### 源码分析相关面试题

* [Volley源码分析](http://www.jianshu.com/p/ec3dc92df581)
* [注解框架实现原理](http://www.jianshu.com/p/20da6d6389e1)
* [okhttp3.0源码分析](http://www.jianshu.com/p/9ed2c2f2a52c)
* [onSaveInstanceState源码分析](http://www.jianshu.com/p/cbf9c3557d64)
* [静默安装和源码编译](http://www.jianshu.com/p/2211a5b3c37f)

### Activity相关面试题

* [保存Activity的状态](http://www.jianshu.com/p/cbf9c3557d64)

### 与XMPP相关面试题

* [XMPP协议优缺点](http://www.jianshu.com/p/2c04ac3c526a)
* [极光消息推送原理](http://www.jianshu.com/p/d88dc66908cf)

### 与性能优化相关面试题

* [内存泄漏和内存溢出区别](http://www.jianshu.com/p/5dd645b05c76)
* [UI优化和线程池实现原理](http://www.jianshu.com/p/c22398f8587f)
* [代码优化](http://www.jianshu.com/p/ebd41eab90df)
* [内存性能分析](http://www.jianshu.com/p/2665c31b9c2f)
* [内存泄漏检测](http://www.jianshu.com/p/1514c7804a06)
* [App启动优化](http://www.jianshu.com/p/f0f73fefdd43)
* [与IPC机制相关面试题](http://www.jianshu.com/p/de4793a4c2d0)

### 与登录相关面试题

* [oauth认证协议原理](http://www.jianshu.com/p/2a6ecbf8d49d)
* [token产生的意义](http://www.jianshu.com/p/9b7ce2d6c195)
* [微信扫一扫实现原理](http://www.jianshu.com/p/a9d1f21bd5e0)

### 与开发相关面试题

* [迭代开发的时候如何向前兼容新旧接口](http://www.jianshu.com/p/cbecadec98de)
* [手把手教你如何解决as jar包冲突](http://www.jianshu.com/p/30fdc391289c)
* [context的原理分析](http://www.jianshu.com/p/2706c13a1769)
* [解决ViewPager.setCurrentItem中间很多页面切换方案](http://www.jianshu.com/p/38ab6d856b56)

### 与人事相关面试题

* [人事面试宝典](http://www.jianshu.com/p/d61b553ff8c9)

经常有人问，后台的activity被系统自动回收的话，怎么回到界面的时候恢复数据，通过一个真实案例给大家讲讲如何保存状态，然后带着大家分析onSaveInstanceState的源码。



img

#### **当前页面侧滑菜单指向专题，用户做了如下操作：**

1）当用户按下HOME键时。 2）长按HOME键，选择运行其他的程序时。 3）按下电源按键（关闭屏幕显示）时。 4）从activity A中启动一个新的activity时。 5）屏幕方向切换时，例如从竖屏切换到横屏时。

失去焦点，activity很可能被进程终止！被KILL掉了，这时候就需要能保存当前的状态，不然下次用户再次进来看到的还是新闻，这样用户体验就不够好，代码有删减，我自己项目就这样使用的，解决方案如下：

@Override  
public void onSaveInstanceState(Bundle outState) {  
 outState.putInt("newsCenter\_position", newsCenterPosition);  
 outState.putInt("smartService\_position", smartServicePosition);  
 outState.putInt("govAffairs\_position", govAffairsPosition);  
 super.onSaveInstanceState(outState);  
}

#### 如上代码可知：

1）界面被回收之后调用onSaveInstanceState方法保存当前的状态，每个侧滑菜单选项都有一个位置。

@Override  
public void onCreate(Bundle savedInstanceState) {  
if (savedInstanceState != null  
 && savedInstanceState.containsKey("newsCenter\_position")) {  
 newsCenterPosition = savedInstanceState  
 .getInt("newsCenter\_position");  
 smartServicePosition = savedInstanceState  
 .getInt("smartService\_position");  
 govAffairsPosition = savedInstanceState  
 .getInt("govAffairs\_position");  
}  
  
