

原创 【android】 ORMLite框架 的使用方法---给你的数据库操作插上翅膀

2017-03-08 18:27:18 da_caoyuan 阅读数 5439 ☆ 收藏 更多

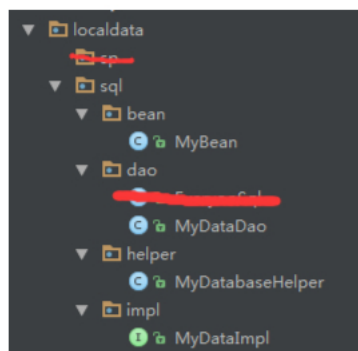
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载时附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/da_caoyuan/article/details/60876336

一：首先下载 ORMLite Jar 包

先去 [ORMLite官网](#) 下载jar包 写博客时，目前最新的，对于Android为：ormlite-android-5.0.jar 和 ormlite-core-5.0.jar；

然后分包处理，建议如图所示分包：



二：配置Bean类

```
1  @DatabaseTable(tableName = "Book")
2  public class MyBean {
3
4      @DatabaseField(generatedId = true)
5      private int id;
6
7      @DatabaseField(columnName = "name")
8      public String name;
9
10     @DatabaseField(columnName = "author")
11     public String author;
12
13     @DatabaseField(columnName = "price")
14     public String price;
15
16     @DatabaseField(columnName = "pages")
17     public int pages;
18
19
20     public String getAuthor() {
21         return author;
22     }
23
24     public void setAuthor(String author) {
25         this.author = author;
26     }
27
28     public String getPrice() {
29         return price;
30     }
31
32     public void setPrice(String price) {
33         this.price = price;
34     }
35
36     public int getPages() {
37         return pages;
38     }
39
40     public void setPages(int pages) {
41         this.pages = pages;
```

```

41         mPagePages = pages;
42     }
43
44     public String getName() {
45         return name;
46     }
47
48     public void setName(String name) {
49         this.name = name;
50     }
51 }

```

三：编写Helper类

```

1  public class MyDatabaseHelper extends OrmLiteSqliteOpenHelper {
2
3      public static final String DB_NAME = "BookStore.db";
4      public static final int DB_VERSION = 1;
5
6
7      public MyDatabaseHelper(Context context) {
8          super(context, DB_NAME, null, DB_VERSION);
9      }
10
11
12      @Override
13      public void onCreate(SQLiteDatabase sqLiteDatabase, ConnectionSource connect
14          try {
15              TableUtils.createTable(connectionSource, MyBean.class);
16          } catch (SQLException e) {
17              e.printStackTrace();
18          }
19      }
20
21      @Override
22      public void onUpgrade(SQLiteDatabase sqLiteDatabase, ConnectionSource connec
23          System.out.println("MyDatabaseHelper.onUpgrade oldVersion=" + oldVersion +
24          try {
25
26              switch (oldVersion) {
27                  case 1:
28                      getDao(MyBean.class).executeRaw("alter table Book add column box
29                      //在数据库版本1的下一版本，Book表中新添加了 book_type 字段
30
31                  case 2:
32                      // TableUtils.createTable(connectionSource, MyBean2.class);
33                      //在数据库版本2的下一版本，新增加了一张表
34                  default:
35                      break;
36              }
37
38              //显然这样处理比较暴力
39              //TableUtils.dropTable(connectionSource, MyBean.class, true);
40              //onCreate(sqLiteDatabase, connectionSource);
41          } catch (SQLException e) {
42              e.printStackTrace();
43          }
44      }
45
46
47      private static MyDatabaseHelper instance;
48
49      /**
50       * 单例获取该Helper
51       *
52       * @param context
53       * @return
54       */
55      public static MyDatabaseHelper getHelper(Context context) {
56          if (instance == null) {
57              synchronized (MyDatabaseHelper.class) {
58                  if (instance == null)
59                      instance = new MyDatabaseHelper(context);
60              }
61          }
62      }

