第十五章

訊息通知與穿戴式裝置

Notification/NotificationExample

訊息通知

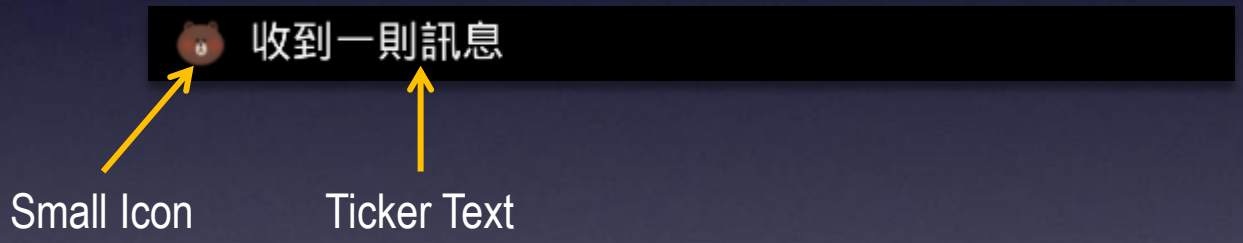
訊息通知

- 訊息通知(Notification)廣泛應用在Android之上，尤其近年來火紅的通訊app，更是訊息通知的使用大宗

- 本章將說明如何使用notification

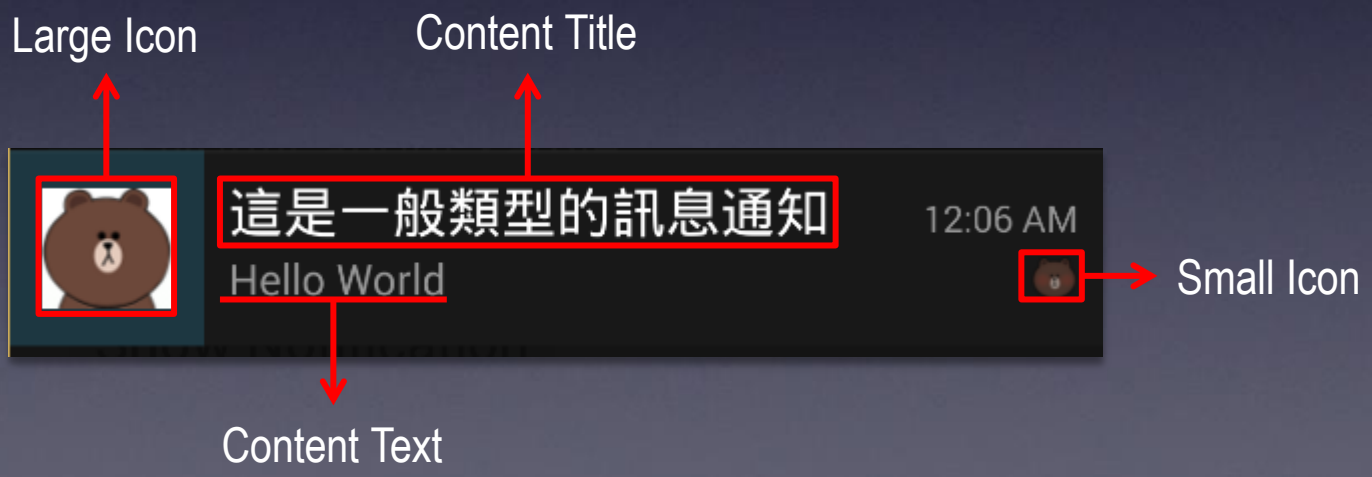
訊息元件

- 一般收到Notification後會有圖示和說明文字
- 圖示稱為Small Icon
- 文字為Ticker Text



訊息元件

- 下拉狀態條列，可以看到完整的訊息通知
- 最左方是Large Icon
- 上方的大文字是Content Title
- 下方的小文字是Content Text



訊息元件

- 點下Notification後會做甚麼，可以藉由PendingIntent決定
  - 如Line收到notification，點下後可以進入聊天室
  - Gmail收到新郵件後會出現notificaiton，點下後開啟郵件

- PendingIntent
  - PendingIntent是一種Intent的延伸
  - PendingIntent可以暫存在Android的作業系統中，等待系統排程後在執行

