**1创建一个rcp工程**

**1．1创建一个plugin工程**

Tycho是一套可以构建eclipse plugin和OSGI bundle的Maven插件。由于Tych在最大程度上复用了RCP开发当中的MANIFEST.MF文件，因此我们可以用极少的配置就可以完成所有的构建工作。Tycho支持三种打包类型：eclipse-plugin、eclipse-feature和eclipse-repository分别对应着RCP开发当中的plugin、feature以及P2 update site\product。

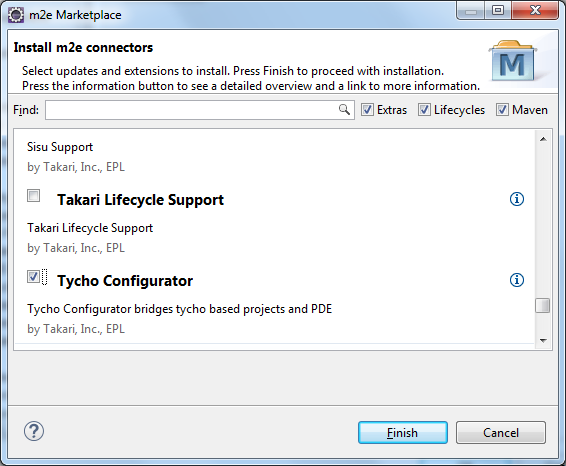
tycho会通过POM文件去读取RCP程序中的配置文件（plugin.xml和MANIFEST.MF等）。在这里要特别的提一下，因为Maven中对Artifact定义和RCP中对Bundle定义是通过不同的标准来实现的。在POM文件中我们应该确保 Artifact ID 和Version 应该等同于MANIFEST.MF文件中的Bundle-Symbolic-Name和Version。同理假如我们一个plugin的version是1.0.0.qualifier，相对应的我们pom中的version应当是1.0.0-SNAPSHOT。

在介绍之前我们需要先做一些准备工作，我们需要让我们eclipse中的Maven插件（m2e）能读的懂Tycho的配置。我们需要做如下几步：

Eclipse-luna 版本：

1. Preferences -> Maven -> Discovery -> Open Catalog

2. 在弹出的对话框中找到Tycho Configurator，然后勾选安装。如下图



经过一段漫长的等待之后，Tycho Configurator安装完毕。如果你本地已经配置了4.4的target platform，接下来我们需要用来生成一个最简单的plugin用来展示Tycho：

    1. File -> New -> Other

    2. 在弹出的对话框中选择Plug-in Project -> Next

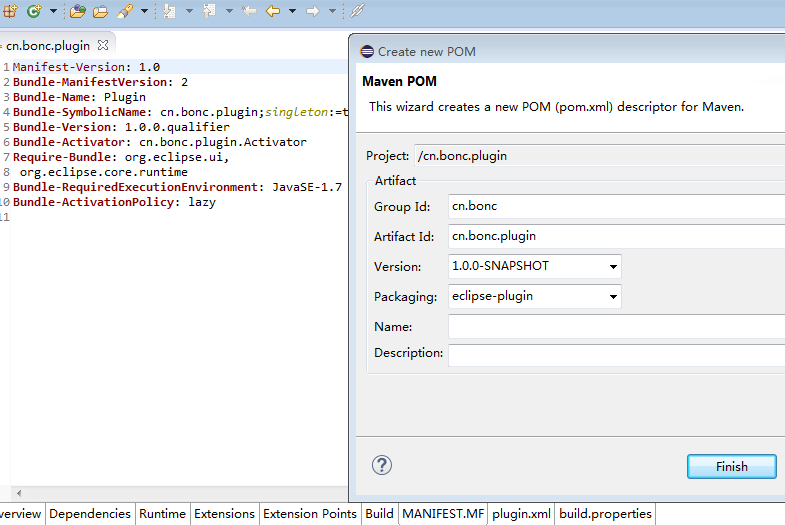
    3. 在Plugin创建的界面中输出一个项目名称-> Next

    4. 在接下来的Content对话框中，我们给出一些Plugin的properties.

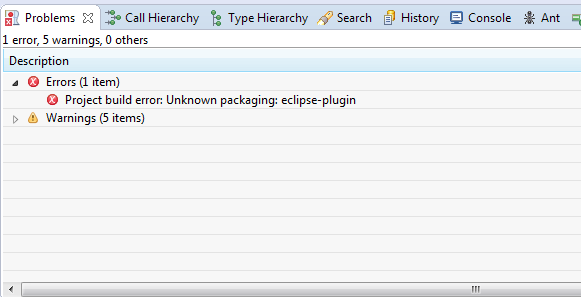
    5. 把Would you like to create a 3.x rich client application置成Yes -> Next

    6. 在Templates对话框中选择RCP Mail Template -> Finish

完成了繁琐的准备工作之后，我们终于开始进入正题。由于生成的Plugin项目并不是一个Maven的项目，我们首先需要把他Convert成为一个Maven的项目。右键项目名 -> Configure -> Convert to Maven Project。这个时候会弹出一个对话框，需要我们给出一下POM的基本信息。根据之前介绍的Tycho插件的命名规范，我们把POM配置如下：



