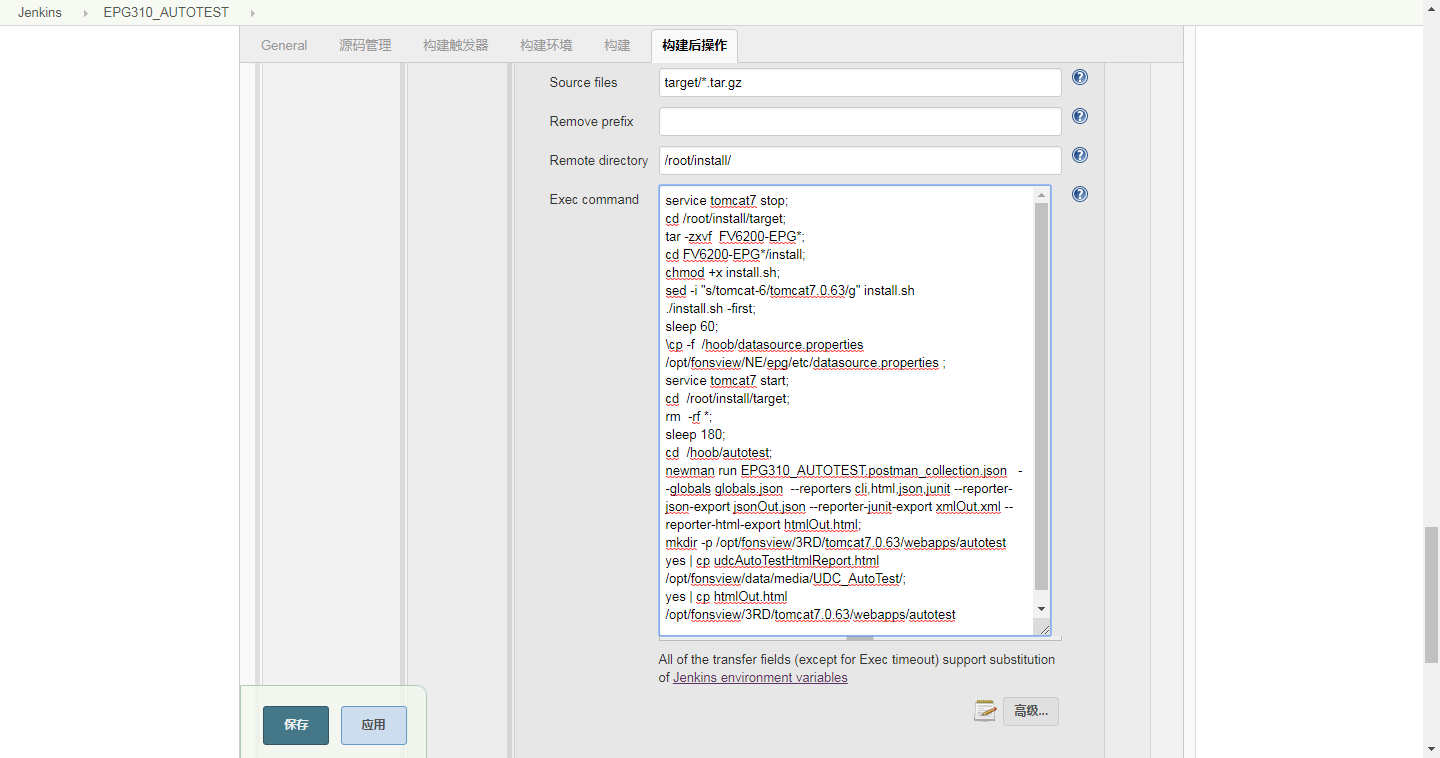
网友分享参考：<https://blog.csdn.net/xiaosongbk/article/details/64921333>

Postman 脚本命令执行需要配合 newman ,newman 需要安装node.js,node.js 需要安装python2.7,安装node.js 时会自动安装python2.7

