一种自 EJB2 向 EJB3 的代码自动转换规则

刘 磊[1,2],倪 宏[2],嵇智辉[1,2],匡振国[1,2]
LIU Lei[1,2],NI Hong[2],JI Zhi-hui[1,2],KUANG Zhen-guo[1,2]

1.中国科学院 研究生院,北京 100039
2.中国科学院 声学研究所 国家网络新媒体工程技术研究中心,北京 100080
1.Graduate University of Chinese Academy of Sciences,Beijing 100039,China
2.National Network New Media Engineering Research Center,Institute of Acoustics,Chinese Academy of Sciences,Beijing 100080,China
E-mail:liul@dsp.ac.cn

LIU Lei,NI Hong,JI Zhi-hui,et al.Principle for automatic migrating from EJB2 to EJB3.Computer Engineering and Applications,2008,44(20):7-10.

Abstract: This paper,according to reference specifications of EJB2 and EJB3,provides a kind of principle for migrating the legacy EJB2-based applications to EJB3-based applications.The migrating is automatic and without any modifications of the legacy code of EJB clients.The automatic migration program is implemented using AST tools,and it's successful to migrate XPETSTORE which is an EJB2 based demo application to EJB3 application with the software.
Key words: Enterprise JaveBeans(EJB);JPA;migration;Abstract Syntax Tree(AST)

摘 要:参考 EJB2 和 EJB3 相关的规范,提出一种自 EJB2 向 EJB3 代码自动转换规则,解决目前基于 EJB2 组件的应用已被业界接受和广泛部署的情况下,EJB2 组件代码自动转换为 EJB3 代码的问题。提出的转换规则保证了 EJB2 的客户端在不修改代码的情况下访问转换后的 EJB3 组件。利用抽象语法树 AST(Abstract Syntax Tree)解析和创建 Java 代码,按照该转换规则,所实现的转换软件已经成功转换了 XPETSTORE 示例。
关键词:企业级 Java 组件;Java 持久 API;代码转换;抽象语法树
DOI:10.3778/j.issn.1002-8331.2008.20.003 **文章编号:**1002-8331(2008)20-0007-04 **文献标识码:**A **中图分类号:**TP311.51

EJB(Enterprise JavaBeans)是用 Java 语言开发的可部署的服务器端组件的体系架构。近年来,EJB2.0[1]、EJB2.1[2]规范被广泛用于开发企业级应用,本文 EJB2 指 EJB2.0 和 EJB2.1。2006 年 5 月 EJB3.0[3]规范最终版本发布,Sun、IBM、Oracle、BEA、JBOSS 等厂商相继推出支持 EJB3.0 的应用服务器。对于现存的基于 EJB2 的应用,其技术过于复杂,维护和二次开发相对困难。在 EJB2 和 EJB3 共存的情况下,一般采用两种处理方式:一种利用 EJB3 应用服务器对 EJB2 的兼容性支持保证 EJB2 和 EJB3 组件共存;另一种方式是手工或者使用工具如 IntelliJ IDEA[4] 辅助升级 EJB2 的应用。前一种方式维护困难,后一种方式手工干预多,易出错,而且 EJB 组件的使用者还需要修改客户端代码,本文提出一种转换规则,将基于 EJB2 的代码自动转换为符合 EJB3 规范的组件,其特点是转换过程是自动的,并且不需要修改客户端代码。

本文结构如下:第 1 章提出 EJB2 向 EJB3 代码转换的问题;第 2 章介绍总体转换规则;第 3 章按照组件类别分别描述转换规则;第 4 章总结本文的工作和后继工作。

1 问题描述

EJB3 组件依赖 Java 元数据标注声明 EJB 组件、接口、生命周期回调方法和资源引用等,减少了接口实现,在持久层使用简单 Java 对象 POJO(Plain Old Java Object)作为域对象,简化了开发和部署。下面简要比较 EJB2 与 EJB3 的组件和客户端。

1.1 组件对比

EJB 会话 Bean 包括 Bean 类、EJBHome 接口、EJBObject 接口,如果需要本地调用,还包括 EJBLocalHome 接口和 EJBLocalObject 接口,Bean 类必须实现 ejbCreate()、ejbActive()、ejbPassivate()、ejbRemove()方法。EJB3 的会话 Bean 仅需要实现商务接口,然后在 Bean 类声明前添加注释 @Stateless 或者 @Stateful,声明会话 Bean 的类型,添加注释 @Remote 或者 @Local 声明远程或者本地接口。

