# 目录

[目录 1](#_Toc361990453)

[版本修订 3](#_Toc361990454)

[1 预研概述 4](#_Toc361990455)

[1.1 背景 4](#_Toc361990456)

[1.2 预研目的 4](#_Toc361990457)

[1.3 预研范围 4](#_Toc361990458)

[2 Blueprint 4](#_Toc361990459)

[2.1 轻量级容器 4](#_Toc361990460)

[2.2 低侵入的编程模型 5](#_Toc361990461)

[2.3 何为BluePrint 5](#_Toc361990462)

[2.4 Blueprint规范 6](#_Toc361990463)

[2.5 Blueprint主流实现 7](#_Toc361990464)

[2.6 BluePrint Container 7](#_Toc361990465)

[3 开发环境与工具 7](#_Toc361990466)

[3.1 开发环境简介 8](#_Toc361990467)

[3.2 准备一系列基础bundle 8](#_Toc361990468)

[3.3 建立Target Platform 9](#_Toc361990469)

[3.4 使用Eclipse开发Blueprint Bundle 12](#_Toc361990470)

[3.5 在Eclipse中运行调试Bundle 18](#_Toc361990471)

[3.6 单元测试 19](#_Toc361990472)

[3.6.1 Junit4OSGi 19](#_Toc361990473)

[3.6.2 Pax Exam 19](#_Toc361990474)

[3.7 导出可部署的Bundle 20](#_Toc361990475)

[3.8 OSGi Bundle的部署与安装 22](#_Toc361990476)

[3.9 Bundle的自动更新 22](#_Toc361990477)

[3.10 为blueprint应用在eclipse中提供xml内容提示 23](#_Toc361990478)

[4 结论 24](#_Toc361990479)

[5 总结 24](#_Toc361990480)

# 版本修订

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 描述 | 修订人 |
| V1.0 | 2013-6-18 | 创建 | 陈绪绍 |
| V1.1 | 2013-7-1 | 修正格式 | 陈绪绍 |
| V1.2 | 2013-7-16 | 增加target platform，blueprint bundle等术语解释 | 陈绪绍 |
| V1.2.1 | 2013-7-17 | 调整部分章节格式 | 陈绪绍 |
| V1.2.2 | 2013-7-18 | WMS变更为物流平台 | 陈绪绍 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 预研概述

## 背景

基于物流平台产品前期所定的有关OSGi的基本框架选型：Apache felix + Apache Aries。

## 预研目的

对当前团队来说，OSGi是一个较新的技术，涉及较为复杂的概念。另外，由于OSGi从4.0之后才逐步开始制定企业级应用相关的规范，相关的技术实现相对滞后，所以有必要对相关技术进行预研。以尽量减小后期开发风险。

## 预研范围

1. 研究与实践基于Eclipse平台进行OSGi开发所涉及的相关内容。包括开发环境搭建、开发步骤、调试方法、测试支持、打包部署等内容。
2. 验证blueprint应用所需最小运行环境(即bundle集合)。

# Blueprint

## 轻量级容器

2003年左右，轻量级或者控制反转（IoC）容器开始出现，如Spring和Apache Avalon。这一系列IoC容器背后的主要思想是，通过优先使用接口避免使用具体类型的方式简化组件的配置和组装。与Ioc容器联合使用的还有依赖注入技术，该技术使得组件依赖于接口类型，接口的实现被注入到组件的实例中。OSGi服务进一步改进了类似的基于接口的方式，但它采用了一种服务定位器（service-locator）模式，并以此来打破组件之间对具体实现的依赖，这与Apache Avalon类似。

与此同时，Service Binder项目为OSGi组件提供了依赖注入框架。由此不难理解为何会出现对二者的比较。无论如何，OSGi对基于接口的服务以及服务定位器模式的使用远超前于当前趋势，没有哪一项技术能够像OSGi那样提供一种完备的动态模块层。现在IoC厂商正经历着重大转变，将他们的基础架构向OSGi框架迁移。

## 低侵入的编程模型

程序员通常需要通过模块化系统API（比如OSGi框架的API）的调用，来实现运行时的有关特性。对API的依赖使得模块非常重量级，脱离了运行时系统，难以测试和执行。为了减少对模块化系统API的依赖，需要提供一个抽象层次，让代码无需直接调用框架API。

事实上，目前也有一些这样的框架和技术，比如[OSGi Blueprint Service](http://elevenframework.org/article/7289212)，Declarative Service， Spring Dynamic Modules和iPojo。这些框架允许你基于POJO来实现模块化的系统，所提供的编程模型使得模块化系统的编程和测试更加方便。

Blueprint允许开发者使用OSGi技术而无需考虑其编程模型。基于Blueprint，我们的代码可以独立于OSGi的API。

## 何为BluePrint

OSGi 4.2 规范定义了Blueprint，该技术其实就是Spring Dynamic Modules的演化和标准化。Blueprint容器（提供依赖注入和反转控制的容器，类似于Spring的功能）规范http://eclipse.org/proposals/gemini/，这一项目将主要实现OSGi企业专家组（ OSGi Enterprise Expert Group）制定的各种规范。目前，它主要由SpringSource和Oracle领衔。从目前公开的信息来看，这一开源项目主要专注于OSGi的如下两方面问题。

