# TODO

matchData

# IntentFilter

## 隐式 Intent匹配规则

并不知道也不关心接收者是谁，有利于降低发送者和接收者之间的耦合，它一般用在没有明确指出目标组件名称的前提下，一般是用于在不同应用程序之间，如下：

Intent intent = new Intent();

intent.setAction("com.wooyun.test");

startActivity(intent);

### 规则描述

* a. action匹配规则：要求intent中的action 存在 且必须和过滤规则中的其中一个相同 区分大小写；
* b. category匹配规则：系统会默认加上一个android.intent.category.DEAFAULT，所以intent中可以不存在category，但如果存在就必须匹配其中一个；
* c. data匹配规则：data由两部分组成，mimeType和URI，要求和action相似。如果没有指定URI，URI但默认值为content和file（schema）

### match

**public final int** match(ContentResolver resolver, Intent intent,  
 **boolean** resolve, String logTag) {  
 String type = resolve ? intent.resolveType(resolver) : intent.getType();  
 **return** match(intent.getAction(), type, intent.getScheme(),  
 intent.getData(), intent.getCategories(), logTag);  
}

**public final int** match(String action, String type, String scheme,  
 Uri data, Set<String> categories, String logTag) {  
 **if** (action != **null** && !matchAction(action)) {  
 **return *NO\_MATCH\_ACTION***;  
 }  
 **int** dataMatch = matchData(type, scheme, data);  
 **if** (dataMatch < 0) {  
 **return** dataMatch;  
 }  
  
 String categoryMismatch = matchCategories(categories);  
 **if** (categoryMismatch != **null**) {  
 **return *NO\_MATCH\_CATEGORY***;  
 }  
  
 **return** dataMatch;  
}

#### matchAction

* intent中的action 存在
* 且必须和过滤规则中的其中一个相同
* 区分大小写

**public final boolean** matchAction(String action) {  
 **return** hasAction(action);  
}

**public final boolean** hasAction(String action) {  
 **return** action != **null** && **mActions**.contains(action);  
}

#### matchData??

data由URI来描述和定位，URI由三部分组成，

scheme://host:port/path 模式://主机:端口/路径

挂载SD卡事件，data的URI为file:///storage/3D6B-1CF9

MountReceiver: intent = Intent { act=android.intent.action.MEDIA\_MOUNTED dat=file:///storage/3D6B-1CF9 flg=0x4000010 (has extras) }

其中模式部分是 file ， 主机:端口部分是空的， path部分是 storage/3D6B-1CF9

**部分匹配原则：只要filter中声明的部分匹配成功，就认为整个URI匹配成功。**

1. 如果data的URI和datatype为空，则 filter 的URI和type也必须为空，才能匹配成功

2. 如果data的URI不为空，但是datatype为空，则 filter 必须定义URI并匹配成功，且type为空，才能匹配成功

3. 如果data的URI为空，但是datatype不为空，则 filter 必须URI为空，定义type并匹配成功，才能匹配成功

4. 如果data的URI和data都不为空，则 filter 的URI和type都必须定义并匹配成功，才能匹配成功。

    对于URI部分，有一个特殊处理，就是即使filter没有定义URI，content和file两种URI也作为既存的URI存在。

   （举个例子，对于 content 和 file 两种模式的data，只要filter定义的datatype可以和事件匹配，就认为匹配成功，

     filter不需要显式的增加 content 和 file 两种模式，这两种模式内置支持

matchData(null, “file”, Uri data)

**public final int** matchData(String type, String scheme, Uri data) {  
 **final** ArrayList<String> types = **mDataTypes**;  
 **final** ArrayList<String> schemes = **mDataSchemes**;  
  
 **int** match = ***MATCH\_CATEGORY\_EMPTY***;  
  
 **if** (types == **null** && schemes == **null**) {  
 **return** ((type == **null** && data == **null**)  
 ? (***MATCH\_CATEGORY\_EMPTY***+***MATCH\_ADJUSTMENT\_NORMAL***) : ***NO\_MATCH\_DATA***);  
 }  
  
 **if** (schemes != **null**) {  
 **if** (schemes.contains(scheme != **null** ? scheme : **""**)) {  
 match = ***MATCH\_CATEGORY\_SCHEME***;  
 } **else** {  
 **return *NO\_MATCH\_DATA***;  
 }  
  
