参考：<http://blog.csdn.net/alburt2008/article/details/46746705>

EJB也许是**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**领域里中最受争议的技术了。有人说EJB是最伟大的发明，也有人说EJB完全是多此一举；当一些人陶醉于EJB的深奥理论时，另外一些人却正被EJB的繁琐复杂所折磨；尝到EJB甜头的人，在EJB的每个新版本中，都能发现盼望已久的惊喜，而被EJB拒之门外的人，则随着EJB的升级，愈发对EJB敬而远之了。

EJB的全称是Enterprise JavaBeans，JavaBeans很普通，不过Enterprise就不那么简单了。什么技术，一旦被冠以Enterprise的名头，就像男人走入婚姻殿堂一样，身上的责任与单身汉不可同日而语了。从定义上看，JavaBeans只是J2SE平台上的一个组件**[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "_blank)**，包含一些业务逻辑，并且可以被重用。EJB不同，作为企业级的JavaBeans，Sun对EJB的定位要远远高于JavaBeans，所以EJB的目标也比JavaBeans要远大得多，除了作为一个包含业务逻辑的可重用组件外，EJB更被赋予了诸如“可移植”、“安全”、“可伸缩”、“交易性”等特征。所有这些EJB必须具备的特征，其实正是企业应用所要求的。这也是Enterprise一词所代表的技术上的含义。企业应用不同于普通应用，企业应用是大规模的、高复杂度的和关键的，它所面临的挑战，要比普通应用艰巨得多。比如，企业应用对可移植性的要求非常高，这是因为，企业都不愿意将自己的未来绑定到某个供应商的身上，除非是不得已而为之；又比如，安全性对企业应用至关重要，谁能使用什么功能、哪些数据哪些人可以看到，都有严格的限制；更不用说的是企业应用的可伸缩性了，当业务规模变大时，你希望全盘推翻旧系统，采购一批崭新的软件和硬件，对IT系统来个彻底的革命吗？增加一台服务器就能应付更多的客户，我想这是头脑正常的企业家都希望的。企业应用的需求，就是EJB的目标。用EJB开发的应用，完全符合企业应用的特征。

EJB是一个规范，只要符合这个规范，EJB可以在不同的**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "_blank)**、不同的应用服务器中无缝地移植；EJB允许开发者在EJB部署描述文件中进行方法级的、基于角色的安全性配置，以统一的方式保护企业应用和数据的安全性；只要你愿意，EJB应用可以全部部署在一台单独的服务器上，也可以任何组合方式分布在一组服务器群中，满足你扩大规模和均衡负载的要求；如果你想保持事务的完整性，那么，EJB的事务管理是一个可靠的、稳健的解决方案。这就是EJB，一个企业应用的集大成者，多种技术的浓缩精华，全能的框架和基础结构。

可就是这样一个将企业应用的开发简化到了前所未有之程度的技术，却成为许多人口诛笔伐的对象。复杂、难以使用、性能低下、繁琐等等，从1998年EJB诞生之日起，各种各样的恶名就伴随左右，直到八年后的今天，当EJB迎来它的第三次大变脸时，质疑之声依然不绝于耳。EJB真的那么糟糕吗？

EJB是企业应用的先驱笔者，接触第一个企业应用，是在1997年。那时PowerBuilder风头正劲，不过，多数人使用PowerBuilder，是因为它的数据窗口。当时笔者在一个项目中遇到一个难题，那就是如何把一台服务器上的应用一分为二，跑在两台服务器上，以提高性能。这是典型的分布式应用，虽然不是一个完整意义上的企业应用，不过，因为应用中需要用到分布式的概念，多少也算和企业应用沾上边了。PowerBuilder其实是个非常不错的开发工具，在1997年的时候，已经提出了分布式应用的概念，并且付诸实施了。在PowerBuilder中，一个组件可以有一个称为代理的对象，这个对象可以运行在与组件不同的机器上，其他组件通过代理可以访问该组件的功能。这是一个很初级的分布式应用框架，不过，那时已经给了笔者很大的震动。我试着编了一个实验性质的程序，当我在一台机器上按下一个按钮时，另外一台机器上赫然弹出一个预期中的对话框，着实让我大吃一惊。没有任何Socket编程，也不需要关心实际的应用跑在哪台机器上，PowerBuilder让我首次见识了分布式应用框架的巨大威力。PowerBuilder解决了分布的问题，但安全性和事务控制，仍然需要程序员自己想办法。十个程序员可以有十种解决方案，每种都不同，而每种都可能含有未经发现的缺陷。在EJB之前，企业应用的开发没有规范可循，每个公司都有自己的一套方案，尽管每个公司都对自己的方案充满信心，但其实这些未经大量应用考验的方案，都有着这样那样的不足或局限。

