

OSGi技术简介



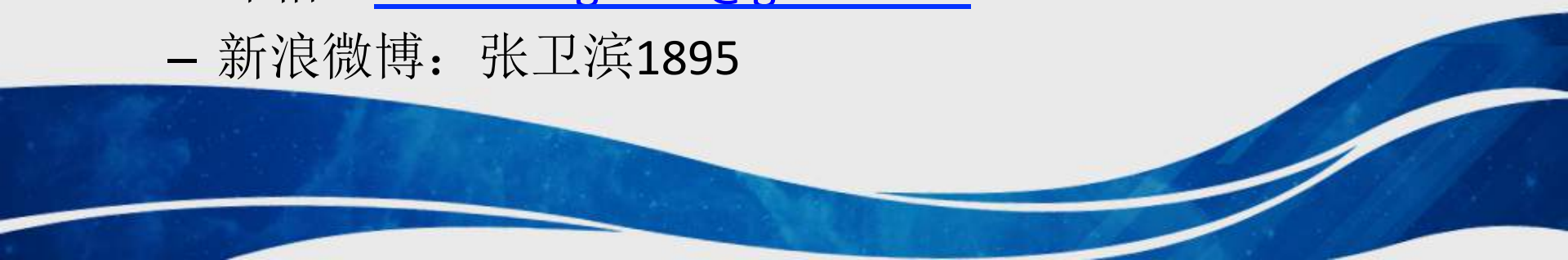
- 工作经历

- 2007.7—2014.08 东软基础软件事业部
- now @NHN
- InfoQ社区编辑 & 技术翻译

- 关注领域

- Web前端开发技术:JavaScript、RIA框架、HTML5
- Java开发技术, 如Spring、Hibernate、OSGi、Eclipse等

- 联系我


- 邮箱: levinzhang1981@gmail.com
 - 新浪微博: 张卫滨1895
- 
- A decorative blue wavy bar at the bottom of the slide.

If you want better modularity, lifecycle management for your module, and totally hot deployment, then, **we have a product for you.**

样例演示。 . .

It's all because of
OSGi technology.



- **What**
 - OSGi的基本概念
 - **Why**
 - 传统Java开发所面临的问题
 - OSGi三层结构
 - **How**
 - 样例开发
 - bundle操作
 - OSGi内置的服务
 - 企业级OSGi
 - Other
- 

Part 01

what is OSGi ?



OSGi的定义



OSGi——Open Services Gateway initiative——*开放服务网关协议*，是由OSGi Allinace制定的Java动态模块化规范。

A trademark
for the technology



现在的定义是The Dynamic Module System For Java，OSGi提供了这些：

- 明确定义了什么是模块；
- 明确定义了模块之间的交互方式；
- 支持模块化部署。
-


what is OSGi ?

