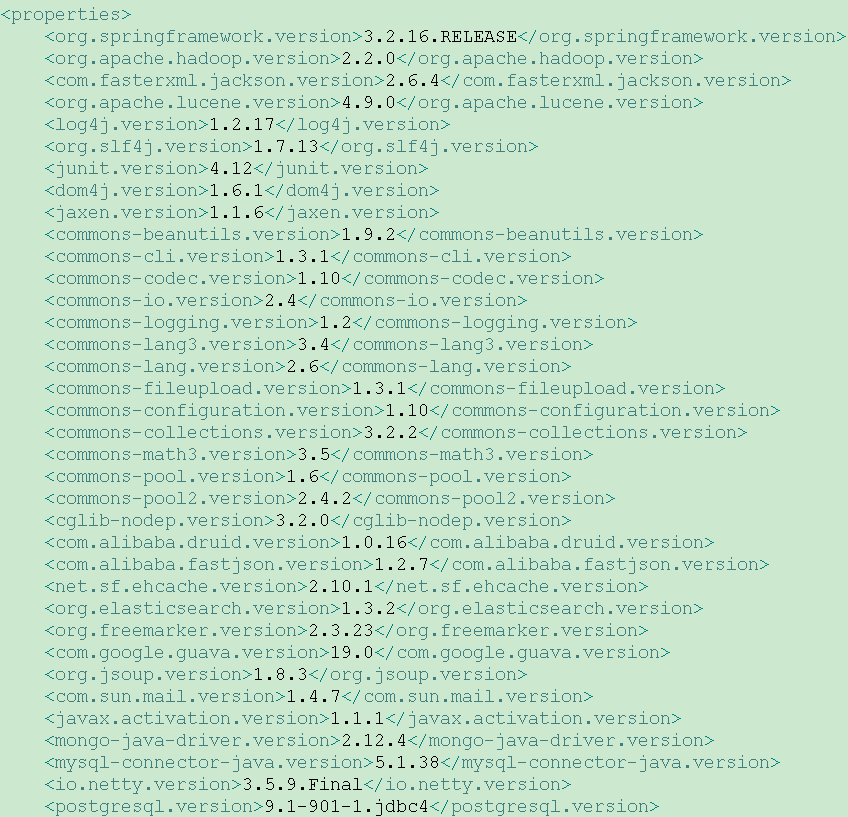


* 1. 父Project：esf-parent

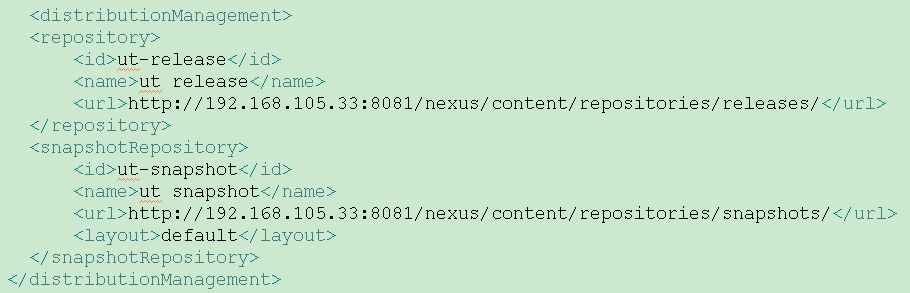
1. Artifact

****

1. Properties



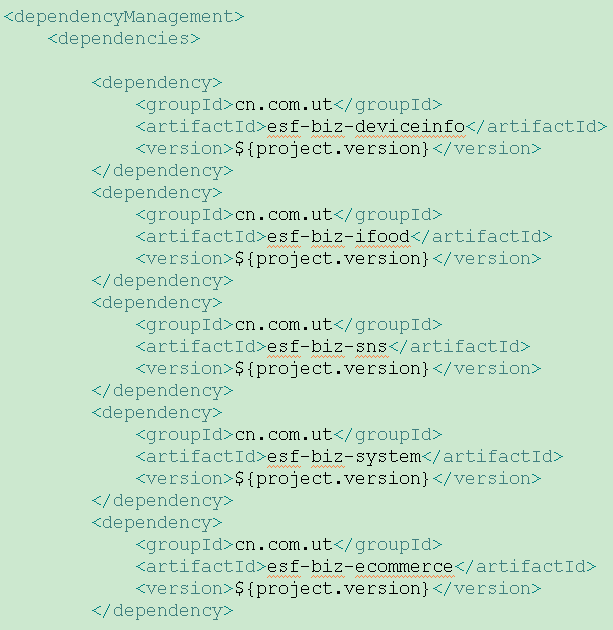
1. distributionManagement

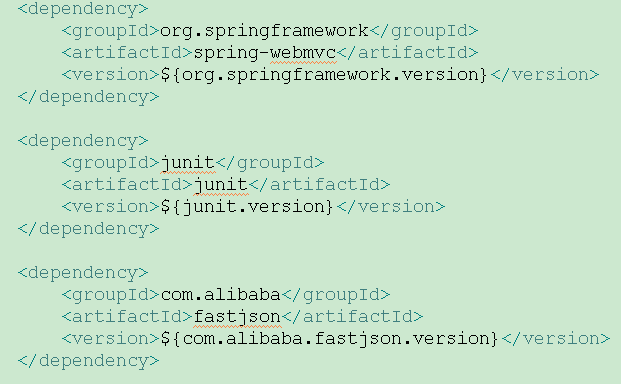


1. build

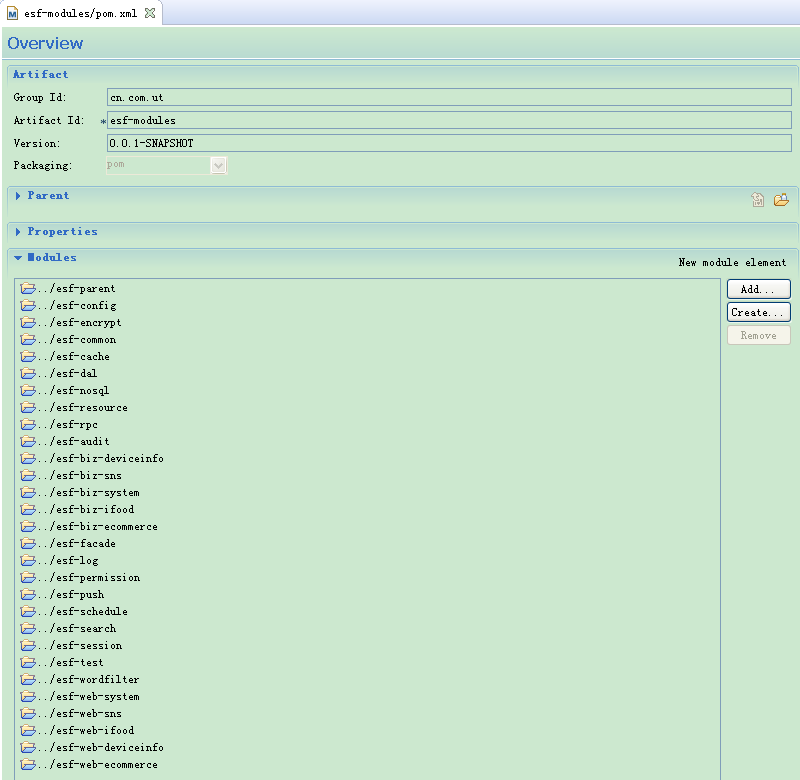


