0activity回顾
①创建新的activity
1.写一个类继承activity
2.重写onCreate() 以及其它生命周期方法(根据业务逻辑需求选择覆盖)
3.onCreate中做初始化操作
3.1初始化界面 setContentView(); findviewByID
3.2 初始化数据
文件 sharedpreferences 数据库 网络 从intent中(开启这个activity的intent)
3.3 初始化点击事件 (按钮设置点击事件 listview条目的点击事件)
4.在清单文件中声明对应的activity节点

②打开新的activity
隐式意图 匹配intent-filter 打开其他应用的activity
显示意图 指定路径 指定.class 对象 打开自己应用的activity
③activity之间数据的传递
Intent
3.1 A->B 同时把数据传递给B
基本数据类型 数组 实现了序列化接口的对象 及其数组和集合 Interger的ArrayList
A Activity中 intent.putExtra(key,value) intent.putExtras()
B Activity中 getIntent(); -> getXXXXExtra(key)

3.2 A打开B 在B关闭时候把数据传递给A
A 打开B的时候 startActivityForResult(intent, requestCode);
重写 onActivityResult(requestCode resultCode Intent data);
B Activity 如果有数据传回来 setResult(resultCode,intent data); finish();
A打开B 拍照 图片 视频
A打开C 扫描二维码 二维码 条形码

④activity生命周期
onCreate
onStart 走完onStart界面可见但是不能操作
onResume 走完onResume界面可见,能操作
onPause 走完onPause界面可见但是不能操作
onStop 不可见不可操作
onDestory释放资源

onRestart

⑤activity任务栈
每一个应用对应一个任务栈 activity的实例都是保存到任务栈中的

启动模式
launchMode
standard
singleTop 栈顶只有一个实例
SingleTask 栈中只有一个实例
SingleInstance 整个设备只有一个实例 独占任务栈
⑥屏幕旋转时activity的生命周期
默认先销毁再创建新的activity
6.1写死方向 screenOrientation portait landscape
6.2 configChanges ScreenSize|orientation|keyboardhidden
onConfigrationchanged

day11
四大组件都运行在主线程中 广播接收者也是运行在主线程中 不能执行耗时操作 如果一定要执行耗时操作 必须开子线程

1 广播接收者概念(broadcastreceiver)&为什么需要广播接收者

broadcastreceiver 用来接收 sendBroadCast方法 发出来的广播 可以通过intent传递数据 抽象类
广播特点 数据的传递方向 单向 调到固定的频率
作用范围有限 只在当前手机里有效 系统 把一些重要的操作 通过广播的形式通知给所有的应用

2 广播接收者案例_ip拨号器

①写一个类继承BroadCastRevceiver重写onReceive方法

```
1.  public class DailReceiver extends BroadcastReceiver {
2.
3.      @Override
4.      public void onReceive(Context context, Intent intent) {
5.          //获取用户输入的ip前缀
6.          SharedPreferences sp = context.getSharedPreferences("info", Context.MODE_P
    RIVATE);
7.          String prefix = sp.getString("prefix", "17951");
8.          //获取打电话的号码
9.          String number = getResultData();
10.         System.out.println("打电话了"+number);
11.         setResultData(prefix+number);
12.     }
13. }
```

②清单文件中注册 receiver节点 通过intent-filter 指定当前广播接收者要处理的广播事件

```
 <receiver android:name="com.itheima.ipdialer.DailReceiver">
            <intent-filter >
                <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
            </intent-filter>
        </receiver>
```

3 广播接收者案例_SD卡状态监听

需要监听的action sd卡状态变化的广播还需要加一个data scheme是file否则收不到广播

```
<action android:name="android.intent.action.MEDIA_UNMOUNTED"/>
<action android:name="android.intent.action.MEDIA_MOUNTED"/>
<data android:scheme="file"/>
```

```
public class SDCardStateReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        String action = intent.getAction();
        if("android.intent.action.MEDIA_UNMOUNTED".equals(action)){
            System.out.println("sdcard卸下来");
        }else if("android.intent.action.MEDIA_MOUNTED".equals(action)){
            System.out.println("sdCard被装上了");
        }
    }
}
```

4 广播接收者案例_短信监听

需要监听的

action`<action android:name="android.provider.Telephony.SMS_RECEIVED"/>`

```
public class SmsReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        System.out.println("来短信了");
        //pdu protocal data unit
        Object[] object = (Object[]) intent.getExtras().get("pdus");
        for(Object obj:object){
            //创建短信的消息对象
```

```
            SmsMessage message = SmsMessage.createFromPdu((byte[])obj);
            //获取短信的发送者
            String from = message.getOriginatingAddress();
            //获取消息内容
            String messageBody = message.getMessageBody();
            System.out.println("from"+from+"body"+messageBody);
            //12345
        }
    }

}
```

```
 <receiver android:name="com.itheima.smslistener.SmsReceiver">
            <intent-filter >
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

5 不同版本广播的特点

4.0之后 没有运行过的应用是不能收到广播的
在应用管理器中 force stop(强制停止)的应用 收不到广播

6 广播接收者案例_卸载安装

竞品分析
监听的动作

7 广播接收者案例_开机启动

监听的动作 `<action android:name="android.intent.action.BOOT_COMPLETED"/>`

需要的权限 `<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>`