正如文中开头所说的，POM中的Artifact ID 和Version 应该等同于MANIFEST.MF文件中的Bundle-Symbolic-Name和Version。如果bundle的version定义是以qualifier结尾，POM中的version也就应该对应以SNAPSHOT结尾。最后，我们在packaging属性中给的是eclipse-plugin，这个属性会告诉Maven，这个项目是一个plugin项目。转化成Maven项目之后，我们会发现项目并不识别eclipse-plugin这种打包方式。



这个错误是因为我们在POM文件中并没有提供Tycho的插件信息。于是我们加入如下的代码到POM中

**Xml代码  [收藏代码](javascript:void())**

<properties>

<tycho.version>0.22.0</tycho.version>

<luna-repo.url>http://download.eclipse.org/releases/luna</luna-repo.url>

</properties>

<repositories>

<repository>

<id>luna</id>

<url>${luna-repo.url}</url>

<layout>p2</layout>

</repository>

</repositories>

<build>

<plugins>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>tycho-maven-plugin</artifactId>

<version>${tycho.version}</version>

<extensions>true</extensions>

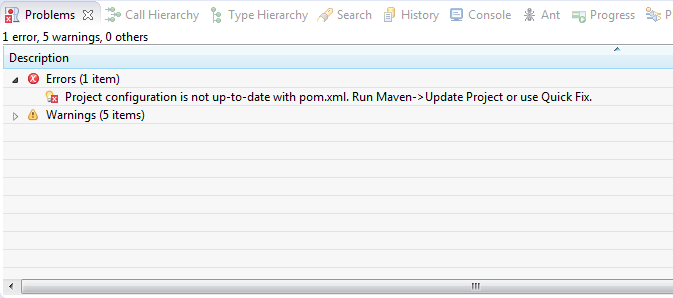
</plugin>

</plugins>

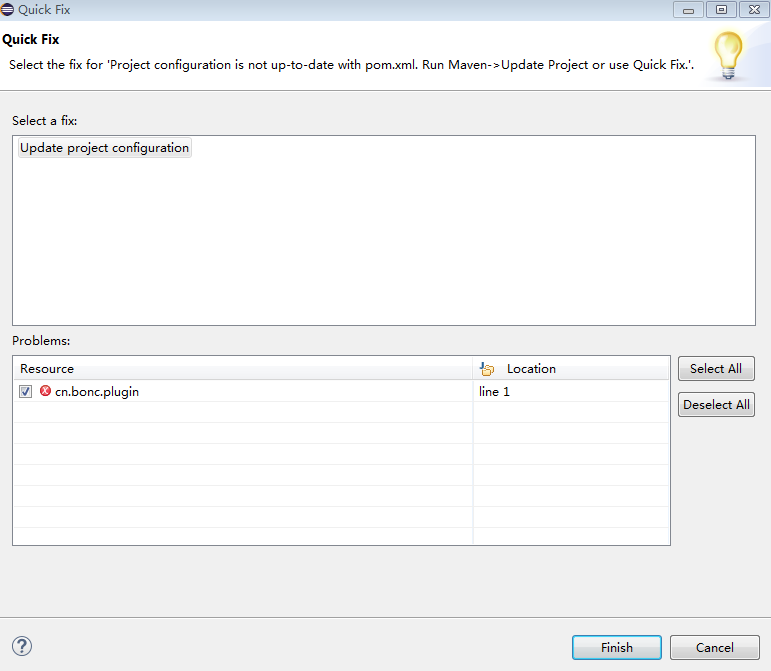
</build>

</project>

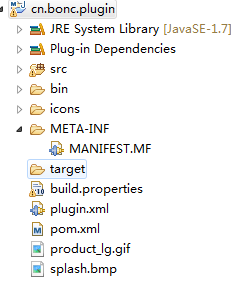
上面的这段XML中，repository中配置的是target platform的信息，在这里我们用的是eclipse 4.4 luna 版本的SDK. plugin当中就是Tycho插件的信息了。我们用的是0.22.0版本。保存POM，发现依然有错误。



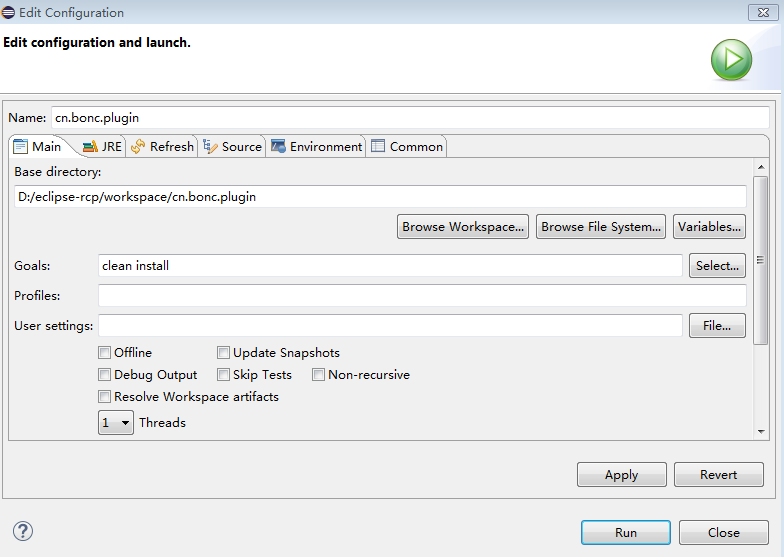
这个错误的原因是我们的项目配置并没有随着POM的更新而更新，选中这个错误然后Ctrl+1会弹出一个Quick Fix对话框，在Quick Fix对话框中，我们快速修复这个错误。