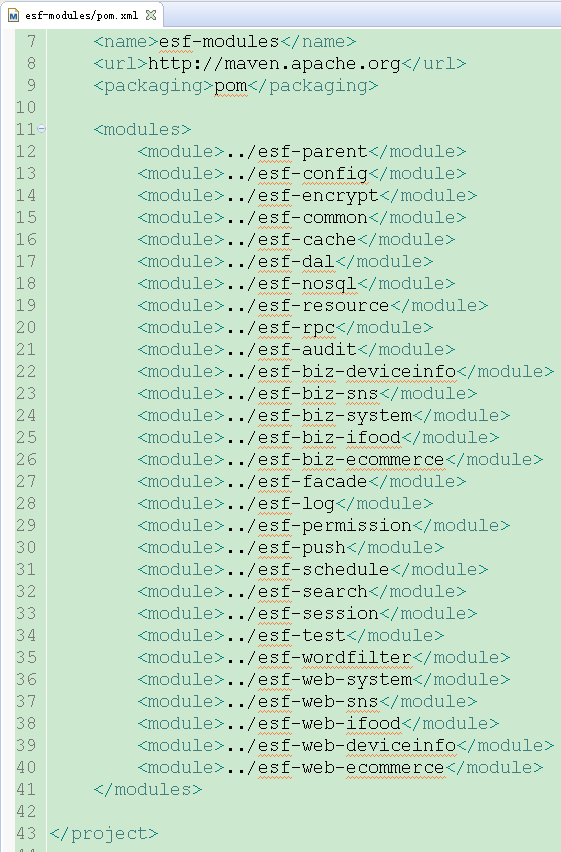
1. dependencyManagement

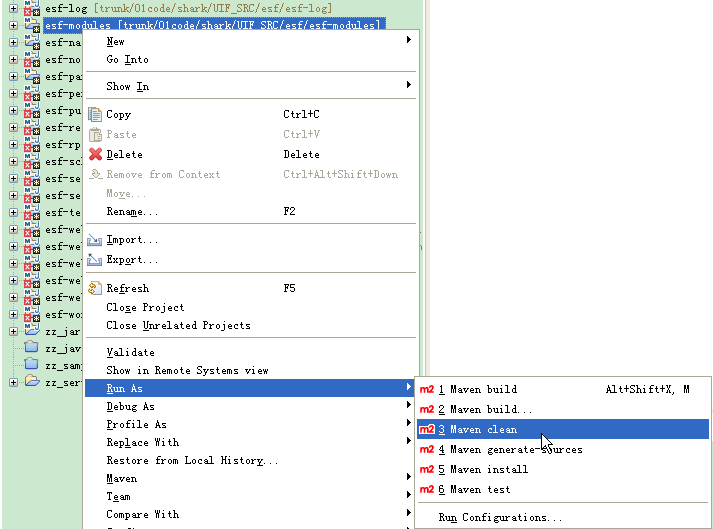


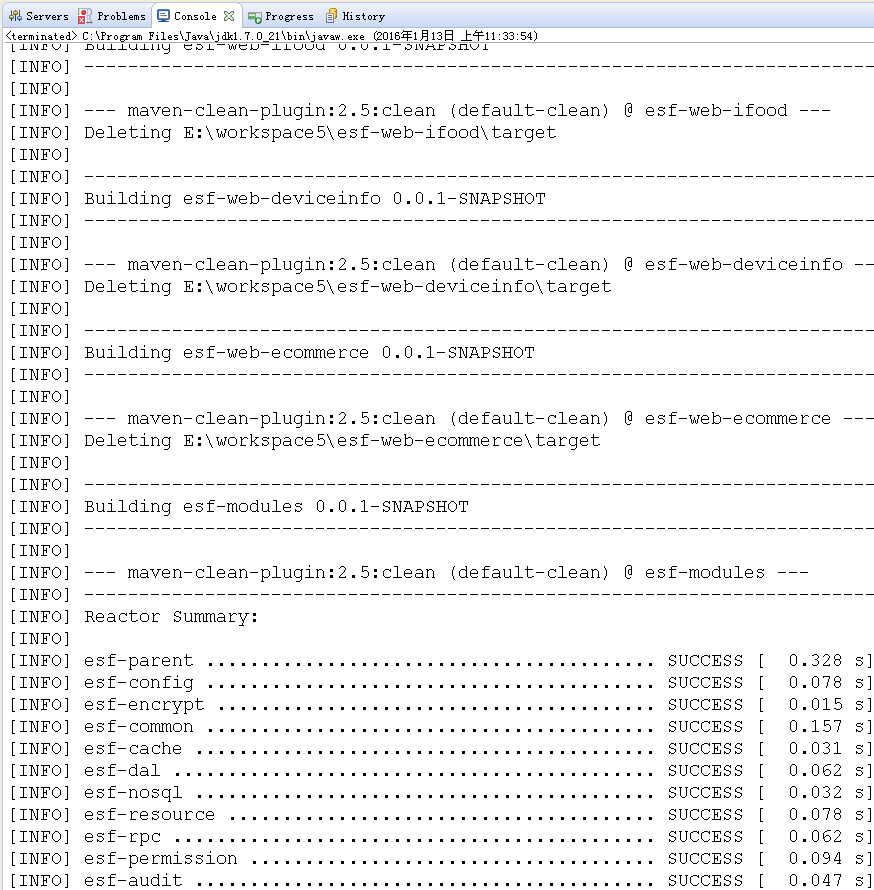


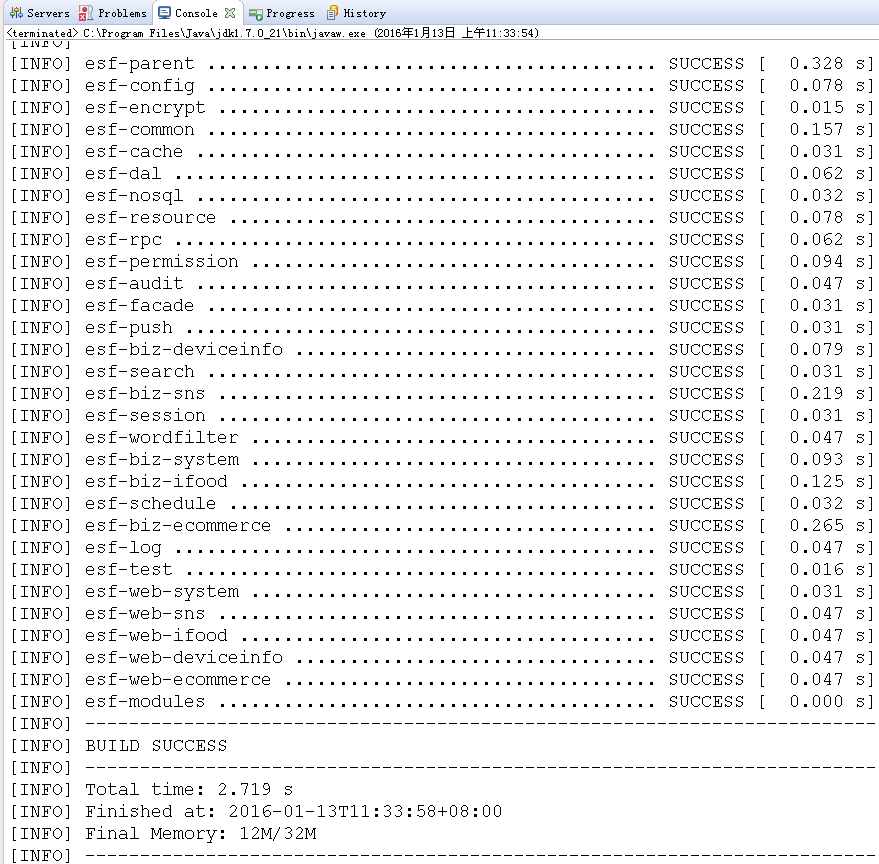
* 1. 聚合Project：esf-modules











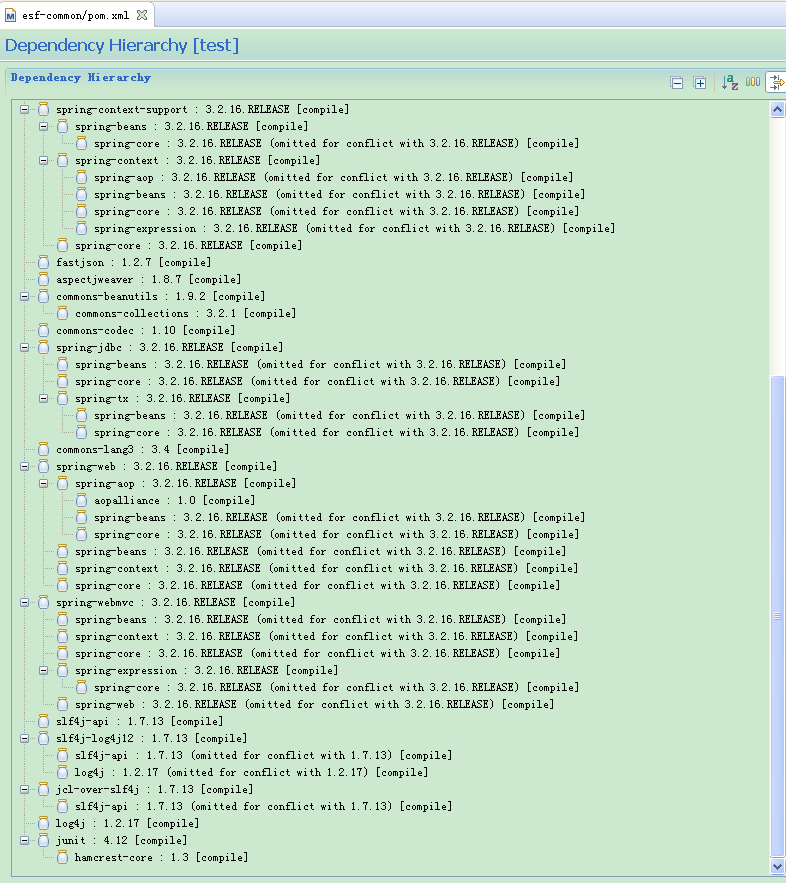
1. 子Project继承：esf-common
   1. Artifact继承（无GroupId和Version）



