How to Create A Java Class Performance Test Using JMeters AbstractJavaSamplerClient

JMeter is a very powerfull tools to do performance testing. In this tutorial, im going to simulate a heavy load on a java object to test its strength and performance testing. My java class is very simple, only an ordinary java class to do inserts into mysql database using Hibernate framework.

first is a sample database and a table

```
CREATE DATABASE test;
USE test;
CREATE
    TABLE student
    (
        id INT NOT NULL AUTO_INCREMENT,
        studentname VARCHAR(60) NOT NULL,
        PRIMARY KEY (id)
    );
```

a java class and xml as representation for database's table

```
package com.edw.bean;

public class Student implements java.io.Serializable
{

    private Integer id;
    private String studentname;

    public Student() {
    }

    public Student(String studentname) {
       this.studentname = studentname;
    }
```

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.edw.bean.Student" table="student" catalog="test">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="identity" />
        </id>
        <property name="studentname" type="string">
            <column name="studentname" length="60" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

---

this is my Hibernate main configuration xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/test</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">xxxxxx</property>
    <property name="hibernate.connection.autocommit">true</property>
    <mapping resource="com/edw/bean/Student.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

---

and my java class to load Hibernate’s configuration

```
package com.edw.hbm;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * @author edw
 */
public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed."+ ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

This is my tested java class, it has to extends AbstractJavaSamplerClient so the jmeter application can test it

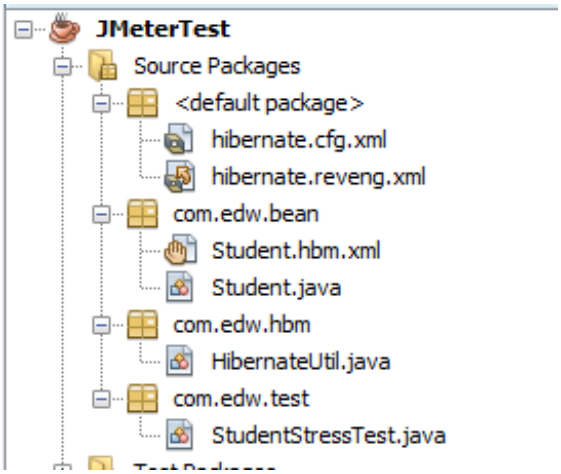
```
package com.edw.test;

import com.edw.bean.Student;
import com.edw.hbm.HibernateUtil;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import org.apache.jmeter.config.Arguments;
import org.apache.jmeter.protocol.java.sampler.AbstractJavaSamplerClient;
import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;
import org.apache.jmeter.samplers.SampleResult;
import org.apache.jmeter.threads.JMeterContextService;
import org.apache.jmeter.threads.JMeterVariables;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

/**
 *  com.edw.test.StudentStressTest
 *
 *  @author edw
 */
public class StudentStressTest extends AbstractJavaSamplerClient {

    private Map<String, String> mapParams = new HashMap<String, String>();
```

```
    private SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

    public StudentStressTest() {
        super();
    }

    @Override
    public void setupTest(JavaSamplerContext context) {
        for (Iterator<String> it = context.getParameterNamesIterator();
it.hasNext();) {
            String paramName =  it.next();
            mapParams.put(paramName, context.getParameter(paramName));
        }
    }

    public SampleResult runTest(JavaSamplerContext context) {
        SampleResult result = new SampleResult();

        try {

            JMeterVariables vars =
JMeterContextService.getContext().getVariables();
            vars.put("demo", "demoVariableContent");

            result.sampleStart();

            Student student = new Student();
            student.setStudentname(mapParams.get("name")+" "+new Date().getTime());
            Session session = sessionFactory.openSession();
            session.save(student);
            session.flush();
            session.close();

            result.sampleEnd();


            result.setSuccessful(true);
            result.setSampleLabel("SUCCESS: " + student.getStudentname());