J2EE是第一个为业界所广为接受的完整的企业应用框架，而EJB在其中扮演重要角色。在J2EE框架的支持下，运行在EJB容器中的EJB，完全符合企业应用关于分布、移植、安全和交易的要求。这对于企业应用的开发者来说，意义非同寻常。首先，现在大家可以在一个公共的平台技术上构建自己的企业应用，不必绞尽脑汁“发明”自己的“轮子”，从而节省大量无谓的、重复性的技术和时间投入；其次，一个公开的平台，让大量的企业应用开发者有了共同语言，可以相互交流平台的使用经验和教训，这样，随着平台之上企业应用的不断增加，平台的优劣得失一览无遗，有利于平台的改进和发展。这就是EJB为企业应用作出的贡献。在EJB之前，多数人不知企业应用为何物，或者虽然有企业应用的模糊概念，但要编写一个企业应用，谈何容易。不同的操作系统、不同的开发语言、不同的网络环境、不同的应用终端，开发一个企业应用，程序员面临着两难的抉择：要么限定应用的软硬件平台，或者牺牲应用的安全性、分布性或交易性，开发一个“伪”企业应用；要么下决心开发一个真正的企业应用，然后累死自己。

EJB是一种思想EJB让程序员编写真正意义上的企业应用而不必累死，应该说，EJB已经是企业应用开发领域的一大进步，让企业应用和普通应用的开发差距缩短到了前所未有的程度，可是，为什么还有很多人抱怨EJB过于复杂呢？EJB的复杂性其实是个伪命题。所谓复杂，一定是相对的。如果和普通应用相比，EJB当然要复杂很多，因为人们对于普通应用没有企业应用那么苛刻的要求。但是，如果将EJB之前和EJB之后的企业应用开发的难度相比，相信人们就会对EJB赞誉有加了。举个不准确的例子，假如EJB之前，企业应用的难度是普通应用的10倍，那么，EJB之后，这个比例缩小到了5倍，批评EJB复杂者，只看到了还剩下的5倍复杂度，却没有看到EJB所减去的5倍复杂度。关于EJB过于复杂的论断，还来自于与其竞争技术的比较。例如，**[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)**就是一个声称比EJB更简单的、轻量级的企业应用框架。Spring确实简单，很多人喜欢Spring，也正是因为Spring简单。可是，Spring的支持者们，忽略了一个基本的事实，那就是Spring的功能要比EJB弱，也就是说，Spring是通过放弃某些不常用的功能来达到简化目的的。Spring好比一辆没有安全气囊的车，尽管依然可以拉客跑运输，但一旦发生碰撞事故，也许司机会陪上性命。没有安全气囊的Spring，在制造上当然要比有安全气囊的EJB简单，而且轻便，跑得快，不过，Spring始终不适合投入正式营运。这就是为什么不推荐在大型企业应用上采用Spring的原因。没有完善的事务处理，不能提供7X24小时的服务，Spring迈不过关键企业应用的门槛。与Spring形影不离的是Java对象持久化的“红人”[**hibernate**](http://lib.csdn.net/base/javaee)。Hibernate的矛头直指EJB的Entity Bean。Entity Bean，尤其是它的持久化技术，是最为程序员所诟病的，成为EJB挥之不去的阴影，并最终促成了Hibernate的辉煌。Hibernate其实并不精深，在技术上也没有太多值得称道的创新，但它的文档非常优秀。我知道很多程序员就是被Hibernate的文档所吸引的，他们只学过一些SQL初步，没有系统的关系**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**理论知识，Hibernate关于数据库表间关系的论述，深入浅出，十分精彩，让他们在对关系数据库的理解上有了迅猛突破的同时，Hibernate轻易的俘虏了他们的心。Hibernate的成功，反衬了EJB在持久化方面的失败，但在我看来，这并不影响EJB的伟大。

与其说EJB是一种技术，不如说EJB的是一种思想更恰当，而不论Hibernate还是Spring，只不过是一种工具，他们只是跟在EJB后面，发现了EJB的某些不足，然后有针对性地加以改进，以迎合普通程序员对于“技术快餐”的需求。他们既没有从形形色色的企业应用中，抽象出隐藏在不同表现后面的本质特征，也没有创造性地用Stateless Session Bean和Stateful Session Bean来描述千变万化的现实世界。工具只是工具，不出两年就会有新的后起之秀，取而代之，但思想的光辉将长久地照亮技术的未来。EJB是一种思想，更是一种理想，尽管理想和现实总是存在差距，但这不能成为我们放弃EJB的理由。一种满足企业应用分布性、扩展性、安全性和交易性要求的、方便使用的框架技术，既是EJB的理想，也是广大程序员的理想。你的观点又是如何?