前置作業

- Notification需要用到Android Support Library v4
- 請先確認已從SDK Manager下載

建立訊息元件

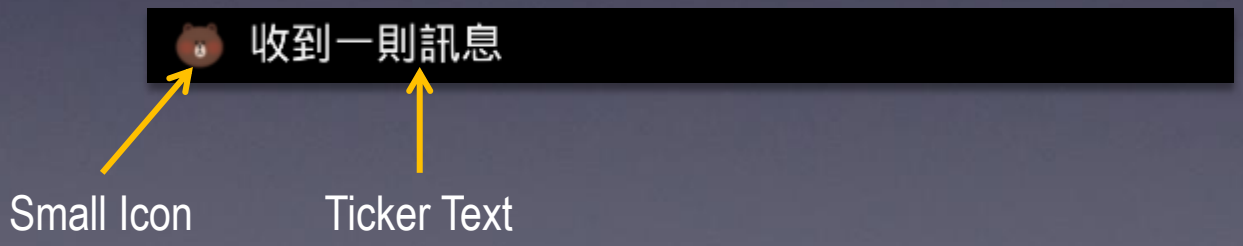
```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

- 使用NotificationCompat.Builder來建立訊息
  - 建構子的參數為Activity

建立訊息元件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

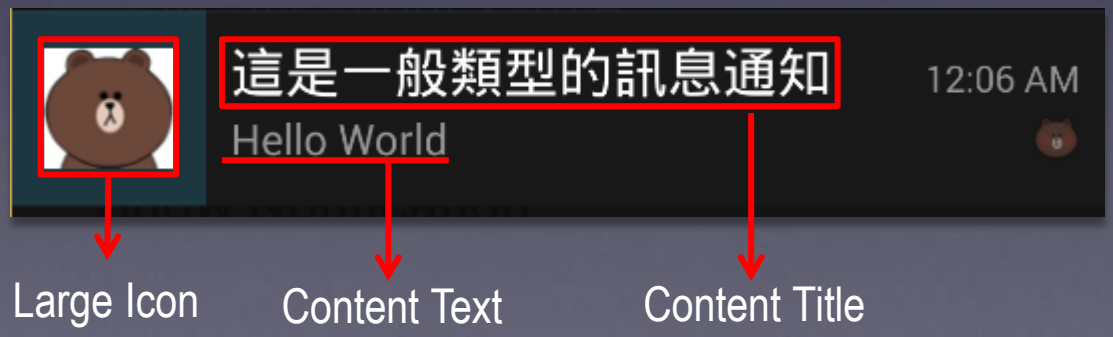
- setSmallIcon() 建立notification在狀態列時的小圖示
- setTicker() 建立小圖示旁的文字



建立訊息元件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

- setContentTitle(), setContentText()設定相關文字
- setLargeIcon() 設定圖示



建立訊息元件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

- getResources().getDrawable(resId)
  - 可以取得drawable
- BitmapDrawable
  - 若drawable是圖檔，系統會產生BitmapDrawable，記得轉型即可
  - BitmapDrawable.getBitmap()，可以取得Bitmap，也就是圖檔

建立訊息元件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

- setDefaults()
  - 設定震動、有系統預設的提示音、有系統預設的LED燈光

建立訊息元件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_notification)
.setTicker("收到一則訊息")
.setContentTitle("這是一般類型的訊息通知")
.setContentText("Hello World")
.setLargeIcon(((BitmapDrawable)getResources().getDrawable(R.drawabl
e.ic_large)).getBitmap())
.setDefaults(Notification.DEFAULT_VIBRATE |
Notification.DEFAULT_SOUND | Notification.FLAG_SHOW_LIGHTS)
.setContentIntent(peddingIntent);
```

- setContentIntent()
  - 設定使用者點下notification後要使用的PendingIntent
  - 要如何建立PendingIntent?

PENDING INTENT

```
Intent intent = new Intent(this, ResultActivity.class);
PendingIntent peddingIntent =
PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

- 之前有說明，PendingIntent是Intent的延伸，差別是可以在Android作業系統中停留的Intent
- 所以首先先還是建立Intent，本Intent是啟動ResultActivity

PENDING INTENT

```
Intent intent = new Intent(this, ResultActivity.class);
PendingIntent peddingIntent =
PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

- 若建立的Intent是為了開啟Activity，就使用PendingIntent.getActivity()
  - 參數1: Activity
  - 參數3: Intent
- 若是Broadcast: PendingIntent.getBroadcast()
- 若是Service: PendingIntent.getService()

發送出NOTIFICATION

```
Notification notification = builder.build();
NotificationManager mgr = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
mgr.notify(0, notification);
```

- 藉由NotificationCompat.Builder的build()方法可以建立Notification
- 要發出Notification必須得使用Android提供的NotificationManager
  - 使用getSystemService(Context.NOTIFICATON_SERVICE)並轉型取得NotificationManager

發送出NOTIFICATION

```
Notification notification = builder.build();
NotificationManager mgr = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
mgr.notify(0, notification);
```

- 使用NotificationManager.notify()發出Notification
  - 參數1: 給予notification的一個索引值，之後要從程式裡刪除或更新notification就要使用這個索引值
  - 參數2: 要發出的notification

PENDING INTENT接收端

- 建立一個ResultActivity負責接收notification點下後的PendingIntent

- 預期的行為
  - 是當notification被點選後會消失

PENDING INTENT接收端

```
NotificationManager mgr = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
mgr.cancel(0);
```

- 要清除notification必須得使用NotificationManager，取得方式與發出notification時相同

- NotificationManager.cancel()
  - 以上方法可以清除notificaiton
  - 參數：notification發出時的索引值

Notification/NotificationExample

其他類型的NOTIFICATION

其他類型的NOTIFICATION

- 在Android 4.1 JellyBean後，Android提供了另外三種型態的Notification
  - BigPicturen
  - BigText
  - InBox

- 以上三種類型都可以展開和收合
  - 使用兩隻手指頭對著notification向上下拖移即可
  - 收合時就與一般notification相同

BIGPICTURE STYLE

BIGPICTURE STYLE

```
NotificationCompat.BigPictureStyle bigPictureStyle =
new NotificationCompat.BigPictureStyle();
bigPictureStyle
.bigPicture(……)
.setBigContentTitle("大圖片展開了")
.setSummaryText("可用兩指上拉關閉大圖");

builder.setStyle(bigPictureStyle);
```

- 使用BigPictureStyle建立大圖片型的物件

BIGPICTURE STYLE

```
NotificationCompat.BigPictureStyle bigPictureStyle =
new NotificationCompat.BigPictureStyle();
bigPictureStyle
.bigPicture(……)
.setBigContentTitle("大圖片展開了")
.setSummaryText("可用兩指上拉關閉大圖");

builder.setStyle(bigPictureStyle);
```