1. 安装node.js 官网下载对应版本的node.js 包 安装即可
2. 安装 newman npm install -g newman
3. Postman 中添加对应测试的URL 后导出对应的.json 文件，然后到jenkins中添加对应job任务。（172.16.6.103服务器上面不素jenkins,测试的EPG310自动部署到172.16.199.10上为例）



service tomcat7 stop;

cd /root/install/target;

tar -zxvf FV6200-EPG\*;

cd FV6200-EPG\*/install;

chmod +x install.sh;

sed -i "s/tomcat-6/tomcat7.0.63/g" install.sh

./install.sh -first;

sleep 60;

\cp –f /hoob/datasource.properties /opt/fonsview/NE/epg/etc/datasource.properties ;

service tomcat7 start;

cd /root/install/target;

rm -rf \*;

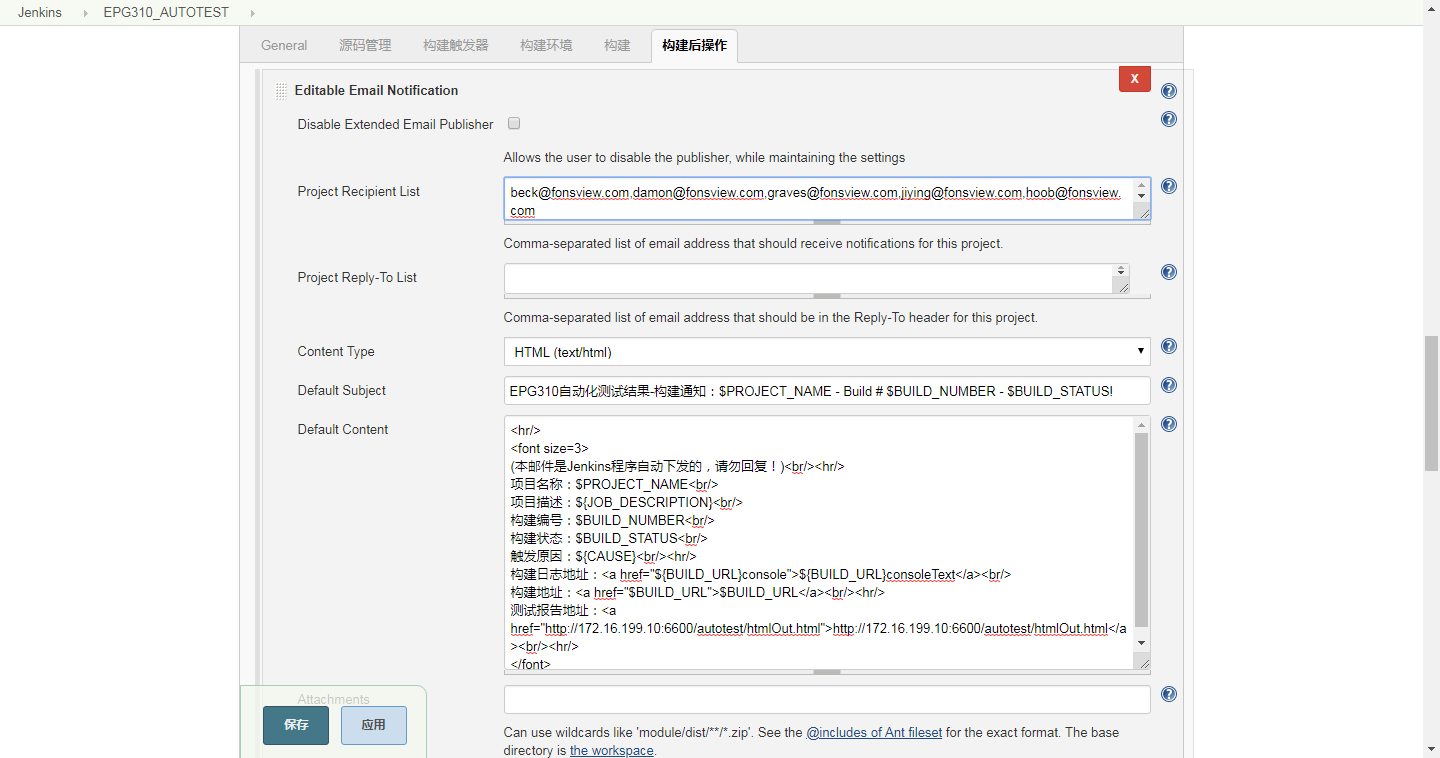
sleep 180;

