Android移动开发课程

实验指导书-Notification部分

【实验目的】

初步了解Android 的 Notification 组件

【实验设计】

本次实验包括三个验证实验和一个自选实验。

其中验证实验为个人实验，已经提供源代码、操作步骤、实验指导视频

要求：

按照实验步骤完成，

以个人为单位提交，提交实验报告一份，实验报告需要回答指导书中问题。

自选实验为小组实验，建议2-4人组队，提供了参考选题和参考资料。

要求：

以小组为单位提交，提交实验报告一份、源代码一份、可以执行的APK文件一个。

【实验内容】

通知是指 Android 在应用的界面之外显示的消息，旨在向用户提供提醒、来自他人的通信信息或应用中的其他实时信息。用户可以点按通知来打开应用，也可以直接在通知中执行某项操作。它是看不见的程序组件(Broadcast Receiver，Service和不活跃的Activity)警示用户有需要注意的事件发生的最好途径。

了解通知的可以展示位置：状态栏和抽屉式通知栏、提醒式通知、锁定屏幕、应用图标的标志、Wear OS 设备(参考：附录A通知的概览-在设备上的外观)

验证实验一：创建一条标准视图的通知

本实验已经提供了源代码，源代码参见（exmaple包下的Exp1Notification和Exp1Activity两个文件）

自 Android 1.0 开始，通知系统界面以及与通知相关的 API 就在不断发展。我们不使用原生的Notification和NotificationManager，而是采用NotificationCompat 及其子类，以

及 NotificationManagerCompat，这样一来，无需编写条件来检查API级别，因为NotificationCompat和NotificationManagerCompat为我们代劳。(参考：附录A通知的概览-通知的兼容性)

1. 了解一条通知的构成,并且创建一条基本通知。掌握setSmallIcon方法、setContentTitle方法、setContentText方法，了解这些方法设置的元素的位置。点击"创建一条基本通知"按钮，查看状态栏。

通知剖析

通知的设计由系统模板决定，您的应用只需要定义模板中各个部分的内容即可。通知的部分详情仅在展开后视图中显示。

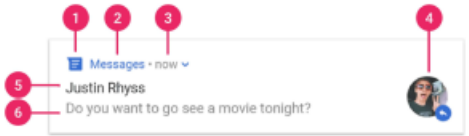

图 7. 包含基本详情的通知

图 7 展示了通知最常见的部分，具体如下所示：

1. 小图标：必须提供，通过 `setSmallIcon()` (/reference/android/support/v4/app/NotificationCompat.Builder#setSmallIcon(int)) 进行设置。
2. 应用名称：由系统提供。
3. 时间戳：由系统提供，但您可以通过 `setWhen()` (/reference/android/support/v4/app/NotificationCompat.Builder#setWhen(long)) 将其替换掉或者通过 `setShowWhen(false)` (/reference/android/support/v4/app/NotificationCompat.Builder#setShowWhen(boolean)) 将其隐藏。
4. 大图标：可选内容（通常仅用于联系人照片，请勿将其用于应用图标），通过 `setLargeIcon()` (/reference/android/support/v4/app/NotificationCompat.Builder#setLargeIcon(android.graphics.Bitmap)) 进行设置。
5. 标题：可选内容，通过 `setContentTitle()` (/reference/android/support/v4/app/NotificationCompat.Builder#setContentTitle(java.lang.CharSequence)) 进行设置。
6. 文本：可选内容，通过 `setContentText()` (/reference/android/support/v4/app/NotificationCompat.Builder#setContentText(java.lang.CharSequence)) 进行设置。

2.创建一条自定义时间戳的通知，了解到部分安卓手机系统可能不支持修改时间戳。掌握setWhen方法。点击"创建一条自定义时间戳的通知"按钮，查看状态栏。

3.创建一条不显示时间戳的通知，掌握setShowWhen方法。点击"创建一条不显示时间戳的通知"按钮，查看状态栏。

4.创建一条拥有大图标的通知，掌握setLargeIcon方法。点击"创建一条拥有大图标的通知"按钮，查看状态栏。

5.创建一条拥有大文本的通知，了解setStyle方法、NotificationCompat.BigTextStyle类点击"。创建一条拥有长文本的通知"按钮，查看状态栏。

6.创建一条拥有大图片的通知，了解setStyle方法，NotificationCompat.BigPictureStyle类。创建一条拥有大图片的通知"按钮，查看状态栏。

7.使用通知创建一个前台服务，了解startForeground用法。点击"创建一个前台服务"，查看状态栏。

本验证实验提交内容：

1. 提交观察到的结果

验证实验二：了解通知管理、掌握创建通知渠道的方法

本实验已经提供了源代码，源代码参见（exmaple包下的Exp2Notification、Exp2Activity和ExpUtils 三个文件）

我们采用NotificationManagerCompat来管理通知，这里采用了桥接设计模式，其实这个类内部调用了getSystemService(Context.NOTIFICATION_SERVICE)获取了一个Notification对象，所有操作都是对这个Notification对象进行的，由于NotificationManagerCompat已经针对不同的API进行检查，因此减少了我们编码的工作量。

从 Android 8.0（API 级别 26）开始，必须为所有通知分配渠道，否则通知将不会显示。使用NotificationManager.createNotificationChannel向系统注册通知渠道。