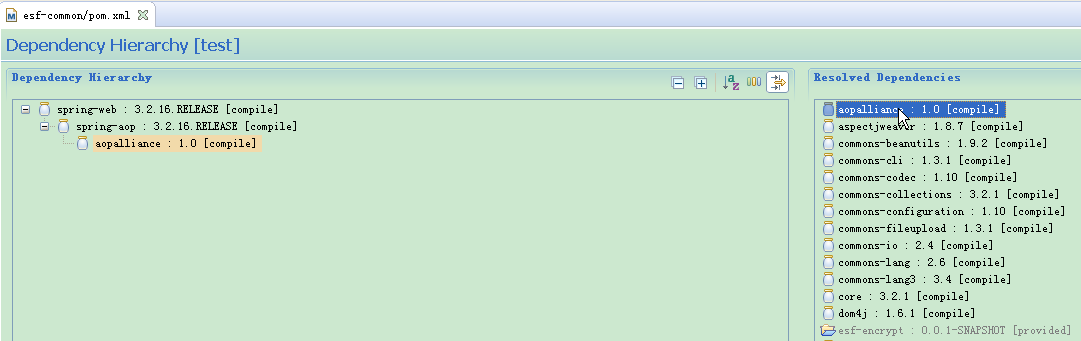
* 1. Dependency继承（无Version）



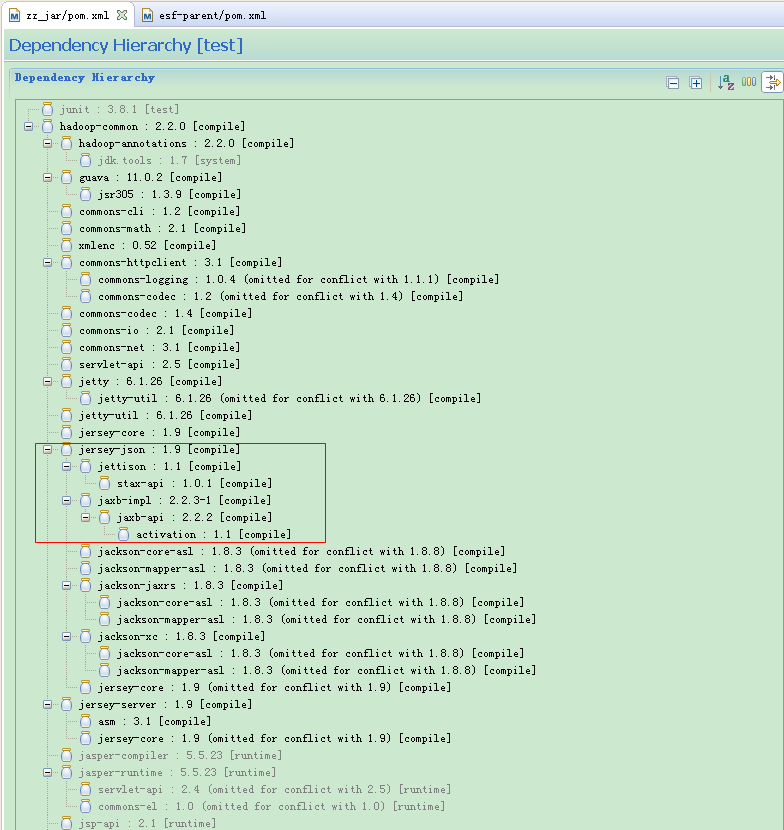
1. 依赖
   1. Dependence Hierarchy视图



* 1. Resolved Dependencies视图



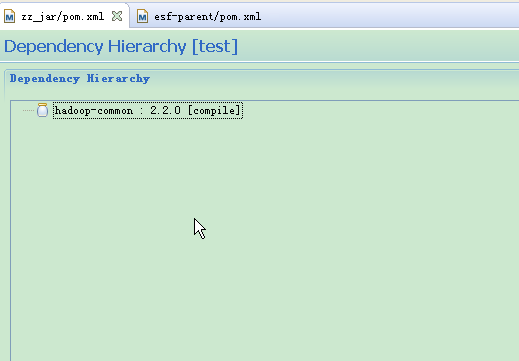
* 1. 依赖传递



* 1. 依赖排除





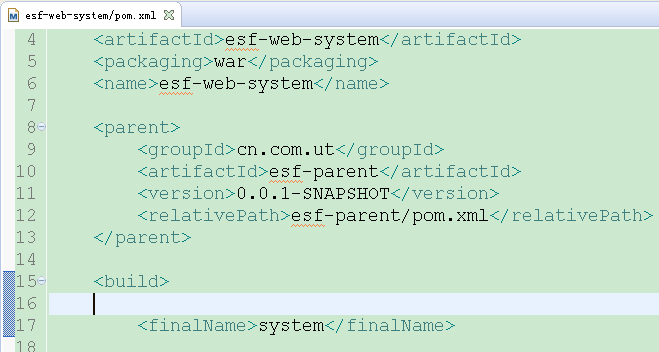


* 1. 依赖范围



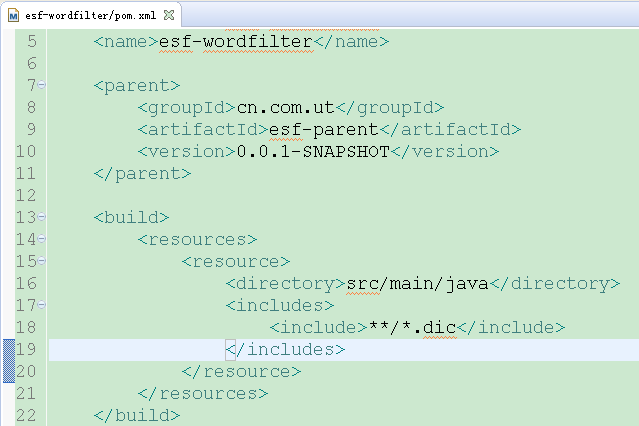
1. build

finalName

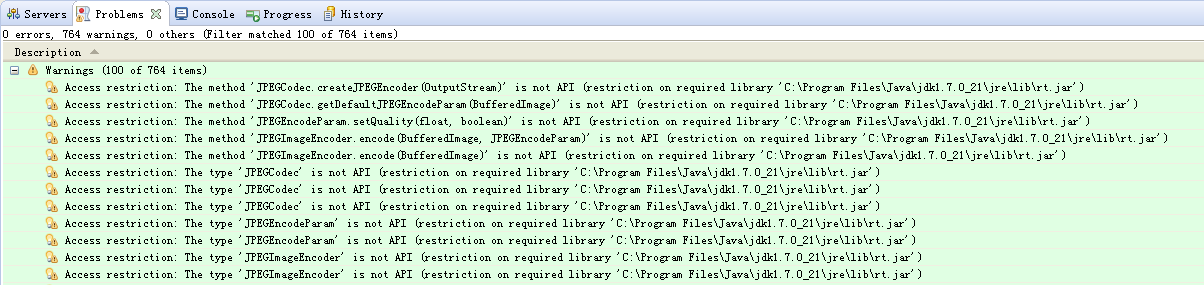


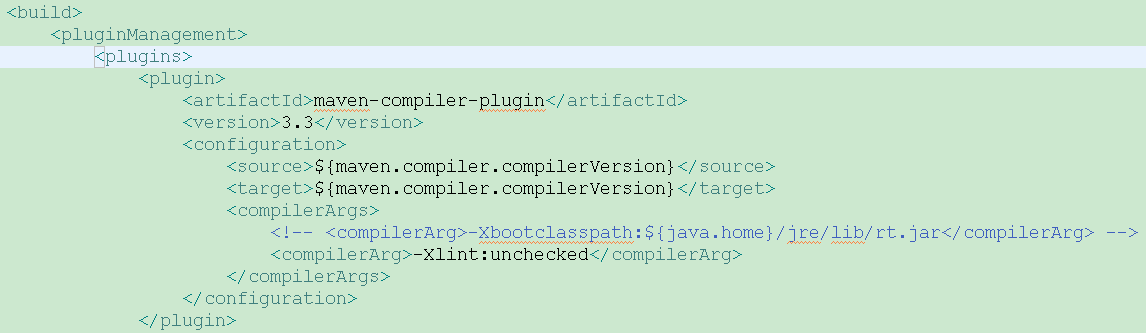
resources



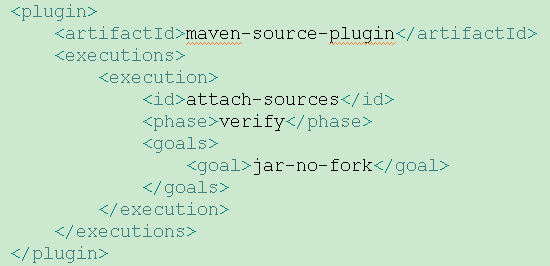


1. 常用插件介绍
   1. 编译：maven-compiler-plugin

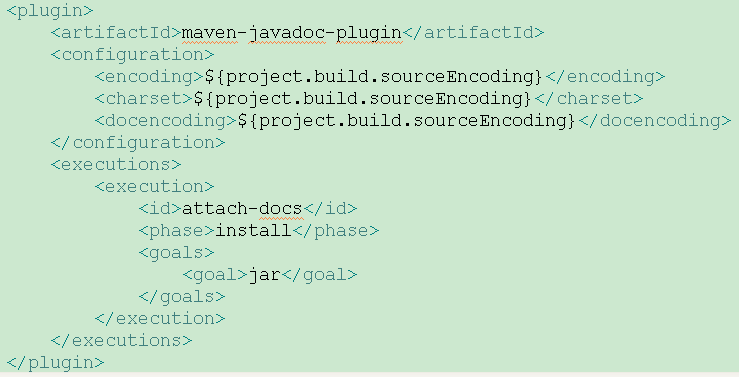




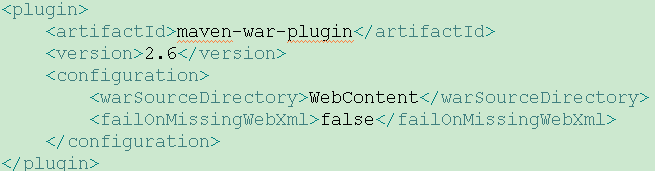
* 1. Source打包：maven-source-plugin



* 1. Javadoc打包：maven-javadoc-plugin

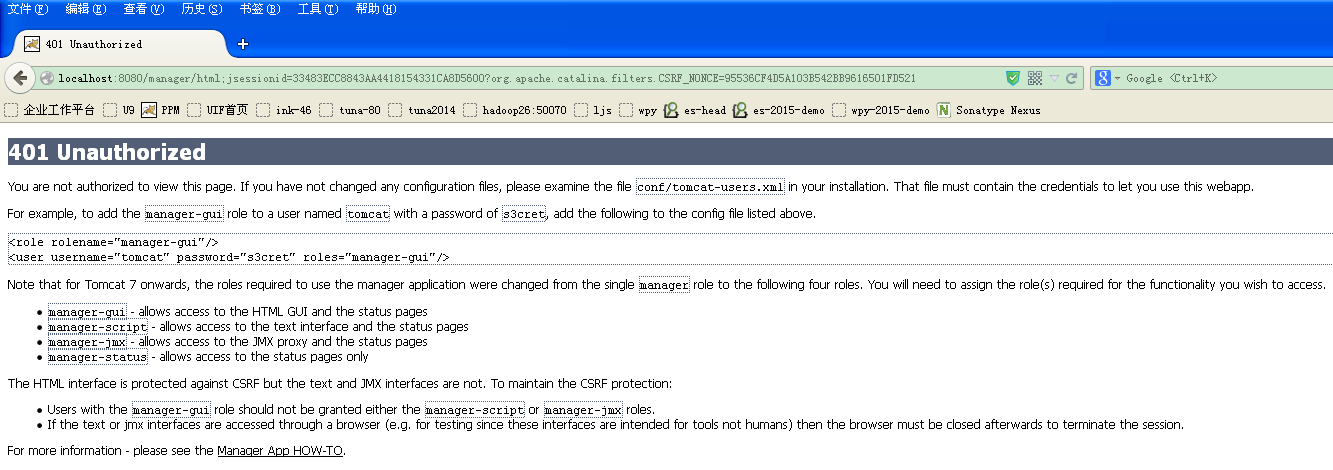


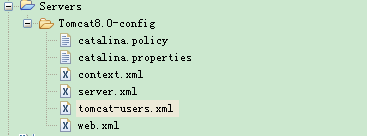
* 1. War包：maven-war-plugin

