

# 中山大学移动信息工程学院本科生实验报告

## ( 2017 年秋季学期 )

课程名称：移动应用开发

任课教师：

年级	15 级	专业 ( 方向 )	软件工程 ( 移动信息工程 )
学号	15352223	姓名	刘朝开
电话	15626489095	Email	1209354383@qq.com
开始日期	2017.12.18	完成日期	2017.12.20

### 一、 实验题目

#### 实验八 数据存储 ( 二 )

### 二、 实现内容

实现一个生日备忘录，要求实现：

使用 SQLite 数据库保存生日的相关信息，并使得每一次运行程序都可以显示出已经存储在数据库里的内容；

使用 ContentProvider 来获取手机通讯录中的电话号码。

功能要求：

A. 主界面包含增加生日条目按钮和生日信息列表；

B. 点击“增加条目”按钮，跳转到下一个 Activity 界面，界面中包含三个信息输入框（姓名、生日、礼物）和一个“增加”按钮，姓名字段不能为空且不能重复；

C. 在跳转到的界面中，输入生日的相关信息后，点击“增加”按钮返回到主界面，此时，主界面中应更新列表，增加相应的生日信息；

D. 主界面列表点击事件：

点击条目：

弹出对话框，对话框中显示该条目的信息，并允许修改；

对话框下方显示该寿星电话号码（如果手机通讯录中有的话，如果没有就显示“无”）

点击“保存修改”按钮，更新主界面生日信息列表。

长按条目：

弹出对话框显示是否删除条目；

点击“是”按钮，删除该条目，并更新主界面生日列表。

### 三、 课堂实验结果

#### (1) 实验截图



从左至右：初始界面，添加一部分条目，名字不能为空



从左至右：名字不能重复，点击条目显示信息（可修改），长按删除条目。

#### (2) 实验步骤以及关键代码

##### 1 以下主要介绍各个函数和类的主要作用：

**1)数据库类 MyDB 代码：**在 onCreat 函数中执行创建表操作，在 onUpdate()函数中实现数据库更新操作。

```

public class MyDB extends SQLiteOpenHelper {

    public static final String create_Item = "create table Item ("
        + "id integer primary key autoincrement,"
        + "name text, "
        + "birthday text, "
        + "gift text, "
        + "phone text);";

    private Context mContext;
    public MyDB(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
        mContext = context;
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) { sqLiteDatabase.execSQL(create_Item); }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
        sqLiteDatabase.execSQL("drop table if exists Item");
        onCreate(sqLiteDatabase);
    }
}

```

2)Item 类：封装了生日备忘录每个人的基本属性和提供了 set 和 get 函数接口，用于获得和修改属性内容。

```

public class Item {
    private String name;
    private String birthday;
    private String gift;
    private String phone;

    public Item(String name, String birthday, String gift, String phone)
    {
        this.name = name;
        this.birthday = birthday;
        this.gift = gift;
        this.phone = phone;
    }

    public String getName() { return name; }
    public String getBirthday() { return birthday; }
    public String getGift() { return gift; }
    public String getPhone() { return phone; }
    public void setBirthday(String birthday) { this.birthday = birthday; }
    public void setGift(String gift) { this.gift = gift; }
    public void setPhone(String phone) { this.phone = phone; }
}

```

3)BaseActivity 实现数据库的创建，增删改查功能。

a.创建数据库，数据库名为 Items.db。

```

db = new MyDB( context: this, name: "Items.db", factory: null, version: 2);
db.getWritableDatabase();//创建数据库

```

b.实现数据库的插入，更新，删除操作。

```

public void insert(String name, String birthday, String gift, String phone) {
    SQLiteDatabase database = db.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", name);
    values.put("birthday", birthday);
    values.put("gift", gift);
    values.put("phone", phone);
    database.insert( table: "Item", nullColumnHack: null, values);
    values.clear();
}

public void update(String name, String birthday, String gift) {
    SQLiteDatabase database = db.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("birthday", birthday);
    values.put("gift", gift);
    database.update( table: "Item", values, whereClause: "name = ?", new String[] {name});
}

public void delete(String name) {
    SQLiteDatabase database = db.getWritableDatabase();
    database.delete( table: "Item", whereClause: "name = ?", new String[]{name});
}

