# 目录

[目录 1](#_Toc359930178)

[版本修订 4](#_Toc359930179)

[1 目的 5](#_Toc359930180)

[2 内容 5](#_Toc359930181)

[3 Blueprint 5](#_Toc359930182)

[3.1 轻量级容器 5](#_Toc359930183)

[3.2 何为BluePrint 5](#_Toc359930184)

[3.3 Blueprint主流实现 6](#_Toc359930185)

[3.4 低侵入的编程模型 6](#_Toc359930186)

[3.4.1 BluePrint Container 6](#_Toc359930187)

[3.5 Blueprint规范 7](#_Toc359930188)

[3.6 Blueprint XML 7](#_Toc359930189)

[3.6.1 Blueprint XML 文件片段 7](#_Toc359930190)

[3.6.2 Bean的装配注入 7](#_Toc359930191)

[3.6.3 发布服务 8](#_Toc359930192)

[3.6.4 引用服务 8](#_Toc359930193)

[3.7 集成JPA 9](#_Toc359930194)

[3.7.1 JDBC 9](#_Toc359930195)

[3.7.2 OpenJPA 9](#_Toc359930196)

[3.8 集成JNDI 9](#_Toc359930197)

[3.9 集成Transaction 9](#_Toc359930198)

[4 开发环境与工具 9](#_Toc359930199)

[4.1 开发环境简介 9](#_Toc359930200)

[4.2 准备一系列基础bundle 10](#_Toc359930201)

[4.3 建立Target Platform 10](#_Toc359930202)

[4.4 使用Eclipse开发blueprint bundle 14](#_Toc359930203)

[4.5 在Eclipse中运行调试Bundle 17](#_Toc359930204)

[4.6 单元测试 18](#_Toc359930205)

[4.7 导出可部署的Bundle 18](#_Toc359930206)

[4.8 OSGi Bundle的部署与安装 20](#_Toc359930207)

[4.9 Bundle的自动更新 20](#_Toc359930208)

[4.10 为blueprint应用在eclipse中提供xml内容提示 21](#_Toc359930209)

[5 为OSGi Web开发提供支持 22](#_Toc359930210)

[5.1 基于OSGi的企业级开发模式 22](#_Toc359930211)

[5.1.1 传统模式 22](#_Toc359930212)

[5.1.2 OSGi in App Server 22](#_Toc359930213)

[5.1.3 App Server On OSGi 23](#_Toc359930214)

[5.2 WAB的概念 23](#_Toc359930215)

[5.3 WAR的概念 23](#_Toc359930216)

[5.4 Jetty 24](#_Toc359930217)

[5.5 嵌入式Jetty Bundle 24](#_Toc359930218)

[5.6 使用HTTP Service 24](#_Toc359930219)

[5.6.1 Http Service主要bundle介绍： 24](#_Toc359930220)

[5.6.2 安装Http Service 24](#_Toc359930221)

[5.6.3 访问Servlet 24](#_Toc359930222)

[5.6.4 访问Html 24](#_Toc359930223)

[5.6.5 访问JSP 24](#_Toc359930224)

[5.6.6 Whiteboard模式 25](#_Toc359930225)

[5.6.7 Servlet Bridge 25](#_Toc359930226)

[5.7 Pax Web 25](#_Toc359930227)

[5.7.1 什么是Pax Web 25](#_Toc359930228)

[5.7.2 将WAR迁移至OSGi环境要做的一些事情 25](#_Toc359930229)

[5.7.3 Pax Web示例 25](#_Toc359930230)

[5.7.4 Pax Web Extender 26](#_Toc359930231)

[5.7.5 Using the Pax Web whiteboard extender 27](#_Toc359930232)

[5.8 将OSGi框架嵌至Web Server 27](#_Toc359930233)

[5.8.1 原理 27](#_Toc359930234)

[5.8.2 将OSGi框架嵌入Apache Tomcat 27](#_Toc359930235)

[6 Apche aries实践 27](#_Toc359930236)

[6.1 Apache aries blueprint最少需要哪些bundle支持 27](#_Toc359930237)

[6.2 使用日志 28](#_Toc359930238)

[7 附录 28](#_Toc359930239)

[7.1 Apache Felix简介 28](#_Toc359930240)

[7.2 Apache Felix基本运行环境配置 28](#_Toc359930241)

[7.2.1 Apache Felix执行环境的下载 28](#_Toc359930242)

[7.2.2 启动Apache Felix框架 28](#_Toc359930243)

[7.3 Apache Felix基本控制命令 29](#_Toc359930244)

[7.4 在Eclipse中建立Felix开发环境 29](#_Toc359930245)

[7.5 Eclipse环境下运行bundle 33](#_Toc359930246)

[7.6 Apache Felix基本配置 34](#_Toc359930247)

[7.7 Web Console 34](#_Toc359930248)

[7.8 Shell 35](#_Toc359930249)

[7.9 File Install 36](#_Toc359930250)

[7.10 Lightweight Http 36](#_Toc359930251)

# 版本修订

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 描述 | 修订人 |
| V1.0 | 2013-6-18 | 创建 | 陈绪绍 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 目的

基于WMS产品前期所定的有关OSGi的基本框架选型：Apache felix + Apache aries。就基于Eclipse平台进行OSGi Bundle的开发环境进行及其中所涉及的技术进行预研和实践。尽量减小后期开发风险。本预研文档，是一个迭代完善的过程。会今后的预研与实践当中，逐步添加与完善相关章节的内容。

# 内容

# 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。

## 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主流实现

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

## 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。

### 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 依赖性注入（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

## 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OSGiuService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 XML

### Blueprint XML 文件片段

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <blueprint xmlns=”http://www.osgi.org/xmlns/blueprint/v1.0.0”>  ...  </blueprint> |

### Bean的装配注入

下面是一个简单的 Blueprint Bundle 的配置文件

|  |
| --- |
| <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">  <bean id="hw" class="com.abc.HelloWorld">  <argument value=”Rex”/>  <argument value=”Wang”/>  <property name=”country” value=”China”/>  <property name=”city” value=”Shanghai”/>  </bean>  </blueprint> |

其中HelloWorld的类定义如下：

|  |
| --- |
| public class HelloWorld {  private String country;  private String city;  public HelloWorld (String FirstName, String FamilyName){  …  }  public setCountry(String country){  this.country = country;  }  public setCity(String city){  this.city = city;  }  } |

因此，上述配置文件的作用相当于

|  |
| --- |
| HelloWorldhw = new HelloWorld(“Rex”, “Wang”);  hw.setCountry(“China”);  hw.setCity(“Shanghai”); |

### 发布服务

|  |
| --- |
| <blueprint xmlns=*"http://www.osgi.org/xmlns/blueprint/v1.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:tx=*"http://aries.apache.org/xmlns/transactions/v1.0.0"*  default-activation=*"lazy"*>  <bean id=*"persistenceImpl"* class  =*"com.ibm.samples.websphere.osgi.blog.persistence.BlogPersistenceServiceImpl"*>  <tx:transaction method=*"\*"* value=*"Required"*/>  </bean>  <service ref=*"persistenceImpl"* interface  =*"com.ibm.samples.websphere.osgi.blog.persistence.api.BlogPersistenceService"*>  </service>  </blueprint> |