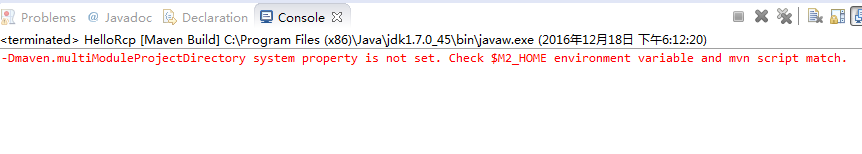
这个时候项目就应该没有错误了。



接下来我们就可以build这个plugin了，右键选择项目名-> Run As -> Manen build. 在弹出的对话框中，我们把Goals设定成为clean install



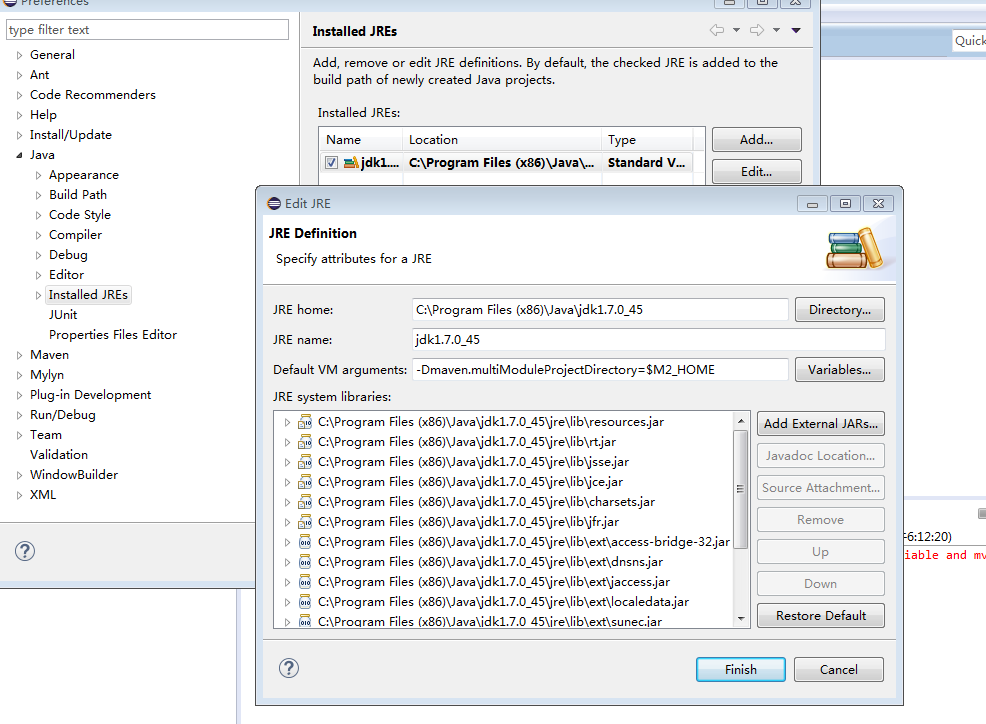
当点击run时将出现



我们需要

1.添加M2\_HOME的环境变量（已经配置好）

2.Preference->Java->Installed JREs->Edit 选择一个jdk,添加  -Dmaven.multiModuleProjectDirectory=$M2\_HOME

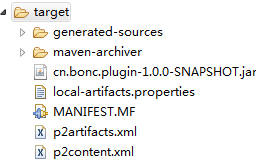


接下来我们运行这个Maven build，如果你是第一次运行，Maven会从eclipse luna P2的site上下载一些所需要的依赖到你本地的repository当中。如果你本地的repository已经包含了这些dependencies，tycho会直接build这个plugin项目并把build出来的jar安装到本地的repository当中。

**Console代码  [收藏代码](javascript:void())**

1. [INFO]
2. [INFO] --- tycho-p2-plugin:0.22.0:update-local-index (default-update-local-index) @ cn.bonc.plugin ---
3. [INFO] ------------------------------------------------------------------------
4. [INFO] BUILD SUCCESS
5. [INFO] ------------------------------------------------------------------------
6. [INFO] Total time: 30.733 s
7. [INFO] Finished at: 2016-12-26T16:46:34+08:00
8. [INFO] Final Memory: 66M/158M
9. [INFO] ------------------------------------------------------------------------

build成功之后，刷新plugin项目，会发现在target文件夹当中多出了许多的东西。



cn.bonc.plugin-1.0.0-SNAPSHOT.jar就是Tycho为这个plugin所build出来的jar包。至此，我们用Tycho构建了你的第一个plugin。

**1．2创建一个parent工程**

创建一个parent项目。为什么要创建一个parent项目呢？我们想象一下，假设我们的RCP项目中有100个plugins项目，为每个项目的POM文件都提供相同的Tycho插件的配置是一件极其不明智的行为。最好的做法就是我们把所有共用的配置都放到一个parent项目当中，这也是Maven推荐的做法。