 super.onCreate(savedInstanceState);  
}

#### **由以上代码可知：**

1）判断当前Bundle 是否有刚刚我们保存的位置，如果不为空，从当前的Bundle取出来，给每一个位置赋值。

public void switchMenu(int type) {   
switch (type) {  
 case NEWS\_CENTER:  
 if (newsCenterAdapter == null) {  
 newsCenterAdapter = new MenuAdapter(ct, newsCenterMenu);  
 newsCenterclassifyLv.setAdapter(newsCenterAdapter);  
 } else {  
 newsCenterAdapter.notifyDataSetChanged();  
 }  
 newsCenterAdapter.setSelectedPosition(newsCenterPosition);  
 break;  
 case SMART\_SERVICE:  
 if (smartServiceAdapter == null) {  
 smartServiceAdapter = new MenuAdapter(ct, smartServiceMenu);  
 smartServiceclassifyLv.setAdapter(smartServiceAdapter);  
 } else {  
 smartServiceAdapter.notifyDataSetChanged();  
 }  
 smartServiceAdapter.setSelectedPosition(smartServicePosition);  
 break;  
 case GOV\_AFFAIRS:  
 if (govAffairsAdapter == null) {  
 govAffairsAdapter = new MenuAdapter(ct, govAffairsMenu);  
 govAffairsclassifyLv.setAdapter(govAffairsAdapter);  
 } else {  
 govAffairsAdapter.notifyDataSetChanged();  
 }  
 govAffairsAdapter.setSelectedPosition(govAffairsPosition);  
 break;  
}

#### **以上代码可知：**

1）根据当前的位置设置到adapter当中，这样下次用户进来就还是专题了。

#### **总结下savedInstanceState的使用，代码如下：**

public class MainActivity extends Activity {   
  
 @Override   
 protected void onCreate(Bundle savedInstanceState) {   
 super.onCreate(savedInstanceState);   
 setContentView(R.layout.activity\_main);   
 if(savedInstanceState != null)   
 System.out.println("onCreate() : " + savedInstanceState.getString("octopus"));   
 }   
  
 @Override   
 protected void onRestoreInstanceState(Bundle savedInstanceState) {   
 super.onRestoreInstanceState(savedInstanceState);   
 System.out.println("onRestoreInstanceState() : " + savedInstanceState.getString("octopus"));   
 }   
  
 @Override   
 protected void onSaveInstanceState(Bundle outState) {   
 super.onSaveInstanceState(outState);   
  
 outState.putString("octopus", "www.baidu.com");   
 System.out.println("onSaveInstanceState() : save date www.baidu.com");   
 }   
  
}

#### **横竖屏切换，打印结果如下：**

I/System.out( 8167): onSaveInstanceState() : save date www.baidu.com   
I/System.out( 8167): onCreate() : www.baidu.com   
I/System.out( 8167): onRestoreInstanceState() : www.baidu.com

从打印结果可以看出来，当前Activity被系统回收之后,会调用onSaveInstanceState()保存状态，然后在activity判断bundler是否有当前状态，如果只是到这，估计你们就会吐槽没啥含金量，没办法硬着头皮上，接着咱们来分onSaveInstanceState()源码，请看如下代码：

@Override  
 public void onSaveInstanceState(Bundle outState) {  
 super.onSaveInstanceState(outState);  
  
 }

#### **以上代码可知**

1）调用父类Activity源码里面的onSaveInstanceState方法，代码如下：

protected void onSaveInstanceState(Bundle outState) {  
 outState.putBundle(WINDOW\_HIERARCHY\_TAG, mWindow.saveHierarchyState());  
 Parcelable p = mFragments.saveAllState();  
 if (p != null) {  
 outState.putParcelable(FRAGMENTS\_TAG, p);  
 }  
 ......  
}

#### **以上代码可知**

1）outState.put一个tag调用了mWindow里面的saveHierarchyState方法，继续分析Window源代码。

2）window是抽象类调用子类PhoneWindow里面的saveHierarchyState方法代码如下：

@Override  
public Bundle saveHierarchyState() {  
 Bundle outState = new Bundle();  
 if (mContentParent == null) {  
 return outState;  
 }  
  
