# BDtools使用文档

## com.zhh.adapter.tools(包名)

com.zhh.adapter.tools包中主要有CommonAdapter适配器，该适配器继承于BaseAdapter。

### 1.1CommonAdapter(类名)

#### 1.1.1 构造方法（new CommonAdapter）

public CommonAdapter<T>(Context mContext, List<T> mList, int resource)

mContext ：上下文

mList ：适配器所需要数据源，例如：List<Map<String, Object>> mList

resource ：将要使用的layout，例如：R.layout.XXXXX

#### 1.1.2内部实现抽象方法(setViewData)

public void setViewData (CommonViewHolder commonViewHolder, View currentView, T item, int position)

commonViewHolder ：当前适配器所使用的ViewHolder视图钩子

currentView ：当前的View

item ：当前position条目的数据源

position ：当前所属条目数

#### 1.1.3 试图钩子使用方法(CommonViewHolder)

pubclic View get(CommonViewHolder commonViewHolder, View currentView, int key)

commonViewHolder ：当前适配器所使用的ViewHolder视图钩子

currentView ：当前的View

key ：将要使用View的Id，例如：R.id.XXXXXX

#### 1.1.4 CommonAdapter使用方法

CommonAdapter是继承BaseAdapter的类，一切可以使用BaseAdapter的View都可以使用CommonAdapter来进行适配。在CommonAdapter中自带视图钩子进行适配器的优化。使用时只需要new一个对象并实现内部抽象方法就可以使用。

**CommonAdapter mApdater = new** CommonAdapter<Map<String, Object>>(activity, list, **R**.**layout**.***activity\_main***) {

***@Override***

**public** **void** **setViewData**(**CommonViewHolder** commonViewHolder, **View** currentView, Map<String, Object> item, **int** position) {

**TextView** **tv** = (**TextView**) commonViewHolder.get(commonViewHolder, currentView, **R**.**id**.***action\_settings***);

tv.setText("zzhh");

}

listView.setAdapter(mAdapter);

gridView.setAdapter(mAdapter);

## com.zhh.sqlite.tools(包名)

Com.zhh.sqlite.tools包主要是提供sqlite服务，包括表的创建、更新，表数据的增、删、改、查等功能。

### 2.1DataBaseFactory(类名)

#### 2.1.1构造方法（new DataBaseFactory）

**public** **DataBaseFactory**(**Context** context,**int** DBVERSION, List<**Class**<?>> list)

context ：上下文

DBVERSION ：数据库版本号

list ：装载想要创建表的JavaBean类对象

#### 2.1.2DataBaseFatory使用方法

DataBaseFatory继承于SQLiteOpenHelper，主要用于进行SQLite的创建、更新。使用时需要在使用之前初始化（通常在Application）。整个app只需创建一次即可。在构造方法的参数中提供数据库版本号（int DBVERSION）参数来维护数据库更新。因为是继承于SQLiteOpenHelper，并重写了其中的onUpgrade方法进行数据库的更新。在传入的参数list中，需要放入已写好的JavaBean类对象（Student.class），JavaBean类对象详见2.2JavaBean的创建。

List<**Class**<?>> **list** = **new** ArrayList<**Class**<?>>();

list.add(**Student**.**class**);

list.add(**Teacher**.**class**);

list.add(**School**.**class**);

**new** DataBaseFactory(**this**, 1, list);

### 2.2JavaBean的创建

普通JavaBean的创建过程这里不做赘述，主要说需要创建表的JavaBean对象。在com.zhh.sqlite.tools包下提供了Table和Id两个注释类。分别用来填写表名称和表主键。需要在类上面注释上表名和表主键对象。成员变量私有，实现get和set方法。目前成员对象类型支持String、Integer（int）、Double（double）三种类型。举例如下：

