参考：<http://blog.csdn.net/li_xiao_ming/article/details/52793879>

# [EJB的优势和使用场景](http://blog.csdn.net/li_xiao_ming/article/details/52793879)

 EJB的优势和使用场景

即使在EJB 2.0备受诟病的时期，笔者也从不掩饰自己对EJB的喜爱，因为它确实体现了一种非常优秀的设计思想和理念。即使在EJB饱受争议的时期，笔者也一直希望大家用更理智的眼光来看一种技术。我们可以尽量从以下两方面来看待一种技术：

这种技术的设置初衷是什么？

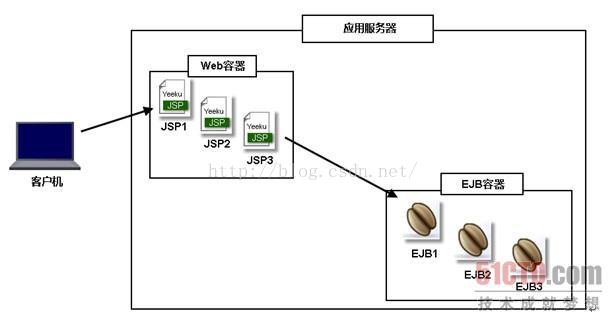
这种技术到底给我们带来了什么？

从某种意义上来看，EJB是一种大型分布式企业应用开发[**架构**](http://lib.csdn.net/base/architecture)的先驱尝试者，它试图解决这种企业应用底层那些系统级的问题，系统提供一种可重用的、通用的解决方案。

回顾EJB出现以前的[**Java**](http://lib.csdn.net/base/java)应用开发，大部分开发者直接用JSP页面，再加上少量**Java** Bean就可以完成整个应用，所有的业务逻辑、[**数据库**](http://lib.csdn.net/base/mysql)访问逻辑都直接写在JSP页面中。系统开发前期，开发者不会意识到有什么问题，但随着开发进行到后期，应用越来越大，开发者需要花费大量时间去解决非常常见的系统级问题，反而无暇顾及真正需要解决的业务逻辑问题。

对于EJB来说，它提供了一种良好的组件封装，EJB容器负责处理如事务、访问控制等系统级问题，而EJB开发者则集中精力去实现业务逻辑；对页面开发者而言，EJB的存在无须关心，EJB的实现无须关心，他们只要调用EJB的方法即可。

图8.1显示了以EJB为核心的应用程序结构。



|  |
| --- |
| （点击查看大图）图8.1  以EJB为核心的应用程序结构 |

从图8.1中可以看出，在以EJB为核心的应用程序中，业务逻辑开发者的主要精力将集中在EJB组件的开发上，EJB组件是一种可移植的、与前端技术无关的服务器端组件。虽然图8.1的结构中绘制的是以JSP页面来调用EJB组件，但实际上前端可能还需要使用MVC框架。

但无论如何，基于EJB的程序架构总体有一个非常优秀的思想：业务逻辑相关的实现集中在EJB中完成，而EJB容器则负责提供带有重复性质的、系统级的功能，这样EJB组件就可对外提供完整的业务服务。

按照Sun公司的设计初衷： EJB容器应该是标准的，那么开发者写好的EJB组件就可以在任何EJB容器之间自由移植；而且按Sun的最初的EJB 1.0规范，EJB本身就是建立在RMI基础之上的，这样就允许客户端程序从远程来调用EJB内的方法。这就难怪Sun公司对EJB寄予厚望了，因为它确实是无数软件开发人的梦想：一个一个的EJB，只要将它们部署在EJB容器中，它们就会组成一个完备的业务层，具有很好的可移植性、很好的可扩展性。

笔者一直对Sun公司充满一种很复杂的感情：伟大的Sun公司，设计出EJB这种划时代的规范，却被无数开发者诟骂，直至沦落为今天被收购。

由于EJB 2拥有无比健壮的特性，而且还考虑到远程访问等大量特性，所以导致EJB 2的开发有点复杂。因此导致许多开发者的批评，以致后来直接催生了[**spring**](http://lib.csdn.net/base/javaee)框架。

在笔者看来，Spring框架实际上大量参考了EJB的设计理念（不知道Spring的开发者是否意识到这一点），只是Spring摈弃了EJB开发中的3大烦琐之处：

EJB组件的接口和类必须继承指定接口或类。

需要大量使用XML配置文件。

EJB组件必须打包成JAR包。

而Spring框架的整体设计，与EJB的设计可谓殊途同归，图8.2显示了以Spring为核心的应用程序结构。

将图8.1和图8.2两张结构图放在一起对比，发现它们之间的差异主要有两点：

Spring容器取代了原有的EJB容器，因此以Spring框架为核心的应用无须EJB容器支持，可以在Web容器中运行。

Spring容器管理的不再是复杂的EJB组件，而是POJO（Plain Old [**java**](http://lib.csdn.net/base/java)Object） Bean。

对于Spring的作者而言，他已经深深地吃透了EJB的设计理念，并遵循这种理念开发出了一个开源的Spring框架。换个角度来看，Spring容器又何尝不是另一个Bean容器，只是这个Bean容器并未遵循Sun公司的EJB容器规范。

|  |
| --- |
| http://img.blog.csdn.net/20161012083136071?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center |
| （点击查看大图）图8.2  以Spring为核心的应用程序结构 |

不管Spring的作者是否意识到，其实他对EJB的很多设计思想还是推崇的，甚至可以说他也是需要EJB、EJB容器这种结构的，只是他觉得开发EJB组件的步骤有点复杂、烦琐。而且对于有些中小型的应用来说，EJB的远程访问支持并不是必需的，而Spring的POJO Bean反而更加简单、易用。

对于应用开发者而言，Spring容器要求的Bean简单得多，这些Bean无须实现任何接口或继承任何基类，无须单独为每个Bean类使用XML配置文件，无须打包成JAR文件……这个世界简单了，于是开发者们笑了。

对于应用使用者而言，轻量级的Spring容器已经取代了昂贵的EJB容器，以Spring为核心的轻量级[**Java EE**](http://lib.csdn.net/base/javaee)应用不再需要应用服务器（如WebLogic、WebSphere等），只需要普通的Web服务器（如Resin、Tomcat等）即可，花费降下来了，所以应用使用者们也笑了。

通过上面介绍，读者可能会发现一个问题，不管是以Spring＋[**hibernate**](http://lib.csdn.net/base/javaee)为核心的轻量级[**java ee**](http://lib.csdn.net/base/javaee)应用，还是以EJB为核心的经典Java EE应用，它们的结构其实殊途同归。如果从花费上来看，轻量级Java EE应用往往更加具有吸引力，这就是我们看到许多中小型企业开发的应用都是以Spring＋Hibernate为核心的原因。

EJB 3的出现成为了EJB规范的巨大转机。就简单、易用性方面来说，EJB 3的开发并不比Spring容器中POJO Bean复杂多少（后面我们会看到）。而且真正商用的应用服务器提供了Spring容器更多的支持，例如EJB的池化管理、服务器节点的集群管理等。

由此可见，对于规模较小、伸缩性要求不大的企业级应用而言，使用以Spring＋Hibernate为核心的技术来开发即可。但对于具有如下3个特征的企业级应用来说，选择以EJB为核心的经典Java EE技术可能更合适。

应用的规模较大，而且增长速度快速。

应用的伸缩性要求很高。

应用可能需要使用除JSP页面之外的其他客户端。

文章来源 http://book.51cto.com/art/201008/220993.htm