<https://blog.csdn.net/qi_lin7/article/details/53759528>

# 【Dubbo】利用Jmeter对Dubbo接口进行压力测试

2016年12月20日 09:48:01 [南\_俊](https://me.csdn.net/qi_lin7) 阅读数：22452 标签： [jmeter](http://so.csdn.net/so/search/s.do?q=jmeter&t=blog" \t "_blank)[dubbo](http://so.csdn.net/so/search/s.do?q=dubbo&t=blog)[压力测试](http://so.csdn.net/so/search/s.do?q=%E5%8E%8B%E5%8A%9B%E6%B5%8B%E8%AF%95&t=blog) 更多

个人分类： [jmeter](https://blog.csdn.net/qi_lin7/article/category/6333488" \t "_blank)[Dubbo](https://blog.csdn.net/qi_lin7/article/category/6632422)[测试](https://blog.csdn.net/qi_lin7/article/category/6577207)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qi\_lin7/article/details/53759528

**Dubbo接口压力测试**

 2017-03-30更新：忘记了之前有个问题没有在blog里面跟大家说，由于Dubbo工程师引入了Spring相关的jar包了，但是Jmeter默认也是自带了spring-core-2.5.6.jar包的(在lib目录下)，所以当我们将测试jar放到jmeter对应目录下，重启jmeter的时候回报spring的jar包冲突，所以在利用就jmeter做Dubbo接口测试的时候，我们需要将Jmeter自带的spring-core-2.5.6.jar包去掉。

一、简介

1、  需要环境工具：

需要环境：Jdk1.7、Maven；

开发工具：intellij idea（也可自行选择eclipse）；

测试工具：Jmeter3.0；

2、  Dubbo接口压力测试：

编写dubbo接口测试代码调用dubbo服务，利用jmeter运行测试代码，进行压力测试

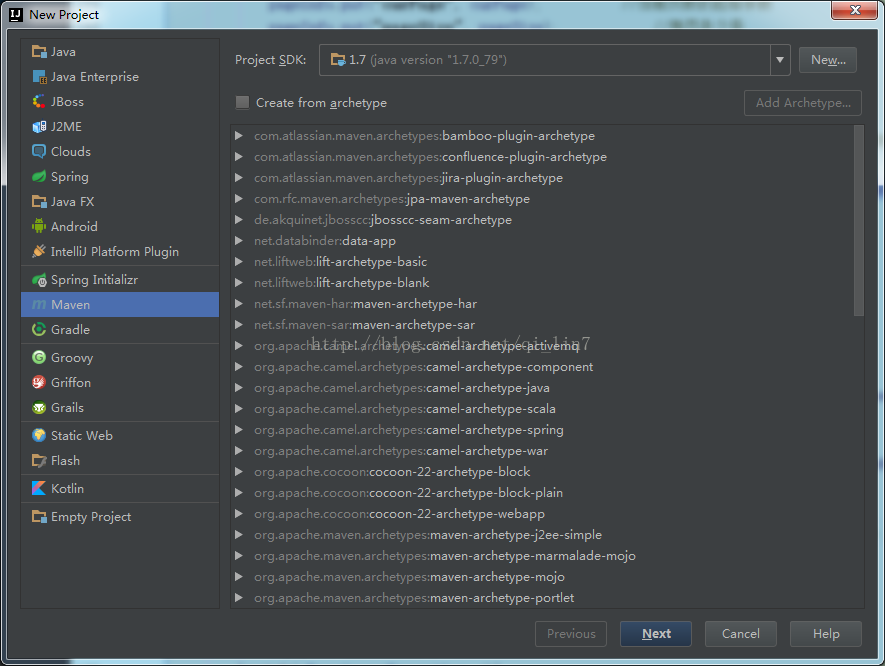
3、  Jmeter测试java接口：<http://jmeter.apache.org/api/org/apache/jmeter/protocol/java/sampler/AbstractJavaSamplerClient.html>