- setPicture(), setBigContentTitle(), setSummaryText() 請對照前面的圖示內容

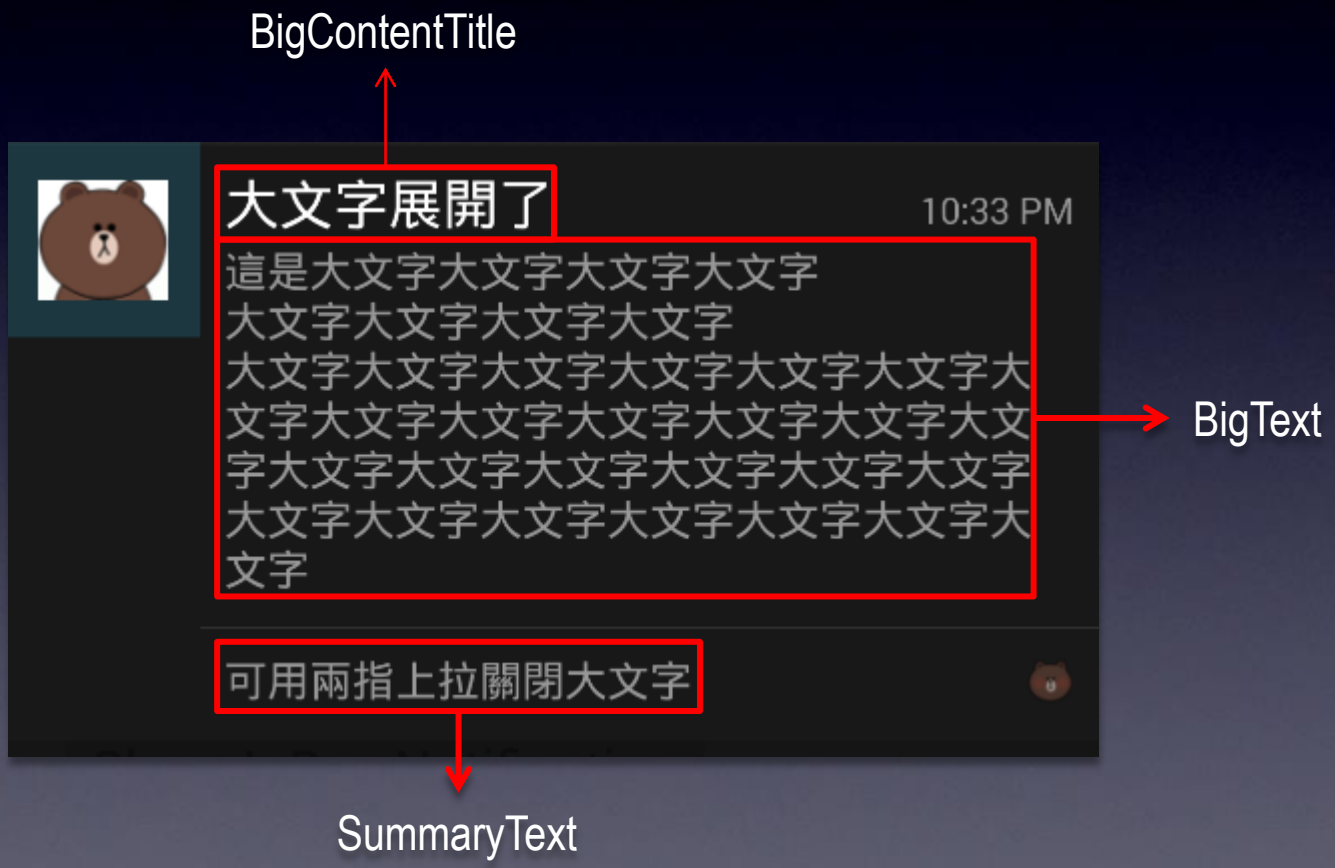
BIGPICTURE STYLE

```
NotificationCompat.BigPictureStyle bigPictureStyle =
new NotificationCompat.BigPictureStyle();
bigPictureStyle
.bigPicture(……)
.setBigContentTitle("大圖片展開了")
.setSummaryText("可用兩指上拉關閉大圖");

builder.setStyle(bigPictureStyle);
```

- builder是NotificationCompat.Builder
- 要將設定好的Style物件設定給Builder

BIGTEXT STYLE



BIGTEXT STYLE

```
NotificationCompat.BigTextStyle bigTextStyle = new
NotificationCompat.BigTextStyle();
bigTextStyle
.setBigContentTitle("大文字展開了")
.setSummaryText("可用兩指上拉關閉大文字")
.bigText(“…”);

builder.setStyle(bigTextStyle);
```

- 使用BigTextStyle建立Style物件

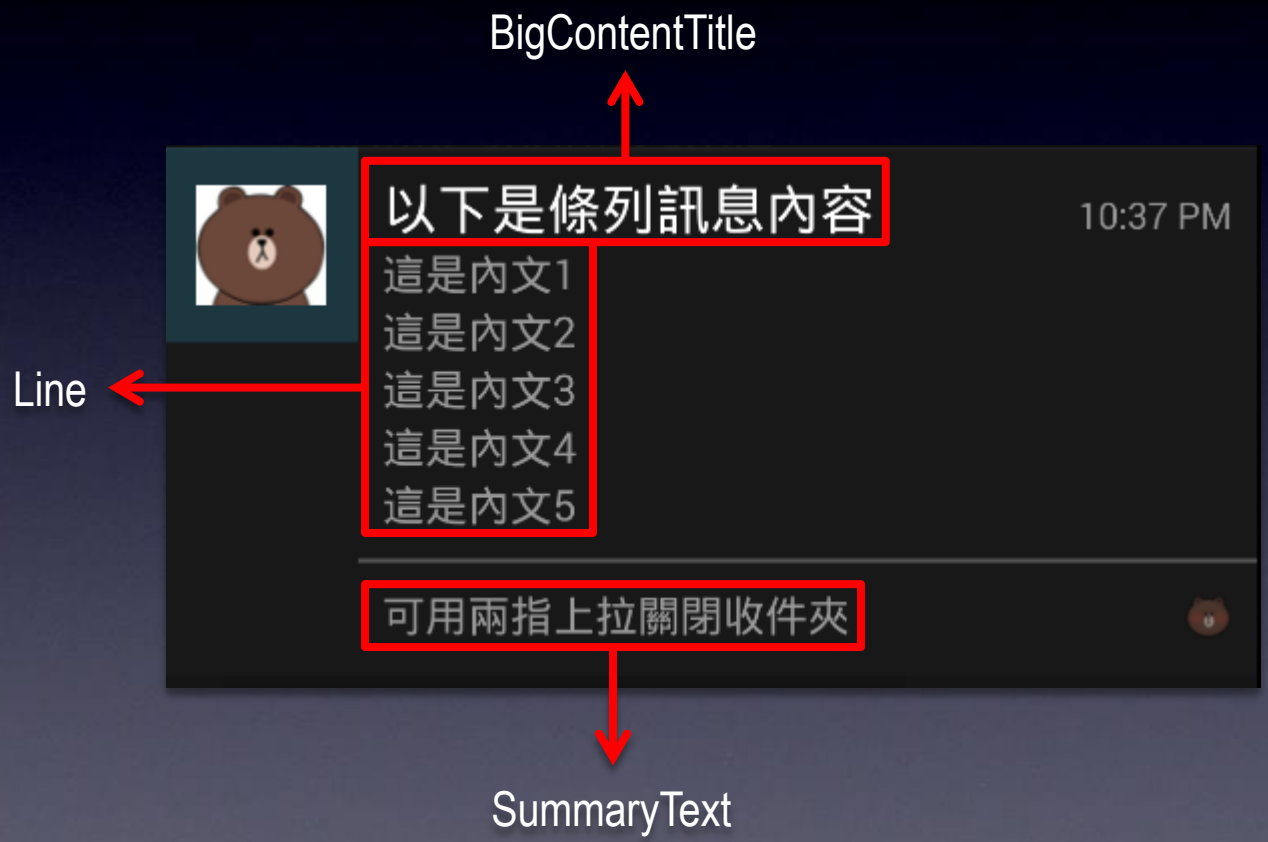
BIGTEXT STYLE

```
NotificationCompat.BigTextStyle bigTextStyle = new
NotificationCompat.BigTextStyle();
bigTextStyle
.setBigContentTitle("大文字展開了")
.setSummaryText("可用兩指上拉關閉大文字")
.bigText(“…”);

builder.setStyle(bigTextStyle);
```

