实验七：数据存储（二）

13331388　庄梓嘉

实验目的

1.学会使用 SQLite 数据库保存数据。

2.学会对 SQLite 数据库里的数据进行增删改查等操作。

实验内容

实现一个通讯录查看程序：

A） 要求使用 SQLite 数据库保存通讯录，使得每次运行程序均能显示当前联系人列表。

B） 主界面包含一个添加联系人按钮和一个联系人列表，每一项显示联系人学号，姓名，手机号码。

C） 点击添加联系人按钮能添加新的联系人。

D） 在次界面输入联系人信息之后点击确定按钮会返回主界面（如果输入的联系人信息有效的话，更新主界面的联系人列表）。

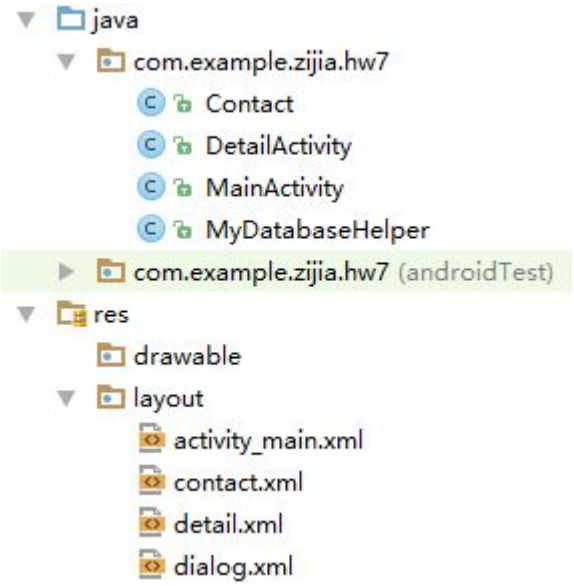
【拓展项】

1.长按某个联系人，弹出对话框，询问是否删除该联系人。

2.单击某个联系人，弹出修改联系人的对话框。

实验步骤

实验文件目录：



（1）创建 MyDatabaseHelper 类，帮助实现功能要求。

```
//创建MyDatabaseHelper类
public class MyDatabaseHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "Contacts.db";
    private static final String TABLE_NAME = "Contacts";
    private static final int DB_VERSION = 1;

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

（2）创建数据库，执行创建数据库的 SQL 语句。

```
//创建数据库，执行创建数据库的SQL语句
public void onCreate(SQLiteDatabase db) {
    String CREATE_TABLE = "create table " + TABLE_NAME + "(_id integer primary key autoincrement, " +
            "_no text not null, _name text not null, _pnumber text);";
    db.execSQL(CREATE_TABLE);
}
```

（3）加入数据库更新、插入、删除功能函数。

```
//数据库更新功能
public int update(Contact entity, String where) {
    SQLiteDatabase db = getWritableDatabase();
    String whereClause = "_no = ?";
    String[] whereArgs = {where};
    ContentValues values = new ContentValues();
    values.put("_no", entity.getNo());
    values.put("_name", entity.getName());
    values.put("_pnumber", entity.getPnumber());
    int rows = db.update(TABLE_NAME, values, whereClause, whereArgs);
    db.close();
    return rows;
}
 //数据库插入功能
 public long insert(Contact entity) {
     SQLiteDatabase db = getWritableDatabase();
     ContentValues values = new ContentValues();
     values.put("_no", entity.getNo());
     values.put("_name", entity.getName());
     values.put("_pnumber", entity.getPnumber());

     long id = db.insert(TABLE_NAME, null, values);
     db.close();
     return id;
 }
```

（4）重载 onUpgrade 实例化

```
//重载onUpgrade实例化
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXIST " + TABLE_NAME);
    onCreate(db);
}
```

（5）添加数据库查询操作代码

```
//数据库查询操作
public Cursor query() {
    SQLiteDatabase db = getWritableDatabase();
    return db.query(TABLE_NAME, null, null, null, null, null, null);
}
```

（6）参照 HW3，编写一个简易适配器，连接数据和 UI 显示。

```
//简易适配器，连接数据和UI显示
public class dbAdapter extends SimpleAdapter {
    private ArrayList<Map<String, String>> mData;
    private LayoutInflater layoutInflater;
    public dbAdapter(Context context, List<? extends Map<String, String>> data, int resource, String[] from, int[] to) {
        super(context, data, resource, from, to);
        this.layoutInflater = LayoutInflater.from(context);
        this.mData = (ArrayList<Map<String, String>>)data;
    }
    public View getView(int position, View view, ViewGroup parent) {
        viewHolder holder = null;
        if (view == null) {
            view = layoutInflater.inflate(R.layout.contact, null);
            holder = new viewHolder();
            holder.no = (TextView) view.findViewById(R.id.noTv);
            holder.name = (TextView) view.findViewById(R.id.nameTv);
            holder.phone = (TextView) view.findViewById(R.id.phoneTv);
            view.setTag(holder);
        } else {
            holder = (viewHolder) view.getTag();
        }
        Map<String, String> item = mData.get(position);
        holder.no.setText(item.get("no"));
        holder.name.setText(item.get("name"));
        holder.phone.setText(item.get("pNumber"));
        return view;
    }
}
```

（7）编写添加数据保存到数据库中代码。