```

c.传入名字，用于新增条目时查重。返回为 boolean 类型,若为 true 则没有重名，否则有重名。

```

public boolean query(String fname) {
    boolean flag = true;
    SQLiteDatabase database = db.getWritableDatabase();
    Cursor cursor = database.query( table: "Item", columns: null,
        selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
    if(cursor.moveToFirst())
    {
        do {
            String name = cursor.getString(cursor.getColumnIndex( S: "name"));
            if(name.equals(fname)) {
                flag = false;
                break;
            }
        } while (cursor.moveToNext());
    }
    cursor.close();
    return flag;
}

```

d.返回数据库中的所有条目，用于初始化列表界面。

```

public List<Item> query() //返回数据库的所有信息。
{
    List<Item> itemList = new ArrayList<>();
    SQLiteDatabase database = db.getWritableDatabase();
    Cursor cursor = database.query( table: "Item", columns: null,
        selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
    if(cursor.moveToFirst())
    {
        do {
            String name = cursor.getString(cursor.getColumnIndex( S: "name"));
            String birthday = cursor.getString(cursor.getColumnIndex( S: "birthday"));
            String gift = cursor.getString(cursor.getColumnIndex( S: "gift"));
            String phone = cursor.getString(cursor.getColumnIndex( S: "phone"));
            itemList.add(new Item(name, birthday, gift, phone));
        } while (cursor.moveToNext());
    }
    cursor.close();
    return itemList;
}

```

#### 4)MyAdapter 类：设置 listview 的内容和布局。(以下只截了 getView 的图)

```
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
    Item item = getItem(position);  
    View view;  
    ViewHolder viewHolder;  
  
    if(convertView == null) {  
        view = LayoutInflater.from(getContext()).inflate(resourceId,parent, attachToRoot: false);  
        viewHolder = new ViewHolder();  
        viewHolder.Name = (TextView) view.findViewById(R.id.name);  
        viewHolder.Birthday = (TextView) view.findViewById(R.id.birthday);  
        viewHolder.Gift = (TextView) view.findViewById(R.id.gift);  
        view.setTag(viewHolder);  
    }  
    else {  
        view = convertView;  
        viewHolder = (ViewHolder) view.getTag();  
    }  
  
    viewHolder.Name.setText(item.getName());  
    viewHolder.Birthday.setText(item.getBirthday());  
    viewHolder.Gift.setText(item.getGift());  
    return view;  
}
```

#### 5)MyEventBus 类：用于传递消息，传递对象为 Item 类型的值。

```
public class MyEventBus {  
  
    private Item item;  
    public MyEventBus(Item item) { this.item = item; }  
    public Item getItem() { return item; }  
}
```

## 2 以下实现基本的数据库功能：

### 1) 查询功能：在 MainActivity 中显示列表列表：调用查询函数初始化 itemList。

```
itemList = query();//初始化itemList
```

### 2) 添加功能：

a 在 MainActivity 中，首先是点击按钮实现跳转功能：

```
//添加按钮  
Add.setOnClickListener((view) → {  
    Intent intent = new Intent( packageContext: MainActivity.this,AddItem.class);  
    startActivity(intent);  
});
```

b 在 AddItemActivity 中，通过点击添加按钮，首先判断名字是否为空，若不为空，则将对应的数据添加到数据库中，并发送消息。

```

add.setOnClickListener((view) → {
    for (int i=0; i<3;i++) {
        getText[i] = editTexts[i].getText().toString();
    }
    if (getText[0].equals("")){
        Toast.makeText(context: AddItem.this, text: "名字为空, 请完善", Toast.LENGTH_SHORT).show();
    }
    else {
        boolean flag = query(getText[0]); //用来记录是否有重复
        if(flag) {
            String phone = readContacts(getText[0]); //读取联系电话
            insert(getText[0], getText[1], getText[2], phone);
            Item item = new Item(getText[0], getText[1], getText[2], phone);
            EventBus.getDefault().post(new MyEventBus(item));
            finish();
        }
        else {
            Toast.makeText(context: AddItem.this, text: "名字重复, 请检查", Toast.LENGTH_SHORT).show();
        }
    }
});
});

```

c 其中读取联系人电话的函数如下：

```

private String readContacts(String name) {
    String phone = "无";
    Cursor cursor = null;
    try {
        cursor = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
            projection: null, selection: null, selectionArgs: null, sortOrder: null);
        if (cursor != null) {
            while (cursor.moveToNext()) {
                String displayName = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
                String number = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
                if (displayName.equals(name)) {
                    phone = number;
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
    return phone;
}

