JBoss AS5, MC, VDF

Jeff Zhang

JBoss Core Developer

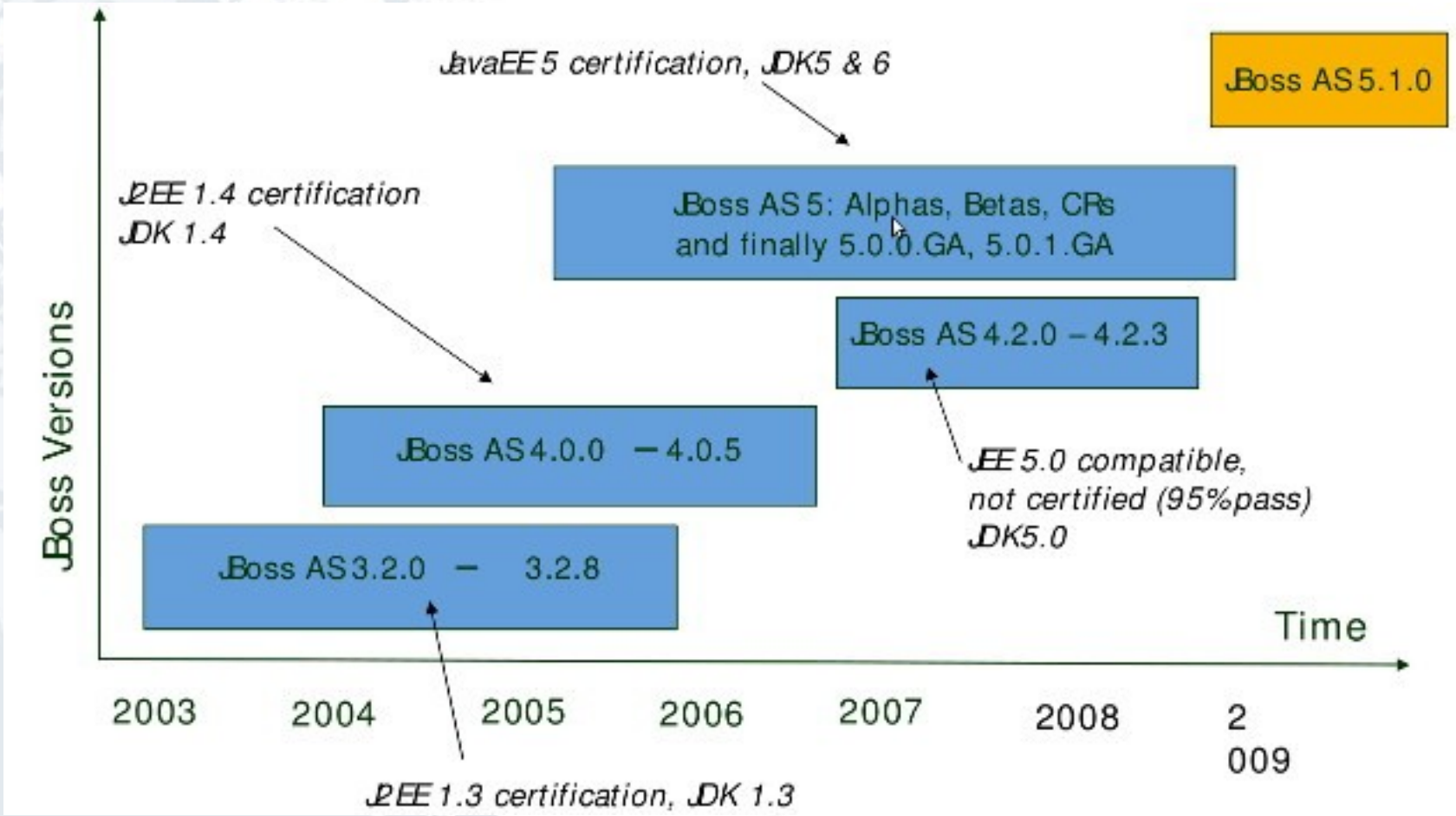
Redhat Senior Engineer

Jeff.zhang@jboss.org

Agenda

- JBoss AS 5 开发历程和常用开发资源
- 功能亮点
- 一些常见问题
- Microcontainer
- Deployer

JBoss AS 时间表


JavaEE 5 certification, JDK5 & 6
JBoss AS 5.1.0
J2EE 1.4 certification
JDK 1.4
JBoss AS 5: Alphas, Betas, CRs
and finally 5.0.0.GA, 5.0.1.GA
JBoss AS 4.2.0 – 4.2.3
JBoss AS 4.0.0 – 4.0.5
JEE 5.0 compatible,
not certified (95%pass)
JDK5.0
JBoss AS 3.2.0 – 3.2.8
JBoss Versions
Time
2003
2004
2005
2006
2007
2008
2009
J2EE 1.3 certification, JDK 1.3

