**1.纯java代码引用文件流的方式（缺点是分割的考虑比较麻烦）**

**[java]** [view plain](http://blog.csdn.net/runming56/article/details/35985479) [copy](http://blog.csdn.net/runming56/article/details/35985479)

[在CODE上查看代码片](https://code.csdn.net/snippets/411209)

1. **package** com.unmi.db;
3. **import** java.io.FileInputStream;
4. **import** java.io.InputStream;
5. **import** java.sql.Connection;
6. **import** java.sql.Statement;
7. **import** java.util.ArrayList;
8. **import** java.util.Arrays;
9. **import** java.util.List;
11. /\*\*
12. \* 读取 SQL 脚本并执行
13. \* @author Unmi
14. \*/
15. **public** **class** SqlFileExecutor {
17. /\*\*
18. \* 读取 SQL 文件，获取 SQL 语句
19. \* @param sqlFile SQL 脚本文件
20. \* @return List<sql> 返回所有 SQL 语句的 List
21. \* @throws Exception
22. \*/
23. **private** List<String> loadSql(String sqlFile) **throws** Exception {
24. List<String> sqlList = **new** ArrayList<String>();
26. **try** {
27. InputStream sqlFileIn = **new** FileInputStream(sqlFile);
29. StringBuffer sqlSb = **new** StringBuffer();
30. **byte**[] buff = **new** **byte**[1024];
31. **int** byteRead = 0;
32. **while** ((byteRead = sqlFileIn.read(buff)) != -1) {
33. sqlSb.append(**new** String(buff, 0, byteRead));
34. }
36. // Windows 下换行是 /r/n, Linux 下是 /n
37. String[] sqlArr = sqlSb.toString().split("(;//s\*//r//n)|(;//s\*//n)");
38. **for** (**int** i = 0; i < sqlArr.length; i++) {
39. String sql = sqlArr[i].replaceAll("--.\*", "").trim();
40. **if** (!sql.equals("")) {
41. sqlList.add(sql);
42. }
43. }
44. **return** sqlList;
45. } **catch** (Exception ex) {
46. **throw** **new** Exception(ex.getMessage());
47. }
48. }
50. /\*\*
51. \* 传入连接来执行 SQL 脚本文件，这样可与其外的数据库操作同处一个事物中
52. \* @param conn 传入数据库连接
53. \* @param sqlFile SQL 脚本文件
54. \* @throws Exception
55. \*/
56. **public** **void** execute(Connection conn, String sqlFile) **throws** Exception {
57. Statement stmt = **null**;
58. List<String> sqlList = loadSql(sqlFile);
59. stmt = conn.createStatement();
60. **for** (String sql : sqlList) {
61. stmt.addBatch(sql);
62. }
63. **int**[] rows = stmt.executeBatch();
64. System.out.println("Row count:" + Arrays.toString(rows));
65. }
67. /\*\*
68. \* 自建连接，独立事物中执行 SQL 文件
69. \* @param sqlFile SQL 脚本文件
70. \* @throws Exception
71. \*/
72. **public** **void** execute(String sqlFile) **throws** Exception {
73. Connection conn = DBCenter.getConnection();
74. Statement stmt = **null**;
75. List<String> sqlList = loadSql(sqlFile);
76. **try** {
77. conn.setAutoCommit(**false**);
78. stmt = conn.createStatement();
79. **for** (String sql : sqlList) {
80. stmt.addBatch(sql);
81. }
82. **int**[] rows = stmt.executeBatch();
83. System.out.println("Row count:" + Arrays.toString(rows));
84. DBCenter.commit(conn);
85. } **catch** (Exception ex) {
86. DBCenter.rollback(conn);
87. **throw** ex;
88. } **finally** {
89. DBCenter.close(**null**, stmt, conn);
90. }
91. }
93. **public** **static** **void** main(String[] args) **throws** Exception {
94. List<String> sqlList = **new** SqlFileExecutor().loadSql(args[0]);
95. System.out.println("size:" + sqlList.size());
96. **for** (String sql : sqlList) {
97. System.out.println(sql);
98. }
99. }
100. }

2.ibatis方式执行脚本

首先引入ibatis-common-2.jar包

import com.ibatis.common.jdbc.ScriptRunner;  
import com.ibatis.common.resources.Resources;

jpetstore测试代码如下：

**[java]** [view plain](http://blog.csdn.net/runming56/article/details/35985479) [copy](http://blog.csdn.net/runming56/article/details/35985479)

