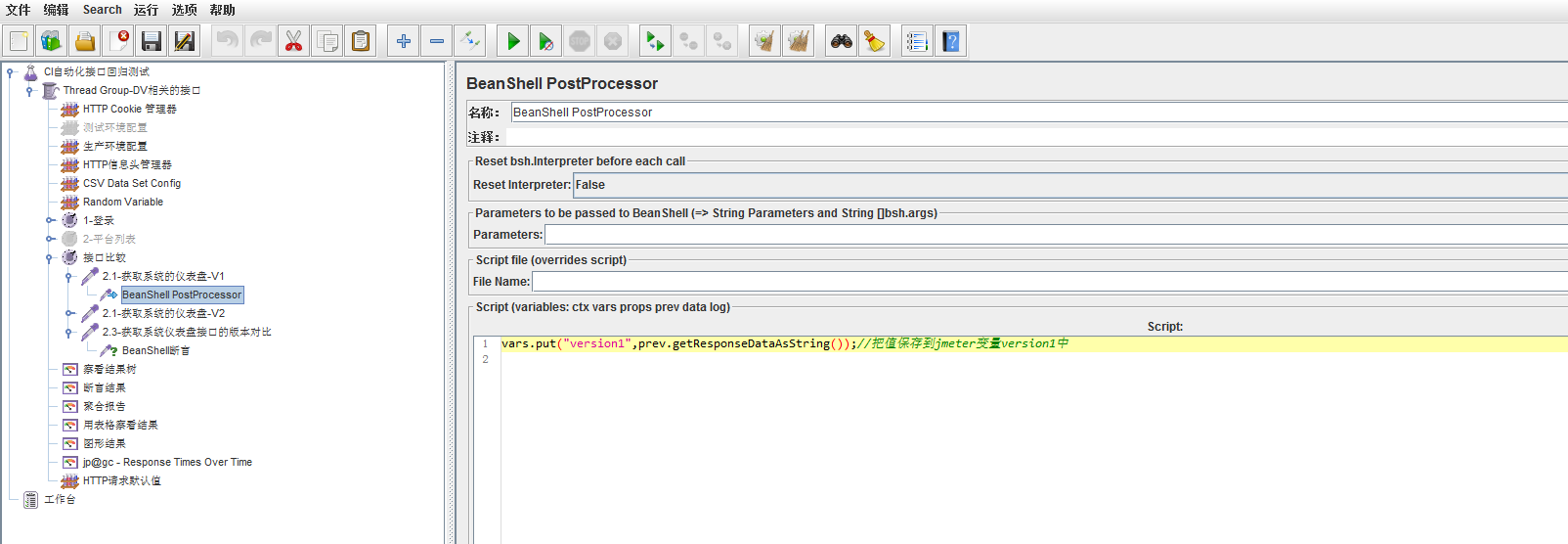
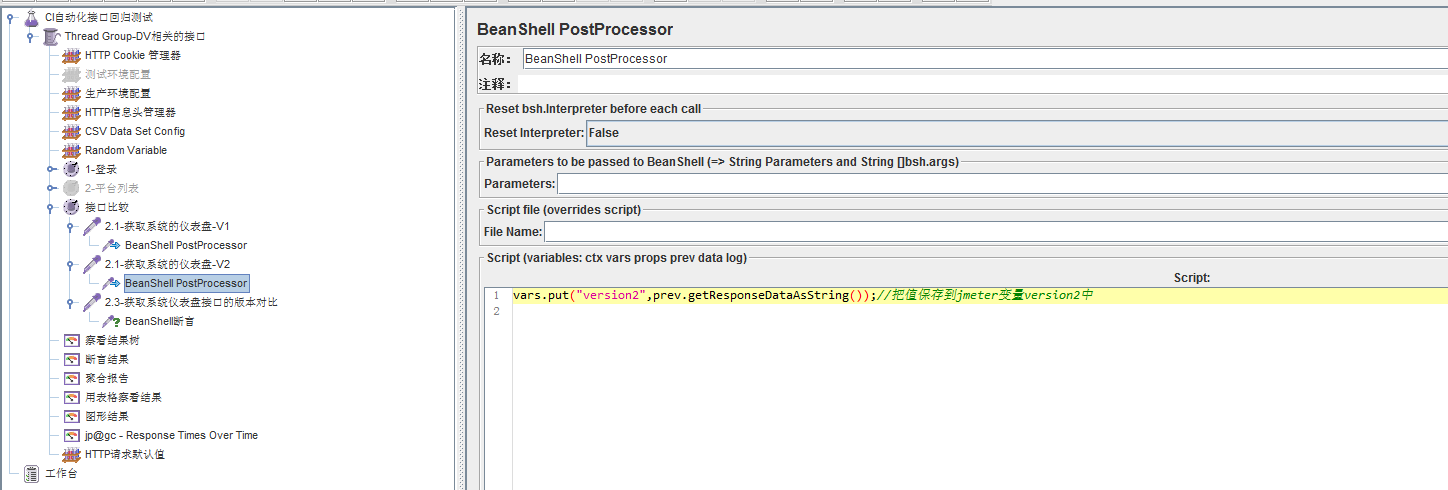
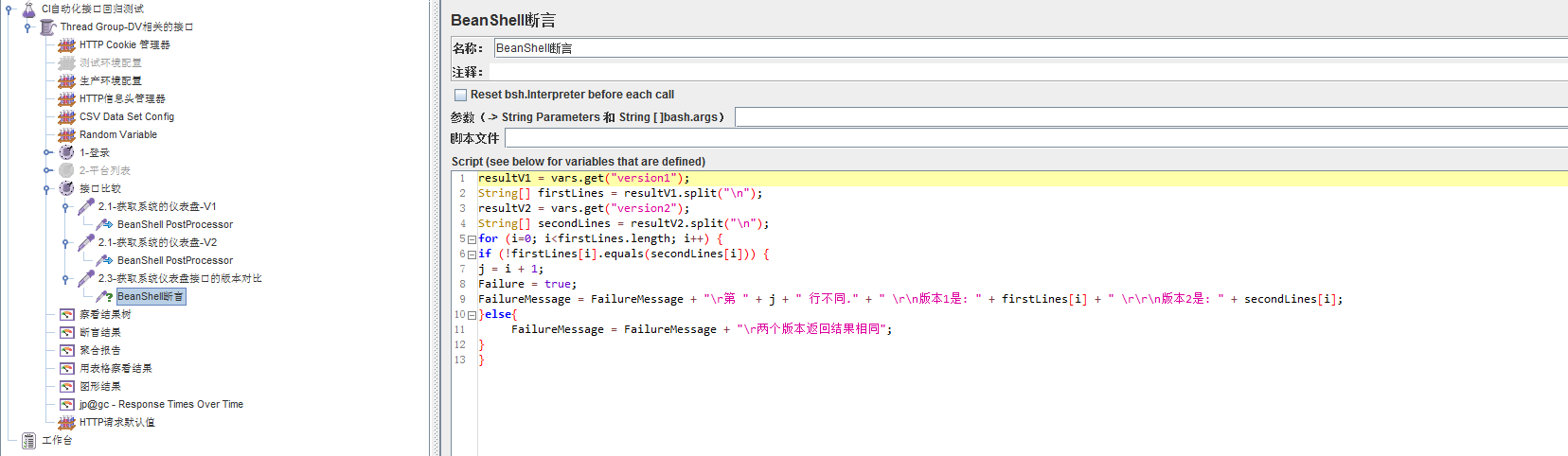
　　大多数人都使用 Jmeter 做过性能测试，但是在使用的过程中你会发现，它不仅可以做性能测试和功能测试，还能够满足基本的**[接口测试](javascript:;" \t "_self)**需求。  
　　相比其他工具，Jmeter 入门门槛较低，安装也比较方便，根据自己的需要可以扩展一些插件，总之一句话: 优点太多了。  
　　那么问题来了，为什么要做接口 diff 测试？  
　　产品迭代较快，提高已有 case 回归的效率，减轻工作量；  
　　通过 diff 可以看到新的改动对已有功能的影响；  
　　可以根据修改的代码设置 case，有针对性的测试；  
　　在开发联调阶段提前定位发现的问题。  
　　下面就介绍下通过 Jmeter 进行 diff 测试的步骤：  
**1.添加测试计划  
　　2.添加 Threads  
　　3.添加 Sampler-HTTP 请求**  
　　旧版本的接口：interfaceV1  
**4.针对 Sampler interfaceV1 添加后置处理器 - BeanShell PostProcessor，如图：**



