现在越来越多人使用了。面对LoadRunner高昂的价格，免费的jmeter真的是让我们很是感动啊。眼泪哗哗地。

于是乎现在关于jmeter应用的各种入门教程啊，手册翻译啊，技巧集锦什么的，也越来越多了。不过大家可能感受最想要而又要不到的是资源监控，比如windows、linux的cpu，memory等。其实有些高手已经自己开发了jmeter的plugin用于监控了，但是似乎目前还没有人share出来，这些高手呢，相信不会是本文的主要读者了。  
日前接受了任务，希望能用jmeter来做一些性能测试。少不得就要面对jmeter缺乏资源监控的问题：有个监视器，但只能利用tomcat的status页监视jvm的内存和线程情况，功能单薄得就像霓裳仙子的那件衣服，马上就要露点了。  
看了一下jmeter监视器的实现思路，是解析tomcat的manage上status的xml文件来获得数据，再展现到jmeter的面板上。于是想到，如果通过修改监视器代码来解析和展现自己想要的内容，再实现动态的更新服务器上的xml文件——获取到资源数据写进去，这样jmeter不就可以做到监控各种服务器了么？  
好，那么就先下手分析jmeter的监控器代码。查了jmeter的开发手册，监视器代码没有跟其它组件一起，而是单独的monitor代码。源码分析简述如下：

监控器代码srcmonitorcomponentsorgapachejmeter  
文件：  
monitorutil  
       MemoryBenchmark.java  
       Stats.java  
visualizers  
       MonitorAclearcase/" target="\_blank" >ccumModel.java  
       MonitorGraph.java  
       MonitorGuiListener.java  
       MonitorHealthPanel.java  
       MonitorHealthVisualizer.java  
       MonitorListener.java  
       MonitorModel.java  
       MonitorPerformancePanel.java  
       MonitorStats.java  
       MonitorTabPane.java  
       ServerPanel.java

Stats.java：计算解析得到的参数  
public static int calculateLoad(Status stat)  
public static int calculateStatus(Status stat)  
public static int calculateMemoryLoad(Status stat)  
public static int calculateThreadLoad(Status stat)

MonitorPerformancePanel.java：监视器面板上的信息

MonitorStats.java：监视器输入数据  
       private static final String HEALTH = "stats.health";  
       private static final String LOAD = "stats.load";  
       private static final String CPULOAD = "stats.cpuload";  
       private static final String MEMLOAD = "stats.memload";  
       private static final String THREADLOAD = "stats.threadload";  
斜体字部分是jmeter弃置不用的了，原因也写在MonitorGraph.java的注释里：”TODO is this needed? It's never read“老外就是tmd能扯啊～

MonitorModel.java：生成数据模型，继承自MonitorStats

MonitorGraph.java：图像绘制  
private void drawSample(int x， MonitorModel model， Graphics g， MonitorModel last)

MemoryBenchmark.java：数据基准  
                            MonitorStats mstats = new MonitorStats(Stats.calculateStatus(st)， Stats.calculateLoad(st)， Stats.calculateCpuLoad(st)， Stats  
                                          .calculateMemoryLoad(st)， Stats.calculateThreadLoad(st)， "localhost"， "8080"， "http"， System  
                                          .currentTimeMillis());  
                            MonitorModel monmodel = new MonitorModel(mstats);  
                            objs.add(monmodel);

MonitorAccumModel.java：监视器数据操作相关