其一，如何将成熟的Java EE技术集成到OSGi环境中。

其二，实现OSGi 4.2中的各种企业规范，比如OSGi Web容器、 Blueprint Services等；

可以认为，这两个问题是OSGi同Java EE融合的最后两道障碍，一旦消除它们，则企业级Java将迎来新的春天。到那时，不仅是OSGi的春天，也是Java EE的春天。

简单的说，blueprint是一套规范。和Servlet、JSP等J2EE规范一样，是一个约定意义上的东西。最新的OSGi Service Platform Release 4 V4.2 规范包括了一个名为 Blueprint Container 的规范。OSGi企业专家组把 Spring DM 引入的依赖注入 (Dependency Injection) 技术在OSGi服务平台的应用方法进行了定义，为 Bundle 提供一个基于 XML 编写的配置文件，位于路径“OSGI-INF/blueprint/”下。在这个配置文件中，定义了 Bundle 中 Bean 对象的初始化方式以及它们之间的关系，同时，也声明了其需要引用的服务对象和打算发布的服务。拥有这样一个配置文件的 Bundle 可以称之为 Blueprint Bundle，它可以被 Blueprint Container 发现并进行相关的操作。Blueprint Container 规范为OSGi在企业级的发展提供了更大的便利。使用 Blueprint Container 发布OSGi服务有以下好处：

* 以类似 spring 的方式管理对象 (Dependency Injection 方式 )
* 通过配置发布获取服务，发布获取方式更加灵活
* 更好的OSGi平台
* 更先进的IOC（ Inversion of Control）容器功能
* 轻量级、透明，基于POJO的编程模型

## Blueprint规范

Blueprint允许开发者使用OSGi技术而无需考虑其编程模型。基于Blueprint，我们的代码可以独立于OSGi的API。

[**Container Indepedence**](http://elevenframework.org/article/7289355)

这正是blueprint带给我们的：让我们的代码不依赖于OSGi的API。

[**External Configuration**](http://elevenframework.org/article/7289356)

Blueprint支持我们定义模块的配置，因此我们可以外部定义模块的服务，而无需让模块或者暴露为服务的类与OSGi API耦合。

[**Implementation Factory**](http://elevenframework.org/article/7289308)

Blueprint提供了一个OSGi Service的工厂，支持这些服务的运行时自动动态连接，比如你可以像注入一个bean一样注入一个OSGi Service

## Blueprint主流实现

* Apache Aries
* Eclipse Gemini Blueprint Gemini前身就是SpringSource捐献的Spring DM 2.0

## BluePrint Container

Blueprint Container规范定义面向服务的组件模型和Spring紧密联合起来，为 OSGi 定义了一个依赖注入（dependency injection）框架。这个规范的基础来自与Spring Dynamic Modules项目。它的目的是处理OSGi的动态特性，即服务可以在任何时间变得可用和不可用。该规范的另一个意图是处理普通旧 Java 对象（POJO），这样相同的对象就可以用于 OSGi 框架的内部和外部。定义并描述应用程序各个组件的 Blueprint XML 文件对Blueprint 编程模型十分重要。规范描述了组件如何被实例化，以及如何相互连接在一起形成一个可以运行的应用程序。一旦扩展器确定某个包是Blueprint包（需要有一些配置文件）后，它将为这个包创建一个Blueprint Container。这个Blueprint Container 负责完成以下操作：

* 解析 Blueprint XML 文件
* 实例化 BluePrint Container
* 将组件连接在一起

# 开发环境与工具

Eclipse本身提供了一个用于插件开发的工具(PDE)，只提供最基本的建立插件工程、运行、调试，导出等功能，对Equinox框架的支持最好，其它的框架，通过一些设置，也能够为基本的开发提供支持。提供更好的向导功能支持的eclipse插件目前较少，一些能用的(Pax Runner)，其官方提供的update site已经失效。另外，IBM WAS开发工具倒是比较合适，但其是收费的。