4、  代码下载地址：

二、编写dubbo测试代码

1、  新建dubbo接口测试工程(建议一个dubbo服务工程对应一个dubbo接口测试工程，这样方便代码的管理)

1)      .打开file -> new-> Project -> maven -> next, 新建maven工程

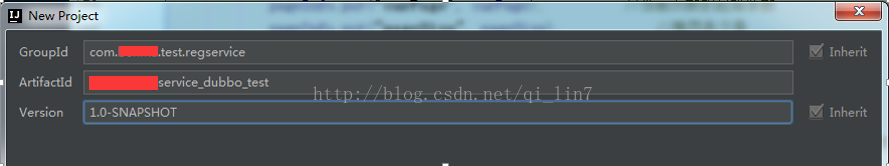


2)      .填写groupId、artifactId、version

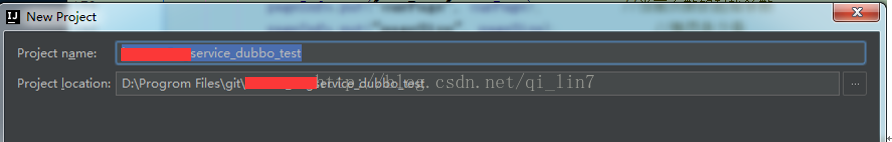
GroupId：项目组织唯一的标识符，实际对应JAVA的包的结构，是main目录里java的目录结构。

