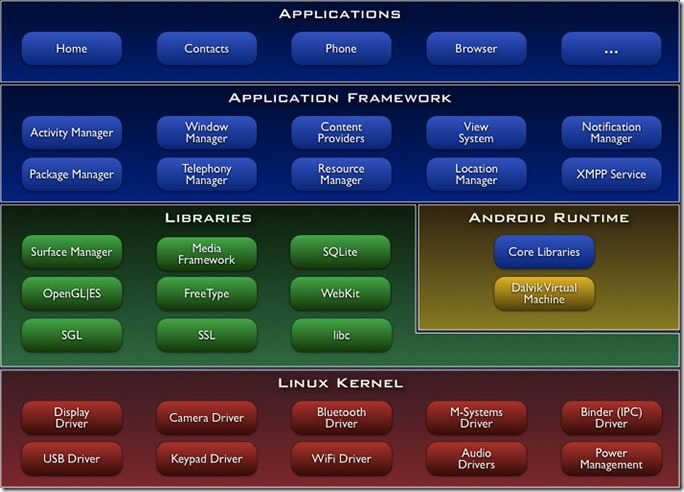
**获取系统服务**

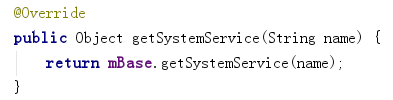
参考：<http://xusx1024.com/2017/02/22/android-system-service-all/>



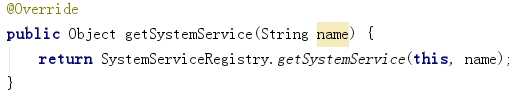
上图第二层就是系统服务，其中包括WindowManager、ActivityManager、PackageMager等。一个app的运行无法避免和系统提供的Binder做交互，**Android系统以服务（Service）的方式暴露出很多Binder代理对象**，**也就是说可以通过系统服务访问这些Binder对象**。一个app以跨进程的方式调用系统的各种服务(即Binder对象)，通常以Context.getSystemService()的方式获取系统服务，这些系统服务有的是普通对象，大部分是对系统Binder对象的封装，系统的IBinder是通过ServiceManager来暴露给app进程的。

注意：如果要拿到其它app进程的Binder一般会使用ServiceConnection连接其他app进程的service来获取Binder。

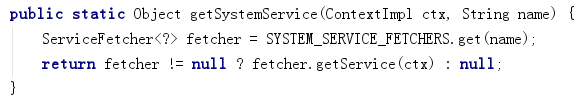
大体了解系统服务后，接下来看app通过Context.getSystemService()获取一个服务的流程，getSystemService方法是Context抽象方法，这个Context既可以是Application的context也可以是一个Activity的context，不管是哪一个最终都是执行的Context的子类ContextWrapper的getSystemService：



这里的mBase指向的是Context的实现类ContextImpl的一个实例，所以getSystemService最终调用的是ContextImpl中的getSystemService方法：



看系统服务原来是由SystemServiceRegistry返回的，SystemServiceRegistry翻译过来是系统服务注册，所以它管理着系统服务的注册，继续看SystemServiceRegistry的getSystemService方法：



SystemServiceRegistry的getSystemService返回的是一个Object对象，这就是为什么我们获取到一个服务后还要强制转换成我们需要的类型。这里返回的对象是通过fetcher的getService方法返回的，而fetcher是由SYSTEM\_SERVICE\_FETCHERS返回的。

* **SystemServiceRegistry**



上图展示了部分代码，首先看L12私有化构造方法，说明SystemServiceRegistry不能被实例化，而且内部也没有SystemServiceRegistry实例对象，就像一个工具类，只能调用它里面的静态方法getSystemService就是一个静态方法。根据java类的初始化顺序来看当SystemServiceRegistry被加载的时候，首先会创建三个私有变量：

SYSTEM\_SERVICE\_NAMES：保存服务名称。

SYSTEM\_SERVICE\_FETCHERS：以键值对的方式存放注册的系统服务。

sServiceCacheSize：记录缓存服务数组的数量。

继续L15开始，是静态代码块，它里面主要是通过执行registerService方法来注册app所需要的服务，看L49在registerService方法里就是把注册好的服务添加进SYSTEM\_SERVICE\_FETCHERS集合中的，然后通过getSystemService方法获取注册的服务。

上面只是展示了一部分服务的注册，开始我们说过一个app注册的服务有两种：

参考：https://juejin.im/entry/58ecc45761ff4b005814d272

1. 直接new一个服务对象，例如L40。
2. 还有一种是通过ServiceManager.getService获取一个系统的Binder代理对象，这样的服务就能访问系统的Binder。