### 引用服务

在配置文件中，我们也可以对需要的服务进行引用，并且发布自己的服务，比如：

|  |
| --- |
| <reference id="helloservice" interface="com.abc.HelloWorld" /> |

表示引用一个接口为 “com.abc.HelloWorld”的 Service 对象，而：

|  |
| --- |
| <service ref="hw" interface="com.abc.HelloWorld" /> |

表示将对象“hw”发布为接口为“com.abc.HelloWorld”的服务。

你可能注意到，对服务的引用和发布与已有的 Declarative Service 十分相似。但是，Blueprint 提供了更为灵活的动态加载机制，当 Service 所在的 Bundle 是一个 Lazy Bundle 时，这个 Service 可以注册一个 Placeholder 来等待其它应用的发现和引用。此外，基于 POJO 的对象实例化、服务的引用和发布方法可以使你避免在你的应用程序中引入一些OSGi平台的 API 代码，如BundleContext.registerService、BundleContext.getServiceReference等，这不仅使你的应用可以独立于OSGi框架而运行，同时也可以方便的编写 Mock 对象从而进行传统单元测试。

## 集成JPA

需要在META-INF\MANIFEST.MF文件中添加Meta-Persistence字段，比如：

META-INF/persistence.xml

参考：http://aries.apache.org/modules/jpaproject.html

### JDBC

### OpenJPA

## 集成JNDI

## 集成Transaction

# 开发环境与工具

目前，使用Eclipse IDE开发Bundld方面的插件较少，一些能用的(Pax Runner)，其官方提供的update site已经失效。

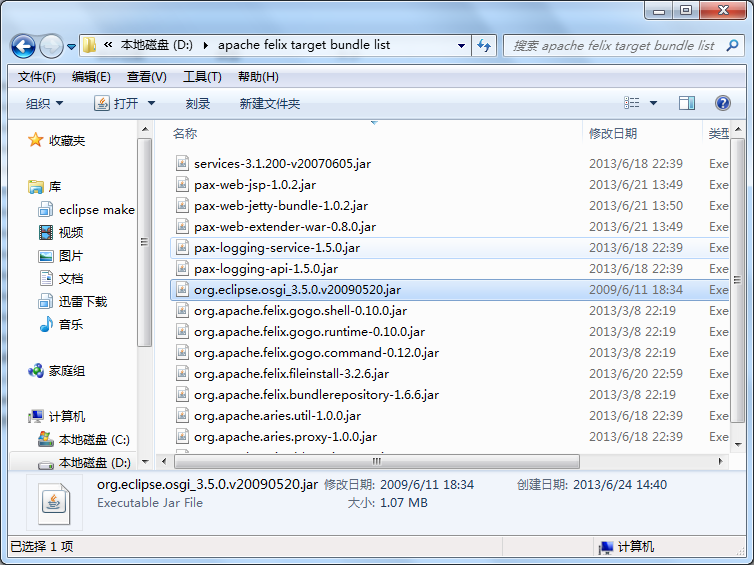
## 开发环境简介

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

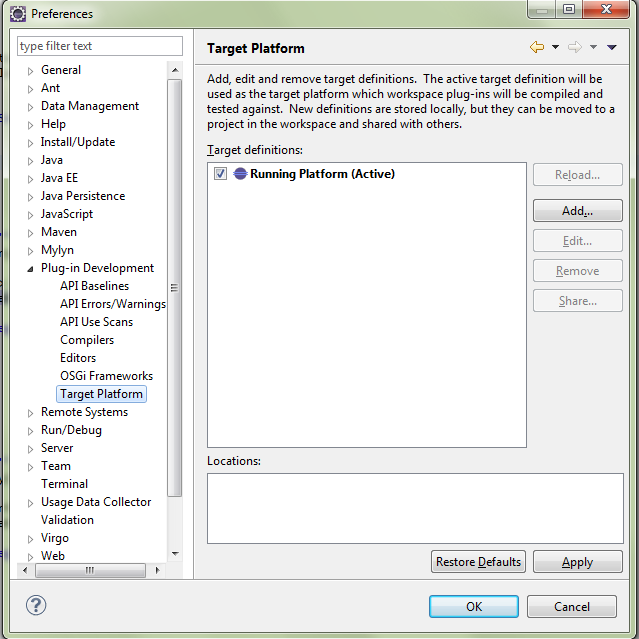
为建立Target Platform作准备。准备一个目录，将所有基础Bundle拷贝至该目录。

主要包括org.eclipse.osgi\*.jar(以Equinox作为内核)，最小的blueprint相关bundle，日志相关bundle， web支持的bundle等。bundle列表一开始不要求是大而全的，在后期可以逐步加入。比如：

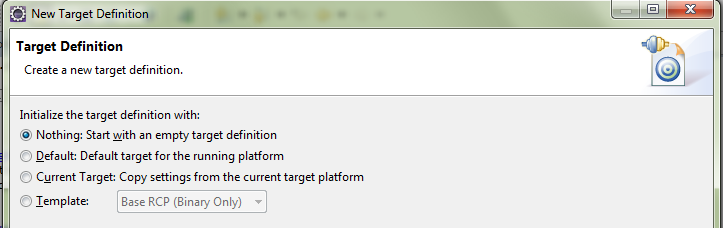


## 建立Target Platform

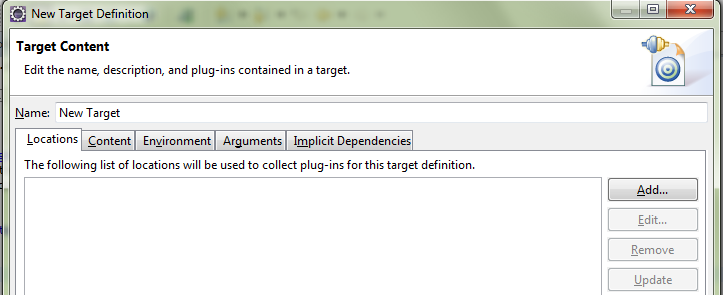
进入Eclipse的window->preferences界面，选中Plug-in Development下的Target Platform：



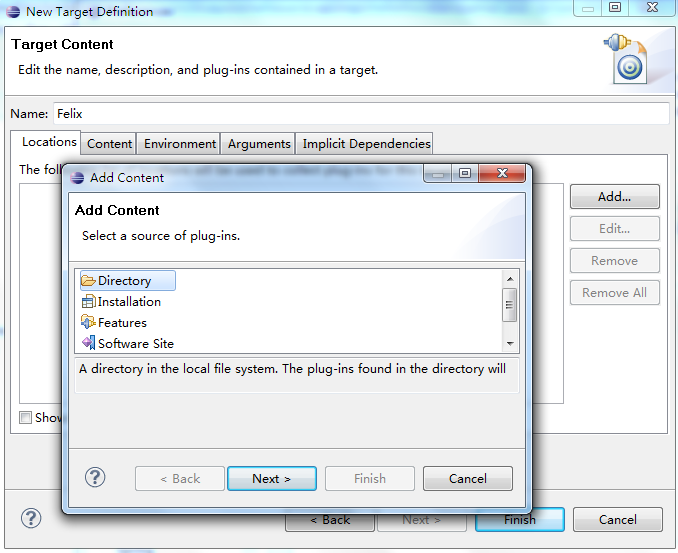
现在右边只有一个Runing Platform的，我们任务是点击“Add…”按钮来增加一个我们自己的的platform，进入如下界面：