**5.添加Sampler-HTTP请求**  
　　新版本的接口：interfaceV2  
**6.针对Sampler interfaceV2添加后置处理器 - BeanShell PostProcessor**



**7.针对 Sampler interfaceV2 添加后置处理器 - BeanShell Sampler  
　　8.在 BeanShell Sampler 添加断言 - BeanShell 断言**



　　下面是针对响应的信息进行简单的比较  
　　resultV1 = vars.get("version1");   
　　String[] firstLines = resultV1.split(" ");   
　　resultV2 = vars.get("version2");   
　　String[] secondLines = resultV2.split(" ");   
　　for (i=0; i<firstLines.length; i++) {   
　　if (!firstLines[i].equals(secondLines[i])){   
　　j = i + 1;   
　　Failure = true;   
　　FailureMessage = FailureMessage + " 第 " + j + " 行不同." + " 版本1是: " + firstLines[i] + " 版本2是: " + secondLines[i];   
　　}else{  
　　FailureMessage = FailureMessage + " 两个版本返回结果相同";   
　　}  
　　}  
**9.添加监听器-断言结果  
　　10.运行脚本后，查看断言结果**



　　经过以上几步就完成了一个简单的接口 diff 测试  
　　如果想实现自动化，可以集成到 jenkins 中。针对不同的环境，要想保证脚本都能够正常运行，可以添加配置元件 - 用户定义的变量，将环境配置作为变量保存。  
　　此外，如有其他小伙伴发现了更好的一些针对 diff 测试的工具或方法，请多多指教。