➤ Dynamic Module System For Java

意味着基于OSGi就可以**模块化地开发**Java应用；
意味着基于OSGi就可以**模块化地部署**Java应用；
还可以**动态管理模块**。

➤ Universal Middleware

这是OSGi近年来新倡导的理念，目标是让OSGi脱离语言限制，成为所有语言的统一模型。

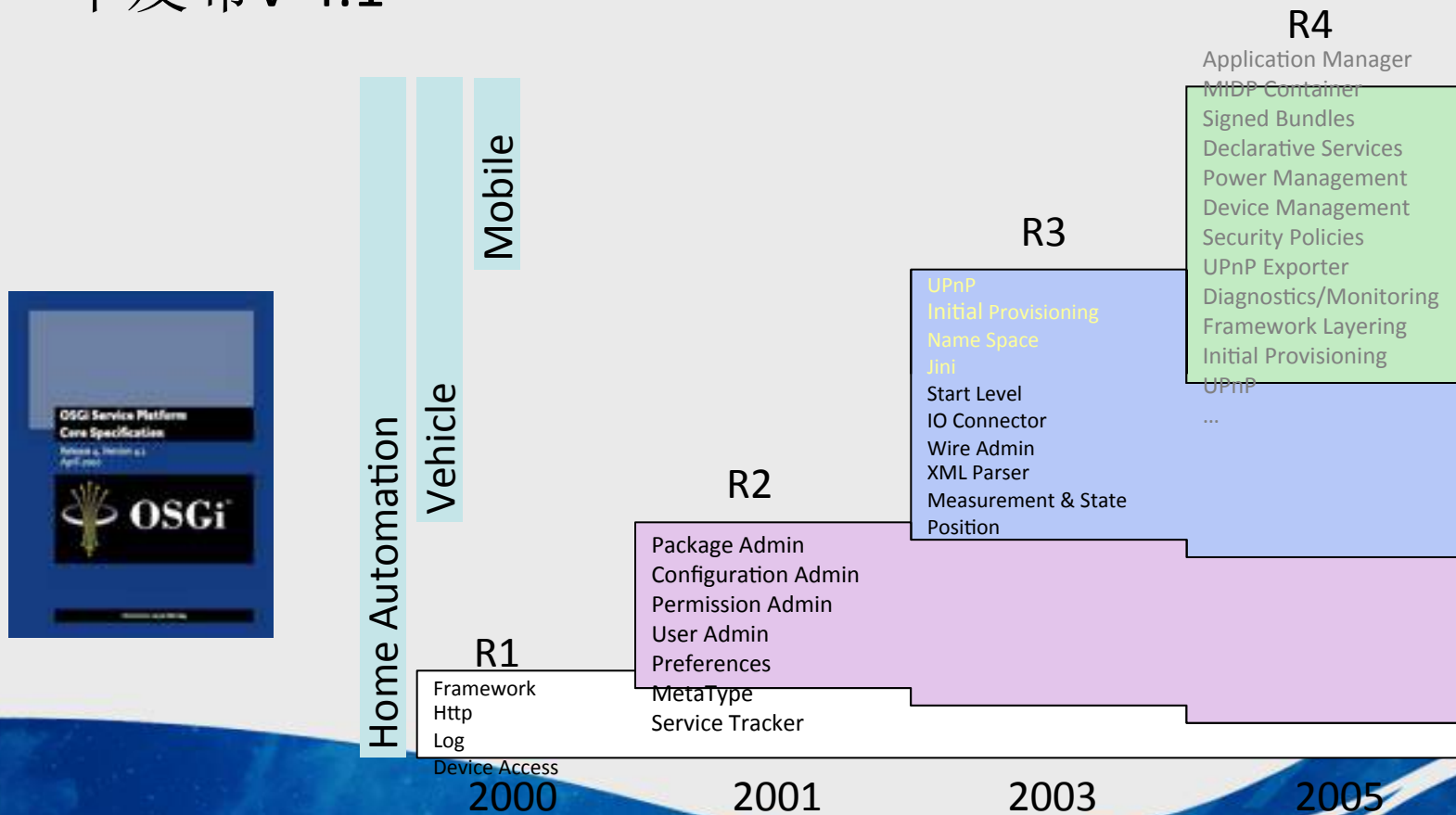


OSGi的发展轨迹

- 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；



- 2000年发布OSGi Service Platform V 1.0，2001年发布V 2.0，2003年发布V 3.0，2005年发布V 4.0，2007年发布V 4.1





PART 01 : what—OSGi概述—发展轨迹

- 2009年，OSGi R4.2版核心规范发布；2010年3月发布企业级规范；2011年发布OSGi R4.3核心规范；2012年7月，OSGi R5核心规范和企业级规范；2014年7月OSGi R6核心规范发布。
 - 从R 4.2版规范开始，重心开始转向企业级应用。
 - R 4.2企业级规范中添加了远程服务、Blueprint以及JNDI、JDBC、JPA等大量与Java EE相关技术的规范。
 - R 5规范中主要包括了基于OSGi技术的模块仓库系统，统一了Equinox P2、OBR等相关技术。
 - R 6添加了注解、数据传输对象等一些技术。

- Eclipse V3.0采用OSGi，是OSGi历史上的里程碑事件，使其脱离移动设备和嵌入式领域；
- 同时Eclipse推出了OSGi RI: Equinox；



- 近两年来OSGi迅猛发展，已经成为业界焦点



PART 01 : what—OSGi概述

谁在使用OSGi :

IBM

WebSphere

Eclipse

Apache

Camel、Karaf、ServiceMix

Siemens、Nokia、BMW、Cisco、NASA

OSGi联盟成员:

IBM、Oracle、NEC、Adobe

大量基于OSGi的创业公司（云计算与物联网领域）:

Paremus、amdatu

A decorative blue wavy line with white highlights, flowing from the bottom left towards the bottom right of the slide.



