OSGi 原理与最佳实践学习笔记

（V0.1 2012-05-29）

作者：huangyineng

个人站点：www.naxsu.com

Itway.iteye.com

交流社区：bbs.naxsu.com

OSGi 简介

OSGi 是什么

下面来看看“维基百科”给出的解释：

OSGi（Open Service Gateway Initiative）有双重含义。一方面它指 OSGi Alliance 组织；另一方面指该组织制定的一个基于 Java 语言的服务（业务）规范——OSGi 服务平台（Service Platform）。

OSGi Alliance 是一个由 Sun Microsystems、IBM、爱立信等于 1999 年 3 月成立的开放的标准化组织， 最初名为 Connected Alliance。该组织及其标准原本主要目的在于使服务提供商通过住宅网关，为各种家庭智能设备提供各种服务。目前该平台逐渐成为一个为室内、交通工具、移动电话和其他环境下的所有类型的网络设备的应用程序和服务进行传递和远程管理的开放式服务平台。

该规范和核心部分是一个框架 ，其中定义了应用程序的生命周期模式和服务注册。基于这个框架定义了大量的 OSGi 服务：日志、配置管理、偏好，HTTP（运行 servlet）、XML 分析、设备访问、软件包管理、许可管理、星级、用户管理、IO 连接、连线管理、Jini 和 UPnP。

这个框架实现了一个优雅、完整和动态的组件模型。应用程序（称为 bundle）无需重新引导可以被远程安装、启动、升级和卸载（其中 Java 包 / 类的管理被详细定义）。API 中还定义了运行远程下载管理政策的生命周期管理。服务注册允许 bundles 去检测新服务和取消的服务，然后相应配合。

OSGi 原先关注于服务网关，其实可用于多个方面。现在 OSGi 规范已经用于从移动电话到开源的 Eclipse（其中包括了与 IBM 的 OSGi 框架 SMF 兼容的开源版本）。 OSGi 服务平台的应用包括：服务网关、 汽车、移动电话、 工业自动化、建筑物自动化、 PDA 网格计算、娱乐（如 iPronto）、和 IDE。

OSGi 规范是由成员通过公开的程序开发，对公众免费而且没有许可证限制。但是 OSGi Alliance 的兼容性程序只对成员开放，目前有 12 个兼容的实现。

2003 年 Eclipse 选择 OSGi 作为其插件的底层运行时架构。Equinox project 对该理念进行了实验，2004 年 6 月在 Eclipse3 R3 中发布。ProSyst 是面向 OSGi 开发者的 Eclipse 插件。

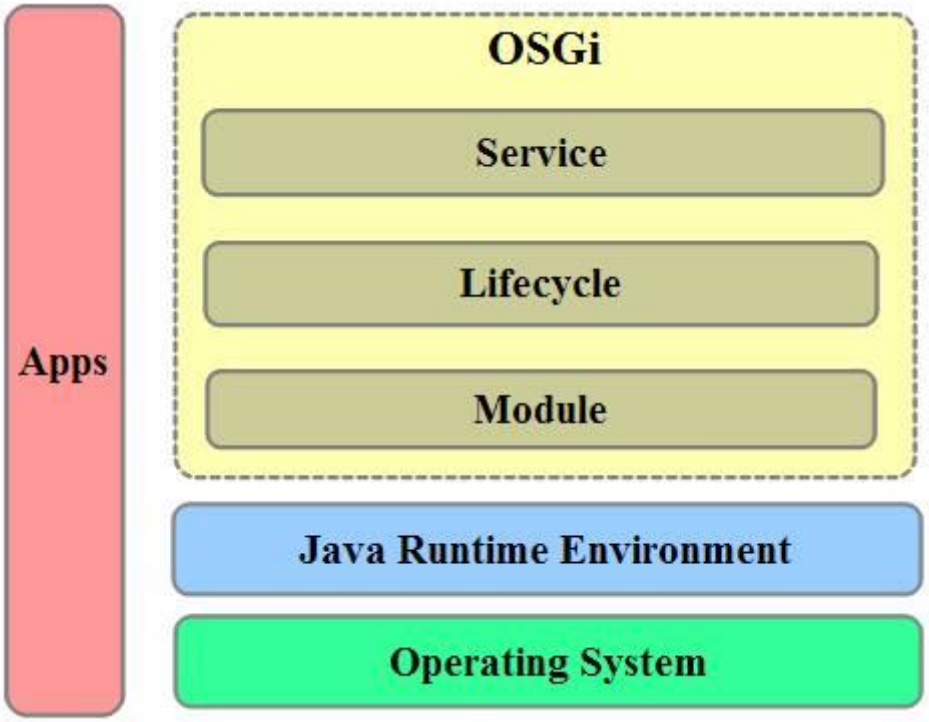
2003 年 10 月，诺基亚、摩托罗拉，ProSyst 和其他 OSGi 成员组建了 Mobile Expert Group (MEG)为下一代智能手机规范业务平台，做为对 MIDP 和 CDC 的补充。