- setBigContentTitle(), setSummaryText(), bigText() 請對照前面的圖示
- 最後記得要將Style物件設定給NotificationCompat.Builder

INBOX STYLE


BigContentTitle
以下是條列訊息內容
10:37 PM
這是內文1
這是內文2
這是內文3
這是內文4
這是內文5
Line
可用兩指上拉關閉收件夾
SummaryText

INBOX STYLE

```
NotificationCompat.InboxStyle inBoxStyle = new
NotificationCompat.InboxStyle();
inBoxStyle
.setBigContentTitle("以下是條列訊息內容")
.setSummaryText("可用兩指上拉關閉收件夾")
.addLine("這是內文1")
.addLine("這是內文2")
.addLine("這是內文3")
.addLine("這是內文4")
.addLine("這是內文5")
builder.setStyle(inBoxStyle);
```

- 使用InBoxStyle建立Style物件

INBOX STYLE

```
NotificationCompat.InboxStyle inBoxStyle = new
NotificationCompat.InboxStyle();
inBoxStyle
.setBigContentTitle("以下是條列訊息內容")
.setSummaryText("可用兩指上拉關閉收件夾")
.addLine("這是內文1")
.addLine("這是內文2")
.addLine("這是內文3")
.addLine("這是內文4")
.addLine("這是內文5");
builder.setStyle(inBoxStyle);
```

- setBigContentTitle()和setSummaryText()請對照前面的圖示

INBOX STYLE

```
NotificationCompat.InboxStyle inBoxStyle = new
NotificationCompat.InboxStyle();
inBoxStyle
.setBigContentTitle("以下是條列訊息內容")
.setSummaryText("可用兩指上拉關閉收件夾")
.addLine("這是內文1")
.addLine("這是內文2")
.addLine("這是內文3")
.addLine("這是內文4")
.addLine("這是內文5");
builder.setStyle(inBoxStyle);
```

- 使用addLine可以加入在notification中每一行要顯示的內容

INBOX STYLE

```
NotificationCompat.InboxStyle inBoxStyle = new
NotificationCompat.InboxStyle();
inBoxStyle
.setBigContentTitle("以下是條列訊息內容")
.setSummaryText("可用兩指上拉關閉收件夾")
.addLine("這是內文1")
.addLine("這是內文2")
.addLine("這是內文3")
.addLine("這是內文4")
.addLine("這是內文5");
builder.setStyle(inBoxStyle);
```

- 最後記得要將Style物件設定給NotificationCompat.Builder

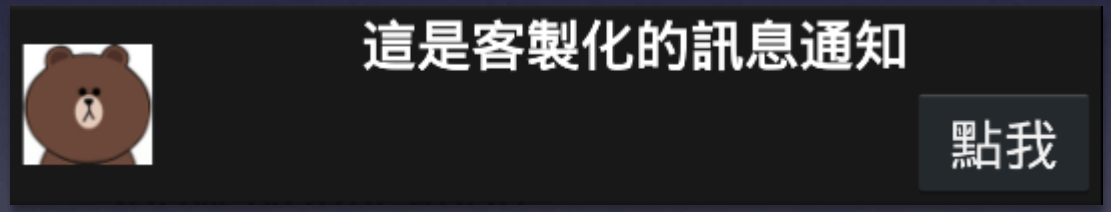
Notification/NotificationExample

自訂NOTIFICATION外觀

自訂NOTIFICATION外觀

- Android中提供使用者客製化notification

- 若要做到下述的notification，而且按鈕還可以點選，該怎麼做？



建立LAYOUT

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <ImageView
        android:id="@+id/icon" .../>
    <LinearLayout ...>
        <TextView
                android:id="@+id/title" .../>
        <Button
        android:id="@+id/btn_toast" .../>
    </LinearLayout>
</RelativeLayout>
```

- 在自訂的notification畫面中，能使用的View不多，大致上是Button, ImageView, TextView, ProgressBar
- 用了其他的View會在執行時出現錯誤
- 隨著Android的版本進化，能使用的View會越來越多
- JellyBean以前的版本，notification的高都相同，無法自訂

建立訊息物件

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder
.setTicker("收到一則訊息")
.setSmallIcon(R.drawable.ic_notification)
.setContentTitle("這是客製化的訊息通知")
.setContentIntent(peddingIntent)
.setContent(remoteViews);
```

- setContent()內擺放RemoteViews
- 對於app來說，要顯示notification是一種遠端操作，所以Android推出了RemoteViews，供給開發者能夠在非自己的app內依然能做出具有互動性的畫面
- 如何建立RemoteViews?

