

4.5.2 Intent之复杂数据的传递

分类 [Android 基础入门教程](#)

Android 基础入门教程(Q群号 : 153836263)

本节引言：

上一节中我们学习了Intent的一些基本使用，知道了Intent的七个属性，显式Intent以及 隐式Intent，以及如何自定义隐式Intent，最后还给大家提供了一些常用的系统Intent！而本节跟大家讲解的是Intent传递数据的问题~好的，开始本节内容~

1.Intent传递简单数据

还记得我们在Activity那里学过如何在两个Activity中互相传递简单数据的方法吗？



就是可以直接通过调用Intent的putExtra()方法存入数据，然后在获得Intent后调用getXxxExtra获得 对应类型的数据；传递多个的话，可以使用Bundle对象作为容器，通过调用Bundle的putXxx先将数据 存储到Bundle中，然后调用Intent的putExtras()方法将Bundle存入Intent中，然后获得Intent以后， 调用getExtras()获得Bundle容器，然后调用其getXXX获取对应的数据！ 另外数据存储有点类似于Map的<键，值>！

2.Intent传递数组

嘿嘿，普通类型倒没问题，但是如果是数组咧？解决方法如下：

写入数组：

```
bd.putStringArray("StringArray", new String[] {"呵呵", "哈哈"});
```

1.0 Android基础入门教程

1.0.1 2015年最新Android基...

1.1 背景相关与系统架构分析

1.2 开发环境搭建

1.2.1 使用Eclipse + ADT + S...

1.2.2 使用Android Studio开...

1.3 SDK更新不了问题解决

1.4 Genymotion模拟器安装

1.5.1 Git使用教程之本地仓...

1.5.2 Git之使用GitHub搭建...

1.6 .9(九妹)图片怎么玩

1.7 界面原型设计

1.8 工程相关解析(各种文件...

1.9 Android程序签名打包

1.11 反编译APK获取代码&...

2.1 View与ViewGroup的概念

2.2.1 LinearLayout(线性布局)

2.2.2 RelativeLayout(相对布...

2.2.3 TableLayout(表格布局)

2.2.4 FrameLayout(帧布局)

2.2.5 GridLayout(网格布局)

2.2.6 AbsoluteLayout(绝对...

2.3.1 TextView(文本框)详解

2.3.2 EditText(输入框)详解

2.3.3 Button(按钮)与ImageB...

2.3.4 ImageView(图像视图)

2.3.5.RadioButton(单选按钮...

2.3.6 开关按钮ToggleButton...

2.3.7 ProgressBar(进度条)

2.3.8 SeekBar(拖动条)

2.3.9 RatingBar(星级评分条)

2.4.1 ScrollView(滚动条)

```
//可把StringArray换成其他数据类型,比如int,float等等...
```

读取数组：

```
String[] str = bd.getStringArray("StringArray")
```

3.Intent传递集合

嗯，数组很简单吧，那我们再来传下集合~这个就稍微复杂点了，分情况处理：

1) List<基本数据类型或String>

写入集合：

```
intent.putStringArrayListExtra(name, value)
intent.putIntegerArrayListExtra(name, value)
```

读取集合：

```
intent.getStringArrayListExtra(name)
intent.getIntegerArrayListExtra(name)
```

2) List< Object>

将list强转成Serializable类型,然后传入(可用Bundle做媒介)

写入集合：

```
putExtras(key, (Serializable) list)
```

读取集合：

```
(List<Object>) getIntent().getSerializable(key)
```

PS:Object类需要实现Serializable接口

3) Map<String, Object>,或更复杂的

解决方法是：外层套个List

```
//传递复杂些的参数
Map<String, Object> map1 = new HashMap<String, Object>();
map1.put("key1", "value1");
map1.put("key2", "value2");
List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
list.add(map1);
```

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定...

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextVie...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTo...

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(Co...

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

```
Intent intent = new Intent();
intent.setClass(MainActivity.this, ComplexActivity.class);
Bundle bundle = new Bundle();

//须定义一个list用于在bundle中传递需要传递的ArrayList<Object>,这个是必须要的
ArrayList bundlelist = new ArrayList();
bundlelist.add(list);
bundle.putParcelableArrayList("list", bundlelist);
intent.putExtras(bundle);
startActivity(intent);
```

4.Intent传递对象

传递对象的方式有两种：将对象转换为Json字符串或者通过Serializable,Parcelable序列化 不建议使用Android内置的抠脚Json解析器，可使用fastjson或者Gson第三方库！

1) 将对象转换为Json字符串

Gson解析的例子：

Model:

```
public class Author{
    private int id;
    private String name;
    //...
}

public class Author{
    private int id;
    private String name;
    //...
}
```