EJB2 容器管理实体 Bean 包括 Bean 类、EJBObject、EJBLocalObject、EJBHome 和 EJBLocalHome 等接口类,Bean 类必须实现 EntityBean 的所有方法。对比 EJB3,EJB3 引入了 JPA,标

基金项目:国家科技支撑计划项目(the National Key Technology R&D Program of China under Grant No.2006BAH02A22)。
作者简介:刘磊(1980-),男,博士研究生,主要从事电信运营支撑管理、宽带网络通信等方面的研究;倪宏(1964-),男,研究员,博士生导师,国家网络新媒体工程技术研究中心副主任,主要研究方向包括多媒体技术、宽带网络通信、嵌入式系统等;嵇智辉,男,博士研究生;匡振国,男,博士研究生。
收稿日期:2008-02-18 **修回日期:**2008-03-24

准化 Java 平台的持久 API，采用 POJO 作为轻量级域对象，采用元数据注释代替 XML 描述符，拥有嵌入、继承等新特性。本文不处理 Bean 管理的实体 Bean，在此不作赘述。

EJB3 的消息驱动 Bean 使用标注描述 Bean 的消息服务的目的地类别和目的地。

1.2 客户端对比

EJB2 组件客户端，通过命名服务先查找到 Home 接口对象引用，再调用 create 方法，获取 Bean 接口对象的引用，最后调用商务方法。EJB3 组件的客户端直接通过命名服务查找到 Bean 接口对象的引用，然后调用商务方法。

对比 EJB2 和 EJB3，从开发者角度来评价，EJB3 比 EJB2 在组件和客户端都简化许多；从测试的角度，EJB3 能实现容器外测试；从部署的角度，EJB3 只需要少量的部署描述，部署描述直接在代码里注释。因此，将 EJB2 应用转换到 EJB3，能降低遗留应用的维护难度，降低 EJB3 应用服务器对 EJB2 的兼容性要求。

2 总体转换规则

代码转换技术相关研究有基于设计模式的代码生成技术[5]、基于模型的转换技术[6]等，本文的代码转换并不将 EJB2 的代码转换为 UML 模型，而是依赖抽象语法树 AST[7]（Abstract Syntax Tree），解析原代码生成新代码，抽象语法树采用 XML 直观地表示源代码的语法结构，对 AST 遍历和操作十分方便，本文采用 Eclipse 的 AST 工具解析和生成 Java 代码。

本文代码转换的输入为基于 EJB2 组件的工程目录，输出为基于 EJB3 组件的新工程目录。转换步骤大体分为三步：第一步，搜索工程目录中的 EJB 部署描述文件，如 ejb-jar.xml、jboss.xml、jbosscmp-jdbc.xml 等；第二步，解析 XML 部署描述文件，获取各组件的部署描述信息；第三步，按照转换规则依次对各组件进行转换。

总体转换规则分为四条：(1)将部署描述信息以注释的方式添加到 Java 文件中；(2)转换过程按照一般类结构顺序进行转换，一般类的结构顺序为包定义、导入声明、类定义、类中成员变量定义、类中成员方法；(3)新增无状态会话 Bean 完成 EJB2 组件中 Home 或者 LocalHome 接口的功能，满足客户端对 EJB2、EJB3 一致性访问的要求；(4)将 CMP 转换为 POJO 轻量级域对象。本文以 JBOSS 应用服务器部署为例描述转换规则，稍作变化便能支持其他应用服务器，下文详述各组件的转换规则。

3 组件转换规则

转换规则按照会话 Bean、实体 Bean 和消息 Bean 的顺序阐述。

3.1 会话 Bean 转换规则

会话 Bean 分为无状态会话 Bean 和有状态会话 Bean 两类，本文采用相同的规则转换这两种会话 Bean。新增无状态会话 Bean，替代 EJB2 会话 Bean 的 home 或者 local-home 接口完成创建会话 Bean 的功能。原 EJB2 客户端先通过命名服务查找到新增无状态 Bean 对象的引用，再调用该引用的 create 方法，获取实际 Bean 接口对象的引用，最后调用实际 Bean 接口对象的商务方法。该规则保证 EJB2 的客户端不修改代码的情况下能调用转换后的 EJB3 会话 Bean。具体转换规则如下：

