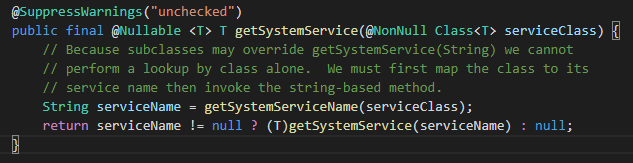
本文主要针对Dymaic Shortcut的创建过程和更新过程进行分析。

在应用中创建Shortcut首先要获取ShortcutManager，即调用如下代码获取ShortcutManager：

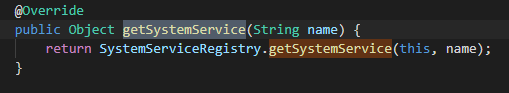
ShortcutManager shortcutManager = getSystemService(ShortcutManager.class);

进入Context. getSystemService(Class serviceClass)中查看源码：

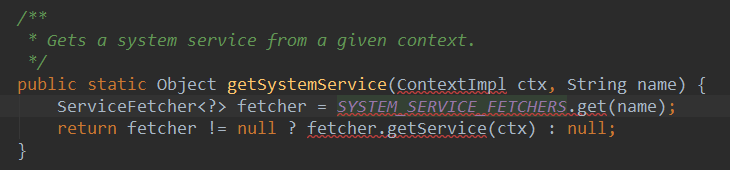
**注意**：快应用是无法通过上述方法获取到ShortcutManager实例的。



实际上，最终调用的还是getSystemService(String serviceName)，该方法的具体实现在ContextImpl中，进入查看源码如下：



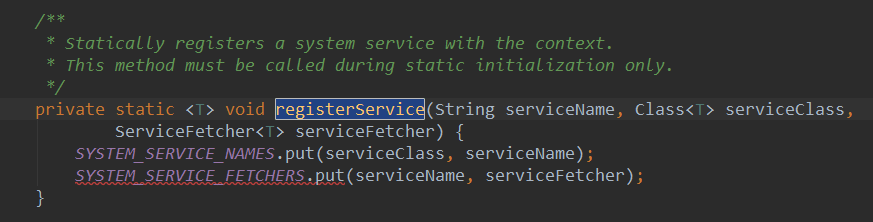
继续查看SystemServiceRegistry.*getSystemService*()中的源码：



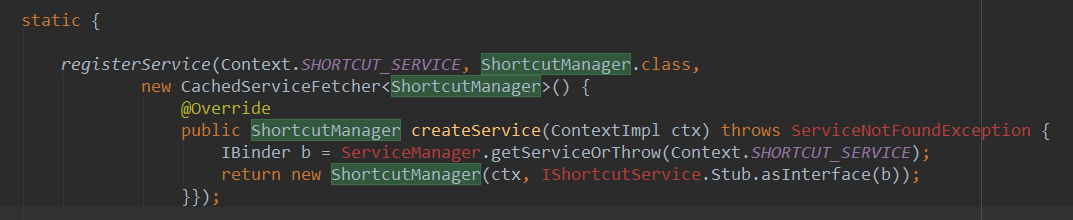
其中SYSTEM\_SERVICE\_MAP是一个HashMap，通过我们服务的名字name字符串，从这个HashMap里取出一个ServiceFetcher，再return这个ServiceFetcher的getService()。

ServiceFetcher是什么？它的getService()又是什么？既然他是从SYSTEM\_SERVICE\_MAP这个HashMap里get出来的，那就找一找这个HashMap都put了什么。

通过搜索SystemServiceRegistry可以找到如下代码：



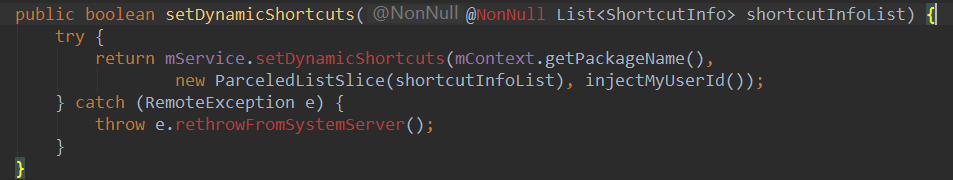
这里往SYSTEM\_SERVICE\_MAP里put了一对String与ServiceFetcher组成的key/value对，registerService()又是从哪里调用的？继续搜索可以发现很多类似下面的代码：