        } catch (Throwable e) {
            result.sampleEnd();
            result.setSampleLabel("FAILED: '" + e.getMessage() + "' || " +
e.toString());
            result.setSuccessful(false);

            e.printStackTrace();
            System.out.println("\n\n\n");
        }

        return result;
    }

    @Override
    public Arguments getDefaultParameters() {

        Arguments params = new Arguments();
```

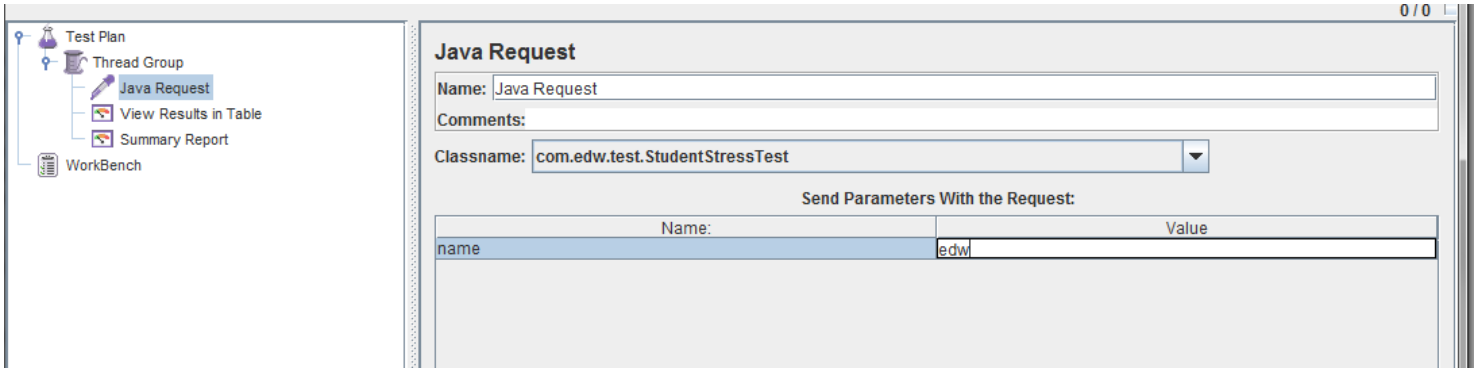
```
        params.addArgument("name", "edw");