选择默认的第一个就好，点击next。

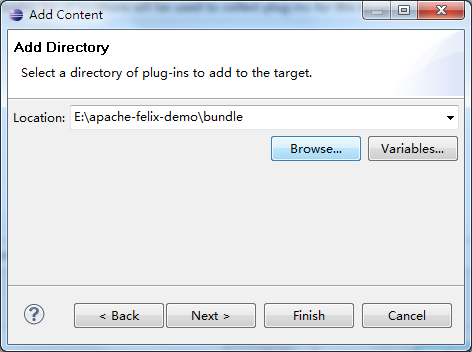


重命名为Felix(可以任意命名)。再点击这里的“Add…”：

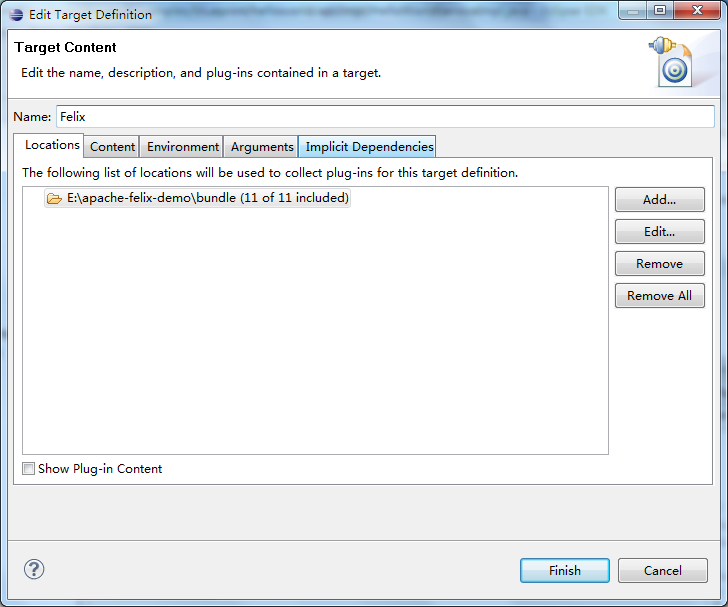


选择“Directory”：

选择准备的包含有你所准备好的bundle的目录。

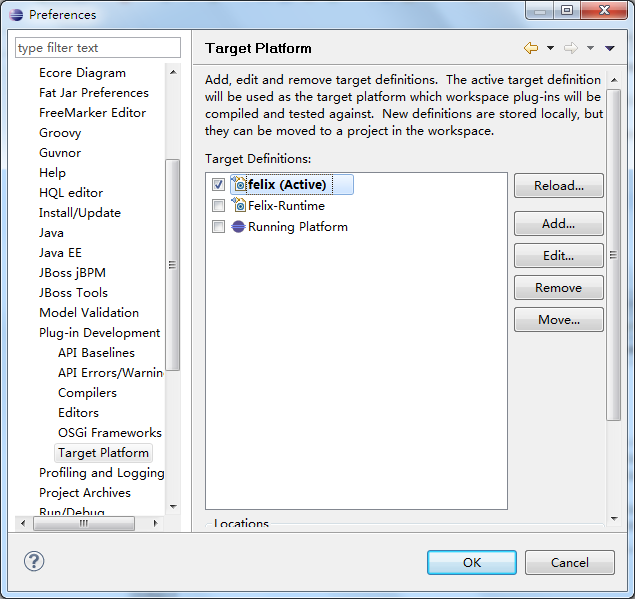


选择你的Equinox的jar包所在的路径,然后点击finish，回到刚才的界面：



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。

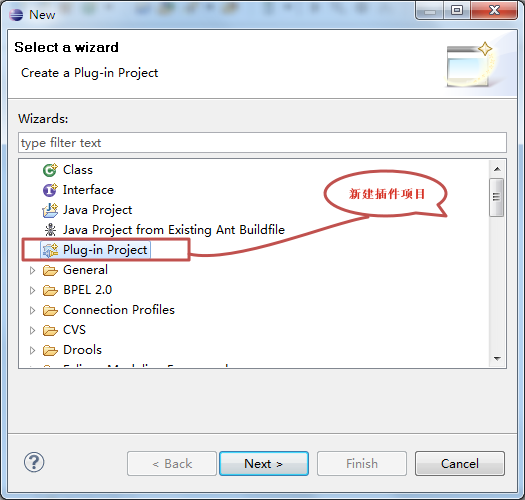
继续点击finish，回到最开始的界面：



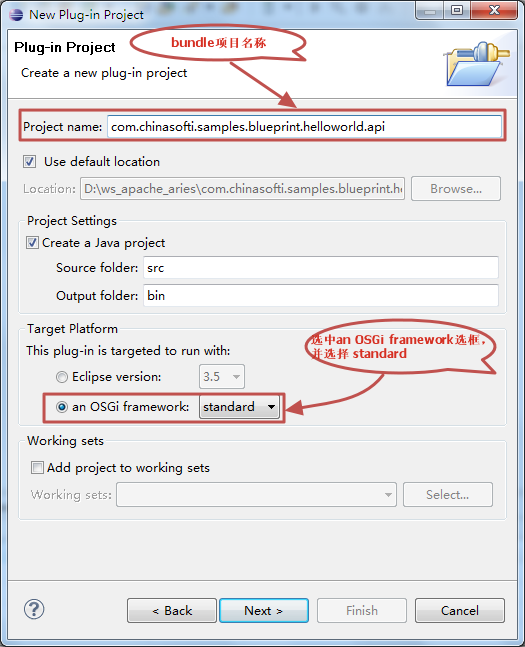
这时候多出来了一个新的target platform，即Feix，勾选上，然后确定。