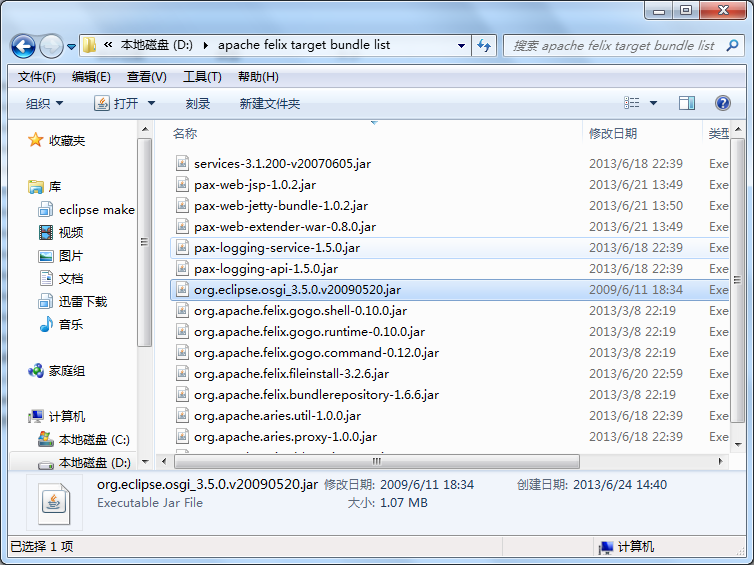
## 开发环境简介

您需要下载一个支持插件开发的Eclipse。安装 Eclipse 3.5或以上版本或Spring tool suit。

## 准备一系列基础bundle

为建立Target Platform作准备。在磁盘任意位置新建一个目录，将所有基础Bundle拷贝至该目录。目前，使用Eclipse开发调试OSGi应用，只能使用Equinox内核(org.eclipse.osgi\*.jar)。我们可以在开发调试阶段使用Equinox内核，在导出之后切换至Felix内核。

主要包括org.eclipse.osgi\*.jar(以Equinox作为内核，使用eclipse，只能以Equinox为内核)，最小的blueprint相关bundle，日志相关bundle， web支持的bundle，JNDI，JPA相关bundle。bundle列表一开始不要求是大而全的，在后期可以逐步加入。相关的bundle集合，在其它文档中有描述。比如：

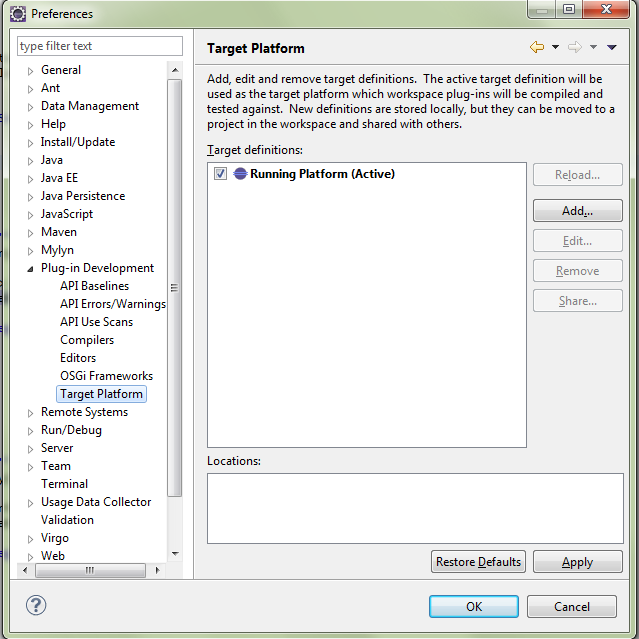


## 建立Target Platform

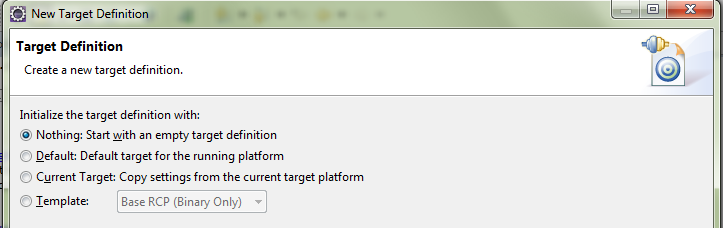
在eclipse上搭建一个OSGi的集成开发环境，实际上就是构建一个eclipse的目标平台(target platform)，这个目标平台就是我们今后开发OSGi应用的运行环境。

可以按照自己的需要去定制一个bundle集合(只包含OSGi开发所必须要用到的Bundle，此外，剔除掉了一些不需要的bundle来达到一个最干净的OSGi运行环境)，然后再利用eclipse的构建目标平台的功能（target definition）去关联到我们定义的这个bundle集合上，来完成一个自定义的OSGI开发平台的构建。在Eclipse中可以建立多个目标平台可以，可以随时切换目标平台(切换不同的bundle集合)，但同一时刻只允许一个目标平台存在，OSGi应用默认在当前目标平台下运行。

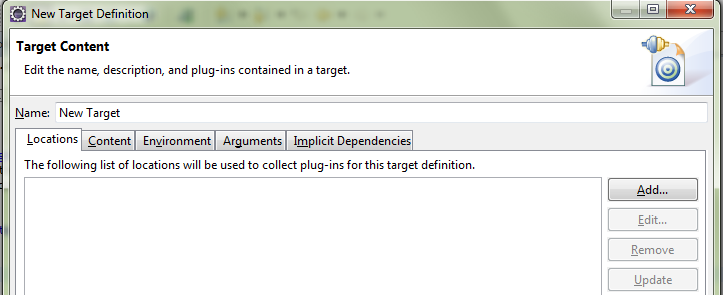
进入Eclipse的window->preferences界面，选中Plug-in Development下的Target Platform：



