EJB3.0新规范概览及其未来发展 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/271/2021_2022_EJB30_E6_96_B0_E8_c104_271648.htm 引言 期待已久的EJB3.0规范在最近发布了它的初稿。在本文中将对新的规范进行一个概要性的介绍，包括新增的元数据支持，EJBQL的修改，实体Bean模型访问bean上下文的新方法和运行时环境等等。作者还讨论了EJB在未来要作出的调整以及EJB3.0与其他开发规范之间的关系。 开始 无论如何由于EJB的复杂性使之在J2EE架构中的表现一直不是很好。EJB大概是J2EE架构中唯一一个没有兑现其能够简单开发并提高生产力的组建。 EJB3.0规范正尝试在这方面作出努力以减轻其开发的复杂性。EJB3.0减轻了开发人员进行底层开发的工作量，它取消或最小化了很多（以前这些是必须实现）回调方法的实现，并且降低了实体Bean及O/R映射模型的复杂性。 在本文中，我首先会介绍EJB3.0中几个主要的改变。它对进一步深入了解EJB3.0是非常重要的。随后，我会从更高的层面来描述已经被提交到EJB3.0规范中的细节，并一个个的讲解新的规范中的改变：实体 Bean,O/R映射模型，实体关系模型和EJB QL(EJB查询语言)等等。 背景 EJB3.0中两个重要的变更分别是：使用了Java5中的程序注释工具和基于Hibernate的O/R映射模型。 Java5中的元数据工具。 Java5 （以前叫J2SE1.5或Tiger）中加入了一种新的程序注释工具。通过这个工具你可以自定义注释标记，通过这些自定义标记来注释字段、方法、类等等。这些注释并不会影响程序的语义，但是可以通过工具（编译时或运行时）来解释这些标记并产生附加的内容（比如部署描述文件），或者强制

某些必须的运行时行为（比如EJB组件的状态特性）。注释的解析可以通过源文件的解析（比如编译器或这IDE工具）或者使用Java5中的APIs反射机制。注释只能被定义在源代码层。由于所有被提交到EJB3.0草案中的注释标记都有一个运行时的RetentionPolicy，因此会增加类文件占用的存储空间，但这却给容器制造商和工具制造商带来了方便。 Hibernate 目前Hibernate非常受欢迎，它是开发源代码的Java O/R映射框架，目的是把开发人员从繁琐的数据持久化编程中解脱出来。它也有一个标准的HQL（Hibernate 查询语言）语言，你可以在新的EJB QL中看到它的影子。Hibernate在处理如数据查询、更新、连接池、事务处理、实体关系处理等方面非常简单。 概览 在已经提交的EJB3.0规范中主要涉及两个方面的改变：1. 一套以注释为基础的EJB编程模型，再加上EJB2.1中定义的通过部署描述符和几个接口定义的应用程序行为。 2. 新的实体Bean持久化模型，EJBQL也有许多重要的改变。 还有一些有关上述的提议，比如：一个新的客户端编程模型，业务接口的使用以及实体Bean的生命周期。请注意EJB2.1编程模型（包括部署描述符和home/remote接口）仍然是有效的。新的简化模型并没有完全取代EJB2.1模型。 EJB注释 EJB 规范组织一个重要的目标是减轻原始代码的数量，并且他们为此给出了一个完美而简介的办法。在EJB3.0的里，任何类型的企业级Bean只是一个加了适当注释的简单Java对象(POJO)。注释可以用于定义bean的业务接口、O/R映射信息、资源引用信息，效果与在EJB2.1中定义部署描述符和接口是一样的。在EJB3.0中部署描述符不再是必须的了；home接口也没有了，你也不必实现业务接口（容器可以为你完成这些事情）。

比如，你可以使用@Stateless注释标记类把Java类声明为一个无状态回话bean。对于有状态回话bean来说，@Remove注释可以用来标记一个特定的方法，通过这个注释来说明在调用这个方法之后bean的实例将被清除掉。 为了减少描述组件的说明信息，规范组织还采纳了由异常进行配置（configuration-by-exception）的手段，意思是你可以为所有的注释提供一个明确的缺省值，这样多数常规信息就可以据此推断得出。 新的持久化模型 新的实体bean也是一个加了注释的简单Java对象(POJO)。一旦它被EntityManager访问它就成为了一个持久化对象，并且成为了持久化上下文（context）的一部分。一个持久化上下文与一个事务上下文是松耦合的；严格的讲，它隐含的与一个事务会话共存。 实体关系也是通过注释来定义的，O/R映射也是，并提供几种不同的数据库规范操作，在EJB2.1中这些要通过开发人员自己的设计模式或者其它技术来完成的（比如，自增长主键策略）。 深入研究 现在是时候详细了解EJB3.0草案了。让我们开始探讨所有EJB中四种企业级bean，并看看他们在新的规范中是什么样子。 无状态回话bean 在EJB3.0规范中，写一个无状态回话bean(SLSB)只需要一个简单的Java文件并在类层加上@Stateless注释就可以了。这个bean可以扩展javax.ejb.SessionBean接口，但这些不是必须的。 一个SLSB不再需要home接口，没有哪类EJB再需要它了。Bean类可以实现业务接口也可以不实现它。如果没有实现任何业务接口，业务接口会由任意 public的方法产生。如果只有几个业务方法会被暴露在业务接口中，这些方法可以使用@BusinessMethod注释。缺省情况下所有产生的接口都是local（本地）接口，

你也可以使用@Remote注释来声明这个接口为remote（远程）接口。 下面的几行代码就可以定义一个HelloWorldbean了。而在EJB2.1中同样的bean至少需要两个接口，一个实现类和几个空的实现方法，再加上部署描述符。import javax.ejb.*;/*** A stateless session bean requesting that a remote business* interface be generated for it.*/@Stateless@Remotepublic class HelloWorldBean {public String sayHello() {return "Hello World!!!".}} 有状态回话bean 除了几个SFSB的特别说明之外，有状态回话bean(SFSB)和SLSB一样精简： 一个SFSB应该有一个方法来初始化自己（在EJB2.1中是通过ejbCreate()来实现的）。在EJB3.0的规范中建议这些初始化操作可以通过自定义方法完成，并把他们暴露在业务接口中。在使用这个bean之前由客户端来调用相应的初始化方法。目前规范组织就是否提供一个注释来标记某个方法用于初始化还存在争议。 Bean的提供者可以用@Remove注释来标记任何SFSB的方法，以说明这个方法被调用之后bean的实例将被移除。同样，规范组织仍然在讨论是否要有一种机制来处理这种特殊的情况，即当这个方法出现异常的情况下bean的实例是否被移除。 下面是对以上问题我个人的观点： 是否应该有一个注释来标明一个方法进行初始化呢？我的观点是应该有，这样容器就可以在调用其他方法之前至少调用一个方法来进行初始化。这不仅可以避免不必要的错误(由于没有调用初始化方法)而且可以使容器更明确的判断是否可以重用SFSB实例。我暂且把这个问题放一放，规范组织只考虑为一个方法提供一个注释来声明它是一个初始化方法。 对于第二个问题我的观点也是肯定的。这有利于Bean的提供者合客户端程序对其进行控制。只

有一个遗留的问题：那就是一旦调用这个方法失败，是否能移除这个bean 的实例？答案是不能，但是它将会在回话结束的时候被移除。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com