[在CODE上查看代码片](https://code.csdn.net/snippets/411209)

1. **package** com.ibatis.jpetstore.test;
3. **import** java.sql.DriverManager;
4. **import** java.util.Properties;
6. **import** com.ibatis.common.jdbc.ScriptRunner;
7. **import** com.ibatis.common.resources.Resources;
8. **import** com.mysql.jdbc.Connection;
10. **public** **class** DBTestSQL {

13. **public** **static** **void** main(String[] args) {
15. **try** {
16. Properties props = Resources.getResourceAsProperties("properties/database.properties");
17. String url = props.getProperty("url");
18. String driver = props.getProperty("driver");
19. String username = props.getProperty("username");
20. String password = props.getProperty("password");
21. System.out.println(url);
22. **if**(url.equals("jdbc:mysql://localhost:3306/jpetstore1")) {
23. Class.forName(driver).newInstance();
24. Connection conn = (Connection) DriverManager.getConnection(url, username, password);
25. ScriptRunner runner = **new** ScriptRunner(conn, **false**, **false**);
26. runner.setErrorLogWriter(**null**);
27. runner.setLogWriter(**null**);
28. runner.runScript(Resources.getResourceAsReader("ddl/mysql/jpetstore-mysql-schema.sql"));
29. runner.runScript(Resources.getResourceAsReader("ddl/mysql/jpetstore-mysql-dataload.sql"));
31. }
33. } **catch** (Exception e) {
34. e.printStackTrace();
35. }

38. }
40. }

[**ScriptRunner(com.ibatis.common.jdbc.\*)用法**](http://czjxdm.javaeye.com/blog/27930)

ScriptRunner类用于执行SQL语句,例如创建数据库schema，或传入缺省或测试数据库等等。从下面的例子可以认识到它的易用性:  
  
    例子1：使用现成的数据库连接  
  
   Connection conn=getConnection();//some method to get a Connection  
  
  ScriptRunner runner=new ScriptRunner();  
  
  runner.runScript(conn,Resources.getResourceAsReader("com/some/resource/path/initialize.sql"));  
  
   conn.close();  
  
  例子2：使用新的数据库连接  
  
  ScriptRunner runner=new ScriptRunner("com.some.Driver","jdbc:url://db","login","password");  
  
  runner.runScript(conn,new FileReader("/user/local/db/scripts/initialize-db.sql"));  
  
例子3：使用新创建的数据连接  
  
  Properties props= getProperties();//some properties form somewhere  
  
ScriptRunner runner =new ScriptRunner(props);  
  
runner.runScript(conn,new FileReader("/user/local/db/scripts/initialize-db.sql"));

3.ant执行SQL脚本

**[java]** [view plain](http://blog.csdn.net/runming56/article/details/35985479) [copy](http://blog.csdn.net/runming56/article/details/35985479)

[在CODE上查看代码片](https://code.csdn.net/snippets/411209)

1. **package** com.unmi;
2. **import** java.io.\*;
3. **import** org.apache.tools.ant.\*;
4. **import** org.apache.tools.ant.taskdefs.\*;
5. **import** org.apache.tools.ant.types.\*;
6. /\*\*
7. \* 调用 ant.jar 的 SQLExec 执行 SQL 脚本文件
8. \* @author Unmi
9. \*/
10. **public** **class** AntExecSql {
11. /\*\*
12. \* @param args
13. \*/
14. **public** **static** **void** main(String[] args) {
15. SQLExec sqlExec = **new** SQLExec();
16. //设置数据库参数
17. sqlExec.setDriver("oracle.jdbc.driver.OracleDriver");
18. sqlExec.setUrl("jdbc:oracle:thin:@10.128.x.x:1521:xxsid");
19. sqlExec.setUserid("xxuser");
20. sqlExec.setPassword("xxpass");
21. //要执行的脚本
22. sqlExec.setSrc(**new** File("src/data.sql"));
23. //有出错的语句该如何处理
24. sqlExec.setOnerror((SQLExec.OnError)(EnumeratedAttribute.getInstance(
25. SQLExec.OnError.**class**, "abort")));
26. sqlExec.setPrint(**true**); //设置是否输出
27. //输出到文件 sql.out 中；不设置该属性，默认输出到控制台
28. sqlExec.setOutput(**new** File("src/sql.out"));
29. sqlExec.setProject(**new** Project()); // 要指定这个属性，不然会出错
30. sqlExec.execute();
31. }
32. }
33. Sql文件必须是ansi格式