

# Camera APP 的互斥实现

孙延宾

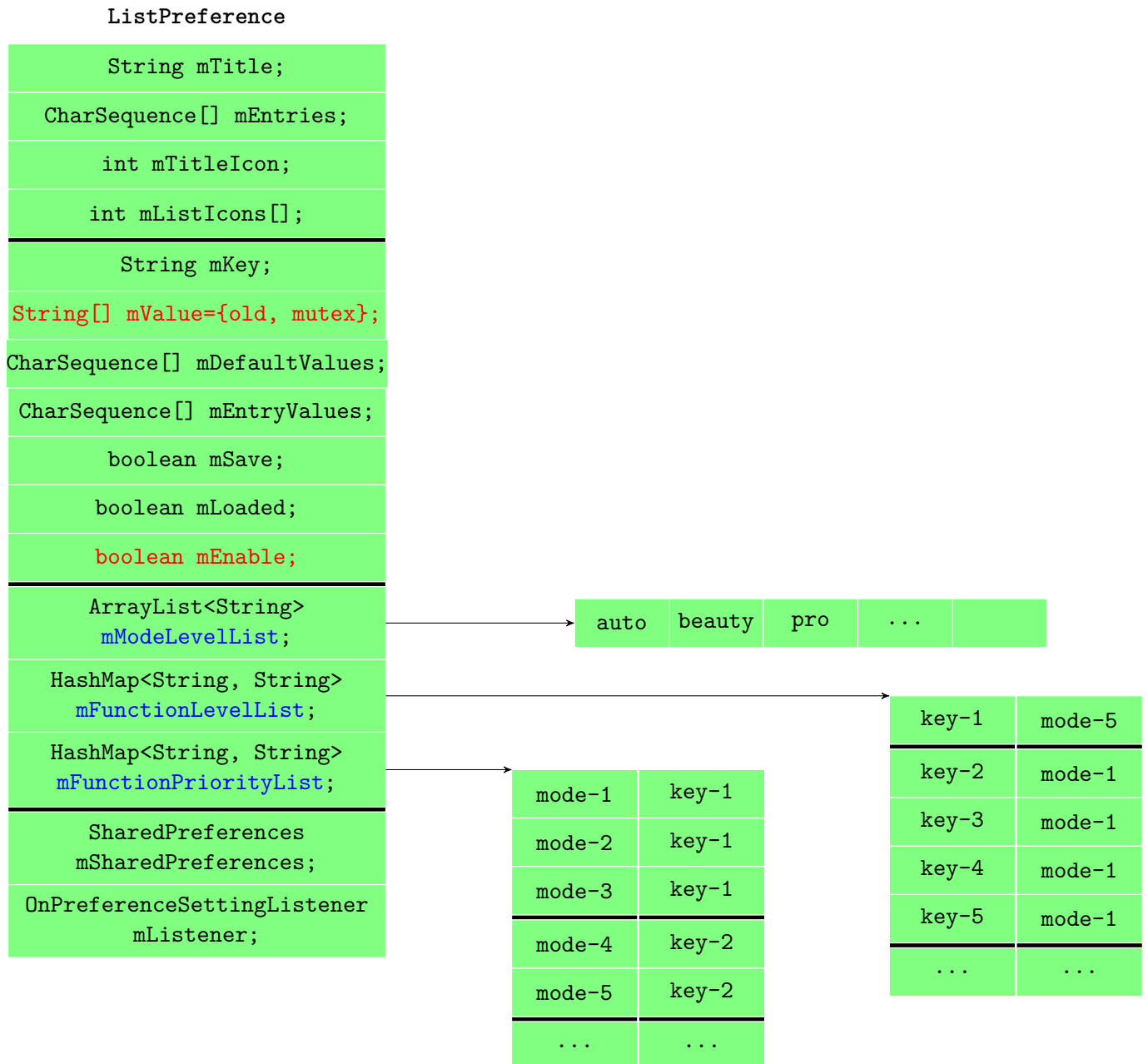
4 月 28 日， 2017 年

## 目 录

I	数据结构	1
II	操作流程	2

## 第 I 部分 数据结构

互斥相关的数据结构只有一个：*ListPreference*，不过并非所有属性都与互斥相关，与互斥相关的只有红色的 *mValue*、*mEnable* 以及蓝色的三个 *list* (*list* 和 *map*)。