建立REMOTEVIEWS

```
RemoteViews remoteViews = new
RemoteViews(getPackageName(), R.layout.custom_view);
remoteViews.setImageViewResource(R.id.icon,
R.drawable.ic_large);
remoteViews.setTextViewText(R.id.title, "這是客製化的訊息
通知");
remoteViews.setOnClickPendingIntent(R.id.btn_toast,
clickPendingIntent);
```

- 使用new來建立RemoteViews
  - 參數1：Activity所在的package名稱
  - 參數2：要使用的layout

建立REMOTEVIEWS

```
RemoteViews remoteViews = new
RemoteViews(getPackageName(), R.layout.custom_view);
remoteViews.setImageViewResource(R.id.icon,
R.drawable.ic_large);
remoteViews.setTextViewText(R.id.title, "這是客製化的訊息
通知");
remoteViews.setOnClickPendingIntent(R.id.btn_toast,
clickPendingIntent);
```

- RemoteViews.setImageViewResource()設定drawable給ImageView
  - 參數1：ImageView的view id
  - 參數2：drawable

建立REMOTEVIEWS

```
RemoteViews remoteViews = new
RemoteViews(getPackageName(), R.layout.custom_view);
remoteViews.setImageViewResource(R.id.icon,
R.drawable.ic_large);
remoteViews.setTextViewText(R.id.title, "這是客製化的訊息
通知");
remoteViews.setOnClickPendingIntent(R.id.btn_toast,
clickPendingIntent);
```

- RemoteViews.setTextViewText()設定文字給TextView
  - 參數1：TextView的View id
  - 參數2：要設定給TextView的文字

建立REMOTEVIEWS

```
RemoteViews remoteViews = new
RemoteViews(getPackageName(), R.layout.custom_view);
remoteViews.setImageViewResource(R.id.icon,
R.drawable.ic_large);
remoteViews.setTextViewText(R.id.title, "這是客製化的訊息
通知");
remoteViews.setOnClickPendingIntent(R.id.btn_toast,
clickPendingIntent);
```

- RemoteViews.setOnClickPendingIntent()設定Button按下後發出的PendingIntent
  - 參數1：要知道按下的View id
  - 參數2：按下後會由系統發出的PendingIntent

JELLY BEAN後的加強

```
if (android.os.Build.VERSION.SDK_INT >
android.os.Build.VERSION_CODES.JELLY_BEAN) {
        RemoteViews remoteBigViews = new
        RemoteViews(getPackageName(), R.layout.custom_big_view);
        remoteBigViews.setTextViewText(R.id.title, "雙指上拉關閉");
        remoteBigViews.setImageViewResource(R.id.icon,
        R.drawable.large_image);
        notification.bigContentView = remoteBigViews;
}
```

- 在JellyBean後，自訂的notification還可以加上展開的自訂畫面，稱為bigContentView
- 藍字為判斷版本的方式

JELLY BEAN後的加強

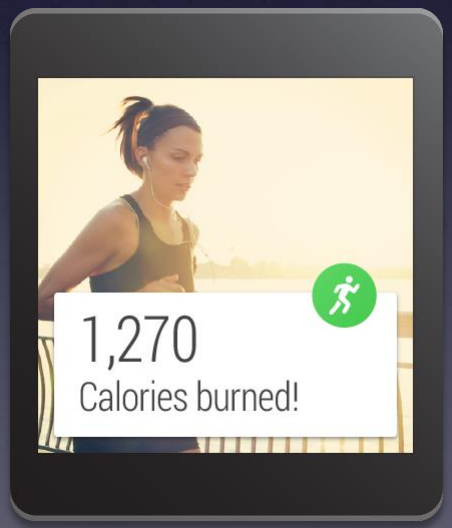
```
if (android.os.Build.VERSION.SDK_INT >
android.os.Build.VERSION_CODES.JELLY_BEAN) {
        RemoteViews remoteBigViews = new
        RemoteViews(getPackageName(), R.layout.custom_big_view);
        remoteBigViews.setTextViewText(R.id.title, "雙指上拉關閉");
        remoteBigViews.setImageViewResource(R.id.icon,
        R.drawable.large_image);
        notification.bigContentView = remoteBigViews;
}
```

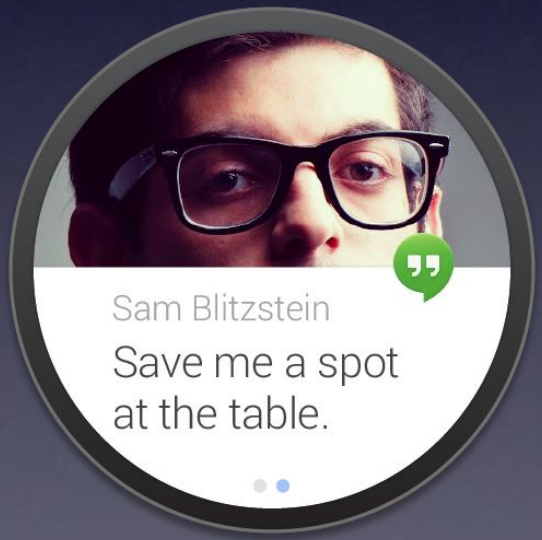
- 使用notification.bigContentView將RemoteViews設定給notification

ANDROID WEAR

ANDROID WEAR

- Android Wear是Google新推出的穿戴式科技
- 目前只支援Android KitKat以及更新的Android
- 目前提供模擬器和SDK，供申請者加入