        return params;
    }
}
```

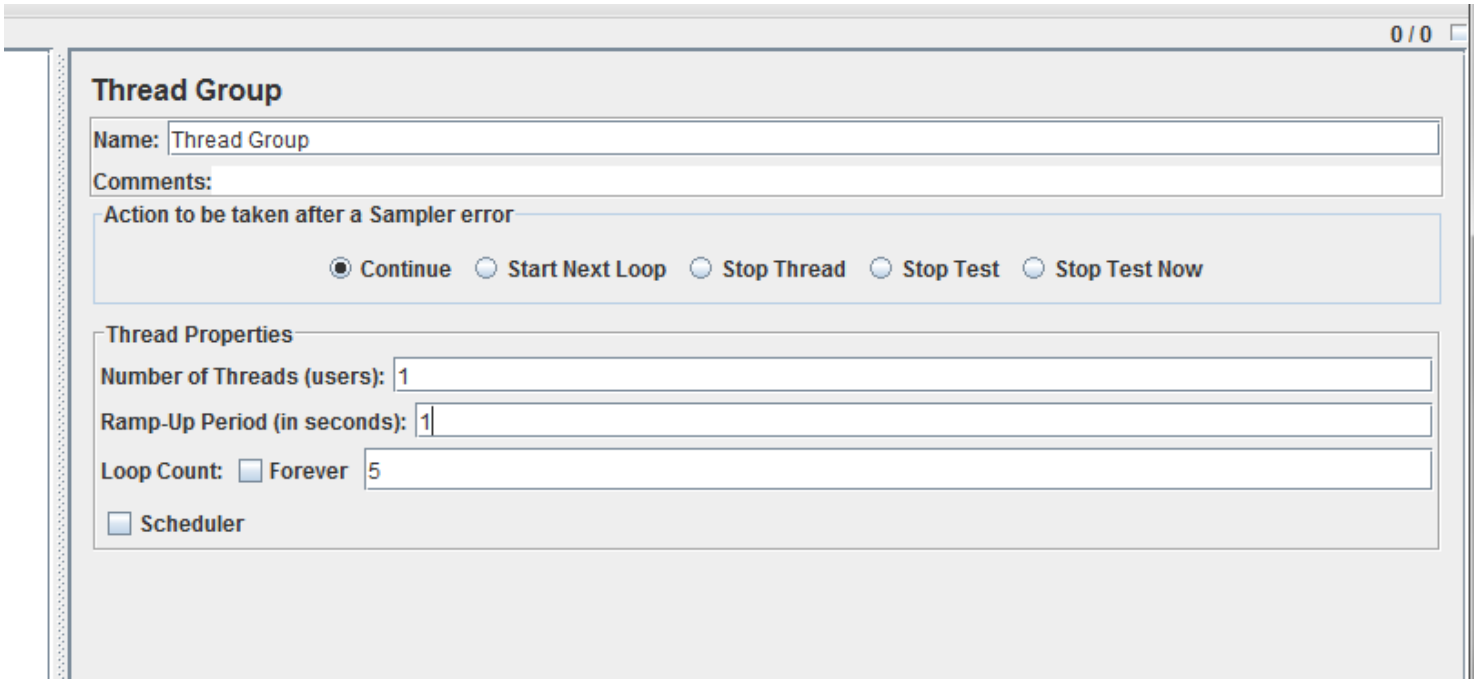
Next is build your project into jar and put it into jmeter's path (\lib\ext), dont forget to copy your application library such as hibernate3.jar.
If you open your JMeter ui, you can see your java class on Add > Sampler > Java Request. You can see the results sumamry on tab summary report.

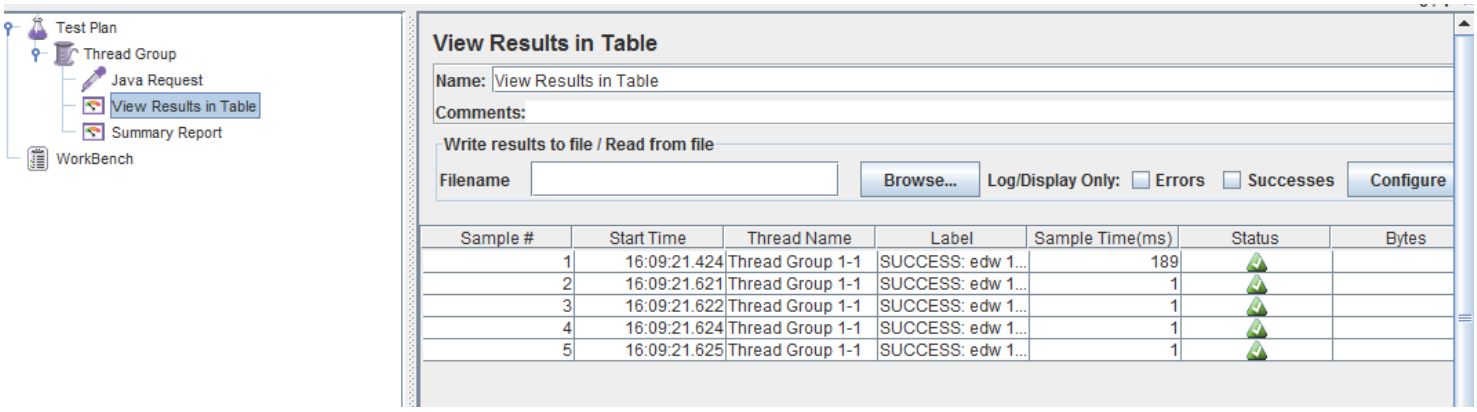
This is my Netbeans project structure, and my jmeter configuration.

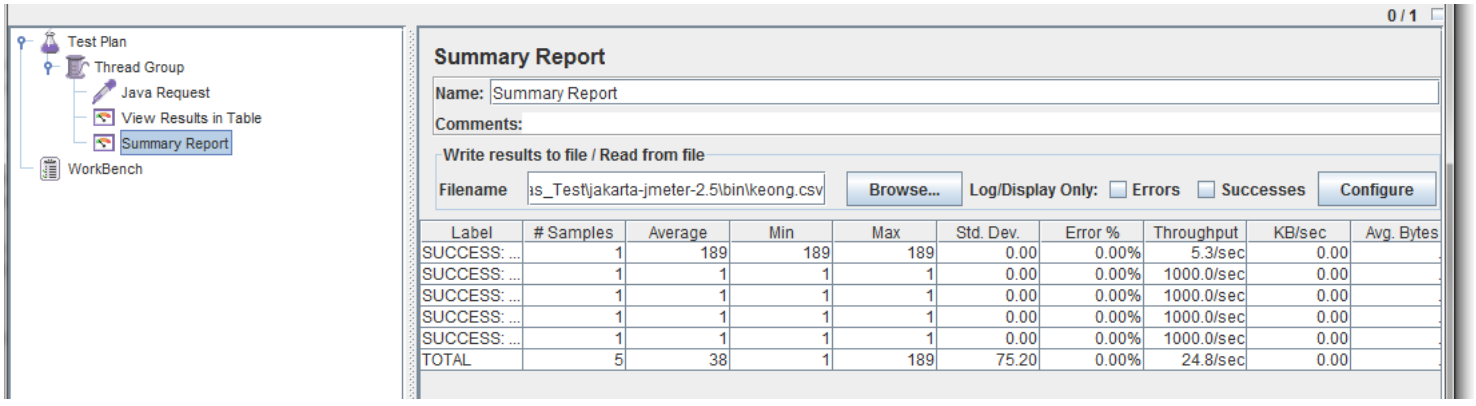


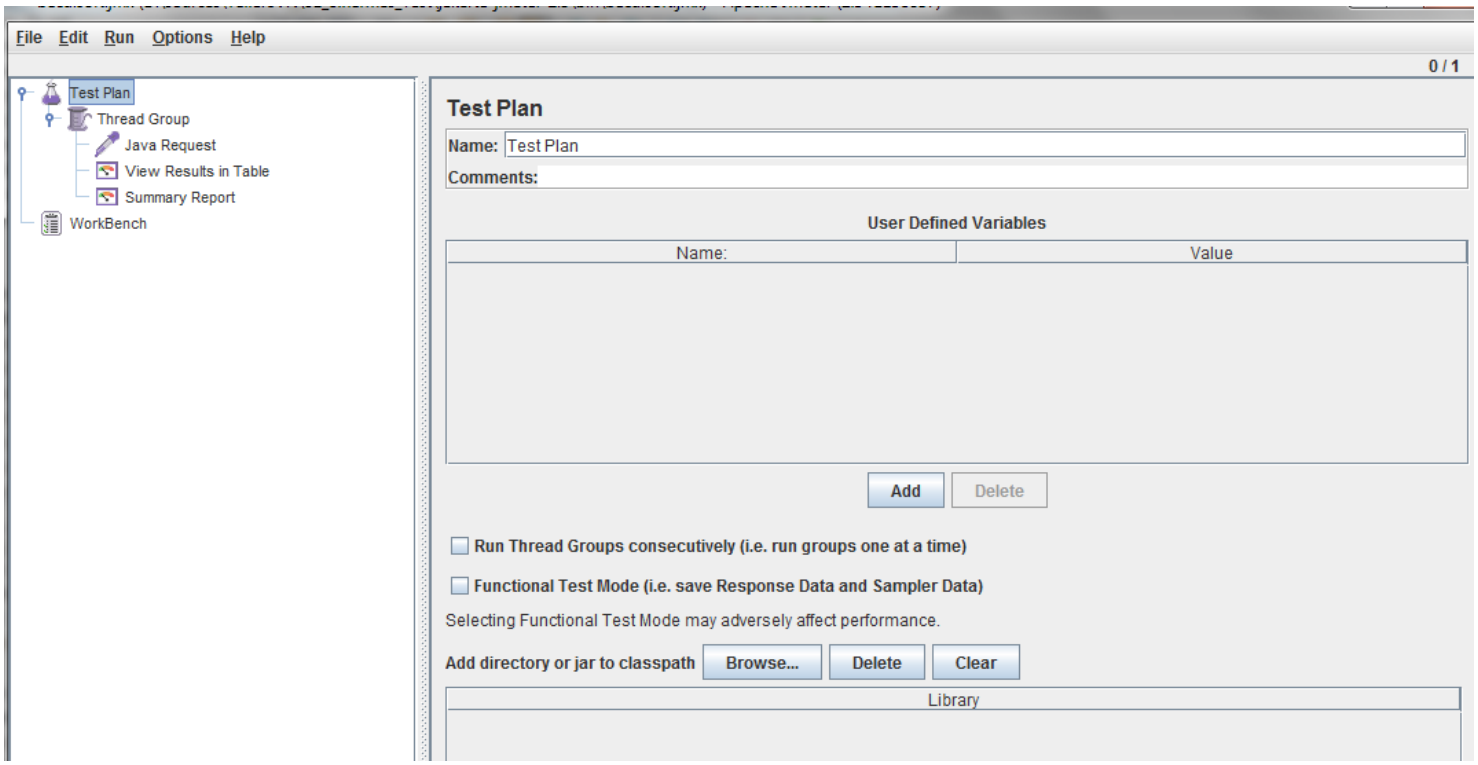

Test Packages
Libraries
ApacheJMeter_core.jar
ApacheJMeter_java.jar
Hibernate - antlr-2.7.6.jar
Hibernate - asm.jar
Hibernate - asm-attrs.jar
Hibernate - cglib-2.1.3.jar
Hibernate - commons-collections-2.1.1.jar
Hibernate - commons-logging-1.1.jar
Hibernate - dom4j-1.6.1.jar
Hibernate - ehcache-1.2.3.jar
Hibernate - jdbc2_0-stdext.jar
Hibernate - jta.jar
Hibernate - hibernate3.jar
Hibernate - hibernate-tools.jar
Hibernate - hibernate-annotations.jar
Hibernate - hibernate-commons-annotations.jar
Hibernate - hibernate-entitymanager.jar