- *mValue* 是一个 *length* 为 2 的数组，第一个元素表示其本来的值，第二个元素表示它跟别的功能冲突而被禁用时的临时值（一般为 *off*）
- *mEnable* 表示互斥的结果，互斥的结果只有两个，要么打开、要么被关闭（有特例）

- 蓝色的三个 list 表示与该 preference 冲突的功能点，注意功能点是与模式相关的，*auto* 模式下的闪光灯跟 *professional* 模式下的闪光灯是不一样的。其中的两个 hashmap 其实是表达一种“多对多”的关系：一个模式对应多个功能、一个功能也可以对应多个模式。

## 第 II 部分 操作流程

数据结构明确了，那 *ListPreference* 里的数据是如何修改的呢？答案就是 *MutualHandle*，任何一个 preference 的打开、关闭，以及与其冲突的功能点设置都是在 *MutualHandle* 中操作的。

该类主要做了两件事情：

1. 为每一个可能存在冲突的功能点定义一个处理函数，注意这里的功能点与模式无关，它与上面定义的“功能点”不同。
2. 设置互斥的入口：*doMutualHandle(manager, key)*，即有 preference 变动就要到互斥里面来走一圈。

下面是一个代码实例，

```
1  /*
2  * xxxMutual:
3  *   当 xxx 功能 (如 flash) 因为冲突 (或取消冲突) 而 disable (或 enable) 时,
4  *   调用该函数来处理 xxx 的 preference 以及处理与该功能存在冲突的其他功能。
5  *
6  * enable: 打开还是关闭
7  * mutualState: 当该功能被关闭时要设置的值, 一般为 off, 表示因为冲突而被关闭
8  * levelValue: 模式级别还是功能级别,
9  *   1. 模式级别表示在某个模式下该功能 enable/disable
10 *   2. 功能级别表示该功能与某个功能冲突, 注意功能是与模式相关的
11 * modeValue: 相关的模式
12 * lastMutualKey: 功能点, 如 flash, 与 modeValue 组成具体的功能点
13 * beAssociate: 处理循环冲突。如 A、B 互相冲突, 水火不容, A 变动会引起 B 变动,
14 *   但是 B 变动是就不要再引起 A 变动了, 否则就死循环了。
15 */
16 private void xxxMutual(SettingManager manager, boolean enable, String mutualState,
17     int levelValue, String modeValue, String lastMutualKey, boolean beAssociate) {
18     // 1. 拿到 pref 对象, 因为要修改的就是它
19     ListPreference pref = ((ListPreference)
20         SettingProvider.getListPreference(SettingFuncConstant.KEY_XXX));
21     if (pref == null)
22         return;
23     // 2. 把 key 添加到变动列表里, 告知 UI 哪些 preference 有变动
24     manager.addChangedKey(pref.getKey());
25     // 3. 记录与 pref 冲突的功能点, 并 enable/disable pref
26     pref.setPriority(enable, levelValue, modeValue, lastMutualKey);
27     // 4. 如果 pref 是被 disable 了, 那 disable 之后要怎么变化呢,
28     //   一般是将其值设置为 "off", 或者 0, 如倒计时, 等等
29     if (!pref.isEnabled()) {
30         pref.setValue(mutualState, ListPreference.UPDATE_VALUE_MUTEX);
31     }
32     // 5. 避免循环冲突
33     if (!beAssociate)
34         return;
35     // 6. 最后, 哪些功能点会与 "我" 冲突呢, 你们该变得也赶紧变吧
36     if ("on".equals(pref.getValue())) {
37         supLightMutual(manager, false, PRIORITY_LEVEL_FUNCTION, modeValue,
38             SettingFuncConstant.KEY_LIVE_SHOOT);
39     } else {
40         supLightMutual(manager, true, PRIORITY_LEVEL_FUNCTION, modeValue,
41             SettingFuncConstant.KEY_LIVE_SHOOT);
42     }
43 }
```