(1)remote 和 local 接口类转换规则

remote 和 local 接口类的转换分为三步：第一步，在接口声明中删除对 EJBObject 或 EJBLocalObject 的继承；第二步，在接口类中增加对应 home 和 local-home 类中的 create 方法，将 create 方法的返回类型改为 void，有状态的会话 Bean 可能存在多个输入参数的 create 方法，都需要在接口类中声明；第三步，在接口类中添加 remove 方法，会话 Bean 的客户端能调用 remove 方法销毁会话 Bean 分配的资源。

(2)Bean 类转换规则

根据从 ejb-jar.xml 和 jboss.xml 中读取的 EJB 部署描述信息，以 ASTVisitor 遍历 Bean 类，在会话 Bean 类导入声明部分增加 Stateless、Stateful、Resource 等必要的导入声明；在类声明前加入注释 @Stateless 或者 @Stateful 描述会话 Bean 的类别；在类声明前加入注释 @Remote 和 @Local 描述会话 Bean 的远程和本地接口类；在类声明前加入注释 @RemoteBinding 和 @LocalBinding 描述远程和本地的 JNDI，为了保证原 EJB2 的客户端能不作修改的情况下调用会话 Bean，该 JNDI 不直接使用部署描述文件中的 JNDI，而是统一在 JNDI 后追加“Bean”字符串，新增的无状态会话 Bean 使用部署描述文件中的 JNDI；在类声明前加入注释 @Resources 说明该会话 Bean 对资源的引用；在类声明前加入注释 @EJBs 描述该会话 Bean 对其他关联 Bean 的引用，beanInterface 值为关联 Bean 的 home 或者 local-home 类名；在 Bean 类声明中删除 SessionBean 接口，增加实现该 Bean 的 remote 和 local 接口；对 Bean 类中的方法，ejbCreate 改名为 create，该方法将由新增的无状态会话 Bean 调用，在 ejbPassivate、ejbActivate、setSessionContext 方法声明前分别添加注释 @PrePassivate、@PostActivate、@Resource，ejbRemove 方法改名为 remove，并且在该方法声明前添加注释 @PreDestroy，对这些方法添加注释是为了保证会话 Bean 在 EJB3 的应用服务器环境中保持与 EJB2 环境中同样的生命周期。

(3)home 和 local-home 类转换规则

home 类和 local-home 类转换相对简单，在接口声明中删除对 EJBHome 或者 EJBLocalHome 的继承。

(4)新增无状态会话 Bean 的生成规则

新增的无状态会话 Bean 完成 EJB2 中 home 和 local-home 接口创建会话 Bean 的功能，如果原会话 Bean 有远程或者本地接口，则只生成一个满足 home 或者 local-home 接口的无状态会话 Bean；如果原会话 Bean 既有远程也有本地接口，则新增满足 home 和 local-home 的两个无状态会话 Bean。新增的会话 Bean 以 home 或者 local-home 接口类名追加“Bean”字符串命名，以转换后的 home 或者 local-home 接口作为远程或者本地接口，以原会话 Bean 的 JNDI 作为新增会话 Bean 的 JNDI，实现 create 方法，有状态会话 Bean 可能会有多个 create 方法。以 XPETSTORE[8]中 Petstore 会话 Bean 为例，create 方法实现如图 1，通过 JNDI，查找到会话 Bean 接口引用后，调用 create 方法，然后返回查找到的 Bean 接口的引用。在本文转换规则下，EJB2 的客户端无需修改代码，首先根据 JNDI 查找 home 或者 local-home 接口，然后调用 create 方法获得会话 Bean 的远程或本地接口引用，最后调用商务方法。

3.2 实体 Bean 转换规则

实体 Bean 分为容器管理的实体 Bean 和 Bean 管理的实体

```
@Stateless(name="Petstore")
@Local(PetstoreLocalHome.class)
@LocalBinding(jndiBinding="PetstoreLocal")
public class PetstoreLocalHomeBean implements PetstoreLocalHome
{
  public PetstoreLocal create() throws CreateException {
    try {
        InitialContext ctx=new InitialContext();
        PetstoreLocal bean=(PetstoreLocal) ctx.lookup ("PetstoreLocal_bean");
        bean.create();
        return bean;
    } catch(javax.naming.NamingException ne) {
        throw new javax.ejb.CreateException();
    }
  }
}
```

