<http://blog.csdn.net/wen294299195/article/details/39964971>

ubuntu下使用java、javac命令行编译工程

原创 2014年10月10日 19:12:48

 标签：

 [java](http://so.csdn.net/so/search/s.do?q=java&t=blog) /

 [javac](http://so.csdn.net/so/search/s.do?q=javac&t=blog) /

 [shell](http://so.csdn.net/so/search/s.do?q=shell&t=blog)

 10688

一、java和javac命令行  
（1）javac命令行  
javac.exe用于编译java源文件，生成.class文件

语法：javac [option] source

常用的[option]选项：

A、-classpath <路径>（-cp缩写）：指定要使用的类路径或要使用的jar包的路径(jar文件、zip文件（里面都是错啦搜索文件）)，使用后

会覆盖CLASSPATH的设定

B、-d <路径> ：指定源文件编译后生成.class文件的存放位置

（2）java命令行  
java.exe用于运行javac编译后生成的.class文件

语法：java [option] classname [arguments]

常用的[option]选项：

A、-classpath <路径>（-cp缩写）：指定要使用的类路径或要使用的jar包的路径(jar文件、zip文件（里面都是错啦搜索文件）)，使用后

会覆盖CLASSPATH的设定

B、[arguments]：传递给main函数的参数

二、例子  
（1）单独一个java程序  
Demo.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. public class Demo{

2. public static void main(String args[]){

3. System.out.println("This is a test");

4. }

5. }

javac Demo.java-----------在当前目录下生成Demo.class文件

java Demo-------------------执行Demo.class

也可以通过-d选项指定生成.class文件的位置，如：

javac -d .. Demo.java ，则就会在当前目录的上一级目录上生成Demo.class文件，那么执行的时候，就要指定class的路径

java -cp .. Demo，表示让java在上一级目录搜索类文件

（2）一个带包的java程序

Demo.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.demo;

2. public class Demo{

3. public static void main(String args[]){

4. System.out.println("This is a test");

5. }

6. }

同样，现编译文件：

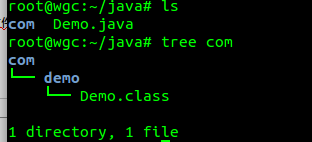
javac Demo.java-----------在当前目录下生成Demo.class文件

java Demo-------------------执行Demo.class

也可以通过-d选项指定生成.class文件的位置，如：

javac -d . Demo.java-----------按照com.demo的包路径保存.class文件

则会在当前目录下生成一个com/demo/Demo.class文件，如下图：



那么在执行的时候，也要指定类文件的路径：

javac com/demo/Demo或者javac com.demo.Demo

（3）同一个包下，一个类调用另一个类  
Tom.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.demo;

2. public class Tom{

3. public String getMyname(){

4. return "This is Tom!";

5. }

6. }

Friend.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.demo;

2.

3. import com.demo.Tom;

4.

5. public class Friend{

6. public static void main(String args[]){

7. Tom tom = new Tom();

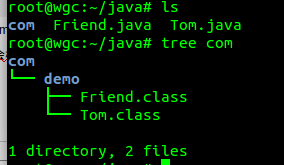
8. System.out.println("hello "+tom.getMyname());

9. }

10. }

由于Friend类条用了Tom类，首先要现编译Tom.java:javac -d . Tom.java，然会会按照包结构生成com/demo/Tom.class文件。

接下来编译Friend.java文件：javac -cp . -d . Friend.java，javac会根据，import com.demo.Tom;这个路径到 -cp 指示的 "."当前目录下寻找，然后按照包结构生成com/demo/Friend.class，见下图：



运行：java com.demo.Friend

（4）一个包下的类条用另一个包下的类

Tom.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.demo1;

2. public class Tom{

3. public String getMyname(){

4. return "This is Tom!";

5. }

6. }

Friend.java

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.demo2;

2.

3. import com.demo1.Tom;

4.

