 说到系统优化，是一个比较复杂的问题，涉及到软件的各个方面：需求、模块划分、数据库设计、程序编码以及一些特殊的优化方法如缓存技术等。而不同的应用又有其特殊的优化策略和技术。同时优化是贯穿系统从需求到实现再到维护的各个阶段的一项活动，而在各个阶段又有其不同的着眼点和具体方法。本文立足于具体的J2EE项目实践，结合一些已有的优化条例，提出自己的一些体会，也算是作为一次对实际项目经验教训的总结。优化一般意义上说是提高已有系统的性能，减少如内存、数据库、网络带宽等资源的占用，是在系统开发告一段落的前提下进行。一般是通过压力测试或具体使用发现性能方面的问题，然后寻找性能瓶颈，并结合项目进度、人员安排、技术储备等因素，提出相应的优化策略。下面结合一些案例，进行具体的讨论，并希望能总结出一些具有代表性的条例：条例一：尽量重用对象，避免创建过多短时对象对象在面向对象编程中随处可见，甚至可以毫不夸张的说是：“ 一切都是对象 ”。如何更好的创建和使用对象，是优化中要考虑的一个重要方面。笔者将对象按使用分为两大类：独享对象和共享对象。独享对象指由某个线程单独拥有并维护其生命周期的对象，一般是通过new 创建的对象，线程结束且无其它对这个对象的引用，这个对象将由垃圾收集机制自动GC。共享对象指由多个线程共享的对象，各线程保持多个指向同一个对象的引用，任何对这个对象的修改都会在其它引用上得到体

现，共享对象一般通过Factory工厂的getInstace()方法创建，单例模式就是创建共享对象的标准实现。独享对象由于无其它指向同一对象的引用，不用担心其它引用对对象属性的修改，在多线程环境里，也就不需要对其可能修改属性的方法加以同步，减少了出错的隐患和复杂性，但由于需要为每个线程都创建对象，增加了对内存的需求和JVM GC的负担。共享对象则需要进行适当的同步（避免较大的同步块，同时防止死锁）。还有几种特殊对象：不变对象和方法对象。不变对象指对象对外不含有修改对象属性的方法（如set方法），外部要修改属性只能通过new新的实例来实现。不变对象最大的好处就是无需担心属性被修改，避免了潜在的bug，并能无需任何额外工作（如同步）就很好的工作在多线程环境下。如jdk的String对象就是典型的不变对象。方法对象简单的说就是仅包含方法，不含有属性的对象。由于没有对象属性，方法中无需进行修改属性的操作，也就能采用static方法或单例模式，避免每次使用都要new对象，减少对象的使用。那么该如何确定创建何种对象，这就要结合对象的使用方式和生命周期、对象大小、构建花销等方面来综合考虑。如果对象生命周期较长，会存在修改操作，不能容忍其它线程对其的修改，就应该采用独享对象，如常见的Bean类。而如果对象生命周期较长，且能为各个线程共享，就可以考虑共享对象。共享有2种常见情况，一种是系统全局对象，如配置属性等，各个线程应该引用同一对象，任何对这个对象的修改都会影响其它线程；另一种是由于对象创建开销较大，各线程对此对象是瞬时访问，且无需再次读取其属性，如常见的Date 对象，一般这种对象的使用是瞬时的，比如把它format成String

，如果每次创建然后等待GC就会浪费大量内存和CPU时间，较好做法就是做成共享对象，各个线程先set再使用，注意对进行set并访问的方法要同步。不变对象一般使用在对象创建开销较小（属性较少，类层次较少），且需要能自由共享的情形。如一个对象里的常量对象，使用public static final AAA=new AAA(…) 创建。方法对象使用较广，如Util类、DAO类等，这些对象提供操作其它对象(一般是bean对象)的接口，能对系统在层次和功能上进行解耦合。 