**Setting下面增加搜索（SearchIndexProvider)**

2016年05月24日 14:14:34 [懒人就是我](https://me.csdn.net/lfx_xianxian) 阅读数：2552

一、

当自己在Settings增加选项a后，如果我们想要被增加的选项a在搜索中也可以被搜索到，那么除了关注自己[界面](https://www.baidu.com/s?wd=%E7%95%8C%E9%9D%A2&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)的那个类，还需要关注下面2个类：

**src/com/android/settings/search/Ranking.java**

**src/com/android/settings/search/SearchIndexableResources.java**

二、步骤

1、步骤一：自己页面内增加SearchIndexProvider内容

拿Settings下面的About phone来举例，代码6.0，[DeviceInfoSettings.java](http://androidxref.com/6.0.1_r10/xref/packages/apps/Settings/src/com/android/settings/DeviceInfoSettings.java" \t "_blank)，在自己相应的xxxSettings.java类里面添加以下内容，命名自己名哦，

About phone模板：

1. public static final SearchIndexProvider SEARCH\_INDEX\_DATA\_PROVIDER =
2. new BaseSearchIndexProvider() {
4. @Override
5. public List<SearchIndexableResource> getXmlResourcesToIndex(
6. Context context, boolean enabled) {
7. final SearchIndexableResource sir = new SearchIndexableResource(context);
8. sir.xmlResId = R.xml.device\_info\_settings;
9. return Arrays.asList(sir);
10. }
11. };

自己页面内增加：

1. public static final SearchIndexProvider SEARCH\_INDEX\_XXX\_PROVIDER =
2. new BaseSearchIndexProvider() {
4. @Override
5. public List<SearchIndexableResource> getXmlResourcesToIndex(
6. Context context, boolean enabled) {
7. final SearchIndexableResource sir = new SearchIndexableResource(context);
8. sir.xmlResId = R.xml.xxx\_xxx\_settings; *//自己配置文件名称*
9. return Arrays.asList(sir);
10. }
11. };

2、步骤二：类SearchIndexableResources.java里面添加内容

About phone模板：

1. sResMap.put(DeviceInfoSettings.class.getName(),
2. new SearchIndexableResource(
3. Ranking.getRankForClassName(DeviceInfoSettings.class.getName()),
4. NO\_DATA\_RES\_ID,
5. DeviceInfoSettings.class.getName(),
6. R.drawable.ic\_settings\_about));

添加

import com.android.settings.XXXSettings;

1. sResMap.put(XXXSettings.class.getName(),
2. new SearchIndexableResource(
3. Ranking.getRankForClassName(XXXSettings.class.getName()),
4. NO\_DATA\_RES\_ID,
5. XXXSettings.class.getName(), *//XXXSettings为自己页面类名字*
6. R.drawable.xx));*//图片资源*

3、步骤三：在Ranking.java文件里面增加自己内容

Adbout phone模板：

sRankMap.put(DeviceInfoSettings.class.getName(), RANK\_DEVICE\_INFO);

在该文件里面增加自己内容哟

1. import com.android.settings.XXXSettings;
3. public static final int RANK\_XXX\_XX = 22;
4. sRankMap.put(XXXSettings.class.getName(), RANK\_XXX\_XX);

其实这个可以grep一下，参照别的页面做的，按部就班

**最后需要注意也是关键的地方**

a.继承SettingsPreferenceFragment，实现Indexable

例如：public class XXXSettings extends **SettingsPreferenceFragment** implements **Indexable**{}

b.添加以上东西前提是Settings.java和SettingsActivity.java里面都已经添加好相关内容了哟。其实没添也不会挂，哈哈。最好标准化添加上。

# Android源码解析--Quick Search in Settings

leagmain 2016-05-17 [原文](https://www.bbsmax.com/link/MU81RTQ0YXJKNw==)

欢迎大家加入群里交流：429664282

基于：android-6.0.1\_r17 f4b8ad6

Android Settings中存在一个SearchIndexablesProvider，它提供了可供快速检索的设置项。通过它，我们可以决定哪些系统设置可以被快速检索，那些可以不被检索。

SearchIndexablesProvider简介

SearchIndexablesProvider是Android标准API，在SDK中可以找到，路径是"android.provider.SearchIndexablesProvider"。它是一个虚基类，提供了一些抽象方法和常用方法。

抽象方法如下：

Cursor queryXmlResources(String[])

Cursor queryRawData(String[])

