Giới thiệu ứng dụng

Bài thực hành này, chúng ta sẽ viết ứng dụng **Droid Async** có giao diện như hình bên dưới.

Một số chức năng:

- Nhấn vào nút **START**: hiển thị các số từ 1 đến N *(N > 10.000)*, thanh Progress chạy từ đầu đến cuối (MAX = N)
- Nhấn vào nút **SHOW TOAST**: hiển thị Toast message **"Hello world!!!"**

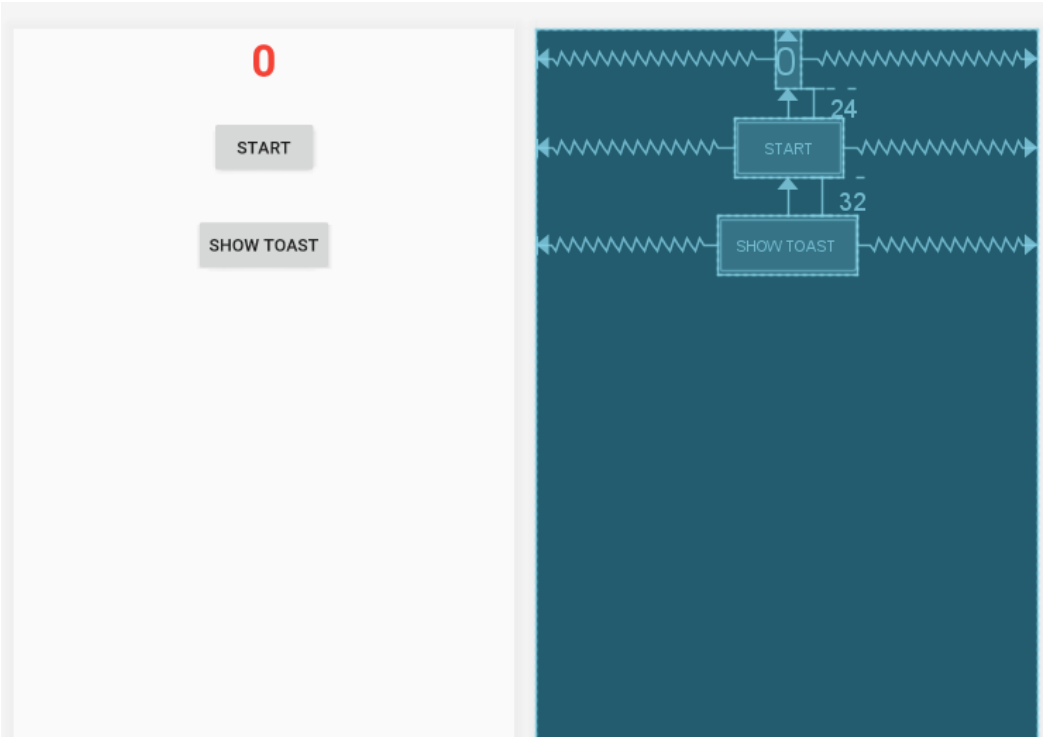
Task 1: Tạo project và Layout

1.1. Tạo project

- Name: Droid Async
- Template: Empty Activity
- Minimum API Level: 24 (Android 7)

1.2. Xây dựng Layout

- Sử dụng Constrain Layout theo gợi ý như hình bên dưới



- Bảng mô tả một số thuộc tính của các View:

View	Thuộc tính	Giá trị
TextView	***ID***	text_number
	textSize	36sp
	textStyle	B (Bold)
Button (START)	***ID***	button_start
	text	@string/button_start
	android:onClick	startHandle
Button (SHOW TOAST)	***ID***	button_show_toast
	text	@string/button_show_toast

	android:onClick	showToast

- Bảng mô tả các string resource:

Key name	Giá trị
button_start	Start
button_show_toast	Show Toast

1.3. Cài đặt các phương thức xử lý sự kiện Click vào các Button

- Button Start:

```
public void startHandle(View view) {
    int n = 1000000000; // Một tỉ
    for(int i=1; i<n+1; i++) {
        Log.d( tag: "TEST_LOG", Integer.toString(i));
    }
    Log.d( tag: "TEST_LOG",  msg: "Done...");
}
```

- Button ShowToast:

```
public void showToast(View view) {
    Toast.makeText(this,"Hello World!!!", Toast.LENGTH_SHORT).show();
}
```

1.4. Chạy ứng dụng

- Click vào buton SHOW TOAST
- Click vào buton START
- Xem LogCat (TEST_LOG)
- Chờ đợi kết quả thực thi của ứng dụng.

Task 2: Cài đặt AsyncTask

2.1. Tạo lớp MyAsyncTask kế thừa từ lớp AsyncTask

- Tạo Lớp: click chuột phải vào package trong **app > java > [package-name]**, chọn **New → Java Class**
  - ***Name**: MyAsyncTask*
  - ***Superclass**: android.os.AsyncTask*
- Khai báo các kiểu dữ liệu của các tham số truyền vào các phương thức:
  - **doInBackground**: Integer
  - **onProgressUpdate**: Void
  - **onPostExecute**: String

```
public class MyAsyncTask extends AsyncTask<Integer, Void, String> { }
```

- Bấm tổ hợp phím **ALT + ENTER → Implement methods →** Chọn phương thức **doInBackground → OK**
- Khai báo biến tham chiếu đến TextView **text_number** trong **MainActivity** ở đầu class:

```
WeakReference<TextView> mTextView;
```

- Cài đặt phương thức khởi tạo:

```
// Phương thức khởi tạo
public MyAsyncTask(TextView txt) {
    this.mTextView = new WeakReference<>(txt);
}
```

