

体会OSGi的热插拔性

制造流通线 物流物联网事业部  
中软国际信息技术有限公司

2013-4-26

# 走近OSGi

## 什么是OSGi？

OSGi的名称来源于其开源组织的名称Open Services Gateway initiative，OSGI是一个标准，它致力于提供给Java项目一个模块化的底层环境，以及一系列通用的服务(Service）。和普通的JVM 程序相比，OSGi的程序天生拥有动态模块的特点，不同的模块(OSGI里称之为Bundle)有着独立的生命周期，可以独立进行安装、启动、停止、卸载的操作，模块间的依赖性管理也由OSGi提供。OSGi非常适合需要进行Plugin管理的项目，一个典型的成功案例就是Eclipse和它众多的Plugin。OSGi标准还规范了一系列我们常见的操作，日志、配置文件、事件队列、Web开发、JPA&JDBC等等，大部分部署 OSGi标准的框架都提供了这些服务，这样一方面规范了我们代码的结构，一方面节约了我们开发的时间。

**个人总结**

OSGI是一套Java模块化开发的规范。OSGI是目前Java动态模块系统事实上的工业标准。

OSGI是一个容器，一个bundle生存的容器。Bundle本身是不能启动的。它们需要在一个环境中启动和操作。这个环境就是OSGI。

这种解释比较抽象不太好理解，我总结了以下相对较粗浅的概念。

OSGI可以控制访问范围，这方面Java本身控制得不够细，比如我某个类只希望供某些指定的类使用，Java是无法做到的（Proptected只能供同包和子类用）。

拿J2EE来说，一个web应用通常只由一个project来完成，所有得实现均在此project中完成，web应用发布后，若以后新增功能，那么只能在此工程下添加包、类等等，然后再编译打包发布。

而OSGI应用是由多个project组成，此project被OSGI命名为Bundle（包），每个bundle有具体的服务或者功能，如果想增加一个全新的功能，那么你只需再开发一个bundle并部署上就可以了（OSGI支持热插拔），Eclipse的插件化概念就是基于OSGI的理念。

OSGI理念，我认为是未来的一个方向，把软件开发转向模块块开发（构件化）的方向，同时这也是一种软件设计理念，OSGI告诉我们，写软件的时候要把职责划分清楚，这是合理划分接口、类、包甚至是服务，组件的基础。我们在软件设计划分时从大块上划分为Platform，以及各产品插件。比如，将某应用划分出最基础的运行部分，称之为Platform，主要包括运行的核心包，例如core包，model包，ui包等等。各产品插件包依赖于platform，各个bundle各司其职，职责清楚，如果想修改某个具体的业务，那么只需到对应的bundle去修改。如果在底层新增个通用的功能，例如调用消息格式，那么只需在platform里修改，其他产品插件就获得了这个调用功能。若要新增了某插件，就需要再创建一个bundle。

## OSGI发展轨迹

1999年3月，OSGi联盟(非盈利性质)成立，目标是建立家庭网关，并通过互联网向家庭网络提供各种服务，例如通过web页面控制咖啡机等。

2000年至2003年，是OSGI规范发展的初级阶段，一共发布了三个版本。这个阶段其主要关注点是在移动和嵌入式设备上的Java模块化应用。这个阶段有一些很成功的案例，比如BMW汽车控制系统。初级阶段OSGI在Java其他主流应用领域(企业级、互联网、服务器、桌面端等)的影响力还比较有限。其版本更新记录如下：

2000年5月发布OSGI Release 1(R1)

2001年10月发布OSGI Release2(R2)

2003年3月发布OSGI Release 3(R3)

2012年5月份之前，OSGI一直在R4上发力，连续发布了好几个小版本。其目标也从“在移动和嵌入式设备上的Java模块化应用”发展为“Java模块化”，去掉了“在移动和嵌入式设备上的”限定语，这意味着OSGI开始脱离Java ME 的约束，向Java其他领域进军。在此期间，Eclipse V3.0采用OSGi，使得OSGi成功进入Java企业应用领域，并成为其中重要一员；同时Eclipse推出了OSGi R4 RI：Equinox。其版本更新记录如下：