Hibernate - javassist.jar
Persistence - ejb3-persistence.jar
mysql-connector-java-5.1.6-bin.jar
JDK 1.6 (Default)
Test Libraries


0 / 0
Test Plan
Thread Group
Java Request
View Results in Table
Summary Report
WorkBench
Java Request
Name: Java Request
Comments:
Classname: com.edw.test.StudentStressTest
Send Parameters With the Request:
Name:
Value
name
edw


0 / 0
Thread Group
Name: Thread Group
Comments:
Action to be taken after a Sampler error
Continue
Start Next Loop
Stop Thread
Stop Test
Stop Test Now
Thread Properties
Number of Threads (users): 1
Ramp-Up Period (in seconds): 1
Loop Count: Forever 5
Scheduler


Test Plan
Thread Group
Java Request
View Results in Table
Summary Report
WorkBench
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Sample # Start Time Thread Name Label Sample Time(ms) Status Bytes
1 16:09:21.424 Thread Group 1-1 SUCCESS: edw 1... 189
2 16:09:21.621 Thread Group 1-1 SUCCESS: edw 1... 1
3 16:09:21.622 Thread Group 1-1 SUCCESS: edw 1... 1
4 16:09:21.624 Thread Group 1-1 SUCCESS: edw 1... 1
5 16:09:21.625 Thread Group 1-1 SUCCESS: edw 1... 1


0 / 1