OSGi实现

- **Equinox**

Realized by Eclipse team;

<http://www.eclipse.org/equinox>.

Used by Eclipse RCP & Eclipse other products.

- **Felix**

Realized by Apache Team;

<http://felix.apache.org>.

- **Knopflerfish**





Bundle

bundle的定义:

A physical unit of modularity **in the form of a JAR file containing code, resources, and metadata**, where the boundary of the JAR file also serves as the encapsulation boundary for logical modularity at execution time.

- ✓ 在OSGi中这不是一个虚拟的概念，而是一个实体；
- ✓ Bundle是一个普通的jar，只是其META-INF中的manifest.mf中描述了一些标准的模块的信息；
- ✓ OSGi的模块划分在原理上依赖于类加载机制；
- ✓ OSGi容器为每个bundle都创建了不同的ClassLoader。



Manifest文件

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Helloworld Plug-in
Bundle-SymbolicName: helloworld
Bundle-Version: 1.0.0
Bundle-Localization: plugin
Bundle-Activator: helloworld.Activator
Import-Package:
    org.osgi.framework;version="1.3.0"
Export-Package:
    ... ..
```

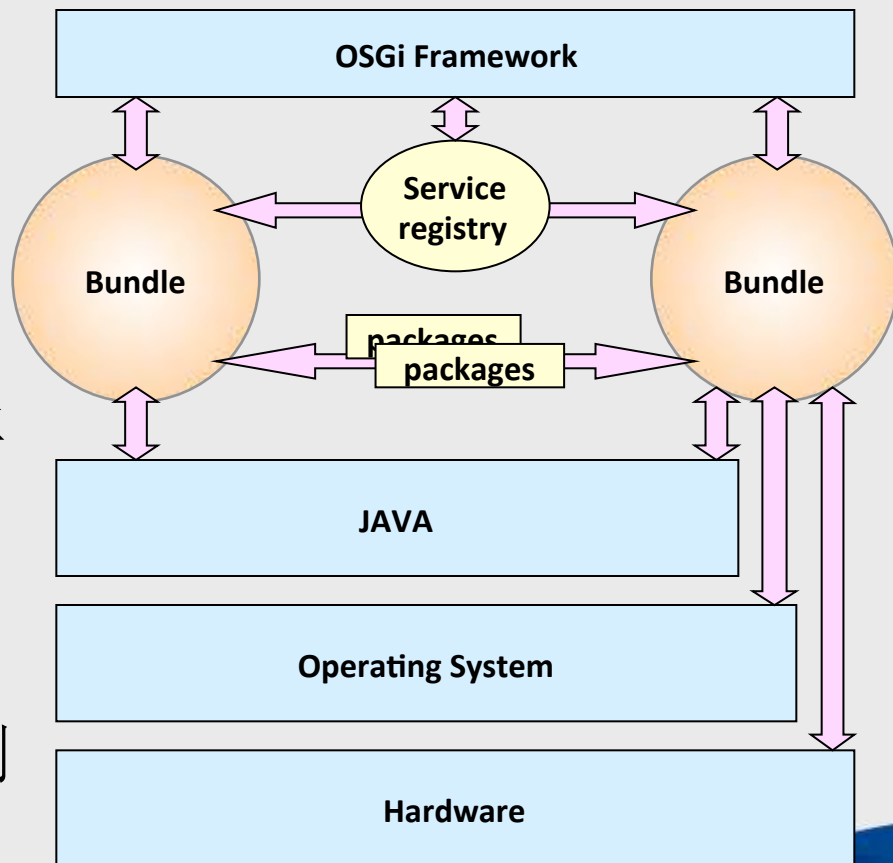
Bundle交互

以package方式交互:

- 通过Export-Package对外提供packages;
- 通过Import-Package使用其他模块的packages。

以服务的方式交互:

- 底层的服务注册和发现机制

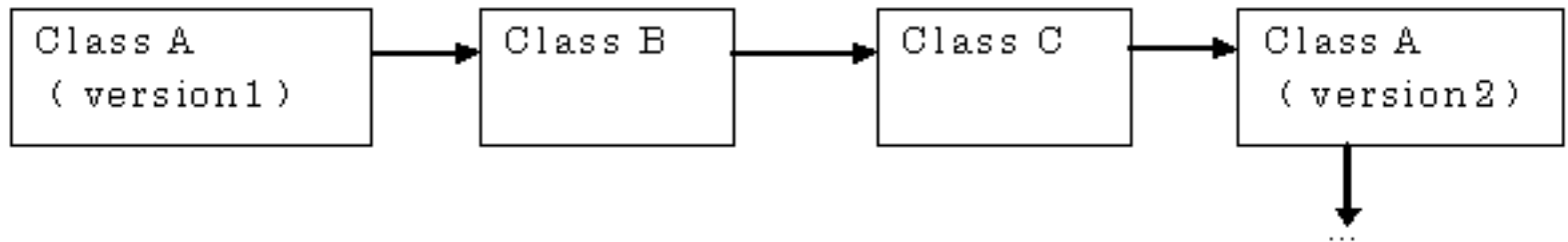


Part 02

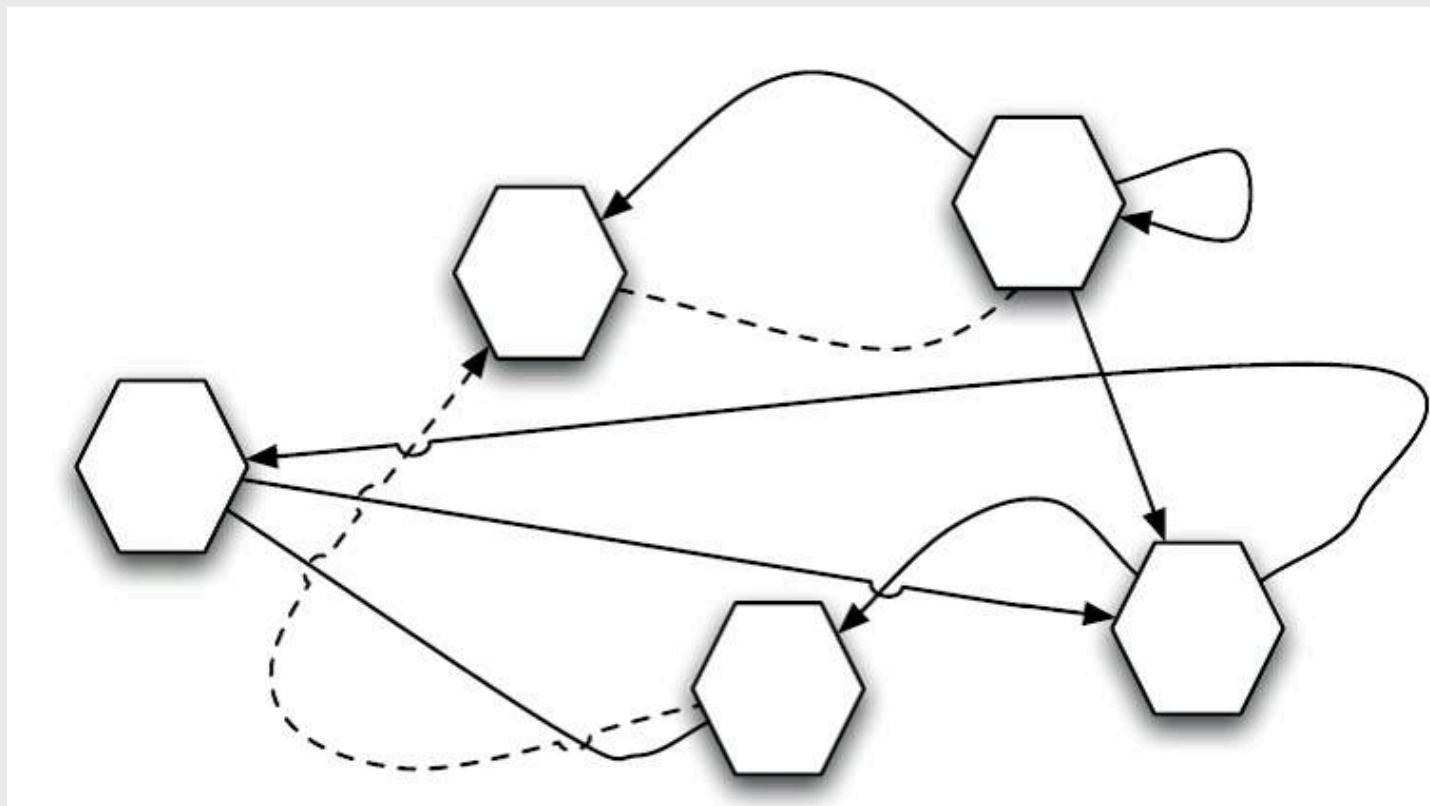
why osgi ?



静态类查找模式所带来的classpath地狱



大型应用代码的复杂性使得应用难以维护





PART 02 : why—OSGi三层结构(三大功能)

if you want a dynamic interface-based development approach

服务层
Service Layer

if you want a way to initialize modules and interact with the module layer

生命周期层
Lifecycle Layer

if you only want better modularity

模块层
Module Layer



模块层

定义了模块化的模型——**Bundle**。

Bundle是开发、部署OSGi应用的基本单元。

提供了**Bundle**的Java包共享、屏蔽的规则。

Bundle通过Export-Package、Import-Package和服务等方式进行交互。

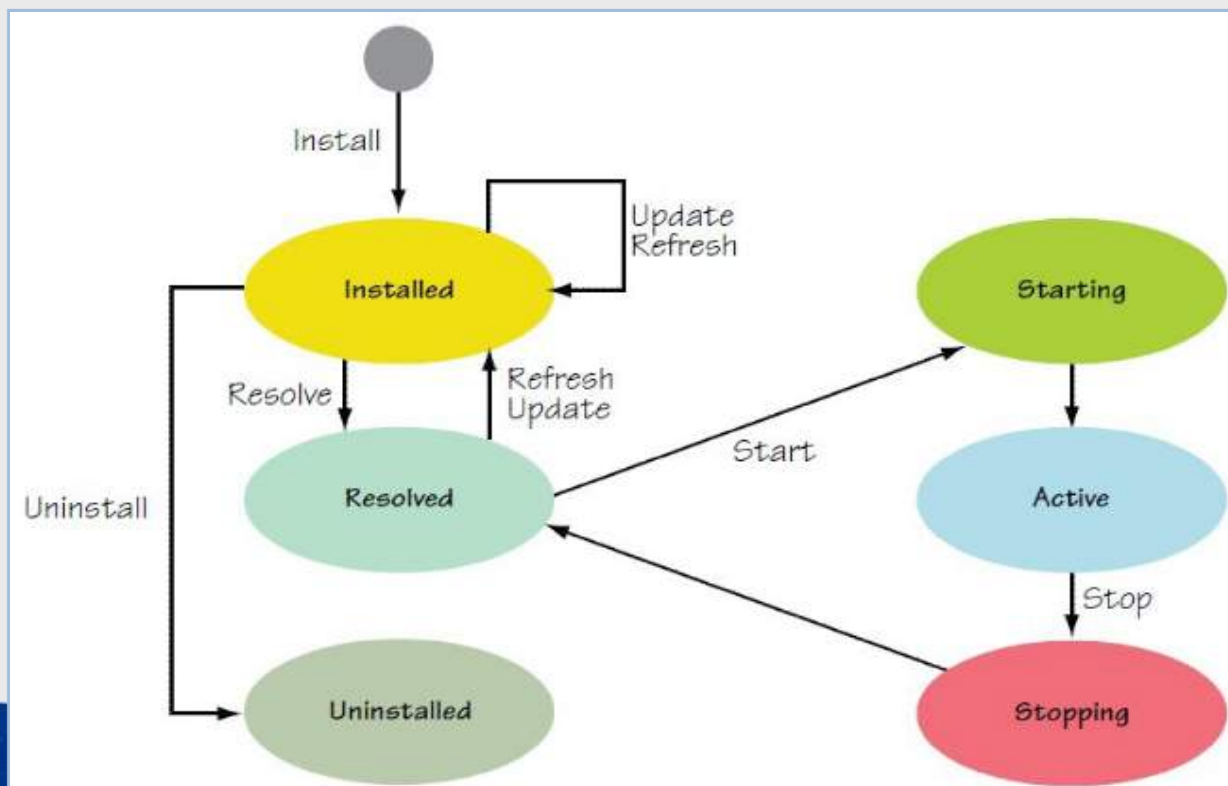