Cursor queryNonIndexableKeys(String[])

queryXmlResources方法返回一个Cursor，这个Cursor中包含了所有可以被索引的XmlResource，方法的参数是个String数组，用来表示传入查询的列。

queryRawData方法同queryRawData很类似，唯一区别是返回所有可以被索引的RawData。

queryNonIndexableKeys返回所有可以不被所有的NonIndexableKey。

检索数据来源：SearchIndexableResources

在Settings中，所有可供检索的数据资源，均被定义在SearchIndexableResources类中。

例如，Wifi设置页面的所有可供检索的数据，都通过WifiSettings.java提供。

static {

sResMap.put(WifiSettings.class.getName(),

new SearchIndexableResource(

Ranking.getRankForClassName(WifiSettings.class.getName()),

NO\_DATA\_RES\_ID,

WifiSettings.class.getName(),

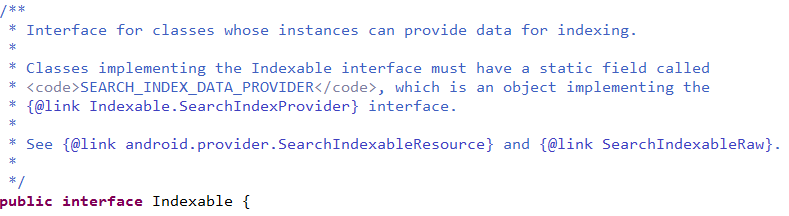
R.drawable.ic\_settings\_wireless));

特殊检索数据：Indexable

在Settings中，提供了一类特殊的检索数据，那就是实现了Indexable接口的类。

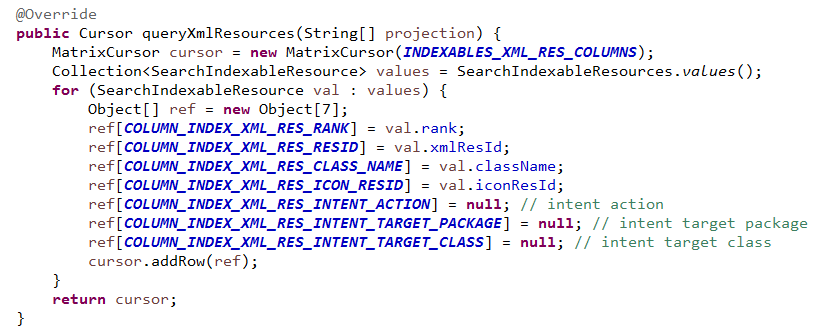
Indexable是定义在Settings内部的一种检索资源，通过代码的方式，返回可供检索的数据，包括SearchIndexableResource和SearchIndexRaw。

所有实现了Indexable的类，必须提供一个public static的SEARCH\_INDEX\_DATA\_PROVIDER变量，用来供Settings利用反射获取到相应的检索数据。

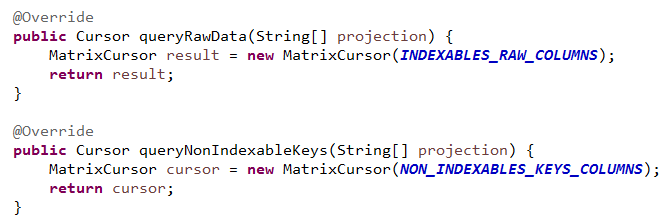


检索数据提供者：SettingsSearchIndexablesProvider

在Settings中，SettingsSearchIndexablesProvider实现了SearchIndexablesProvider，并且提供了可供检索的数据资源。这个数据资源以静态map的方式定义在SearchIndexableResources中，并且在queryXmlResources方法中返回给数据请求者。

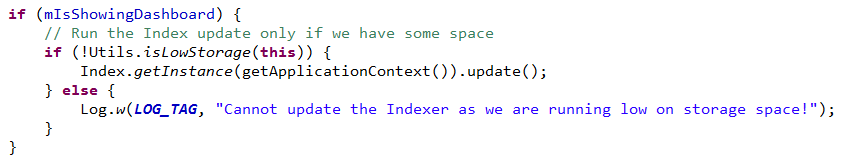


Settings并未提供任何的RawData和NonIndexableKey，所以它的queryRawData返回了一个空的Cursor。

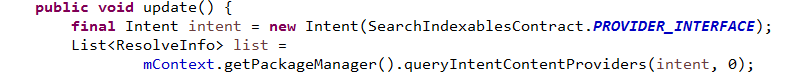


更新检索数据库

在每次打开Settings导航页面的时候，Index.update()方法都会被调用，用来更新检索数据。



Step1. 获取所有SearchIndexablesProvider.



