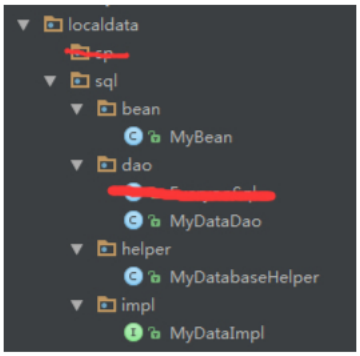
原创 【android】ORMLite框架 的使用方法---给你的数据库操作插上翅膀

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

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

```java
@DatabaseTable(tableName = "Book")
public class MyBean {

    @DatabaseField(generatedId = true)
    private int id;

    @DatabaseField(columnName = "name")
    public String name;

    @DatabaseField(columnName = "author")
    public String author;

    @DatabaseField(columnName = "price")
    public String price;

    @DatabaseField(columnName = "pages")
    public int pages;


    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public int getPages() {
        return pages;
    }

    public void setPages(int pages) {
        this.pages = pages;
```

```
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

三：编写Helper类

```
public class MyDatabaseHelper extends OrmLiteSqliteOpenHelper {

    public static final String DB_NAME = "BookStore.db";
    public static final int DB_VERSION = 1;


    public MyDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }


    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase, ConnectionSource connect
        try {
            TableUtils.createTable(connectionSource, MyBean.class);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, ConnectionSource conne
        System.out.println("MyDatabaseHelper.onUpgrade oldVersion=" + oldVersion +
        try {

            switch (oldVersion) {
                case 1:
                    getDao(MyBean.class).executeRaw("alter table Book add column boo
                    //在数据库版本1的下一版本，Book表中新添加了 book_type 字段

                case 2:
                    // TableUtils.createTable(connectionSource, MyBean2.class);
                    //在数据库版本2的下一版本，新增加了一张表
                default:
                    break;
            }


            //显然这样处理比较暴力
            //TableUtils.dropTable(connectionSource, MyBean.class, true);
            //onCreate(sqLiteDatabase, connectionSource);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static MyDatabaseHelper instance;

    /**
   * 单例获取该Helper
   *
   * @param context
   * @return
   */
    public static MyDatabaseHelper getHelper(Context context) {
        if (instance == null) {
            synchronized (MyDatabaseHelper.class) {
                if (instance == null)
                    instance = new MyDatabaseHelper(context);
            }
        }
```

```
        }
        return instance;
    }


    private Map<String, Dao> daos = new HashMap<>();

    public synchronized Dao getDao(Class clazz) throws SQLException {
        Dao dao = null;
        String className = clazz.getSimpleName();
        if (daos.containsKey(className)) {
            dao = daos.get(clazz);
        }
        if (dao == null) {
            dao = super.getDao(clazz);
            daos.put(className, dao);
        }
        return dao;
    }


    @Override
    public void close() {
        super.close();
        for (String key : daos.keySet()) {
            Dao dao = daos.get(key);
            dao = null;
        }
    }


}
```

四：编写DAO类

1：接口编写：

```
public interface MyDataImpl {

    void insert(ArrayList<MyBean> beanArrayList);

    void insert(MyBean myBean);

    void update(String name, String price);

    void update2(String columnName, String columnValue);

    void update3(String queryColumnName, String queryColumnValue, String setColumn


    void delete(String name);

    int deleteAll();


    ArrayList<String> queryPrice(String name);

    String queryAuthor(String name, String price);

    long queryCount();

    ArrayList<MyBean> queryId(int id);

    ArrayList<MyBean> queryAll();


}
```

```
public class MyDataDao implements MyDataImpl {
    private MyDatabaseHelper mHelper;
    private Dao<MyBean, Integer> dao;
    private Context mContext;
    private static MyDataDao instance;

```

```
    protected MyDataDao(Context context) {
        this.mContext = context;
        try {
            mHelper = MyDatabaseHelper.getHelper(mContext);
            dao = mHelper.getDao(MyBean.class);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


    public static MyDataDao getInstance(Context context) {
        if (instance == null) {
            synchronized (MyDataDao.class) {
                if (instance == null) {
                    instance = new MyDataDao(context);
                }
            }

