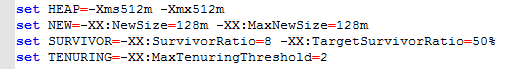
一：开源的力量

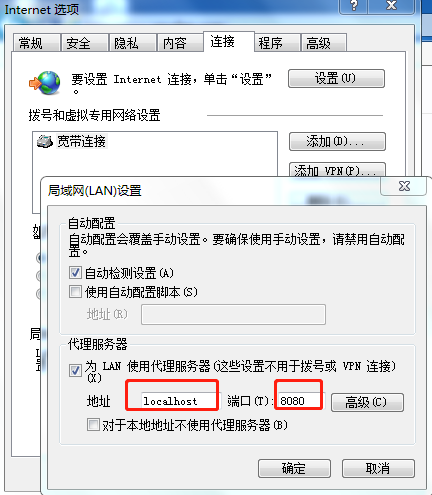
1. JMeter适合自动化和持续集成
2. Jmeter不支持执行Javascript脚本
3. 性能测试工具是通过协议和服务器交付的；自动化测试工具是通过对象识别技术来实现自动化的；
4. Jmeter支持第三方jar包，导入使用的方法有二个：
5. 将jar包放到.\JMeter\apache-jmeter-3.1\lib目录下
6. 在工具的testplan界面上的Library中导入jar包
7. Jmeter保存的测试计划就是一个以.jmx后缀的xml文件

二：初认Jmeter

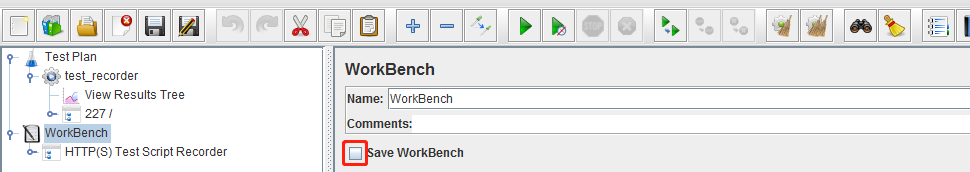
1. 做复杂线程多的性能测试时，发现jmeter内存不够用时，可以调高jmeter的内存堆，即java虚拟机启动配置，修改文件jmeter.bat文件中以下相关配置：



1. Jmeter中的录制：即浏览器中的请求是先给到jmeter，再由jmeter发送给远程的Web服务器，即jmeter作为代理服务器接收请求后发送给web服务器，同时抓包录制脚本：
2. 在WorkBench添加HTTP(S) Test Script Recorder；
3. 设置Port，如为8080；
4. Test Plan content中的Target Controller：即录制到的脚本收集在哪里，一般是选择当前测试计划里的线程组，因为生产脚本后肯定需要调试修改的；
5. Grouping：如何分组（Do not group samplers：不分组；Store 1st sampler of each group only:每组只录制第一个采集器的请求；Put each group in a new controller：为每个组（采样器）创建一个控制器）
6. HTTP Sampler settings选择目前最新的HttpClinet4；
7. 在IE浏览器上设置代理如下图：



PS：WorkBench默认不会保存到.jmx文件中，需要勾选以下项才能被保存到.jmx文件中：



三：jmeter重要组件

1. 组件的执行顺序是怎么样的？以及作用域？
2. 配置元件先执行，再执行取样器，之后是监听器；
3. 一个组件上有多个配置元件时，会得到多个配置元件合并结果，且优先取离取样器近的配置元件；
4. 先执行记时器，再执行取样器；
5. 同一类型的组件在同一层时，从上到下执行组件；