2005年10月发布V 4.0

2007年5月发布V 4.1

2010年3月发布V4.2

2011年4月发布V4.3

2012年6月发布OSGI R5(同时发布了核心规范和企业级规范的OSGI R 5版本），这是目前最新的OSGI规范版本。OSGI R5 的一个主要目标是建立一套基于OSGI的模块仓库系统(为下一步的OSGI In Clund做准备)。

更详细的发展轨迹可以在OSGi联盟如下页面查看到：

[http://www.OSGi.org/Specifications/HomePage](http://www.osgi.org/Specifications/HomePage)

## OSGI的应用场景

OSGi服务平台的应用包括：

* + 嵌入式领域
    - 服务网关
    - 汽车
    - 工业自动化
    - 建筑物自动化
    - PDA网格计算
  + Mobile（移动终端）
  + IDE
  + WEB APP
  + OSGi提出了Universal Middleware，目标是让OSGi脱离语言限制，成为所有语言的统一模型。

## 基于OSGI的商业产品案例

OSGI在嵌入式领域和桌面领域取得了较大的成功。

◆BMW汽车的应用控制系统

BMW汽车的应用控制系统采用OSGI作为其底层架构，系统主要用来控制汽车上的音箱、灯光等等设备，总共由**1000**多个Bundle构成，但BMW汽车的应用控制系统启动时间却只需要3.5秒。

◆Eclipse

Eclipse作为Java业界成功的IDE ，在3.0以前的版本它采用的是自己设计的一套插件体系结构，在3.0版本时直接采用OSGI作为其插件体系结构。 Eclipse 3.1版本以后大家可以明显的感觉到启动速度的提升。 Eclipse之所以要抛弃自己那套已经比较成熟的插件体系结构而转而采用OSGI，就是因为OSGI的规范性以及OSGI对于插件体系结构更为完整的定义。