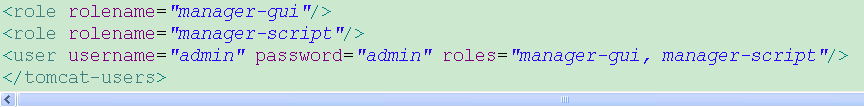


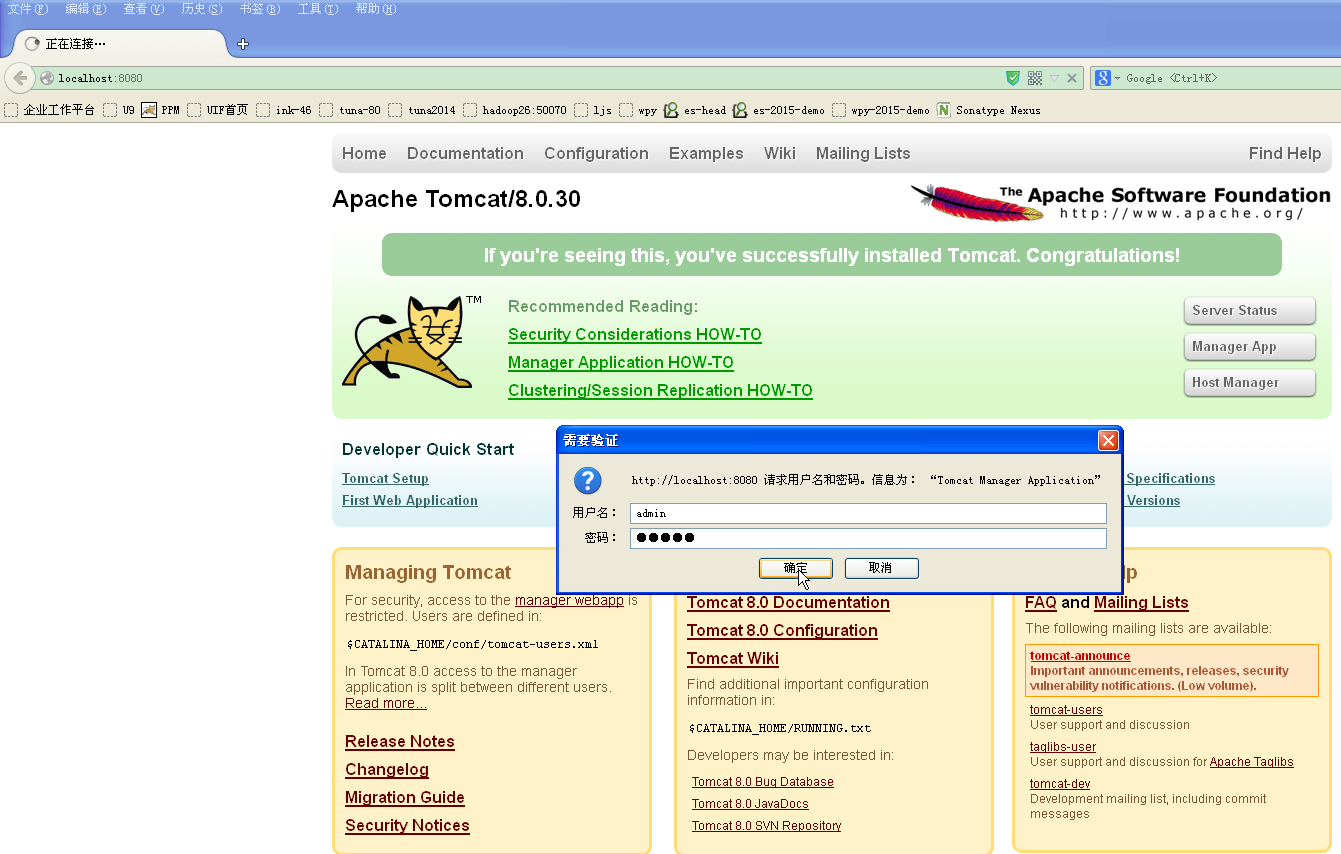
* 1. Tomcat远程部署：tomcat7-maven-plugin

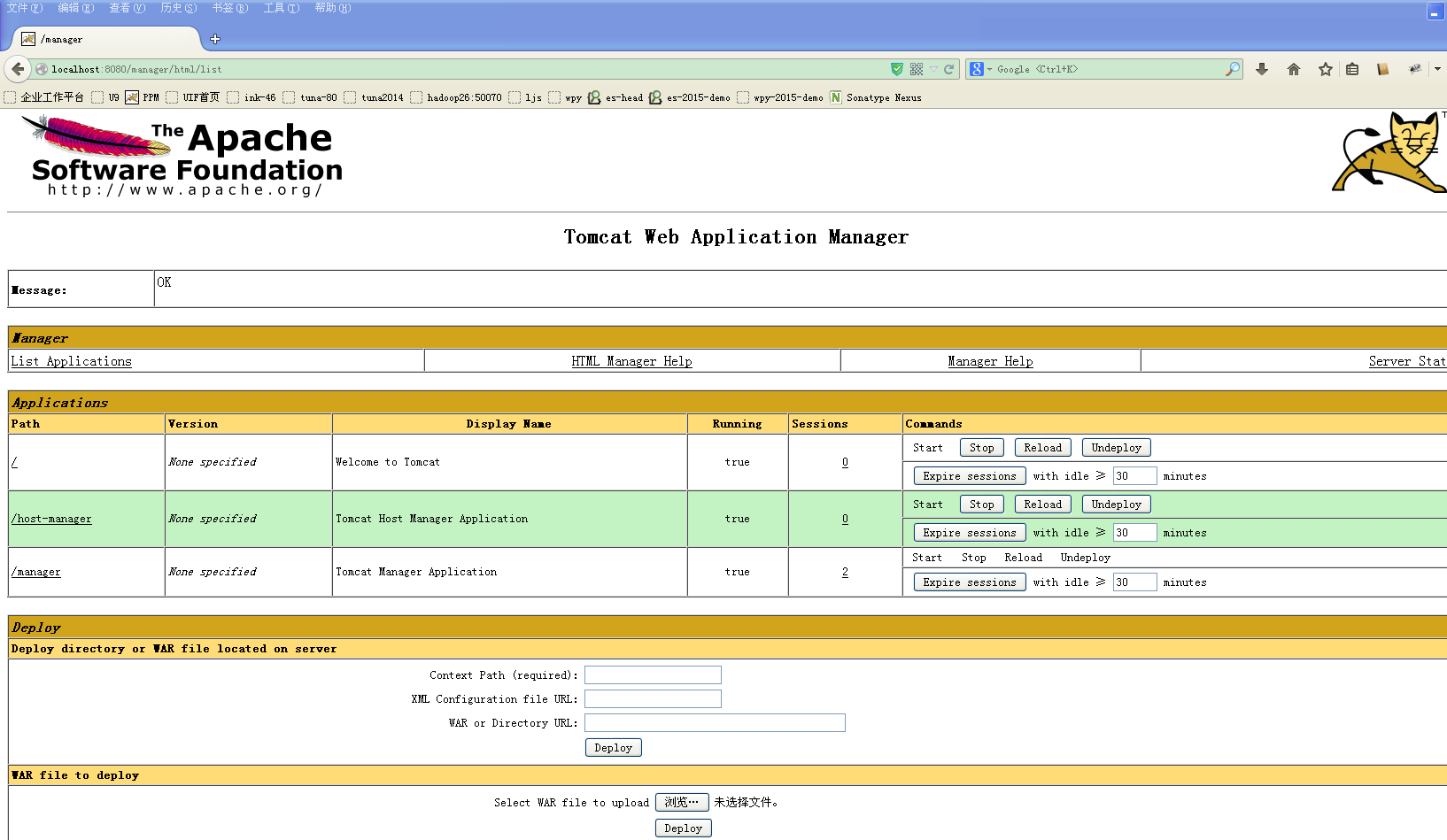




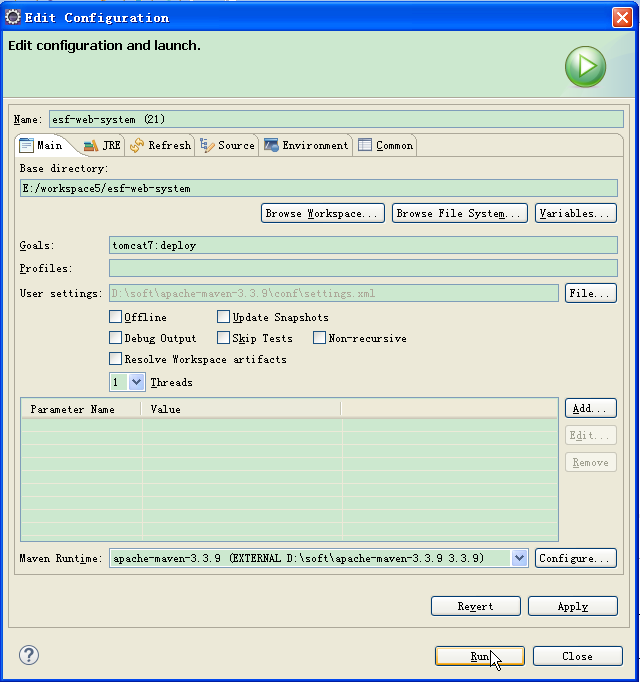


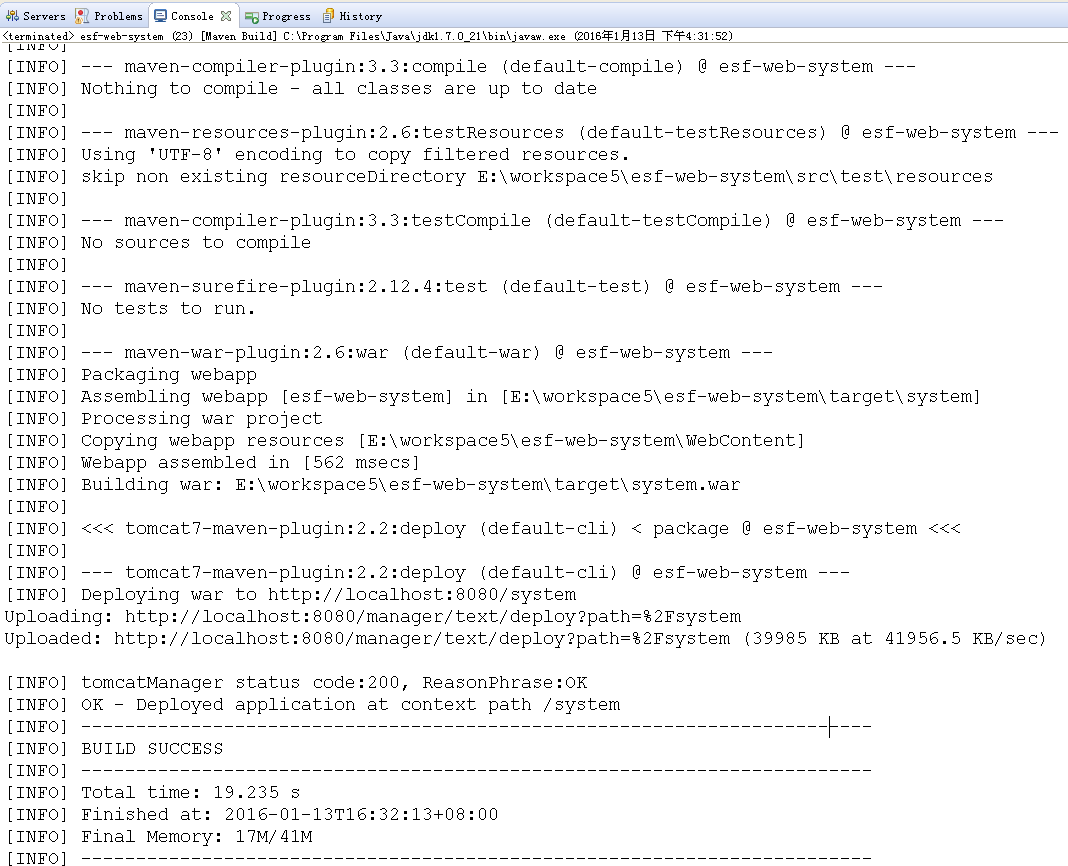


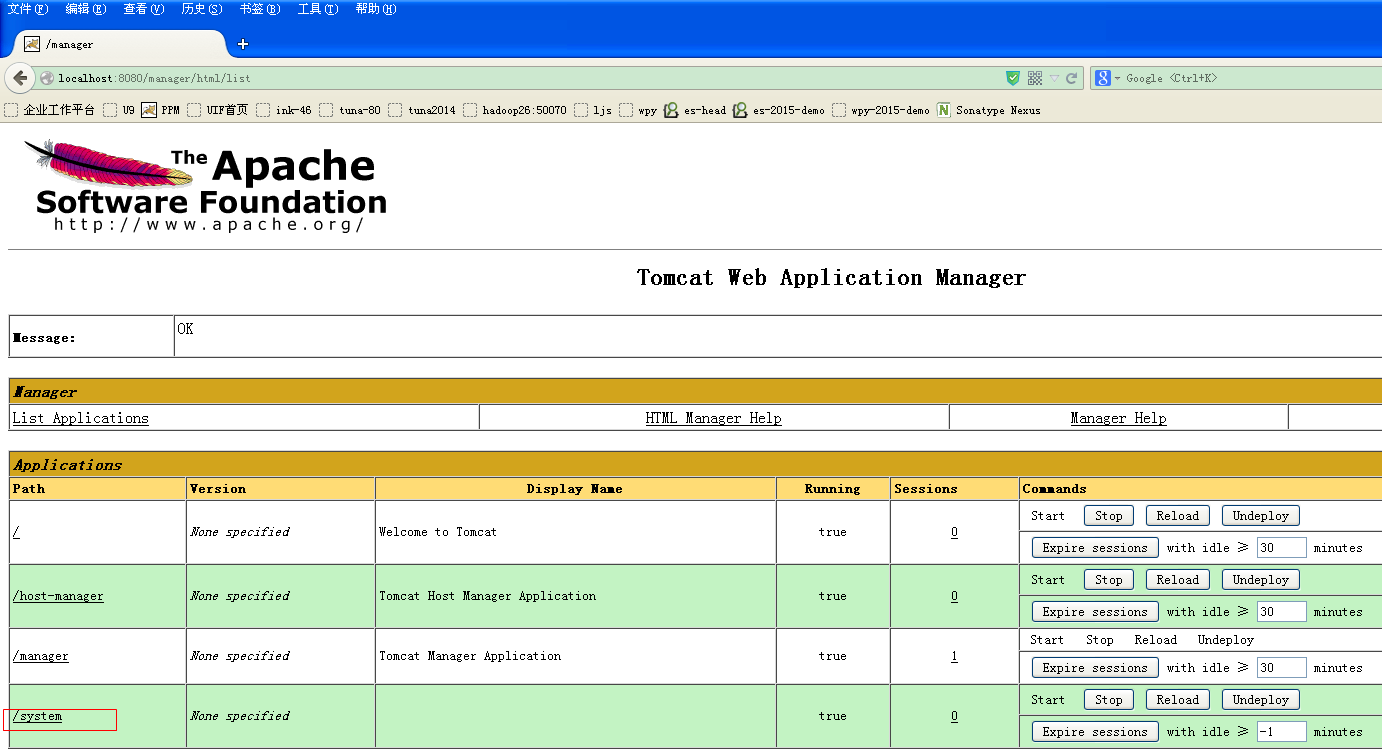


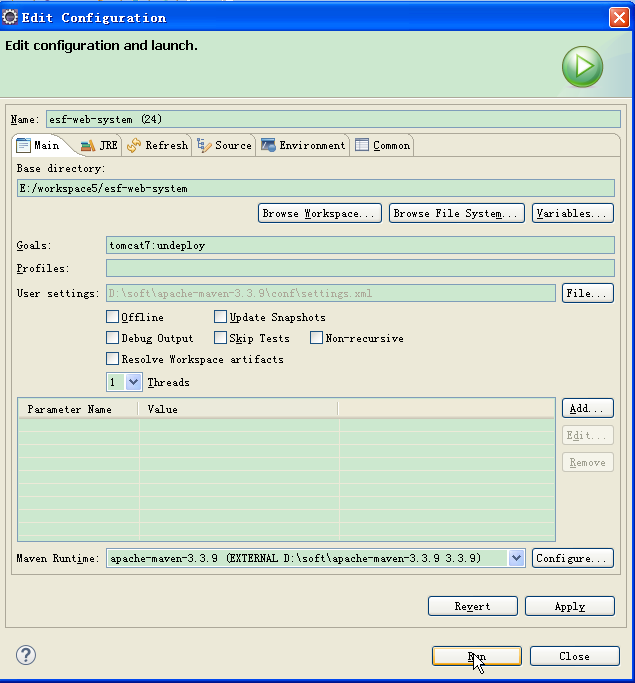




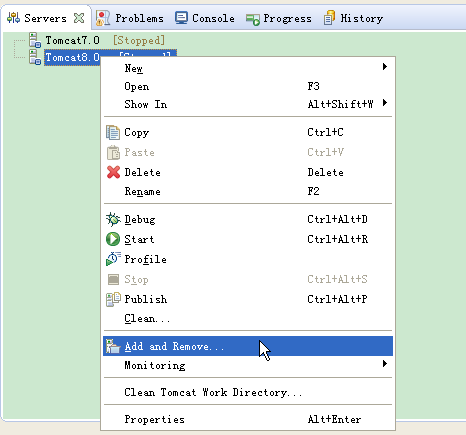


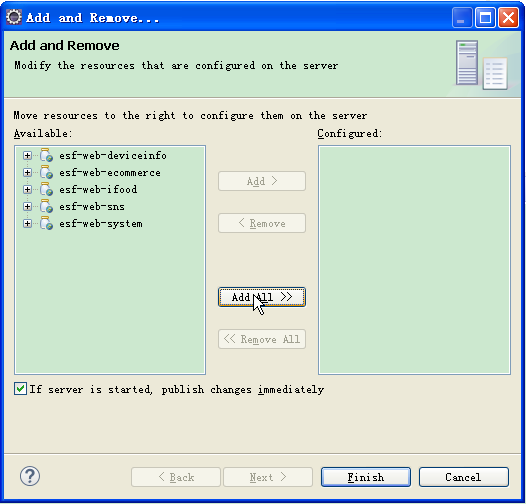


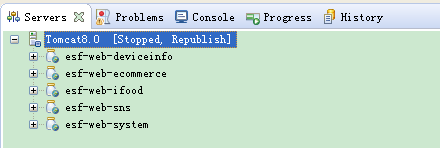




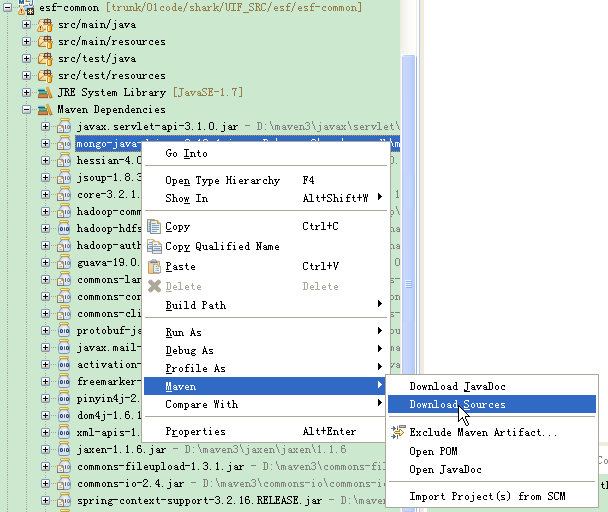
1. Tomcat本地开发

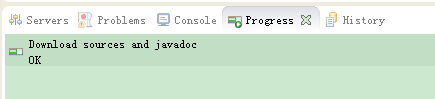


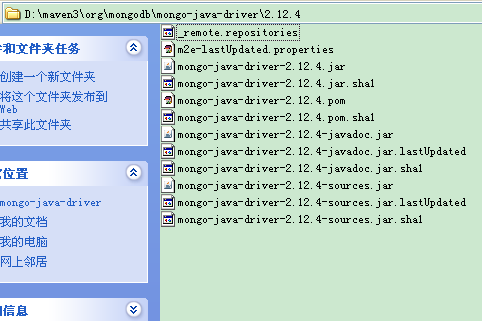




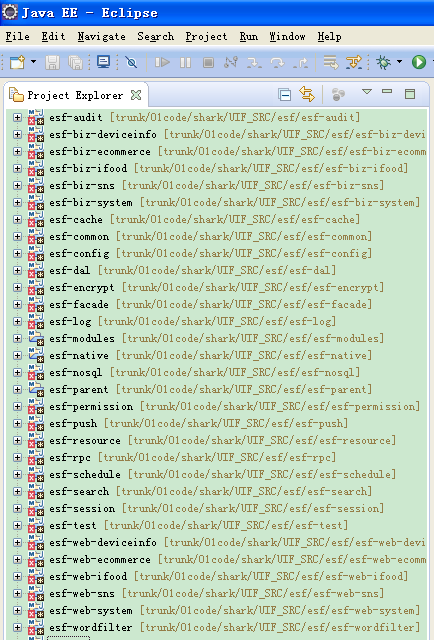
1. 下载Javadoc和source

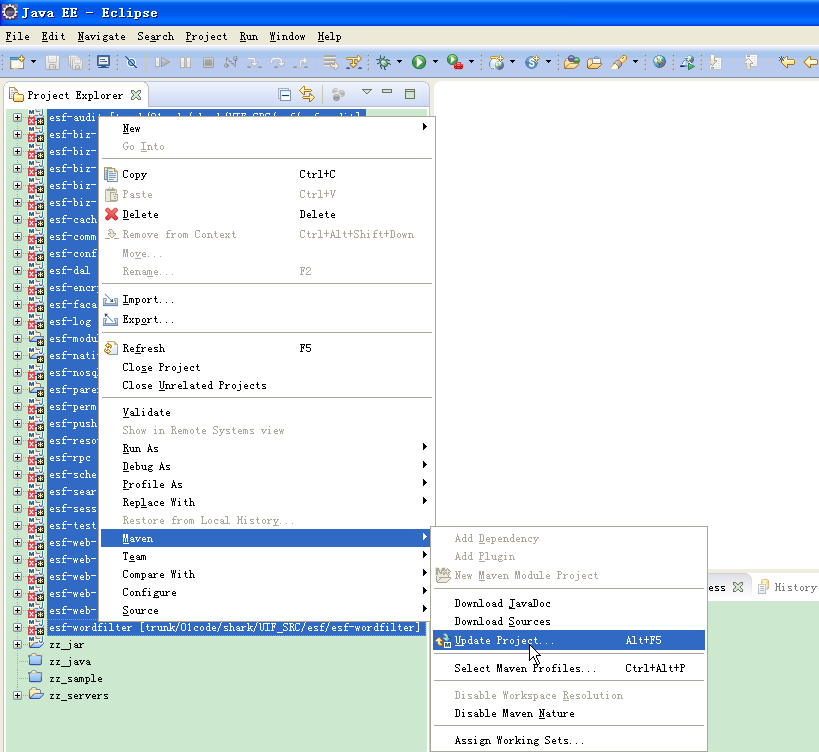


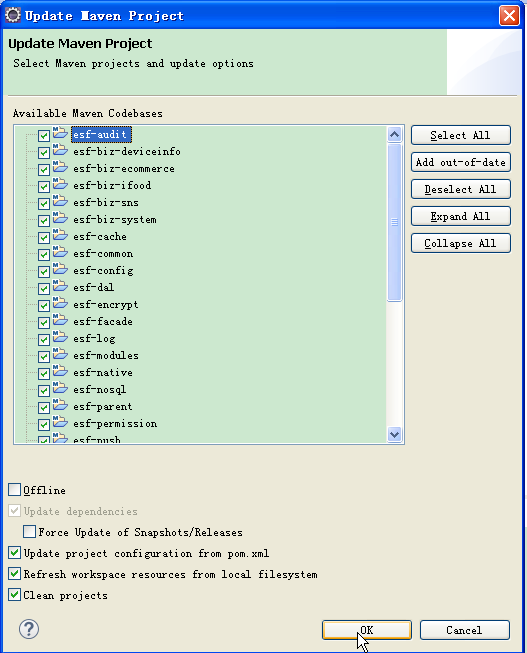




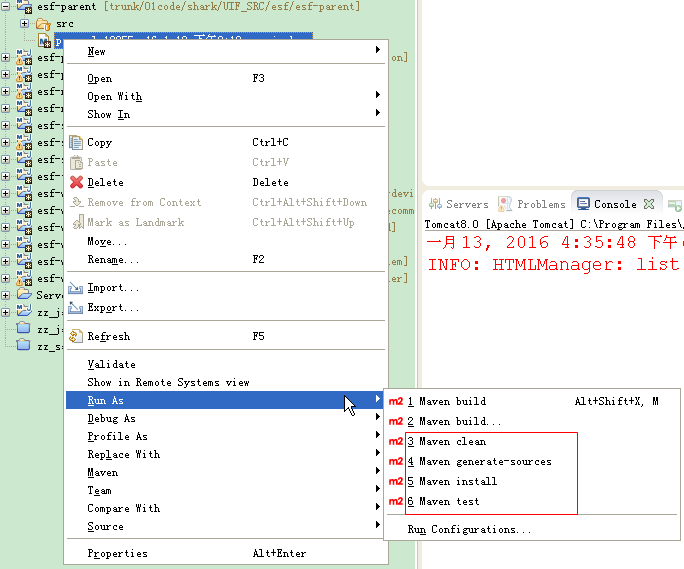