JBoss AS 开发回顾

- 开发周期比较长 >3 年
- 支持 JavaSE 5
- 新的内核， MC(MicroContainer)
- 通过 JavaEE5 TCK
- 2008 年 12 月 AS 5.0 发布
- 2009 年 5 月发布 5.1， EAP5 将基于这个版本

AS 5 和 AS 4 比较

- 内核变化，基于 MC，原来是 JMX
- 新的部署框架， VDF(Virtual Deployer Framework)
- JavaEE5 支持
- 保留了绝大多数原有的服务，迁移方便

JBossAS5 资源

- 项目主页 http://www.jboss.org/jbossas/
  - 可以找到文档，wiki 页面等
- 代码库 http://anonsvn.jboss.org/repos/jbossas/
- 每日构建 http://hudson.jboss.org/hudson/view/JBoss%20AS/

AS 5 代码特点

- 整个 AS 由很多模块构成， trunk 下面的代码越来越少，很多已经迁移出来变成模块，比如 bootstrap
- 和 AS 紧密相关的大多放置在 /projects 下面
- trunk 以后的趋势只有粘合代码，比如 ejb3，里面的是引入 /projects/ejb3 的代码
- 模块和第三方版本可以在 component-matrix/pom.xml 里面统一看到

打包机制

- 目前 Ant 和 Maven 同时在使用
- Trunk 下面很多目录是用 Maven 来管理，编译，单元测试，安装，远程部署
- 整体的模块管理可以通过 trunk 下 pom.xml/build.sh 来完成
- 强烈推荐看 build/build.xml ，可以弄明白打包的细节
- projects 下面各个模块独立打包，有些使用 Ant 编译测试 +Ivy 依赖性管理，只用 Maven 的远程库管理