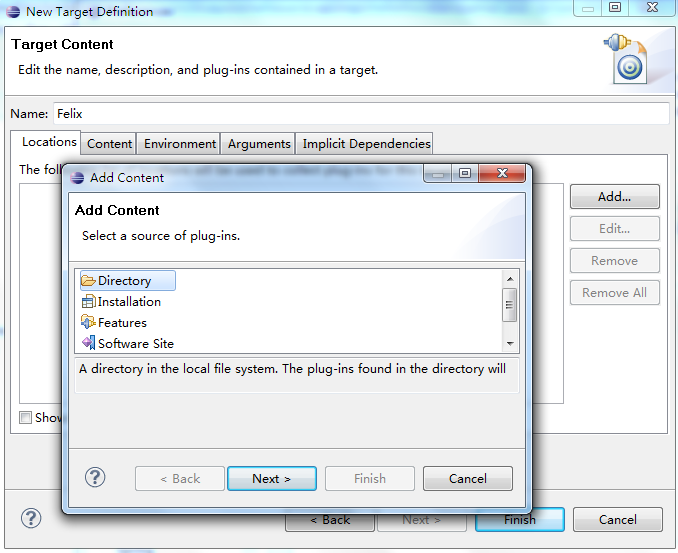
现在右边只有一个Runing Platform的，我们任务是点击“Add…”按钮来增加一个platform，进入如下界面：