1. Eclipse Update project

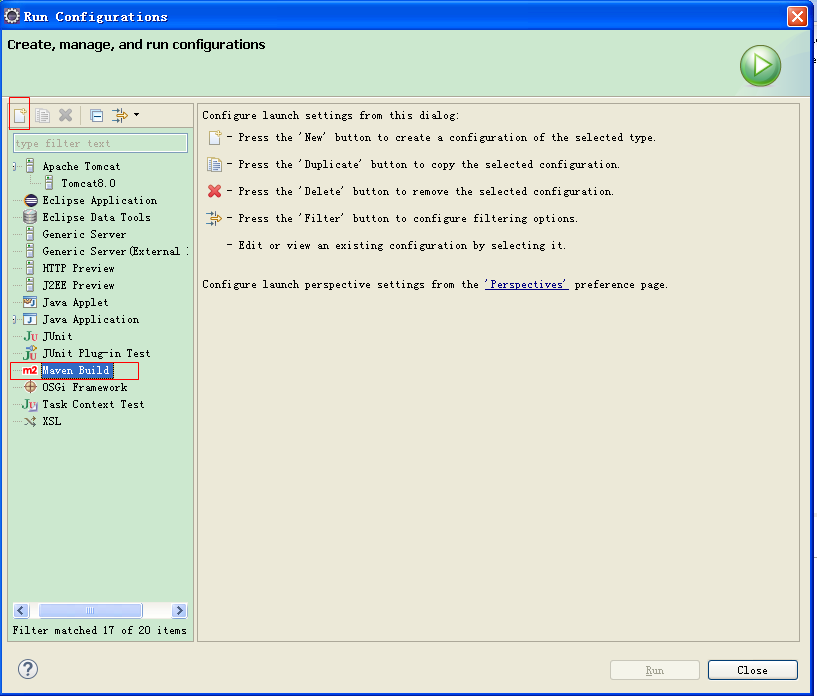


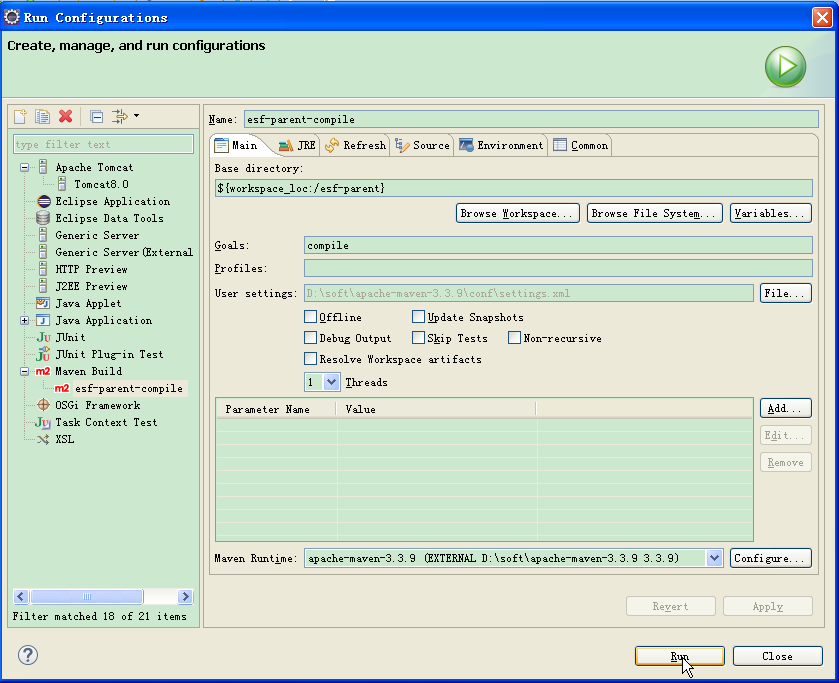




1. maven run





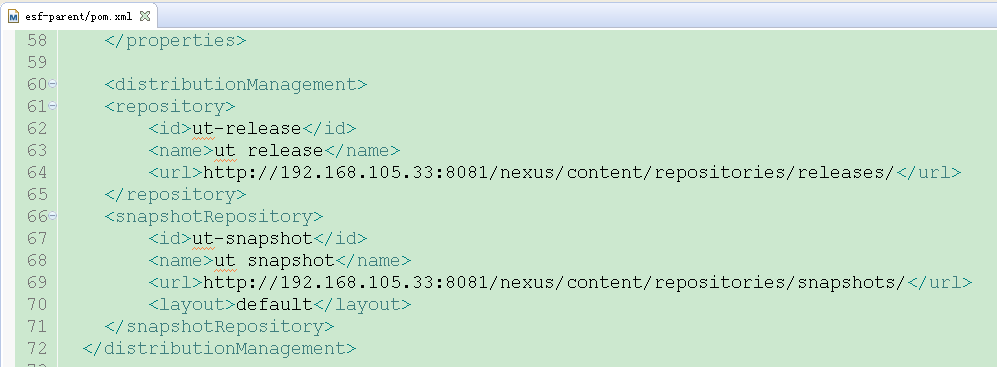


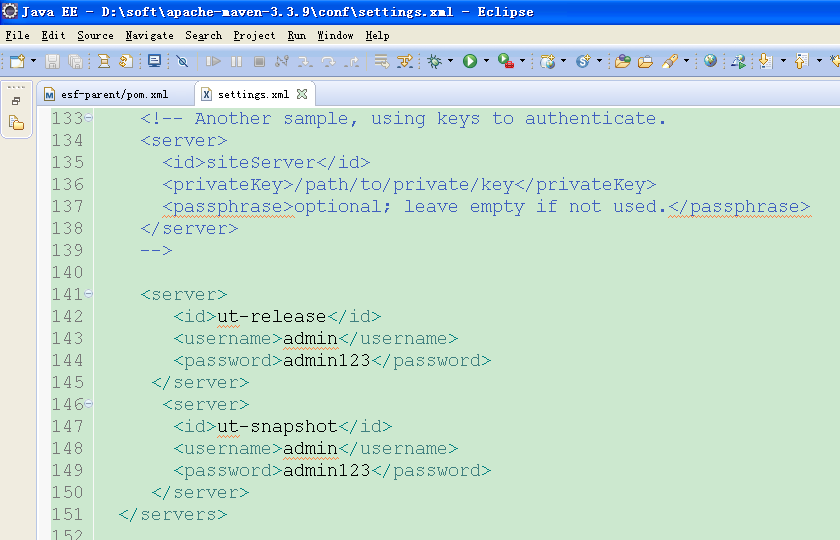
* 1. clean
  2. compile
  3. install

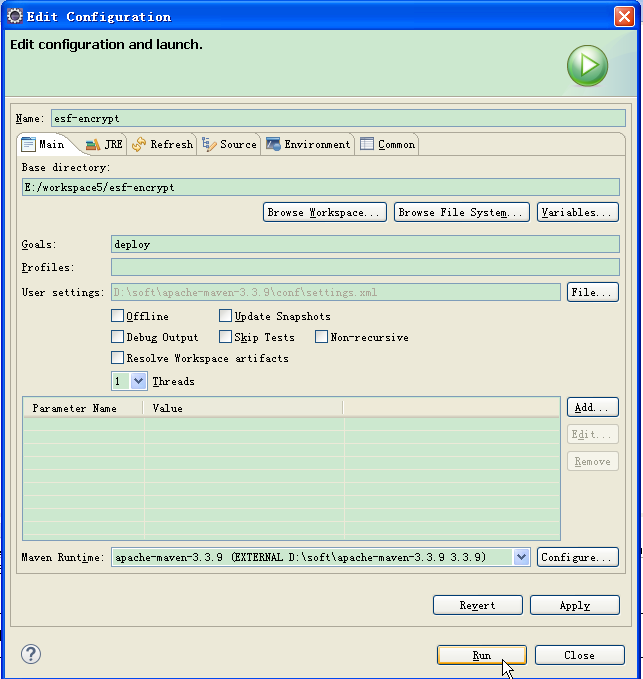
install到本地仓库

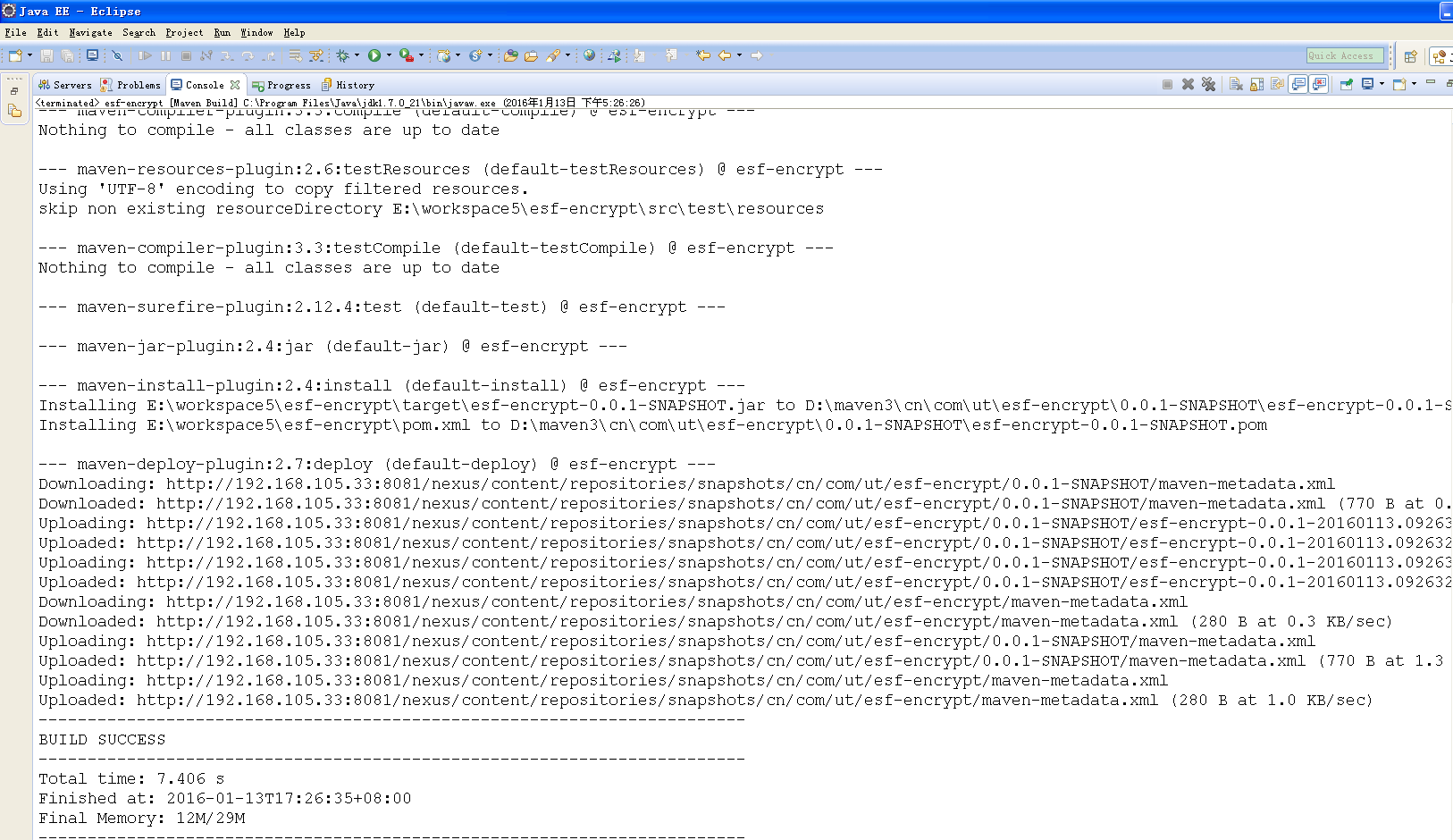
|  |
| --- |
| mvn install:install-file -Dfile=D:\tools\pinyin4j-2.5.0\lib\pinyin4j-2.5.0.jar -DgroupId=net.sourceforge -DartifactId=pinyin4j -Dversion=2.5.0 -Dpackaging=jar -DgeneratePom=true -DcreateChecksum=true |

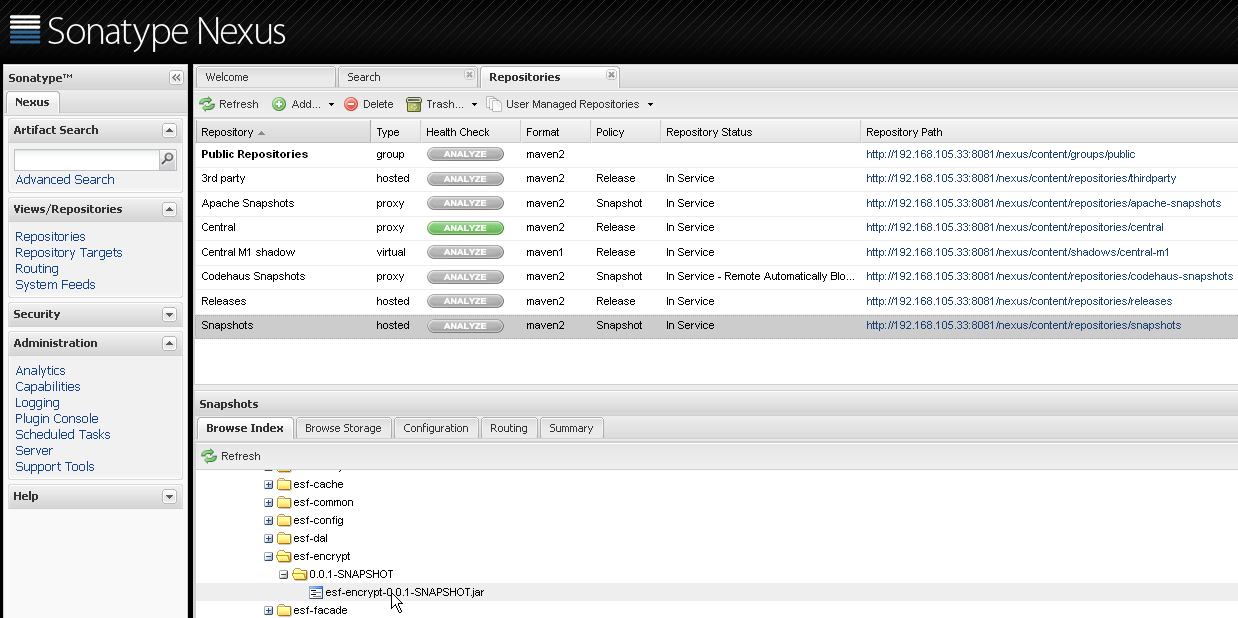
* 1. deploy











* 1. test

1. 生命周期和插件

一个完整的项目构建过程通常包括清理、编译、测试、打包、集成测试、验证、部署等步骤，Maven从中抽取了一套完善的、易扩展的生命周期。Maven的生命周期是抽象的，其中的具体任务都交由插件来完成。Maven为大多数构建任务编写并绑定了默认的插件，如针对编译的插件：maven-compiler-plugin。用户也可自行配置或编写插件。

* 1. 三套生命周期

Maven定义了三套生命周期：clean、default、site，每个生命周期都包含了一些阶段（phase）。三套生命周期相互独立，但各个生命周期中的phase却是有顺序的，且后面的phase依赖于前面的phase。执行某个phase时，其前面的phase会依顺序执行，但不会触发另外两套生命周期中的任何phase。

* + 1. clean生命周期

1. pre-clean

执行清理前的工作；

1. clean

清理上一次构建生成的所有文件；

1. post-clean

执行清理后的工作

* + 1. default生命周期

default生命周期是最核心的，它包含了构建项目时真正需要执行的所有步骤。

1. validate
2. initialize
3. generate-sources
4. process-sources
5. generate-resources
6. process-resources

复制和处理资源文件到target目录，准备打包；

1. compile

编译项目的源代码；

1. process-classes
2. generate-test-sources
3. process-test-sources
4. generate-test-resources
5. process-test-resources
6. test-compile

编译测试源代码；

1. process-test-classes
2. test

运行测试代码；

1. prepare-package
2. package

打包成jar或者war或者其他格式的分发包；

1. pre-integration-test
2. integration-test
3. post-integration-test
4. verify
5. install