5. public class Friend{

6. public static void main(String args[]){

7. Tom tom = new Tom();

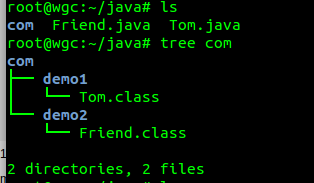
8. System.out.println("hello "+tom.getMyname());

9. }

10. }

还是一样，现编译Tom.java，javac -d . Tom.java，生成com/demo1/Tom.class；

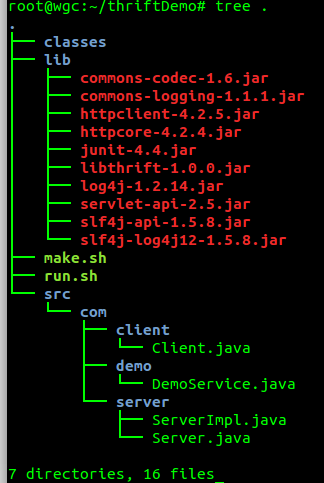
然后编译Friend.java，javac -cp . -d . Friend.java，生成com/demo2/Friend.class；



运行：java com.demo2.Friend;

（5）更接近项目的java工程  
一般，一个工程下面都会有lib（引入的必要jar包），classes（保存.class文件），src（.java源代码）三个文件夹。

java文件夹的目录结构如下，文件之间的关系是Client.java调用了DemoService.java的方法；ServerImpl.java调用了DemoService.java的方法；Server.java调用了ServerImpl.java和DemoService.java的方法；



下面的文件，我只显示他们引用的包和类。

src/com/client/Client.java:

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.client;

2. import org.apache.thrift.TException;

3. import org.apache.thrift.protocol.TBinaryProtocol;

4. import org.apache.thrift.protocol.TProtocol;

5. import org.apache.thrift.transport.TSocket;

6. import org.apache.thrift.transport.TTransport;

7.

8. import com.demo.DemoService;

9.

10.

11. public class Client {

12.

13. .................................................................................

14.

15. }

src/com/demo/DemoService.java:  
**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. /\*\*

2. \* Autogenerated by Thrift Compiler (0.9.0)

3. \*

4. \* DO NOT EDIT UNLESS YOU ARE SURE THAT YOU KNOW WHAT YOU ARE DOING

5. \* @generated

6. \*/

7.

8. package com.demo;

9. import org.apache.thrift.scheme.IScheme;

10. import org.apache.thrift.scheme.SchemeFactory;

11. import org.apache.thrift.scheme.StandardScheme;

12.

13. import org.apache.thrift.scheme.TupleScheme;

14. import org.apache.thrift.protocol.TTupleProtocol;

15. import org.apache.thrift.protocol.TProtocolException;

16. import org.apache.thrift.EncodingUtils;

17. import org.apache.thrift.TException;

18. import java.util.List;

19. import java.util.ArrayList;

20. import java.util.Map;

21. import java.util.HashMap;

22. import java.util.EnumMap;

23. import java.util.Set;

24. import java.util.HashSet;

25. import java.util.EnumSet;

26. import java.util.Collections;

27. import java.util.BitSet;

28. import java.nio.ByteBuffer;

29. import java.util.Arrays;

30. import org.slf4j.Logger;

31. import org.slf4j.LoggerFactory;

32.

33. public class DemoService {

34. .................................................................

35. }

src/com/server/Server.java:

**[java]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. package com.server;

2. import org.apache.thrift.TProcessor;

3. import org.apache.thrift.protocol.TBinaryProtocol;

4. import org.apache.thrift.server.TServer;

5. import org.apache.thrift.server.TSimpleServer;

6. import org.apache.thrift.transport.TServerSocket;

7. import org.apache.thrift.transport.TTransportException;

8.

9. import com.demo.DemoService;

10. import com.server.ServerImpl;

11.

12. public class Server {

13.

14. ..................................................................................

15.

16. }

make.sh脚本如下：

**[python]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. #!/bin/bash

2.