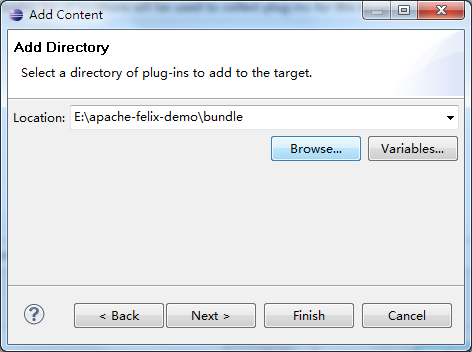
选择默认的第一个就好，点击next。



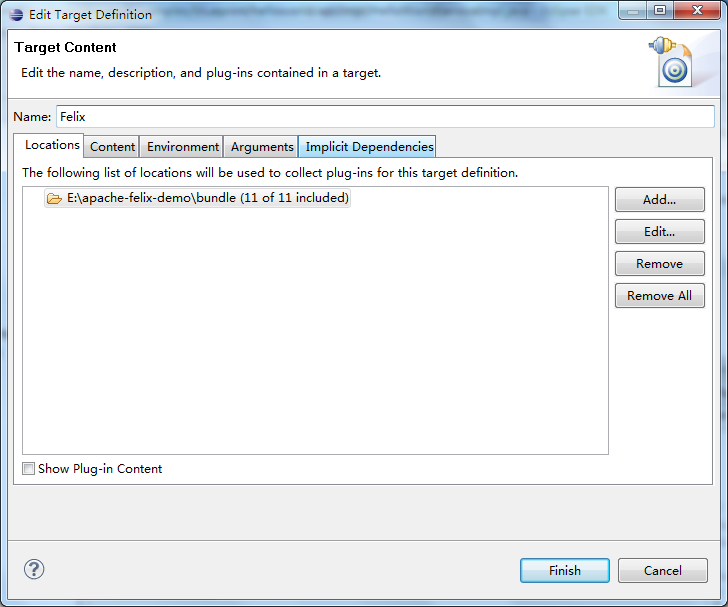
重命名为Felix(可以任意命名)。再点击这里的“Add…”：



选择“Directory”，选择准备的包含有你所准备好的bundle的目录。

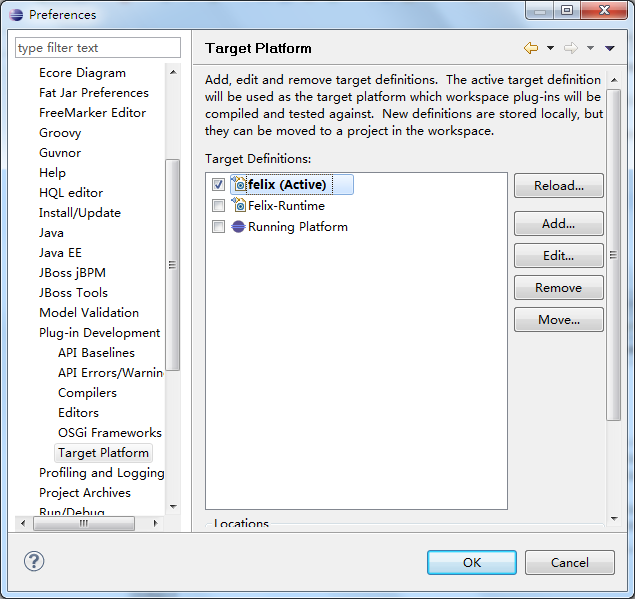


然后点击finish，回到刚才的界面：



这时候你就会发现里面多出来了你刚刚设置的路径，路径后面描述的“11 of 11 included”则就是说的我们放置的felix的jar包。

继续点击finish，回到最开始的界面：



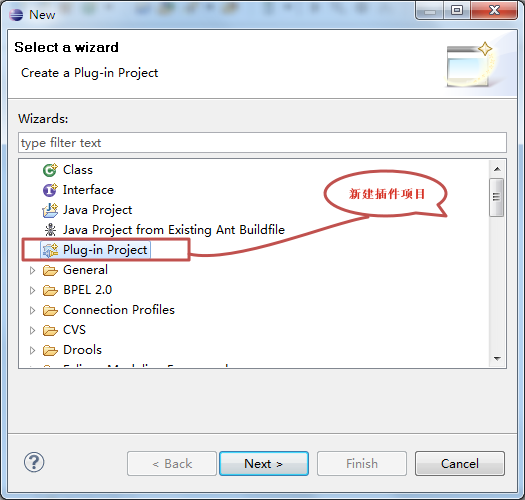
这时候多出来了一个新的target platform，即Feix，勾选上，然后确定。至此，基于Apache Aries的开发环境搭建完成。

## 使用Eclipse开发Blueprint Bundle

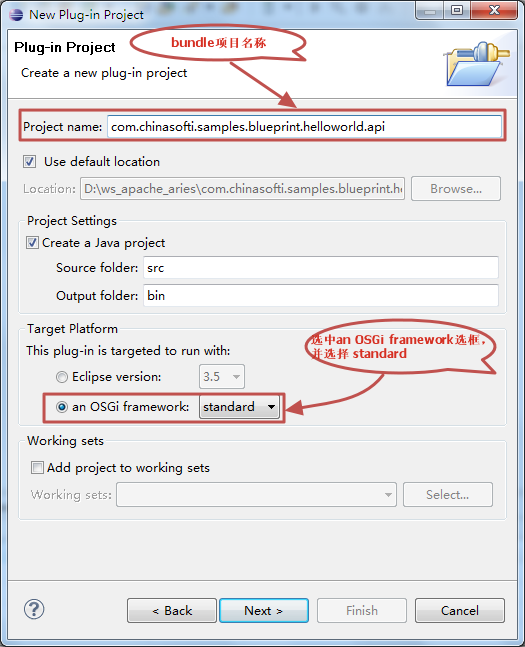
对一个OSGi Bundle而言，在bundle根目录的/OSGI-INF/blueprint 路径下有\*.xml配置文件，该bundle就可以被称之为 Blueprint Bundle，它可以被 Blueprint Container发现并进行相关的操作。默认的配置文件为/OSGI-INF/blueprint 路径下的 blueprint.xml 文件。非默认路径下的 blueprint.xml 文件，可以通过 MANIFEST.MF文件中通过Bundle-Blueprin进行指定。

