

# Lab5: 提示訊息元件

## 一、本節目的：

- 學習利用 Toast 的方法顯示文字訊息
- 學習利用客製化 Toast 的方法同時顯示文字和圖片訊息
- 透過 AlertDialog 顯示提示訊息
- 透過 AlertDialog 顯示陣列資料

## 二、觀念說明：

我們很常在應用程式中遇到按下某個按鈕或是畫面時，系統會彈出訊息或是對話框於畫面上，本章節會教導如何實現這幾種常用的提示訊息。

### 1 Toast-快顯訊息

#### 1.1 介紹

Toast 是一種很快速的即時訊息，常用在通知使用者各種立即的資訊上，顯示後幾秒內就會消失。Toast 主要可以應用在反應某個操作下的回饋，例如告知使用者某些設定上的成功與否。也很常被作為 debug 手段。



## 1.2 用法

Toast 最簡單的使用方法是透過 Toast 的靜態函式 `makeText` 來產生文字內容，方法如下：

```
Toast.makeText(this, "message", Toast.LENGTH_SHORT).show();
```

`makeText` 的第一個的參數要傳入調用 Toast 的對象，通常為自身的 Activity 實體(`this`)。第二個的參數為傳入字串，作為輸出畫面的內容。第三個的參數為持續時間，`LENGTH_SHORT` 持續時間較短，`LENGTH_LONG` 持續時間較長。

`makeText` 產生後的結果會是個 Toast 的實體，就可以使用 `show()` 的方法將 Toast 資訊顯示到螢幕上。

## 1.3 客製化 Toast

除了用 `makeText` 簡單而快速的產生 Toast 之外，也可做到位置改變或是自訂顯示的內容。這邊我們就需要先了解幾個 Toast 提供方法，實現的程式碼如下：

```
Step1：初始化 Toast
Toast toast = new Toast(context: MainActivity.this);

Step2：Toast 在畫面中顯示位置
toast.setGravity(Gravity.TOP, xOffset: 0, yOffset: 50);

Step3：Toast 在畫面中顯示的持續時間
toast.setDuration(Toast.LENGTH_SHORT);

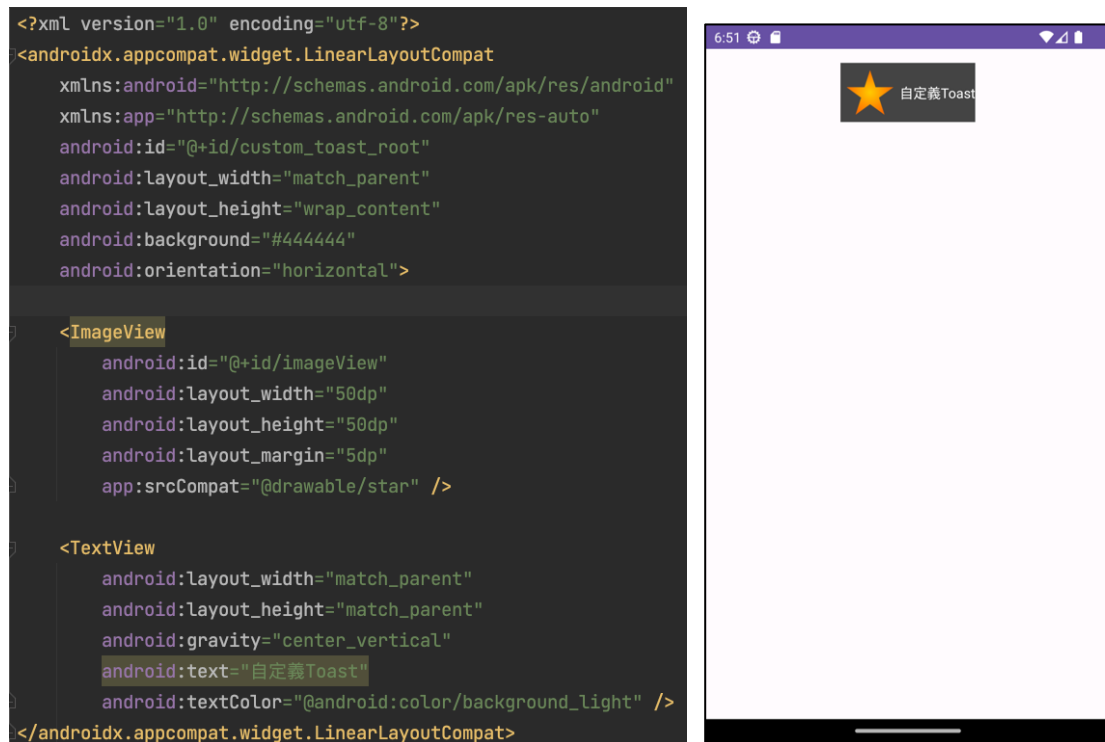
Step4：放入自定義的畫面
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast
    , (ViewGroup)findViewById(R.id.custom_toast_root));
toast.setView(layout);

Step5：顯示畫面
toast.show();
```