```
//设置数据保存到数据库中
public void setData(List<Map<String, String>> list) {
    Map<String, String> data;
    Cursor c = myDatabaseHelper.query();
    while (c.moveToNext()) {
        data = new HashMap<String, String>();
        data.put("no", c.getString(c.getColumnIndex("_no")));
        data.put("name", c.getString(c.getColumnIndex("_name")));
        data.put("pNumber", c.getString(c.getColumnIndex("_pnumber")));
        list.add(data);
    }
}
```

（8）设计主页面布局，以及添加按钮监听事件。

```
addNew.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, DetailActivity.class);
        startActivity(intent);
        MainActivity.this.finish();
    }
});
```

（9）添加自定义联系人类 Contact，包括学号、姓名、手机等信息及其访问方法。

```
package com.example.zijia.hw7;
//自定义联系人类Contact，包括学号、姓名、手机等信息及其访问方法
public class Contact {
    private String _no;
    private String _name;
    private String _pnumber;
    public Contact() {
        _no = "";
        _name = "";
        _pnumber = "";
    }
    public void setNo(String no) { _no = no; }
    public void setName(String name) { _name = name; }
    public void setPnumber(String pnumber) { _pnumber = pnumber; }
    public String getNo() { return _no; }
    public String getName() { return _name; }
    public String getPnumber() { return _pnumber; }
}
```

（10）添加页面的布局，及按钮监听事件。

```
Ok.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String no = No.getText().toString();
        String name = Name.getText().toString();
        String phone = Phone.getText().toString();
        if (no.equals("") || name.equals("") || phone.equals("")) return;
        Contact myData = new Contact();
        myData.setNo(no);
        myData.setName(name);
        myData.setPnumber(phone);
        myDatabaseHelper.insert(myData);
        Intent intent = new Intent(DetailActivity.this, MainActivity.class);
        startActivity(intent);
        DetailActivity.this.finish();
    }
});
```

【拓展项】

（1）设置长按联系人监听事件，弹出对话框，对话框布局及按钮监听事件。

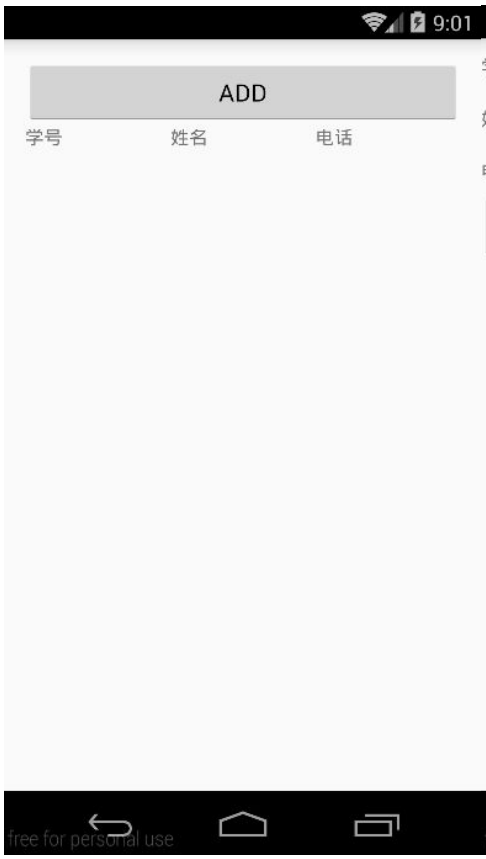
```
//设置长按联系人监听事件，弹出对话框，对话框布局及按钮监听事件
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int position, final long id) {
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setMessage("确认删除吗？");
        builder.setTitle("提示");
        builder.setPositiveButton("确认", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
                Map<String, String> item = mapList.get(position);
                Contact contact = new Contact();
                contact.setNo(item.get("no"));
                contact.setName(item.get("name"));
                contact.setPnumber(item.get("pNumber"));
                myDatabaseHelper.delete(contact);
                mapList.clear();
                setData(mapList);
                listView.setAdapter(adapter);
                dialog.dismiss();
            }
        });
        builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        builder.show();
        return true;
    }
});
```

（2）设置单击联系人监听事件，弹出对话框，对话框布局及按钮监听事件