所以，我们先来创建一个parent项目：

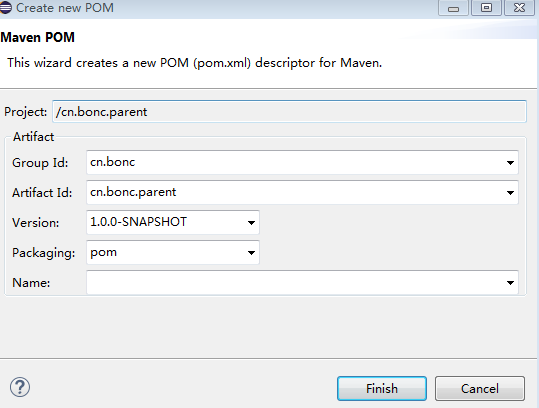
    1. File -> New -> Project

    2. 在弹出的对话框中选择 General -> Project -> Next

    3. 在New Project对话框中输入项目名：cn.bonc.parent -> Finish

    4. 右键项目名称 -> Configure -> Convert to Maven Project

5. 因为我们仍然会用到我们创建的plugin，因此我们保持相同group id 和version. 打包方式 是pom，因为他是一个父类的Maven项目，如下图。



6. 把之前创建的plugin中POM文件里的Tycho信息剪切到parent的pom当中

**Xml代码  [IMG_257](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>pom</packaging>

<properties>

<tycho.version>0.22.0</tycho.version>

<luna-repo.url>http://download.eclipse.org/releases/luna</luna-repo.url>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<repositories>

<repository>

<id>luna</id>

<url>${luna-repo.url}</url>

<layout>p2</layout>

</repository>

</repositories>

<build>

<plugins>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>tycho-maven-plugin</artifactId>

<version>${tycho.version}</version>

<extensions>true</extensions>

</plugin>

</plugins>

</build>

</project>

7. 与此同时，plugin项目中的POM文件也需要更新，plugin项目会直接继承parent项目。

**Xml代码  [IMG_258](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../cn.bonc.parent/pom.xml</relativePath>

</parent>

<artifactId>cn.bonc.plugin</artifactId>

<packaging>eclipse-plugin</packaging>

</project>

试着再build一下plugin项目，如果成功，则说明我们的parent项目已经成功创建成功。

**1．3 创建一个product工程**

product就是一个standalone的可执行程序，他可以包含很多的plugin，也可以针对于不同的平台定制不同的参数。

 对Product的概念有了一定的了解之后，我们便可以开始创建product项目了：

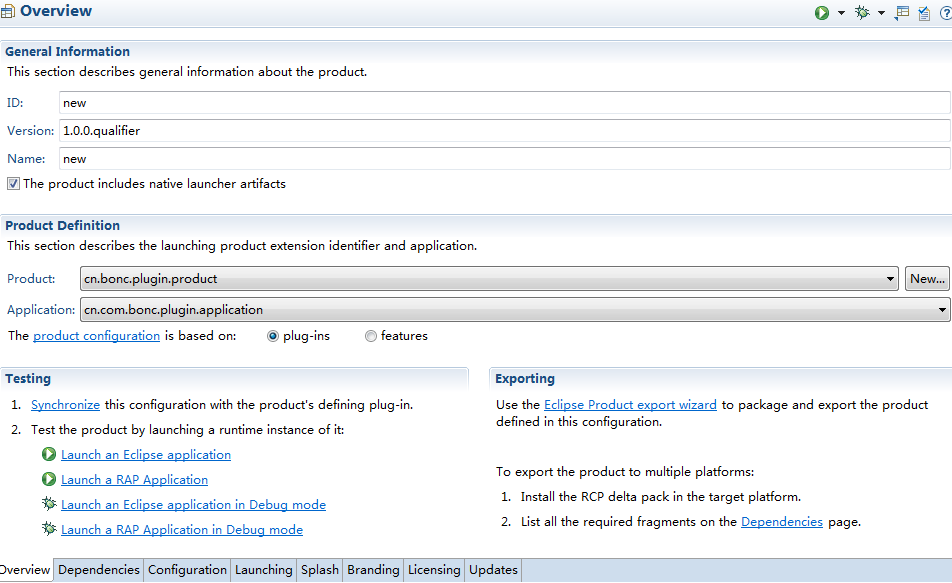
1. 首先还是 New -> Project -> General -> Project

2. 在接下来的对话框中输入我们的项目名称，这里我们用cn.bonc.product

3. 项目创建完成之后我们需要创建product文件，还是New -> Other -> Plug-in Development -> Product Configuration -> Next