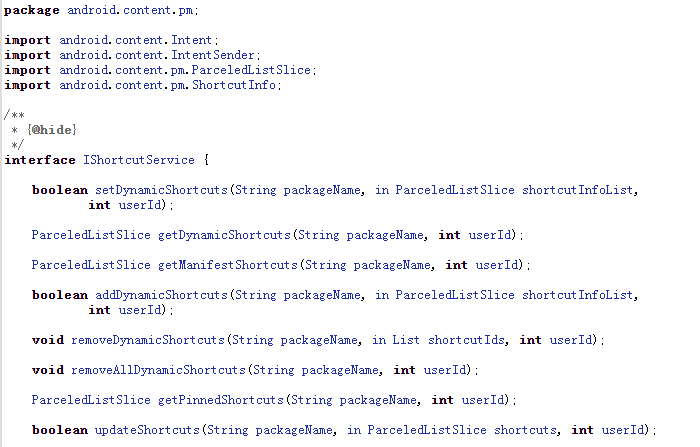
SystemServiceRegistry的static代码块里通过registerService注册了很多的系统服务，其中就包括我们正在调查的SHORTCUT\_SERVICE，通过结合上面的分析代码可以可以知道getSystemService(ShortcutManager.class)得到的是一个ShortcutManager的实例。

至此ShortcutManager实例就获取到了，接下来分析Shortcut是如何设置和更新的，即setDynamicShortcuts()和updateShortcuts()方法的内部实现。

查看setDynamicShortcuts()的内部实现，源码如下：



其中mService为IShortcutService类型，该参数是在上一步初始化ShortcutManager的时候传过来的，即IShortcutService.Stub.asInterface()，搜索Android系统源码发现，IShortcutService其实是一个aidl文件，其路径/[frameworks](http://androidxref.com/9.0.0_r3/xref/frameworks/)/[base](http://androidxref.com/9.0.0_r3/xref/frameworks/base/)/[core](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/)/[java](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/java/)/[android](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/java/android/)/[content](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/java/android/content/)/[pm](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/java/android/content/pm/)/[IShortcutService.aidl](http://androidxref.com/9.0.0_r3/xref/frameworks/base/core/java/android/content/pm/IShortcutService.aidl)。