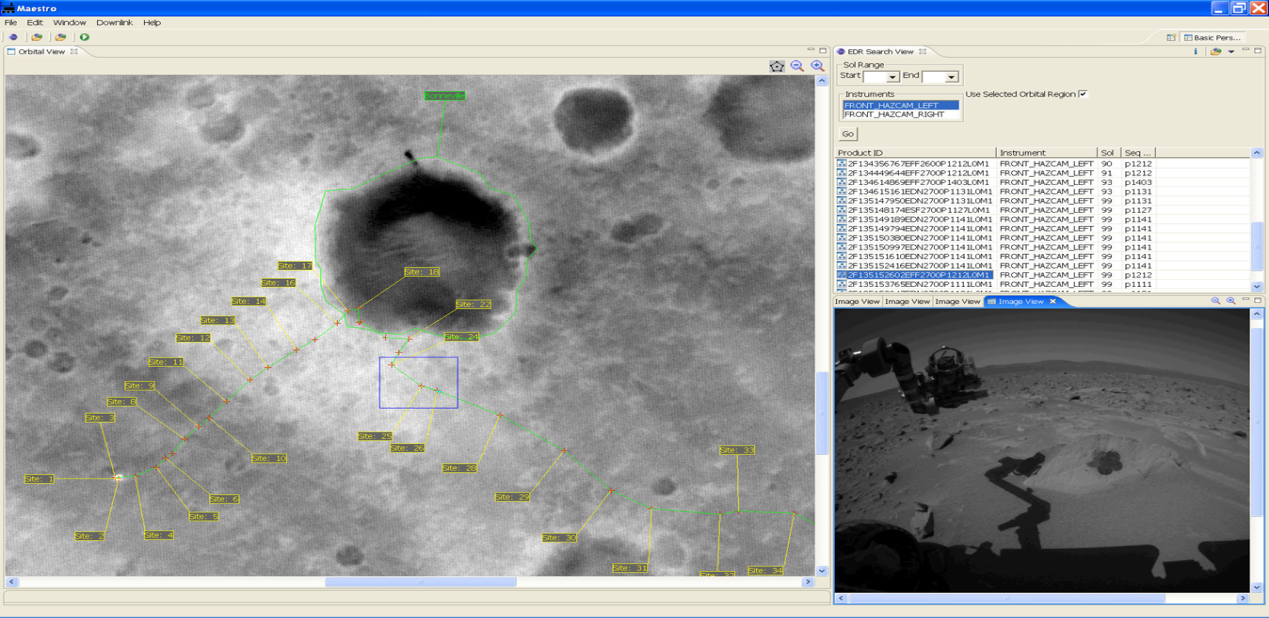
◆IBM WAS V6.1、Websphere

◆WebLogic

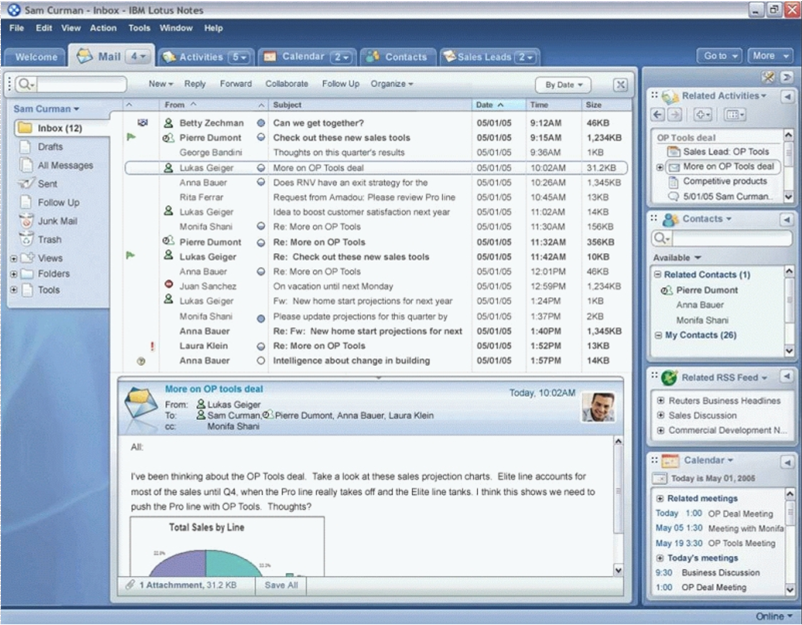
◆JBoss AS 5.0

◆Oracle Glassfish

◆基于Eclipse RCP的NASA火星探测系统



◆IBM Lotus系列产品



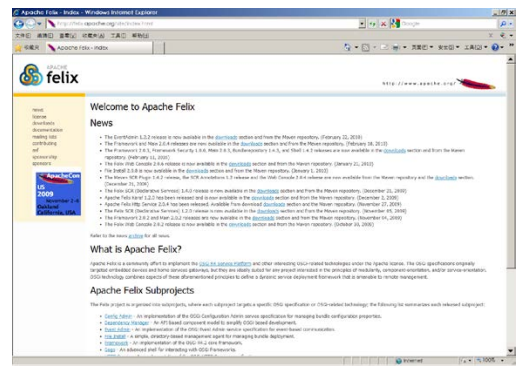
## 四大Java OSGI框架

OSGI的发展已经得到了众多企业、厂商、开源组织的支持，发展出了Apache Felix，Equinox，Knopflerfish，Spring DM 等多个开源 OSGI框架，主流的 Java 应用服务器也都纷纷采用 OSGI，如GlashFish，JBoss，Weblogic，其作为 Java 模块化标准已成为事实。

**Apache Felix 最全面的框架**

[Apache Felix](http://felix.apache.org/)是 Apache旗下的一个OSGi框架，项目本身非常成熟，已经被用到了很多其他的项目中。

官方网站：<http://felix.apache.org/>



**Equinox 与Eclipse完美结合**

[Equinox](http://www.eclipse.org/equinox/)是 Eclipse旗下的OSGi框架，本身也被Eclipse采用，是Eclipse注明的PDE开发环境的底层。Equinox本身也是相当的全面的框架。后续部分的特性演示也将基于 Equinox 框架来进行。

