1. 数据存储的几种方式

1. 保存成一般File格式
2. SharedPerference
3. 数据库 当我们有大量相似结构的数据，并想实现增删查改，推荐使用数据库
4. 通过内容提供者 保存到别的应用的数据库里面 提供访问方式
5. 网络

如果想自己创建一个SharedPerference生成的文件

```
public static void saveXmlByStringBuilder(Context c, String passport,
        String password) throws Exception {
    // <map>
    // <string name="pwd">123</string>
    // <string name="username">5abc#12345</string>
    // </map>
    // 1.创建需要写入的字符串
    StringBuilder sb = new StringBuilder();
    sb.append("<map>");
    sb.append("<string name=\"pwd\">" + passport + "</string>");
    sb.append("<string name=\"username\">" + password + "</string>");
    sb.append("</map>");
    String result = sb.toString();
    // 2.设置将要写入的文件位置
    // 因为getFilesDir()在Context中定义 这里需要传入Context
    File file = new File(c.getFilesDir(), "qqinfo.xml");
    // 3.写入到流中（字符流）
    BufferedWriter writer = new BufferedWriter(new FileWriter(file));
    // 4.开始写入
    writer.write(result);
    writer.close();
}
```

2. Xml序列化器生成xml文件

XmlSerializer开发步骤:

1. 创建一个xml序列化工具
2. 设置产生的文件保存到某个位置
3. 开始创建文件
    - 开始写文档
    - 开始标签
    - 文本
    - 结束标签

- 结束文档

4. 代码

```
public static void saveXmlByXmlSerializer(Context c, String passport,
        String password) throws Exception {
    // 1.创建xmlSerializer 用来创建xml的序列化工具
    XmlSerializer xmlSerializer = Xml.newSerializer();
    // 2.设置文件输出到什么位置
    FileOutputStream fos = c.openFileOutput("qqinfo.xml",
            Context.MODE_PRIVATE);
    xmlSerializer.setOutput(fos, "utf-8");
    // <map>
    // <pwd>123</pwd>
    // <username>zhangsan</username>
    // </map>
    // 3.开始写入文档
    // <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
    xmlSerializer.startDocument("utf-8", true);

    xmlSerializer.startTag(null, "map");

    xmlSerializer.startTag(null, "pwd");
    xmlSerializer.text(password);
    xmlSerializer.endTag(null, "pwd");

    xmlSerializer.startTag(null, "username");
    xmlSerializer.text(passport);
    xmlSerializer.endTag(null, "username");

    xmlSerializer.endTag(null, "map");
    //4.结束写入文档 此时会将内存中的数据写入到硬盘中
    xmlSerializer.endDocument();
}
```

3. Pull解析器工作的过程

解析xml的几种方式：

1. Dom解析

   DOM（文档对象模型）是W3C标准，提供了标准的解析Xml方式，但其解析效率一直不尽如人意，这是因为DOM解析XML文档时，把所有内容一次性的装载入内存，并构建一个驻留在内存中的树状结构（节点数）。如果需要解析的XML文档过大，或者我们只对该文档中的一部分感兴趣，这样就会引起性能问题。

   特性：基于树状的xml的结构

支持：Dom4j dom4j是一个Java的XML API，它也是一个开放源代码的软件，是jdom的升级品，用来读写XML文件的。

2. SAX解析

   SAX（Simple API for XML）是一种XML解析的替代方法。相比于DOM，SAX是一种速度更快，更有效的方法，它既是一个接口，也是一个软件包.但作为接口，SAX是事件驱动型XML解析的一个标准接口不会改变 SAX的工作原理简单地说就是对文档进行顺序扫描，当扫描到文档（document）开始与结束、元素（element）开始与结束、文档（document）结束等地方时通知事件处理函数，由事件处理函数做相应动作，然后继续同样的扫描，直至文档结束。

3. Pull解析

   Pull是Android内置的xml解析器。Pull解析器的运行方式与SAX 解析器相似。它提供了类似的事件，如：开始元素和结束元素事件，使用parser.next()可以进入下一个元素并触发相应事件。事件将作为数值代码被发送。