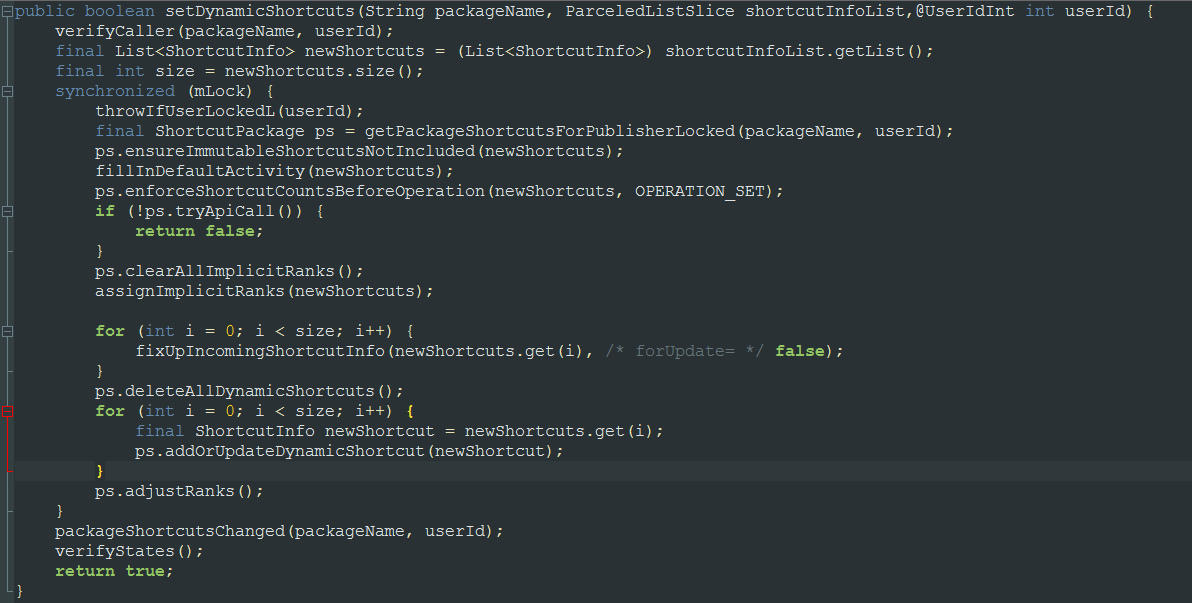


该文件内声明了setDynamicShortcuts()和updateShortcuts()方法，继续搜索源码发现[/frameworks/base/services/core/java/com/android/server/pm/](http://androidxref.com/9.0.0_r3/xref/frameworks/base/services/core/java/com/android/server/pm/)下的ShortcutService继承了IShortcutService.stub类，（关于Android AIDL的分析请移步《跨进程通信》，此处就简单带过，不在重点分析了）。

C:\Users\Stan\AppData\Local\Temp\1566025432(1).png

查看ShortcutSerivce的源码发现，setDynamicShortcuts()和updateShortcuts()方法具体实现在该类中，先分析Shortcut的设置流程，即setDynamicShortcuts()方法的具体实现。

调用方App中所有已存在的Dynamic Shortcut都将被替换。如果已存在具有相同ID的Pinned Shortcut，则会将其进行更新。

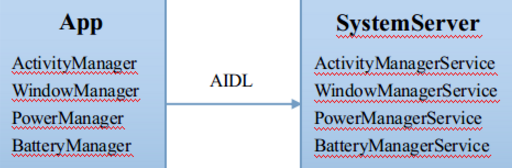


updateShortcuts()较复杂一些，不过总体上来说，其主要的效果就是通过ID来更新所有现有Dynamic Shortcut。

扩展：

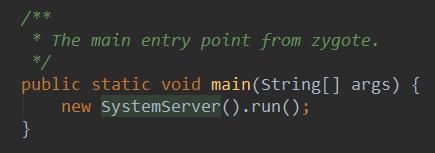
前面只分析了App应用如何调用系统ShortcutService中的方法，但是ShortcutService是如何注册成为系统服务的呢？

如下图所示，ActivityManager、WindowManager和前述分析的ShortcutManager等manager均是运行在应用进程。我们获取manager对象，manager持有能够与系统进程通信的BinderProxy，调用其逻辑方法，逻辑方法中便会与系统进程对应的Service进行通信，使用Service提供的服务。

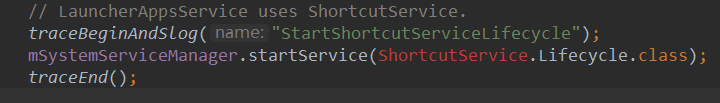


根据上述概述，显然SystemServer是关键，故首先分析一下SystemServer是如何初始化这些系统服务的。

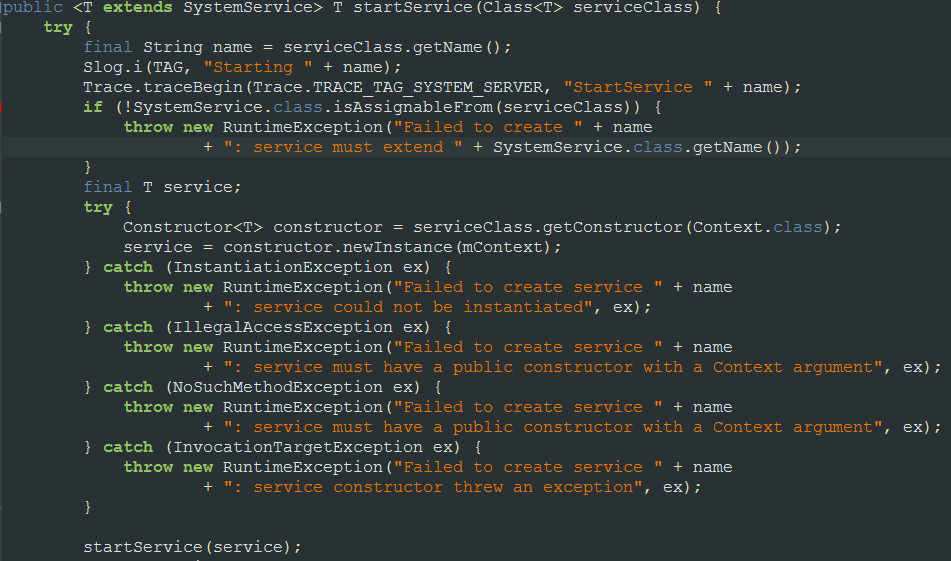
首先，SystemServer由zygote进程启动。启动后，会调用SystemServer.main()方法。