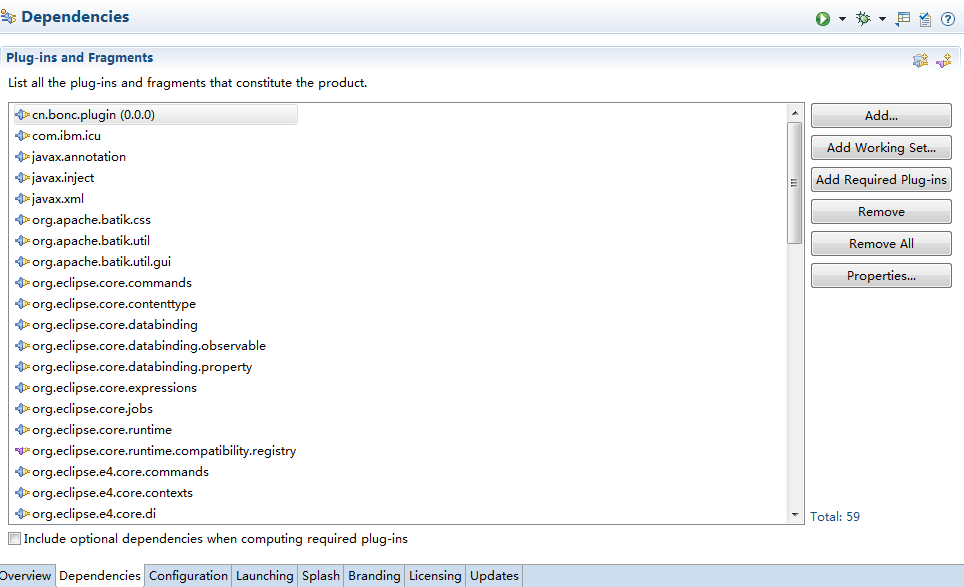
4. 在接下来的对话框中,选中我们刚刚创建的product项目，然后在下面的File Name里输入我们的product文件名，这里我们用new.product  -> Finish

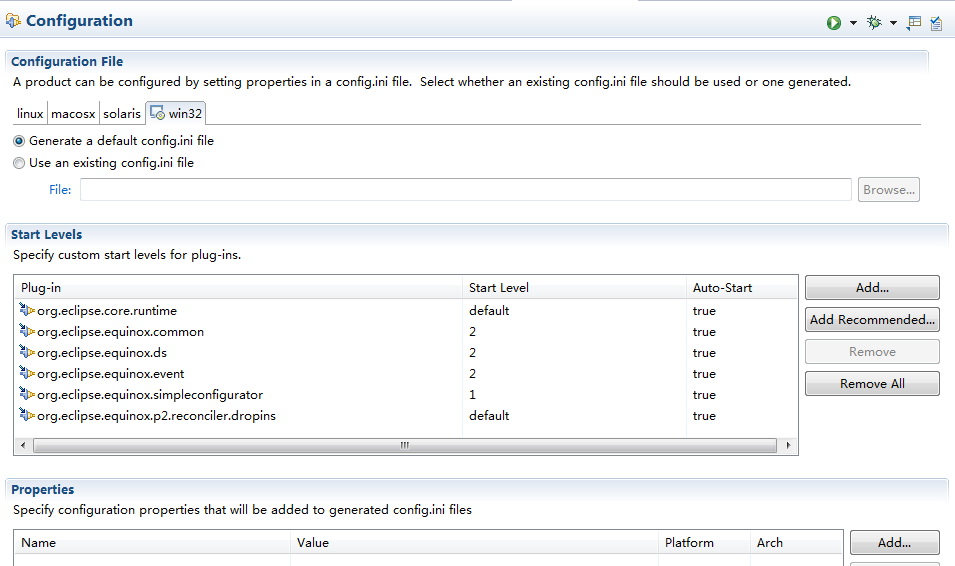
5. 在new.product配置的overview tab界面中，给出相应的信息。值得一提的是，我们需要选择这个product是based on在features上的。其次，ID是必须填写的，因为Tycho会使用ID加上你build的平台后缀组合成一个完成的文件名。



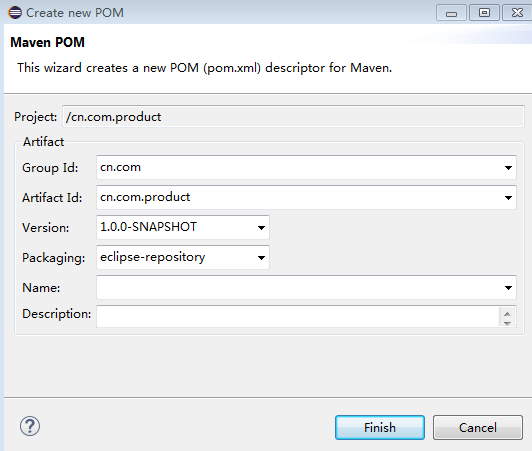
6. 在Dependencies tab界面中，把我们之前创建的plugin添加进来。然后在Run Configurations的Plug-ins选项卡里点击Add Required Plug-ins添加所需要的依赖，这是因为我们还缺少一些RCP程序所需要target platform级别的plugin。

在接下来我们就需要把我们刚创建好的Product项目变成一个支持Tycho的Maven项目。还是右键product项目名，然后把他转换成一个Maven项目，在Maven的配置框中，其他的属性和我们之前介绍的plugin和feature的命名规范类似，唯一不同的一点就是packaging变成了eclipse-repository。



7. 接下来在Configuration tab中，我们需要添加Start Levels。这么做是因为，在传统的PDE build过程中，PDE会自动添加start levels。但是Tycho不会自动添加这些配置，所以就需要我们手动添加。否则Tycho就会在build过程中出错。因为我们这里只是个demo程序，所以不做过多的考虑，直接点击Add Recommanded，使用系统推荐配置即可。****