总结：不同组件的执行顺序

1—配置节点

2—前置处理器

3—定时器

4—取样器

5—后置处理器（只在有结果可用情况下执行）

6—断言（只在有结果可须知情况下执行）

7—监听器（中在有结果可用情况下执行）

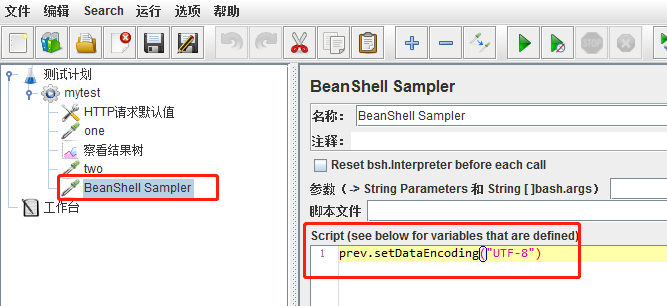
1. 响应结果出现乱码，怎么解决？

答：方法1，jmeter工具上设置content encoding为uft-8

方法2，修改bin目录下的jmeter.properties文件的配置：

sampleresult.default.encoding=UTF-8

方法3，增加一个BeanShall Samplerr，增加脚本如下：



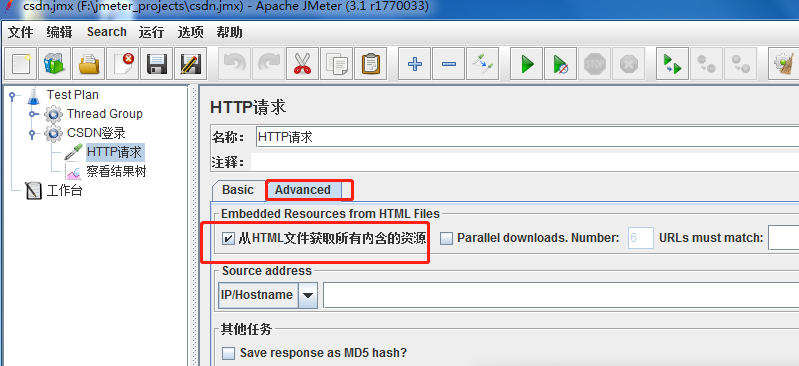
1. Jmeter组件：sampler，Timer，Pre Processors和Post Processors，Assertions，Controller（控制脚本运行流程或逻辑），Listener，Config Element

