# Android数据传递方法汇总

Android开发中，在不同模块(如Activity)间经常会有各种各样的数据需要相互传递，我把常用的几种 方法都收集到了一起。它们各有利弊，有各自的应用场景。

## 利用Intent对象携带简单数据

利用Intent的Extra部分来存储我们想要传递的数据，可以传送int, long, char等一些基础类型,对复杂的对象就无能为力了。

**传递参数代码如下：**

Intent intentSimple = new Intent();

intentSimple.setClass(MainActivity.this,SimpleActivity.class);

Bundle bundleSimple = new Bundle();

bundleSimple.putString("usr", "xcl");

bundleSimple.putString("pwd", "zj");

intentSimple.putExtras(bundleSimple);

startActivity(intentSimple);

**接受参数代码如下：**

Bundle bunde = this.getIntent().getExtras();

String eml = bunde.getString("usr");

String pwd = bunde.getString("pwd");

## 利用Intent对象携带如ArrayList之类复杂些的数据

**传递参数代码如下：**

//传递复杂参数  
Map<String,Object> map=new HashMap<String,Object>();  
map.put("username", "tom");  
Map<String,Object> map1=new HashMap<String,Object>();  
map1.put("username", "mike");  
ArrayList list=new ArrayList();//这里需要使用ArrayList声明  
 list.add(map);  
 list.add(map1);  
bundleSimple.putParcelableArrayList("list", list);  
intentSimple.putExtras(bundleSimple);//参数放入Intent中  
startActivity(intentSimple);//跳转到另一个页面

**接受参数代码如下：**

Bundle bunde = this.getIntent().getExtras();  
  
 ArrayList list= bunde.getParcelableArrayList("list");  
 System.out.println("\*\*2\*\*\*"+list);

## 利用Java语言本身的特性，通过将数据序列化后，再将其传递出去。

**传递参数代码如下：**

//通过Serializable接口传参数的例子

HashMap<String,String> map2 = new HashMap<String,String>();

map2.put("key1", "value1");

map2.put("key2", "value2");

Bundle bundleSerializable = new Bundle();

bundleSerializable.putSerializable("serializable", map2);

Intent intentSerializable = new Intent();

intentSerializable.putExtras(bundleSerializable);

intentSerializable.setClass(MainActivity.this,

                            SerializableActivity.class);

startActivity(intentSerializable);

**接受参数代码如下：**

HashMap<String,String> map = (HashMap<String,String>)bunde.getSerializable("serializable");

## 通过实现Parcelable接口

这个是通过实现Parcelable接口，把要传的数据打包在里面，然后在接收端自己分解出来。这个是Android独有的，在其本身的源码中也用得很多， 效率要比Serializable相对要好。

首先要定义一个类，用于 实现Parcelable接口 ，因为其本质也是序列化数据，所以这里要注意定义顺序要与解析顺序要一致。

**存储数据的工具类定义代码如下：**

**public class** PracelableBean **implements** Parcelable {  
 *//定义要被传输的数据* **public int age**;  
 **public** String **name**;  
 **public** HashMap<String,String> **mMap** = **new** HashMap<String,String>();  
 **public** PracelableBean(){  
  
 }  
  
 **protected** PracelableBean(Parcel in) {  
 *//将映射在Parcel对象中的数据还原回来  
 //警告，这里顺序一定要和writeToParcel中定义的顺序一致才行!!!* **age** = in.readInt();  
 **name** = in.readString();  
 **mMap** = in.readHashMap(HashMap.**class**.getClassLoader());  
 }  
  
 **public static final** Creator<PracelableBean> ***CREATOR*** = **new** Creator<PracelableBean>() {  
 @Override  
 **public** PracelableBean createFromParcel(Parcel in) {  
 **return new** PracelableBean(in);  
 }  
  
 @Override  
 **public** PracelableBean[] newArray(**int** size) {  
 **return new** PracelableBean[size];  
 }  
 };  
  
 @Override  
 **public int** describeContents() {  
 **return** 0;  
 }  
  
 @Override  
 **public void** writeToParcel(Parcel dest, **int** flags) {  
 *//等于将数据映射到Parcel中去* dest.writeInt(**age**);  
 dest.writeString(**name**);  
 dest.writeMap(**mMap**);  
 }  
}

**传递参数代码如下：**

PracelableBean xp = **new** PracelableBean();  
xp.**age** = 18;  
xp.**name** = **"范冰冰"**;  
xp.**mMap** = **new** HashMap<String,String>();  
xp.**mMap**.put(**"接口参数"**, **"哈哈"**);  
intentSimple.putExtra(**"Parcelable"**, xp);

**接受参数代码如下：**

PracelableBean xp = **this**.getIntent().getParcelableExtra(**"Parcelable"**);  
System.***out***.println(**"\*\*\*\*\*\*\*4\*\*\*\*\*\*\*\*\*"**+xp.**name**+**":"**+xp.**age**+**":"**+xp.**mMap**);

## 通过单例模式实现参数传递

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。这样很容易实现， 在一个Activity中设置参数，在另一个Activity中直接访问就能过获取参数。这是几种方法中效率最高的。

**定义一个单实例的类来保存参数：**

**public class** SingletonBean {  
 *//单例模式实例* **private static** SingletonBean *instance* = **null**;  
  
 *//synchronized 用于线程安全，防止多线程同时创建实例* **public synchronized static** SingletonBean getInstance(){  
 **if**(*instance* == **null**){  
 *instance* = **new** SingletonBean ();  
 }  
 **return** *instance*;  
 }  
  
 **final** HashMap<String, Object> **mMap**;  
 **public** SingletonBean ()  
 {  
 **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);  
 }  
}

**传递参数代码如下：**

SingletonBean.getInstance().put("key1", "value1");

SingletonBean.getInstance().put("key2", "value2");

Intent intentSingleton = new Intent();

intentSingleton.setClass(MainActivity.this,

                        SingletonActivity.class);

startActivity(intentSingleton);

**接受参数代码如下：**

  HashMap<String,Object> map = SingletonBean.getInstance().mMap;