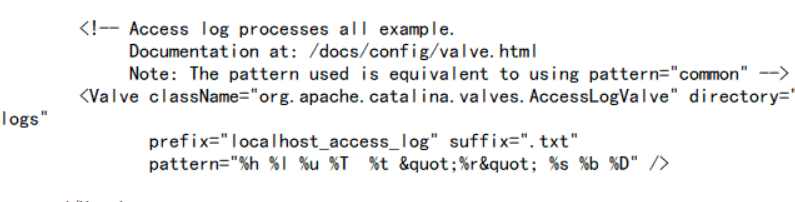
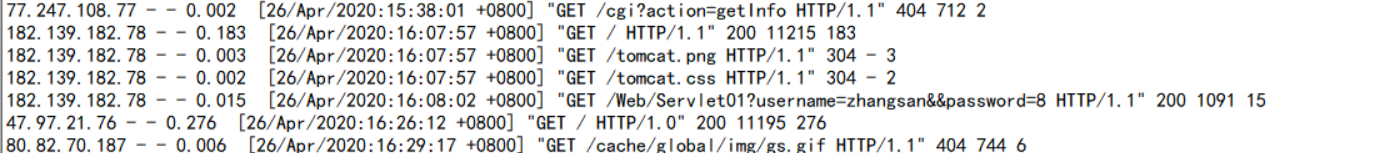
**1：配置   tomcat  ,  要求打开访问日志 并在日志中输出 每个请求处理时长**