- 1) 我們需要自行創建出 Toast 實體，這部分與 `makeText` 相同，都要傳入使用對象。
- 2) `setGravity()` 方法可以指定我們的 Toast 位置。第一個的參數要傳入 Toast 要貼齊的方向。第二與第三個的參數則是傳入要與貼齊方向的長與寬間距。

- 3) `setDuration()` 為持續時間，用法與 `makeText` 的第三個參數一樣。
- 4) `Toast` 不只是能顯示文字，不過當我們希望呈現出更複雜的畫面，例如有圖片或文字，或兩者並存顯示的 `Toast` 時，就需要自行設計 `Xml` 畫面。我們可以與 `Activity` 設計顯示元件方式一樣，透過 `layout(xml)` 設計畫面，然後將完成的 `layout(xml)` 透過 `setView()` 方法來放入 `Toast` 中呈現。

■ 如下為 `toast_layout.xml` 即為要讓 `Toast` 顯示的畫面設計。

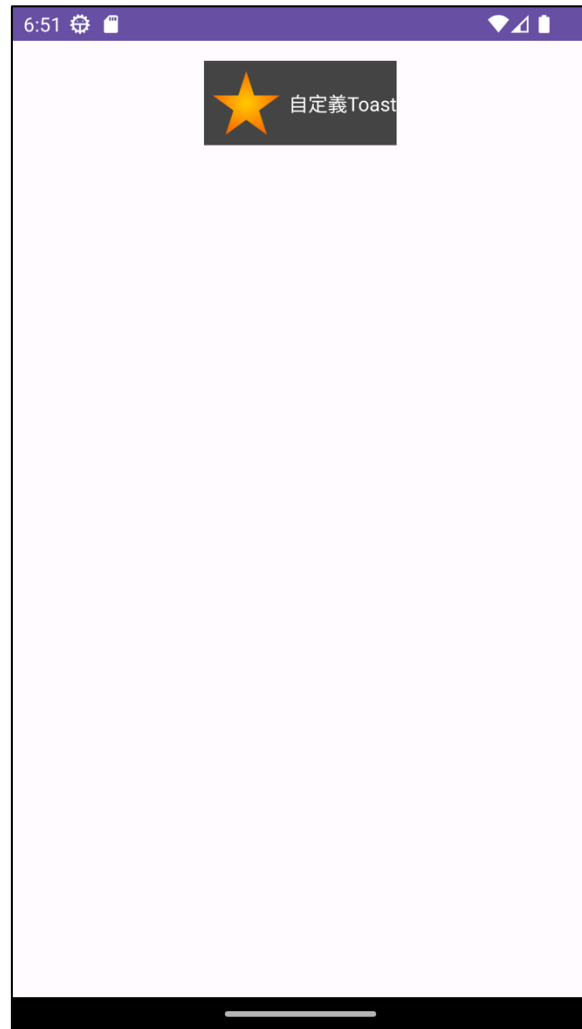


■ 而程式中需要使用

`getLayoutInflater().inflate(R.layout.toast_layout,(ViewGroup)findViewById(R.id.toast_layout_root))` 這方法來得到 `toast_layout` 的 `layout`。之後便可將該 `layout` 傳入 `setView()` 以設定顯示畫面