ArtifactId：就是项目的唯一的标识符，实际对应项目的名称，就是项目根目录的名称。

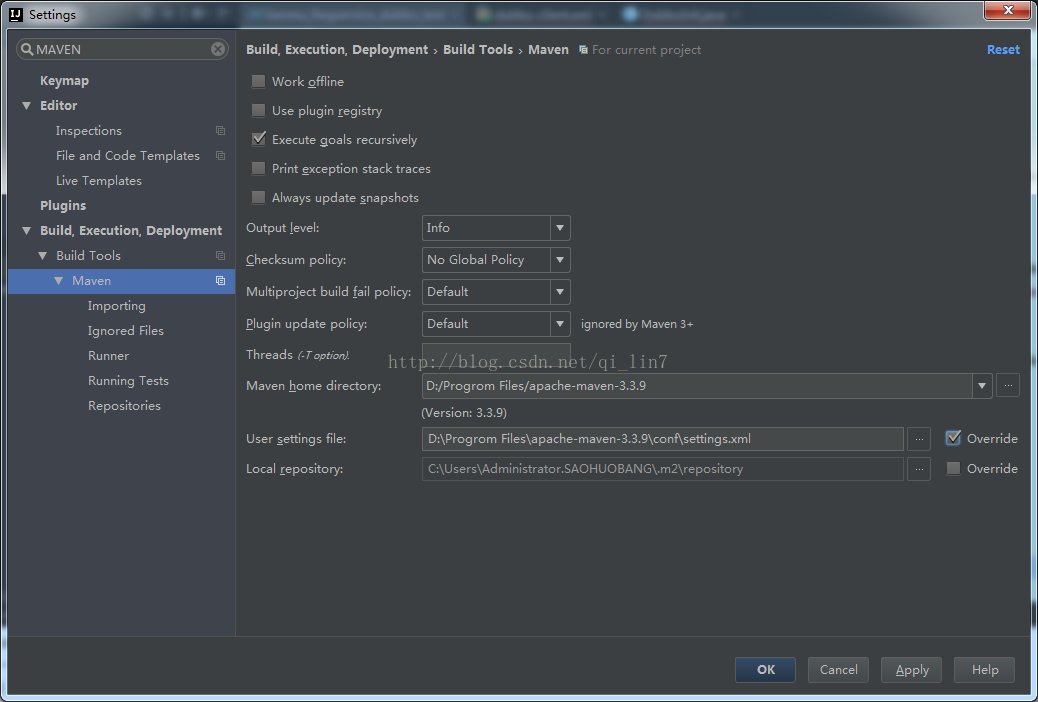
Version：版本号。



3)      填写project name



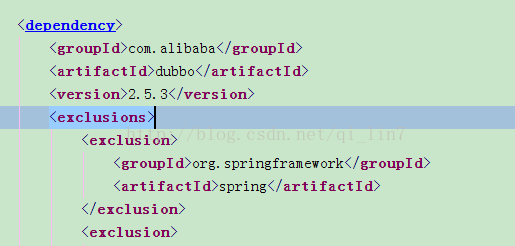
2、  配置pom.xml文件，注意：maven一定要配置成你本地的



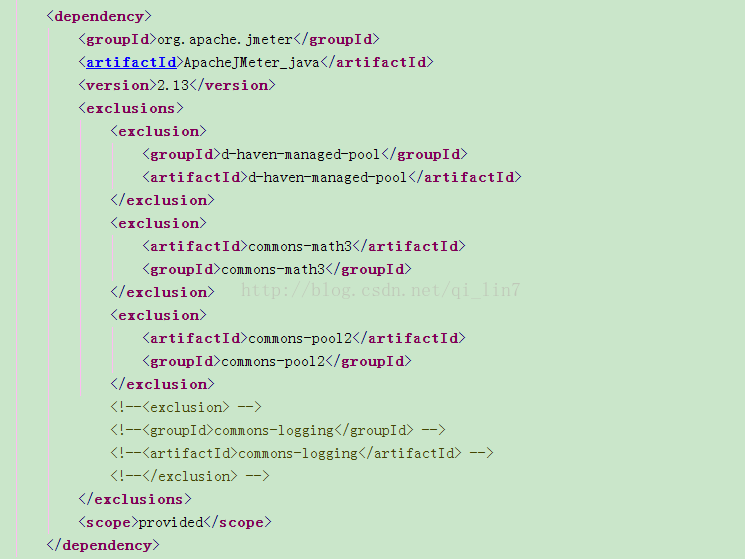
1)       配置需要测试的dubbo服务



2)      配置需要的一些工具jar包(dubbo工程会有，可参考父工程pom.xml文件)。篇幅所致，此处只截图展示部分。



3)      配置Jmeter相关的jar包，使用的是ApacheJMeter\_java，版本号此处使用2.13，可自行在jmeter官网查看版本号，根据需要配置。



4)      配置build相关（打包需要），建议就使用如下配置，不要做其他更改，使用其他配置，可能导致产生一些无法预知的问题。

1. <build>
2. <plugins>
3. <plugin>
4. <groupId>org.apache.maven.plugins</groupId>
5. <artifactId>maven-shade-plugin</artifactId>
6. <version>1.4</version>
7. <executions>
8. <execution>
9. <phase>package</phase>
10. <goals>
11. <goal>shade</goal>
12. </goals>
13. <configuration>
14. <filters>
15. <filter>
16. <artifact>\*:\*</artifact>
17. <excludes>
18. <exclude>META-INF/\*.SF</exclude>
19. <exclude>META-INF/\*.DSA</exclude>
20. <exclude>META-INF/\*.RSA</exclude>
21. </excludes>
22. </filter>
23. </filters>
25. <transformers>
26. <transformer
27. implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
28. <resource>META-INF/spring.handlers</resource>
29. </transformer>
30. <transformer
31. implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
32. <mainClass>com.fxc.rpc.impl.member.MemberProvider</mainClass>
33. </transformer>
34. <transformer
35. implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
36. <resource>META-INF/spring.schemas</resource>
37. </transformer>
38. </transformers>
39. </configuration>
40. </execution>
41. </executions>
42. </plugin>
43. </plugins>
44. </build>