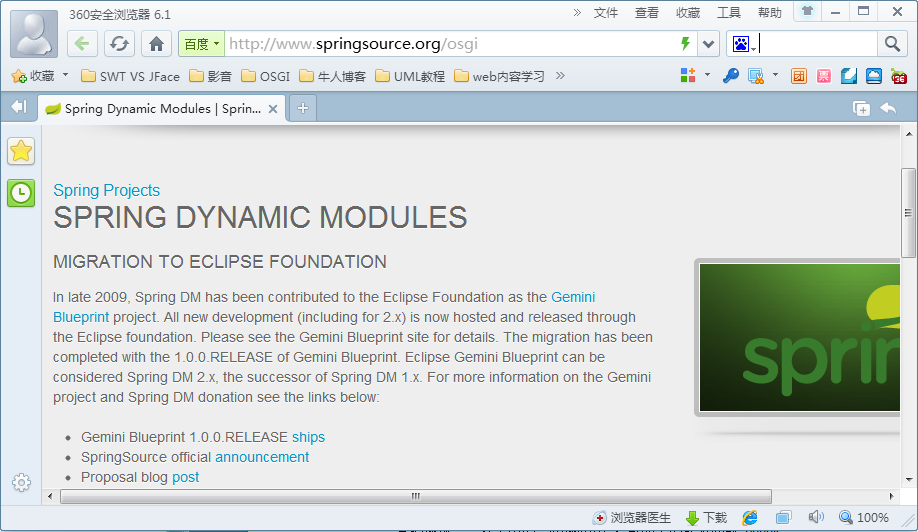
官方网站：<http://eclipse.org/equinox/>



**Spring DM 畸形的需求产物**

[Spring DM](http://www.springsource.org/osgi)是Spring旗下的OSGi框架。Spring和核心就是一个IoC，当然后来它的外延扩大了，提供了越来越多乱七八糟的功能。OSGI规范本身就制定了一系列IoC的功能标准，尤其是其中的BluePrint其实相当多的借鉴了 Spring，因此完全没有必要再引入Spring充当新的IoC了。Spring本身无论是ClassLoader还是配置文件上都与OSGi格格不入，之所以有这种需求是因为现在有大量基于Spring的项目想要过渡到OSGI上。

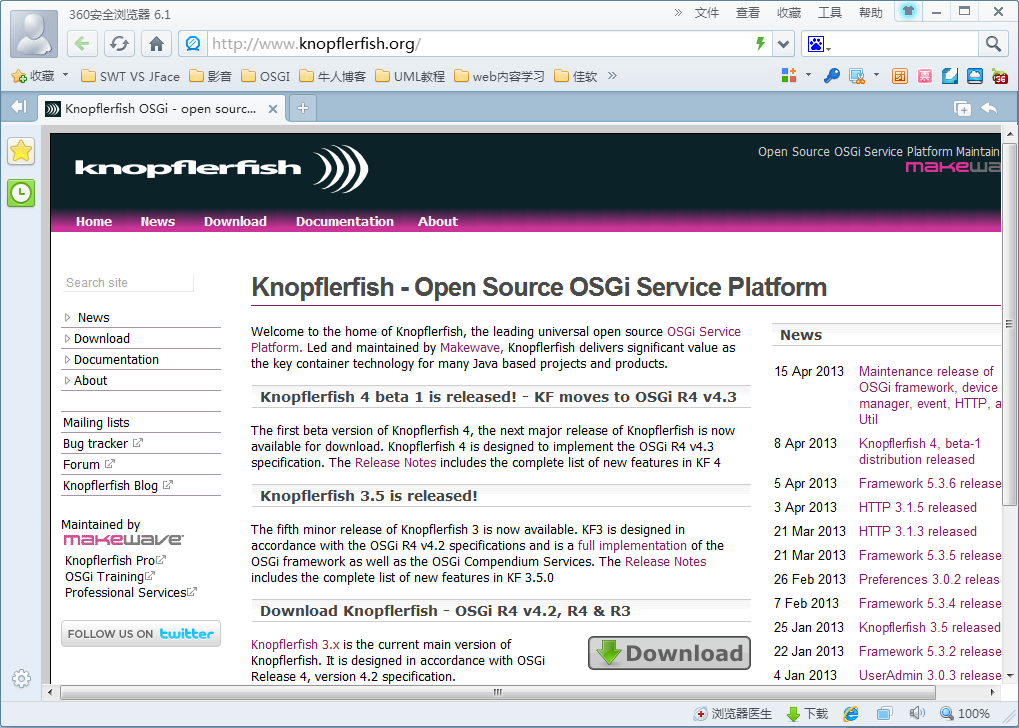
官方网站：[http://www.springsource.org/OSGi](http://www.springsource.org/osgi)



**Knopflerfish 孤独孤傲**

[Knopflerfish](http://www.knopflerfish.org/)其实是OSGi的先行者，但是由于没有强力的靠山，再后来的竞争中显然不如前三者有人气。它本身是一个相当标准OSGi框架，提供了绝大多数标准功能，但是无论在人气上，开发进度上，文档完善上都不如其他的三者。