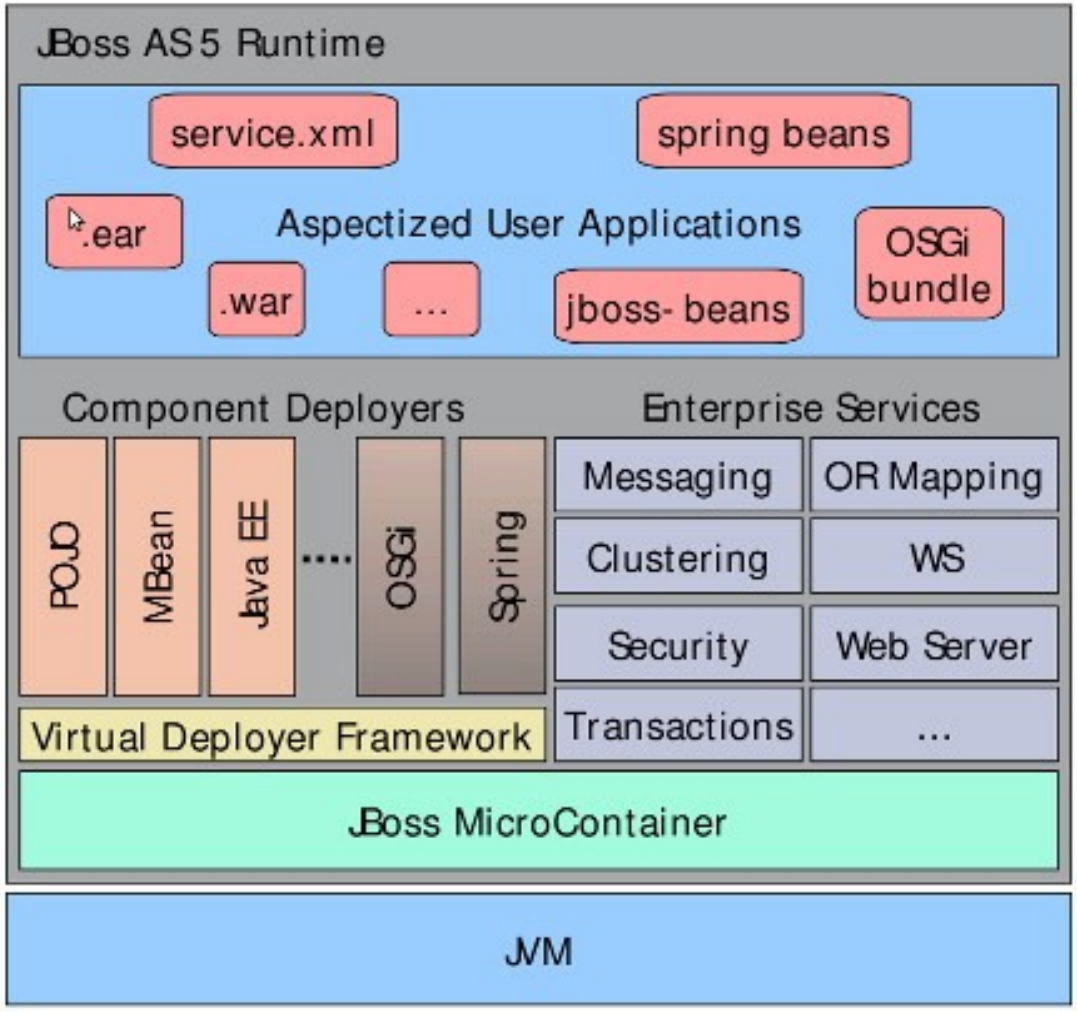
测试

- 绝大多数测试在 trunk/testsuite， 每日都跑
- Trunk 使用 JUnit 3.8.1
- 各个项目有使用 JUnit 4 的
- MC 等基础项目的测试覆盖性做得相当好
- 大多数模块的测试都有抽象测试类作为父类

AS 建议学习方法

- 对 JavaEE 有所了解
- 下载编译，安装，跑几个例子 War, Ejb ear
- 了解 JBoss 目录结构和配置文件， google 一篇”JBoss 目录结构说明”
- 看文档和 wiki 页，如果能获得，红帽的培训教材，很不错 – JB161,JB261,JB336,JB439
- JBoss 论坛，查询和阅读相关帖子
- 书籍 **JBoss in Action**, JBoss4 The official guide , JBoss at Work: A Practical Guide
- 对应模块的测试，代码