        }
        return instance;
    }


    @Override
    public void insert(MyBean myBean) {


        try {

            //事务操作
           /* TransactionManager.callInTransaction(mHelper.getConnectionSource(), new Callable<
        @Override
        public Void call() throws Exception {
          return null;
        }
      });*/


            dao.create(myBean);
            //dao.createOrUpdate(myBean);//和上一行的方法效果一样
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


    @Override
    public void insert(ArrayList<MyBean> beanArrayList) {
        try {
            dao.create(beanArrayList);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void update(String name, String price) {
        ArrayList<MyBean> list = null;
        try {
            list = (ArrayList<MyBean>) dao.queryForEq("name", name);
            if (list != null) {
                for (MyBean bean : list) {
                    bean.setPrice(price);
                    dao.update(bean);
                    //dao.createOrUpdate(bean);//和上一行的方法效果一样
                }

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

```

```
    @Override
    public void update2(String columnName, String columnValue) {
        try {
            //下面这两个代码的意思一样
            dao.updateBuilder().updateColumnValue(columnName, columnValue).u
            //dao.updateRaw("update Book set " + columnName + "=?", new String[]{columnValue},
        } catch (SQLException e) {
            e.printStackTrace();
        }


    }

    @Override
    public void update3(String queryColumnName, String queryColumnValue, String
        try {
            String sql = "update Book set " + setColumnName + "='" + setColumnVal
            System.out.println("MyDataDao.update3 sql=" + sql);
            dao.updateRaw(sql);

            //dao.updateRaw("update Book set price='33333元' where name='西游记'");//等价于上
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void delete(String name) {
        ArrayList<MyBean> list = null;
        try {
            list = (ArrayList<MyBean>) dao.queryForEq("name", name);
            if (list != null) {
                for (MyBean bean : list) {
                    dao.delete(bean);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
   * @return -1:删除数据异常 0：无数据
   */
    @Override
    public int deleteAll() {
        int number = -1;
        try {
            number = dao.deleteBuilder().delete();//返回删除的数据条数 例如：删除1

            //dao.deleteBuilder().where().eq("name", "记").reset();////???
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return number;
    }

    @Override
    public ArrayList<String> queryPrice(String name) {
        List<MyBean> list = null;
        ArrayList<String> strings = null;
        try {
            list = dao.queryForEq("name", name);
            if (list != null) {
                strings = new ArrayList<>();
                for (MyBean myBean : list) {
                    strings.add(myBean.getPrice());
                }
                /*for (int i = 0; i < list.size(); i++) {
          strings.add(list.get(i).getPrice());
        }*/
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
        return strings;
    }

    @Override
    public String queryAuthor(String name1, String price1) {
        List<MyBean> list = null;
        String author = "";

        try {
            list = dao.queryBuilder().where().eq("name", name1).and().eq("price
            if (list != null) {
                for (MyBean myBean : list) {
                    author = myBean.getAuthor();
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        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
    }

    /**
     * @return 表中数据的个数
     */
    @Override
    public long queryCount() {
        long number = 0;
        try {
            number = dao.queryBuilder().countOf();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return number;
    }

    /**
     * @param id 这个id 就是表中，每次插入数据，自己递增的id 字段
     */
    @Override
    public ArrayList<MyBean> queryId(int id) {
        ArrayList<MyBean> list = null;

        try {
            MyBean myBean = dao.queryForId(id);
            if (myBean != null) {
                list = new ArrayList<>();
                list.add(myBean);
            }
            return list;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    @Override
    public ArrayList<MyBean> queryAll() {
        ArrayList<MyBean> list = null;
        try {
            list = (ArrayList<MyBean>) dao.queryForAll();


            if (list != null) {
                return list;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }


    public boolean delteTables(Context context, String DBname) {
        //?????
```

```
231             return false;
232         }
233 
234 
235         /**
236         * 这个方法可以的
237         */
238         public boolean delteDatabases(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