## 四大组件学习记录：

**Activity（活动）：**

Activity是用户操作的可视化界面，为用户提供了一个操作指令的窗口。当完成activity之后，我们需要调用setContentView（）；方法来完成界面的实现。以此来作为用户交互的入口，在android的app中几乎所有的能看到的都依托于activity，所以activity是android开发中最频繁的一个组件。

通常Activity就是单独的一个屏幕（窗口）；

Activity之间通过Intent进行通信。

Android 应用中每一个activity’都需要在androidManifest.xml配置文件中进行声明，否则系统将不识别不执行该activity。

**Activity的生命周期：**

Android中会维持一个Activity的stack，每一个新的activity被创建时都会在栈顶，此时处于运行状态，如果再来一个activity被创建的话，那么之前的activity就被压入并且进入后台。

一个activity实际上有四种状态：

1.运行中（Running/Active）：此时处于栈顶，**可见状态，并且可以与用户交互，此为运行状态**。

2.暂停（paused）：当Activity失去焦点，**不能跟用户交互了，但依然可见，就处于暂停状态。**当一个新的非全屏的Activity或者一个透明的Activity放置在栈顶，Activity就处于暂停状态；这个时候Activity的各种数据还被保持着；**只有在系统内存在极低的状态下，系统才会自动的去销毁Activity。**

3.停止（Stoped）：**当Activity1被另一个activity2完全覆盖了，或者是点击home键退入了后台，这时候Activity1就处于停止状态。**此时和暂停状态有一点类似，那就activity1的各种数据都会被保持；只有当系统其他地方需要用到内容时，系统才会自动去销毁activity；

4.销毁（Detroyed）：**当我们点击返回键或者系统再内存不够用的情况下就会把Activity从栈里移除销毁，被系统回收，这时候，Activity处于销毁状态。**

**Service（服务）：**

Service也是四大组件之一，它通常用在后台处理耗时的逻辑，和activity一样，它存在自己的生命周期，也需要在androidmanifest,xml配置相关信息。

**它适用于去执行那些不需要和用户交互并且还要求长期运行的任务。**任务的运行不依赖于用户界面，即使程序被切换到后台，或者用户打开了另外一个应用程序，服务仍然能够保持正常运行。

Service分为两种：

第一种Started（启动）：当应用程序组件（比如activity）调用startService（）方法启动服务时，服务处于started状态；

第二种bound（绑定）：当应用程序组件调用bindService（）方法绑定到服务时，服务处于bound状态。

**两者的区别：**

**Started service 其生命周期与其他组件无关，可以在后台无期限运行。所以它需要在完成任务后调用Stopself（）停止，或者由其他组件调用StopService（）方法停止。**

**Bindservice（），调用者与服务绑定在了一起，调用者一旦退出，服务也就终止**，大有“不求同时生，必须同时死”的特点。

开发人员需要在配置文件中声明全部的service，使用<service></service>标签。

Service通常位于后台运行，它一般不需要与用户进行交互，所以没有图形用户界面。**Service组件需要继承service基类。**它通常用于为其他组件提供后台服务或者监控其他组件运行状态。

**Broadcast Receive（广播接收者）：**

**广播是一种广泛运用在应用程序之间传输信息的机制。而广播接收器是对发送出来的广播进行过滤接受并且响应的一类组件。**broadcastReceice可以在androidmanfest.xml中注册，也可以在运行时的代码中使用context.registerReceice()进行注册。

Broadcastreceice分为两种：程序动态注册（运行时代码中使用Context,registerReceice()进行注册）和AndroidManifest文件中进行静态注册。

**两者的区别：动态注册广播接收器**是用来注册的activity关掉后，广播也失效了；而**静态注册**无须担忧广播接收器是否被关闭，只要设备是开启状态，广播接收器也是打开着的。就是说哪怕app本身未启动，该app订阅的广播在触发时也会对它起作用。

**Content Provider：（内容提供者）：**

Android 平台提供了**content Provider使一个应用程序的指定数据集提供给其他应用程序**。其他应用可以提供ContentResolver类从该内容提供者中获取或存入数据。

只有需要在多个应用程序间共享数据时才需要内容提供者。它的好处是同意数据访问方式（例如通讯录被多个程序访问），

ContentProvider实现实现数据共享。它**用于保存和获取数据**，并使其对所有应用程序可见。**这是不同app间共享数据的唯一方式，因为android没有提供所有应用共同访问的公共存储区。**

一般不会之间使用contentprovider类对象，大多数是提供ContentResolver对象实现对ContentProvider的操作。

ContentProvider 使用URl来唯一标识其数据集，（例如：URI以content：//作为前缀，表示该数据由ContentProvider来管理）。