图 1　新增 PetstoreLocalHomeBean 无状态会话 Bean

Bean,实体 Bean 一般由本地接口、本地 Home 接口、远程接口、远程 Home 接口、实体 Bean 实现组成。本地接口和远程接口定义实体 Bean 中字段的设置和获取方法,Home 接口统指本地 Home 接口和远程 Home 接口，定义实体 Bean 的创建、查询和删除方法。实践中远程接口和远程 Home 接口使用较少,基本被摒弃。本文仅考虑将容器管理的实体 Bean 自动转换为 JPA 持久对象。

本文将本地接口的容器管理的实体 Bean 自动转换为一个无状态会话 Bean 和一个 POJO 对象。无状态会话 Bean 完成原 EJB2 中 Home 接口完成的创建、查询、删除实体等功能。转换主要分为三步，下面以 XPETSTORE 中的 OrderItem 订单条目为例进行说明:

(1)生成 POJO 对象

根据 ejb-jar.xml 描述文件中对 OrderItem Bean 的描述,读入 ejb-class 文件，此例为 OrderItemCMP，利用 AST 解析 OrderItemCMP，生成一个 POJO 对象。以 ASTVisitor 遍历 OrderItemCMP，将包定义修改为与本地接口相同的包定义;在导入声明部分加入必要的导入声明，如 Entity、Id、Column、JoinColumns、Table 等；修改类定义，在类定义前增加 Table、Entity 注释,删除 abstract 修饰,改类名为本地接口名,删除实现的 EntityBean 接口，增加 Serializable 接口；删除 ejbCreate、ejbPostCreate、ejbActivate、ejbPassivate、setEntityContext、usetEntityContext、unsetEntityContext 方法，在 ejbLoad、ejbRemove 和 ejbStore 方法声明前分别添加注释 @PostLoad、@PreRemove 和 @PrePersist;修改 getter 和 setter 方法,下文详述;保留 ejb-class 中的其他方法;最后将 POJO 对象保存以本地接口类名命名的类文件,覆盖原本地接口类文件。

对 getter 和 setter 方法的处理是生成 POJO 的关键,本文结合 ejb-jar.xml 和 jbosscmp-jdbc.xml 中域对象的字段描述，生成 POJO 轻量级域对象。如图 2,对于以“get”字符串开始的非字段获取方法,需要在方法声明前增加 @Transient 注释。对于字段获取方法的转换,首先删除 abstract 修饰符;然后增加注释,如果是主键,加入 @Id 注释,如果是域对象的非关联字段,则加入 @Column 注释,如果是关联字段,根据关联关系,增加关联关系注释,如 @OneToOne、@ManyToOne、@OneToMany、@ManyToMany、@JoinTable、@JoinColumn 等注释;然后补全方法体,最后根据字段名和 getter 方法的返回类型生成对应的私有类变量。对于 setter 方法的转换,首先删除 abstract 修饰符,然后补全方法体。自此,POJO 的域对象生成完毕。

```
...
@Transient public double getSubTotal(){
    return Math.max(getQuantity()*getUnitPrice(),0);
}
private Integer orderItemUId;
@Id @Column (name="orderItemUId") public Integer getOrderItemUId(){
  return orderItemUId;
}
public void setOrderItemUId(Integer orderItemUId) {
  this.orderItemUId=orderItemUId;
}
private OrderLocal orderLocal;
@ManyToOne @JoinColumn (name="order_fk") public OrderLocal getOrderLocal (){
  return orderLocal;
}
public void setOrderLocal(OrderLocal arg){
  this.orderLocal=arg;
}
```

图 2　OrderItemLocal 程序片段

(2)local-home 类的转换规则

local-home 类转换相对简单，在接口声明中删除对 EJBLocalHome 的继承，增加 remove 方法的定义,remove 方法的定义与 EJBLocalHome 中的 remove 方法定义相同。