3、  Dubbo配置

1)      编写dubbo消费端配置文件

Resources文件夹下面新建dubbo-client.xml

<dubbo:registry > : dubbo服务注册配置

<dubbo:application> : dubbo应用名称

<dubbo:reference> : 生成远程服务代理，可以像使用本地bean一样使用远程服务。

<dubbo:consumer> : 消费端的一些缺省配置。



2)      Dubbo初始化工具类编写DubboInit。

1)      新建util包，新建DubboInit类，代码如下

1. package util;
3. import org.springframework.context.ApplicationContext;
4. import org.springframework.context.support.ClassPathXmlApplicationContext;
6. */\*\**
7. *\* Created by wangdingjin on 2016/9/2.*
8. *\*/*
9. public class DubboInit {
10. private static DubboInit init = null;
11. private DubboInit(){}
12. private static ApplicationContext context;
13. public synchronized static DubboInit getInstance(){
14. if(init == null){
15. init = new DubboInit();
16. }
17. return init;
18. }
20. public static void initApplicationContext(){
21. context = new ClassPathXmlApplicationContext("classpath:/dubbo-client.xml");
22. if(context==null)
23. {
24. throw new IllegalArgumentException("Load dubbo-client.xml fail");
25. }
26. }
28. public Object getBean(String beanName) {
29. return context.getBean(beanName);
30. }
31. }

3)      编写测试代码（此处建议现行编写Junit测试代码，测试通过之后，再编写jmeter所需的测试代码）

1)      编写junit测试代码

在test/java中新建package：directRegisterInfoService(根据自己喜好命名即可，我的喜好是测试一个服务接口就新建一个包，因为一个服务里面有很多方法，我们测试这些方法的时候，会针对每一个方法写一个测试类)

此处用例为测试取号列表（queryTakeRegNo），新建TqueryTakeRegNo类，编写测试代码如下：

1. package directRegisterInfoService;
3. import com.saohuobang.web.platform.bjmedicalcard.po.DirectRegisterInfo;
4. import com.saohuobang.web.platform.bjmedicalcard.service.DirectRegisterInfoInf;
5. import org.joda.time.DateTime;
6. import org.junit.Before;
7. import org.junit.Test;
8. import org.springframework.beans.factory.annotation.Autowired;
9. import util.DubboInit;
11. import java.util.ArrayList;
12. import java.util.HashMap;
13. import java.util.List;
14. import java.util.Map;
16. */\*\**
17. *\* Created by Administrator on 2016/9/2.*
18. *\*/*
19. public class TqueryTakeRegNo {
21. */\**
22. *\*定义DirectRegisterInfoInf的变量*
23. *\*/*
24. @Autowired
25. private DirectRegisterInfoInf directRegisterInfoInf;
27. */\**
28. *\*初始化directRegisterInfoService服务*
29. *\*/*
30. @Before
31. public void setupTest() {
32. DubboInit init = DubboInit.getInstance();
33. init.initApplicationContext();
34. directRegisterInfoInf = (DirectRegisterInfoInf)init.getBean("directRegisterInfoService");
35. }
37. */\**
38. *\*测试代码*
39. *\*/*
40. @Test
41. public void runTest() {
42. List<DirectRegisterInfo> list = new ArrayList<DirectRegisterInfo>();
43. try {
44. Map map = new HashMap();
45. map.put("dateTime", new DateTime().toString("yyyy-MM-dd 00:00:00"));*// TODO*
46. *// map.put("dateTime","2016-01-26 00:00:00");*
47. map.put("hosCode", "T113411");
48. map.put("cardNo", "98000001004327");
49. list = directRegisterInfoInf.getTakeNoList(map);
50. System.out.println("\n\n\n\n================"+list.get(0).getDeptname() + "==================\n\n\n\n");
51. } catch (Exception e) {
52. e.printStackTrace();
53. }
54. System.out.println();
55. System.out.println();
56. }
58. }

2)      启用debug调试，确认junit测试是否通过。（此处不做详解）

3)      Junit测试通过之后，编写jmeter需要的测试代码。

Main/java新建directRegisterInfoService包，新建LqueryTakeRegNo测试类，该测试类需要extendsAbstractJavaSamplerClient,