利用Eclipse开发一个BluePrint Bundle步骤如下：

1. 新建插件项目

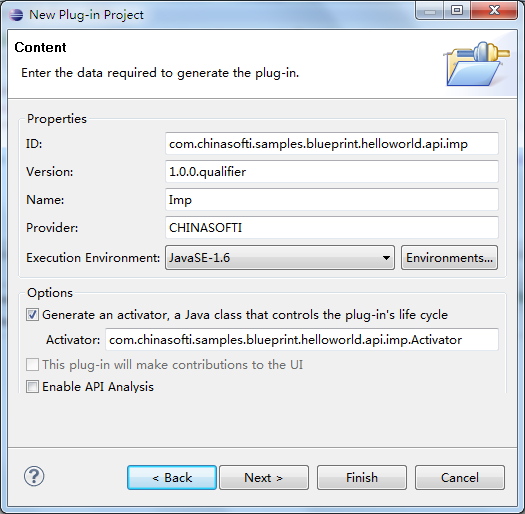


1. 选择an OSGi Framework

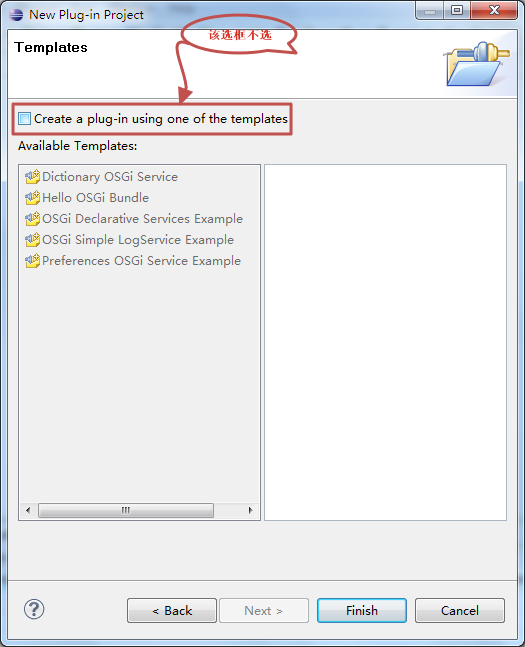


1. 建立纯净的OSGi Bundle工程模板

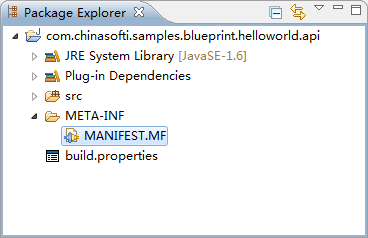
设置Bundle的相关信息。如ID，版本，提供者，运行环境及启动器等。



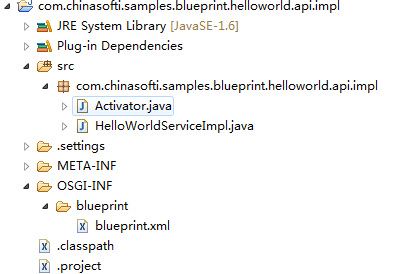
点击“下一步”