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast
    , (ViewGroup)findViewById(R.id.custom_toast_root));
toast.setView(layout);
```

5) 透過 show() 的方法，將 Toast 做顯示。



## 2 AlertDialog-對話方塊

### 2.1 介紹

當我們想要彈出訊息，並且希望使用者能與其互動，這時我們會使用 AlertDialog。

AlertDialog 對話方塊像 Windows 的彈跳視窗，他功能非常強大，不只是可以放上文字，還可以放上任何元件。



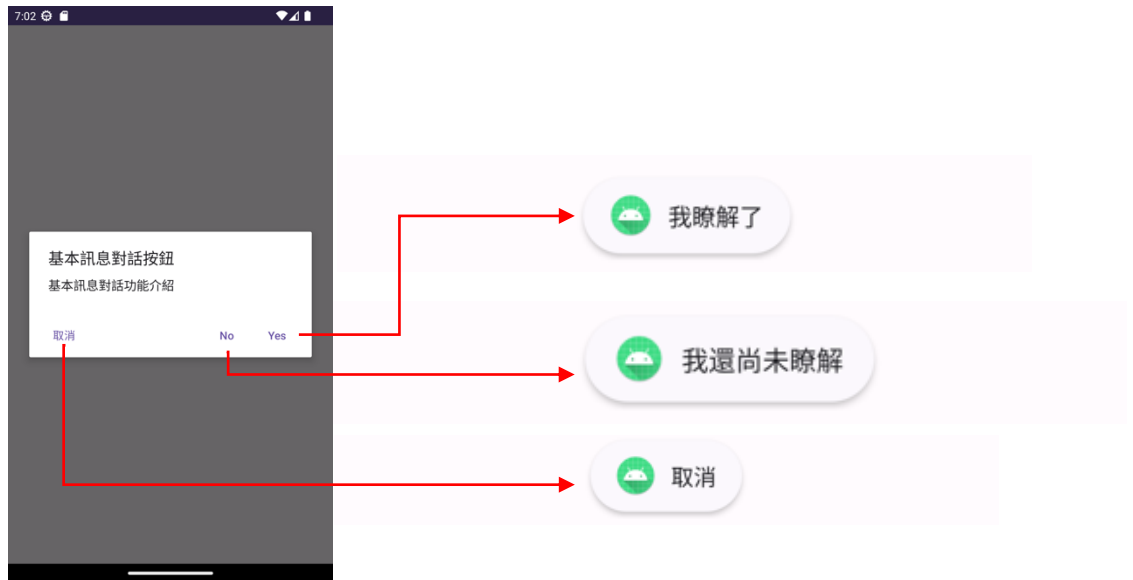
### 2.2 用法

與 Toast 相比，AlertDialog 的功能複雜很多，因此我們先從基本的提供的功能理解起：

- setTitle(): 對話方塊的標題
- setMessage(): 對話方塊的文字內容
- setItems(): 對話方塊的列表內容
- setPositiveButton(): 在對話方塊中加入正面的按鈕
- setNegativeButton(): 在對話方塊中加入負面的按鈕
- setNeutralButton(): 在對話方塊中加入中立的按鈕
- show(): 顯示對話方塊

在產生的 AlertDialog 實體中，對話方塊會依據裝置的不同會有不同的顯示面板，以下是幾個實作的案例：

- 含確定、拒絕與取消按鈕的對話框



```
final AlertDialog.Builder dialog = new AlertDialog.Builder(context: MainActivity.this);
dialog.setTitle("基本訊息對話按鈕");
dialog.setMessage("基本訊息對話功能介紹");

dialog.setNeutralButton(text: "取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(context: MainActivity.this, text: "取消", Toast.LENGTH_SHORT).show();
    }
});

dialog.setNegativeButton(text: "No", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(context: MainActivity.this, text: "我還尚未瞭解", Toast.LENGTH_SHORT).show();
    }
});

dialog.setPositiveButton(text: "Yes", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(context: MainActivity.this, text: "我瞭解了", Toast.LENGTH_SHORT).show();
    }
});