cd /hoob/autotest;

newman run EPG310\_AUTOTEST.postman\_collection.json --globals globals.json --reporters cli,html,json,junit --reporter-json-export jsonOut.json --reporter-junit-export xmlOut.xml --reporter-html-export htmlOut.html;

mkdir -p /opt/fonsview/3RD/tomcat7.0.63/webapps/autotest；

yes | cp htmlOut.html /opt/fonsview/3RD/tomcat7.0.63/webapps/autotest；



newman run EPG310\_AUTOTEST.postman\_collection.json --environment env.json --iteration-data data.csv --globals globals.json --timeout-request 5000

newman run EPG310\_AUTOTEST.postman\_collection.json --globals globals.json --reporters cli,html,json,junit --reporter-json-export jsonOut.json --reporter-junit-export xmlOut.xml --reporter-html-export htmlOut.html

1. 清除一个全局变量

Clear a global variable

对应脚本：

postman.clearGlobalVariable("variable\_key");

参数：需要清除的变量的key

2.清除一个环境变量

Clear an environment variable

对应脚本：

postman.clearEnvironmentVariable("variable\_key");

参数：需要清除的环境变量的key

3.response包含内容

Response body:Contains string

对应脚本：

tests["Body matches string"] =responseBody.has("string\_you\_want\_to\_search");

参数：预期内容

4.将xml格式的response转换成son格式

Response body:Convert XML body to a JSON Object

对应脚本： var jsonObject = xml2Json(responseBody);

参数：（默认不需要设置参数,为接口的response）需要转换的xml

5.response等于预期内容

Response body:Is equal to a string

对应脚本：

tests["Body is correct"] = responseBody === "response\_body\_string";

参数：预期response

6.json解析key的值进行校验

Response body:JSON value check

对应脚本：

tests["Args key contains argument passed as url parameter"] = 'test' in responseJSON.args

参数：test替换被测的值，args替换被测的key

7.检查response的header信息是否有被测字段

Response headers:Content-Type header check

对应脚本：

tests["Content-Type is present"] = postman.getResponseHeader("Content-Type");

参数：预期header

8.响应时间判断

Response time is less than 200ms

对应脚本：

tests["Response time is less than 200ms"] = responseTime < 200;

参数：响应时间

9.设置全局变量 Set an global variable

对应脚本：

postman.setGlobalVariable("variable\_key", "variable\_value");

参数：全局变量的键值

10.设置环境变量Set an environment variable

对应脚本：

postman.setEnvironmentVariable("variable\_key", "variable\_value");

参数：环境变量的键值

11.判断状态码

Status code:Code is 200

对应脚本：

tests["Status code is 200"] = responseCode.code != 400;

参数：状态码

12.检查code name 是否包含内容

Status code:Code name has string

对应脚本：

tests["Status code name has string"] = responseCode.name.has("Created");

参数：预期code name包含字符串

13.成功的post请求

Status code:Successful POST request

对应脚本：

tests["Successful POST request"] = responseCode.code === 201 || responseCode.code === 202;

14.微小验证器

Use Tiny Validator for JSON data

对应脚本：

var schema = {"items": {"type": "boolean"}};

var data1 = [true, false];

var data2 = [true, 123];

console.log(tv4.error);

tests["Valid Data1"] = tv4.validate(data1, schema);

tests["Valid Data2"] = tv4.validate(data2, schema);

参数：可以修改items里面的键值对来对应验证json的参数