## 使用Eclipse开发blueprint bundle

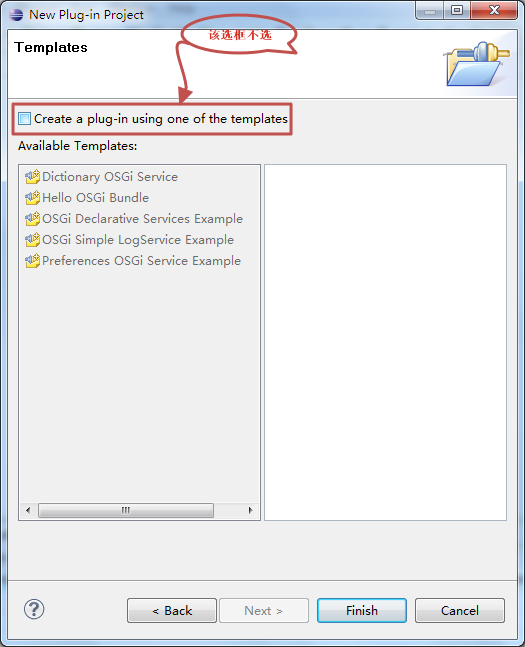
1. 新建插件项目



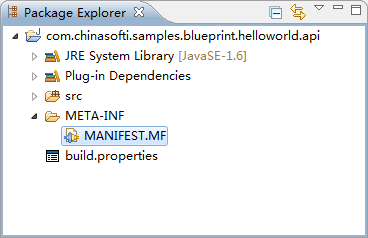
1. 选择an OSGi Framework



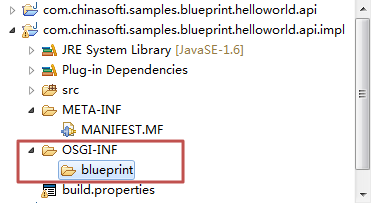
1. 建立纯净的OSGi Bundle工程模板



Bundle工程的基本目录结构

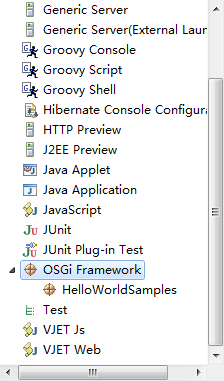


1. 工程根目录下建立OSGI-INF/blueprint目录

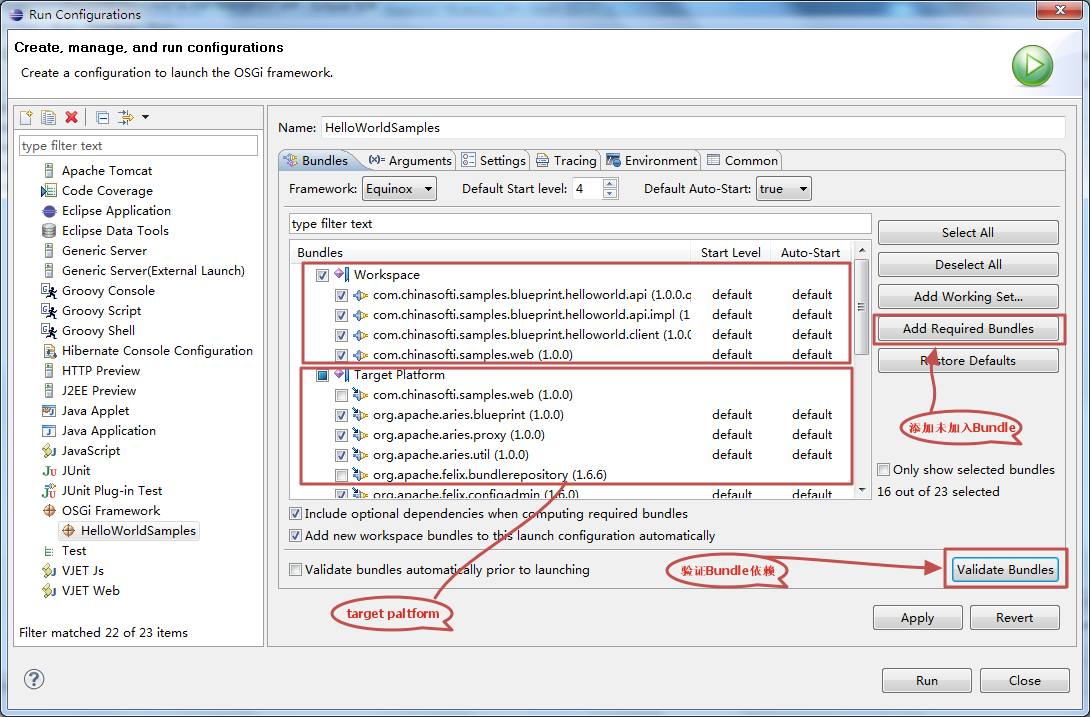


## 在Eclipse中运行调试Bundle

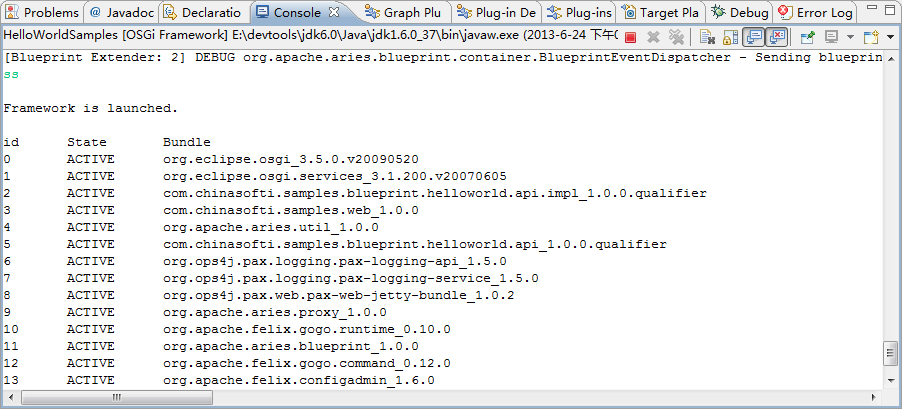
1. 运行：Run-->Run Configurations 调试：Debug--> Debug Configurations
2. 双击点击OSGi Framework，或右键-->”New”。新建一个Run configuration节点。



