Java进阶：优化EntityBea 的七条守则 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/461/2021_2022_Java_E8_BF_9B_E9_98_B6_c104_461602.htm Entity beans提供了一个清楚的模型以描述应用程序中持久稳固的事务对象以及它们的设计。在面向对象的模型中，简单的Java对象通常被直接地描述，但是并不包括事务处理的功能，这通常需要使用事务对象来完成。Entity beans不仅仅考虑到模型中的简单类型，它也考虑到面向对象模型中的事务模型，它将所有的复杂的工作都交给了bean和容器服务。这使得应用程序可以象使用通常的Java对象那样来应用它。在保持存储数据的公开性和灵活性的同时，容器会对entity beans进行优化，而我们只需要考虑如何部署它。 基于Enterprise JavaBeans的开发方案，导致了面向对象的方法的广泛使用和对entity beans的大量使用。Sun的工程师对entity beans的实际使用作了大量的实践和研究。这篇文章是对开发中的一些经验的总结：探索各种优化方案对达到最佳的性能和适应性提供规则和建议讨论如何避免一些已知的问题一、尽可能地使用容器持久化管理容器持久化管理(CMP)不仅仅可以大量减少编程的工作，而且在容器和容器生成的数据库访问代码中存在许多优化的可能。容器可以访问内存中的bean的缓冲，这使得它可以监控缓冲中的任何改变。如果缓冲没有被改变，这可以在提交事务前避免将缓冲存储到数据库中。这减少了不必要的调用数据库的开销。另外一个优化的实例是对find方法的调用。根据对一个entity的参考搜寻一个entity，而接下去最可能的情况是使用这个entity。这通常包括两个数据库访问：在数据库中寻找记录

并得到主键将记录的数据读入到内存中CMP会进行这样的优化，只要它认为可以这样做，它会使用一个数据库访问来代替这两个数据库访问，它会在一个数据库访问中同时得到主键和记录的数据。二、编写同时支持Bean持久化管理和容器持久化管理的代码在许多情况下，EJB的作者无法控制EJB将如何被部署，也无法预知部署使用的容器是否支持CMP。同样的，部署者在目标容器可能会使用bean持久化管理(BMP)。你必须寻找一个方法以执行这个bean以允许BMP部署而不用破坏对更多的CMP机制优化的破坏。一个简单的方法是将商业逻辑和持久化机制分离开来。在你的CMP类中执行商业逻辑，如果选择了CMP，它可以被单独部署。实现持久化的代码被写入到BMP类中，它继承自CMP类。在CMP的超类中保持了所有的商业逻辑，并在BMP子类中增加了数据库访问的代码。这个模型听起来很容易实现，但是事实上存在一些问题：对执行类的继承是不可能的。这样做意味着子类必须同时继承CMP超类和BMP超类。此外，BMP子类还必须继承它的直接的CMP执行。这导致了多重的类的继承，而这样的继承是不被Java程序所允许的。 没有简单的方法支持改变的持久化执行这可能是很有用的，例如，可能要执行不同的数据库提供商或是不同类型的数据库(例如关系型、对象型或其它类型)所特定的数据库定义代码。要解决这些问题，你可以对目前的模型作一些改变。对所有BMP类抽象出一个辅助类以执行基本的持久化代码。这样的辅助类被称之为数据访问对象(DAO)。你可以通过对DAO接口的实现产生多个DAO子类，然后在实际应用时选择合适的DAO子类进行实例化。有许多方法可以选择合适的DAO子类进行实例化，例如，通过读

取环境变量或是判断DB类型。使用这个模型，CMP执行和包含这个执行的DAO可以很方便地被继承；除了BMP类外，所有的东西都可以被执行。因为这些BMP类包含了一些具有代表性的代码的框架，对于第一个entity bean它看上去都是差不多的，你可以很容易地将一个bean拷贝到另一个而只需作很小的改动。而最终，你甚至可以使用工具来自动完成这项工作。尽管你可能需要扩充一个entity bean以重新使用另一个entity bean所提供的逻辑，但是在EJB 1.1的说明书中不允许对一个继承自home接口的entity类型进行扩充。对一个调用的定位及建立在理论上总是被认为是一种远程调用。虽然在很多时候你可能需要使用这种子类型（例如，将它们作为一个factory方法），但是不幸地是，这样的子类型并没有被提供。这阻碍了对许多遵守EJB 1.1说明书的执行EJB的有用的设计平台的使用。 三、在ejbStores中尽量减少对数据库的访问 在使用CMP时，bean完全不受ejbStore的控制，所有的优化工件都是由容器来完成的。容器所提供的CMP优化成了区别容器性能的主要指标之一。然而在使用BMP来部署bean时，为缓冲区维持一个dirty标志是一种很有效的做法。对缓冲区的所有改变都会对dirty标志进行设置，ejbStore将会对这个标志进行检查。如果dirty标志未被设置，这意味着缓冲区没有被改变，ejbStore将不执行所有的开销巨大的对数据库的访问。这个决窍对于那些经常对数据库进行查询而不是作真正的改变的bean尤为有效。而这通常组成了一个应用程序中的绝大部分(例如，查找一个表)。对于这种技术，有一点需要注意。因为dirty是为了数据库的访问而设立的，因此它对于BMP的代码会更适合，也就是说，它并不太适合在CMP代码中使

用。当使用我们前面介绍的执行平台时，它既可以在BMP代码中进行设置也可以在DAO中进行设置。 因为DAO不会调用任何的商业逻辑，因此它不是设置dirty标志的最合适的地方，最好你还是在BMP代码中对这个标志进行设置。这可能会使得BMP代码变得更加复杂，因为你不能在商业逻辑中考虑对超类的授权。完全从一种科学的观点来看，EJB的作者并不需要处理系统级的问题，包括对缓冲区的处理。但是不幸的是，使用BMP的EJB 1.1并不提供这种系统优化的选择，因此bean的作者还是不得不手工对dirty标志进行设置。 四、对于调用的查找和定位的总是使用Reference Cache无论是对于entity beans还是session beans，Reference Cache都是一项有用的技术。JNDI对例如数据资源、参考bean或者是环境实体这样的EJB资源的搜寻是相当耗费资源的，而要防止这种多余的搜寻，方法并不太复杂。要解决这个问题，通常的做法是：将这些references定义为一个实例变量。 