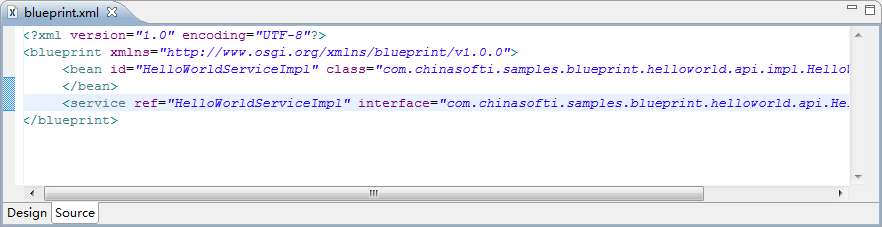
Bundle工程的基本目录结构



1. 工程根目录下建立OSGI-INF/blueprint目录，并在该目录下新建blueprint.xml文件

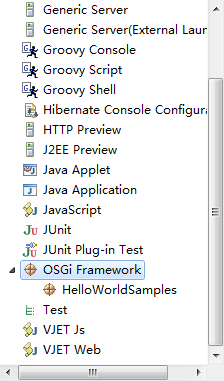


1. 编辑OSGI-INF/blueprint/blueprint.xml文件，进行bean注入，服务发布，引用等

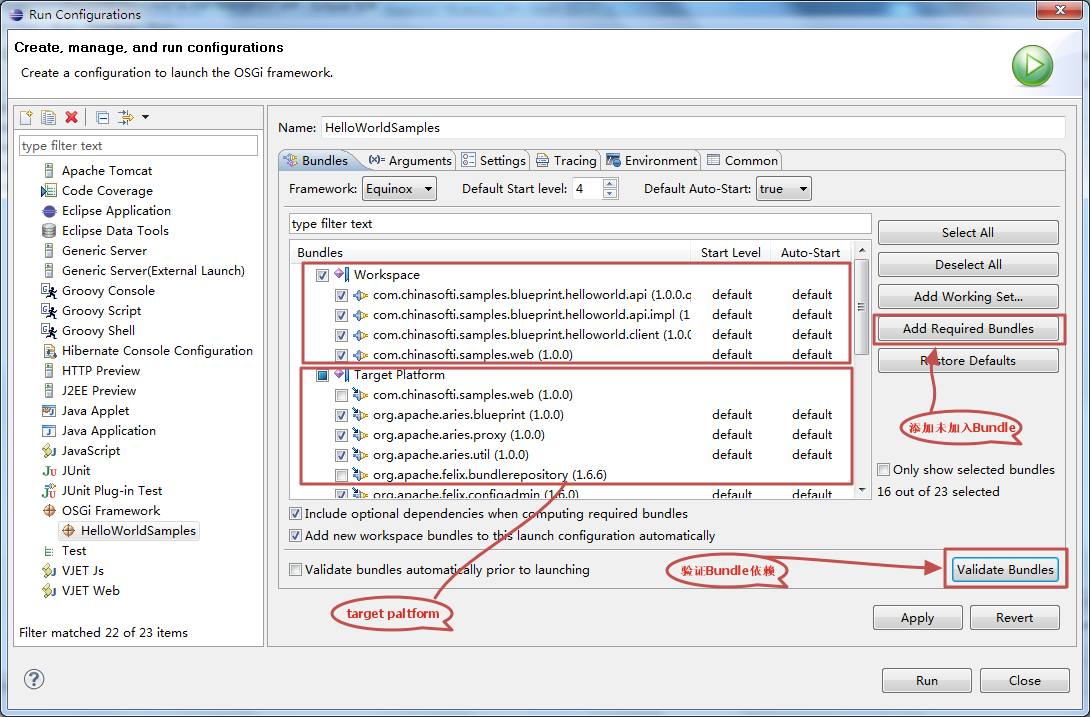


## 在Eclipse中运行调试Bundle

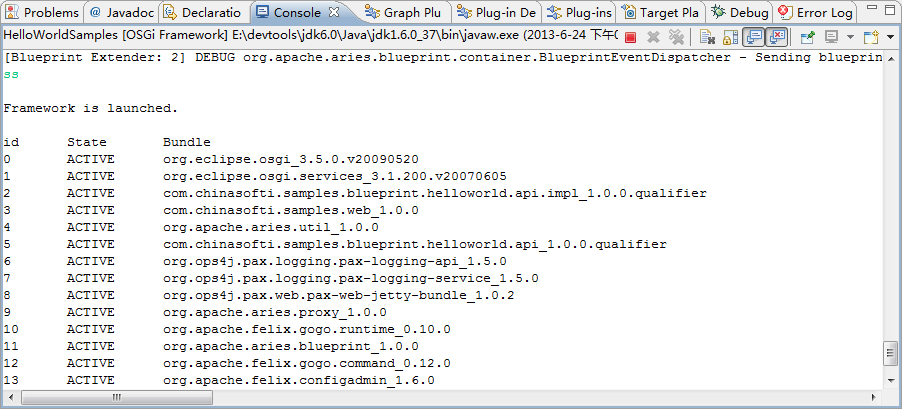
1. 运行：Run-->Run Configurations 调试：Debug--> Debug Configurations
2. 双击点击OSGi Framework，或右键-->”New”。新建一个Run configuration节点。



弹出如下界面：



点击Run，即可在Eclipse平台运行或调试Bundle。



## 单元测试

### Junit4OSGi

Junit4OSGi继承于junit的简单测试框架，可以在OSGI运行环境中进行测试。针对集成测试或是需要在一个OSGI环境中进行单元测试。需要maven支持。

### Pax Exam

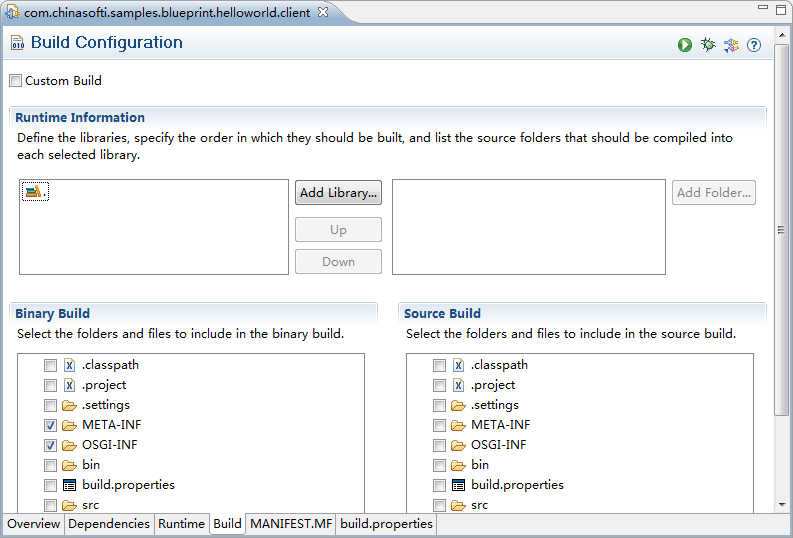
Pax Exam是由OPS4J提供的用来测试OSGi应用的工具。支持Equinox，Felix，Karaf等框架。需要maven支持。

它负责的自动化case执行顺序大致如下：

1. 启动一个OSGi的容器，比如equinox，flex，这是基础，因为所有的OSGI组件都要生存在一个OSGI容器之上。
2. 启动需要用到的所有bundle，包括被测bundle，以及被测bundle依赖的bundles
3. 动态的将test代码project创建为一个bundle
4. 把test bundle发布到已经启动了的OSGI容器内执行test函数

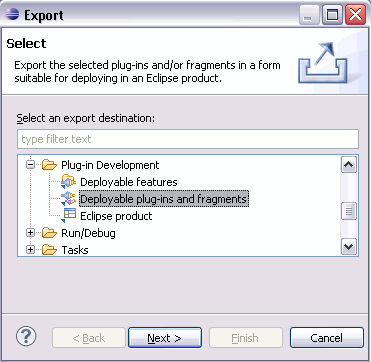
## 导出可部署的Bundle

打开bundle中的MANIFEST.MF文件,选择build页签，然后勾选要导出的内容。

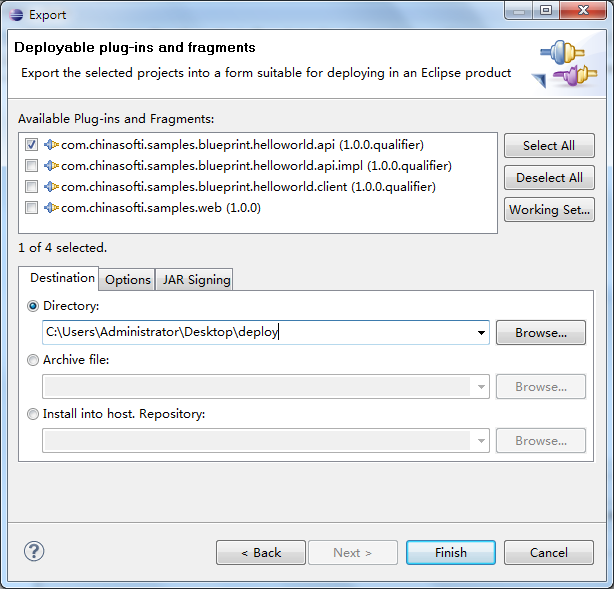


在build标签中选中将OSGI-INF导出，这一点很重要。

 Select your bundle and choose File -> Export -> Plug-in Development -> "Deployable plug-ins and fragment"。