```
//设置单击联系人监听事件，弹出对话框，对话框布局及按钮监听事件
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {
        final View dia = (View) getLayoutInflater().inflate(R.layout.dialog, null);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setTitle("修改");
        builder.setView(dia);
        builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                EditText eno, ena, eph;
                eno = (EditText) dia.findViewById(R.id.xueHao);
                ena = (EditText) dia.findViewById(R.id.xingMing);
                eph = (EditText) dia.findViewById(R.id.dianHua);
                String no = eno.getText().toString();
                String name = ena.getText().toString();
                String phone = eph.getText().toString();
                Contact myData = new Contact();
                myData.setNo(no);
                myData.setName(name);
                myData.setPnumber(phone);
                Map<String, String> item = mapList.get(position);
                String where = item.get("no");
                myDatabaseHelper.update(myData, where);
                mapList.clear();
                setData(mapList);
                listView.setAdapter(adapter);
                dialog.dismiss();
            }
        });
        builder.show();
    }
});
```

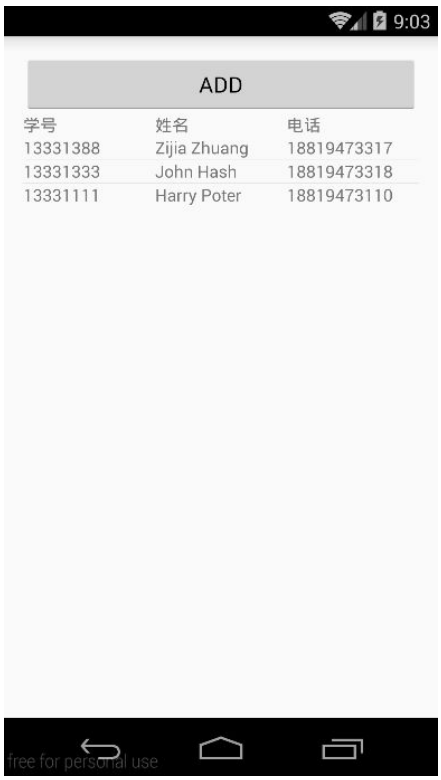
实验结果

主页面



添加页面

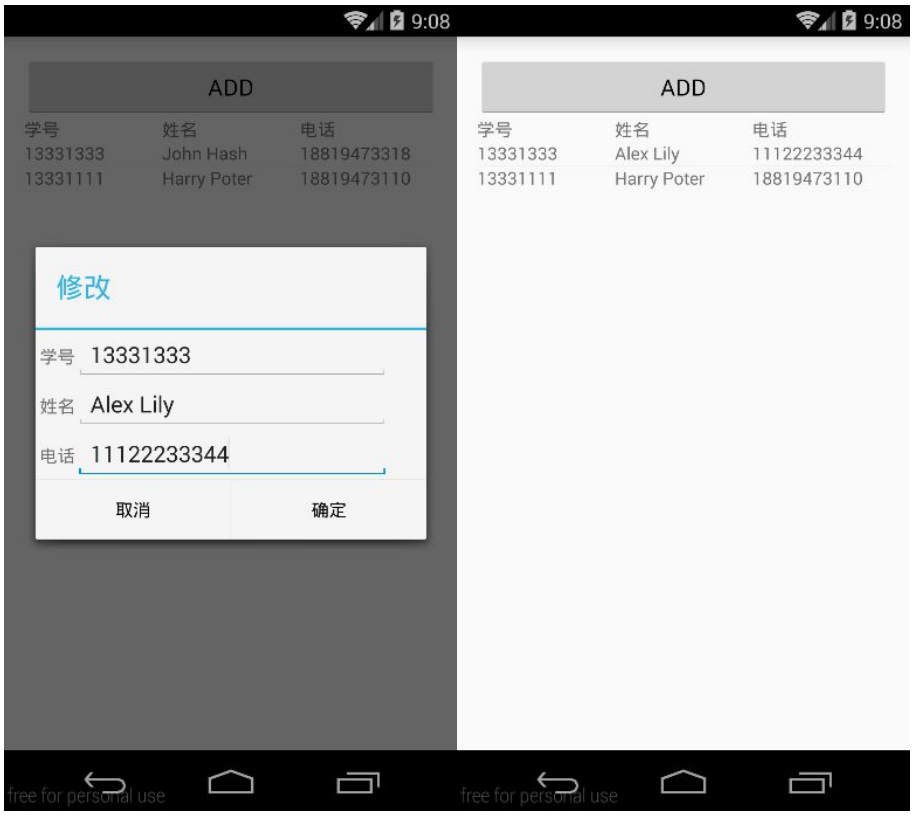
添加完成后页面



【拓展项】

长按联系人　　确定删除后

单击联系人　　确定修改后



参考资料

Android 之 inflate()类方法用途

http://blog.csdn.net/it1039871366/article/details/9796965

Cursor 类介绍

http://www.cnblogs.com/TerryBlog/archive/2010/07/05/1771459.html

Android 之 Adapter 用法总结

http://blog.csdn.net/fznpcy/article/details/8658155/

SimpleAdapter 类使用方法

http://blog.csdn.net/xing1716263268/article/details/7912665

实验总结

这次实验要求使用 SQLite 数据库保存数据，以及对数据库内数据增删查改的功能实现。其中重点是创建数据库、各功能的实现、适配器连续数据和 UI。对话框的使用，需要注意最后要 dimiss 对话框。虽然中间有一些重要功能的代码在实验文档已经给出了，但是还是要自己理解了之后才可以继续进行下去，在获取数据保存到数据库中时，不仔细查询的话就不会知道 moveToNext()就是光标移动到下一行，把数据取出。还有又重温了一次 Adapter 的内容，将 ListView 和数据相互绑定在一起。应该算的上是之前学的东西的一个整合升级版作业。还是能学到不少东西的。