目录

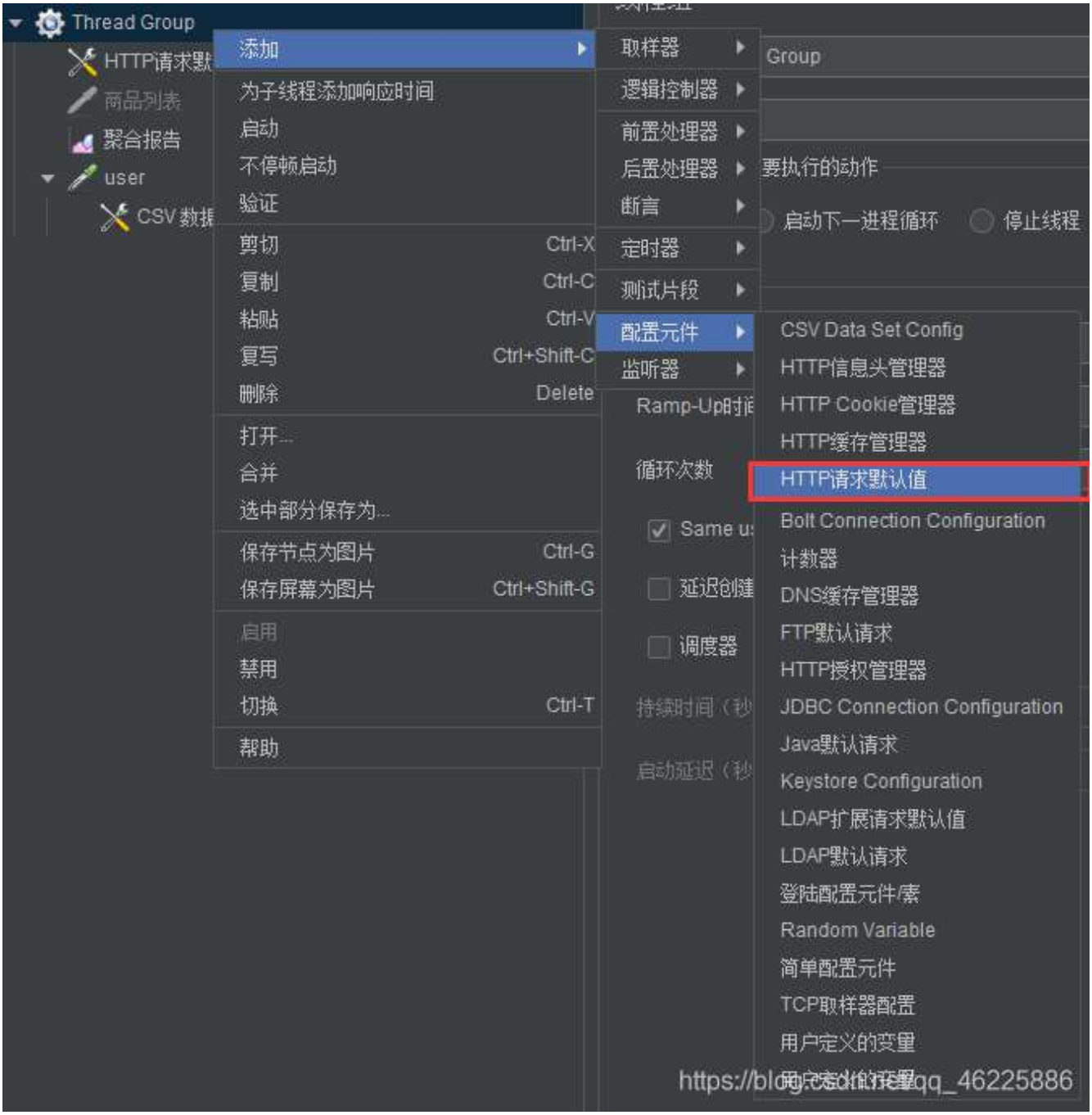
---

1. JMeter压力测试

1.1 测试过程

1. 打开jmeter.bat

2. 设置HTTP默认请求



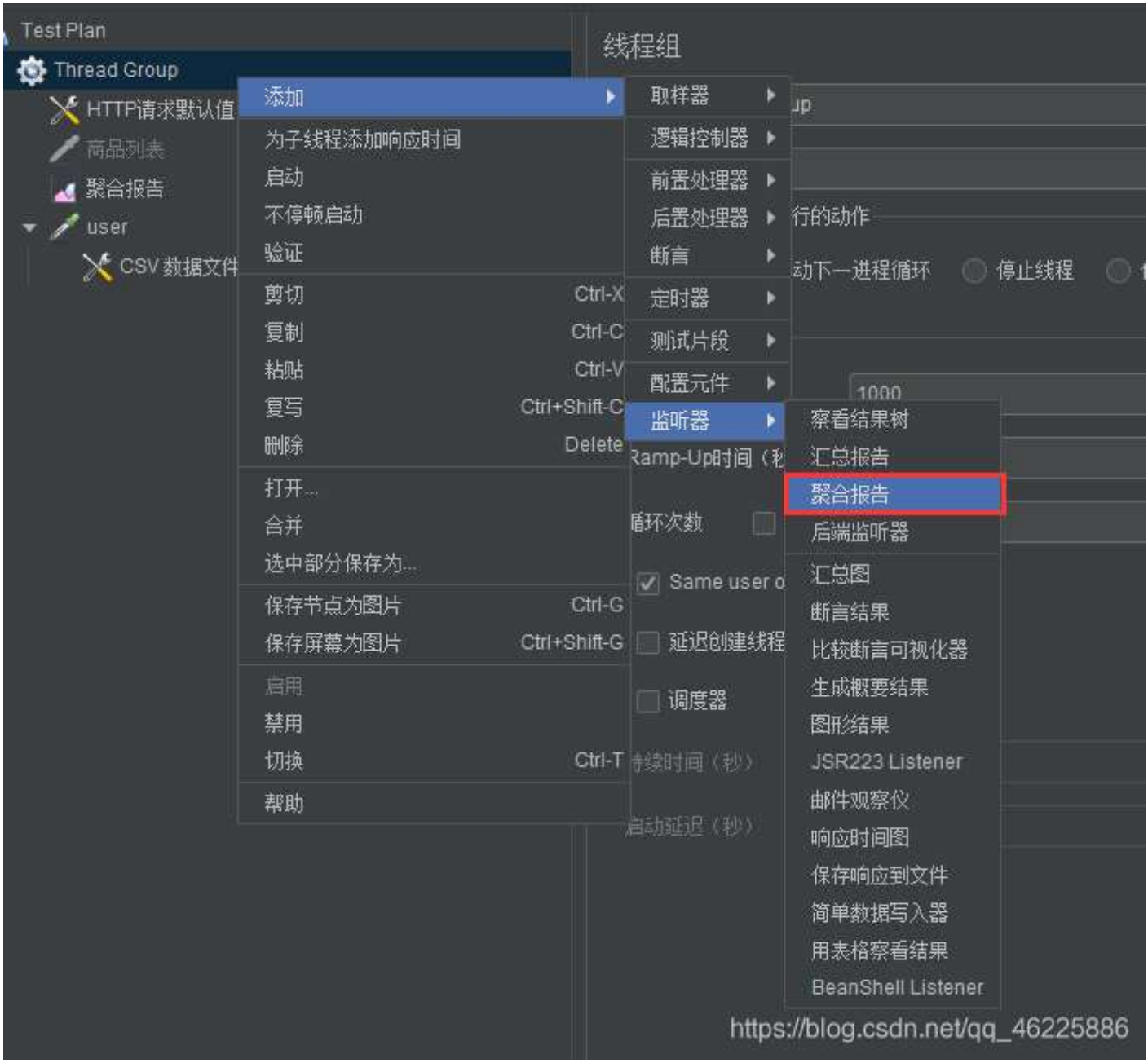
编写协议和端口号

3. 编写测试HTTP请求

因为我们已经写过 默认设置 ， 我们就可以不用编写协议和地址了，如下，只需编写 请求类型 和 地址 即可

4. 添加聚合报告

我们即可在报告中查看压测信息



1.2 Linux top命令

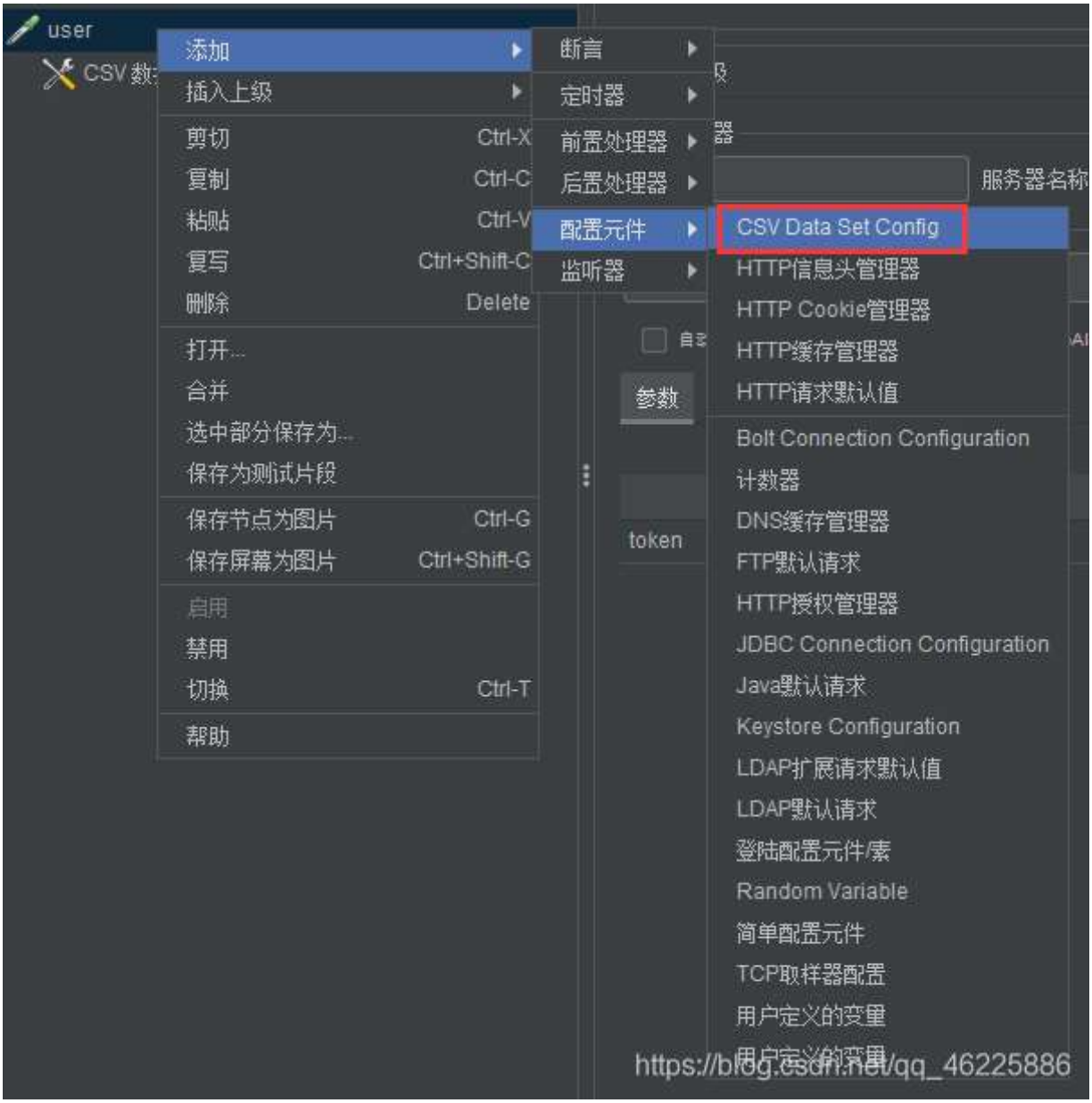
- top：相当于Windows下的任务管理器，可以动态显示当前进程的状况

---

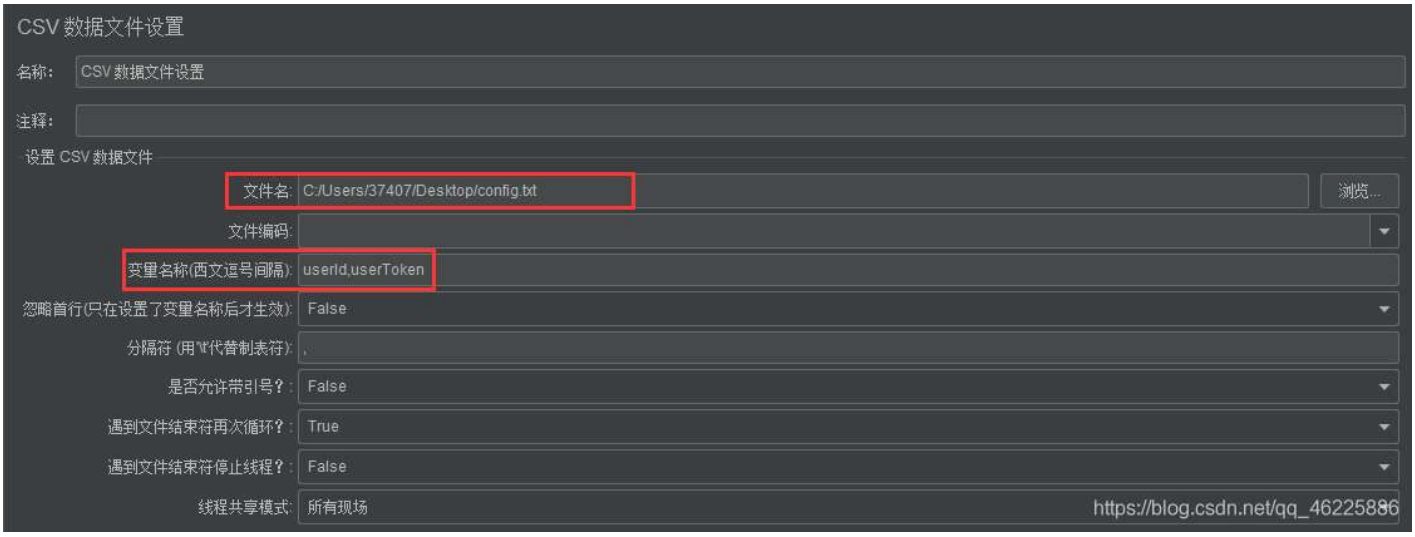
2. 自定义配置文件JMeter压测

2.1 测试过程

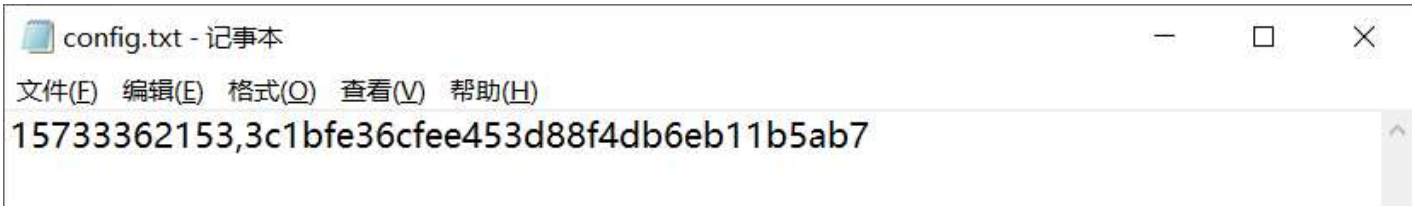
与上方基本一致，不过，要在测试的请求上， 添加CSV数据文件设置



读取我们自己编写的配置文件，并且标注变量名称，如此，即可开始压测。



其中配置文件信息，用英文逗号隔开



3. Redis压测

```
#100个并发连接，100000个请求