第4讲-1，2，3

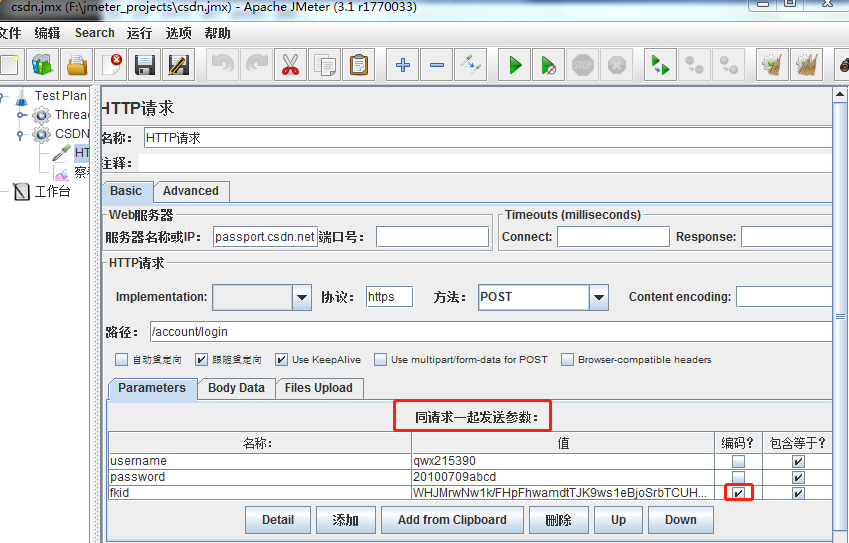
Jmeter脚本组成和组件搭配

1. 用抓包工具获取一个请求的响应后，获取到多个请求，而用jemter发送同一个请求时，即只获取了这个发送的请求，怎么办？

答：在http请求取样器中勾选上，从HTML文件获取所有内含的资源，从而就可以解析出所有的嵌入资源，如下图：



1. 如果URL的参数里有特殊符号需要传输，那在配置参数时，需要勾选这上一列，如下图：



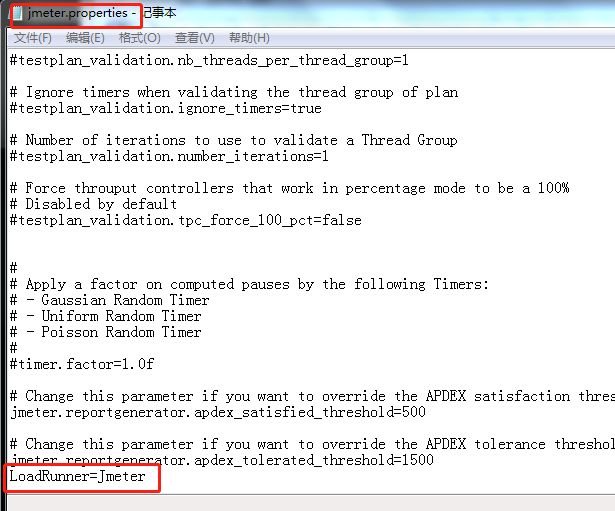
包含等于?的意思是：名称和值之间是用=连接，即username=qwx215390

1. Http cookie manager：不需要配置什么信息，会自动的处理网站的cookie发送和接收，即通过站点上获取的cookie，该cookie是每个线程各自的独立管理自己的；如果在该组件里定义了一个cookie（增加维护了一个cookie），是被所有线程共享的；
2. Cookie Policy（cookie策略）：一般选择compatibility（兼容性）；
3. Implementation（协议）：一般选择HC4CookieHandler
4. 部分组件仅仅使用在脚本开发模式下，在真正的生产环境下不要使用，如Listener越少越好，因为它非常占用资料，可以使用将运行结果保存在文档中的listener，即Simple Data Writer
5. Manager组件和Defaults组件的区别：
6. Manager组包含：HTTP Authorization Manager，HTTP Cache Manager，HTTP Cookie Manager，HTTP Header Manager
7. Defaults组件包含：FTP Request Defaults，HTTP Request Defaults，Java Request Defaults

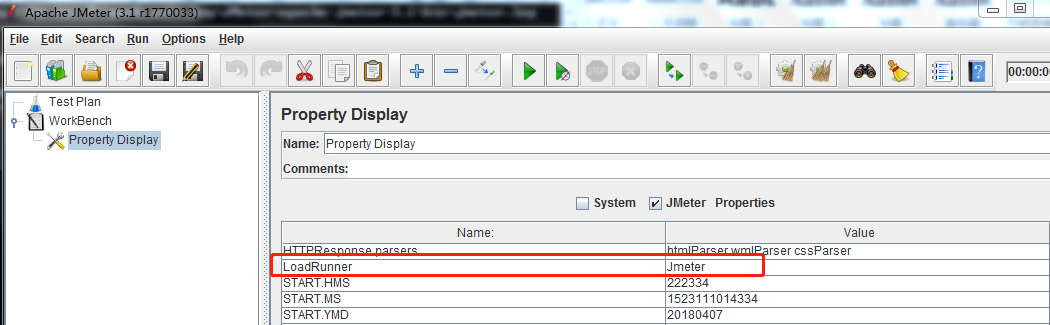
3>区别：在同一个作用域下，不可以使用二个相同的Manager组件（使用了会导致jmeter产生不可预知的错误），但是Defaults组件可以（使用了多个，就会将多个Defaults参数叠加作用到sampler上；）

第4讲-4：

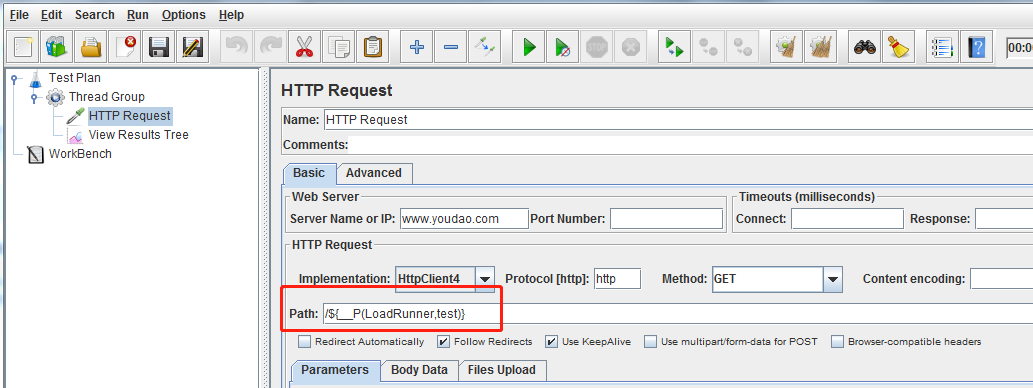
JMeter属性

1. 属性查看，在WorkBench 🡪 Add 🡪 Non Test Elements 🡪 Property Display
2. 属性是共所有线程共享的，即全局，如果某个线程修改了一个属性，则其他的线程也针对该属性也会被修改，实现了在线程间传递信息；
3. 定义jmeter属性，只需在jmeter.properties文件中增加，如LoadRunner=Jmeter，如下图： 