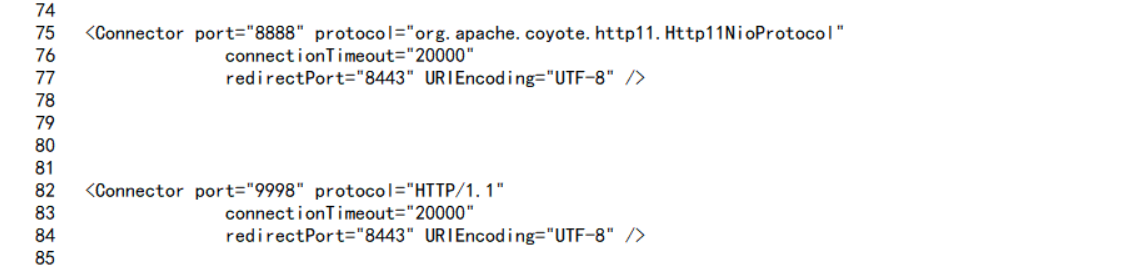


配置了 %T 以及 %D 访问日志文件如下



**2：配置 tomcat 使用Http1.1 NIO 和 Http 分别监听在 8888 端口 和 9999端口 URI 编码使用UTF-8,**

在Serverl.xml中配置如下



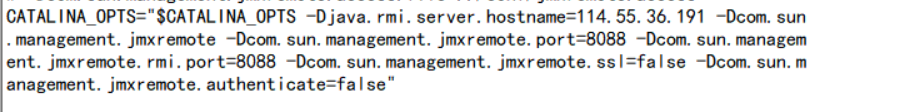
**3:将 server.xml   添加上中文注释  作为作业附件.**

在作业3另一个txt附件中：server.xml配置中注解说明.txt

4：配置 tomcat

**4.1：打开JMS 并使用 JConsole 连接 上 tomcat ,找到 刚才配置的 Connector 截图**

在catalina.sh配置





**4.2:配置内存 最大 2G ,最小 1G  perm size 为128M (java7),**

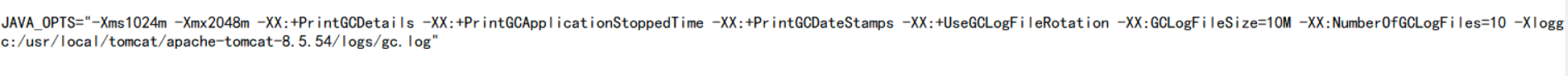
我的是tomcat8 在catalina.sh下配置

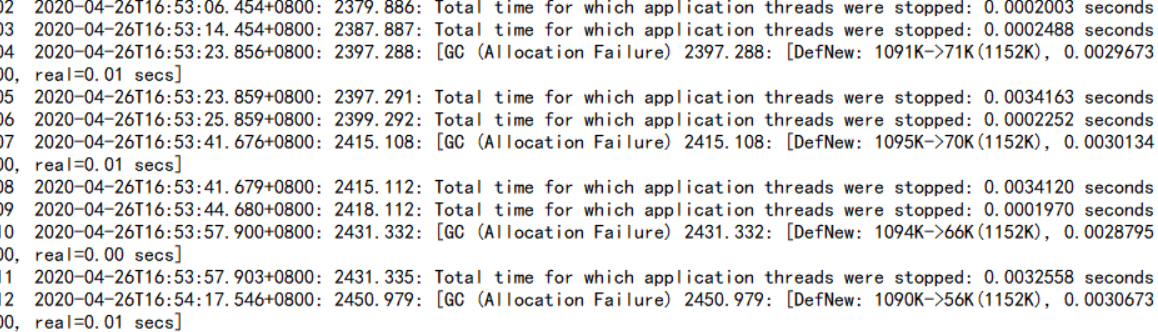


**4.3：配置 GC日志输出到 tomcat logs 目录下的 gc.log  ,gc输出的 时间格式 使用 日期格式 不要启动毫秒数**

在catalina.sh下配置:







**4.4:配置日志切割**

在catalina.sh下配置，可以打开转存储,每个日志文件大小为10M GC日志文件个数为10



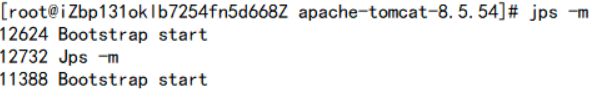
在logs下面: 当前为第一个GC日志文件

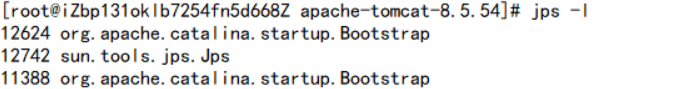


**4.5:启动 刚才配置好的 tomcat,  找到 其 PID**

    使用 jps、jstack、jmap、jhat、jstat、hprof  命令观察 JVM 状态  每个命令 和 输出 截图

**Jps:用于输出JAVA系统参数及命令行参数**





**Jinfo:来查看正在运行的java应用程序的扩展参数（JVM中-X标示的参数）；支持在运行时修改部分参数**



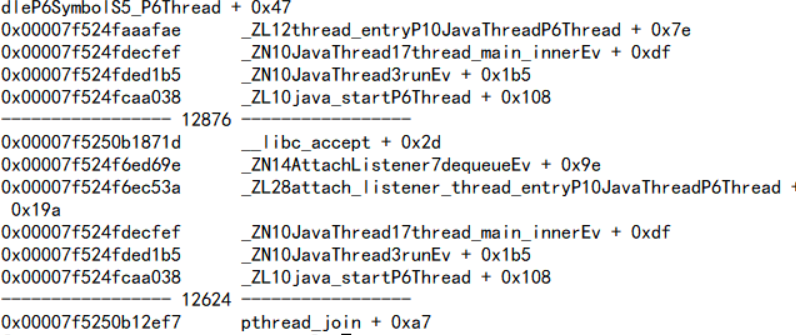


**jstack命令: 主要用来查看Java线程的调用堆栈的，可以用来分析线程问题（如死锁**

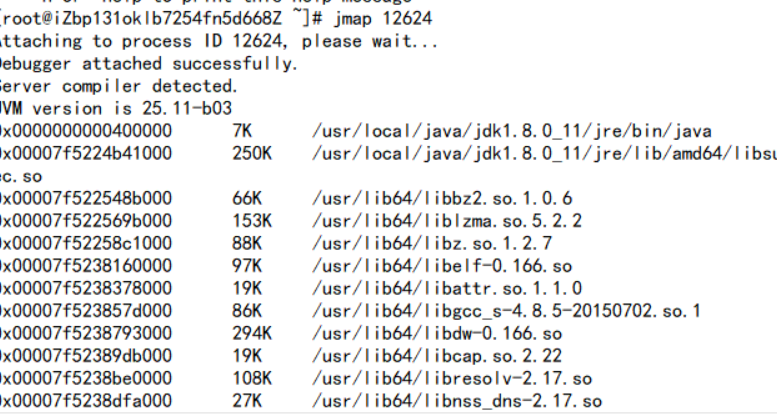
**Jstack 12624**

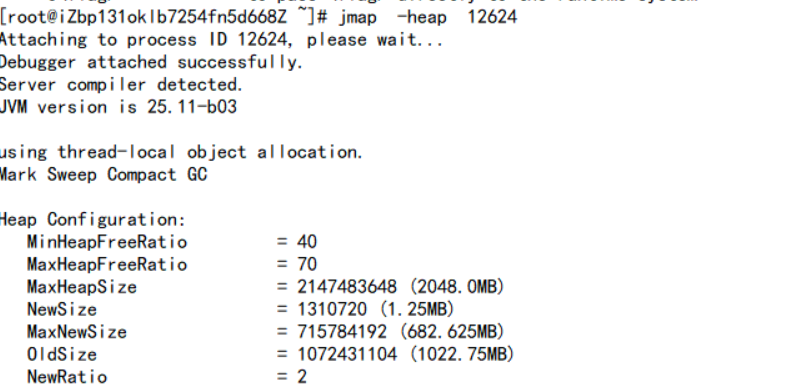


**Jstack -m 16264**



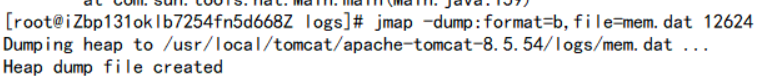
**Jmap:可以输出所有内存中对象的。**

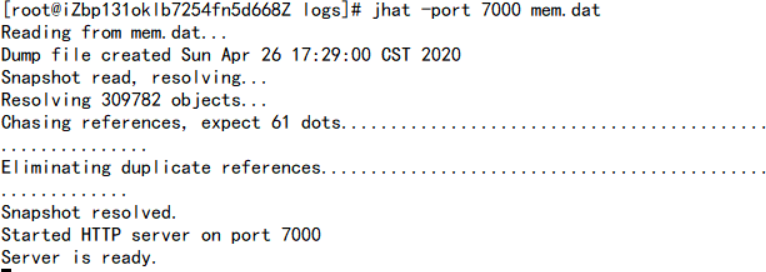


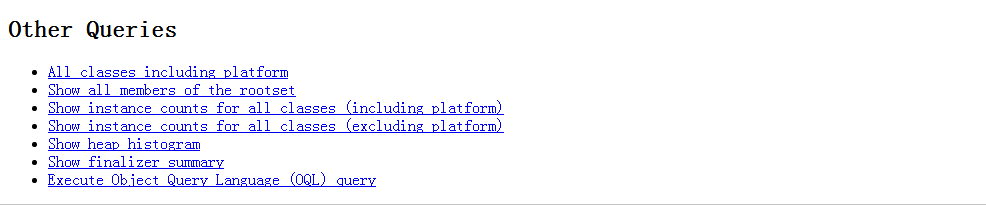


**jhat主要是用来分析java堆的命令，可以将堆中的对象以html的形式显示出来，包括对象的数量，大小等等，并支持对象查询语言。**

jmap -dump:format=b,file=mem.dat pid #将内存使用的详细情况输出到mem.dat 文件，通过jhat -port 7000 mem.dat可以将mem.dat的内容以web的方式暴露到网络，访问http://ip-server:7000查看。

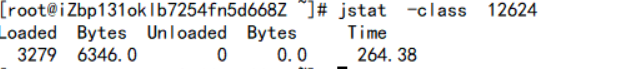




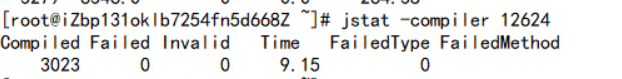


**Jstat:用来监视VM内存内的各种堆和非堆的大小及其内存使用量。详细查看堆内各个部分的使用量，以及加载类的数量.**

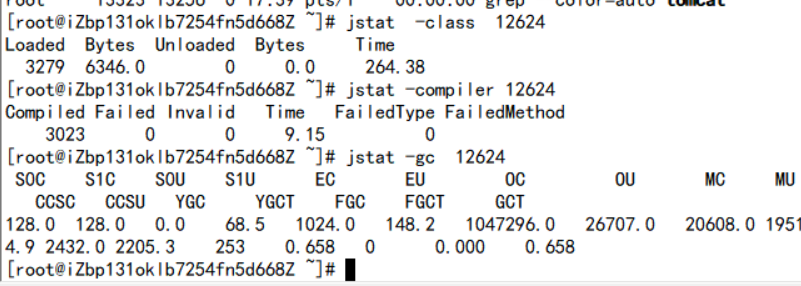
显示加载class的数量，及所占空间等信息。



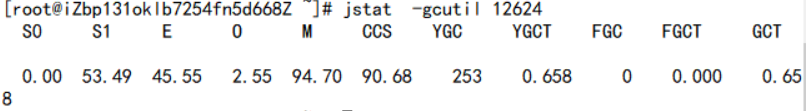
显示VM实时编译(JIT)的数量等信息。



显示gc相关的堆信息，查看gc的次数，及时间。



统计gc信息



**hprof命令:对java程序的cpu和heap进行 profiling，**

**内存溢出生成hprof文件**