将打好的包安装到本地仓库，供其他项目使用；

1. deploy

将打好的包安装到远程仓库，供其他项目使用；

* + 1. site生命周期

1. pre-site
2. site

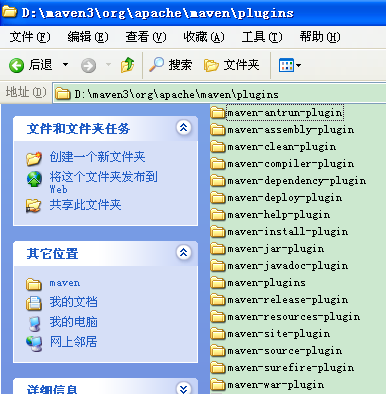
生成项目的站点文档；

1. post-site
2. site-deploy

发布生成的站点文档

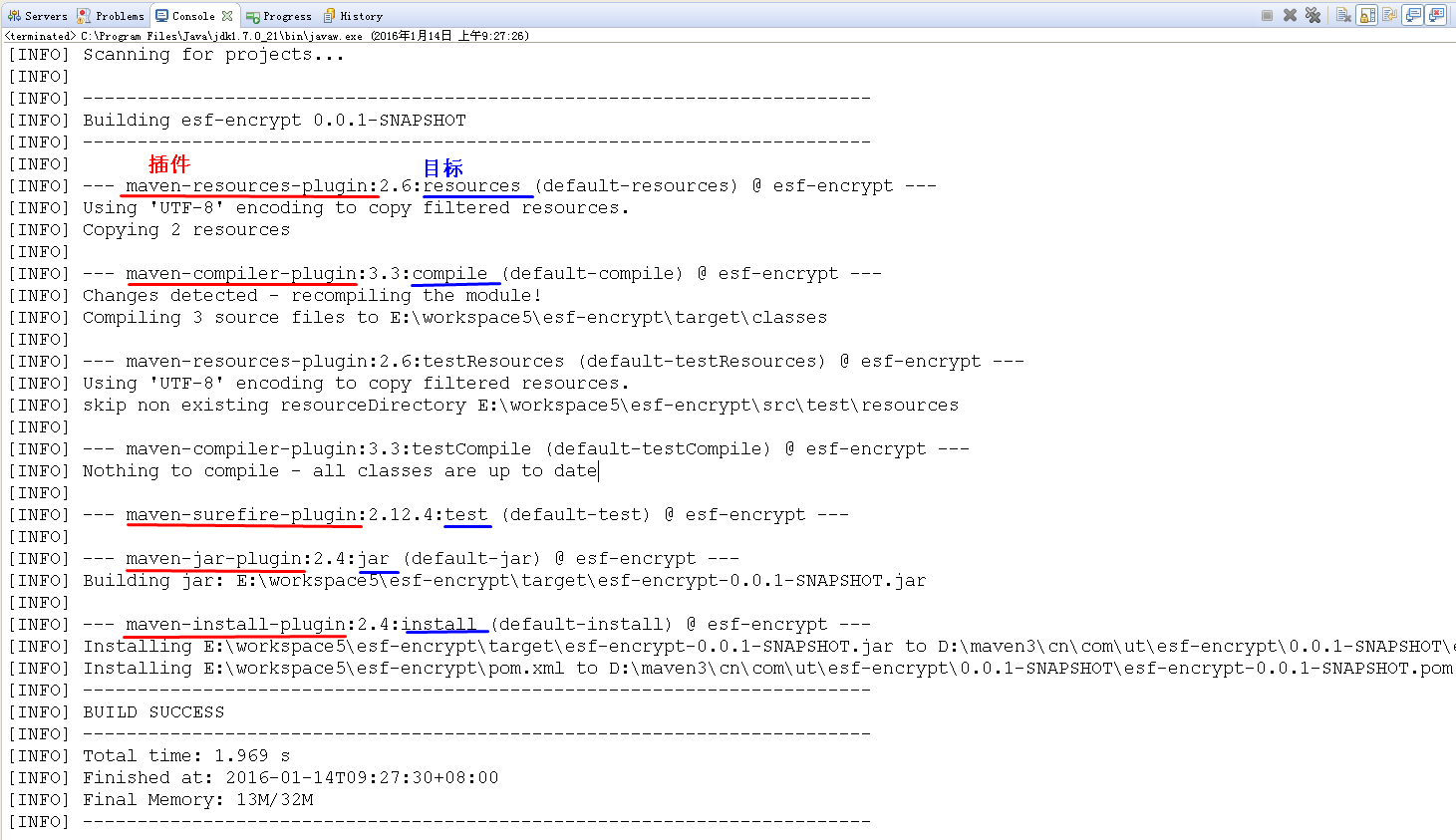
* 1. Maven的插件

Maven的核心文件很小，主要的任务都是由插件来完成。定位到：%本地仓库%\org\apache\maven\plugins，可以看到一些下载好的插件。



* + 1. 插件的目标

一个插件通常可以完成多个任务，每一个任务就叫做插件的一个目标。如执行mvn install命令时，调用的插件和执行的插件目标如下：

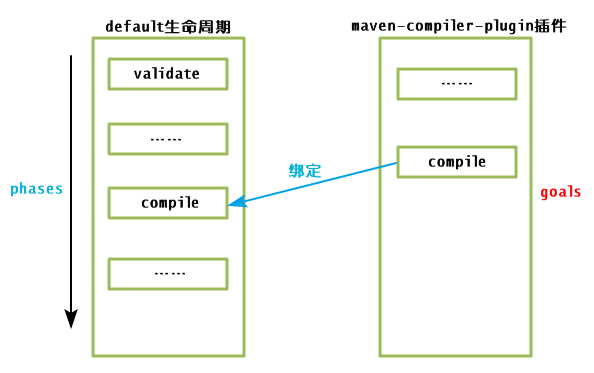


每个插件都有哪些个目标，官方文档有更详细的说明：

<http://maven.apache.org/plugins/index.html>

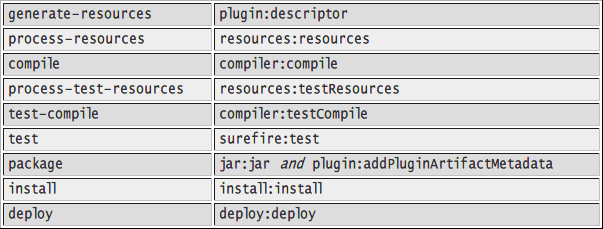
* 1. 将插件绑定到生命周期

Maven的生命周期是抽象的，实际需要插件来完成任务，这一过程是通过将插件的目标（goal）绑定到生命周期的具体阶段（phase）来完成的。如：将maven-compiler-plugin插件的compile目标绑定到default生命周期的compile阶段，完成项目的源代码编译：



* + 1. 内置的绑定

Maven对一些生命周期的阶段（phase）默认绑定了插件目标，因为不同的项目有jar、war、pom等不同的打包方式，因此对应的有不同的绑定关系，其中针对default生命周期的jar包打包方式的绑定关系如下：



第二列中，冒号后面即是绑定的插件目标，冒号前面是插件的前缀（prefix），是配置和使用插件的一种简化方式。

* + 1. 自定义绑定

用户可以根据需要将任何插件目标绑定到任何生命周期的阶段，如：将maven-source-plugin的jar-no-fork目标绑定到default生命周期的package阶段，这样，以后在执行mvn package命令打包项目时，在package阶段之后会执行源代码打包，生成如：ehcache-core-2.5.0-sources.jar形式的源码包。

|  |
| --- |
| <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-source-plugin</artifactId>  <version>2.2.1</version>  <executions>  <execution>  <id>attach-source</id>  <phase>package</phase><!-- 要绑定到的生命周期的阶段 -->  <goals>  <goal>jar-no-fork</goal><!-- 要绑定的插件的目标 -->  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build> |

* 1. 配置插件

Maven插件高度易扩展，可以方便的进行自定义配置。如：配置maven-compiler-plugin插件编译源代码的JDK版本为1.7：

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin> |

也可以对插件的各个目标进行更具体的配置：

<http://maven.apache.org/guides/mini/guide-configuring-plugins.html>

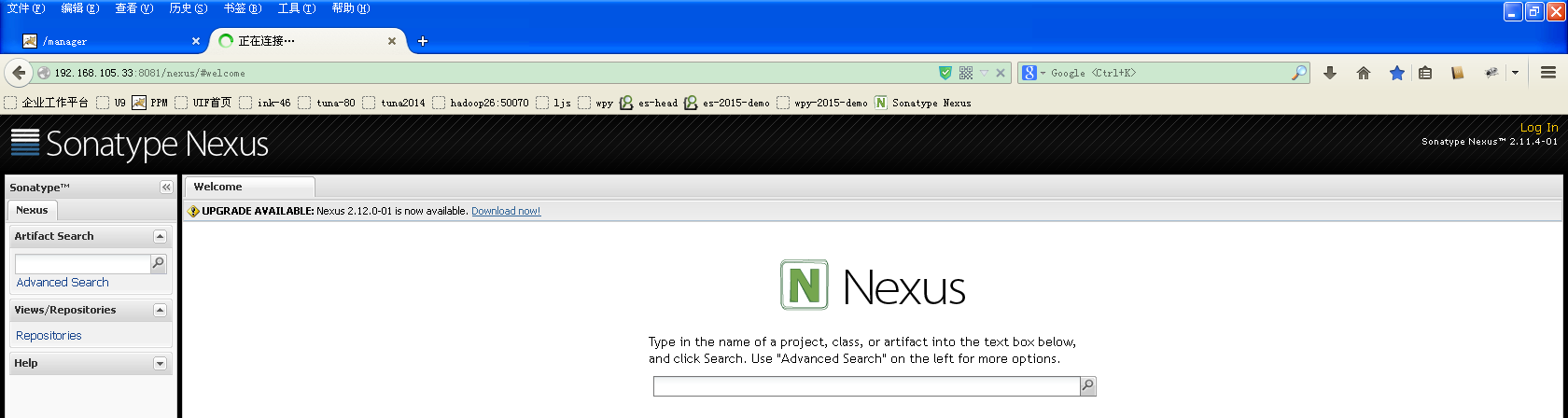
* 1. 插件仓库

跟其他构件一样，插件也是根据坐标存储在Maven仓库中。超级POM中Maven配置的默认插件远程仓库如下：

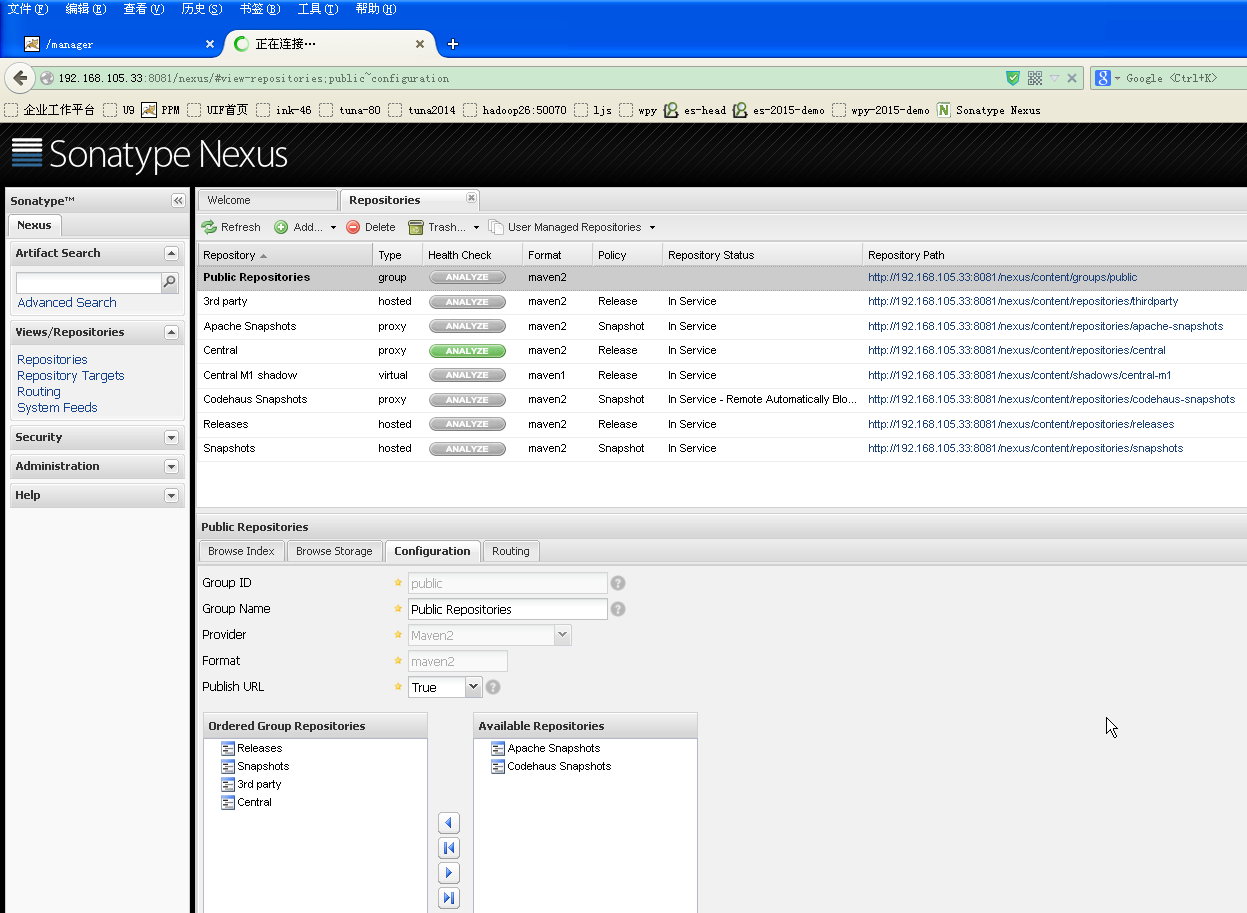
|  |
| --- |
| <pluginRepositories>  <pluginRepository>  <id>central</id>  <name>Central Repository</name>  <url>http://repo.maven.apache.org/maven2</url>  <layout>default</layout>  <snapshots>  <enabled>false</enabled>  </snapshots>  <releases>  <updatePolicy>never</updatePolicy>  </releases>  </pluginRepository>  </pluginRepositories> |