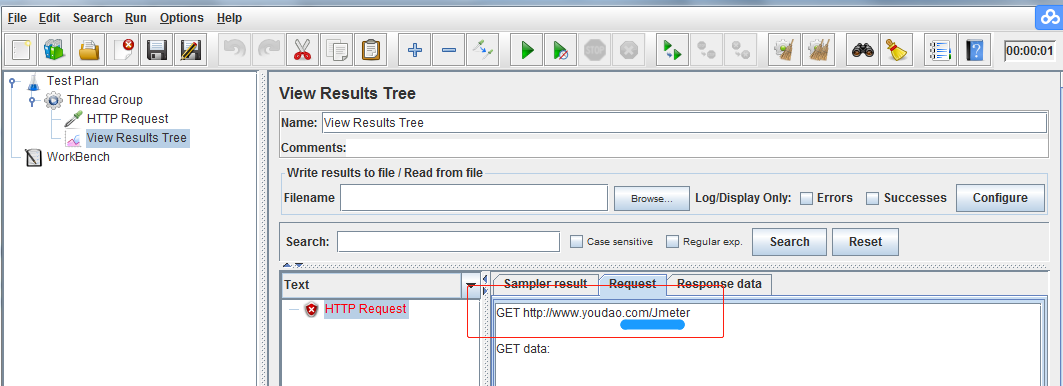
重启后即可以看到新增加的属性：



1. 使用属性：通过\_\_P来引用，${\_\_P(LoadRunner,test)}，配置如下：



FNJMGVY00发送使用的URL为http://www.youdao.com/Jmeter，如下图：



第4讲-5

Jmeter变量

1. jmeter的性能指标说明：

样本数目：总共发送到服务器的请求数.

最新样本：代表时间的数字,是服务器响应最后一个请求的时间.

吞吐量：服务器每分钟处理的请求数.

平均值：总运行时间除以发送到服务器的请求数.

中间值：时间的数字,有一半的服务器响应时间低于该值而另一半高于该值.

偏离：服务器响应时间变化、离散程度测量值的大小,或者,换句话说,就是数据的分布.

关于你说的测试值范围,可根据你的不同测试目的进行设置.简单来讲,线程数代表有多少个线程,也就是代表多少个用户；Ramp-Up Period(in-seconds)代表隔多长时间执行,0代表同时并发;循环次数就是代表执行几次.

1. Ramp-Up Period(in seconds)、循环次数，其中Ramp-Up Period(in seconds)表示在这时间内创建完所有的线程。如有8个线程，Ramp-Up = 200秒，那么线程的启动时间间隔为200/8=25秒，这样的好处是：一开始不会对服务器有太大的负载；
2. Regular Expression Extractor(正则表达式提取器)：

（假设需要匹配的字符为：Set-Cookie: login\_uid=CC-31-CF-CB-3F-C2-CF-41）

1> Regular Expression：正则表达式，如Set-Cookie:(.+?);

2> Template(模板)：设置为$1$，代表取正则表达式中的(.+?)匹配的数据，结果为cookie= login\_uid=CC-31-CF-CB-3F-C2-CF-41；设置为$0$，代表取正则表达式的所有匹配数据，结果为cookie=Set-Cookie: login\_uid=CC-31-CF-CB-3F-C2-CF-41；

3> Match No.(0 for Random)(匹配数字)：正则表达式匹配结果是一个数组，匹配数字即可以看作是数组的第几个元素；为0时，随机返回匹配的一个数据；为1时，表示返回数组的第1个元素；为2时，表示返回数组的第2个元素；为-1时，表示返回匹配的全部数据；