**package** com.zhh.domin;

**import** com.zhh.sqlite.tools.Id;

**import** com.zhh.sqlite.tools.Table;

/\*\*

\* 作者： zzhh

\*

\* 时间：@2016年12月23日 下午3:45:43

\*/

***@Table***(name = "t\_a\_student") //表名

***@Id***(name = "id") //表中主键

**public** **class** **Student** {

**private** **double** id; //学生ID

**private** **String** name; //姓名

**private** **int** gender; //性别

**private** **int** age; //年龄

**private** **String** phone; //联系电话

**private** **int** grade; //年级号

**private** **int** clazz; //班级号

**private** **double** teacherId; //班主任ID

**public** **double** **getId**() {

**return** id;

}

**public** **void** **setId**(**double** id) {

**this**.id = id;

}

**public** **String** **getName**() {

**return** name;

}

**public** **void** **setName**(**String** name) {

**this**.name = name;

}

**public** **int** **getGender**() {

**return** gender;

}

**public** **void** **setGender**(**int** gender) {

**this**.gender = gender;

}

**public** **int** **getAge**() {

**return** age;

}

**public** **void** **setAge**(**int** age) {

**this**.age = age;

}

**public** **String** **getPhone**() {

**return** phone;

}

**public** **void** **setPhone**(**String** phone) {

**this**.phone = phone;

}

**public** **int** **getGrade**() {

**return** grade;

}

**public** **void** **setGrade**(**int** grade) {

**this**.grade = grade;

}

**public** **int** **getClazz**() {

**return** clazz;

}

**public** **void** **setClazz**(**int** clazz) {

**this**.clazz = clazz;

}

**public** **double** **getTeacherId**() {

**return** teacherId;

}

**public** **void** **setTeacherId**(**double** teacherId) {

**this**.teacherId = teacherId;

}

}

### 2.3BaseService（类名）

BaseService是一个单例模式的工具类，主要通过实体类（JavaBean）对sqlite表进行操作

#### 2.3.1public <T> T insert(T entity)

插入一条数据

entity ：插入一个表的一条数据（实体类）

也可适用于更新操作（如果主键已存在）

例如：Student stu = new Student();

。。。

进行set(),get()操作；

。。。

BaseService.getInstance.insert(student);

#### 2.3.2public <T> List<T> insert(List<T> entitys)

插入多条数据

entitys ：差入一个表的多条数据（实体类集合）

也可适用于更新操作（如果主键已存在）

例如：List<Student> list = new ArrayList<Student>();

。。。

进行数据处理

。。。

BaseService.getInstance.insert(list);

#### 2.3.3public <T> Integer update(T entity)

更新一条数据

entity ：更新一个表的一条数据（实体类）

在表中已存在（根据主键进行判断）

例如：Student stu = BaseService.getInstance.findOn(123, Student.class);

。。。

进行set(),get()操作；

。。。

BaseService.getInstance.insert(student);

#### 2.3.4public <T> T findOne(Serializable id, Class<?> clazz)

查询某一条数据（已知主键）

id ：表中主键

clazz ：实体类类型

例如：Student stu = BaseService.getInstance.findOn(123, Student.class);

#### 2.3.5**public <T> List<T> findAll(Class<?> clazz)**

查询某一个表的所有数据，返回值为List<T>类型

clazz ：实体类类型

例如：List<Student> list = BaseService.getInstance.findAll(Student.class);

#### 2.3.6**public Integer deleteOne(Serializable id, Class<?> clazz)**

删除表中的一条数据

id ：表中主键

clazz ：实体类类型

例如：BaseService.getInstance.deleteOne(123, Student.class);

#### 2.3.7**public void deleteMore(List<Serializable> ids, Class<?> clazz)**

删除多条数据

ids ：存放表中主键的List集合

clazz ：实体类类型

例如：List<Integer> list = new ArrayList<Integer>();

list.add(123); list.add(456); 。。。。。。。。

BaseService.getInstance.deleteMore(list, Student.class);

#### 2.3.8**public void deleteAll(Class<?> clazz)**

删除表中所有数据

clazz ：实体类类型

例如：BaseService.getInstance.deleteAll(Studeng.class);

#### 2.3.9**public <T> List<T> queryForEntitys(String sql, String[] paramValues, Class<?> clazz)**

查询一个表中的多条数据，可使用sql语句与占位符，类似rawQuery方法。返回值为List集合

sql ：sql语句

paramValuse ：占位符中字符

clazz ：实体类类型

例如：String sql = “select \* from t\_a\_student where age = ?”;

String[] paramValues = new String[](“14”);

List<Student> list = BaseService.getInstance.queryForEntitys(sql, paramValuse, Student.class);

#### 2.3.10**public List<Map<String, Object>> queryForMaps(String sql, String[] paramValues)**

查询一个表中多条数据的某些字段，可使用sql语句与占位符，支持复杂查询，返回值为List<Map<String, Object>>

sql ：sql语句

paramValuse ：占位符中字符

例如：String sql = “select name,age,gender from t\_a\_student where age = ?”;

String[] paramValues = new String[](“14”);

List<Map<String, Object> list = BaseService.getInstance.queryForMaps(sql, paramValuse);