第二章

先从看得到的入手—探究活动

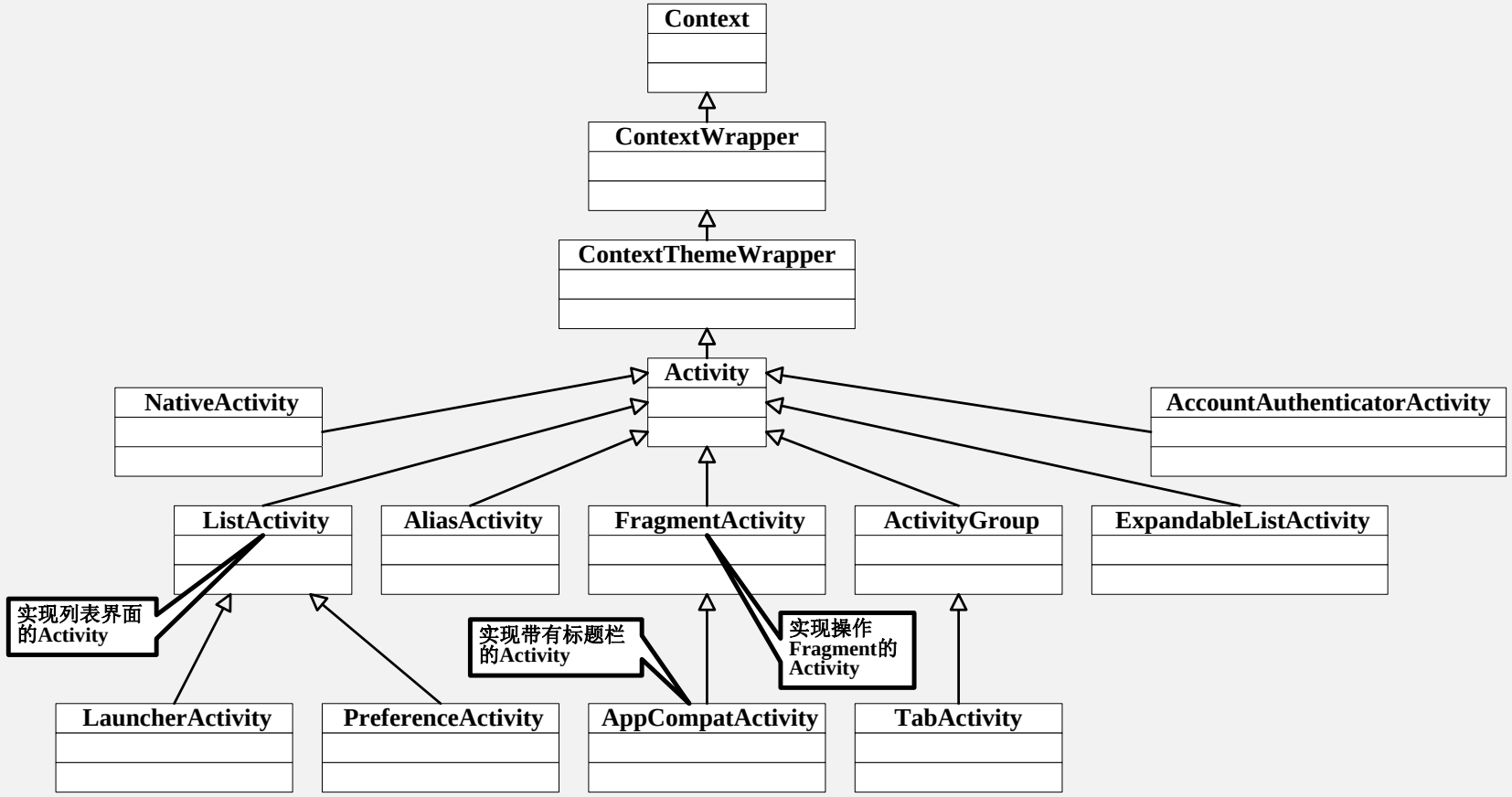
主讲：王海

本章目标

- 掌握Activity的创建及生命周期方法
- 能够访问Android中的各种资源
- 理解AndroidManifest.xml清单文件
- 掌握Android应用程序生命周期
- 掌握Application类及生命周期事件