写入数据：

```
Book book=new Book();
book.setTitle("Java编程思想");
Author author=new Author();
author.setId(1);
author.setName("Bruce Eckel");
book.setAuthor(author);
Intent intent=new Intent(this, SecondActivity.class);
intent.putExtra("book", new Gson().toJson(book));
startActivity(intent);
```

读取数据：

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

```
String bookJson=getIntent().getStringExtra("book");
Book book=new Gson().fromJson(bookJson,Book.class);
Log.d(TAG,"book title->"+book.getTitle());
Log.d(TAG,"book author name->"+book.getAuthor().getName());
```

2) 使用Serializable,Parcelable序列化对象

1.Serializable实现:

- ①业务Bean实现：Serializable接口,写上getter和setter方法
- ②Intent通过调用putExtra(String name, Serializable value)传入对象实例 当然对象有多个的话多个的话,我们也可以先
Bundle.putSerializable(x,x);
- ③新Activity调用getSerializableExtra()方法获得对象实例:
eg:Product pd = (Product)
getIntent().getSerializableExtra("Product");
- ④调用对象get方法获得相应参数

2.Parcelable实现:

一般流程:

- ①业务Bean继承Parcelable接口,重写writeToParcel方法,将你的对象序列化为一个Parcel对象;
- ②重写describeContents方法,内容接口描述,默认返回0就可以
- ③实例化静态内部对象CREATOR实现接口Parcelable.Creator
- ④同样通过Intent的putExtra()方法传入对象实例,当然多个对象的话,我们可以先 放到Bundle里Bundle.putParcelable(x,x),再
Intent.putExtras()即可

一些解释:

通过writeToParcel将你的对象映射成Parcel对象,再通过
createFromParcel将Parcel对象映射 成你的对象。也可以将Parcel看成是一个流,通过writeToParcel把对象写到流里面, 在通过
createFromParcel从流里读取对象,只不过这个过程需要你来实现,因此写的 顺序和读的顺序必须一致。

实现Parcelable接口的代码示例:

```
//Internal Description Interface,You do not need to manage
@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel parcel, int flags){
```

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...

```
parcel.writeString(bookName);
parcel.writeString(author);
parcel.writeInt(publishTime);
}

public static final Parcelable.Creator<Book> CREATOR = new Creator<Book>() {
    @Override
    public Book[] newArray(int size) {
        return new Book[size];
    }

    @Override
    public Book createFromParcel(Parcel source) {
        Book mBook = new Book();
        mBook.bookName = source.readString();
        mBook.author = source.readString();
        mBook.publishTime = source.readInt();
        return mBook;
    }
};
```

Android Studio生成Parcelable插件：

IntelliJ/Android Studio插件android-parcelable-intellij-plugin 只要ALT+Insert，即可直接生成Parcelable接口代码。

另外：Android中大量用到Parcelable对象，实现Parcelable接口又是非常繁琐的，可以用到 第三方的开源框架:Parceler,因为Maven的问题,暂时还没试。

参考地址:[\[Android的Parcelable自动生成\]](#)

3.两种序列化方式的比较：

两者的比较:

- 1) 在使用内存的时候，Parcelable比Serializable性能高，所以推荐使用Parcelable。
- 2) Serializable在序列化的时候会产生大量的临时变量，从而引起频繁的GC。
- 3) Parcelable不能使用在要将数据存储在磁盘上的情况，因为Parcelable不能很好的保证数据的 持续性在外界有变化的情况下。尽管Serializable效率低点，但此时还是建议使用Serializable。

5.Intent传递Bitmap

bitmap默认实现Parcelable接口,直接传递即可

实现代码：

```
Bitmap bitmap = null;
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putParcelable("bitmap", bitmap);
intent.putExtra("bundle", bundle);
```

6.传来传去不方便，直接定义全局数据

如果是传递简单的数据，有这样的需求，Activity1 -> Activity2 -> Activity3 -> Activity4，你想在Activity中传递某个数据到Activity4中，怎么破，一个个页面传么？

显然不科学是吧，如果你想某个数据可以在任何地方都能获取到，你就可以考虑使用 **Application全局对象**了！

Android系统在每个程序运行的时候创建一个Application对象，而且只会创建一个，所以Application 是单例(singleton)模式的一个类，而且Application对象的生命周期是整个程序中最长的，他的生命周期等于这个程序的生命周期。如果想存储一些比静态的值(固定不改变的，也可以变)，如果你想使用 Application就需要自定义类实现 Application类，并且告诉系统实例化的是我们自定义的Application而非系统默认的，而这一步，就是在AndroidManifest.xml中卫我们的application标签添加:**name属性**！

关键部分代码：

1) 自定义Application类：

```
class MyApp extends Application {
    private String myState;
    public String getState() {
        return myState;
    }
    public void setState(String s) {
        myState = s;
    }
}
```