OSGi 的主要职责就是为了让开发者能够建动态化、模块化的 Java 系统。

OSGi 的组成

通常，OSGi 框架从概念上可以分为三层：模块层、生命周期层和服务层。各层的主要功能如下：

- Module Layer —— 模块层主要涉及包及共享的代码；
- Lifecycle Layer —— 生命周期层主要涉及 Bundle 的运行时生命周期管理；
- Service Layer —— 服务层主要涉及模块之间的交互和通信。



OSGi 层次结构

模块层

Bundle 是 OSGi 中的基本组件，其表现形式仍然为 Java 概念中传统的 Jar 包，同时通过 META-INF 目录下的 MANIFEST.MF 文件对其予以进一步的定义。通常一个 MANIFEST.MF 文件的内容如下所示：

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Util
Bundle-SymbolicName : com.ibm.director.la.util
Bundle-Version : 1.0.0.qualifier
Bundle-Activator : com.ibm.director.la.util.Activator
Bundle-Vendor : IBM
Bundle-RequiredExecutionEnvironment : JavaSE-1.6
Import-Package : org.osgi.framework;version="1.3.0"
Bundle-ActivationPolicy : lazy
Export-Package : com.ibm.director.la.util;uses:="org.osgi.framework"
Bundle-ClassPath : libs/jfreechart-1.0.13-swt.jar,
libs/jfreechart-1.0.13.jar,
libs/jfreechart-1.0.13-experimental.jar
```

MANIFEST.MF 文件示例

也就是说，MANIFEST.MF 文件存储的实际上是 Bundle 的元数据，元数据的内容能够精确的定义 Bundle 的各种特征，同时能够更好的对 Bundle 进行标识同时帮组用户的对 Bundle

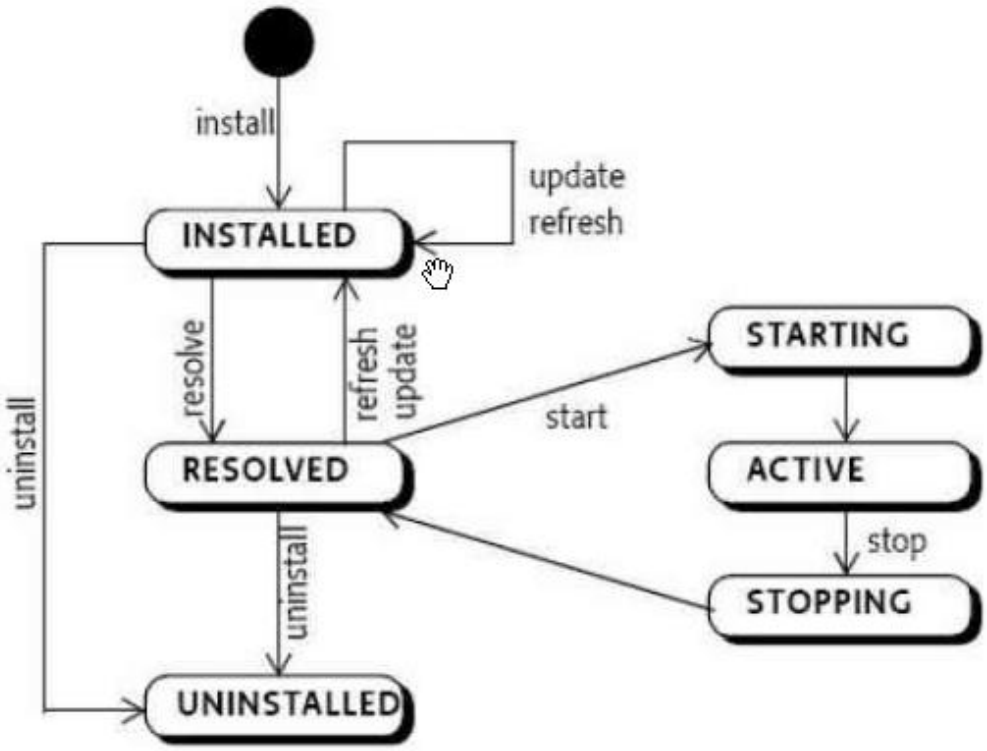
进行理解。例如，通过配置 Import-Package 可以指明当前 Bundle 需要哪些其他的包，而通过配置 Export-Package 则可以表示当前 Bundle 的哪些 package 对外部可见。不难理解，通过这种方式可以有效的对 Bundle 内部和外部进行隔离。其他更多配置请参考相关资料。

Bundle 可以被动态地安装、启动、停止和卸载。
Bundle 是服务(Service)和组件(Component)的载体。