Activity

- 每个Activity被定义为一个独立的类，并继承android.app.Activity类或其子类。



Activity简介

Activity类中常用的方法：

方法	功能描述
setContentView(int layoutResID)	设置Activity界面布局
onCreate(Bundle savedInstanceState)	Activity生命周期的方法，用于第一次创建Activity
onStart()	Activity生命周期的方法，用于启动Activity
onPause()	Activity生命周期的方法，用于暂停Activity
onStop()	Activity生命周期的方法，用于停止Activity
onDestory()	Activity生命周期的方法，用于销毁Activity
onResume()	Activity生命周期的方法，将Activity由暂停状态恢复使用
onRestart()	Activity生命周期的方法，将Activity由停止状态恢复使用
onKeyDown(int keyCode,KeyEvent event)	键盘按键按下时的动作事件处理方法
onKeyUp(int keyCode,KeyEvent event)	键盘按键抬起时的动作事件处理方法
onTouchEvent(MotionEvent event)	监听屏幕的触摸事件处理方法

Activity简介

Activity类中常用的方法：

方法	功能描述
openContextMenu(View view)	开启上下文菜单
setResult(int resultCode)	返回数据给上一个Activity
startActivityForResult(Intent intent, int requestCode)	携带数据并跳转Activity
finish()	结束当前Activity

创建Activity

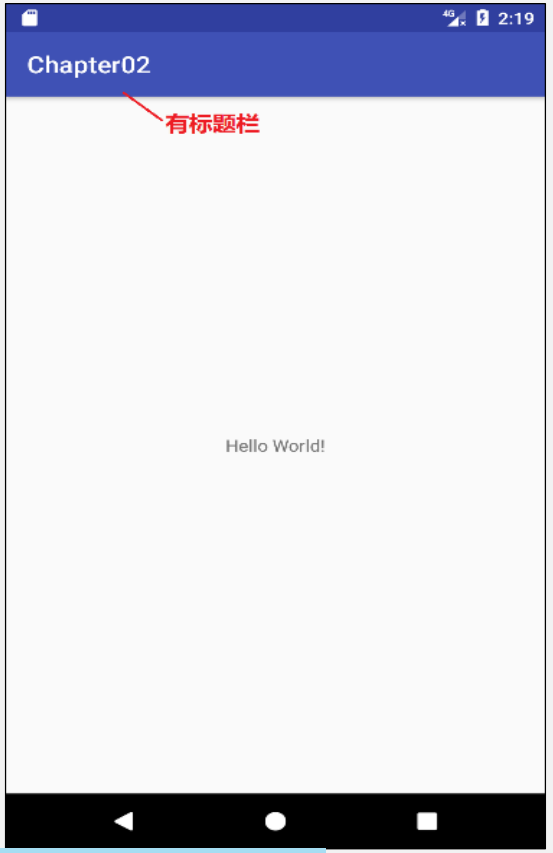
- 通过继承Activity基类的方式实现自定义的BaseActivity类

```
import android.app.Activity;
import android.os.Bundle;
public class BaseActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
   }
}
```