dialog.show();
```

setPositiveButton()、setNegativeButton()、setNeutralButton()主要影響按鈕位置，實際使用時的可不依照定義去使用。其中，兩個參數中第一個是按鈕名稱，第二個則要傳入 DialogInterface 類別下的監聽器(OnClickListener)來做事件處理。

- 含列表的對話框



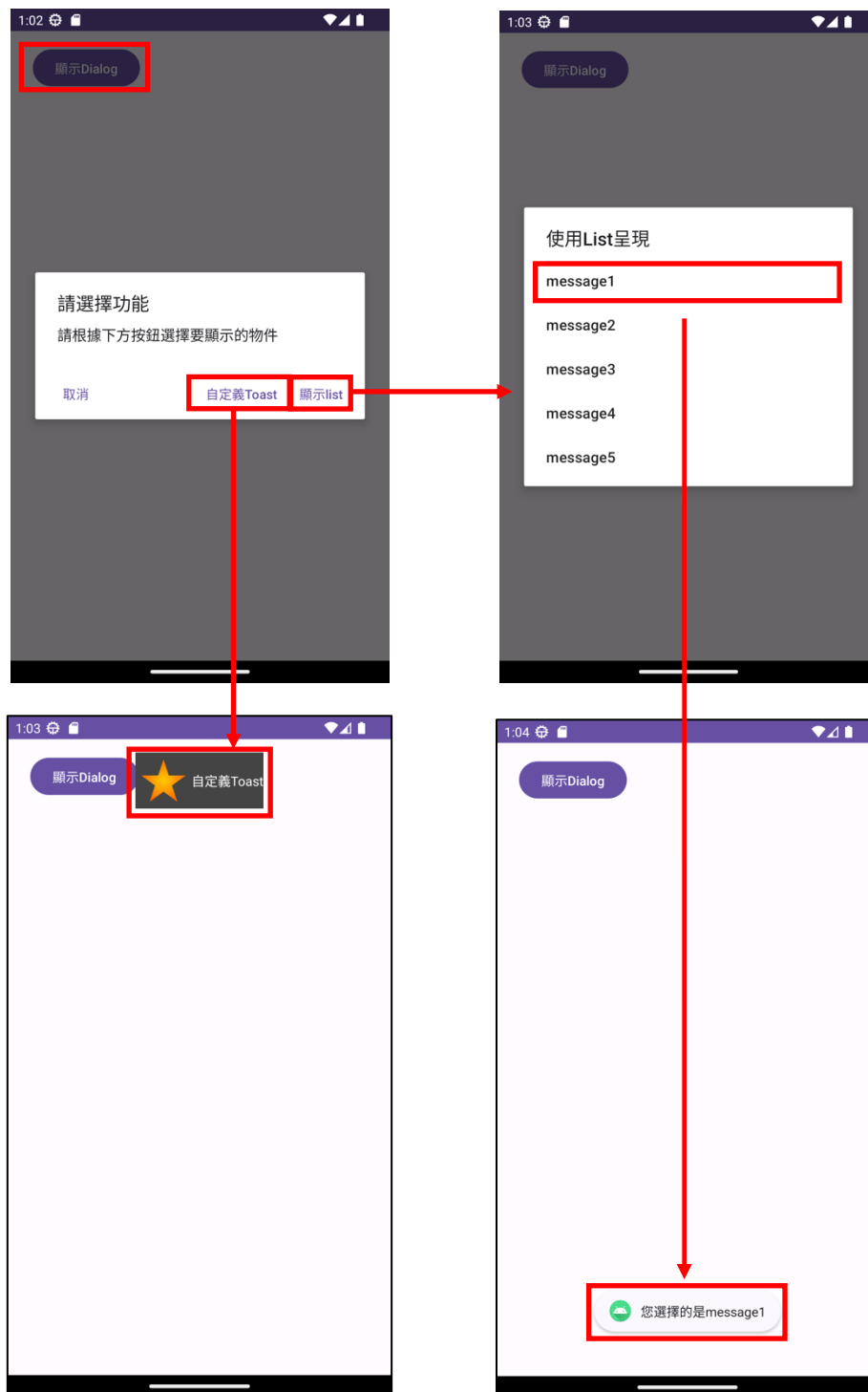
建立要顯示在列表上的字串