1. JDBC Request：这个sampler可以向数据库发送一个jdbc请求（sql语句），并获取返回的数据进行操作，它经常需要和JDBC Connection Configuration配置元件一起使用；
2. Transaction Controller(事务控制器)：接口测试时，有些接口依赖前者接口的结果，因而就需要将这些接口统一看作一个事务，进行性能测试，得到的性能测试结果才会更加接近真实场景。另外：一般性能测试时，仅考虑数据的返回，不考虑页面渲染所需要的数据；但做页面的性能测试时，尤其是含有大图片，大文件等，就必须考虑这些资料加载时的性能消耗，此时就需要使用事务控制器，使页面性能测试结果更加准确；

1> Generate parent sample：勾选该配置项，则不会看到事务控制器所包含的所有子请求

1. Troughput Controller(吞吐量控制器)：控制该控制器下的采集器执行的频率，有二种模式：Percen Executions(百分比)和Total Executions(总次数)

1> Total Executions(总次数)：

不勾选Per User：线程组的Number of Threads(users)设置为2，吞吐量设置为3，则吞吐量控制器下的线程会执行2次；

勾选Per User：线程组的Number of Threads(users)设置为2，吞吐量设置为3，则吞吐量控制器下的线程会执行3次；

PS：故Per User表示的就是线程组的Number of Threads(users)线程数，勾选了就代表每个线程就会被执行；

2> Percen Executions(百分比)：吞吐量为1-100整数

不勾选Per User：吞吐量控制器执行的总次数 = 线程数 \* 循环次数 \* (N/100) ；例如1：线程组的Number of Threads(users)设置为6，吞吐量设置为50，则吞吐量控制器下的线程会执行3次（6\*1\*(50/100)=3），且执行的线程分别为Thread Group 1-2，Thread Group 1-4，Thread Group 1-6；例如2：线程组的Number of Threads(users)设置为3，Loop Count设置为2，吞吐量设置为50，则吞吐量控制器下的线程会执行3次（6\*1\*(50/100)=3），且执行的线程分别为Thread Group 1-1，Thread Group 1-2，Thread Group 1-3；

勾选Per User：若吞吐量设置大于50，则就会按线程组设置的次数执行吞吐量控制器下的线程次数；若吞吐量设置小于或等于50，则吞吐量控制器下的线程一次都不会被执行；例如1：线程组的Number of Threads(users)设置为6，吞吐量设置为50.1，吞吐量控制器下的线程就会被执行6次；

1. Switch Controller(开关控制器)：该控制器下的线程由switch value的值决定哪些线程是开着的，可以被执行，哪些是关闭的，不会被执行；

Swith Value：控制打开线程的间隔；例如1：该控制器下有A,B,C三个线程，swith value设置为1，则只有B会被执行；例如2：该控制器下有A,B,C三个线程，swith value设置为2，则只有C会被执行；

1. Simple Controller(简单控制器)：将该控制器下的采集器，逻辑控制器等组件看作一个模块功能，该控制器本身不提供任何功能；
2. Interleave Controller(交替控制器)：每次循环执行该控制器下的一个采集器，且按顺序第一次循环执行采集器1，第二次循环执行采集器2，第三次循环执行采集器3，.....

1> Ignore sub-controller blocks：

不勾选该项，该控制器下的其他控制器的采集器会看作一个整体，某次循环时会被一次执行；例如1：该控制器下有采集器A，一个simple Controller(下面有采集器B,C)，还有采集器D,E,F，线程组的Loop Count设置为3，执行结果是：第一次循环执行采集器A，第二次循环执行simple controller下面的采集器B,C，第三次循环执行采集器D；

勾选该项，该控制器下的其他控制器的采集也会交替循环执行；例如1：该控制器下有采集器A，一个simple Controller(下面有采集器B,C)，还有采集器D,E,F，线程组的Loop Count设置为3，执行结果是：第一次循环执行采集器A，第二次循环执行simple controller下面的采集器B，第三次循环执行采集器D；

1. Interleave across threads：还没有搞明白用法？？
2. Random Controller(随机控制器)：类似于交替控制器，区别是交替控制器是按顺序循环执行采集器的，而它是无序的，且该控制器下的采集器被执行的概率不同；
3. Random Order Controller(随机顺序控制器)：类似于简单控制器，区别是简单控制器下的采集器会按顺序执行，而该控制器下的采集器执行顺序是随机的；