创建Activity

- 通过继承AppCompatActivity类的方式实现Activity

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends
AppCompatActivity {
    @Override
    public void onCreate(Bundle
savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
    }
}
```

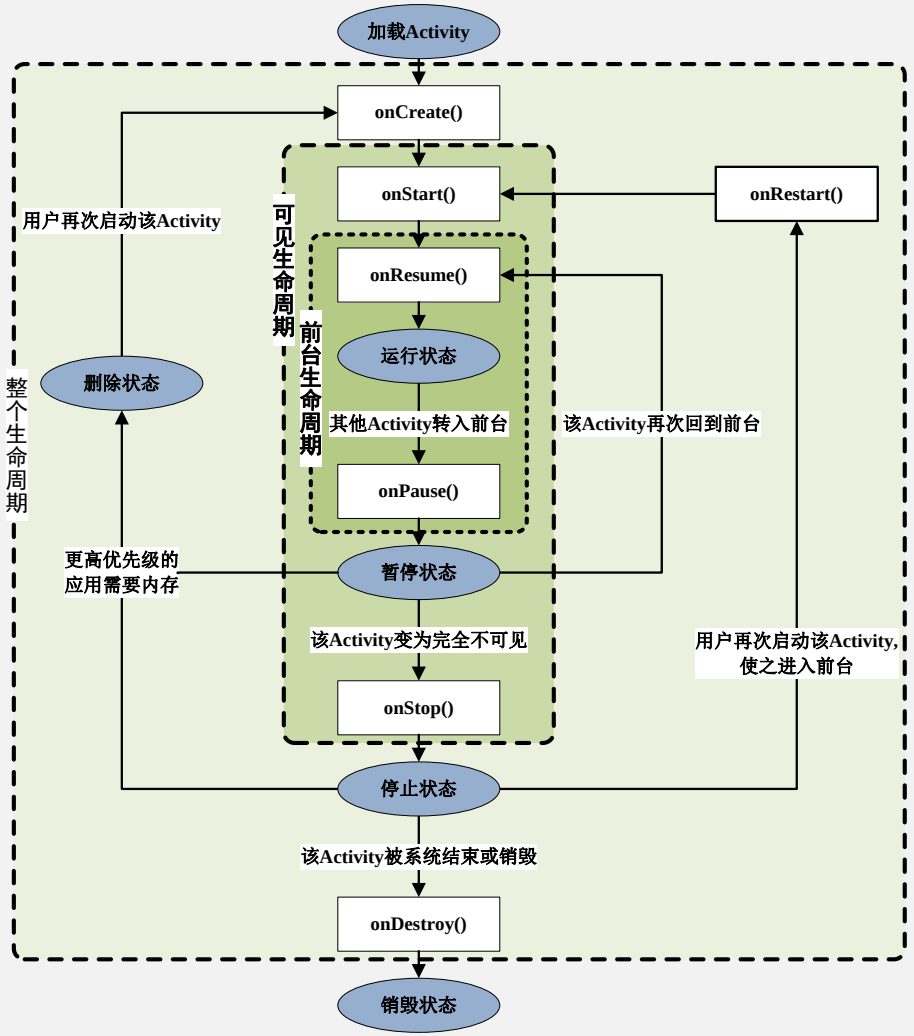


注意

在实际开发过程中，Activity与AppCompatActivity在方法应用上并无很大区别，可根据实际需要选择合适的Activity的基类或者子类进行开发。

Activity的生命周期

- Activity有四种本质区别的状态：
  - 运行状态
  - 暂停状态
  - 停止状态
  - 销毁状态
- Activity有三个关键的循环：
  - 整个生命周期
  - 可见生命周期
  - 前台生命周期



Activity类的定义