每个**Bundle**有单独的类加载器

该机制保证了Bundle间的物理隔离，是OSGi动态能力的基础。

A decorative graphic at the bottom of the slide consisting of several flowing, wavy blue lines of varying shades, creating a sense of movement and depth.

生命周期层

用于控制Bundle的安全和生命周期操作。
Bundle可处于以下状态中的一种：



Bundle状态解析：

INSTALLED — 成功安装bundle。

RESOLVED — 所有bundle需要的Java类可用。这个状态标志着 bundle已经是启动就绪或者是已经停止。

STARTING — 正在启动bundle。调用了bundle激活器的start方法，而且还没有从方法中返回。

ACTIVE — bundle已经启动完毕，正在运行中。

STOPPING — 正在停止bundle。调用了bundle激活器的stop方法，而且还没有从方法中返回。

UNINSTALLED — bundle已经卸载完毕，不能进入其他状态。



服务层

- 定义了动态协作模型，该模型是一个发布、查找和绑定的模型。
- Bundle可以注册服务、搜索服务、监听服务。
通过BundleContext完成注册和获取服务，
通过ServiceListener监听服务的状态。
- OSGi 4推出了Declarative Services(DS)，DS 提出了完整的 Service-Oriented Component Model (SOCM)，面向服务的组件化模型—更好支持服务注册、获取、监听；OSGi 4.2企业级规范中引入了Blueprint规范，脱胎于Spring DM，更加简便地定义和使用服务。