弹出如下界面：



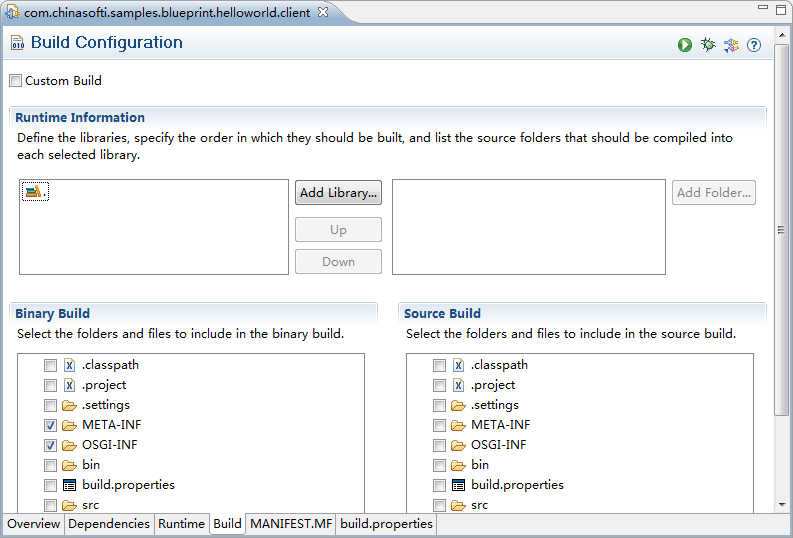
点击Run，即可在Eclipse平台运行Bundle。



## 单元测试

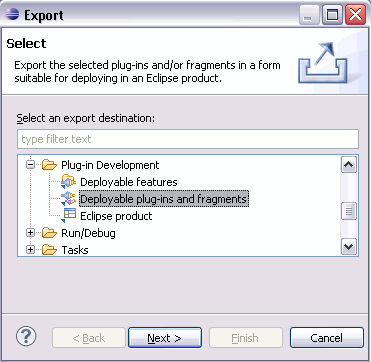
## 导出可部署的Bundle

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

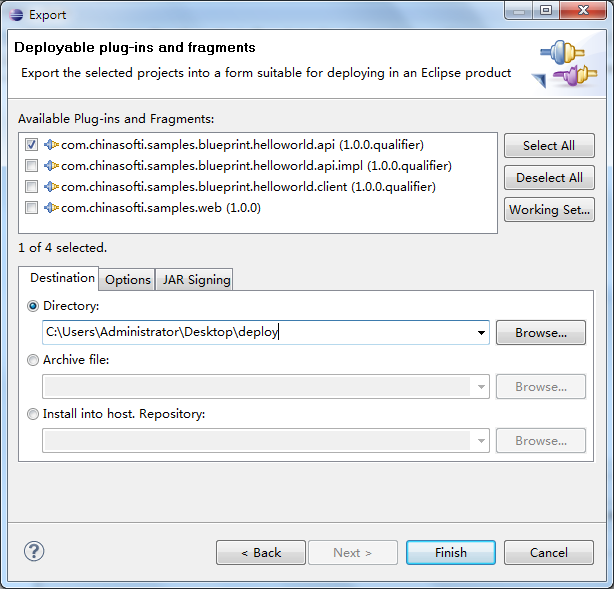


在build标签中选中将OSGI-INF导出，这一点很重要。

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



选中Deployable plug-ins and fragment



## OSGi Bundle的部署与安装

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。

## 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：

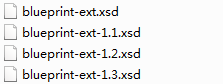
felix.fileinstall.dir=deploy

felix.fileinstall.poll=5000

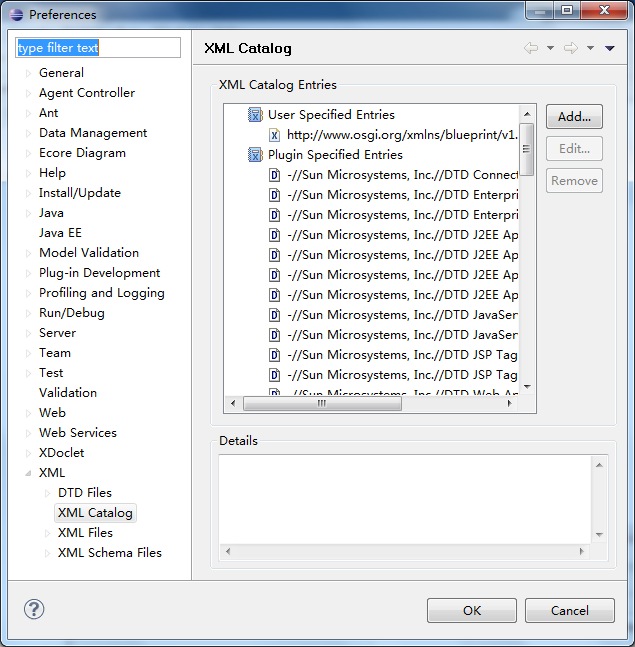
## 为blueprint应用在eclipse中提供xml内容提示

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。



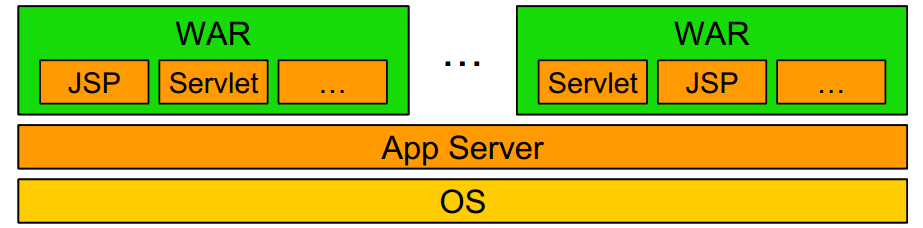
# 为OSGi Web开发提供支持

通过对OSGi平台上的Web应用的透明支持。可以让使用Servlet、JSP、taglib等技术的WAR无需更改或很小的更改就能在OSGi平台上使用。详细描述页面：

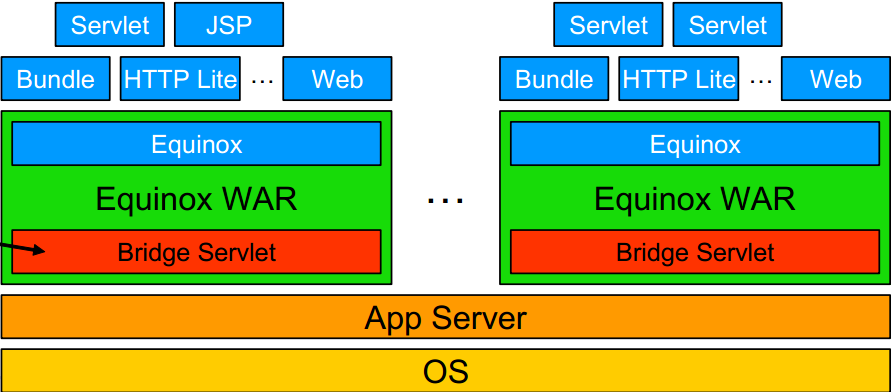
<http://felix.apache.org/documentation/subprojects/apache-felix-http-service.html>

## 基于OSGi的企业级开发模式

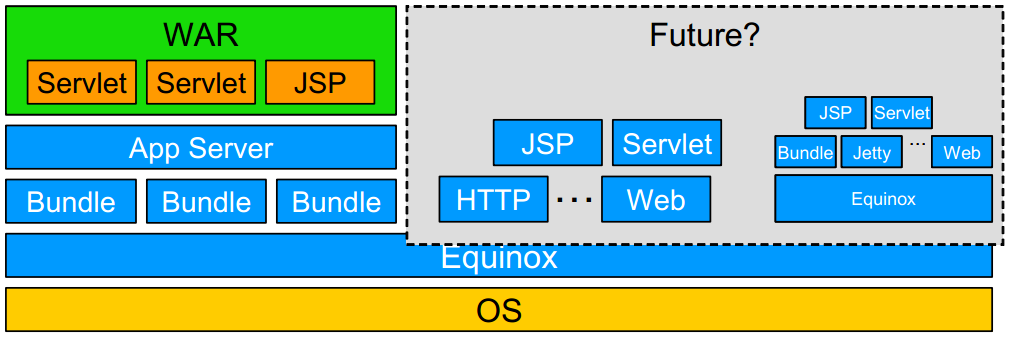
### 传统模式



### OSGi in App Server



### App Server On OSGi



## WAB的概念

OSGi R4.2制定了Web Applications规范，使OSGi中包含Web页面的模块可以使用标准WAR格式来打包（打包后的产品为以.wab为扩展名的JAR格式文件），允许将这些模块直接安装到支持OSGi和Web Applications规范的应用服务器之中。

Web Application Bundle(WAB)，它与WAR在Java EE中扮演的角色是相同的，可以认为是OSGi环境下web应用的打包、部署模型。WAB实际上也是一种标准的OSGi bundle，只不过它可以和OSGi环境及生命周期模型交互，而不是标准的Java EE环境。

## WAR的概念

WAR是Sun提出的一种Web应用程序格式，与JAR类似，也是许多文件的一个压缩包。这个包中的文件按一定目录结构来组织：通常其根目录下包含有HTML和Jsp文件或者包含这两种文件的目录，另外还会有一个WEB-INF目录，这个目录很重要。通常在WEB-INF目录下有一个web.xml文件和一个classes目录，web.xml是这个应用的配置文件，而classes目录下则包含编译好的Servlet类和Jsp或Servlet所依赖的其它类（如JavaBean）。通常这些所依赖的类也可以打包成JAR放到WEB-INF下的lib目录下，

简单的说，war是一个web模块，其中需要包括WEB-INF，是可以直接运行的WEB模块。

## Jetty

Jetty是一个嵌入式的Servlet服务器。你可以简单的把它类比为是Tomcat。

## 嵌入式Jetty Bundle

为了在OSGi环境中支持Web请求，一些主流OSGi框架，将Jetty包装为一个Bundle部署到OSGi环境中，即将web服务器嵌至OSGi框架中。目前这也是唯一实现了的Servlet容器。提供Jetty Bundle包装的框架和项目包括：Equinox、Felix、Pax Web等。

## 使用HTTP Service

### Http Service主要bundle介绍：

* org.apache.felix.http.jetty - HTTP Service implementation that is embedding Jetty server.
* org.apache.felix.http.whiteboard - Whiteboard implementation that uses any HTTP Service implementation.
* org.apache.felix.http.bridge - HTTP Service implementation that uses the host applicaiton server (bridged mode). Must be used with proxy.
* org.apache.felix.http.bundle - All in one bundle that includes all of the above.
* org.apache.felix.http.proxy - Proxy that is needed inside WAR when deployed inside an application server.