```
public class Activity extends ContextThemeWrapper {
    protected void onCreate(Bundle icicle){...}
    protected void onStart(){...}
    protected void onRestart(){...}
    protected void onResume(){...}
    protected void onFreeze(Bundle outIcicle) {...}
    protected void onPause(){...}
    protected void onStop(){...}
    protected void onDestroy(){...}
}
```

Log日志类

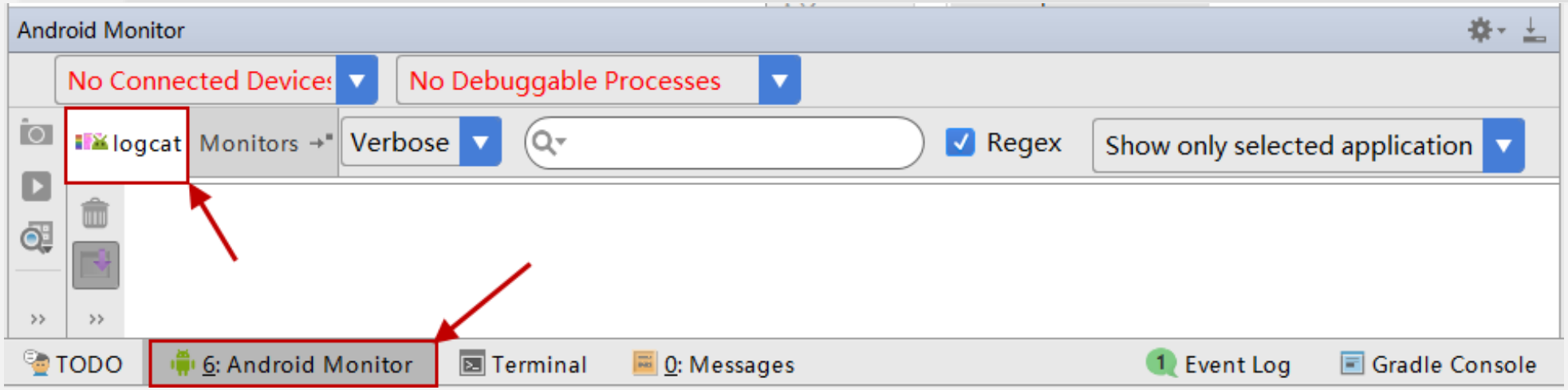
- Log日志类能够记录程序运行过程中的相关信息

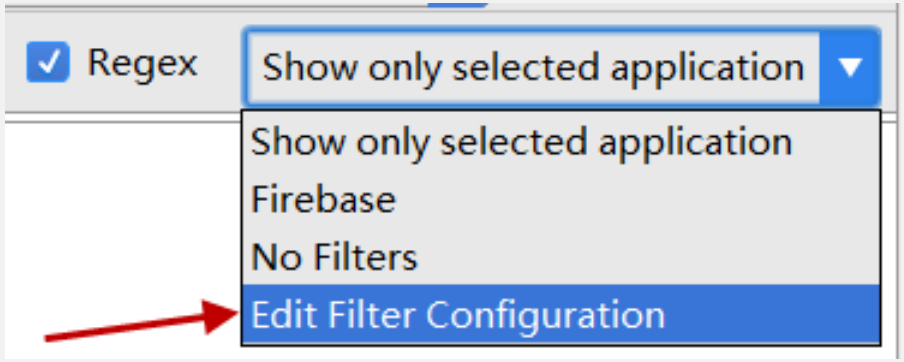
方 法	功能描述
Log.e()	记录错误信息
Log.w()	记录警告信息
Log.i()	记录一般提示性信息
Log.d()	记录调试信息
Log.v()	记录详细的信息

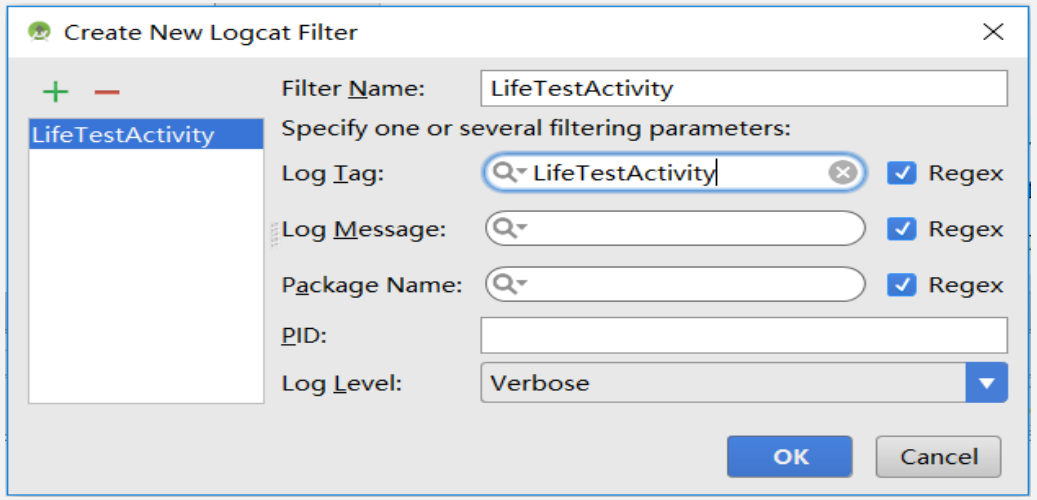
LogCat调试

- LogCat是用来捕获系统日志信息的工具，并能将捕获的信息显示在IDE集成开发环境中。
- LogCat能够捕获信息包括：Dalvik虚拟机产生的信息、进程信息、Android运行时信息、ActivityManager信息、PackagerManager信息、Windows Manger信息和应用程序信息等。

打开LogCat窗口并编辑LogCat过滤器







AndroidManifest.xml清单文件

清单文件中通常包含以下六项信息：

- 声明应用程序的包名
- 描述应用程序组件
- 确定宿主应用组件进程
- 声明应用程序拥有的权限
- 定义应用程序所支持API的最低等级
- 列举应用程序必须链接的库

使用规则

在使用这些元素及元素的属性时，需要遵守几项规则：

- 元素：在所有的元素中只有<manifest>和<application>是必需的且只能出现一次
- 属性：元素的属性大部分是可选的但有少数属性是必须设置的
- 定义类名：所有的元素名都对应其在SDK中的类名
- 多数值项：如果某个元素有超过一个数值时，必须通过重复的方式来说明该元素的某个属性具有多个数值项，且不能将多个数值项一次性说明在一个属性中
- 资源项说明：需要引用某个资源时，采用“@[package:]type:name”格式进行引用
- 字符串值：类似于其他语言

AndroidManifest.xml

<manifest>
节点

<application>
节点

<activity>
节点

<intent-filter>
节点

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.zhaokl.chapter02">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

自定义权限使用<permission>元素声明