主要实现setupTest()：初始化操作

getDefaultParameters()：设置默认的jmeter的Parameters

SetValues(JavaSamplerContextarg0)：根据jmeter填写的参数设置变量值

runTest(JavaSamplerContextjavaSamplerContext)：运行测试代码。teardownTest(JavaSamplerContext arg0)：测试收尾操作。

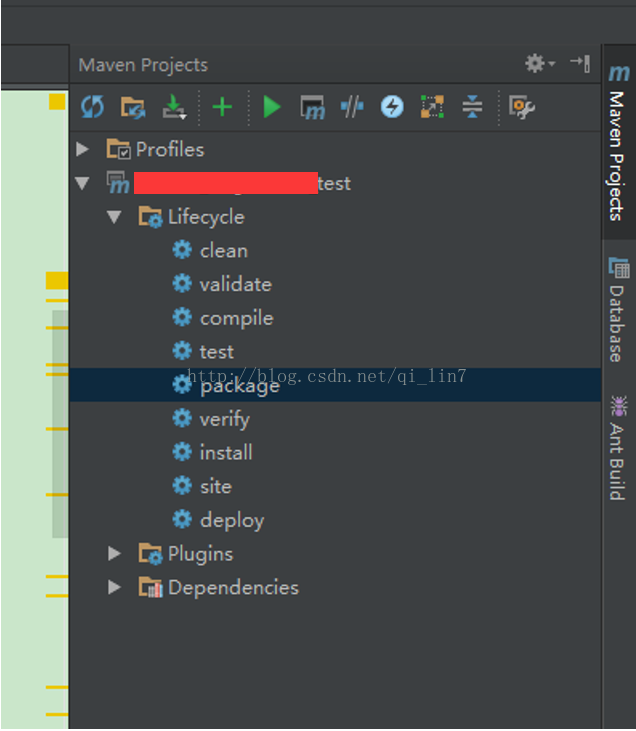
用例代码如下（供参考）：

1. package directRegisterInfoService;
3. import com.saohuobang.web.platform.bjmedicalcard.po.DirectRegisterInfo;
4. import com.saohuobang.web.platform.bjmedicalcard.service.DirectRegisterInfoInf;
5. import org.apache.jmeter.config.Arguments;
6. import org.apache.jmeter.protocol.java.sampler.AbstractJavaSamplerClient;
7. import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;
8. import org.apache.jmeter.samplers.SampleResult;
9. import org.joda.time.DateTime;
10. import org.springframework.beans.factory.annotation.Autowired;
11. import util.DubboInit;
13. import java.util.ArrayList;
14. import java.util.HashMap;
15. import java.util.List;
16. import java.util.Map;
18. */\*\**
19. *\* Created by Administrator on 2016/9/2.*
20. *\*/*
21. public class LqueryTakeRegNo extends AbstractJavaSamplerClient {
23. *//需要测试的接口*
24. @Autowired
25. private DirectRegisterInfoInf directRegisterInfoInf;
27. private long start = 0;*//记录测试开始时间；*
28. private long end = 0;*//记录测试结束时间；*
29. private String hosCode = "T113411";
30. private String cardNo = "98000001004327";