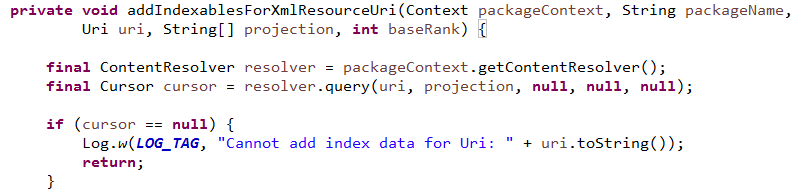
Step2. 添加可供检索的数据到缓存：addIndexablesFromRemoteProvider

对于每一个SearchIndexablesProvider，都需要进行这个操作。

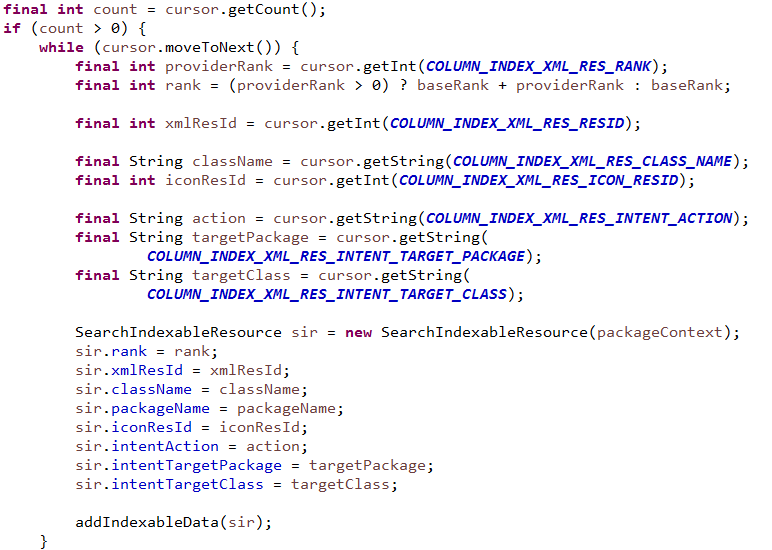
Step2.1. 从XmlResource中添加检索数据：addIndexablesForXmlResourceUri

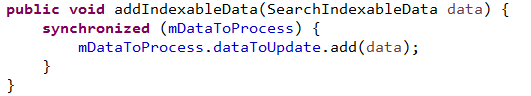
所有定义在SearchIndexableResources中的检索数据，均会在这一步中被添加进来。接下来，让我们看看具体实现：

Step2.1.1 获取检索数据的Cursor



Step2.1.2 创建SearchIndexableResource，并加入到更新数据的缓存列表：





Step2.2. 从RawData中添加检索数据：addIndexablesForRawDataUri

Settings并未提供任何RawData，所以我们可以忽略这个步骤。

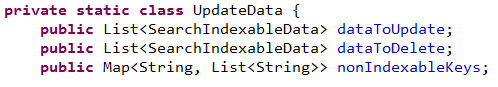
Step3. 添加不可供检索的数据到缓存

Settings并未提供任何NonIndexableKey，所以我们可以忽略这个步骤。

Step4. 更新缓存的检索数据

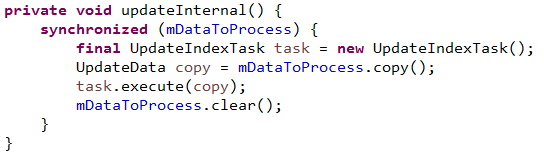
更新缓存的检索数据在updateInternal()方法中实现，下面我们来逐步分析下。

缓存数据结构



到现在为止，Settings提供的检索数据均被添加到"dataToUpdate"列表中。其他两个未包含任何Settings提供的检索数据。

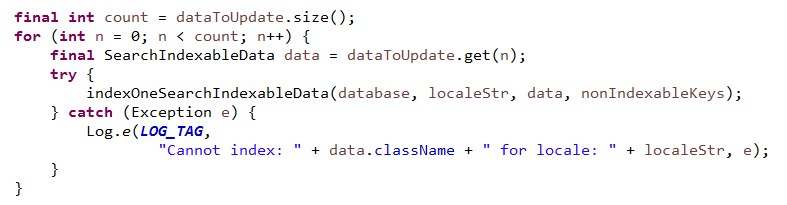
Step4.1 创建一个UpdateIndexTask的后台任务来更新检索数据库