选中Deployable plug-ins and fragment



勾选需要导出的bundle，点击finish即可。

## OSGi Bundle的部署与安装

以Apache Felix框架为例。将导出的bundle拷贝至FELIX\_HOME\bundle目录下，启动Apache Felix框架，即完成了部署。Bundle的自动安装与更新，参见4.7章节。如果框架已经启动，则可通过gogo shell或web console来进行bundle的安装与部署。当然，如果你使用了fileinstall bundle，则不需要重启框架即可完成bundle的安装部署。

## Bundle的自动更新

 OSGi的最大优点就是能实现热部署（hot deployment)，所谓热部署，就是在不停止系统运行的情况下（或者说在不影响用户体验前提下）动态更新其服务内容（和硬件领域的热插拔的概念很类似）。我们可以在OSGi控制台手动更新bundle（通过update,refresh,install等命令），现在有个新问题：如果对某个bundle需要更新时，需要手动把这个新版本的bundle拷贝到原来bundle的目录下，然后在手动在OSGi框架下手动点击update，refresh等命令进行更新。对于需要频繁更新bundle或者更新的bundle数量比较多的场合，未免太麻烦了，有没有办法让系统自动检测bundle更新情况，并安装运行，从而实现系统真正意义上的自动运行呢。答案是可以。

    各个OSGi框架都提供了相应的Bundle来是实现这个功能（Apache Felix下的File Install； Knopflerfis下的 Directory Deployer）。使用非常简单。以Apache Felix为例。

* 1. 在FELIX\_HOME目录下新建名为文件夹load。
  2. 将org.apache.felix.fileinstall-3.2.6.jar，拷贝至FELIX\_HOME\bunlde目录
  3. 运行felix框架。
  4. 当我们要批量更新自己部署的bundle时，只需将新版本复制到load文件夹中，系统将会以固定周期扫描该文件夹里的bundle，对比版本后，自动完成更新。

说明：load文件夹是Apache Felix的默认目录，默认的比对周期是2秒。可以在FELIX\_HOME\conf\config.properties中进行详细配置。如：

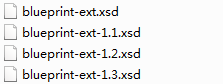
felix.fileinstall.dir=deploy

felix.fileinstall.poll=5000

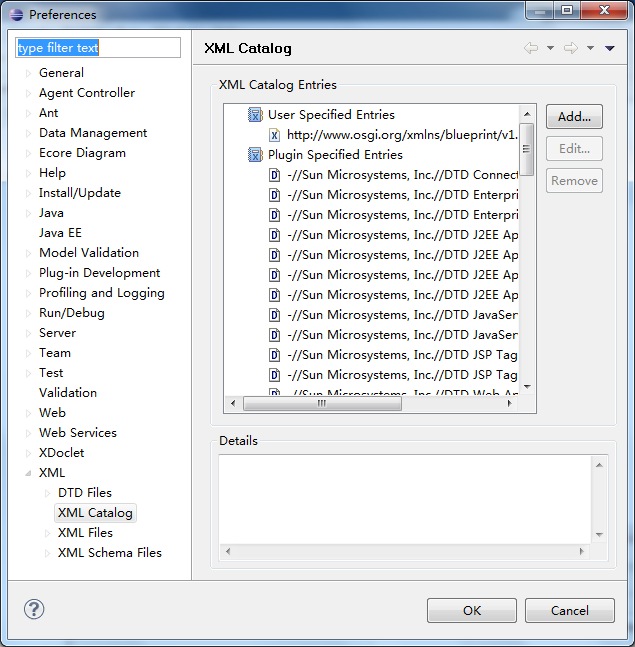
## 为blueprint应用在eclipse中提供xml内容提示

下载Apache aries源码，在如下目录下可以找到如下xsd文件。

pache-aries-src\_home\truck\blueprint\blueprint-core\src\main\resources\org\apache\aries\blueprint\ext\impl



点击Preferences->XML->XML Catalog。



# 结论

利用Eclipse PDE可以完成基本OSGi Bundle的开发，调试，导出，以及单元测试等内容。另外也有一些工具，支持更好的向导化开发OSGi，比如IBM提供的WAS工具，不过软件需要付费。

# 总结

Eclipse PDE本身提供了图形化编辑MANIFEST.MF文件编辑器，在调试，导出方面也提供了支持。利用Eclipse PDE是能够较好支持OSGi开发的。只要多少掌握一些OSGi方面的理论即可上手。另外，单元测试部分也是OSGi开发一个比较重要的主题，如何在OSGi容器中对bundle进行测试，比如对某bundle发布的服务情况进行测试等。单元测试部分内容需要在后期抽时间继续进行预研。