这里说只要org.apache.felix.http.bundle就可以了，经试验后发现只安装这个仍然不行，还是要安装org.apache.felix.http.jetty。

### 安装Http Service

最基本的WEB支持，只需要需要如下两个bundle：

* org.apache.felix.http.jetty -提供了嵌入式的jetty服务器
* org.apache.felix.http.bundle –包括了jetty，whiteboard，servlet等api

### 访问Servlet

### 访问Html

### 访问JSP

### Whiteboard模式

简化web访问

http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf

### Servlet Bridge

Servlet Bridge提供一个底层的Hook servlet，使得OSGi框架可以嵌入到现有的应用服务器中运行。该Bundle中的servlet启动嵌入的OSgi框架并使其可以处理来自底层应用服务器接收的HTTP请求。

## Pax Web

### 什么是Pax Web

Pax web项目主页：https://ops4j1.jira.com/wiki/display/paxweb/Pax+Web

### 将WAR迁移至OSGi环境要做的一些事情

1. 修改MANIFEST.MF文件内容，比如：

|  |
| --- |
| Manifest-Version: 1.0  Bundle-ManifestVersion: 2  Bundle-Name: Hello Web  Bundle-SymbolicName: com.chinasofti.samples.web  Bundle-Version: 1.0  Bundle-ClassPath: .,WEB-INF/classes  Bundle-Vendor: chenxushao  Bundle-ContactAddress: http://www.knopflerfish.org  Bundle-Description: web demo  Webapp-Context: helloweb  Import-Package: javax.servlet;version="2.5.0",  javax.servlet.http;version="2.5.0" |

1. 将WAR包文件后缀名修改为.jar
2. 将修改好后缀名的jar文件部署至OSGi环境

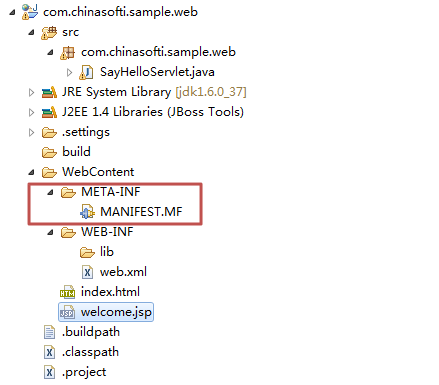
### Pax Web示例

1. 将pax web相关的bundle拷贝至Target Platform目录

Pax web需要使用如下几个bundle：

* pax-web-jsp-1.0.2.jar (install)
* pax-web-jetty-bundle-1.0.2.jar (install & start)
* pax-web-extender-war-0.8.0.jar(install & start)

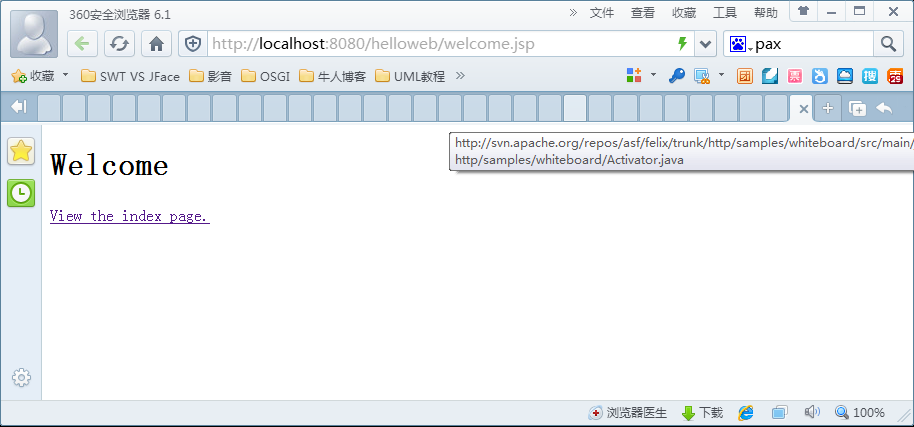
1. 准备WAR包，比如：



1. 部署WAR包至OSGi环境

以Apache Felix为例，扔进FELIX\_HOME/bundle目录即可

1. 浏览器访问效果



### Pax Web Extender

### Using the Pax Web whiteboard extender

## 将OSGi框架嵌至Web Server

### 原理

### 将OSGi框架嵌入Apache Tomcat

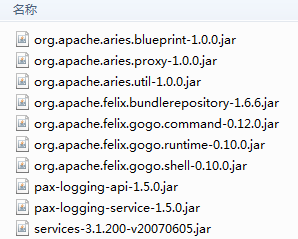
# Apche aries实践

## Apache aries blueprint最少需要哪些bundle支持

基于apache felix环境，需要jdk6.0，否则报错

* 最基本的felix bundle，即通过官方下载的distribution包。Gogo shell之类的不是必需的，完全可以使用其它的shell，比如Equinox的。
* org.apache.aries.blueprint-1.0.0.jar
* org.apache.aries.proxy-1.0.0.jar [依赖slf日志组件]
* org.apache.aries.util-1.0.0.jar
* pax-logging-service-1.5.0.jar [来自pax web项目，对slf日志的包装]
* pax-logging-api-1.5.0.jar [来自pax web项目，对slf日志的包装]
* services-3.1.200-v20070605.jar

如下图所示：



## 使用日志

# 附录

## Apache Felix简介

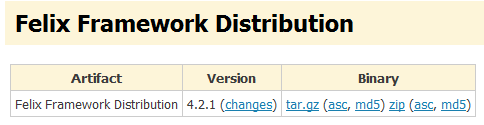
Apace Felix是Apache Foundation关于OSGi R4的一个实现。包括了OSGi框架和标准的服务，同时也提供并且支持其他的OSGi相关技术。Felix最终的目标是要提供和OSGi框架和标准服务完全兼容的一个实现。目前Felix已经实现了OSGi R4规范中的大部分内容，具体可以参考http://felix.apache.org/site/index.html