我们下面只关心dataToUpdate的相关操作。

Step4.2 更新dataToUpdate缓存：processDataToUpdate

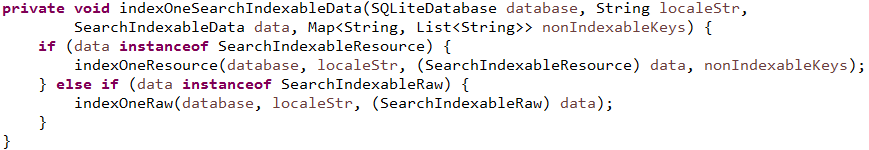
在这个方法中，我们对每一个SearchIndexableData都使用indexOneSearchIndexableData方进行更新。



Step4.3 对检索数据进行索引：indexOneSearchIndexableData

这里的检索数据可以包括RawData和XmlResource，又或者Settings返回的SearchIndexableResource。

这里为什么是SearchIndexableResource？因为SettingsSearchIndexablesProvider把所有可以检索的数据都已SearchIndexableResource插入到Cursor中，并且返回给数据查询者，这里也就是Settings本身。

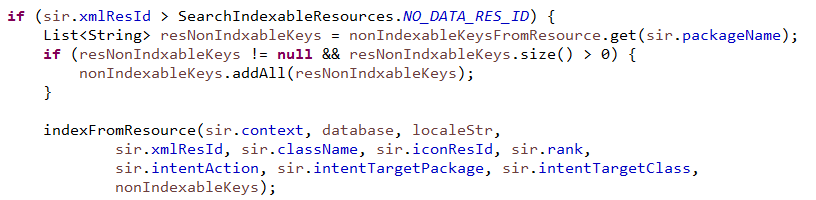


Step4.4 对每一条检索数据资源进行数据检索：indexOneResource

前面我们提到过，Settings里面有一种特殊的检索数据资源：Indexable。

在这里，如果我们发现这条检索数据资源的xmlResId是NO\_DATA\_RES\_ID，也就是说，当前这个检索数据资源是一条特殊的检索数据资源。我们就需要利用反射机制得到它的pusblic static的SEARCH\_INDEX\_DATA\_PROVIDER。

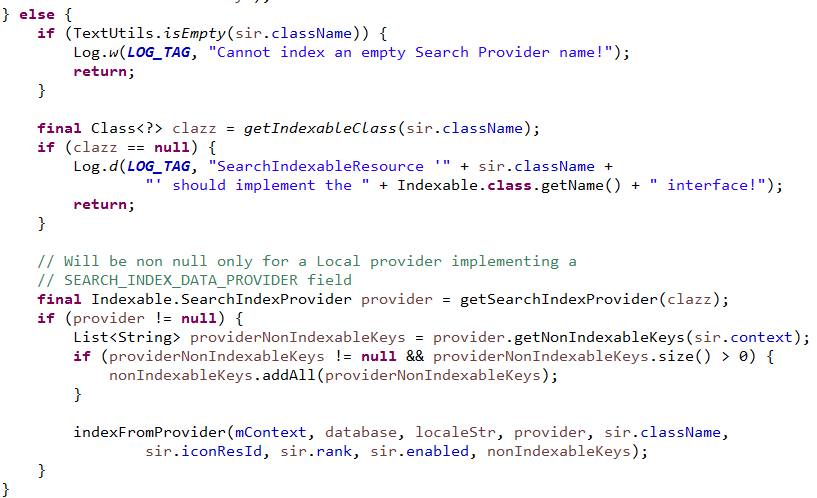
Step4.4.1. 处理普通的检索数据资源：xmlResId != NO\_DATA\_RES\_ID



Step4.4.1.1. 从XML中解析检索数据：indexFromResource

在这里就到了整个实现流程的最末端了，解析整个PreferenceScreen的XMl文件，并且把每一个符合条件的项加入到检索数据库。

Step4.4.2. 处理特殊的检索数据资源：Indexable



Step4.4.2.1. 从特殊的检索数据源获取检索数据：indexFromProvider

当拿到SEARCH\_INDEX\_DATA\_PROVIDER变量后，我们得到它的两个方法返回值，根据返回值的具体内容，我们将其更新到数据库。