Test Plan
Thread Group
Java Request
View Results in Table
Summary Report
WorkBench
Summary Report
Name: Summary Report
Comments:
Write results to file / Read from file
Filename ...s_Test\jakarta-jmeter-2.5\bin\keong.csv
Browse...
Log/Display Only: Errors Successes
Configure
Label # Samples Average Min Max Std. Dev. Error % Throughput KB/sec Avg. Bytes
SUCCESS: ... 1 189 189 189 0.00 0.00% 5.3/sec 0.00
SUCCESS: ... 1 1 1 1 0.00 0.00% 1000.0/sec 0.00
SUCCESS: ... 1 1 1 1 0.00 0.00% 1000.0/sec 0.00
SUCCESS: ... 1 1 1 1 0.00 0.00% 1000.0/sec 0.00
SUCCESS: ... 1 1 1 1 0.00 0.00% 1000.0/sec 0.00
TOTAL 5 38 1 189 75.20 0.00% 24.8/sec 0.00


File Edit Run Options Help
0 / 1
Test Plan
Thread Group
Java Request
View Results in Table
Summary Report
WorkBench
Test Plan
Name: Test Plan
Comments:
User Defined Variables
Name:
Value
Add
Delete
Run Thread Groups consecutively (i.e. run groups one at a time)
Functional Test Mode (i.e. save Response Data and Sampler Data)
Selecting Functional Test Mode may adversely affect performance.
Add directory or jar to classpath
Browse...
Delete
Clear
Library