redis-benchmark -h 127.0.0.1 -p 6379 -c 100 -n 100000

#存取大小为100字节的数据包
redis-benchmark -h 127.0.0.1 -p 6379 -q -d 100

#测试set和lpush命令的QPS，其中-q为简化输出
redis-benchmark -t set,lpush -q -n 1000000

#测试单条命令的QPS
redis-benchmark -n 100000 -q script load "redis.call('set','foo','bar')"
```

4. Linux环境下，命令行压测

1. 在Windows目录下写好 jmx文件
2. 命令行： `sh jmeter.sh -n -t xxx.jmx -l result.jtl`
3. 再将result.jtl 导入到jmeter中

4.1 打成jar包

```
maven clean package
```

打开jar包，我们进入META-INF目录下，打开MANIFEST.MF文件，我们可以发现如下语句

```
Manifest-Version: 1.0
Spring-Boot-Classpath-Index: BOOT-INF/classpath.idx
Implementation-Title: miaosha
Implementation-Version: 0.0.1-SNAPSHOT
Start-Class: com.imooc.miaosha.MiaoshaApplication
Spring-Boot-Classes: BOOT-INF/classes/
Spring-Boot-Lib: BOOT-INF/lib/
Build-Jdk-Spec: 1.8
Spring-Boot-Version: 2.3.1.RELEASE
Created-By: Maven Jar Plugin 3.2.0
Implementation-Vendor: Pivotal Software, Inc.
Main-Class: org.springframework.boot.loader.JarLauncher
```

其中Main-Class为SpringBoot框架的启动类，在这个类中可以跟进看源码
Start-Class为我们自己编写的启动类

4.2 上传到Linux服务器上

```
#执行如下命令，之后即可根据如下地址访问
#http://182.92.xxx.xxx:8080/login
java -jar miaosha.jar
```

4.3 编写.jmx文件

在Windows上用JMeter 编写.jmx脚本 ，上传到服务器上，执行如下命令行

```
jmeter.sh -n -t good_list.jmx -l result.jtl
```

之后，下载result.jtl到Windows本地，进行报告分析

5. SpringBoot 打war包

1. 在pom.xml文件中，添加打包为war包的标签

```
<packaging>war</packaging>
```

2. 添加tomcat provided编译时的依赖

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
```

3. 在主类中， 实现SpringBootServletInitializer，重写configure()方法

```
@SpringBootApplication
public class MiaoshaApplication extends SpringBootServletInitializer {

    public static void main(String[] args) {
        SpringApplication.run(MiaoshaApplication.class, args);
    }

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
        return builder.sources(MiaoshaApplication.class);
    }
}
```

4. 将ROOT目录删除，并且把我们的war包修改为ROOT.war，放在webapps目录下，即可访问