```

```

61         }
62         return instance;
63     }
64
65
66     private Map<String, Dao> daos = new HashMap<>();
67
68     public synchronized Dao getDao(Class clazz) throws SQLException {
69         Dao dao = null;
70         String className = clazz.getSimpleName();
71         if (daos.containsKey(className)) {
72             dao = daos.get(clazz);
73         }
74         if (dao == null) {
75             dao = super.getDao(clazz);
76             daos.put(className, dao);
77         }
78         return dao;
79     }
80
81
82     @Override
83     public void close() {
84         super.close();
85         for (String key : daos.keySet()) {
86             Dao dao = daos.get(key);
87             dao = null;
88         }
89     }
90
91
92 }

```

四：编写DAO类

1: 接口编写:

```

1  public interface MyDataImpl {
2
3      void insert(ArrayList<MyBean> beanArrayList);
4
5      void insert(MyBean myBean);
6
7      void update(String name, String price);
8
9      void update2(String columnName, String columnValue);
10
11     void update3(String queryColumnName, String queryColumnValue, String setColumn
12
13
14     void delete(String name);
15
16     int deleteAll();
17
18
19     ArrayList<String> queryPrice(String name);
20
21     String queryAuthor(String name, String price);
22
23     long queryCount();
24
25     ArrayList<MyBean> queryId(int id);
26
27     ArrayList<MyBean> queryAll();
28
29
30 }

```

```

1  public class MyDataDao implements MyDataImpl {
2      private MyDatabaseHelper mHelper;
3      private Dao<MyBean, Integer> dao;
4      private Context mContext;
5      private static MyDataDao instance;
6

```

```

7     protected MyDataDao(Context context) {
8         this.mContext = context;
9         try {
10             mHelper = MyDatabaseHelper.getHelper(mContext);
11             dao = mHelper.getDao(MyBean.class);
12         } catch (SQLException e) {
13             e.printStackTrace();
14         }
15     }
16
17
18     public static MyDataDao getInstance(Context context) {
19         if (instance == null) {
20             synchronized (MyDataDao.class) {
21                 if (instance == null) {
22                     instance = new MyDataDao(context);
23                 }
24             }
25         }
26         return instance;
27     }
28 }
29
30
31 @Override
32 public void insert(MyBean myBean) {
33
34
35     try {
36
37         //事务操作
38         /* TransactionManager.callInTransaction(mHelper.getConnectionSource(), new Callable<
39         @Override
40         public Void call() throws Exception {
41             return null;
42         }
43     };*/
44
45
46         dao.create(myBean);
47         //dao.createOrUpdate(myBean);//和上一行的方法效果一样
48     } catch (SQLException e) {
49         e.printStackTrace();
50     }
51 }
52
53
54 @Override
55 public void insert(ArrayList<MyBean> beanArrayList) {
56     try {
57         dao.create(beanArrayList);
58     } catch (SQLException e) {
59         e.printStackTrace();
60     }
61 }
62
63 @Override
64 public void update(String name, String price) {
65     ArrayList<MyBean> list = null;
66     try {
67         list = (ArrayList<MyBean>) dao.queryForEq("name", name);
68         if (list != null) {
69             for (MyBean bean : list) {
70                 bean.setPrice(price);
71                 dao.update(bean);
72                 //dao.createOrUpdate(bean);//和上一行的方法效果一样
73             }
74         }
75     } catch (SQLException e) {
76         e.printStackTrace();
77     }
78 }
79 }
80