Part 03

how to use OSGi ?





PART 03 : how—样例开发

开发OSGi平台意味着你需要使用OSGi API编写你的应用，然后将其部署到OSGi容器中。

**Let's head for a demo
'hello world'**

A decorative blue wavy line with white highlights, flowing across the bottom of the slide.

Bundle相关操作

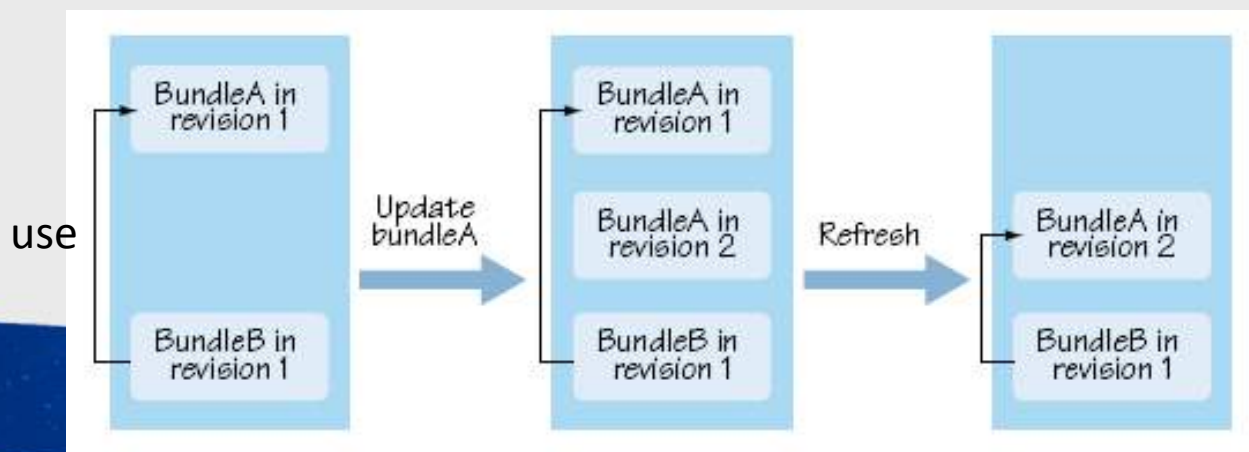
install——安装一个bundle

start ——启动bundle

stop——停止bundle

uninstall——卸载bundle

update——更新bundle





OSGi的服务及实现是目前OSGi社区中更为活跃的话题？

OSGi核心规范所提供的服务

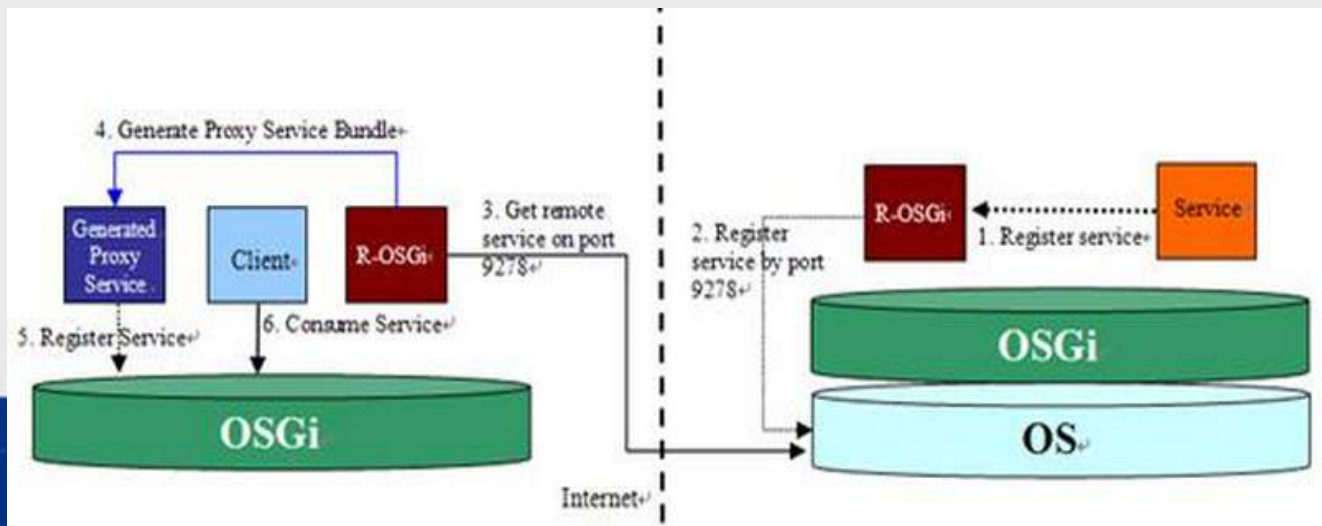
- 启动级别、URL处理、Service Hook、Bundle Hook……

OSGi企业级规范所提供的服务

- 基础服务：日志、用户管理、远程服务、Http……
- Java EE服务：JPA、JNDI、JDBC、JTA、Web应用……
- 其他规范：声明式服务、Blueprint、子系统、Bundle解析与仓库服务……

bundle的分布式服务

- OSGi的服务是同一个JVM之内的服务，有别于Web Service以及REST等理念（SOA in JVM、 μ Service）；
- OSGi同时提供了分布式服务的规范；
- 支持OSGi分布式服务的技术包括：[R-OSGi](#)、Apache CXF D-OSGi。





OSGi仓库理念的发展：