 **final** ArrayList<PatternMatcher> schemeSpecificParts = **mDataSchemeSpecificParts**;  
 **if** (schemeSpecificParts != **null** && data != **null**) {  
 match = hasDataSchemeSpecificPart(data.getSchemeSpecificPart())  
 ? ***MATCH\_CATEGORY\_SCHEME\_SPECIFIC\_PART*** : ***NO\_MATCH\_DATA***;  
 }  
 **if** (match != ***MATCH\_CATEGORY\_SCHEME\_SPECIFIC\_PART***) {  
 *// If there isn't any matching ssp, we need to match an authority.* **final** ArrayList<AuthorityEntry> authorities = **mDataAuthorities**;  
 **if** (authorities != **null**) {  
 **int** authMatch = matchDataAuthority(data);  
 **if** (authMatch >= 0) {  
 **final** ArrayList<PatternMatcher> paths = **mDataPaths**;  
 **if** (paths == **null**) {  
 match = authMatch;  
 } **else if** (hasDataPath(data.getPath())) {  
 match = ***MATCH\_CATEGORY\_PATH***;  
 } **else** {  
 **return *NO\_MATCH\_DATA***;  
 }  
 } **else** {  
 **return *NO\_MATCH\_DATA***;  
 }  
 }  
 }  
 *// If neither an ssp nor an authority matched, we're done.* **if** (match == ***NO\_MATCH\_DATA***) {  
 **return *NO\_MATCH\_DATA***;  
 }  
 } **else** {  
 *// Special case: match either an Intent with no data URI,  
 // or with a scheme: URI. This is to give a convenience for  
 // the common case where you want to deal with data in a  
 // content provider, which is done by type, and we don't want  
 // to force everyone to say they handle content: or file: URIs.* **if** (scheme != **null** && !**""**.equals(scheme)  
 && !**"content"**.equals(scheme)  
 && !**"file"**.equals(scheme)) {  
 **return *NO\_MATCH\_DATA***;  
 }  
 }  
  
 **if** (types != **null**) {  
 **if** (findMimeType(type)) {  
 match = ***MATCH\_CATEGORY\_TYPE***;  
 } **else** {  
 **return *NO\_MATCH\_TYPE***;  
 }  
 } **else** {  
 *// If no MIME types are specified, then we will only match against  
 // an Intent that does not have a MIME type.* **if** (type != **null**) {  
 **return *NO\_MATCH\_TYPE***;  
 }  
 }  
  
 **return** match + ***MATCH\_ADJUSTMENT\_NORMAL***;

#### matchCategories

不能为null

**public final** String matchCategories(Set<String> categories) {  
 **if** (categories == **null**) {  
 **return null**;  
 }  
  
 Iterator<String> it = categories.iterator();  
  
 **if** (**mCategories** == **null**) {  
 **return** it.hasNext() ? it.next() : **null**;  
 }  
  
 **while** (it.hasNext()) {  
 **final** String category = it.next();  
 **if** (!**mCategories**.contains(category)) {  
 **return** category;  
 }  
 }  
  
 **return null**;  
}

## 使用案例

（1）如果我们想要匹配 http 以 “.pdf” 结尾的路径，使得别的程序想要打开网络 pdf 时，用户能够可以选择我们的程序进行下载查看。

我们可以将 scheme 设置为 “http”，pathPattern 设置为 “.\*//.pdf”，整个 intent-filter 设置为：

<intent-filter> <action android:name="android.intent.action.VIEW"></action> <category android:name="android.intent.category.DEFAULT"></category> <data android:scheme="http" android:pathPattern=".\*//.pdf"></data> </intent-filter>

如果你只想处理某个站点的 pdf，那么在 data 标签里增加 android:host=”yoursite.com” 则只会匹配<http://yoursite.com/xxx/xxx.pdf>，但这不会匹配 www.yoursite.com，如果你也想匹配这个站点的话，你就需要再添加一个 data 标签，除了 android:host 改为 “www.yoursite.com” 其他都一样。

（2）如果我们做的是一个IM应用，或是其他类似于微博之类的应用，如何让别人通过 Intent 进行调用出现在选择框里呢？我们只用注册 android.intent.action.SEND 与 mimeType 为 “text/plain” 或 “*/*” 就可以了，整个 intent-filter 设置为：

<intent-filter> <action android:name="android.intent.action.SEND" /> <category android:name="android.intent.category.DEFAULT" /> <data mimeType="\*/\*" /> </intent-filter>

这里设置 category 的原因是，创建的 Intent 的实例默认 category 就包含了 Intent.CATEGORY\_DEFAULT ，google 这样做的原因是为了让这个 Intent 始终有一个 category。



（3）如果我们做的是一个音乐播放软件，当文件浏览器打开某音乐文件的时候，使我们的应用能够出现在选择框里？这类似于文件关联了，其实做起来跟上面一样，也很简单，我们只用注册 android.intent.action.VIEW 与 mimeType 为 “audio/\*” 就可以了，整个 intent-filter 设置为：

<intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.DEFAULT" /> <data android:mimeType="audio/\*" /> </intent-filter>