```

```

81  @Override
82  public void update2(String columnName, String columnValue) {
83      try {
84          //下面这两个代码的意思一样
85          dao.updateBuilder().updateColumnValue(columnName, columnValue).u
86          //dao.updateRaw("update Book set " + columnName + "=?", new String[]{columnValue},
87      } catch (SQLException e) {
88          e.printStackTrace();
89      }
90
91
92  }
93
94  @Override
95  public void update3(String queryColumnName, String queryColumnValue, String
96      try {
97          String sql = "update Book set " + setColumnName + "=" + setColumnVali
98          System.out.println("MyDataDao.update3 sql=" + sql);
99          dao.updateRaw(sql);
100
101          //dao.updateRaw("update Book set price= '33333元' where name= '西游记');//等价于上
102      } catch (SQLException e) {
103          e.printStackTrace();
104      }
105  }
106
107  @Override
108  public void delete(String name) {
109      ArrayList<MyBean> list = null;
110      try {
111          list = (ArrayList<MyBean>) dao.queryForEq("name", name);
112          if (list != null) {
113              for (MyBean bean : list) {
114                  dao.delete(bean);
115              }
116          }
117      } catch (SQLException e) {
118          e.printStackTrace();
119      }
120  }
121
122  /**
123  * @return -1:删除数据异常 0: 无数据
124  */
125  @Override
126  public int deleteAll() {
127      int number = -1;
128      try {
129          number = dao.deleteBuilder().delete();//返回删除的数据条数 例如: 删除1
130
131          //dao.deleteBuilder().where().eq("name", "记").reset();//????
132      } catch (SQLException e) {
133          e.printStackTrace();
134      }
135      return number;
136  }
137
138  @Override
139  public ArrayList<String> queryPrice(String name) {
140      List<MyBean> list = null;
141      ArrayList<String> strings = null;
142      try {
143          list = dao.queryForEq("name", name);
144          if (list != null) {
145              strings = new ArrayList<>();
146              for (MyBean myBean : list) {
147                  strings.add(myBean.getPrice());
148              }
149              /*for (int i = 0; i < list.size(); i++) {
150                  strings.add(list.get(i).getPrice());
151              }*/
152          }
153      } catch (SQLException e) {
154          e.printStackTrace();
155      }

```

```

156         return strings;
157     }
158
159     @Override
160     public String queryAuthor(String name1, String price1) {
161         List<MyBean> list = null;
162         String author = "";
163
164         try {
165             list = dao.queryBuilder().where().eq("name", name1).and().eq("price", price1);
166             if (list != null) {
167                 for (MyBean myBean : list) {
168                     author = myBean.getAuthor();
169                 }
170             }
171         } catch (SQLException e) {
172             e.printStackTrace();
173         }
174
175         return author; //说明: 如果这个 author 是唯一的, 可以这样的返回。如果是多个的话, 要返回多个
176     }
177
178     /**
179     * @return 表中数据的个数
180     */
181     @Override
182     public long queryCount() {
183         long number = 0;
184         try {
185             number = dao.queryBuilder().countOf();
186         } catch (SQLException e) {
187             e.printStackTrace();
188         }
189         return number;
190     }
191
192     /**
193     * @param id 这个id 就是表中, 每次插入数据, 自己递增的id 字段
194     */
195     @Override
196     public ArrayList<MyBean> queryId(int id) {
197         ArrayList<MyBean> list = null;
198
199         try {
200             MyBean myBean = dao.queryForId(id);
201             if (myBean != null) {
202                 list = new ArrayList<>();
203                 list.add(myBean);
204             }
205             return list;
206         } catch (SQLException e) {
207             e.printStackTrace();
208         }
209         return list;
210     }
211
212     @Override
213     public ArrayList<MyBean> queryAll() {
214         ArrayList<MyBean> list = null;
215         try {
216             list = (ArrayList<MyBean>) dao.queryForAll();
217
218             if (list != null) {
219                 return list;
220             }
221         } catch (SQLException e) {
222             e.printStackTrace();
223         }
224         return list;
225     }
226 }
227
228
229 public boolean deleteTables(Context context, String DBname) {
230     //?????

```

```
231         return false;
232     }
233
234
235     /**
236     * 这个方法可以的
237     */
238     public boolean deleteDatabases(Context context, String DBname) {
239         return context.deleteDatabase(DBname);
240     }
241
242
243 }
```

五：测试

源码下载地址

参考文章：

[Android快速开发–使用ORMLite操作数据库](#)

鸿洋的博客：

[Android ORMLite 框架的入门用法](#)

[Android 快速开发系列 ORMLite 框架最佳实践](#)

[SQL 语法](#)

[SQL UPDATE 语句](#)

[OrmLite 官网](#)