Java注释A otatio 能使J2EE简易化吗? PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/290/2021_2022_Java_E6_B3_A8_E9_87_8A_c104_290408.htm 随着J2EE进入5.0时代后，Java EE5.0的很多特性也被广泛应用在J2EE程序中。而Java EE5.0的注释（Annotations）特性就是其中应用最广泛的特性之一。如果稍微浏览一下最新的Java EE5.0（EJB3.0，JPA）的标准规范，就可以发现，这些规范的制定者或是支持者们宣称最多的莫过于，利用这些规范可使开发变得像开发POJOs一样的简单与简洁。但是，如果对那些源代码稍加浏览或查看，最引人注目的可能就是那些取代了XML描述作用的注释。那么，笔者们禁不住要问一句，注释的使用真的可以简化复杂的J2EE或组件的开发吗？ 一、引言 在以前的J2EE版本中，都是使用大量的配置文件来设置Web程序、EJB等。但这一切在Java EE5.0中得到了彻底的改善。Java EE5.0中的注释（Annotation）是专门针对Web和EJB程序而设计的。如@Resource、 @EJB和@WebServiceRef等，以及其它一些与安全相关的注释，如@RunAs和@DeclareRoles. 在Java EE5.0中，这种对元数据（Meta-data）的支持的数据就是注释。通过使用注释， 程序开发人员可以在不改变原有逻辑的情况下，在源文件嵌入一些补充的信息。代码分析工具、开发工具和部署工具可以通过这些补充信息进行验证或者进行部署。举个例子，例如希望某个方法的参数或者返回值不为空，虽然可以在Java doc中进行说明，但是表达同样意思的作法有很多，比如“ The return value should not be null ”或者“ null is not allowed here ”。测试工具很难根据这些语言来分析出程序员

所期望的前提条件（Pre-condition）和执行后的条件（Post-condition）。而使用注释（Annotation），这个问题就可以轻而易举的解决。 注释，简单的说，就是代码里的标记，这些标记可以在类加载、运行时或是编译的时候可以被解释。这给人的感觉极像C 的macros. 其实说起注解语法，对于任何一个Java开发人员来说都已经耳熟能详了，我们每天都在使用着 的@author、 @param等等，其实都是在编写注释，然后用Javadoc生成文档。Java的这种方便的文档生成方法受到了开发者的普遍赞誉。而从JDK1.5开始，注释语法提供了更为强大的功能。 二、注释真的能简化吗？ 笔者问过一些做Java开发的朋友关于对注释的了解情况。很多都说，在实际的项目当中，依然在使用Java1.4及EJB2.0，因为还没有到不得不使用注释的时候。当然，在平时的学习当中，可能会有涉及到注释。 注释是代码文件中的伪代码，而代码之外的一些配置文件，如XML及*.properties，感觉更加容易从编译过的部署类里具体化。这两者可能各有优点，那我们普通的开发人员依据什么来决定，到底是把元数据写在源文件代码里，还是写在单独的配置文件中呢？ 有一点可以肯定，其它能像Java这样提供注释功能的语言并不多。引入注释的目的无非就是想把一些需要单独或是额外解决的问题，引用于源文件中一并解决，但这并不一定能起到药到病除的效果。 让我们看看使用注释在类文件中写入配置信息的情况：这意味着需要重新编译才能反映配置信息的改变，配置不能在Java程序外面单独的加以操作与配置。同时，对于使用那些支持注释及非注释两种类型的框架，无疑会使项目的配置更加混乱，增加维护难度。 如果一个软件在试运行或是真正使用时，碰到

用户需求变化或者原来功能不能正常运行，如果一些基本的配置信息都写进Java源文件中，一般的作法是：需求反馈到该程序设计程序员，程序员修改代码，再进行测试，可能测试不通过，影响其他功能了，再测试，折腾很长时间，最后编译打包，交付客户。 如果使用XML和Java分离方式：水平较低的维修人员赶到现场，修改一下配置XML，无须编译Java代码，测试，马上解决问题。很明显哪个更快呢？ 所以，开发软件不能只顾自己开发时方便，还要考虑到运行维护时是否方便，软件不像冰箱，制作好交给用户，很坚固，很稳定，用户也不会提出什么修改意见，当然海尔的定制化冰箱有这个意思，但是这种水平不是一般厂商水平能够做出来的。当然，笔者并不反对或是拒绝任何可以提高软件开发效率及节约时间的方法或技术，但这有个前提，就是这种技术或方法的成本或是代价。有些人会说，将一些部署逻辑或是信息嵌入在代码中，这样可以减少文件的间接访问，增加代码的集中度。但是，在一个满是注释的源文件中，去提取特定的配置注释，这无疑会增加代码的分析时间。此外，笔者们又如何给那些没有源文件的类文件进行注释呢？最后，注释又为何要整一套自己的新语法？它本来就像Java语法，那有必要另起炉灶，再整一套自己的语法吗？ 当然，笔者也承认，注释有它肯定的一面，并且可以肯定，其初衷是好的。但笔者认为，在EJB3.0规范中，其注释的规范有些太过极端了。因为很明显，这已经违背了软件的简易性原则，增加了开发的复杂性，使代码的可维护性降低了。 三、注释使J2EE开发变得容易一些 我们知道，注释其实也是一种修饰符，在可以使用其它修饰符（比如public，static，或 final）的任何地方都可以

使用注释。按规定，注释优先于其它修饰符。它包括一个@符号，@后接注释类型和被括号括起来的元素值对。这些值必须是编译时常量。也就是说 Java本身就提高了注释信息的详细列表。注释并不直接影响程序的语义，但它影响工具和库处理程序的方式，从而对运行程序的语义产生影响。注释可从源文件、class文件中阅读，也会在运行时得到反映。将定义从执行中分离出来并提供一种可以内省约束的方式，这使得执行过程更加灵活。大多数Java开发者已经很熟悉注释了，比如所有的JavaDoc标签和瞬时标签都是注释的例子。