3. # Get current position

4. TOP\_DIR=$(pwd)

5.

6. # Add all necessary jars

7. LIBPATH=lib/commons-codec-1.6.jar:lib/commons-logging-1.1.1.jar:lib/httpclient-4.2.5.jar:lib/httpcore-4.2.4.jar:lib/junit-4.4.jar:lib/libthrift-1.0.0.jar:lib/log4j-1.2.14.jar:lib/servlet-api-2.5.jar:lib/slf4j-api-1.5.8.jar:lib/slf4j-log4j12-1.5.8.jar

8.

9. #compile java file

10. javac -cp $LIBPATH src/com/demo/DemoService.java -d ./classes/.

11. javac -cp $TOP\_DIR/classes:$LIBPATH src/com/server/ServerImpl.java -d ./classes/.

12. javac -cp $TOP\_DIR/classes:$LIBPATH src/com/server/Server.java -d ./classes/.

13. javac -cp $TOP\_DIR/classes:$LIBPATH src/com/client/Client.java -d ./classes/.

run.sh脚本如下：

**[python]** [view plain](http://blog.csdn.net/wen294299195/article/details/39964971#) [copy](http://blog.csdn.net/wen294299195/article/details/39964971#)

1. #!/bin/bash

2.

3. #Get client/server

4. SIDE=$1

5.

6. # Get current position

7. TOP\_DIR=$(pwd)

8.

9. # Add all necessary jars

10. LIBPATH=lib/commons-codec-1.6.jar:lib/commons-logging-1.1.1.jar:lib/httpclient-4.2.5.jar:lib/httpcore-4.2.4.jar:lib/junit-4.4.jar:lib/libthrift-1.0.0.jar:lib/log4j-1.2.14.jar:lib/servlet-api-2.5.jar:lib/slf4j-api-1.5.8.jar:lib/slf4j-log4j12-1.5.8.jar

11.

12. #run program

13. if [ $SIDE == "server" ];then

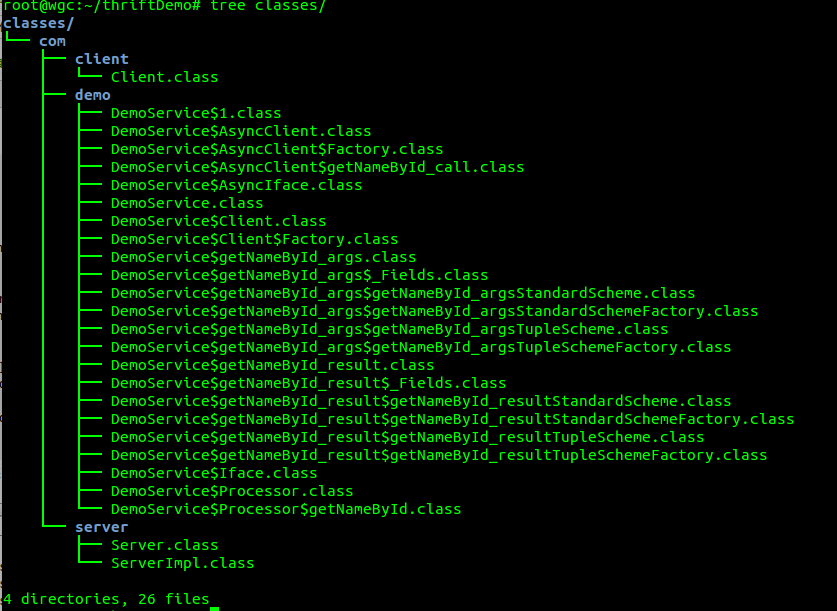
14. java -cp $TOP\_DIR/classes:$LIBPATH com/server/Server

15. else

16. java -cp $TOP\_DIR/classes:$LIBPATH com/client/Client

17. fi

进入thriftDemo文件夹下，执行./make.sh脚本，在classes文件夹下生成的文件如下：



参考：wenku.baidu.com/view/f4c19dbc65ce0508763213c6.html