在 OSGi 中，每个 Bundle 都有自己独立于其他 Bundle 的 ClassLoader，正因为这样，各个 Bundle 内部的类是隔离的。

Bundle 之间的交互方式：
1.通过 Package 的 Export(对外暴露自己的一个或多个 package)和 Import(导入别人的一个或多个 package)来进行。
2.通过 Service 的方式进行。一个 Bundle 作为 Service 提供方，对外提供 Servcie .使用者可以查找到提供的 Service. 并使用这个 Servîce. 而提供/使用 Service 又存在两种方式：一种是经典的做法，通过 BundlcContext (Bundle 的上下文)来提供和获取.一种是使用 Declarative Service 来实现.

生命周期层

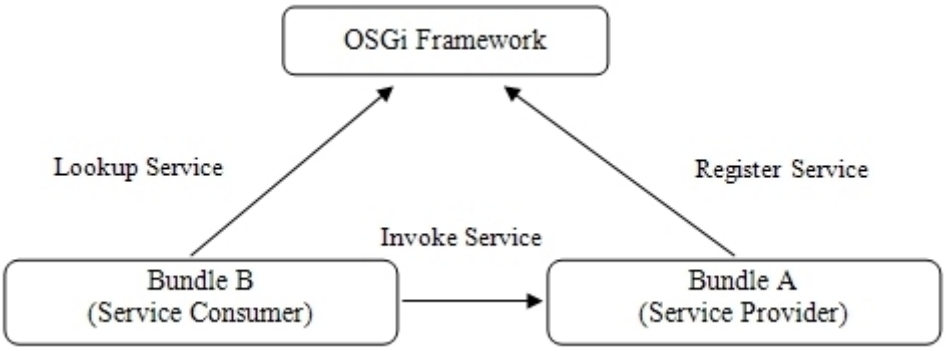
Bundle 的生命周期被 OSGi 框架所管理，具有如下几个状态：INSTALLED 、RESOLVED 、UNINSTALLED 、STARTING 、ACTIVE、STOPPING.状态之间的转换关系如下图：



Bundle 状态转换关系图

服务层

一个 OSGi Service 就是注册到 OSGi 框架中的一个 Java 对象。这注册的时候可以设置这个 Service 的属性。而在获取 Service 的进候可以根据属性进行过虑(Filter)，Bundle 可以通过 Bundle 的上下文去注册 Service 或去查询 Service。

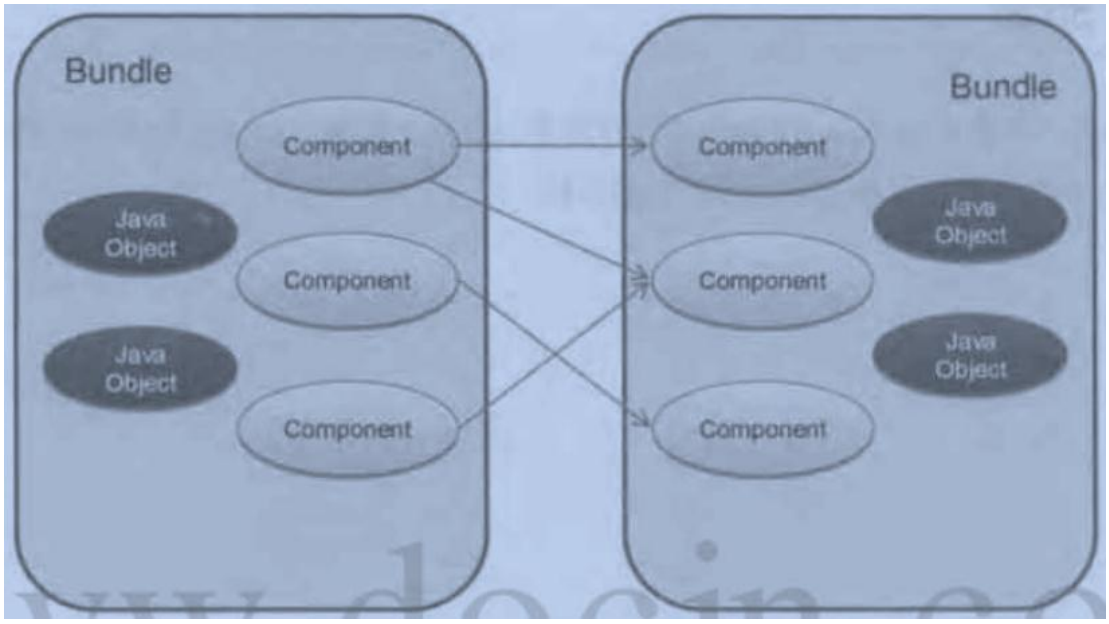


服务模型示例图

Service-Oriented Component Model (SOCM)

首先来看 Component 的慨念.Component 和 Service 从定义上看差不多，任何-个普通的 J ava 对象都可以通过配置文件中的定义而变为一个 Component.Component 对外提供了服务并且可以使用其他 Component 提供的服务，Component 的生命周期被 OSGi 框架所管理.我们可以看到，Component 是提供和使用服务的另外一种方式，并且具有生命周期.

SOCM 在字面上的意思就是面向服务的组件模型.在这个模型中.Component 是服务的载体，提供对外使用的服务并可能使用外部的服务，而 Component 存在于 Bundle 之中，系统由多个 Bundle 组成.



SOCM 示意图

Declarative Service (DS)

Declarative Service (DS)是 OSGi Core Framework 的一个标准服务。DS 让我们在 Bundle 中定义 Component，通过配置的方式发布服务、获取服务，以帮助我们实现前面提到的 SOCM。有了 DS，我们就可以按照 Component+Service 的方式进委系统的设计与开发。