- Eclipse P2 & MarketPlace
- OBR(OSGi Bundle Repository)
- Maven仓库的OSGi Plugin
- 为什么仓库很重要？



OBR(OSGi Bundle Repository)

- OBR提供了一种服务，能够搜寻bundle、解析依赖、部署bundle、管理bundle仓库。
- OBR主要做两件事情：(1)**Discovery**—Provide a simple mechanism to discover which bundles are available for deployment. (2)**Dependency deployment**—Provide a simple mechanism to deploy a bundle and its transitive set of dependencies.
- 分布式资源以“组件仓库”作为组织形式，每个仓库维护一个与其对应的资源描述文件(repository.xml)，它作为一个基于XML的元数据文件，描述资源基本信息、提供能力(export package, provide bundle, export service)、所需依赖(import package, require bundle, import service)等等。

为什么需要企业级OSGi？


- OSGi与Java EE原本是两个并行发展的领域
- 传统企业级应用的功能较为复杂，代码量较大，难以维护和管理。
- 依赖管理复杂，缺少JAR包或版本冲突问题折磨开发人员。
- 不同应用服务器实现机制存在较大差异，不同的应用服务器使用不同了JAR，会导致应用出现迁移问题。



企业级OSGi现状

- 参考实现：Apache Arise; Eclipse Gemini;