```
final String[] list = {"message1", "message2", "message3", "message4", "message5"};
AlertDialog.Builder dialog_list = new AlertDialog.Builder(context: MainActivity.this);
dialog_list.setTitle("使用List呈現"); // 建立 AlertDialog 物件
dialog_list.setItems(list, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(context: MainActivity.this, text: "您選擇的是"+list[i],
            , Toast.LENGTH_SHORT).show(); // 依照被點擊的項目顯示字串
    }
});
dialog_list.show();
```

列表使用 `setItems()` 來顯示列表項目，第一個參數需要傳入一個字串陣列，第二個則要傳入 `DialogInterface` 類別下的監聽器 (`OnClickListener`) 來做事件處理，`onClick` 事件處理的第二個參數會回傳點擊的項目編號 (依照陣列的順序)。

### 三、設計重點：

- 設計一個 AlertDialog，並根據下方按鈕選擇顯示不同的功能
- 點擊”顯示 LIST” 按鈕，顯示 AlertDialog 的 List
- 點擊”自定義 TOAST” 按鈕，顯示客製化的 Toast
- 點擊”取消” 按鈕，顯示原生 Toast

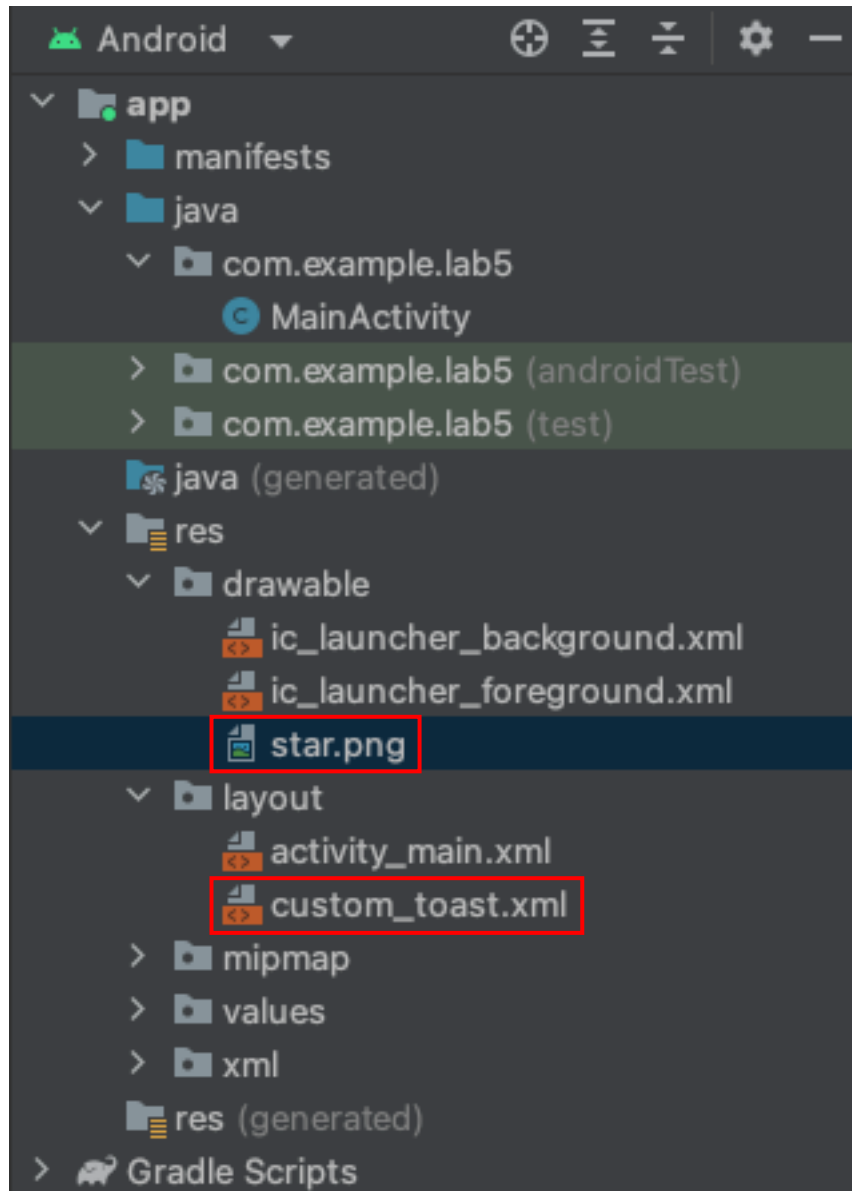




#### 四、設計步驟:

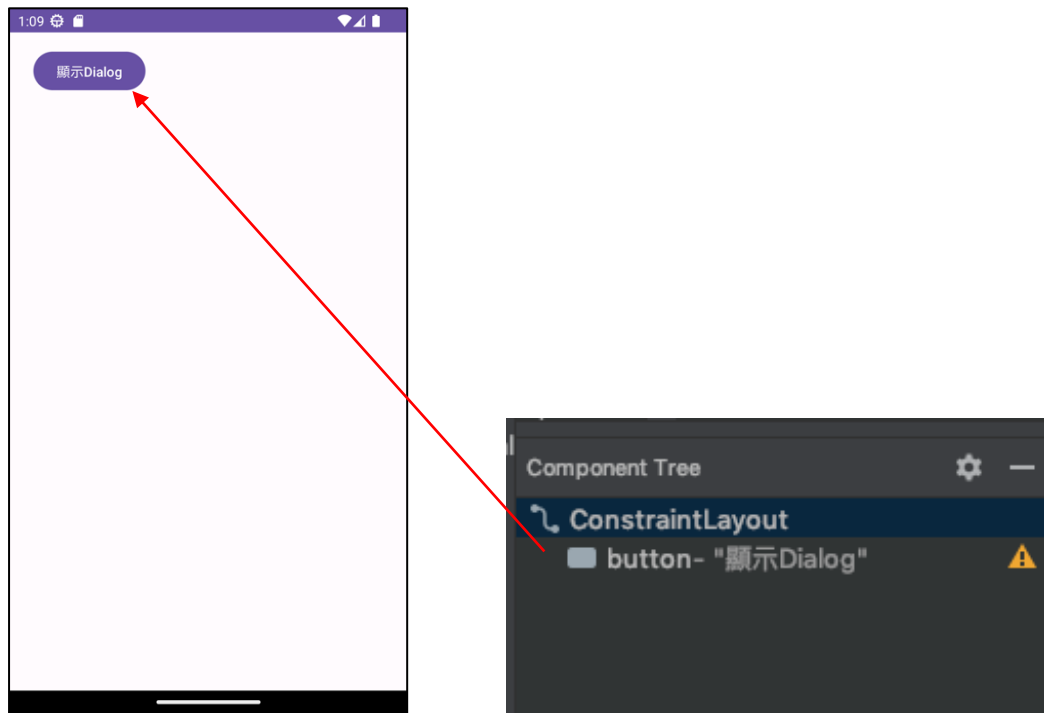
##### Step1

建立專案，並將附件的 star.png 放於 drawable 底下



## Step2

### 1) 繪製 activity\_main.xml



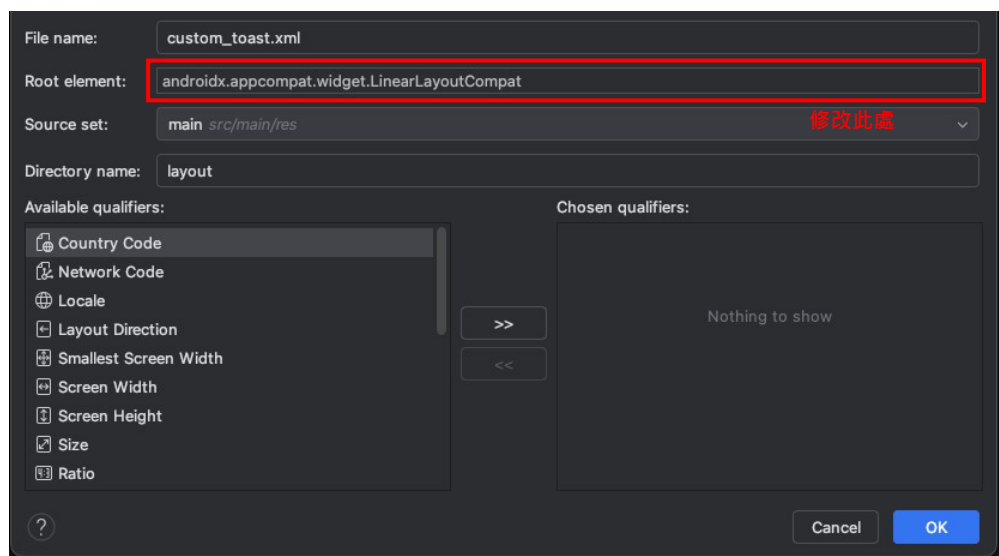
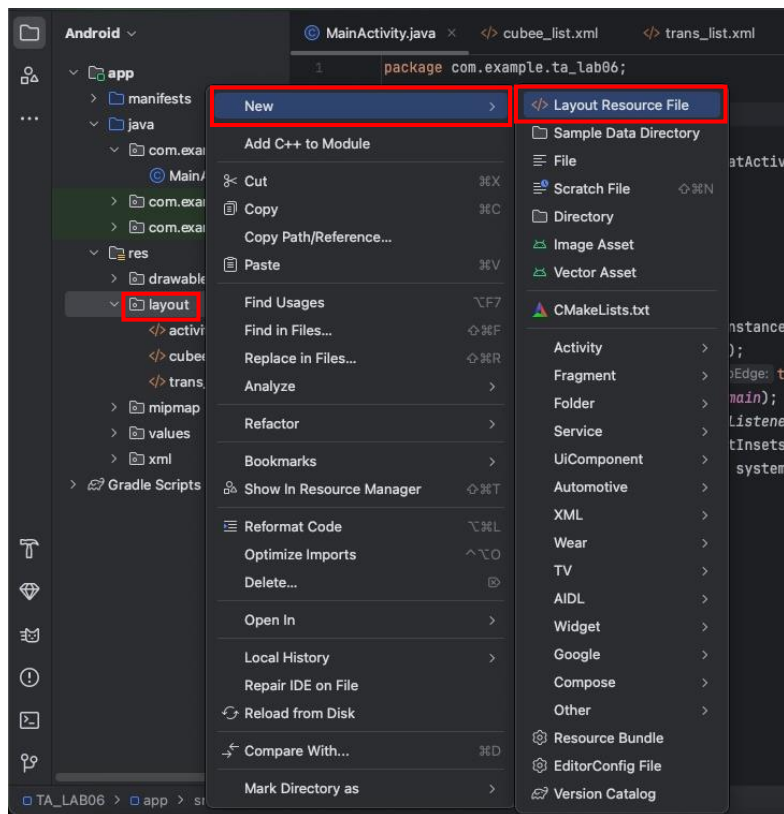
對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

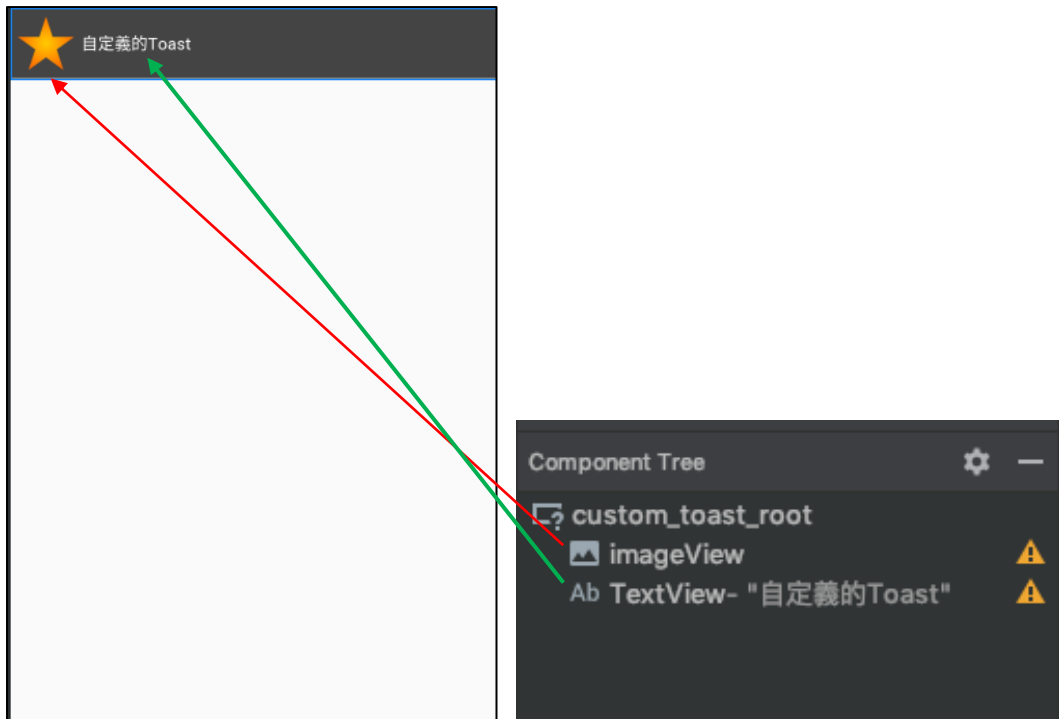
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="16dp"
        android:text="顯示Dialog"
        android:visibility="visible"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

## 2) 建立 custom\_toast.xml



## 3) 繪製 custom\_toast.xml，來顯示客製化的 toast 樣式



對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/custom_toast_root"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#444444"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
        app:srcCompat="@drawable/star" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_vertical"
        android:text="自定義Toast"
        android:textColor="@android:color/background_light" />
</androidx.appcompat.widget.LinearLayoutCompat>
```