官方网站：<http://www.knopflerfish.org/>



当前最新的OSGI规范已经更新至R5，但目前这四大框架都只支持至R4。

## OSGI带来了哪些可能

OSGi至少为我们在构建系统时带来如下可能：

◆OSGI为Java 程序模块化和重用提供了可能。

是当前Java领域唯一的模块化解决方案。

◆热插拔――梦想中的“即插即用”，可动态的安装卸载 bundle

OSGi 使得热插拔技术在软件界成为现实。 基于 OSGi 的系统，可通过安装新的 Bundle、更新或停止现有的 Bundle 来实现系统功能的插拔。

◆实现可扩展的OSGi程序

OSGi可以在系统运行时，动态改变系统的功能。，OSGI设计初衷是使系统组件化、模块化，进而实现动态化。

◆为构建更稳定、高效的系统提供了可能

基于OSGi 的系统采用的是微核机制，微核机制保证了系统的稳定性，微核机制的系统只要微核是稳定运行的，那么系统就不会崩溃，也就是说基于OSGi的系统不会受到运行在其中的 Bundle的影响，不会因为Bundle的崩溃而导致整个系统的崩溃。

◆规范的、可积累的模块

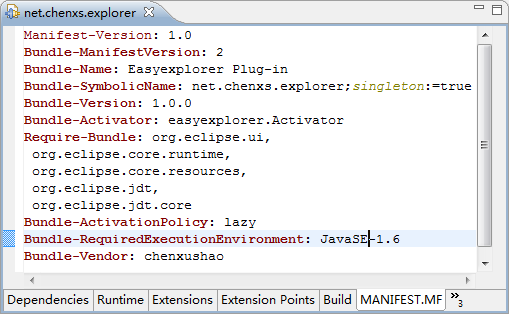
可以将常用的模块划分为不同bundle，积累成bundle库，以方便后续开发重用。

# OSGI理论准备

如下章节将描述本次分享所会涉及到的一些OSGI理论。

## 什么是Bundle

Bundle是OSGI环境中最小的部署单元。从本质上来讲，Bundle其实就是一个jar文件，这个jar文件和普通的jar文件唯一的不同地方就是META-INF目录下的MANIFEST.MF文件的内容，关于Bundle的所有信息都在MANIFEST.MF进行描述，可以称它为bundle的元数据，这些信息中包含有象bundle的名称、id、描述、开发商、classpath、需要导入的包以及输出的包等信息。如下图即为某bundle的Manifest.MF文件，它描述了该bundle的基本信息，导入和导出等约束。



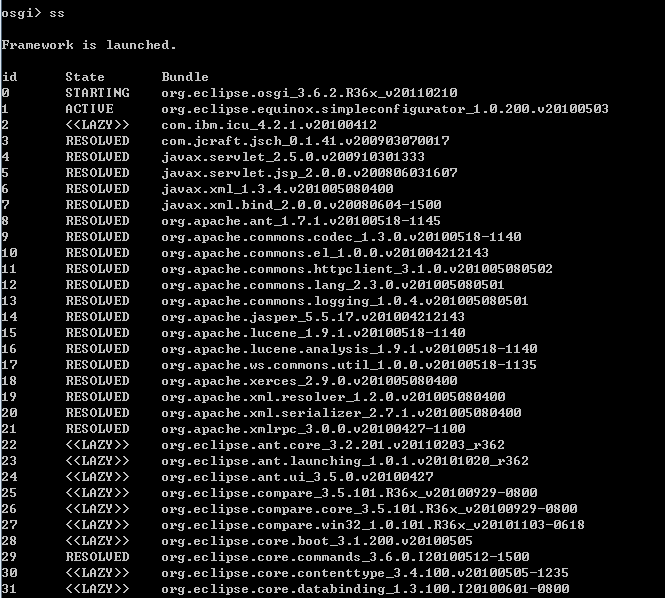
## 什么是系统Bundle

框架本身也是一个bundle。这个bundle称之为系统Bundle。系统Bundle是一种特殊的Bundle，它的启动在任何其它的Bundle之前，并且不能被卸载（Uninstall），系统Bundle状态的变化会发出FrameworkEvent事件。在Equinox平台中，这个Bundle就是org.eclispe.OSGi\_3.4.0.v20080605-1900.jar(不同版本eclipse版本号不同)。