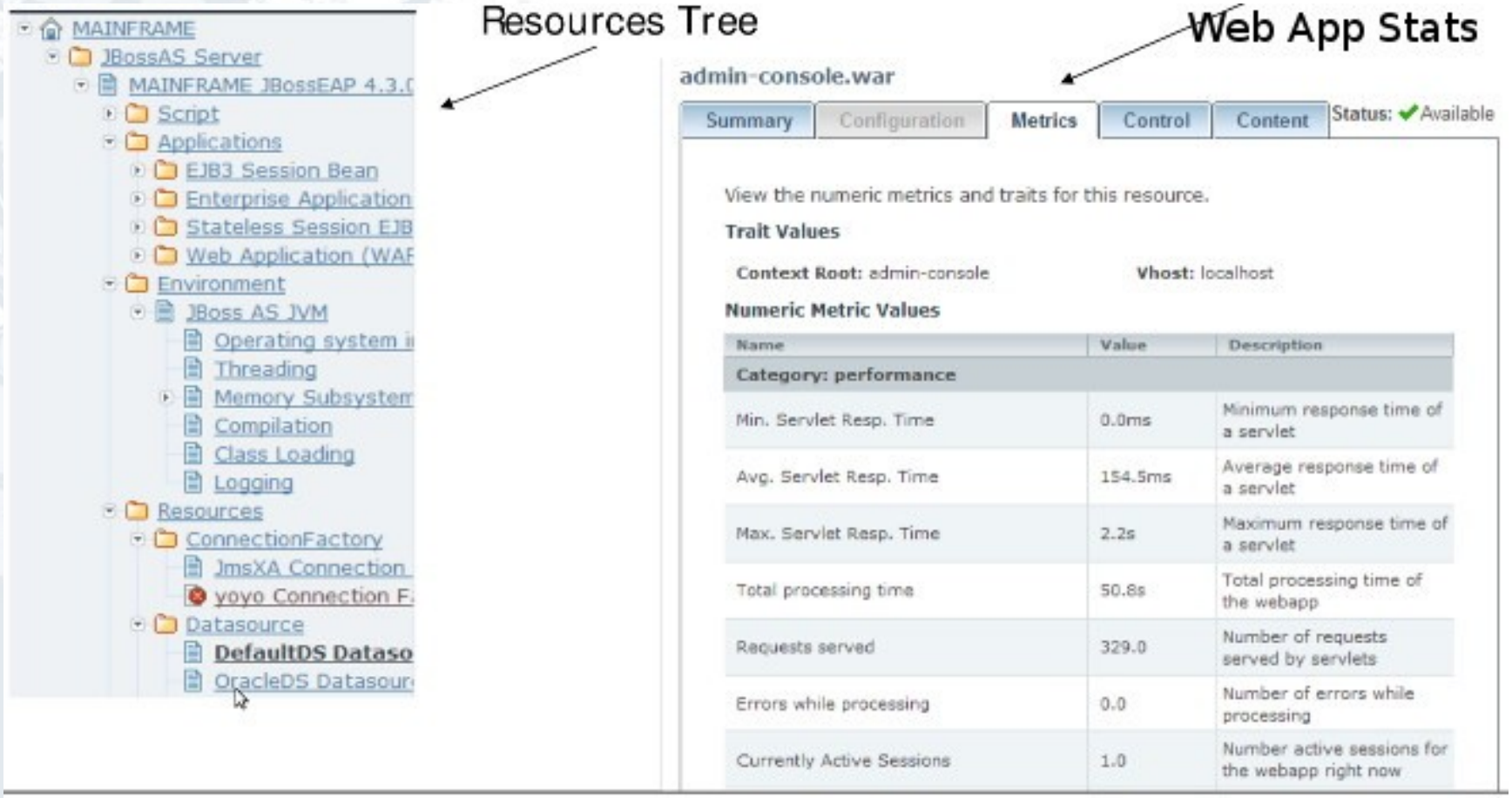
AS 5 体系结构


JBoss AS 5 Runtime
service.xml
spring beans
.ear
Aspectized User Applications
OSGi bundle
.war
...
jboss- beans
Component Deployers
Enterprise Services
POJO
MBean
Java EE
....
OSGi
Spring
Messaging
OR Mapping
Clustering
WS
Security
Web Server
Transactions
...
Virtual Deployer Framework
JBoss MicroContainer
JVM

AS 5.1 中包含了新的管理控制台

- Embedded Jopr
  - http://jboss.org/embjopr
- 部署，配置和监控
  - 数据源
  - 消息队列
  - 应用程序
- 保存修改
- 可扩展的插件结构

Embedded Jopr 图例



JBoss Messaging 1.4.3

- 高性能的 JMS 1.1 实现
- XA 分布式事务处理能力
- Queues&Topics 集群
- 透明的容错和恢复
- 内存中消息复制模式
- 非常大的消息体支持
- 各种数据库支持
- HTTP/SSL 消息传输，穿过防火墙
- …

JBoss Web 2.1.3

- 基于 Tomcat6
- 提供本地代码的 Connector(基于 APR) ， 和 Apache httpd 高吞吐能力匹配
  - 高并发连接 10k+
  - 静态文件处理，减少 CPU 和内存消耗
  - 集成 OpenSSL
  - 多平台支持

JBoss Web Service 3.1.2

- 完备的 JAX-RPC, JAX-WS 支持
- EJB2.1 EJB3 JavaSE 方法发布
  - 支持 MTOM/XOP
  - WS-Security
  - WS-Addressing
  - WS-Eventing
  - WS-Policy
- 支持 3 个 Web Service 实现栈
  - JBoss Native
  - Apache CXF
  - Sun Metro

集群支持

- 集成 JBoss Cache 3.1.0 / Jgroups 2.6.10
- SFSB 复制
- MVCC 多版本并发控制
- EJB3 Entity 和 Hibernate Entity 的缓冲性能
- 共享的 JGroups channels
- JGroups 性能提升

一些常见问题 (一)