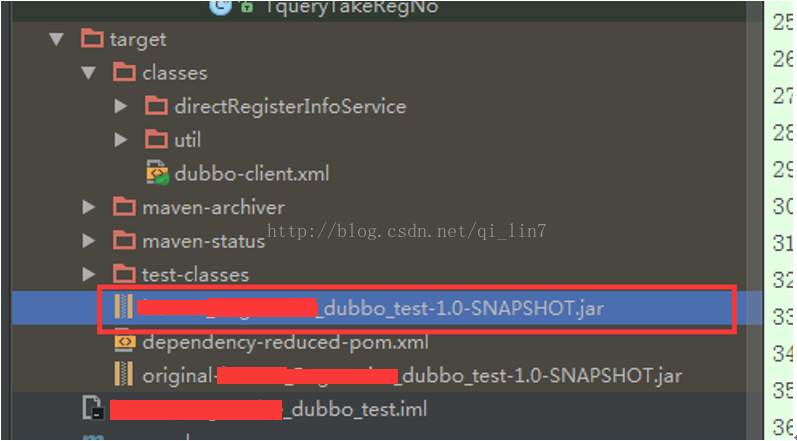
33. *//初始化操作*
34. @Override
35. public void setupTest(JavaSamplerContext arg0) {
36. DubboInit init = DubboInit.getInstance();
37. init.initApplicationContext();
38. directRegisterInfoInf = (DirectRegisterInfoInf) init.getBean("directRegisterInfoService");
39. }
41. */\*\**
42. *\**
43. *\* 设置默认值*
44. *\**
45. *\* @return*
46. *\*/*
47. public Arguments getDefaultParameters() {
48. Arguments params = new Arguments();
49. params.addArgument("hosCode", hosCode);
50. params.addArgument("cardNo", cardNo);
51. return params;
52. }
54. */\*\**
55. *\* 获取jmeter输入的参数值*
56. *\**
57. *\* @return*
58. *\*/*
59. public void setValues(JavaSamplerContext arg0) {
60. hosCode = arg0.getParameter("hosCode",hosCode);
61. cardNo = arg0.getParameter("cardNo",cardNo);
62. }
64. public SampleResult runTest(JavaSamplerContext javaSamplerContext) {
65. SampleResult sr = new SampleResult();
66. setValues(javaSamplerContext);
67. sr.sampleStart();
68. start = System.currentTimeMillis();
69. try {
70. List<DirectRegisterInfo> list = new ArrayList<DirectRegisterInfo>();
72. Map map = new HashMap();
73. map.put("dateTime", new DateTime().toString("yyyy-MM-dd 00:00:00"));*// TODO*
74. *// map.put("dateTime","2016-01-26 00:00:00");*
75. map.put("hosCode", hosCode);
76. map.put("cardNo", cardNo);
77. list = directRegisterInfoInf.getTakeNoList(map);
78. if (list != null && list.size() != 0) {
79. sr.setSuccessful(true);
80. sr.setResponseData("医院：" + list.get(0).getHospname(), null);
81. sr.setDataType(SampleResult.TEXT);
82. } else {
83. sr.setSuccessful(false);
84. getLogger().error("LqueryTakeRegNo response is null");
85. }
86. } catch (Exception e) {
87. getLogger().error("LqueryTakeRegNo response error : " + e.getMessage());
88. sr.setSuccessful(false);
89. } finally {
90. sr.sampleEnd();
91. }
92. return sr;
93. }
95. @Override
96. public void teardownTest(JavaSamplerContext arg0) {
97. end = System.currentTimeMillis();
98. getLogger().info(" cost time: " + (end - start) + "ms");
99. }
100. }

4)      生成jar包：

Idea右侧->mavenproject->Lifecycle->package即可生成jar包（\*\*\*\*\*\*service\_dubbo\_test-1.0-SNAPSHOT.jar）