### Step3

編寫 MainActivity 的程式

- 1) 編寫含確定、拒絕與取消按鈕的對話框。其中 NeutralButton 執行 Toast.makeText()方法,NegativeButton 執行副程式 showToast(),PositiveButton 執行副程式 showListDialog()。

The image shows a code editor with Java code for MainActivity and a corresponding dialog box UI mockup. Red arrows connect the code logic to the UI elements.

**Code Snippet:**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);    初始化 Activity
    setContentView(R.layout.activity_main);    連接 main1.xml 畫面

    Button btn = findViewById(R.id.button);    連接 Button 元件
    btn.setOnClickListener(new View.OnClickListener() {    Button 點擊事件
        @Override
        public void onClick(View view) {

            final AlertDialog.Builder dialog = new AlertDialog.Builder(context: MainActivity.this);    建立 AlertDialog 物件
            dialog.setTitle("請選擇功能");
            dialog.setMessage("請根據下方按鈕選擇要顯示的物件");

            dialog.setNeutralButton(text: "取消", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    Toast.makeText(context: MainActivity.this, text: "dialog關閉", Toast.LENGTH_SHORT).show();
                }
            });    使用 makeText 顯示訊息

            dialog.setNegativeButton(text: "自定義Toast", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    showToast();
                }
            });    執行副程式來顯示客製化 Toast

            dialog.setPositiveButton(text: "顯示list", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    showListDialog();
                }
            });    執行副程式來顯示含列表的對話框
            dialog.show();
        }
    });
}
```

**Dialog Box UI Mockup:**

- Title: 請選擇功能
- Message: 請根據下方按鈕選擇要顯示的物件
- Buttons: 取消, 自定義Toast, 顯示list

Red arrows indicate the flow: from the dialog UI to the code, and from specific code blocks to their corresponding UI buttons.

## 2) showToast()中編寫顯示客製化 Toast

```
private void showToast(){
    Toast toast = new Toast(context: MainActivity.this);
    toast.setGravity(Gravity.TOP, xOffset: 0, yOffset: 50);
    toast.setDuration	Toast.LENGTH_SHORT);
    LayoutInflater inflater = getLayoutInflater();
    View layout = inflater.inflate(R.layout.custom_toast
        , (ViewGroup) findViewById(R.id.custom_toast_root));
    toast.setView(layout);
    toast.show();
}
```

Step1: 初始化  
Step2: Toast 在畫面中顯示位置  
Step3: Toast 在畫面中顯示的持續時間  
Step4: 取得自定義的畫面  
Step5: 放入自定義的畫面(custom\_toast.xml)  
Step6: 顯示畫面




## 3) showListDialog ()中編寫含列表的對話框

```
private void showListDialog(){
    final String[] list = {"message1", "message2", "message3", "message4", "message5"};
    AlertDialog.Builder dialog_list = new AlertDialog.Builder(context: MainActivity.this);
    dialog_list.setTitle("使用LIST呈現");
    dialog_list.setItems(list, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            Toast.makeText(context: MainActivity.this
                , text: "你選得是"+list[i], Toast.LENGTH_SHORT).show();
        }
    });
    dialog_list.show();
}
```

建立要顯示在的列表上的字串  
建立 AlertDialog 物件  
依照被點擊的項目用 Toast 顯示字串

使用List呈現

- message1
- message2
- message3
- message4
- message5



您選擇的是message1