- Server 目录下配置项除了原有的 minimal, default, all，多出 standard 和 web
- 运行配置项 ./run.sh –c all
- 不同配置项共用的 lib，移到 common/lib 中，所以 default/lib 为空， all/lib 是 all 需要的
- AS 5 除了 Deploy 目录，多出 deployer，放置部署器

一些常见问题 (二)

- 数据源配置都是通过 -ds.xml ， 由 JCA 层处理
- 数据源会绑定到 JNDI 的 Context 上
- 生产环境必须重新配置 ds.xml ， 默认的 Hsqldb 需要被替换。参考写法在 docs/examples/jca 中
- 数据库连接密码可以加密处理
- 连接池调整是调优的重要工作
- 运行时信息可以通过 JMX 查询和修改

一些常见问题 (三)

- Conf/bootstrap.xml 是关键启动配置文件，引入 bootstrap 目录下的文件
- 定义的 bean 由 MC 创建，注入属性值
- 原来 jboss-service.xml 文件依然存在，但很多服务已经移到 jboss-bean.xml 中。但依然定义了一些重要的服务，如 logging, threadpool, jndi, JAAS, JMX service, remoting 等服务配置。这些服务依然通过原有 ServiceMBean 方式配置
- 未来 JBoss-service 将被移除

一些常见问题 (四)

- 配置 conf/login-config.xml，定义不同包中的日志等级，对于监视应用状态，排除 BUG 很有帮助
- 一个机器上启动多个实例，会遇到端口冲突，如今修改端口变得容易，只需要定义端口偏移值。参考 conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml
- 通过 JMX-console 查看 JNDIView 非常有用

一些常见问题 (五)

- 默认热部署目录是 deploy，可以修改和自定义。 Conf/bootstrap/profile.xml
- 部署可以拷贝打包文件或者目录，如果是目录，切记先解压 / 拷贝到同一分区其他目录，再移动 (mv)
- 可以重新定义 ROOT
- 可以去除用不到的服务，如 mail, schedule, jsf，等等

微内核 JMX -> MC

- JMX 不支持 POJO
- 受到 JMX 定义集的限制
  - 没有配置的 API
  - 不容易扩展
- 拥有自己的 Ioc 实现，便于扩展和提供新功能
- 向 spring 学习，依赖注入
- 向 guice 学习，基于 annotation 注入
- MC 是 IoC 的完备实现

MC 还包括很多

- 为了服务于 AS， MC 组同时还开发维护着
  - JBoss-cl 类加载模块，负责整体的类加载
  - JBoss-man metatype 元数据类型定义 managed 可管理化，将 pojo 定义为可管理的
  - JBoss-mdr 元数据仓库 元数据管理，保存，查询，抽取，范围定义等
  - JBoss-osgi 提供对 Osgi 支持，开发中
  - Vfs 虚拟文件系统 借鉴 linux，对各种文件资源统一定义
  - JBoss-deployers 部署框架
  - Microcontainer Mc 本部分，包含 dependency 和 kernel， 2.2 以后代码转移到 projects/kernel 目录

理解依赖注入

- 避免在全局范围使用像 server = new Server()
- 而使用类似 MC.getBean(“server”)
- 让容器管理类的生命期
- 更重要的是：容器帮助进行**依赖注入**
  - 原来需要显式的调用 server.setEjbContainer(ejbContainer)
  - 现在定义 <inject bean=”EjbContainer/> 或者在类中 @EjbContainer 就可以由容器自动注入

依赖注入的好处

- 创建对象，生命期，类间引用，统一由容器进行管理。便于开发，维护，调试
- 方便测试。这个太重要了，可以说 TDD(测试驱动开发) 促进了 loc 的发展
- 更容易接口和实现分离。

他山之石

- Spring 无疑是最成功的 Ioc 框架。 MC 的资料很少，关于 Ioc 这一块，可以通过 Spring 书籍学习，基本是一致的
- Guice 基于 annotation 的注入机制是巨大进步 ,MC 也支持 .
- 注入定义 @annotation+xml bean

看看代码

- Dependency
  - 9 种状态 Error/NotInstall/PreInstall/Described /Instantiated/ Configured/Create/Start/Installed
  - 当一个“bean”创建时，会经过以上的 8 个状态
  - 当所有 bean 都到达 Installed 状态时，系统就绪
    - PreInstall 设置 Scope/Classloader
    - Described 描述 POJO 和依赖类信息
    - Instantiated 创建类，传递构造参数，调用工厂方法等
    - Configured 设置属性值
    - create/start/installed 状态变化，调用对应方法