1. 了解NotificationManager和NotificationManagerCompat，并且使用mContext.getSystemService(Context.NOTIFICATION_SERVICE)来获得NotificationManager对象，或者使用NotificationManagerCompat.from(Context)来获得一个NotificationManagerCompat对象。

2. 使用NotificationManagerCompat中的createNotificationChannel方法注册一个通知渠道。知道创建采用其原始值的现有通知渠道不会执行任何操作。

3. 使用NotificationManagerCompat.notify来显示一条通知，该方法需要一个唯一的ID以及一个Notification对象。点击"在渠道二发起一条ID为10的通知"按钮，查看状态栏。

4. 了解notify方法的第一个参数的作用，并且用这个参数更新通知。点击"在渠道二更新一条ID为10的通知"按钮，查看状态。

5. 利用id来展示一个带进度条的通知，了解setProgress方法。点击"在渠道二发起一条ID为11的通知\n带进度条，每次点击可以增加10%"按钮，查看状态栏。

6. 创建一个拥有IMPORTANCE_HIGH重要度的通知渠道，并且发布在这个渠道上的通知与发布在IMPORTANCE_DEFAULT渠道上通知的异同，了解提醒式通知。点击"在渠道三发起一条ID为12的通知"按钮，查看状态栏。（如果没有提醒式通知，检查系统通知设置中允许使用提醒式通知设置是否启用。）

7. 在手机系统的通知设置中找到该应用，观察应用注册过的渠道。点击"进入系统通知设置"按钮和"进入系统关于频道三的通知设置"按钮，查看状态，了解通过隐式Intent快速打开通知设置。

8. 了解通知的移除,掌握NotificationManagerCompat.cancel方法。点击在渠道二发起一条ID为10的通知"按钮，然后点击"移除ID为10的通知"按钮，查看状态栏。

9. 了解通知的移除,掌握NotificationManagerCompat.cancelAll方法。点击"移除本应用发出的全部通知"按钮，查看状态栏。

本验证实验提交内容：

1. 提交观察到的结果

验证实验三：了解通知上的活动。

本实验已经提供了源代码，源代码参见（exmaple包下的Exp3Notification和Exp3Activity两个文件）

1. 了解PaddingIntent。PaddingIntent可以将返回的对象移交给其他应用程序，以便以后可以执行描述的操作。通过getActivity(Context, int, Intent, int)，getActivities(Context, int, Intent[], int)， getBroadcast(Context, int, Intent, int)，和getService(Context, int, Intent, int)来获取PaddingIntent对象。

PendingIntent本身只是对系统维护的令牌的引用，该令牌描述了用于检索令牌的原始数据。这意味着，即使其拥有的应用程序的进程被杀死，PendingIntent本身也将在已赋予它的其他进程中保持可用。

2. 每个通知都应该对点按操作做出响应，通常是在应用中打开对应于该通知的Activity。 掌握setContentIntent(pendingIntent)方法。点击"创建一条可以启动本应用的通知"按钮，点击通知栏上的通知，观察状态。

3.通知操作个通知最多可以提供三个操作按钮，让用户能够快速响应，例如暂停提醒，甚至回复短信。但这些操作按钮不应该重复用户在点按通知时执行的操作。掌握addAction方法。 点击"创建一条具有操作按钮的通知"按钮，分别执行点按通知、点击"发送一个广播"按钮、点击"启动一个服务"按钮，观察状态态。

4. 显示紧急消息。应用可能需要显示紧急的时效性消息，例如来电或响铃警报。在这些情况下，可以将全屏 Intent 与通知关联。调用通知时，根据设备的锁定状态，用户会看到以下情况之一：

如果用户设备被锁定，会显示全屏 Activity，覆盖锁屏。

如果用户设备处于解锁状态，通知以展开形式显示，其中包含用于处理或关闭通知的选项。

如果应用的目标平台是 Android 10（API 级别 29）或更高版本，必须在应用清单文件中请求 USE_FULL_SCREEN_INTENT 权限，以便系统启动与时效性通知关联的全屏 Activity

点击"创建一条显示紧急消息的通知"按钮，观察状态。

本验证实验提交内容：

1. 提交观察到的结果

自选实验

从以下实验中选择一个实验或者自主设计一个实验，建议1-4人组队完成

1. 安卓消息推送

场景：您的APP账户拥有n位联系人，他们会不定时的向您发送消息，请采用通知的方式提醒用户消息已经到达。

验收：

1. 通知显示：能够正确显示所有通知。包括通知小图标，通知标题、通知内容

2. 点按功能：点击通知时可以跳转到ContactActivity，并且在ContactActivity能够正确显示姓名。

3. 提醒式通知： 能够以提示式通知的形式弹出通知。

加分项：

1. 能够按照联系人进行分组。使用Group分类。

2. 能够直接进行回复。使用RemoteInput。

3. 使用MessagingStyle。

扩展：

1. 参见google 通知文档：

https://developer.android.com/guide/topics/ui/notifiers/notifications

2. 对应小图标要求：Android从5.0系统开始，对于通知栏图标的设计进行了修改。现在Google要求，所有应用程序的通知栏图标，应该只使用alpha图层来进行绘制，而不应该包括RGB图层。

3. 对于前台服务，检查前台权限设置。即：Android-Manifest.xml文件中是否有<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />这行代码

4. Intent可以携带数据，在通知中也不例外。

5. 消息推送往往是以通知的形式提醒用户的，有兴趣可以了解一下MQTT协议实现、XMPP协议实现、中国统一推送

6. 。。。