 SparseArray<Parcelable> states = new SparseArray<Parcelable>();  
 mContentParent.saveHierarchyState(states);  
 outState.putSparseParcelableArray(VIEWS\_TAG, states);  
 ......  
 return outState;  
}

#### **以上代码可知**

1 ) Bundle outState = new Bundle()初始化Bundle对象，Bundle实现了Parcelable接口。

2）states = new SparseArray()并且把自己放到outState当中。

3）mContentParent.saveHierarchyState(states)，整个View树的顶层视图保存了层级状态代码如下：

public void saveHierarchyState(SparseArray<Parcelable> container) {  
 dispatchSaveInstanceState(container);  
}

#### 以上代码可知：

1）调相应的dispatchSaveInstanceState方法，代码如下：

protected void dispatchSaveInstanceState(SparseArray<Parcelable> container) {  
 if (mID != NO\_ID && (mViewFlags & SAVE\_DISABLED\_MASK) == 0) {  
 mPrivateFlags &= ~PFLAG\_SAVE\_STATE\_CALLED;  
 Parcelable state = onSaveInstanceState();  
 if ((mPrivateFlags & PFLAG\_SAVE\_STATE\_CALLED) == 0) {  
 throw new IllegalStateException(  
 "Derived class did not call super.onSaveInstanceState()");  
 }  
 if (state != null) {  
 // Log.i("View", "Freezing #" + Integer.toHexString(mID)  
 // + ": " + state);  
 container.put(mID, state);  
 }  
 }  
 }

#### 以上代码可知：

1） mID != NO\_ID 判断一个View必须有一个id，也就是说你要么在xml里通过android:id指定要么在代码里通过setId，但是你从如上代码压根是看不出来谷歌想干啥，必须全局搜索NO\_ID 和 mID ，一般在源码里面都会有谷歌工程师的注释方便我们理解，搜索NO\_ID 可知代码如下：

/\*\*  
 \* Used to mark a View that has no ID.  
 \*/  
 public static final int NO\_ID = -1;

原来NO\_ID用来标记没有id的View，搜索mID可知原来在如下代码赋值

public void setId(@IdRes int id) {  
 mID = id;  
 if (mID == View.NO\_ID && mLabelForId != View.NO\_ID) {  
 mID = generateViewId();  
 }  
 }

经常当我们看不懂谷歌源码的时候，可以通过曲线救国的方式，看看英文注释，看看源码哪个地方用到当前的类或者方法或者变量，这样就好理解了，好了扯远了，继续分析代码；

2）通过if判断，检测子类是否调用父类的onSaveInstanceState()方法，否则会抛异常，突然看到这才明白，还记得刚刚开始学Android的时候，经常一不小心就把代码里面的super.onCreate(savedInstanceState);这行代码删掉，报了错误还看不懂，原来系统在这里检测了，都怪自己曾经太年轻。

3）container.put(mID, state)这行代码，将state放进SparseArray中，以view自身的id为key，并且从注释来看打印mID的Hex值用来保证每页的id必须是唯一的，难怪每当我给view取id的时候，一个页面有重复的id就会报错，谷歌大婶在这里做判断了，腻害了word哥，总是百思不得其姐，凭啥不让我共用id(因为取名字太难了)，原来是想把id做为key来使用。

4）走到这onSaveInstanceState()，调用如下代码：

@CallSuper  
protected Parcelable onSaveInstanceState() {  
 mPrivateFlags |= PFLAG\_SAVE\_STATE\_CALLED;  
 ......  
 return BaseSavedState.EMPTY\_STATE;  
}

#### 以上代码可知：

1）设置位标志， 默认不save任何东西，状态为空，这就是为啥我们每次随便写个类继承activity实现onCreate方法的时候可以使用参数savedInstanceState保存状态，因为默认为null，代码如下：

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 savedInstanceState.putString("key","value");  
}

至此整个savedInstanceState保存状态源码分析完毕。

* 欢迎关注微信公众号,长期推荐技术文章和技术视频
* 微信公众号名称：Android干货程序员

img

img