```

d. MainActivity 中进行接收消息和增加内容。

```

@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(MyEventBus event)
{
    Item item = event.getItem();
    itemList.add(item);
    adapter.notifyDataSetChanged(); //更新数据
}

```

3) 修改功能：主要调用了数据库更新函数，并实现了列表的更新操作。

```

final AlertDialog.Builder alertDialog = new AlertDialog.Builder(context: MainActivity.this);
alertDialog.setView(view1);
alertDialog.setTitle(" `(★▽)ノ~★恭喜发财★~`");
alertDialog.setPositiveButton(text: "保存修改", (dialogInterface, i) → {
    tname = Name.getText().toString();
    tbirthday = Birthday.getText().toString();
    tgift = Gift.getText().toString();

    update(tname, tbirthday, tgift); //更新数据库
    itemList.get(position).setBirthday(tbirthday);
    itemList.get(position).setGift(tgift);
    adapter.notifyDataSetChanged();
});
alertDialog.setNegativeButton(text: "放弃修改", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
    }
});
alertDialog.show();

```

#### 4) 删除功能：调用数据库删除函数和列表更新操作。

```

listView.setOnItemClickListener(
    (adapterView, view, position, id) → {
        final Item item = itemList.get(position);
        final AlertDialog.Builder deleteDialog = new AlertDialog.Builder(context: MainActivity.this);
        deleteDialog.setMessage("是否删除");
        deleteDialog.setPositiveButton(text: "是", (dialogInterface, i) → {
            delete(item.getName());
            itemList.remove(position);
            adapter.notifyDataSetChanged();
        });
        deleteDialog.setNegativeButton(text: "否", new DialogInterface.OnClickListener() { ... });
        deleteDialog.show();
        return true;
    }
);

```

#### (3) 实验遇到困难以及解决思路

困难：

- 1 不同 Activity 之间无法操作同一数据库。
2. 关于 list 对象更新的问题。

解决方法：

1 新建一个基 Activity，声明私有成员 MyDB 的对象，并提供相应的增删改查函数接口。将其他需要进行数据库操作的 Activity 继承于基 Activity。那么就可以调用基 Activity 的方法，从而实现数据库的增删改查操作。

2. 一开始的想法是直接调用 clear 函数实现全部删除的操作，然后再将数据库的内容进行重新赋值（因为数据库已经更新），但总觉得太过暴力，效率也不高。后来改变做法。增加改为使用 EventBus 传值回 MainActivity 然后调用 add 函数将添加的内容加入 list 中。删除直接调用 remove 函数，修改改为调用 set 函数设置对应位置的属性值修改。提高了效率（当数组大将更加明显）。

## 四、 课后实验结果

实现了动态申请权限，代码如下：

```
private void verifyStoragePermission(Activity activity)
{
    try {
        int permission = ActivityCompat.checkSelfPermission(activity,
            permission: "android.permission.READ_CONTACTS");
        if (permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(activity, new String[]
                {Manifest.permission.READ_CONTACTS}, requestCode: 1);
        }
        else {
            read_contacts = true; //记录可读可写内存卡
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    @NonNull int[] grantResults) {
    if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        read_contacts = true;
    }
    else {
        System.exit(status: 0);
    }
}
}
```

只有获得了相应的权限才能添加人物列表的内容，点击添加按钮才能跳转。

```
//添加按钮
Add.setOnClickListener((view) -> {
    Intent intent = new Intent(packageContext: MainActivity.this, AddItem.class);
    if (read_contacts) {
        startActivity(intent);
    }
});
```

## 五、 实验思考及感想

这次实验主要复习了 listview 的用法，学习了数据库和内容提供器的基本操作。总体而言，实验难度不是很大。通过这次实验，熟悉了数据库的增删改查基本操作。由于上过数据库的课程，对 SQL 查询过程还是比较熟悉，所以理解代码并不难，主要难点是不理解数据库在两个 Activity 中如何进行操作。一开始的想法是通过 Intent 将数据库对象进行传入，但是发现无法序列化。之后使用自定义基 Activity 的方法，其他 Activity 就可以使用基 Activity 的数据库和基本操作。后来明白在不同 Activity 中只要定义数据库的名字相同，由于同名的数据库不会再次创建（没有更新版本的情况下），就可以使用同一数据库的内容，并且可以对同一个数据库进行操作，反而更加简单了。

作业要求：

1. 命名要求: 学号\_姓名\_实验编号，例如 15330000\_林 XX\_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。