2) AndroidManifest.xml中声明：

```
<application android:name=".MyApp" android:icon="@drawable/icon"
"
    android:label="@string/app_name">
```

3) 在需要的地方调用：

```
class Blah extends Activity {
    @Override
    public void onCreate(Bundle b) {
        ...
        MyApp appState = ((MyApp)getApplicationContext());
        String state = appState.getState();
        ...
    }
}
```

高逼格写法

：在任何位置都能获取到Application全局对象。

Applicaiton是系统的一个组件，他也有自己的一个生命周期，我们可以在onCraete里获得这个 Application对象。贴下修改后的代码吧！

```
class MyApp extends Application {  
    private String myState;  
    private static MyApp instance;  
  
    public static MyApp getInstance() {  
        return instance;  
    }  
  
    public String getState() {  
        return myState;  
    }  
    public void setState(String s) {  
        myState = s;  
    }  
  
    @Override  
    public void onCreate() {  
        onCreate();  
        instance = this;  
    }  
}
```

然后在任意地方我们就可以直接调用：MyApp.getInstance（）来获得Application的全局对象！

注意事项：

Application对象是存在于内存中的，也就有它可能会被系统杀死，比如这样的场景：

我们在Activity1中往application中存储了用户账号，然后在Activity2中获取到用户账号，并且显示！

如果我们点击home键，然后过了N久候，系统为了回收内存kill掉了我们的app。这个时候，我们重新 打开这个app，这个时候很神奇的，回到了Activity2的页面，但是如果这个时候你再去获取Application 里的用户账号，程序就会报NullPointerException，然后crash掉~

之所以会发生上述crash，是因为这个Application对象是全新创建的，可能你以为App是重新启动的，其实并不是，仅仅是创建一个新的Application，然后启动上次用户离开时的Activity，从而创造App并没有被杀死的假象！所以如果是比较重要的数据的话，建议你还是进行本地化，另外在使用数据的时候 要对变量的值进行非空检查！还有一点就是：不止是Application变量会这样，单例对象以及公共静态变量 也会这样~

7.单例模式传参

上面的Application就是基于单例的，单例模式的特点就是可以保证系统中一个类有且只有一个实例。这样很容易就能实现，在A中设置参数，在B中直接访问了。这是几种方法中效率最高的。

范例代码：(代码来自于网上~)

①定义一个单例类：

```
public class XclSingleton
{
    //单例模式实例
    private static XclSingleton instance = null;

    //synchronized 用于线程安全，防止多线程同时创建实例
    public synchronized static XclSingleton getInstance() {
        if(instance == null) {
            instance = new XclSingleton();
        }
        return instance;
    }

    final HashMap<String, Object> mMap;
    private XclSingleton()
    {
        mMap = new HashMap<String, Object>();
    }

    public void put(String key, Object value) {
        mMap.put(key, value);
    }

    public Object get(String key)
    {
        return mMap.get(key);
    }
}
```

②设置参数:

```
XclSingleton.getInstance().put("key1", "value1");
XclSingleton.getInstance().put("key2", "value2");
```

本节小结：

好的，关于Intent复杂数据传输就到这里，本节除了讲述使用Intent来传递复杂数据外，还教大家使用Application和单例模式来传递参数！相信会对大家在数据传递方面带来方便，谢谢~

← 4.5.1 Intent的基本使用

5.1 Fragment基本概述 →

阿里云

阿里云 嘉年华 2015

秒杀! 1折抢!

阿里云 嘉年华

续费\新购 满就送最高 ¥1000

立享优惠

	<div>在线实例</div> <div><div>· HTML 实例</div><div>· CSS 实例</div><div>· JavaScript 实例</div><div>· Ajax 实例</div><div>· jQuery 实例</div><div>· XML 实例</div><div>· Java 实例</div></div>	<div>字符集&工具</div> <div><div>· HTML 字符集设置</div><div>· HTML ASCII 字符集</div><div>· HTML ISO-8859-1</div><div>· HTML 实体符号</div><div>· HTML 拾色器</div><div>· JSON 格式化工具</div></div>	<div>最新更新</div> <div><div>· Swift 正式开源</div><div>· PHP 7 正式发布</div><div>· Shell 编程快速入门</div><div>· Shell 文件包含</div><div>· Shell 输入/输出...</div><div>· Shell printf 命令</div><div>· Shell 基本运算符</div></div>	<div>站点信息</div> <div><div>· 意见反馈</div><div>· 免责声明</div><div>· 关于我们</div><div>· 文章归档</div></div>
			<div><div>^</div><div>二维码</div><div>反馈</div></div> <div>官方微信</div> <div></div>	<div>Copyright © 2013-2015 菜鸟教程 runoob.com All Rights Reserved. 备案号：闽ICP备15012807号-1</div>