```
public class BootReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        System.out.println("机器开启了");
```

```
        Intent intent2 = new Intent(context,MainActivity.class);
        //现在是在广播接收者中创建一个activity 当前应用没有任何activity在运行 所以不存在一个任务栈
        //需要通过指定一个flag 在创建activity的同时创建一个任务栈
        intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent2);
    }

}
```

8 有序广播和无序广播

发送无序广播 创建intent 设置action 通过sendBroadcast(intent) 就可以把广播发出去 当前的设备上 只要有广播接收者注册了相同的action 就可以收到广播 并且在发广播的时候 可以通过intent传递数据

```
public void sendbroadcast(View v){
        Intent intent = new Intent();
        intent.setAction("com.itheima.broadcast");
        intent.putExtra("key", "hello");
        sendBroadcast(intent);
    }
```

接收无序广播
注册广播接收者 指定对应的action就可以收到这个广播

```
<receiver android:name="com.itheima.receivercostumbroadcast.CostumReceiver">
            <intent-filter >
                <action android:name="com.itheima.broadcast"/>
            </intent-filter>
        </receiver>
```

如何区分有序广播 无序广播
接收到广播之后 在onReceive方法中调用abortbroadcast方法 如果没有异常说明是有序广播
如果 BroadcastReceiver trying to return result during a non-ordered broadcast 有这个异常说明是无序广播

	接收的顺序	是否可以中断	发送的方法
有序广播(可以修改广播内容)	可以通过priority 设置接收顺序	abortbroadcast();可以中断	sendOrderedbroadcast
无序广播(不可以修改)	大家一起收到	不可以中断	sendBroadCast()

9 特殊广播接收者

四大组件只有广播接收者可以不在清单文件中注册
静态注册 在清单文件中 通过声明一个receiver节点 指定intent-filter 这种方式就是静态注册

动态注册 `registerReceiver(receiver, filter);`
注销的方法 `unregisterReceiver(receiver);`

```
1.  public class MainActivity extends Activity {
2.
3.      private BroadcastReceiver receiver;
4.
5.
6.      @Override
7.      protected void onCreate(Bundle savedInstanceState) {
8.          super.onCreate(savedInstanceState);
9.          setContentView(R.layout.activity_main);
10.         receiver = new ScreenLightReceiver();
11.         //意图过滤器对象
12.         IntentFilter filter = new IntentFilter();
13.         //给意图过滤器添加action 这个action就是要监听的广播对应的action
14.         filter.addAction("android.intent.action.SCREEN_OFF");
15.         filter.addAction("android.intent.action.SCREEN_ON");
16.         //动态注册一个广播接收者
17.         registerReceiver(receiver, filter);
18.     }
19.
20.
21.     @Override
22.     protected void onDestroy() {
23.         super.onDestroy();
24.         //动态注册的广播接收者 在当前activity销毁的时候需要注销掉 unregisterReceiv
    er 注销动态注册的广播接收者
25.         unregisterReceiver(receiver);
26.     }
27.
28. }
```

10 样式和主题

样式 把控件中用到的相同属性 抽取成主题 res->values->style.xml 声明style节点 在style节点下可以声明 item 每个item对应一个具体的控件使用的属性

```
1.  <style name="MyTextViewStyle" >
2.          <item name="android:layout_width">wrap_content</item>
```

```
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#ff0000</item>
        <item name="android:textSize">26sp</item>
        <item name="android:background">#22000000</item>
    </style>
```

样式声明之后 可以在xml布局文件中直接使用

```
<TextView
       style="@style/MyTextViewStyle"
       android:text="@string/hello_world" />
```

作用 多处控件使用相同的样式 如果需要修改 只需在style.xml中修改

主题 作用跟样式类似 只不过作用 的范围不同 主题是作用在整个应用的需要在androidManifest.xml中Application节点下声明对应的
android:theme 这里声明的就是当前应用使用的主题

```
<application
       android:allowBackup="true"
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/MyAppTheme" >
```

```
<style name="MyAppTheme" >
        <item name="android:background">#66ff0000</item>
    </style>
```

主题的在styles.xml中声明的方式跟样式一样 只不过作用的范围更大

11 国际化 internationalization i18n

支持不同的语言环境
如果支持中文环境 需要在res目录下创建一个values-zh目录 在这个目录中放一个string.xml
这个xml文件中所有的内容都是用中文写的

写代码的时候 涉及到字符串的内容 能使用R.string
或者布局文件中 可以使用@string/都要用 这种方式去写

12 常见对话框 getApplicationContext();

AlertDialog

普通对话框