## OSGI控制台

通过OSGI控制台，您可以对系统中所有的bundle进行生命周期的管理，另外也可以查看系统环境，启动、停止整个框架，设置启动级别等等操作。如下图，键入 ss 就可以查看所有 bundle的状态：



在控制台中键入help，可以获得所有命令的列表。下面列举常用命令如下。

|  |  |
| --- | --- |
| **命令** | **描述** |
| **start** | 启动给定了 ID 或符号名称的包 |
| **stop** | 停止给定了 ID 或符号名称的包 |
| **install** | 为当前实例添加一个给定了 URL 的包 |
| **uninstall** | 删除当前实例的具有给定 URL 的包 |
| **update** | 为当前实例更新给定 URL 的包 |
| **active** | 列出当前实例中所有活动的包 |
| **headers** | 列出具有给定 ID 或符号名称的包的标头 |
| **ss** | 列出在当前实例中注册的所有包的简短状态 |
| **diag** | 在给定 ID 或符号名称的包上运行诊断程序 |

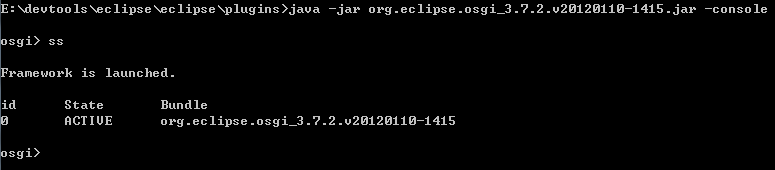
## Equinox的控制台

### 如何启动Equinox框架？

java -jar org.eclipse.OSGi\_version.jar –console

或

java -jar org.eclipse.equinox.launcher\_version.jar –console



### 如何停止Equinox框架？

在OSGi控制台键入exit即可。

### 如何在Equinox上安装新的Bundle?

install reference:file:plugins/net.chenxs.explorer\_1.0.0.jar

start bundle\_id





### 如何卸载Bundle？

uninstall reference:file:plugins/net.chenxs.explorer\_1.0.0.jar



## 重新认识Eclipse

OSGI作为一个行业标准，已经被广泛实现和应用于很多知名的产品。比如著名的开源 Eclipse 就是基于 OSGI 标准并且提供了自己的实现。

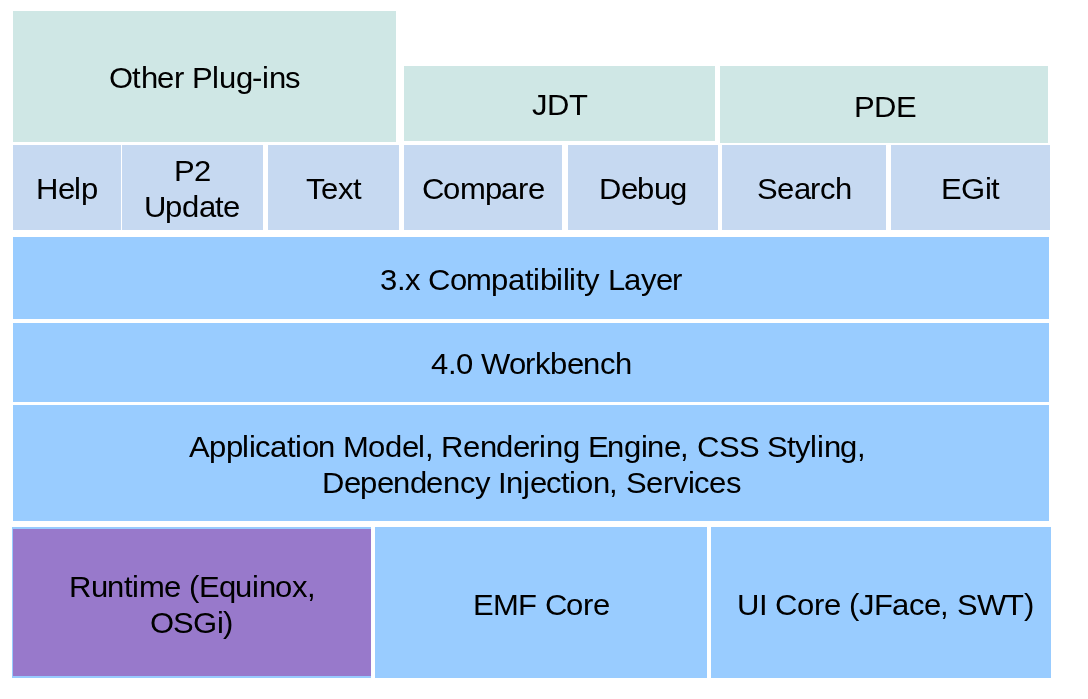
大多数Java开发人员通过作为IDE的功能认识了Eclipse，但它并不仅作为一个Java IDE(实际上，包括C/C++，Ruby，Python等)而存在，它更是一个工业级的产品。