(3)新增无状态会话 Bean

同本文前文会话 Bean 转换规则一样，该新增的无状态会话 Bean 完成 POJO 域对象的创建、查找和删除功能。新增的无状态会话 Bean 类以该实体 Bean 的 ejb-class 包名作为包名,以 local-home 为本地接口，绑定原实体 Bean 的 JNDI，实现 local-home 接口;增加 EntityManager em 私有变量,在该变量定义前添加注释 @PersistenceContext;然后依次实现各创建、查找和删除方法，下文详述；最后保存以 ejb-class 命名的类文件,覆盖原 ejb-class 类文件。

按照 JPA 规范依次实现各创建、查找和删除方法,create 方法是通过重构原 ejb-class 中的 ejbCreate 方法实现的，首先修改方法返回类型为域对象,方法名替换为 create,方法体前面加入域对象变量的定义,变量名为 obj,并且用类默认构造方法初始化该变量,方法体中的 setter 方法将“set”替换为“obj.set”,在 create 方法体返回语句前调用 em.persist(obj),最后删除原返回语句，添加返回 obj 语句。findByPrimaryKey 方法调用 EntityManager 的 find 方法,返回强制类型转换后的变量。其他 find 方法实现,根据 ejb-jar.xml 中 ejb-ql 的定义,将 ejb-ql 中的参数“?1”、“?2”等依次替换为“:arg1”、“:arg2”等,然后调用 EntityManager 的 createQuery 创建一个查询，依次调用 setParameter 设置参数，最后根据返回值是域对象或者集合对象,调用 getSingleResult 或者 getResultList 方法。remove 方法实现类似 find 方法,在方法体最后添加“em.remove(obj)”。OrderItemCMP 转换后的程序片段,如图 3 所示。

3.3　消息 Bean 转换规则

消息 Bean 转换分为三步,转换规则如下:第一步,在 EJB2 消息 Bean 中添加 ActivationConfigProperty、PreDestroy、PostConstruct、MessageDriven 和 Resource 等必要的导入声明;第二步,按照从

```
public OrderItemLocal create (int quantity,double unitPrice) throws CreateException {
   OrderItemLocal obj=new OrderItemLocal( );
…
   obj.setUnitPrice(unitPrice);
   em.persist(obj);
   return obj;
}
public OrderItemLocal findByPrimaryKey(Integer key) throws FinderException {
   OrderItemLocal obj=(OrderItemLocal) em
                .find(OrderItemLocal.class,key);
   return obj;
}
public void remove(Object key) throws RemoveException,EJBException {
   OrderItemLocal obj=(OrderItemLocal) em
                .find(OrderItemLocal.class,key);
   em.remove(obj);
}
```

图 3 转换后的 OrderItemCMP 程序片段

ejb-jar.xml 和 jboss.xml 中读取的消息 Bean 的部署描述符,包括事务类型、目的地类型、订阅持久性、目的地 JNDI 等,以注释的方式添加到消息 Bean 实现类定义前, 删除 MessageDriven-Bean 接口,保留 MessageListener 接口;第三步,在 ejbCreate、ejbRemove 和 setMessageDrivenContext 方法前分别添加注释 @PostConstruct、@PreDestroy 和 @Resource。消息 Bean 的转换规则相对于会话 Bean 和实体 Bean 简单,XPETSTORE 示例中,订单处理和邮件服务采用消息 Bean。

4 总结及进一步工作展望

本文按照上述规则实现了转换程序,并且对开源 EJB 示例 XPETSTORE 应用中的 EJB 组件进行自动转换,手工将 web.xml 中对 EJB2 引用修改为 EJB3 的引用,编译后,成功地在应用服务器 JBOSS4 上部署并运行。本文提出了一种自 EJB2 向EJB3 代码的自动转换规则,特点是 EJB2 代码经过自动转换后,无需修改 EJB 客户端代码,该规则已验证是可行的,代价是增加无状态会话 Bean 完成 EJB2 中 Home 接口类似的功能。目前还需要进一步研究 EJB2 中安全角色权限,容器管理 Bean 的复杂主键、双向关联、复杂 EJBQL 等情况,以及对其他应用服务器的扩展支持。本规则应用广泛,开发人员能用于对基于 EJB2 的应用的自动升级; 企业用户能用于对已经部署的基于 EJB2 的遗留应用进行升级;集成开发工具和应用服务器厂商可以考虑在本规则基础上扩展开发 EJB2 向 EJB3 的代码自动转换工具。

参考文献:

[1] Sun Microsystems.Enterprise JavaBeans specification version:2.0 [EB/OL].[2001-08-22].http://java.sun.com/products/ejb/docs.html.