- OSGi Web Application Bundle, 将Web应用打包为WAB格式, 运行在支持企业级OSGi规范的容器中;
- 以服务的方式使用JPA与事务;
- R 5规范中新增的打包格式: ESA (Enterprise Subsystem Archives), 基于仓库进行应用装配;
- JNDI、JMX、JTA……

最佳实践

- 谨慎设计模块以及模块间的交互协议；
 - 高内聚；
 - 可重用；
 - 非循环依赖；
 - 谨慎对待版本号；
 - 面向接口编程，依赖于抽象元素；
 - 借助构建工具，强制依赖关系。
- 

互补还是互掐？

- Java模块化历史上的恩怨纠葛；
- 一再一再跳票的Jigsaw；
- Jigsaw与OSGi的不同；
 - 优先解决的是JDK本身的模块化；
 - Jigsaw通过module-info.java声明模块信息；
 - 没有模块的生命周期管理
- 未来会怎样？
 - penrose



PART 03 : how—可选的软件栈

- 工具
 - Eclipse PDE
 - Bndtools
- 规范实现
 - Felix
 - Equinox
- 企业级OSGi的部署环境
 - Apache Karaf、Eclipse Virgo
 - IBM WebSphere、GlassFish、JBoss

谁偷了谁的概念？

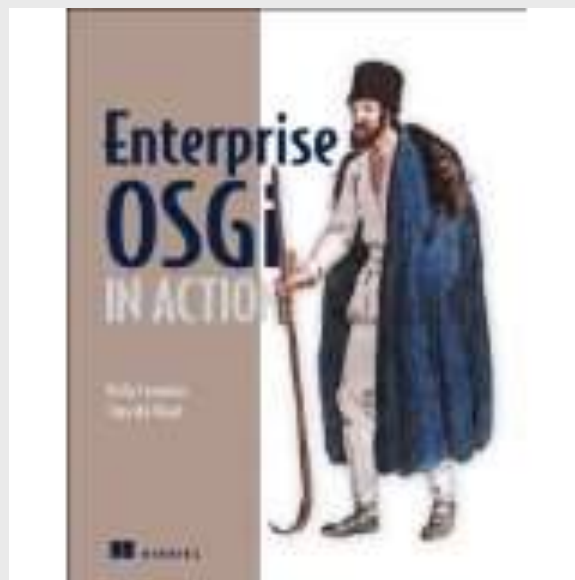
- 微服务是最近非常火爆的概念；Spring Boot & DropWizard
- 倡导服务的拆分，每个服务可以作为单独的应用进行部署和管理；
- 当然也出现了一些负面的声音；
- 似乎与OSGi服务的概念颇为类似？
 - [Peter Kriens的博文](#)

总结：OSGi会给我们带来什么？

架构的理念也许比实际的技术更为重要！



参考书目



Thanks for your attention !



