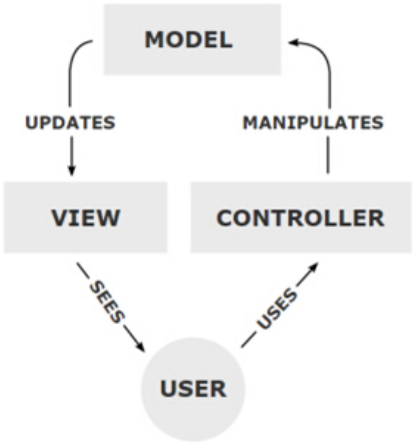
从闪光灯功能看SystemUI中的MVC架构

从Android5.0开始，原生的系统引入了Camera2的API可以更方便地完全控制安卓设备相机，在以前的Camera（android.hardware.Camera）中，对相机的手动控制需要更改系统才能实现，而且API也不友好，而且可以发现在Android5.0以后的原生系统里，在手机的快速设置面板里多了手电筒功能（当然手机必须支持闪关灯功能才能看到）。而最近刚好需要使用到手电筒的功能，所以阅读了这部分相关部分的源代码（主要在com.android.systemui包下），并对SystemUI的设计原理有了一点见解与认识，所以整理出来做个小结。

两条主线

- 快捷设置菜单选项按钮对用户动作的监听响应，是怎么下发动作的
- 设备监听系统参数是怎么将变化后的参数上传到快捷设置菜单界面进行更新

简单介绍一下MVC模式



MVC是一种经典的软件设计思路，MVC将业务逻辑、数据控制与界面显示分离，真正实现了代码的高内聚低耦合，是一种上乘的软件设计框架。

- Model： 用来处理应用程序的数据逻辑部分。通常模型对象负责处理在数据库中存取数据，负责数据的增删改查。要处理的数据接收自控制器。
- View： 用来处理数据显示部分，通常视图是依据模型数据创建的，目的是将数据高效的展示给用户。要显示的数据接收自模型。
- Controller： 用来处理用户交互的部分，通常控制器负责从视图中读取数据，控制用户输入。接收到用户的输入后向模型发送数据。

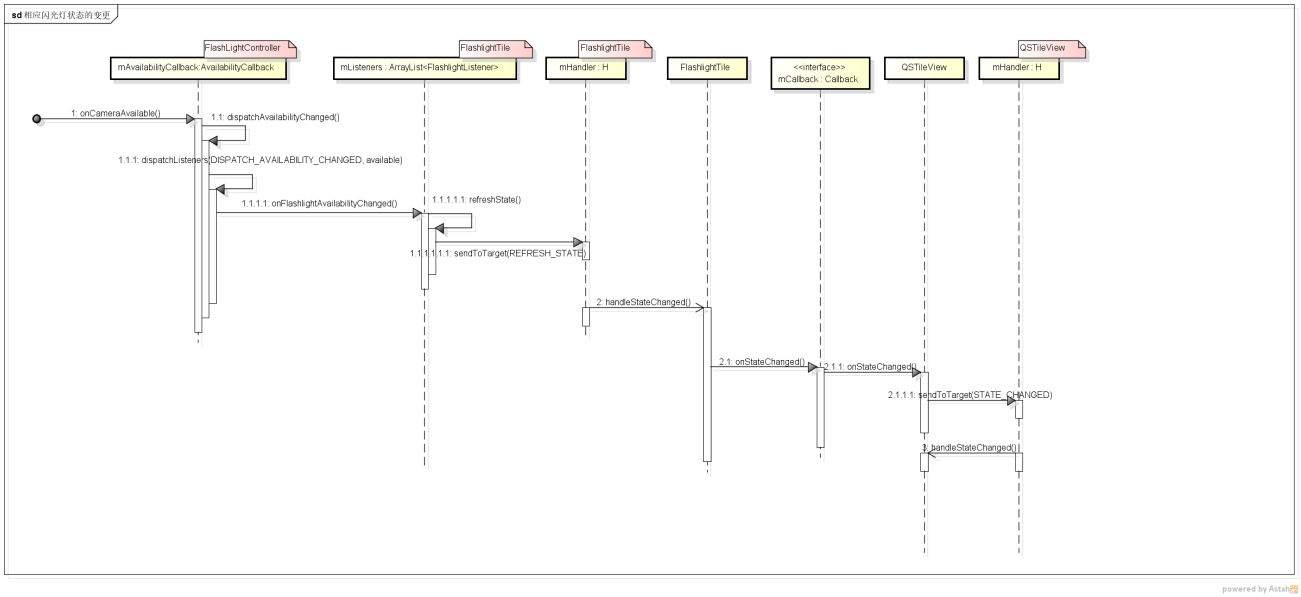
MVC框架最显著的优点是低耦合，现在开源社区的出现很多MVP、MVVM的模式都可以说是在MVC中发展而来的。比如在Android的SystemUI中，则就可以看到MVC的影子。

