J2EE应用中常见的反模式(anti-patterns) PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022_J2EE_E5_BA_94_E7_94_A8_c104_144738.htm J2EE应用中有一些常见的毛病和错误的观念，按照时下流行的说法，叫反模式。稍不注意，我们自己也会犯，所以大概整理一下，一个是备忘，也是供需要的朋友参考：1- 无EJB不叫J2EEEJB一直发展到今天的2.1仍然被广为诟病，它提供了很多时候我们并不需要的东西，而且我们在很多情况下一旦选用EJB就没有其他的方式不去使用那些笨重的功能。但是很多所谓范例让我们有一种错觉，好像不用EJB就不是J2EE应用。有一些折中的方案是使用Session Fa?ade模式，Entity Bean采用BMP 本地接口，然后提供一层无状态的Session Bean，采用远程和本地接口，这样的设计模式，我想，多半是出于无奈。如今，甚至我们经常都能看到不使用EJB的言论，炒得很火的Spring则为这种完全不用EJB开发J2EE项目提供了实际的、强有力的佐证。 2- 过度分层J2EE这个规范肤浅的来看，就是为我们定义了很多“层”，然后还有很多分工明确的“角色”，加上J2EE的蓝本应用程序就分了很多“层”，以至于大家都觉得J2EE的应用就应该是很多层的，其实不然，需要具体情况具体分析。 3- 频繁的往返调用EJB的看似简单造成我们经常忽略可能在使用过程中出现的远程调用，比如有时候为了更新一条记录，每个字段都是远程的去set，大大增加了不必要的开销，于是我们意识到在调用中使用DTO是一个建议遵循的方案。 4- 过度使用有状态的Session Bean一般来讲，一个Session Bean实例，如果它是有状态的，那么它只对某个固定的用户服务，如果是无

状态的，则可以满足不同用户的调用。这有点类似（只是有点类似）一个类的静态方法和非静态方法的区别。我们在实际应用中，应该尽量避免使用有状态的Session Bean，除非特别必要。我们可以把状态保留在Session Bean之外，如Web容器的session对象或者我们自定义的类中，而不是完全依赖有状态的Session Bean去帮我们做。 5- 过度会话Web容器的session对象是个好东西，用起来也很方便和直截了当，这造成了我们很多人对它的滥用，什么东西都往里面放。这有两个突出的问题，一个是资源浪费；另一个，万一Web服务器崩溃，那些本来需要持久化的数据就丢失了。我们需要考虑好，哪些数据本可以用request的，哪些数据又是需要持久化到数据库的，等等，不能一味依赖session。 6- 万能Servlet或者万能JSPJ2EE为我们提供了Web层丰富的技术选择，Servlet或者JSP都只是其中一种，虽然它很强大，但是也不应该由它一个来承担所有MVC三个部分的功能。现实中我们的Struts很好的规范了这个问题：Servlet负责调度，专门的Action负责处理逻辑，而JSP用于用户界面显示。JSP和Servlet本质上是同一个东西，只是从不同的角度来处理问题，它们各有所长，互为补充。 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