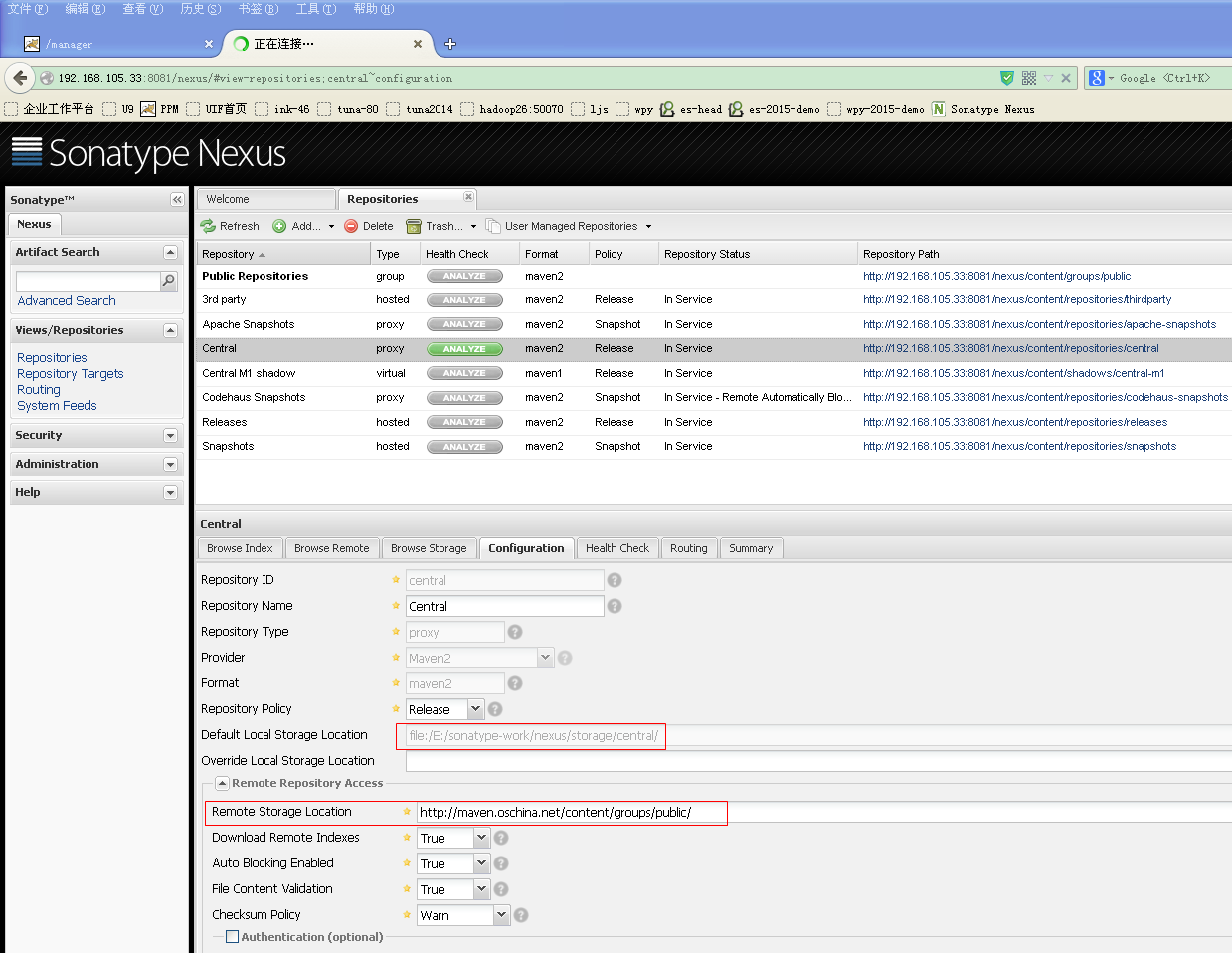
1. 私服
   1. Server

<http://192.168.105.33:8081/nexus/>

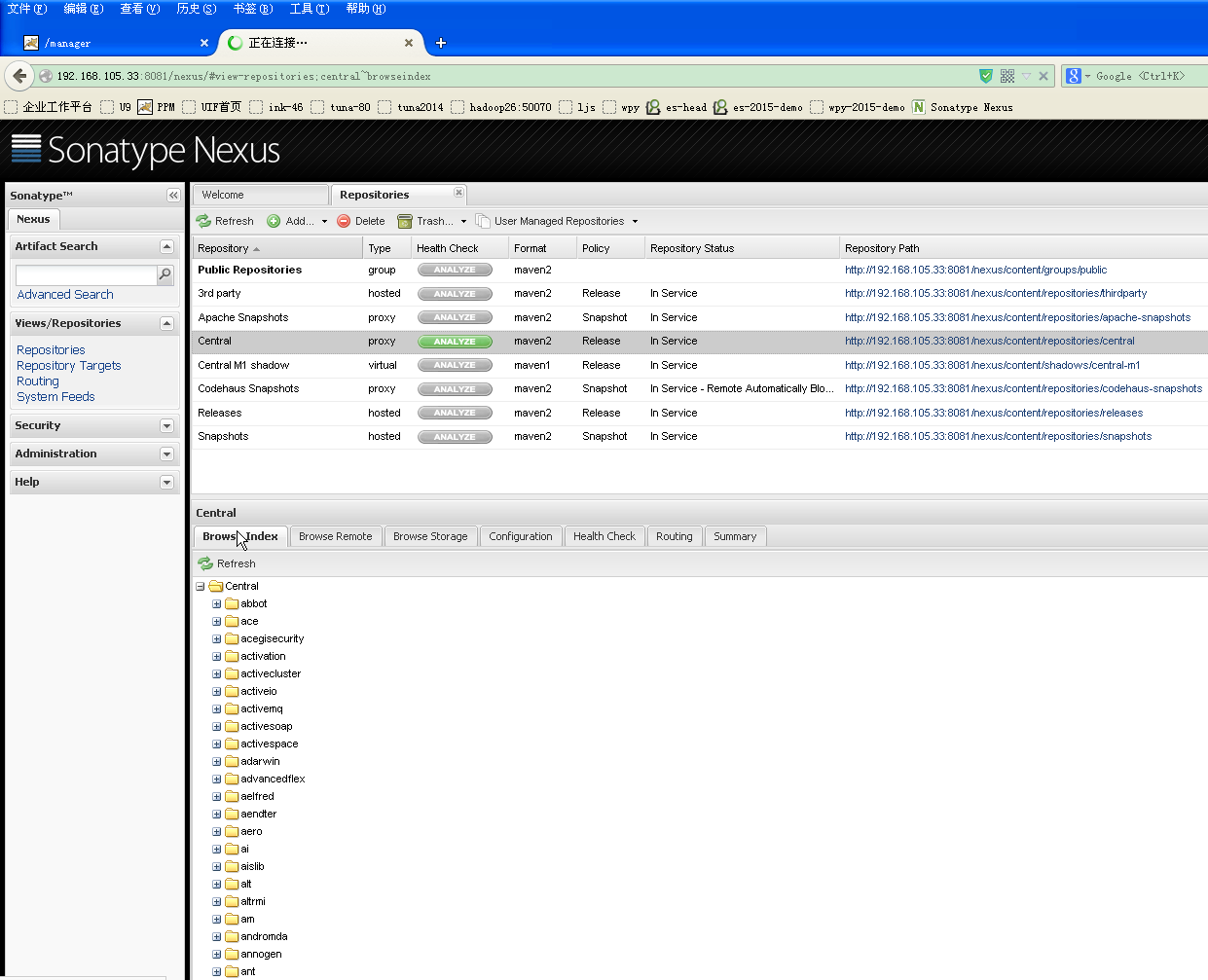


* 1. 仓库组

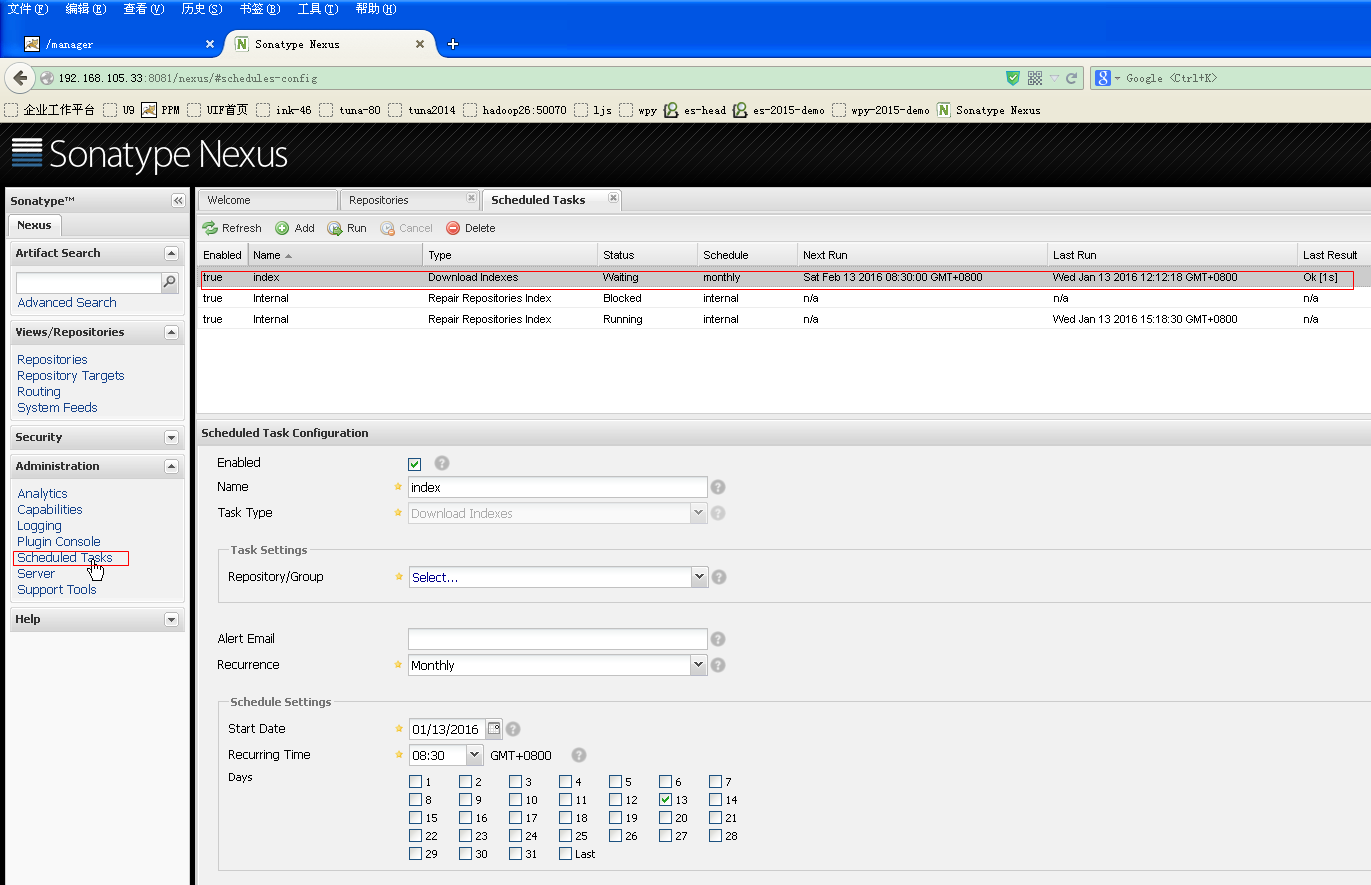




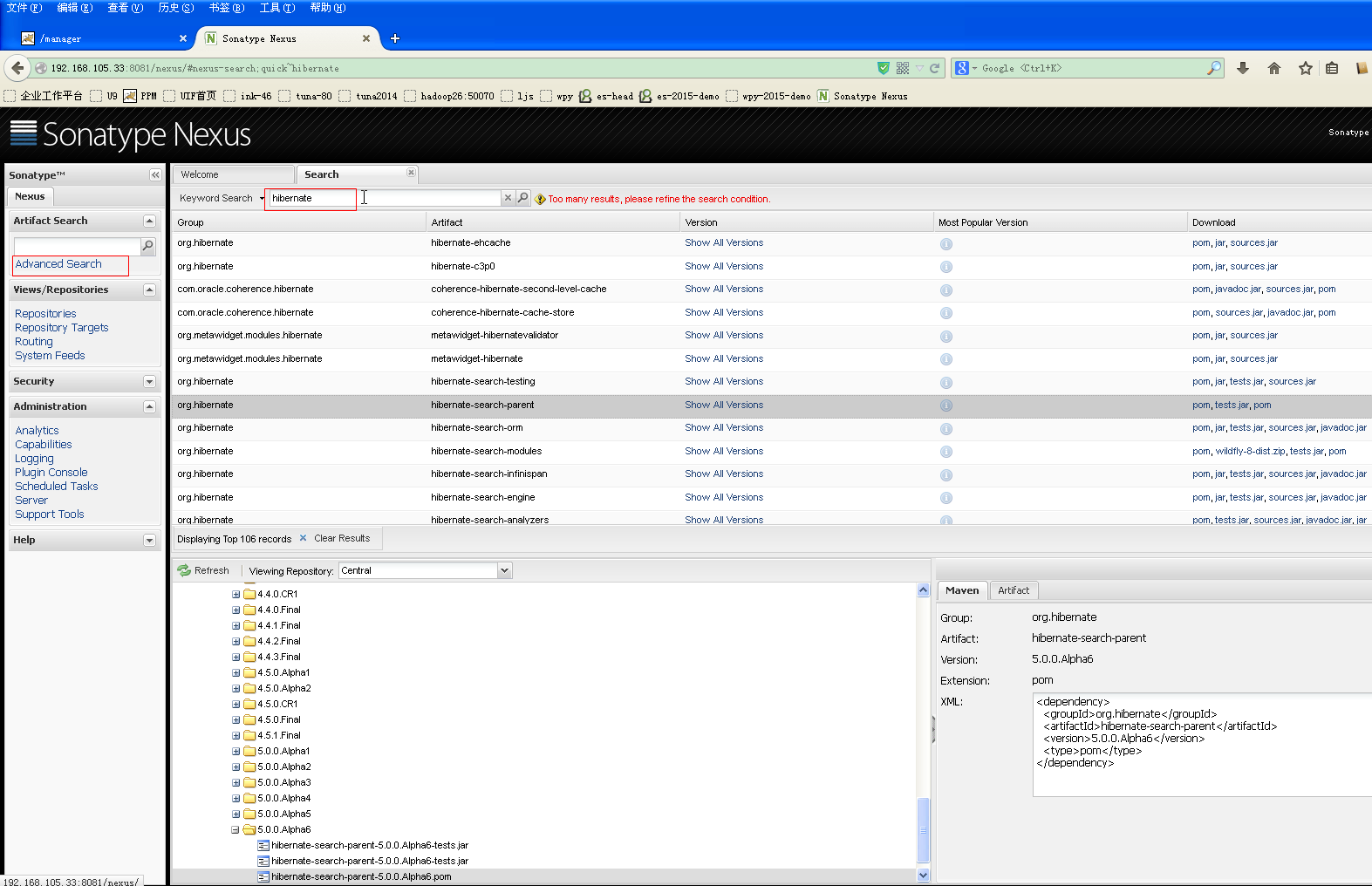
* 1. 仓库索引



下载索引



Artifact Search



1. 参考

许晓斌《Maven实战》. 机械工业出版社，2011.1