Pull开发步骤：

1. 创建一个Pull解析器
2. 将文件读进来
3. 开始解析文档
   - 判断如果没有读取文档完毕 就循环遍历
   - 拿到开标签
   - 拿到开标签后面的文本 pullParser.nextText();
   - 移动游标

代码

```
public static String parseXmlFile(Context c) throws Exception {
    String result="";
    // 1.创建Pull解析器
    XmlPullParser pullParser = Xml.newPullParser();
    // 2.绑定需要解析的文件
    FileInputStream fis = c.openFileInput("qqinfo.xml");
    pullParser.setInput(fis, "utf-8");
    // 3.获取读取到的事件类型
    int eventType = pullParser.getEventType();
    //Log.v("520it", eventType + " ");
    // 3.2pullParser.next() 读取下一个事件类型
    //eventType = pullParser.next();
    // 3.3 pullParser.getName() 获取标签名
    //Log.v("520it", pullParser.getName() + " ");
    // 3.4 如果事件类型是text 那么getname则会返回null
    // 3.5 pullParser.nextText() 获取标签后面的文本
    while (eventType!=XmlPullParser.END_DOCUMENT) {
        //  <map>  <pwd>   <username>
        if (eventType==XmlPullParser.START_TAG) {
            if (pullParser.getName().equals("username")) {
                result+="   username="+pullParser.nextText());
            }else if (pullParser.getName().equals("pwd")) {
                result+="   pwd="+pullParser.nextText());
            }
        }
        eventType = pullParser.next();
    }

    return result;
}
```

4. 如何创建数据库

Android下的Sqlite数据库

SQLite，是一款轻型的数据库，它包含在一个相对小的C库中。它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，SQLite第一个Alpha版本诞生于2000年5月。 至2016年已经有16个年头，SQLite也迎来了一个版本 SQLite 3已经发布。

如何创建数据库

- 创建数据库帮助类（构造器）

```
  //数据库表文件
  public static final String DB_NAME="contact.db";
  //数据库版本
  public static final int DB_VERSION=1;

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

- 创建一般文件

```
  File file=new File(getFilesDir(),"mock.txt");
  file.createNewFile();
```

- 创建数据库文件

```
  MyOpenHelper helper=new MyOpenHelper(this);
  helper.getReadableDatabase();
```

- 创建数据库表

```
  @Override
  //数据库文件创建的时候调用该方法
  public void onCreate(SQLiteDatabase db) {
      System.out.println("onCreate");
      //创建数据库表
      db.execSQL("create table contactinfo(_id integer primary key
autoincrement," +"username varchar(20),phone varchar(15));");
  }
```

- 数据库更新

```
  //数据库更新 只要跟之前的数据库版本不一样 则会调用 该方法可以修改数据库表的各种数
据
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
      Log.v("520it", "onUpgrade"+"  oldVersion="
                  +oldVersion+" newVersion="+newVersion);
  }
```

保存数据库文件：/data/data/包名/databases/xxx.db

5. 数据库增删改查的SQL语句

增加（插入联系人 手机号到数据库）：

```
insert into contactinfo (username,phone) values('zhangsan','150188888888');
```

修改（设置zhangsan的手机为1501111111）：

```
update contactinfo set phone='150111111111' where username='zhangsan';
```

删除(删除那个叫李四的联系人):

```
delete from contactinfo where username='lisi';
```

查询（查询电话为15022222222的用户）：

```
select username,phone from contactinfo where phone='15022222222';
```

6. 数据库增删改查代码实现

开发步骤:

- 首先在Sqlite expert工具中确保SQL语句正常运行.
- 创建Dao层 封装你需要的Dao业务（CURD）
- 创建界面去调用

代码

```
public class ContactDao {

    private DbOpenHelper mHelper;

    public ContactDao(Context c) {
        mHelper=new DbOpenHelper(c);
    }

    /**
```

```
     * 添加一个联系人
     */
    public void insertContact(String username,String phone){
        SQLiteDatabase db = mHelper.getWritableDatabase();
        db.execSQL("insert into
contactinfo("+DbCons.COLUMN_USERNAME+","+DbCons.COLUMN_PHONE+")
values(?,?);",
                new String[]{username,phone});
    }