在接下来我们就需要把我们刚创建好的Product项目变成一个支持Tycho的Maven项目。还是右键product项目名，然后把他转换成一个Maven项目，在Maven的配置框中，其他的属性和我们之前介绍的plugin的命名规范类似，唯一不同的一点就是packaging变成了eclipse-repository。



在生成的POM文件中，添加parent属性，让他继承我们之前创建的parent项目。然后添加Tycho product build相关的插件。

**Xml代码  [IMG_260](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../cn.bonc.parent/pom.xml</relativePath>

</parent>

<artifactId>cn.bonc.product</artifactId>

<packaging>eclipse-repository</packaging>

<build>

<plugins>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>tycho-p2-repository-plugin</artifactId>

<version>${tycho.version}</version>

<configuration>

<includeAllDependencies>true</includeAllDependencies>

</configuration>

</plugin>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>tycho-p2-director-plugin</artifactId>

<version>${tycho.version}</version>

<executions>

<execution>

<id>materialize-products</id>

<goals>

<goal>materialize-products</goal>

</goals>

</execution>

<execution>

<id>archive-products</id>

<goals>

<goal>archive-products</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

这个时候我们虽然告诉Tycho如何去build这个product项目了，但是我们还没有提供我们build出的程序应当支持那种平台的信息。返回parent项目，修改POM文件，添加target-platform-configuration插件

**Xml代码  [IMG_261](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>pom</packaging>

<properties>

<tycho.version>0.22.0</tycho.version>

<luna-repo.url>http://download.eclipse.org/releases/luna</luna-repo.url>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<repositories>

<repository>

<id>luna</id>

<url>${luna-repo.url}</url>

<layout>p2</layout>

</repository>

</repositories>

<build>

<plugins>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>tycho-maven-plugin</artifactId>

<version>${tycho.version}</version>

<extensions>true</extensions>

</plugin>

<plugin>

<groupId>org.eclipse.tycho</groupId>

<artifactId>target-platform-configuration</artifactId>

<version>${tycho.version}</version>

<configuration>

<environments>

<!-- <environment>

<os>linux</os>

<ws>gtk</ws>

<arch>x86</arch>

</environment>

<environment>

<os>linux</os>

<ws>gtk</ws>

<arch>x86\_64</arch>

</environment> -->

<!--<environment>

<os>win32</os>

<ws>win32</ws>

<arch>x86</arch>

</environment> -->

<environment>

<os>win32</os>

<ws>win32</ws>

<arch>x86\_64</arch>

</environment>

<!-- <environment>

<os>macosx</os>

<ws>cocoa</ws>

<arch>x86\_64</arch>

</environment> -->

</environments>

</configuration>

</plugin>

</plugins>

</build>

</project>

添加完成之后我们会发现，parent的pom中多了很多的平台信息，因为我们只需要build出来的程序是一个win32.x86的程序，所以在这里我们注释掉其他的平台信息。这个时候你也许会问，我们可以直接build product了么？答案是否定的，因为你的product项目只有一个孤零零的product文件，他最终依赖的仍然是你的feature和plugin。所以我们需要在运行时提供这些依赖。如果有非常多的plugin，然后plugin A依赖plugin B，plugin B依赖plugin C的时候，这个时候我们怎么办？难道要按照依赖关系一个一个来构建么？很显然，全部手动是一种反人类的做法。更何况你在build product的时候即便全部手动也不一定能够build成功。这个时候，我们就需要一个聚合器，偏巧我们的Maven也支持这样的做法。

**1．4在parent工程聚合其他子工程**

把plugin和product加入parent的pom当中，让他们变成parent项目的modules。

**Xml代码  [IMG_261](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>pom</packaging>

...

<modules>

<module>../cn.bonc.plugin</module>

<module>../cn.bonc.product</module>

</modules>

...

</project>

完成之后clean install  cn.bonc.parent这个项目，控制台上面的信息告诉我们build成功。

**写道**

[INFO] ------------------------------------------------------------------------  
[INFO] Reactor Summary:  
[INFO]   
[INFO] cn.bonc.plugin ........................ SUCCESS [ 1.563 s]  
[INFO] cn.bonc.product ....................... SUCCESS [ 0.186 s]  
[INFO] cn.bonc.parent ......................... SUCCESS [ 0.063 s]  
[INFO] ------------------------------------------------------------------------  
[INFO] BUILD SUCCESS  
[INFO] ------------------------------------------------------------------------  
[INFO] Total time: 25.715 s  
[INFO] Finished at: 2016-12-25T15:21:46+08:00  
[INFO] Final Memory: 74M/178M  
[INFO] ------------------------------------------------------------------------