[2] Sun Microsystems.Enterprise JavaBeans specification version:2.1 [EB/OL].[2003-11-24].http://java.sun.com/products/ejb/docs.html.

[3] Sun Microsystems.JSR-000220 Enterprise JavaBeans v.3.0[EB/OL].[2006-05-08].http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html.

[4] JetBrains.IntelliJ IDEA6.0.6[CP].[2007-10-16].http://www.jetbrains.com/idea.

[5] Budinsky F J,Finnie M A,Vlissides J M,et al.Automatic code generation from design patterns[J].IBM Systems Journal,1996,35(2):151-171.

[6] Oldevik J,Solberg A,Elvesaeter B,et al.Framework for model transformation and code generation[C]//the Sixth International Enterprise Distributed Object Computing Conference(EDOC'02),2002:181-189.

[7] Fischer G,Lusiardi J,von Gudenberg J W.Abstract syntax trees-and their role in model driven software development[C]//International Conference on Software Engineering Advances(ECSEA'07),2007:38-38.

[8] Herve Tchepannou.xpetstore 3.1.3[CP].[2003-08-07].http://sourceforge.net/projects/xpetstore.

(上接 6 页)

息数为 $d+1+\beta/N$。这个结果比 E-G 要小。

KPSBM 中,关于 Blom 矩阵的计算开销较大,但是这样的计算只在初始时运行一次,和 Wenliang Du[6]方案相比,只有少数具有外簇邻居节点的节点才需要运行计算 Blom 矩阵, 从总体上来看,节点的计算开销远小于 Wenliang Du 方案。

6 结论

本文提出的无线传感器网络密钥管理方案 KPSBM 和现有的密钥管理方案相比,安全性更高,连通性和可扩展性更好,存储和通信开销小,能够较好地应用于传感器网络中。

参考文献:

[1] Akyildiz Ian F,Su Weilian,Sankarasubramaniam Yogesh,et al.A survey on sensor networks[J].IEEE Communications Magazine,2002,40(8):102-114.

[2] 苏忠,林闯,封富君,等.无线传感器网络密钥管理的方案和协议[J].软件学报,2007,18(5):1218-1231.

[3] Eschenauer L,Gligor V.A key management scheme for distributed sensor networks[C]//Proc of the 9th ACM Conf on Computer and Communications Security,Washington,DC,2002:41-47.

[4] Zhu S,Setia S.LEAP:efficient security mechanisms for large-scale distributed sensor networks[C]//Proc of the 10th ACM Conf on Computer and Communications Security,Washington,DC,2003:62-72.

[5] Perrigy A,Szewczyk R.SPINS:security protocols for sensor networks[J].Wireless Networks,2002,8(5):521-534.

[6] Du Wengliang,Deng Jing.A pairwise key predistribution scheme for wireless sensor networks[C]//Proc of the 10th ACM Conf on Computer and Communications Security(CCS),Washington,DC,2003:42-51.

[7] Wander A S,Gura N,Eberle H,et al.Energy analysis of public-key cryptography for wireless sensor networks[M].New York:Pervasive Computing and Communications,2005:324-328.

[8] Yang Geng,Rong Chunming,Veigner C,et al.Identity-based key agreement and encryption for wireless sensor networks[J].Journal of China Universities of Posts and Telecommunications,2006,13(4):54-60.

[9] Zhang Yanchao,Liu Wei,Lou Wenjing,et al.Location-based compromise-tolerant security mechanisms for wireless sensor networks[J].IEEE Journal on Selected Areas in Commnuications,2006,24(2):247-260.

[10] Blom R.An optimal class of symmetric key generation systems[C]//Thomas Beth,Norbert Cot,Ingemar Ingemarsson.Lecture Notes in Computer Science:Advances in Cryptology,Proceedings of EUROCRYPT 84.Berlin,Heidelberg:Springer-Verlag,1985:335-338.

[11] Du Wenliang,Deng Jing,Han Y S,et al.A key management scheme for wireless sensor networks using deployment knowledge[C]//INFOCOM 2004,HongKong,2004,1:586-597.