## Apache Felix基本运行环境配置

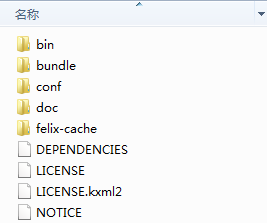
### Apache Felix执行环境的下载

进入下载页面，选择Felix Framework Distribution的下载项：

<http://felix.apache.org/site/downloads.cgi>。如下图所示：



Windows用户下载zip格式，Linux用户下载tar.gz格式。下载完成后，解压，得到如下图所示的目录结构。

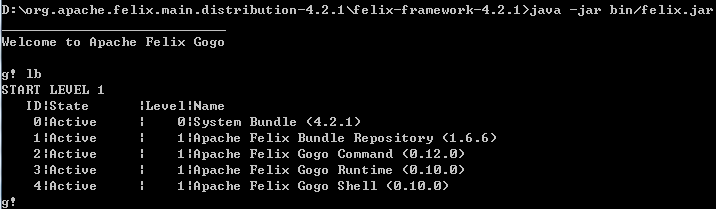


### 启动Apache Felix框架

进入felix根目录，执行如下命令，进入到Felix的命令行：

java -jar bin/felix.jar

如下图所示：



这样我们就进入了Felix的控制台，可以通过Shell与框架进行一些交互。

## Apache Felix基本控制命令

lb

bundles

install install file:bundle/listener.jar or start file:/c:/tutorial/example1.jar

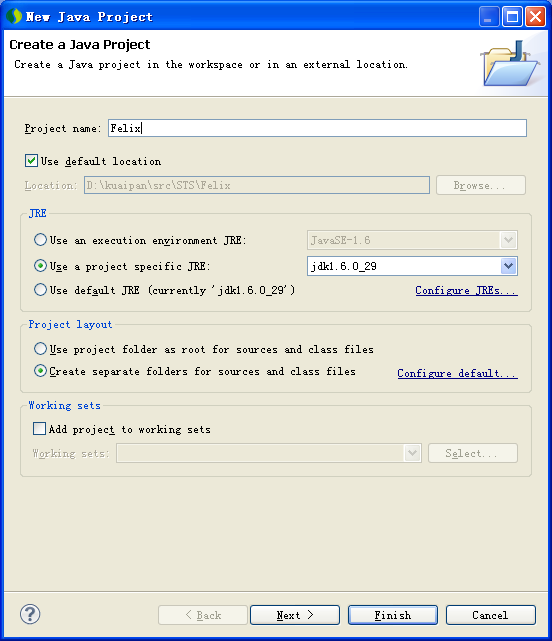
start start 7

## 在Eclipse中建立Felix开发环境

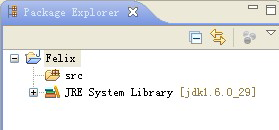
先下载Apache Felix并解压至某目录。参考7.2.1。

接下来我们通过Eclipse来配置Apache Felix

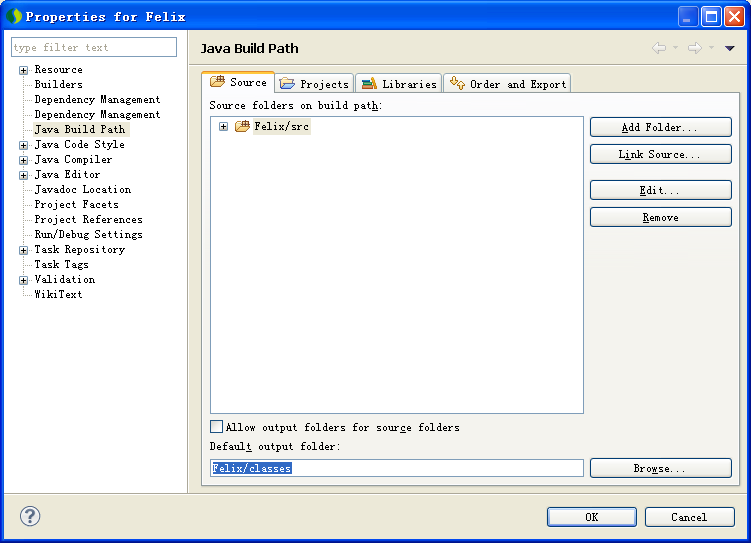
1. 创建一个新的Java工程：



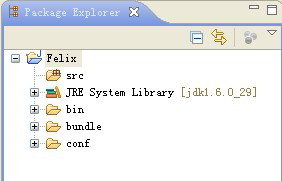
完成后的工程如下图所示：



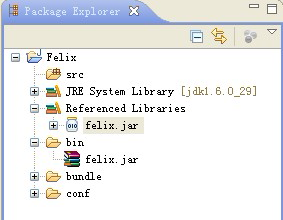
1. 修改默认Output文件夹，选中Felix工程，Alt+Enter快捷键，将输出文件夹改为Felix/classes，单击OK按钮。



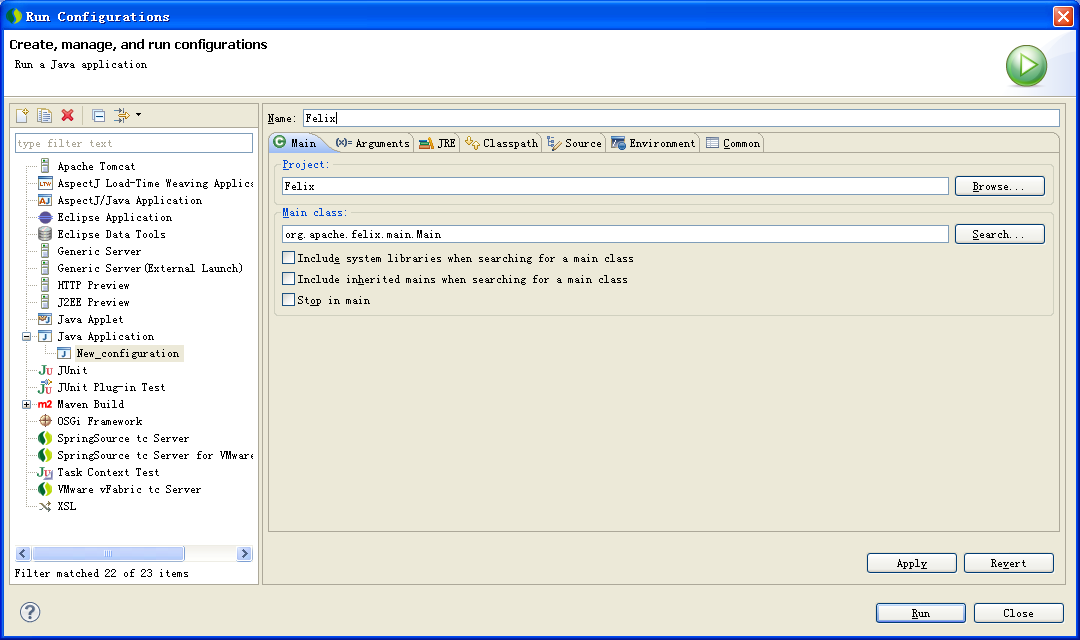
1. 复制 Felix Framework Distribution 文件。将刚才解压的Felix Framework Distribution的bin、bundle、conf文件复制到新创建的Felix的工程目录中，完成后的工程显示如下图所示。



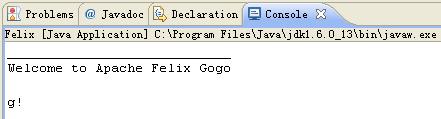
1. 将bin文件中felix.jar加入到Build path中，右击Felix.jar-->Build Path-->Add to Build Path，完成后的工程显示如下图所示。



1. 配置Run Configurations



单击Run按钮，就可以启动Felix了，运行的结果如下图所示。



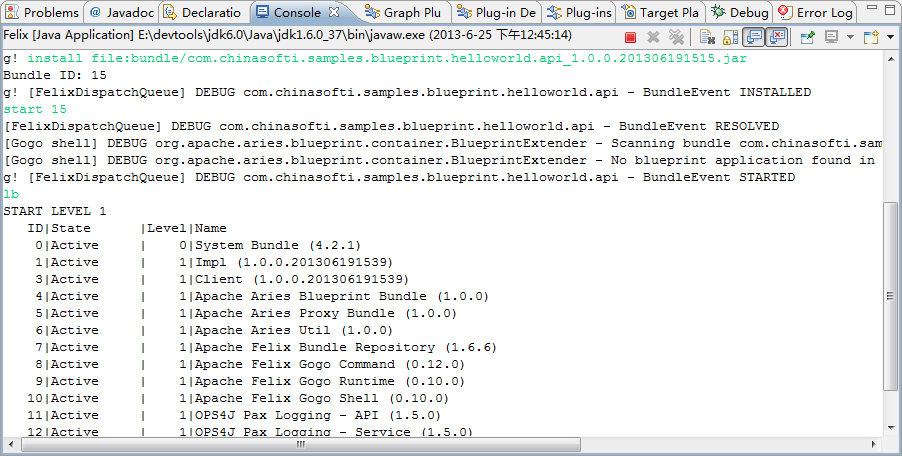
## Eclipse环境下运行bundle

1. 新建一个插件工程并导出为可部署的bundle。参考4章节内容。
2. 接着就是安装、运行了。

有三种方法可以安装、运行一个bundle。

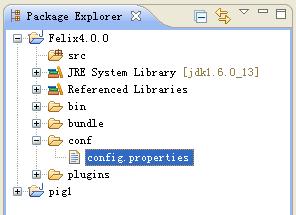
1. 使用命令。

首先，启动Felix，在Console中先使用install命令安装bundle，接着使用start命令启动bundle，如下图：



启动的时候，start命令后接着那个bundle的启动ID就可以启动bundle了，如上图的15。当然，你也可以使用uninstall命令卸载一个bundle，用法如install命令。