    /**
     *  对某个电话号码里面的联系人改名
     */
    public void updateContact(String phone,String newUsername){
        SQLiteDatabase db = mHelper.getWritableDatabase();
        db.execSQL("update contactinfo set "+DbCons.COLUMN_USERNAME+"=?
where "+DbCons.COLUMN_PHONE+"=?;"
                ,new String[]{newUsername,phone});
    }

    /**
     * 查询某个联系人的信息
     */
    public void queryContact(String username){
        SQLiteDatabase db = mHelper.getWritableDatabase();
        Cursor cursor = db.rawQuery("select "+DbCons.COLUMN_PHONE+" from
contactinfo where "+DbCons.COLUMN_USERNAME+"=?;",
                new String[]{username});
//      cursor.moveToNext()  移动到下一行 如果有下一行 则返回true
        while (cursor.moveToNext()) {
            String phone = cursor.getString(0);
            Log.v("520it", phone);
        }
    }

    /**
     * 删除某个联系人
     */
    public void deleteContact(String username){
        SQLiteDatabase db = mHelper.getWritableDatabase();
        db.execSQL("delete from contactinfo where
"+DbCons.COLUMN_USERNAME+"=?;",new String[]{username}
    );
}
```

7. Sqlite3工具的使用

Sqlite3工具的位置：

..\android-adt-bundle\sdk\tools\sqlite3.exe

如何进入sqlite3命令行：

1. 先执行adb -s 模拟器名称 shell
2. linux shell命令： cd /data/data/包名/databases 到databases目录下 pwd 查看当前的目录位置

Sqlite3工具常用操作：

sqlite3 数据库文件名（进入了数据库）

查看该文件下的所有数据库表 .tables

查看某个数据库表结构 select * from 表名;

退出sqlite3 .quit

8. 数据库增删改查的Google实现

- 插入联系人名称和电话号码

```
public boolean insertContact(String username,String phone){
    SQLiteDatabase db = mHelper.getWritableDatabase();
    //nullColumnHack  _id  username  phone  指定该列对应字段的值不能为空
    //values  将数据通过数据库字段保存到数据库里面
    ContentValues values=new ContentValues();
    values.put(DbCons.COLUMN_USERNAME, username);
    values.put(DbCons.COLUMN_PHONE, phone);
    long insert = db.insert(DbCons.TABLE_NAME,null, values);
    return insert!=-1;
}
```

- 修改电话为phone的联系人用户名为：

```
  public boolean updateContact(String phone,String newUsername){
      SQLiteDatabase db = mHelper.getWritableDatabase();
      ContentValues values=new ContentValues();
      values.put(DbCons.COLUMN_USERNAME, newUsername);
      //whereArgs 绑定的值
      //返回 修改了n行     就返回多少n
      int update = db.update(DbCons.TABLE_NAME, values,"phone=?", new
String[]{phone});
      return update>0;
  }
```

- 插入某个用户的电话号码

```
  public String queryContact(String username){
      SQLiteDatabase db = mHelper.getWritableDatabase();
      //columns 结果集返回了几列
      Cursor cursor = db.query(DbCons.TABLE_NAME, new String[]
{DbCons.COLUMN_PHONE},
              DbCons.COLUMN_USERNAME+"=?", new String[]{username},
null, null, null);
      //当返回的结果确定只有一行数据
      if (cursor.moveToFirst()) {
      //  通过字段名 返回字段的索引
          int columnIndex =
cursor.getColumnIndex(DbCons.COLUMN_PHONE);
      //  columnIndex 通过索引返回某一行所在的列的值
          return cursor.getString(columnIndex);
      }
      return "";
  }
```

- 删除某个联系人

```
  public boolean deleteContact(String username){
      SQLiteDatabase db = mHelper.getWritableDatabase();
      int rowEffected = db.delete(DbCons.TABLE_NAME,
DbCons.COLUMN_USERNAME+"=?", new String[]{username});
      return rowEffected>0;
  }
```