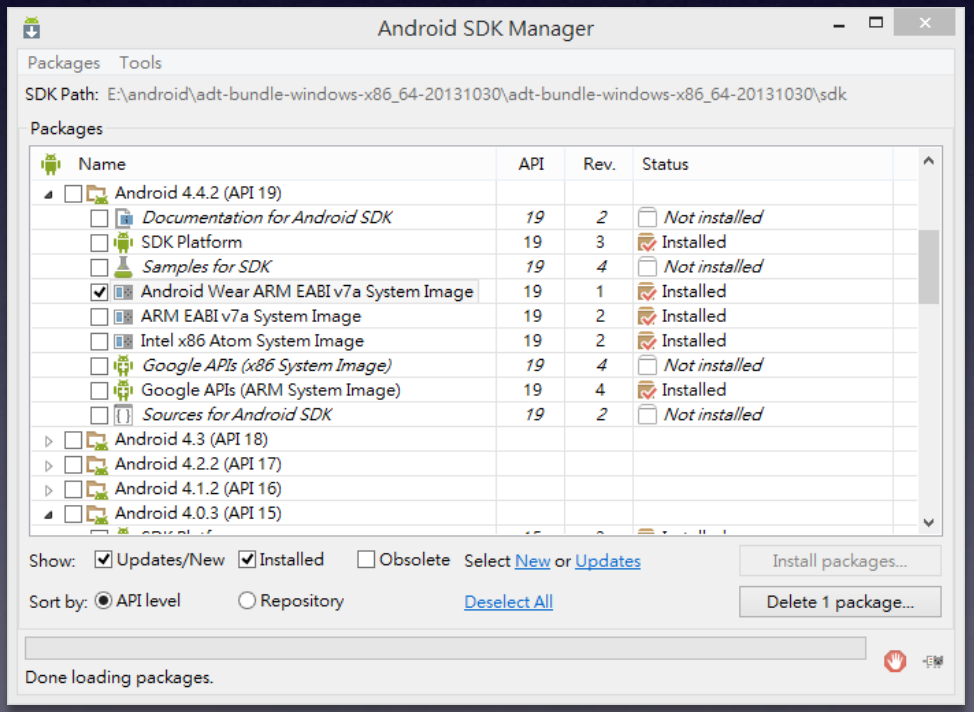


ANDROID WEAR

- 註冊處
  - http://developer.android.com/wear/preview/start.html
- 註冊完24小時後(有時候會比較快)會收到認證信函，其中就包含了範例和手機端需要用到的App (Android Wear Preview)