通过上面分析，当一个app内的SystemServiceRegistry类第一次被虚拟机加载的时候，它会在静态代码块里注册所有的系统服务，并保存在SYSTEM\_SERVICE\_FETCHERS集合常量（由private static final修饰）中。具体SystemServiceRegistry什么时候被加载，目前我知道的是在ContextImpl类被实例化的时候就会SystemServiceRegistry就会被加载。由于SystemServiceRegistry是个final的所以他不能被重写，它的构造方法是私有化的并且内部没有它的实例存在，所以它内部保存服务的SYSTEM\_SERVICE\_FETCHERS集合在app进程内有且只有一份是个单例模式的存在了，所以要通过getSystemService来获取相应的服务。

SYSTEM\_SERVICE\_FETCHERS集合存储的是ServiceFetcher对象，下面看看通过注册服务它存储的到底是什么对象：



ServiceFetcher是个接口，CachedServiceFetcher、StaticServiceFetcher是它的两个实现类，不过也是个抽象类，他们内部都有一个createService的抽象方法，看名字就知道是用来创建服务的。这两个实现类里面getService方法的实现是不同。在SystemServiceRegistry类的静态代码块中调用的registerService方法时往SYSTEM\_SERVICE\_FETCHERS中添加的就是CachedServiceFetcher、StaticServiceFetcher的实现类，实现类里面主要实现了createService方法用来创建并返回不同的服务。所以CachedServiceFetcher、StaticServiceFetcher管理了服务的创建和获取过程。每一个服务都对应一个CachedServiceFetcher或StaticServiceFetcher实例被存放在SYSTEM\_SERVICE\_FETCHERS集合中。

CachedServiceFetcher和StaticServiceFetcher是有区别的，它们的区别就在于:

**CachedServiceFetcher创建的服务是缓存在调用getSystemService的ContextImpl的实例中的，它是伴随着ContextImpl的实例存在而存在的。而StaticServiceFetcher创建的服务是缓存在StaticServiceFetcher中即伴随SYSTEM\_SERVICE\_FETCHERS集合存在而存在的。**

CachedServiceFetcher和StaticServiceFetcher类的主要在getService方法中：

**CachedServiceFetcher：**

一个app中会存在好多ContextImpl实例（例如，好多Activity），每个实例中都会有一个mServiceCache对象数组，这个数组就是用来缓存由CachedServiceFetcher创建的服务对象：

1506590906(1)

在getService方法中L85首先会从调用getSystemService方法的ContextImpl的实例的mServiceCache数**组中获**取这个服务，如果不存在则L87行创建这个服务，L88创建完服务后会将服务缓存到调用getSystemService方法的ContextImpl的实例的mServiceCache数组中。也就是说哪个ContextImpl实例通过getSystemService方法获取了CachedServiceFetcher创建的服务，那么这个ContextImpl实例就会在mServiceCache数组中缓存一份这个服务。当这个实例下一次在获取这个服务的时候直接从mServiceCache数组获取。每个ContextImpl实例都有属于自己的CachedServiceFetcher创建的服务，当这个ContextImpl实例被系统回收，那么这个服务也就被回收了。

**StaticServiceFetcher**：

在getService方法中L103会判断mCachedInstance是否为空，如果为空在L104创建服务返回给mCachedInstance。这里可以看出由StaticServiceFetcher创建的服务会缓存在StaticServiceFetcher实例中，由于StaticServiceFetcher是伴随SYSTEM\_SERVICE\_FETCHERS集合存在而存在的，也就是说StaticServiceFetcher创建的服务在每个app进程中有且只有一份。不管哪个ContextImpl实例获取这个服务他们使用的都是同一个服务对象。

另外，从getService方法中也看出，StaticServiceFetcher用不到传进来的Cotext对象，而CachedServiceFetcher却用得到。

到这里context如何获取系统服务就搞清楚了，这里总结下：当一个app启动后，因为会创建ContextImpl实例，所以虚拟机会加载SystemServiceRegistry类，与此同时在SystemServiceRegistry类中首先会创建一个HashMap常量集合SYSTEM\_SERVICE\_FETCHERS存放系统服务的辅助类ServiceFetcher，然后静态代码块中会创建所有系统服务的辅助类ServiceFetcher最终实现类并且添加进SYSTEM\_SERVICE\_FETCHERS集合中。当ContextImpl通过getSystemService获取系统服务时，就会直接返回SYSTEM\_SERVICE\_FETCHERS集合中的辅助类并且调用辅助类的getService方法返回服务对象。