1. 使用Felix配置文件，打开conf/config.properties，如下图：



打开config.properties，找到felix.auto.start.1参数，值写成

felix.auto.start.1=bundle/com.chinasofti.samples.blueprint.helloworld.api\_1.0.0.201306191515.jar

如果你有多个bundle，之间用空格隔开。

参数写好后，启动Felix，你就可以看到bundle自动安装并启动了

1. 第三种方法就是使用File Install了，使用Apache Felix的File Install bundle，我们可以安装和启动bundle而无需启动Felix，参见7.9章节。

## Apache Felix基本配置

配置文件位于FELIX\_HOME\conf\config.properties

* org.osgi.framework.storage.clean - 决定bundle缓存是否刷新。值是“none”或“onFirstInit”。“none”不刷新缓存；“onFirstInit”当框架首次初始化时刷新缓存。默认值是“none”。
* org.osgi.service.http.port=8181 –配置http服务端口
* felix.fileinstall.dir=deploy 安装fileinstall服务后对监控目录进行配置

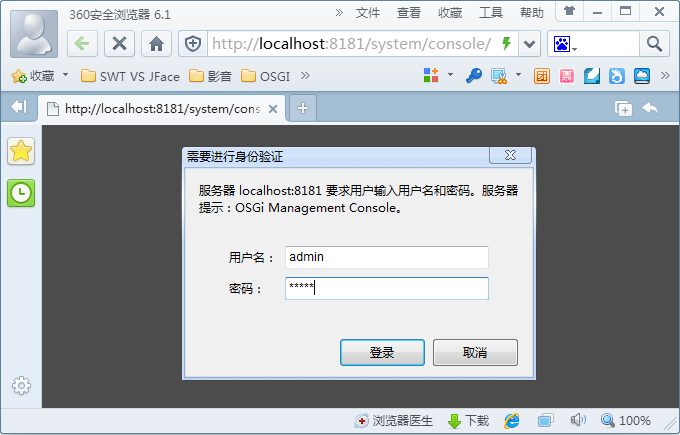
## Web Console

Web Console可以对OSGi环境进行如下一系列操作，包括bundle的安装，启动，停止，更新，日志查看等。

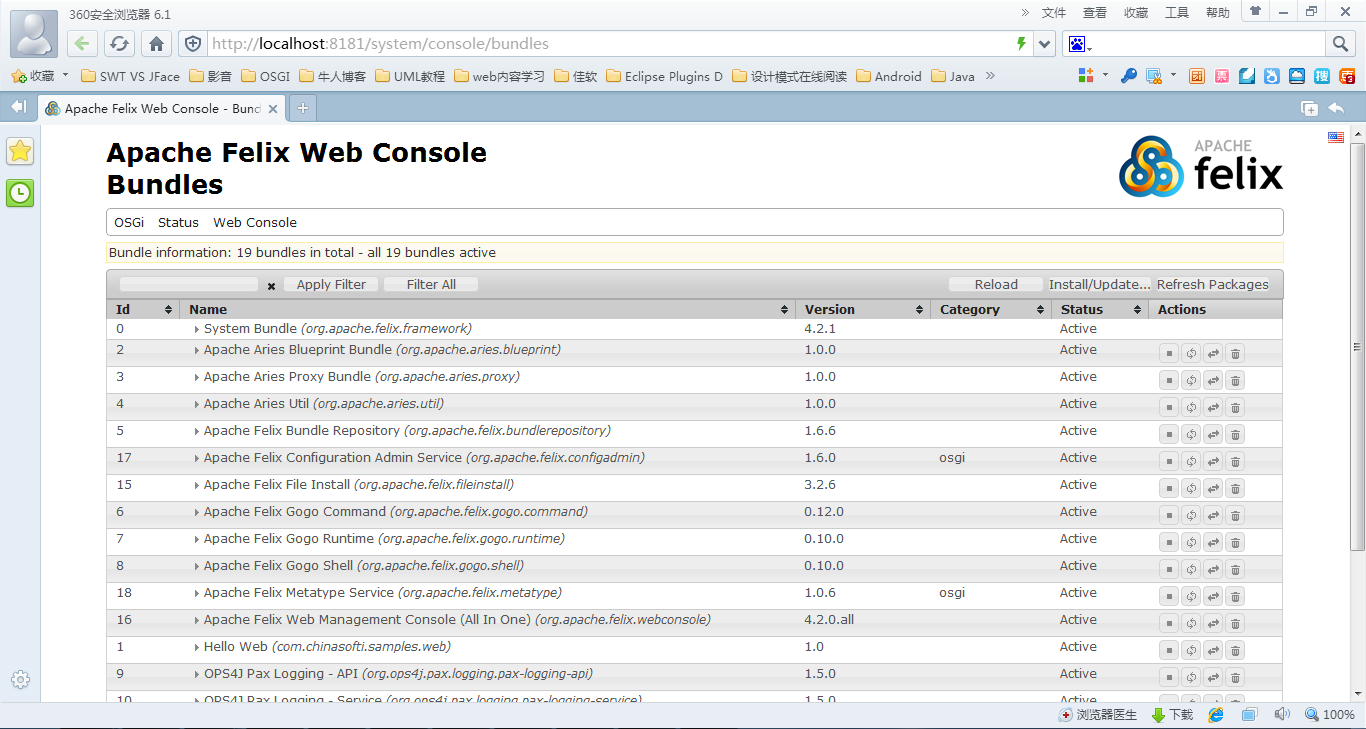
要使用Web Console，需要安装如下所列bundle：

* org.apache.felix.http.jetty-2.2.0.jar
* org.apache.felix.webconsole-4.2.0-all.jar
* org.apache.felix.metatype-1.0.6.jar [可选]
* org.apache.felix.configadmin-1.6.0.jar [可选]

启动felix框架。在浏览器地址栏中键入<http://localhost:8080/system/console>。就可以访问felix的控制台了。



默认登录用户和密码为amdin/admin。登录后界面如下：



更多介绍与使用在如下页面有详述：

<http://felix.apache.org/site/apache-felix-web-console.html>

## Shell

包括如下几个项目，使用起来大同小异。

* [Gogo](http://felix.apache.org/site/apache-felix-gogo.html) - 用来和OSGi交互的较为常用的shell，除了基本的框架命令，其中还包含grep，cat，echo这类指令，并且可以自定义宏命令。
* [Shell](http://felix.apache.org/site/apache-felix-shell.html) - 这个子项目只提供了单纯的shell服务API，而没有提供用户界面。
* [Shell TUI](http://felix.apache.org/site/apache-felix-shell-tui.html) - Apache Felix Shell的文字界面，相比gogo来说更为简单，但是不怎么常用，他的使用依赖于shell bundle。
* [Remote Shell](http://felix.apache.org/site/apache-felix-remote-shell.html) - Apache Felix Shell的远程文字界面，我们通过这个工具可以在其他终端上远程操作Felix。

## File Install

File Install Bundle可以为OSGi框架提供如下功能：可在osgi框架运行过程中，监控某目录bundle的变化，并将其自动安装。从而实现系统真正意义上的自动运行。

其对应的bundle为org.apache.felix.fileinstall-3.2.6.jar，直接将其扔至FELIX\_HOME\bunlde目录下，启动felix框架即可。

可以在FELIX\_HOME\conf\config.properties中对监控目录进行配置。比如

felix.fileinstall.dir=deploy

有关File Install Bundle更多特性配置在如下页面有详述：

<http://felix.apache.org/site/apache-felix-file-install.html>

## Lightweight Http

http://felix.apache.org/documentation/subprojects/apache-felix-lightweight-http-service.html#apache-felix-lightweight-http-service