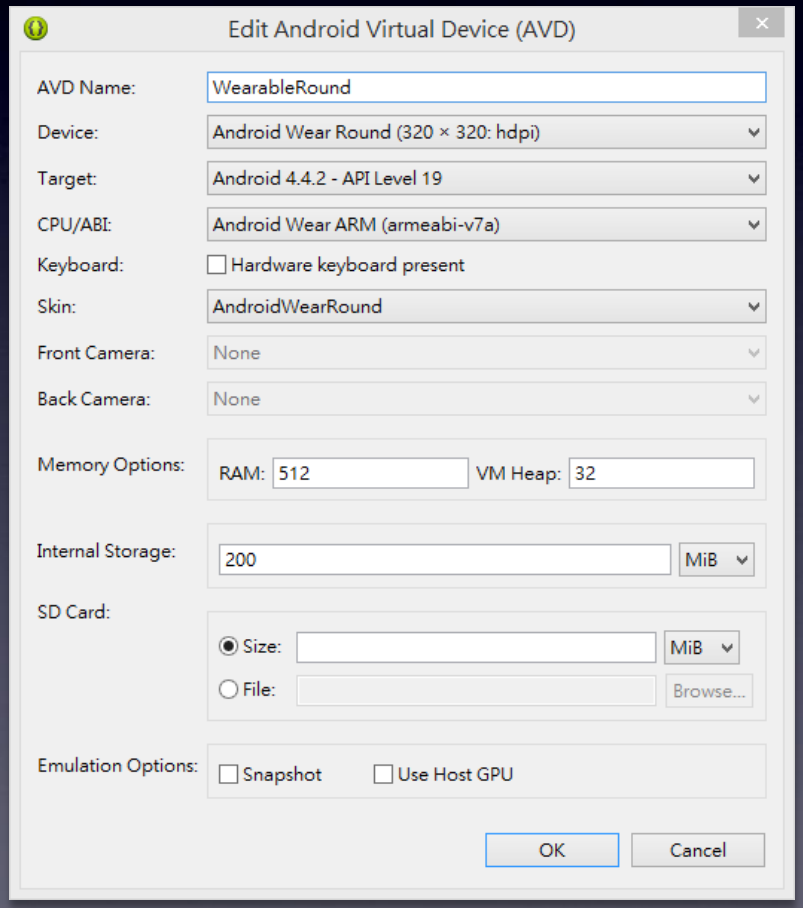
ANDROID WEAR

- 設定穿戴裝置模擬器
  - 打開SDK Manager
  - 勾選Android Wear ARM EABI v7a System Image並下載



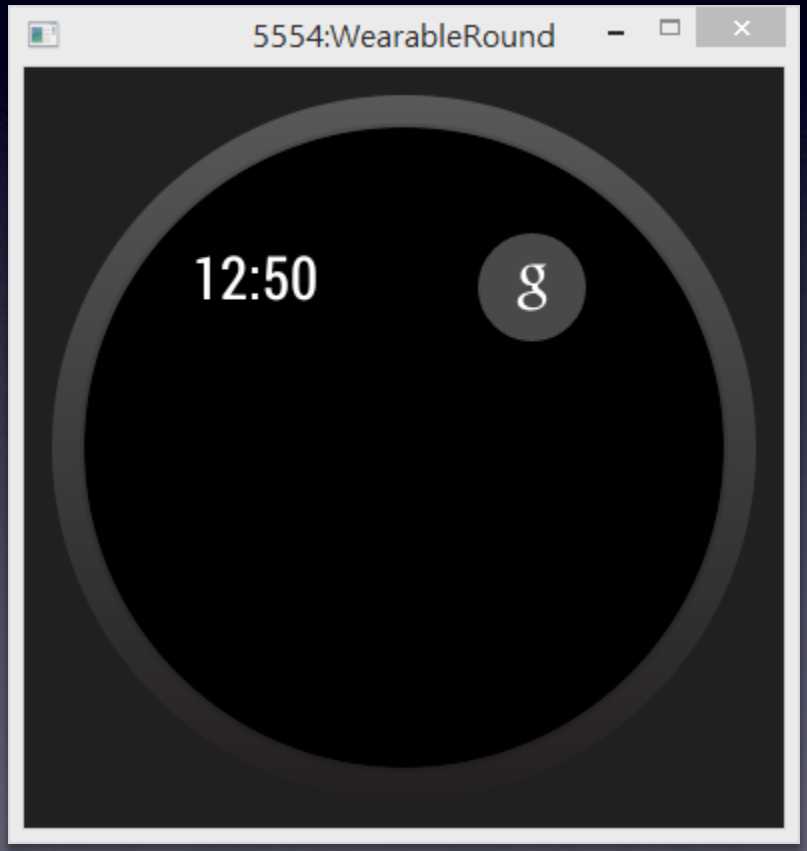
ANDROID WEAR

- 打開AVD Manager建立穿戴裝置模擬器



ANDROID WEAR

- 啟動Android Wear Simulator，並確認啟動成功

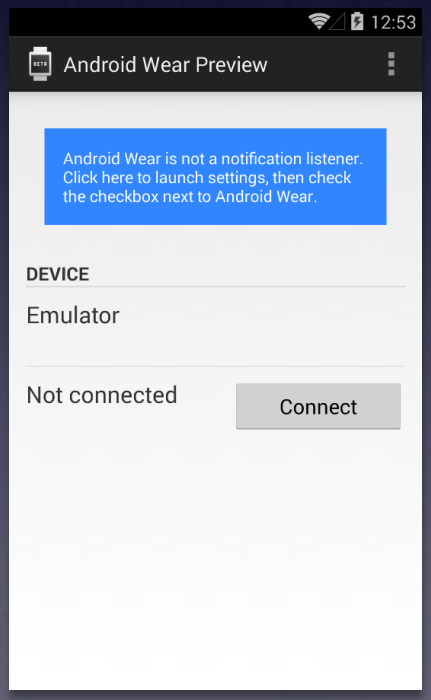


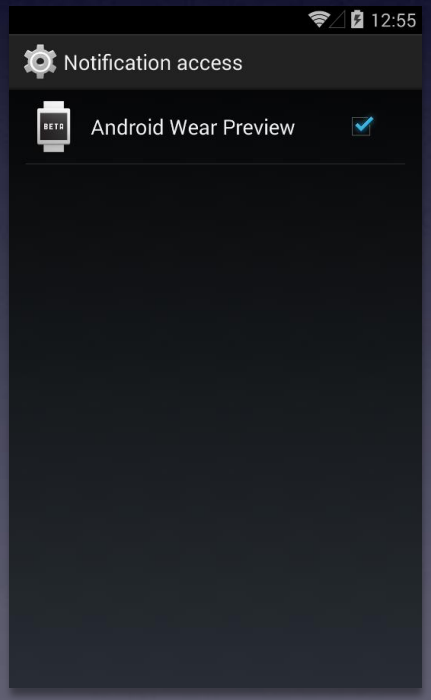
ANDROID WEAR

- 安裝由認證信中寄出的Android Wear Preview app到手機上
- 手機由USB連接到電腦，確認模擬器已經執行成功
- 在指令模式打入以下指令
  - `adb -d forward tcp:5601 tcp:5601`

ANDROID WEAR

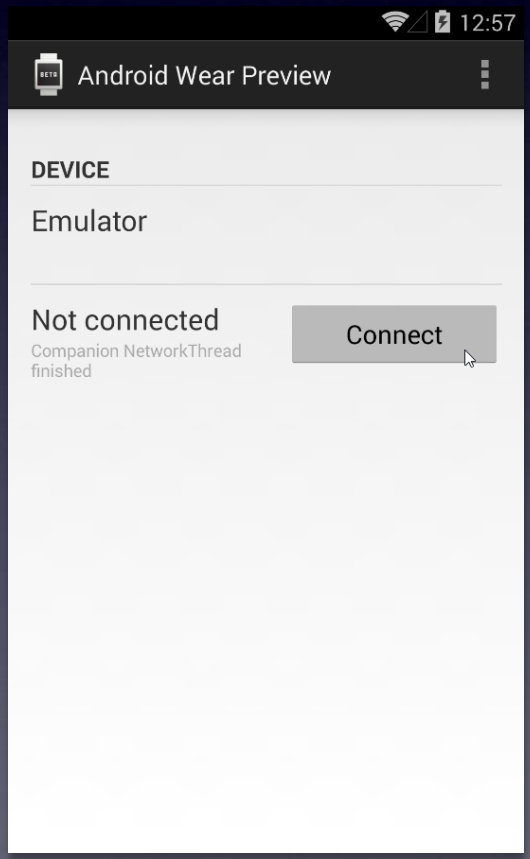
- 打開Android Wear Preview app
- 點選藍色按鈕
- 勾選Android Wear Preview