看看代码

- Controller 接口操纵 Bean 状态
- ControllerContext 定义 bean 相关上下文
- DependencyInfo 依赖项信息
- ScopeInfo 范围信息， Bean 的生存范围

看看代码

- Kernel
  - Kernel 中心类
  - KernelInitializer 初始化接口
  - KernelBus 消息传送总线
  - KernelConfig 配置接口
  - KernelController 控制器，继承自 Controller
  - KernelEventManager 事件管理器，管理 KernelEvent
  - KernelMetaDataRepository 元数据仓库
  - KernelConfigurator 配置器用于 bean 配置
  - KernelRegistry 注册器

MC 的特性

- 基于构造和 set 方法的注入方式
- 支持任意类定义方法，包括静态类，抽象工厂等
- 回调定义
- XML 定义以及 Annotation
- 自动装配 autowire
- 自定义 classloader
- 控制模式 auto/on-demand/manual
- 声明到达需要的状态
- 延迟加载 lazy
- 生命期方法定义

MC 的 Bean 定义

- XML 定义的 Schema 可以在 kernel resource 中找到 bean-deployer_2-0.xsd
- annotation 在 org.jboss.beans.metadata.api.annotations 中
- 阅读 tests 是学习的好方法
- kernel/docs 下面有一些文档，可以用 docbook 生成可阅读格式

MC 是 AS 的基础

- 今后 AS 中所有的组件将会用 Bean 方式部署
- 继续兼容 AS4 中 MBean 定义，转换为 MC Bean
- 我们将会看到所有的部署文件 *.xml 最终都是在描述或者定义 bean
- ServiceMBean 依赖的也是 MC

Deployer 框架

- AS 5 最重要改进之一
- 借鉴 Unix 思路，每个工具 (步骤) 都很简单，组合使用则功能强大
- 组件就是 POJO 类 +Metadata
- Deploy 的过程就是把 Metadata 抽取出来，按照一定的规则把 POJO 部署在 MC 之中
- Metadata 是可以转换的
- JavaEE 定义了很多 Metadata

Deployer 基本分类

- 结构 structure
  - 将打包文件进行结构分析
- 解析 Parsing
  - XML 等文件解析为基本 metadata 模型
- 类加载 Classloader
  - 创建类加载器
- 组件 component
  - 将复杂的 metadata 转换为基本的 (unit)
- 实际 real
  - 已经得到的 bean， 通过 MC 部署

部署过程状态

- not_installed
- parse 解析
- post_parse 解析后，用于处理解析字符串未完情况
- pre_describe 描述前
- describe 描述，用于对 metadata 处理，加入信息
- classloader 类加载
- post_classloader Aop 处理 ,metadata 合并等
- pre_real
- real 真正 bean 部署
- installed

Deployer 属性

- inputs/outputs 想象成 unix 管道
- order 部署顺序
- Suffix 可处理的文件名后缀

- 部署框架由众多的接口和抽象类组成
  - eclipse 通过 Deployer 接口的继承关系图可以看到

重要类和接口

- Deployer 接口
- AbstractClassLoaderDeployer 抽象类
- AbstractParsingDeployer 抽象类
- AbstractRealDeployer 抽象类
- Deployers 接口
- StructuralDeployers 接口
- DeploymentUnit 接口
- DeploymentContext 接口

AS 加载过程

- bootstrap 创建 MC，其中包括 KernelDeployer
- 通过 profile service 加载对应 profile 的 bootstrap 目录下的 xml 文件
- 其中 deployers.xml 包含基本的 deployers，主要分为四大类 structure, bean, aop, service
- 随后加载 deployers 目录下的 deployer，比如 ejb3, jbossweb, jbossws, jboss-jca 等
- deploy 目录下的组件由对应的 deployers 载入

deployers 优点

- 结构清楚，每个 deployers 完成一个步骤
- 便于扩展，可以写针对自己应用的 deployer，处理好 metatdata，重用已有的 deployer
- 便于剪裁应用服务器规模，不需要的服务不必启用对应的 deployer

END