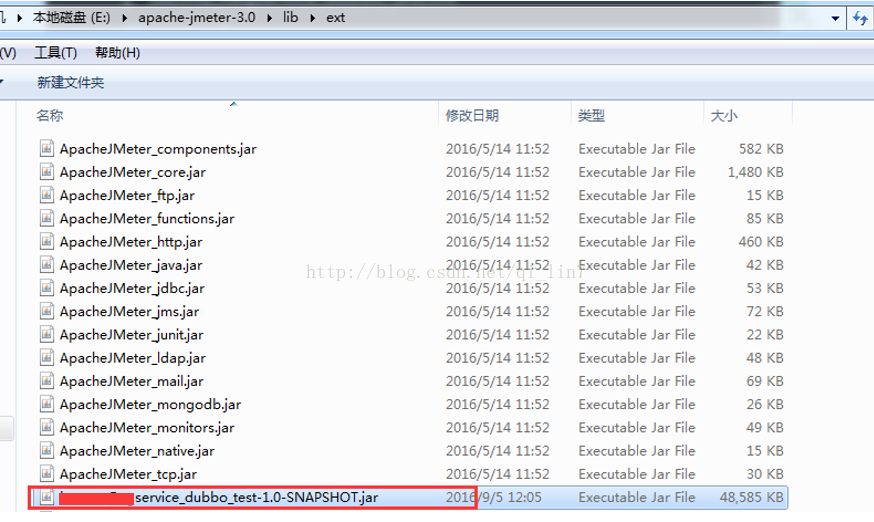
Jar包位置如下：



三、Jmeter运行测试用例

1、  导入jar包

将\*\*\*\*\*\*service\_dubbo\_test-1.0-SNAPSHOT.jar放到\apache-jmeter-3.0\lib\ext目录下。

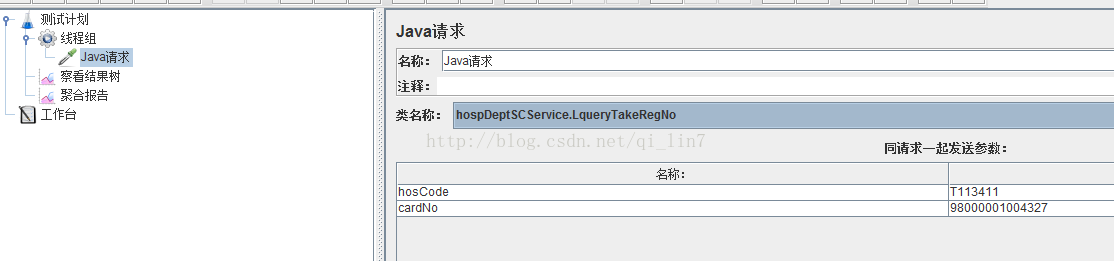


2、  编写jmeter测试脚本

新建线程组->java请求（右键sampler选择java请求）；

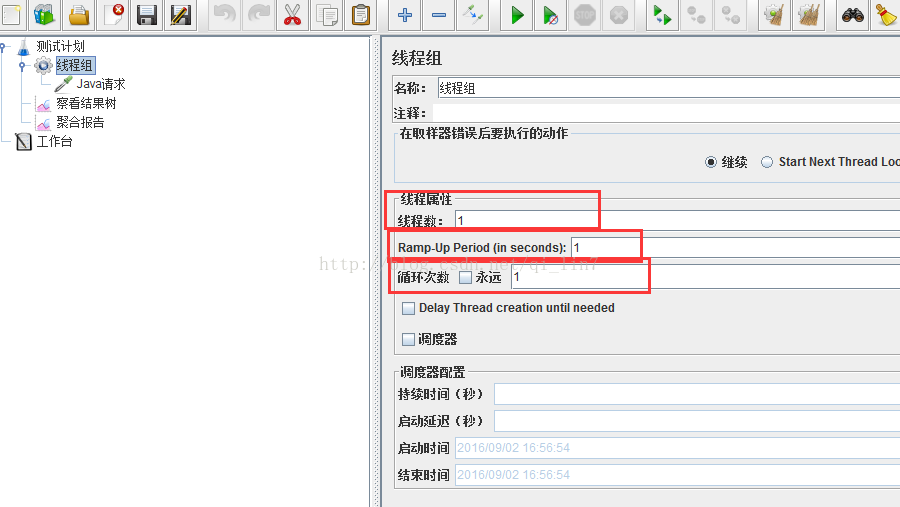
新建查看结果树（做性能测试的时候，选择）、聚合报告等组件。

类名称选择我们之前编写的测试类即可。



进行性能测试的时候，填写线程数、启动线程时间、循环次数等。

进行性能测试的时候，查看结果树务必勾选仅显示日志错误。



后面Jmeter运行用例，分析报告此文不做详解