**1．5**  **Tycho对Junt的集成**

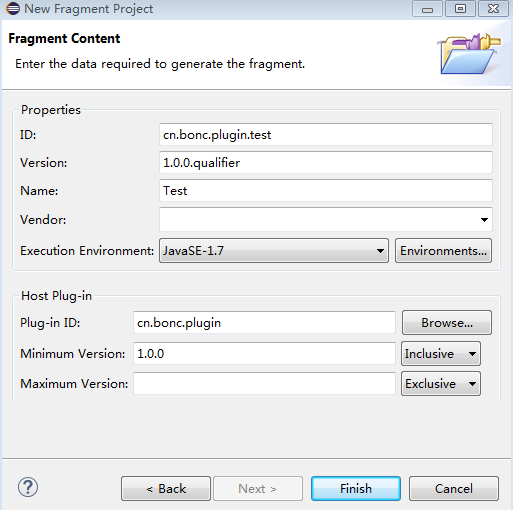
在Tyhco中对Junit的集成是通过tycho-surefire-plugin插件来实现的。从插件的名字上来看很像Maven里的surefire插件。但其实Tycho的surefire和Maven的surefire还是有一些差别的，这里的差别主要体现在生命周期上。tycho-surefire-plugin插件是在integration-test phase执行的，而maven-surefire-plugin插件是在test phase执行的。也许你对integration-test这个phase不太了解，其实这个pahse是位于我们经常做的package和install之间的。

针对于plugin项目的测试，在这里推荐的是Fragment Project。你也许会好奇为什么选择Fragment项目？原因有两点：

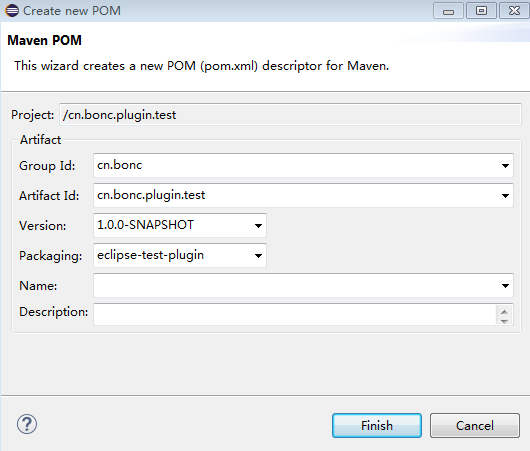
1. Fragment可以访问他的主plugin中的代码而不必去导出他们，因为他们本身就处在一个plugin当中，而且Fragment中的代码其实是是用主plugin的classloader来加载的。

2. 因为从外部来看Fragment和和主plugin项目其实是一个plugin，他们用的是一套的classloader，所以假如你的测试class和你主class是在一个包内，在我们的测试class当中甚至可以直接访问主class的私有方法。

我们通过eclipse的项目创建向导创建一个Fragment项目。因为我们测试的plugin是我们之前创建的cn.bonc.plugin，因此在这里我们的fragment的项目名取名为cn.bonc.plugin.test。在接下来的host plugin指定的对话框中我们指定cn.bonc.plugin作为host plugin



项目创建完成之后我们还是convert to Maven project。因为这是一个Junit测试项目，因此在这里我们的packaging设置成为eclipse-test-plugin



很显然，生成的pom文件会有错误，因为他识别不出来eclipse-test-plugin这种packaging方式。我们依然需要让他继承我们的parent项目的pom。

**Xml代码  [IMG_258](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../cn.bonc.parent/pom.xml</relativePath>

</parent>

<artifactId>cn.bonc.plugin.test</artifactId>

<packaging>eclipse-test-plugin</packaging>

</project>

转化成Maven项目之后，我们来添加一个junit test case。我们创建一个和host plugin一模一样的包名，然后在其之内创建一个PerspectiveTest测试类

**Java代码  [IMG_259](http://chnic.iteye.com/blog/javascript:void())**

**package** cn.bonc.plugin;

**import** org.junit.Test;

**public** **class** PerspectiveTest {

@Test

**public** **void** testID() {

assertEquals("cn.bonc.plugin.perspective", Perspective.***ID***);

}

**private** **void** assertEquals(String string, String id) {

// **TODO** Auto-generated method stub

}

}

创建完成之后我们会发现测试类报错了，那是因为我们没有把junit加到fragment项目的Require-Bundle里。ctrl+1修复编译错误。

**Xml代码  [IMG_260](http://chnic.iteye.com/blog/javascript:void())**

Manifest-Version: 1.0

**Bundle-ManifestVersion**: 2

**Bundle-Name**: Test

**Bundle-SymbolicName**: cn.bonc.plugin.test

**Bundle-Version**: 1.0.0.qualifier

**Fragment-Host**: cn.bonc.plugin;*bundle-version*="1.0.0"

**Bundle-RequiredExecutionEnvironment**: JavaSE-1.7

**Require-Bundle**: org.junit

所有的工作完成之后，我们要做的就是吧我们的测试项目添加到我们的聚合项目cn.bonc.parent中去。

**Xml代码  [IMG_261](http://chnic.iteye.com/blog/javascript:void())**

<project 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/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>cn.bonc</groupId>

<artifactId>cn.bonc.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../cn.bonc.parent/pom.xml</relativePath>

</parent>

<artifactId>cn.bonc.build</artifactId>

<packaging>pom</packaging>

<modules>

<module>../cn.bonc.plugin</module>

<module>../cn.bonc.plugin.test</module>

<module>../cn.bonc.product</module>

</modules>

</project>

Build我们的com.bonc.tycho.mail.build项目，我们会发现Maven在build的时候控制台上多出了如下的信息

**写道**

-------------------------------------------------------  
T E S T S  
-------------------------------------------------------  
Running cn.bonc.plugin.PerspectiveTest  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec - in cn.bonc.plugin.PerspectiveTest  
testID(cn.bonc.plugin.PerspectiveTest) Time elapsed: 0 sec  
  
Results :  
  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
  
[INFO] All tests passed!