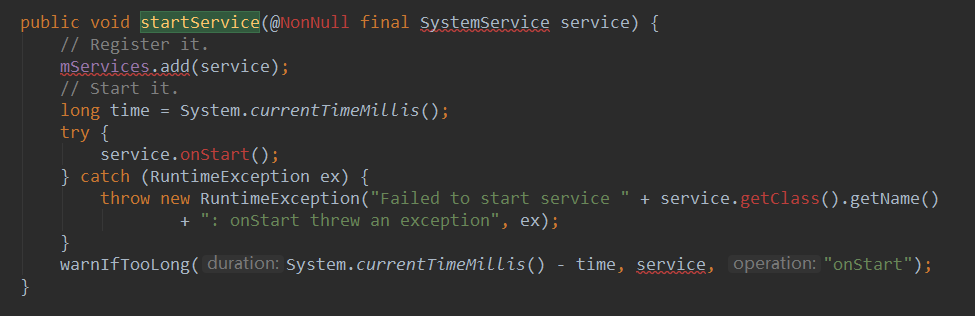
继续查看run()方法，run()方法内进行了一些初始化的工作，例如调用createSystemContext创建系统Context，还有前述分析的ShortcutSerivce服务的注册等。



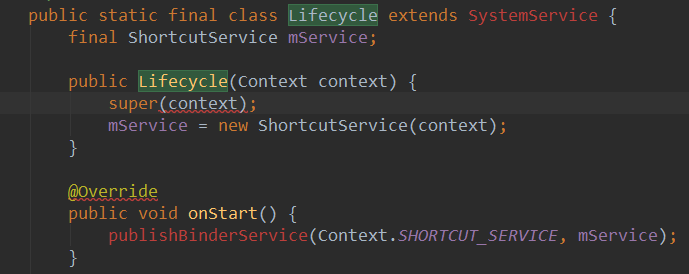
继续查看SystemServiceManager.StartService()中的实现，其中ShortcutService.Lifecycle继承了SystemService。可以看到此处利用反射机制获取到了对应系统service的构造器并进行实例化。



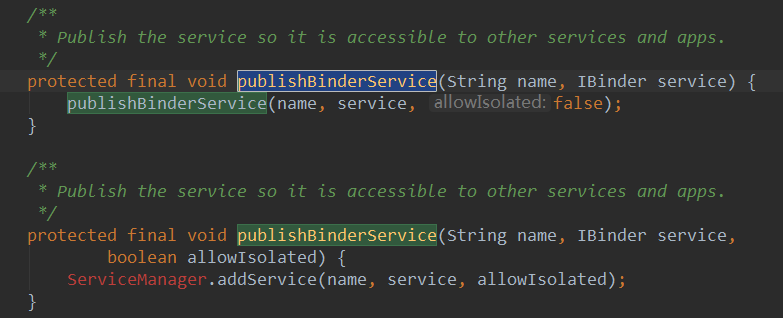
获取到对应系统Service的实例后，调用了startService()方法，继续查看其中的实现。其调用了service.onStart()方法，



查看SystemService中的onStart()方法，发现其是一个抽象方法，其真正的实现在ShortcutService.Lifecycle中，查看ShortcutService.Lifecycle的onStart()方法。



在onStart()方法中，调用了[publishBinderService](http://androidxref.com/9.0.0_r3/s?refs=publishBinderService&project=frameworks)()方法，该方法实际上就是将该Service进行发布，以便于其他Service和App去调用它。



至此，ShortcutService就成功的注册为系统Service并发布出去供上层应用层App进行调用。