Eclipse IDE 实际上由叫做插件的交互式组件的集合组成。这些插件组成IDE的基础，它们还可用于创建其他桌面应用程序。创建基于Eclipse的应用程序所需的最小插件集称为Eclipse Rich Client Platform（RCP)。IBM的诸多软件比如MQ，Lotus系列软件都是基于RCP来做的。

在我们这次分享的主题当中，我们可以这样认为，Eclipse就是使用OSGI开发的一个应用系统。

Eclipse在本质上是由OSGI(Equinox)驱动的。它由微核(Core Runtime)+OSGI(Equinox)构成其最基本的基石。

Eclipse的系统架构如下图所示。



Eclipse的设计思想是：一切皆插件。在这里需要说明的，在Eclipse中，一个插件就是一个OSGi Bundle。紫色部分即是Eclipse的核心部分， Eclipse核心很小，其它所有功能都以插件的形式附加于Eclipse核心之上。比如图形API (SWT/Jface)， Java开发环境插件（JDT），插件开发环境（PDE）等。

**Eclipse中的系统bundle**

Eclipse以Equinox框架为基石，其SystemBundle是org.eclipse.OSGi\_版本号.jar

**Eclipse的Launcher包装了系统bundle**

org.eclipse.equinox.launcher\_版本号.jar

# 热插拔特性演示

## 何为热插拔

基于OSGI的应用程序可动态更改运行状态和行为。在 OSGi 框架中，每一个bundle上都是可热插拔的，因此，对某一特定的bundle的修改并不会影响到容器中的所有应用，运行中的大部分应用程序依旧能照常工作，部署一个新的Bundle时也不需要重新启动服务器。就像笔记本中使用USB接口外设一样，插入U盘并不需要重启笔记本。在应用开发中，我们无需重新启动整个应用，就能够对应用进行打补丁、升级。热插拔主要强调两点：模块的可替换性和热部署。

因为热插拔，我们能够动态化的搭建系统。能帮助我们做到：

◆ 添加新功能时不需要重启系统；

◆ 修改已存在的功能时不需要重启系统；

◆ 删除一些不需要的功能时不需要重启系统；

◆ 修改系统中的配置时可以不需要重启系统即刻生效；

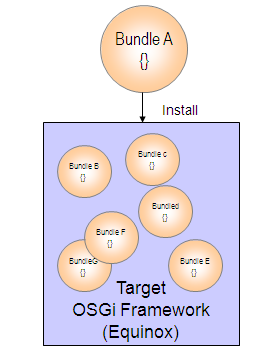
◆ 系统的业务行为可动态的改变。

## 为Eclipse增加新功能

实际上，OSGI目标平台已经为我们准备了N个bundle，它们提供各种各样的服务，OSGi 中，这些 bundle 的名字叫system bundle，就好比精装修的房子，您只需要拎包入住，不再需要自己铺地板，装吊顶了。

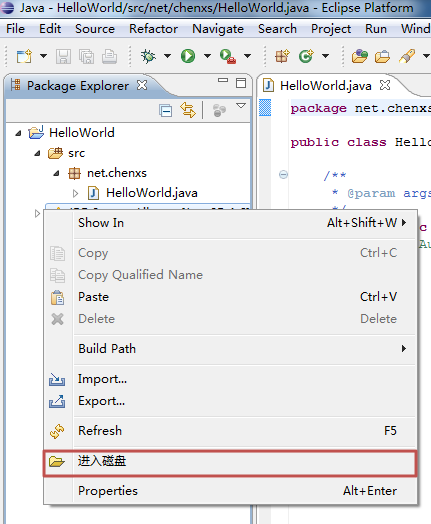
同样，我们要为Eclipse增加新功能，开发一个bundle，将其安装至Eclipse平台中即可。