SystemUI中的MVC

下面选取的是和闪光灯功能相关的代码和时序图做分析，为了方便理解，首先介绍下相关的类

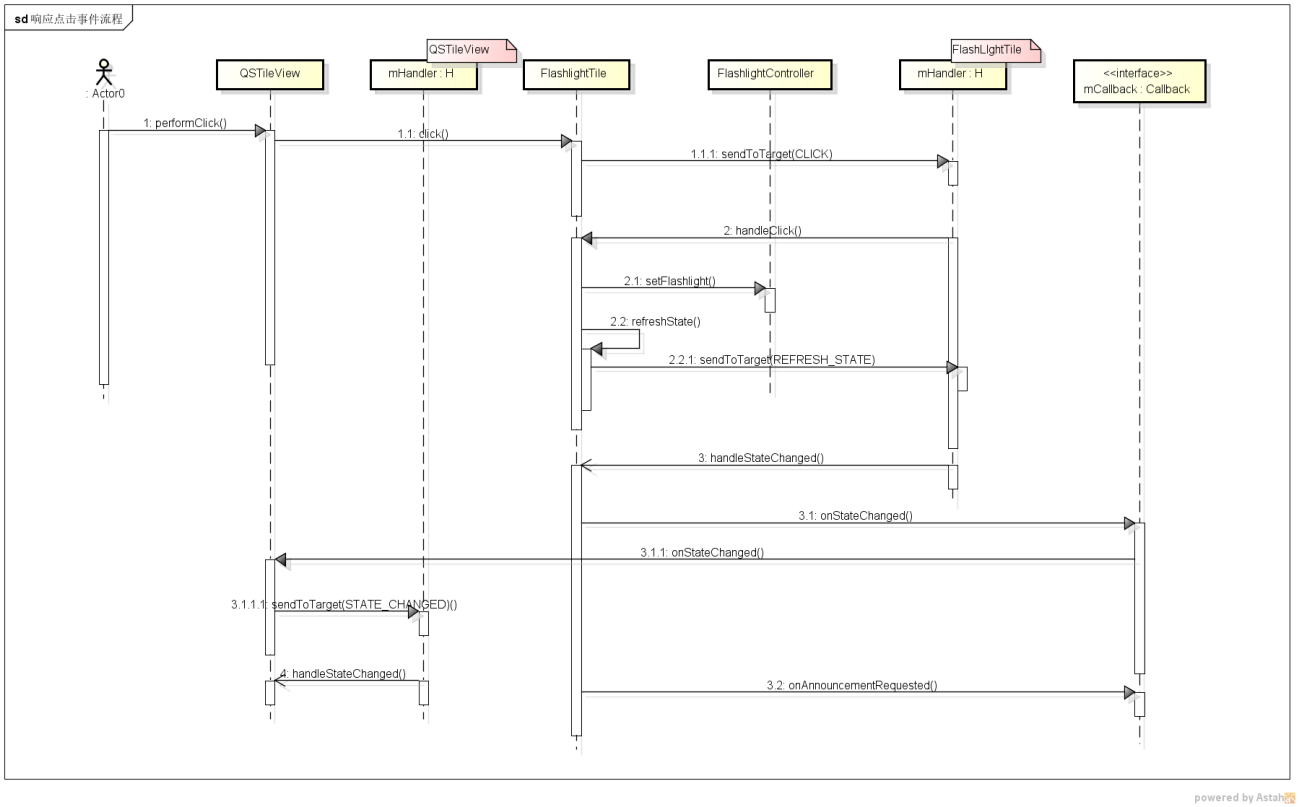
- FlashLightController： 用于控制闪光灯的开启和关闭等操作，并响应系统对摄像机的资源可用的回调，属于**Controller**层
- FlashTile： 记录了闪光灯的状态，并在状态改变的时候通知View去更新，属于**Model**层
- QSTileView： 继承自 `ViewGroup` ，快捷设置面板的某个选项，主要由icon和title组成，属于**View**层

监听系统参数的变化



从时序图中我们可以看到，在摄像设备可以获取的时候， `FlashLightController` 通过 `AvailabilityCallback` 接收系统的回调，从名字可以知道它属于**Controller**，接着通过方法 `onFlashlightAvailabilityChanged` 分发到**Model**层，也就是 `FlashTile` ，其实现了 `FlashLIghtListener` 接口，用于接收来自**Controller**的消息，当 `FlashTile` 处理完状态的变换之后，通过 `Callback` 接口反馈到**View**层，这里是 `QSTileView` （继承自ViewGroup）接收回调后更新UI。这个过程还是挺清晰简单的。

响应用户动作



当点击快捷面板的某个选项的时候，根据时序图可以看到事件被分发到了**Model**层的 `FlashTile` 来处理， `FlashTile` 首先还是分发到 `FlashLightController` 来控制闪光灯的开关（这里不是首先分发到**Controller**的确有点奇怪，大概是因为这里并不需要关注 `FlashLightController` 操作的结果，但也是不影响各层之间的解耦），接着才是自身状态的修改，当 `FlashTile` 处理完状态的变换之后，通过 `Callback` 接口反馈到**View**层的 `QSTileView` 来更新UI。

小结

从Android的SystemUI中可以看到MVC的影子，但却并不是完全地遵循这个模式，毕竟没有一个完美的开发模式框架，关键是理解它们的思想，灵活使用，做好模块的分层解耦就足够了