```
<permission
    android:label="自定义..."
    android:description="@...
    android:name="com.example.project.TEST"
    android:protectionLevel="normal"
    android:icon="@drawable/ic_launcher">
</permission>
```

权限标题

权限描述

权限名称

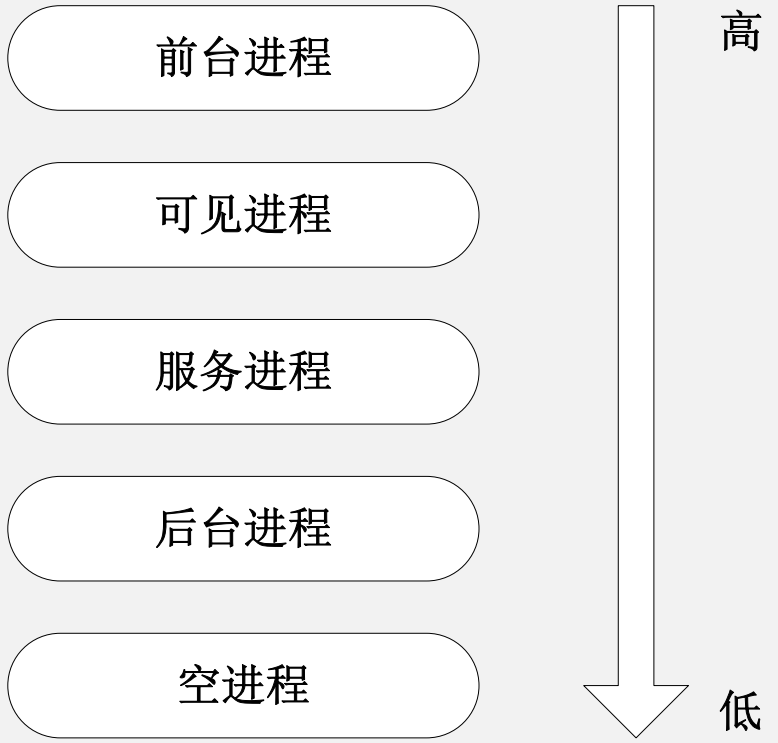
权限级别

- **Android的四种不同权限级别的区分如下：**
  - normal——低风险权限
  - dangerous——高风险权限
  - signature——签名权限
  - signatureOrSystem——签名或系统权限

Android应用程序生命周期

Android根据应用程序的组件及组件当前运行状态将所有的进程按重要性程度从高到低划分了五个优先级：

- 前台进程
- 可见进程
- 服务进程
- 后台进程
- 空进程



Application类

通过扩展Application类，可以完成3项工作：

- 对Android运行时广播的应用程序级事件（如低内存）做出响应
- 在应用程序组件之间传递对象
- 管理和维护多个应用程序组件所使用的资源

Application生命周期事件

Application类为应用程序的创建和终止、低可用内存和配置的改变提供了事件处理程序：

- onCreate()
- onLowMemory()
- onTrimMemory()
- onConfigurationChanged()

在重写这些方法时必须调用父类的事件处理程序。

实现Application

- 实现自定义的Application的步骤：
  - ① 创建一个类继承Application类
  - ② 在Activity中使用Application类
  - ③ 运行并查看结果引用在其他XML中已经定义的资源。

使用Application类可以实现多窗口或其他组件（如Service等）之间的数据共享和传递。

至于Application类的其他功能请参考Android官方文档。

本章总结

- Activity是Android系统最重要组件，是Android程序开发的入口点，深刻领会Activity编程的步骤对于Android开发非常重要
- Activity有运行、暂停、停止和销毁四种状态
- 资源管理是Android编程的一大亮点，体现了MVC编程的优势，对于提高程序的可读性以及可靠性提供了有效的手段
- AndroidManifest.xml清单文件是整个Android应用程序的全局描述配置文件，也是每一个Android应用程序必须有的且放在根目录下的文件
- Android应用程序从高到低划分了五个优先级：前台进程、可见进程、服务进程、后台进程和空进程
- Application类代表当前运行的应用程序，应用程序启动时，系统会自动创建对应Application类的实例，并一直伴随应用程序的生命周期，而且始终维持一个实例