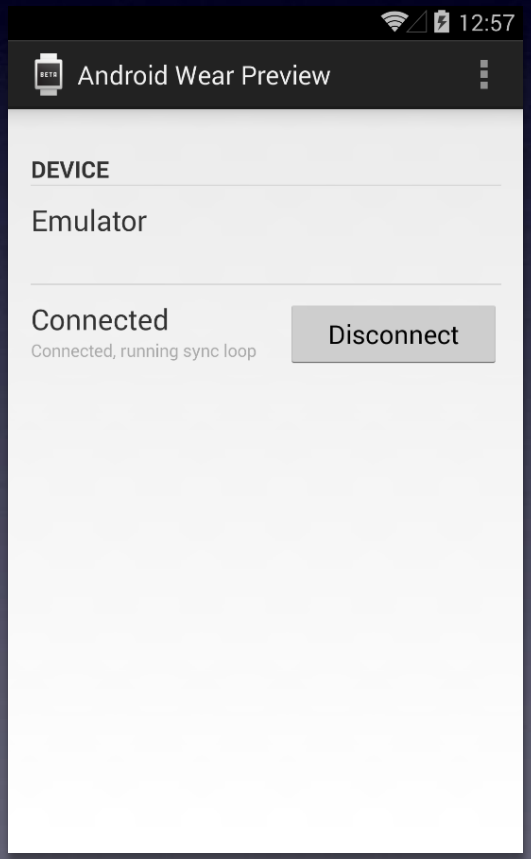




ANDROID WEAR

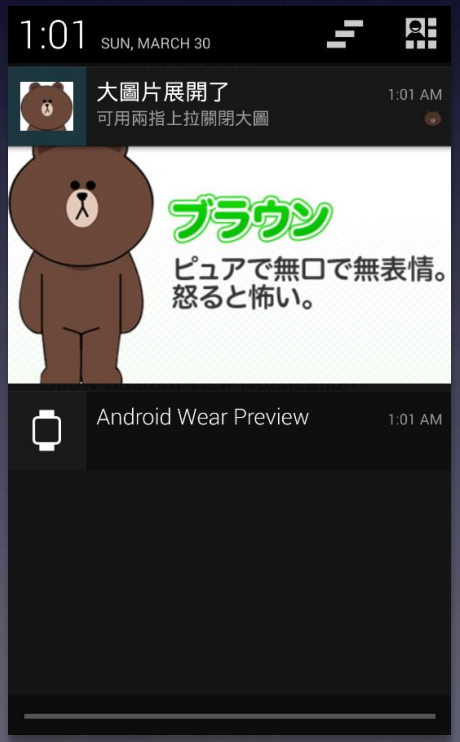
- 回到以下畫面，並按下Connect連結手機
- 右圖為連結完成





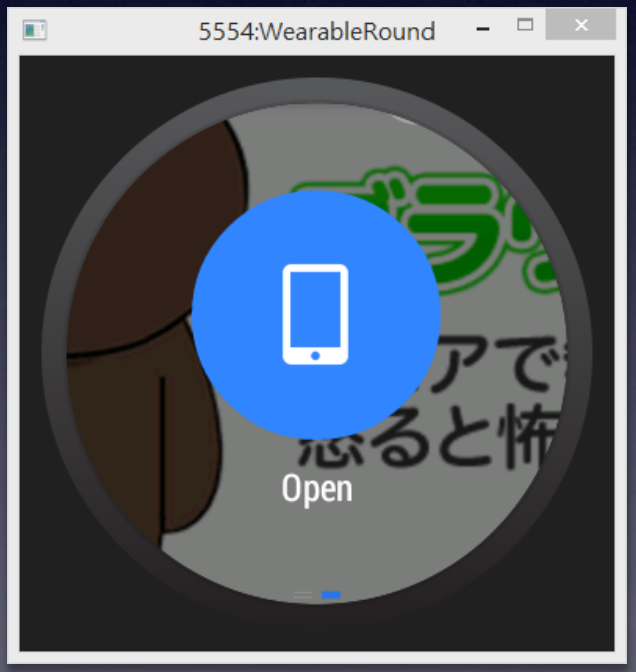
ANDROID WEAR

- 連結完成後，運行我們之前的範例
- 可以看到出現了各種notification，就會反映在Wear的模擬器上



ANDROID WEAR

- 可以將Wear模擬器上的notification拖到左邊，就會出現選項可供選擇
- 選擇的項目將會直接影響到連接的手機

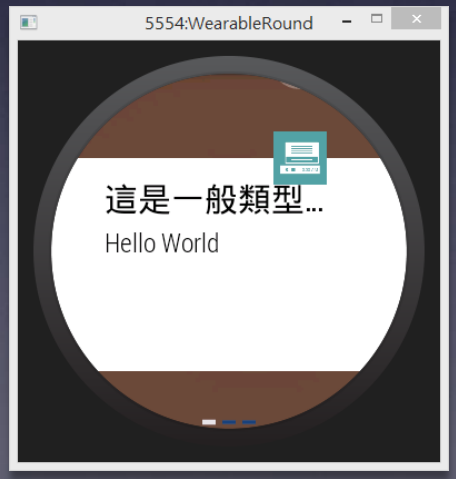


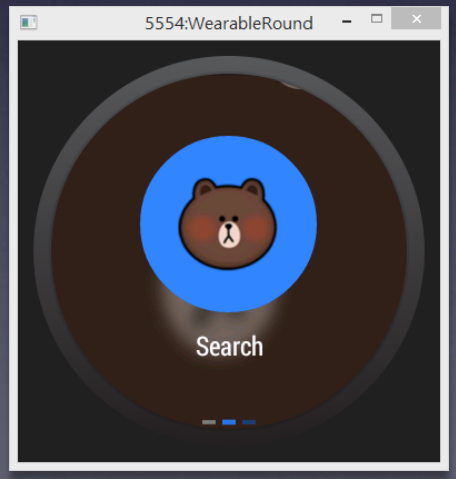
Notification/NotificationExample

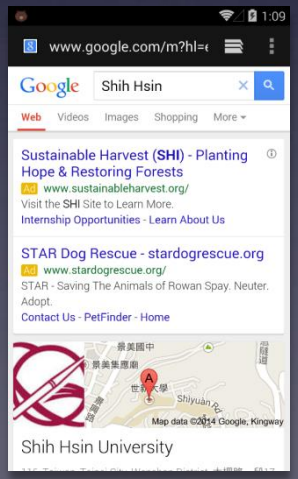
增加穿戴裝置的控制

增加穿戴裝置的控制

- 可以增加notification在Wear裝置上對手機的控制
- 本例子將會產生notificaton，在Wear裝置上收到該notification後，可以讓使用者選擇搜尋的功能
- 當使用者按下搜尋，連接的裝置便會開啟瀏覽器進行關鍵字的搜尋







增加控制

```
Intent searchIntent = new Intent(Intent.ACTION_WEB_SEARCH);
searchIntent.putExtra(SearchManager.QUERY, "Shih Hsin");
PendingIntent actionIntent = PendingIntent.getActivity(this,
0, searchIntent, PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
builder……
.addAction(R.drawable.ic_search, "Search", actionIntent);
```

- addAction
  - 參數1: 圖示
  - 參數2: 呈現的文字
  - 參數3: 點下後由Wear裝置發給手機的PendingIntent
- 一個notification，addAction()可以設定不只一次

更多資料

更多資料

- http://developer.android.com/wear/index.html?utm_source=ausdroid.net

- http://developer.android.com/reference/android/preview/support/v4/app/NotificationManagerCompat.html