以Equinox为例，OSGI框架的架构如下图：



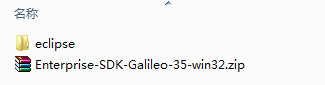
## 要实现的功能与效果

为Eclipse添加一个点击工程/文件夹/文件进入相应的磁盘目录功能。



## 开发一个Bundle

### 开发工具

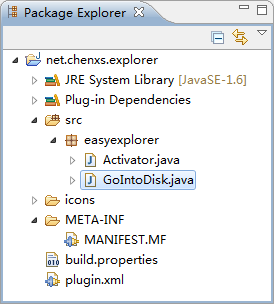


### 测试环境



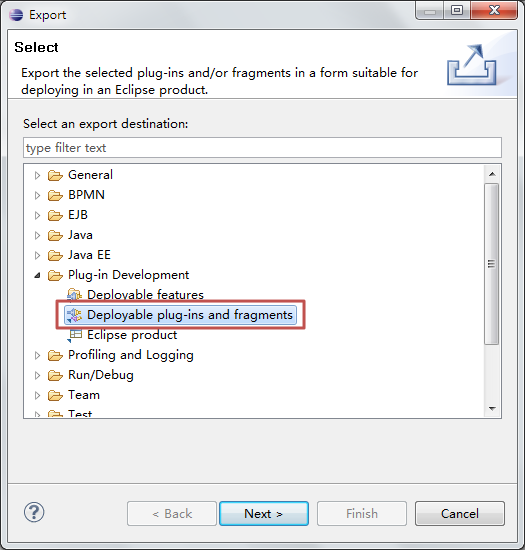
### Bundle结构

要开发的net.chenxs.explorer的结构：

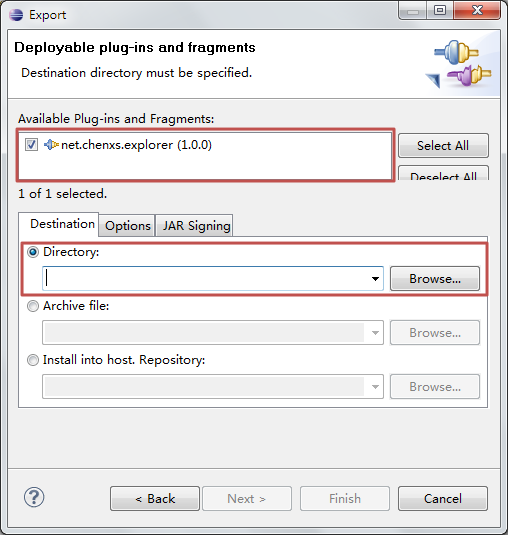


### 导出Bundle

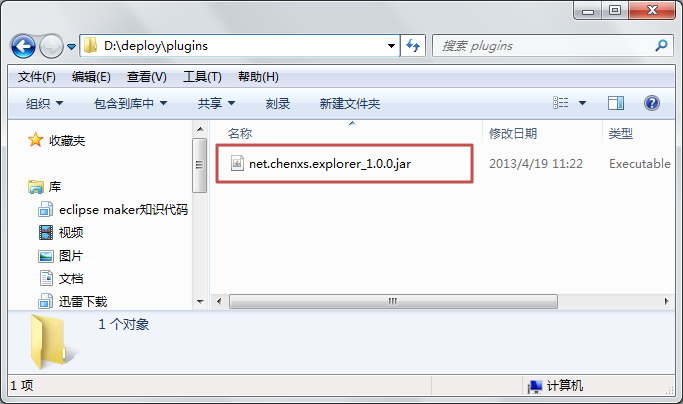
Step1:



Step2:



Step3:



### 部署Bundle

Step1: 通过OSGI控制台启动Eclipse

java -jar plugins/org.eclipse.equinox.launcher\_1.0.0.v20070208a.jar -console

Step2:拷贝bundle至Eclipse\_Home\plugins\目录

Step3:安装bundle

install reference:file:plugins/net.chenxs.explorer\_1.0.0.jar

Step4:启动bundle

start bundle\_id

Step5:卸载bundle

uninstall reference:file:plugins/net.chenxs.explorer\_1.0.0.jar

# 探讨