```
public void normal(View v) {
        //通过Builder
        AlertDialog.Builder builder = new Builder(this);
        //给对话框设置一个标题
        builder.setTitle(R.string.dialog_title);
        builder.setMessage("对话框要显示的具体内容");
        //设置确定的按钮 需要显示的内容 以及点击按钮之后的业务逻辑
        //第一个参数 显示在按钮上的文字
        //第二个参数 按钮点击之后要走的方法 接收OnClickListener 是一个接口  按钮被点击之后 会调用接口中的 onclick方法
        builder.setPositiveButton("确定", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "确定", Toast.LENGTH_LONG).show();
            }
        });
        //设置取消的按钮 需要显示的内容 以及点击按钮之后的业务逻辑
                //第一个参数 显示在按钮上的文字
                //第二个参数 按钮点击之后要走的方法 接收OnClickListener 是一个接口  按钮被点击之后 会调用接口中的 onclick方法
        builder.setNegativeButton("取消", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "取消", Toast.LENGTH_SHORT).show();
            }
        });
        //调用show方法 秀一下
        builder.show();
    }
```

单选对话框

```
public void singleChoice(View v) {
        AlertDialog.Builder builder = new Builder(this);
        builder.setTitle("请选择你喜欢的明星");
        //等待选择的选项
        final String[] items = {"宝强","tf-boys","刘德华","成龙","冰冰"};
        builder.setSingleChoiceItems(items, 4, new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), items[which], Toast.LENGTH_SHORT).show();
                //让当前的对话框消失
```

```
                dialog.dismiss();
            }
        });
        builder.show();
    }
```

多选对话框

```
public void multiChoice(View v) {
        AlertDialog.Builder builder = new Builder(this);
        builder.setTitle("请选择你喜欢的水果");
        //等待选择的条目
        final String[] items = {"西瓜","芒果","香蕉","榴莲","苹果","荔枝"};
        //用来确定 那些条目是默认被选中的 可以通过这个数组来记录 各个条目选择的状态  boolean数组 元素个数要跟 items 数据个数对应
        checkedItems = new boolean[]{true,false,false,true,false,true};
        //设置多选的界面
        builder.setMultiChoiceItems(items, checkedItems, new OnMultiChoiceClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                // which 那个条目被选中 对应的索引
                // isChecked 被点击之后条目的选择状态
//              Toast.makeText(getApplicationContext(), items[which]+(isChecked?"被选中":"没选中"), Toast.LENGTH_SHORT).show();
//              dialog.dismiss();
                checkedItems[which]= isChecked;
                for(int i = 0;i<items.length;i++){
                    if(checkedItems[i])
                    System.out.println("选中"+items[i]);
                }
            }
        });
        builder.setPositiveButton("确定", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        builder.show();
    }
```

进度条对话框

```
public void progress(View v) {
        final ProgressDialog dialog = new ProgressDialog(this);
        //设置当前的进度条对话框样式为水平
```

```
        dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        //给对话框设置标题
        dialog.setTitle("正在玩命下载中....");
        //给对话框设置最大进度
        dialog.setMax(100);
        //显示对话框
        dialog.show();
        //开线程模拟下载进度
        new Thread(){
            public void run() {
                for(int i = 0;i<=100;i++){
                    //进度条对话框可以在子线程更新进度
                    dialog.setProgress(i);
                    SystemClock.sleep(100);
                }
                //让当前的对话框消失
                dialog.dismiss();
            };
        }.start();

    }
```

创建对话框的时候也需要使用一个上下文 这个上下文必须是activity
因为对话框要显示到当前的activity中 是activity的一部分 创建对象的时候 需要通知系统当前对话框显示在哪个activity里
所以必须传activity对象作为上下文 如果用getApplicationContext就会崩溃

吐司是系统级的显示控件 传getApplicationContext 作为上下文没有影响

内容回顾
广播接收者 ☆☆☆☆☆
开发的流程 静态注册
①创建class 继承broadcastReceiver
②重写onreceive
③清单文件注册receiver
④某些广播需要添加权限 在清单文件声明对应的权限
动态注册
①创建class 继承broadcastReceiver
②重写onreceive
③ 在代码中调用registerReceiver(BroadCastReceiver,IntentFilter);
3.1创建BroadCastReceiver对象
3.2 创建意图过滤器对象
3.3给意图过滤器 设置action 这个action就是要收到的广播的action
④ 当activity销毁的时候再onDestroy方法中 调用 unregisterReceiver(BroadCastReceiver) 注销广播

接收者

发送自定义广播 ☆☆☆☆☆ 短信
sendBroadCast()
sendOrderedBroadCast()

对话框 ☆☆☆☆
国际化/主题/样式 ☆☆☆