控制的信息告诉我们，test case已经被执行，并且case是测试通过的。由于我们的测试fragment项目只会在integration-test phase被执行，并不会被打包到最终的product中，你在product的plugins的文件夹中也找不到相关的jar文件。

在这里还要额外的说一句，由于Tycho的surefire plugin是支持基于UI和不基于UI的测试的，如果你的test case是一个基于UI的test case，那么你需要在POM文件中，显示的指定useUIHarness为true。

Xml代码  [IMG_262](http://chnic.iteye.com/blog/javascript:void())

<project 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/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <parent>

    <groupId>cn.bonc</groupId>

    <artifactId>cn.bonc.parent</artifactId>

    <version>1.0.0-SNAPSHOT</version>

    <relativePath>../cn.bonc.parent/pom.xml</relativePath>

  </parent>

  <artifactId>cn.bonc.plugin.test</artifactId>

  <packaging>eclipse-test-plugin</packaging>

  <build>

    <sourceDirectory>src</sourceDirectory>

    <plugins>

      <plugin>

        <groupId>org.eclipse.tycho</groupId>

        <artifactId>tycho-surefire-plugin</artifactId>

        <version>${tycho.version}</version>

        <configuration>

          <useUIHarness>true</useUIHarness>

       </configuration>

      </plugin>

    </plugins>

  </build>

</project>

至此，我们也完成了Tycho对Junit的集成。cn.bonc.plugin.test项目代码，可以在附件中找到。

1.在Tycho build当中我们在控制台上发现很多的关于enconding的warning

**写道**

[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!

乱码问题想必大家都遇见过，如果在Tycho里使用UTF-8编码而不是现在的GBK呢？很简单只需要在你的parent pom的properties节点下添加如下配置文件

**Xml代码  [IMG_256](http://chnic.iteye.com/blog/javascript:void())**

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

再次build，所有的enconding的warning都已经消失不见了。

2.对于plugin项目来说，Tycho会把他build成为一个jar文件；但是如何build成一个native plugin，也就是文件夹的形式呢？

方法同样很简单，我们需要更改一下MANIFEST.MF文件，在其中添加一句配置。

**Xml代码  [IMG_257](http://chnic.iteye.com/blog/javascript:void())**

Eclipse-BundleShape: dir

再次build，原来的plugin在product的plugins文件夹内会以一个文件夹而非jar包的形式存在。

3. 如何更改product build出的文件名？我希望添加版本号或者时间戳。

你也许会说用maven-antrun-plugin不就可以做到。没错，ant当然可以做得到，但是其实Tycho也提供了他自己的方法。首先我们需要在parent pom当中定义我们的版本号和timestamp格式。

**Xml代码  [IMG_258](http://chnic.iteye.com/blog/javascript:void())**

<mail.version>1.0.0</mail.version>

<mail.timestamp>${maven.build.timestamp}</mail.timestamp>

<maven.build.timestamp.format>yyyyMMddHHmm</maven.build.timestamp.format>

maven.build.timestamp.format是Maven内置的一个properties，因为Tycho在build中识别不了Maven内置的properties（这是Tycho的一个bug）。因此我们需要再多定义个mail.timestamp来引用他。之后，我们需要更改一下product项目中的pom，添加一段archiveFileName的配置

Xml代码  [IMG_259](http://chnic.iteye.com/blog/javascript:void())

<project 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/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <parent>

    <groupId>cn.bonc</groupId>

    <artifactId>cn.bonc.parent</artifactId>

    <version>1.0.0-SNAPSHOT</version>

    <relativePath>../cn.bonc.parent/pom.xml</relativePath>

  </parent>

  <artifactId>cn.bonc.product</artifactId>

  <packaging>eclipse-repository</packaging>

  <build>

    <plugins>

      <plugin>

        <groupId>org.eclipse.tycho</groupId>

        <artifactId>tycho-p2-repository-plugin</artifactId>

        <version>${tycho.version}</version>

        <configuration>

          <includeAllDependencies>true</includeAllDependencies>

        </configuration>

      </plugin>

      <plugin>

        <groupId>org.eclipse.tycho</groupId>

        <artifactId>tycho-p2-director-plugin</artifactId>

        <version>${tycho.version}</version>

        <executions>

          <execution>

            <id>materialize-products</id>

            <goals>

              <goal>materialize-products</goal>

            </goals>

          </execution>

          <execution>

            <id>archive-products</id>

            <goals>

              <goal>archive-products</goal>

            </goals>

          </execution>

        </executions>

        <configuration>

          <products>

            <product>

             <id>Mail</id>

             <archiveFileName>Mail\_${mail.version}\_${mail.timestamp}-client</archiveFileName>

            </product>

          </products>

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

这里我们添加了两个属性：id和archiveFileName。archiveFileName很好理解，就是我们build出的文件名，那么id是什么呢？其实这里的id就是我们在Mail.product文件里定义的id，两者必须一致。配置完成之后再次build，build出的文件名已经改变。