# MeterSphere 接口请求及响应特殊内容的脚本化处理

MeterSphere 在做接口测试过程中，有些请求及响应的特殊内容需要通过脚本进行处理，才能达到期望的测试结果，本文针对特殊内容的脚本处理做简单介绍，能够帮助大家利用脚本灵活处理接口请求以及相应内容。

## 一、通过前置脚本处理接口请求内容

### 1.1 如何通过前置脚本添加或者修改请求头信息

针对某些接口请求头需要进行逻辑性的参数添加及修改，常规的函数或者变量无法达到要求，此时可以通过前置脚本进行处理。具体BeanShell脚本内容如下：

import org.apache.jmeter.protocol.http.control.HeaderManager;

import org.apache.jmeter.protocol.http.control.Header;

// 获得请求头信息

HeaderManager headers = sampler.getHeaderManager();

// 打印全部请求头信息

log.info("添加前" + headers.getHeaders().getStringValue());

// new一个Header对象

myHeader = new Header("x-parse-session-token", "123456432");

// 添加Header到请求头管理器

headers.add(myHeader);

// 打印全部请求头信息

log.info("添加后" + headers.getHeaders().getStringValue());

### 1.2 如果通过前置脚本获取请求中的域名或者host信息

某些场景需要通过前置脚本针对不同的域名或者请求host进行逻辑处理，就需要通过如下BeanShell脚本获取信息：

import org.apache.jmeter.config.Arguments;

import java.util.Map.Entry;

Arguments args = sampler.getArguments();

//获取域名或者host信息

log.info("domain: "+sampler.getDomain());

//获取请求参数

Map map = args.getArgumentsAsMap();

Iterator itor = map.entrySet().iterator();

while(itor.hasNext()){

  Entry entry = (Entry) itor.next();

  log.info("==========key:"+entry.getKey());

}

## 二、通过后置脚本处理接口响应内容

### 2.1 通过后置脚本将Unicode转码为中文

当响应内容为Unicode时，在MeterSphere上显示的内容为乱码，此时就需要通过后置脚本将响应内容转化为UTF-8，以便于正常显示。在对应接口下添加一个后置脚本，详细的BeanShell脚本如下：

String s=new String(prev.getResponseData(),"UTF-8");

        char aChar;

        int len= s.length();

        StringBuffer outBuffer=new StringBuffer(len);

        for(int x =0; x <len;){

            aChar= s.charAt(x++);

            if(aChar=='\\'){

                aChar= s.charAt(x++);

                if(aChar=='u'){

                    int value =0;

                    for(int i=0;i<4;i++){

                        aChar= s.charAt(x++);

                        switch(aChar){

                            case'0':

                            case'1':

                            case'2':

                            case'3':

                            case'4':

                            case'5':

                            case'6':

                            case'7':

                            case'8':

                            case'9':

                                value=(value <<4)+aChar-'0';

                                break;

                            case'a':

                            case'b':

                            case'c':

                            case'd':

                            case'e':

                            case'f':

                                value=(value <<4)+10+aChar-'a';

                                break;

                            case'A':

                            case'B':

                            case'C':

                            case'D':

                            case'E':

                            case'F':

                                value=(value <<4)+10+aChar-'A';

                                break;

                            default:

                                throw new IllegalArgumentException(

                                        "Malformed   \\uxxxx  encoding.");}}

                    outBuffer.append((char) value);}else{

                    if(aChar=='t')

                        aChar='\t';

                    else if(aChar=='r')

                    aChar='\r';

                    else if(aChar=='n')

                    aChar='\n';

                    else if(aChar=='f')

                    aChar='\f';

                    outBuffer.append(aChar);}}else

                outBuffer.append(aChar);}

prev.setResponseData(outBuffer.toString());

String s=new String(prev.getResponseData(),"UTF-8");

### 2.2 通过后置脚本将文件内容的响应内容存储到服务器

当接口响应内容为文件时，MeterSphere 会将文件内容显示为乱码，此种方式对于测试人员不太友好，所以可以通过后置脚本将响应的文件内容写入实际的文件到服务器，然后从服务器上的目录下进行拿取（或者写入可访问的文件服务器）。详细的BeanShell脚本内容如下：

import java.io.\*;

byte[] result = prev.getResponseData();

String file\_name = "/opt/metersphere/data/xxxx.xxx";

File file = new File(file\_name);

FileOutputStream out = new FileOutputStream(file);

out.write(result);

out.close();