2.2. Cài đặt các phương thức trong lớp MyAsyncTask

- doInBackground():

```
@Override
protected String doInBackground(Integer... integers) {
    int n = integers[0];
    for(int i=1; i<n+1; i++) {
        Log.d( tag: "TEST_LOG", Integer.toString(i));
    }
    return "Done...";
}
```

- onPostExecute():

```
@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    // Hiển thị kết quả lên TextView
    this.mTextView.get().setText(s);
}
```

2.3. Tạo và chạy Background Thread với lớp MyAsyncTask

- Cài đặt lại phương thức **startHandle** trong **MainActivity**:

```
public void startHandle(View view) {
    int n = 1000000000; // Một tỉ
    TextView textView = findViewById(R.id.text_number);

    // Chạy background thread
    new MyAsyncTask(textView).execute(n);
}
```

- Chạy ứng dụng:
  - Click vào nút SHOW TOAST
  - Click vào nút START
  - Click lại vào nút SHOW TOAST và xem kết quả **(có thực thi được không?)**

2.4. Hiển thị giá trị của biến i trong mỗi lần lặp lên TextView

- Thay đổi kiểu dữ liệu của tham số truyền vào phương thức **onProgressUpdate()** từ **Void** thành **Integer**

```
public class MyAsyncTask extends AsyncTask<Integer, Integer, String>
```

- Gọi phương thức **publishProgress()** trong mỗi lần lặp trong phương thức **doInBackground()**

```
@Override
protected String doInBackground(Integer... integers) {
    int n = integers[0];

    for(int i=0; i<n; i++) {
        publishProgress(i);
        Log.d( tag: "TEST_LOG", Integer.toString(i));
    }

    return "Done...";
}
```

- Cài đặt phương thức **onProgressUpdate()**

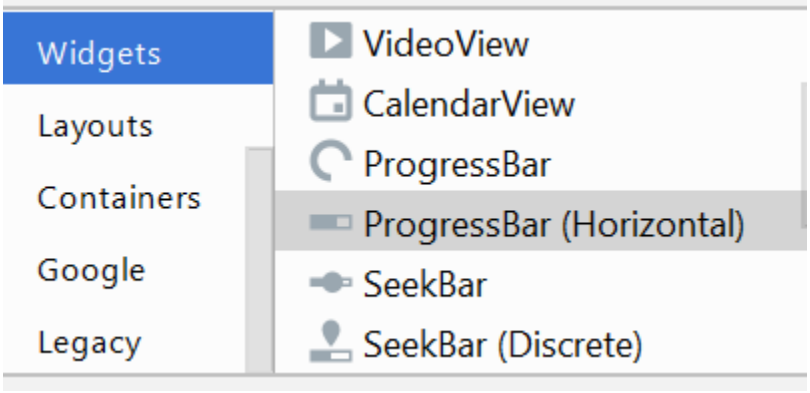
```
@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate(values);
    this.mTextView.get().setText(Integer.toString(values[0]));
}
```

- Chạy ứng dụng: click vào nút START và xem kết quả

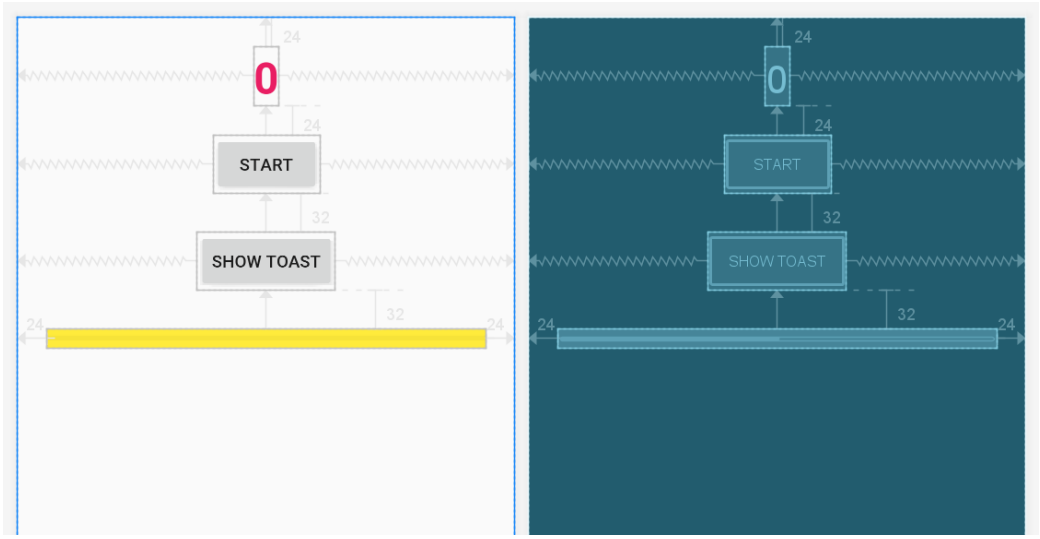
Task 3: Thêm ProgressBar vào ứng dụng

3.1. Thêm ProgressBar (Horizontal) vào Layout

- Kéo ProgressBar (Horizontal) từ Palette vào Layout **main_activity.xml**



- Tạo các constrain cho ProgressBar theo mô tả như hình bên dưới



- Thay đổi các thuộc tính của ProgressBar như sau:
  - **ID**: progress_bar
  - **background**: #FFEB3B
  - **progressTint**: #E91E63

3.2. Cập nhật ProgressBar trong mỗi lần thực hiện vòng lặp

- Khai báo biến tham chiếu đến ProgressBar **progress_bar** trong **MainActivity** ở đầu class **MyAsyncTask**:

```
WeakReference<TextView> mTextView;
WeakReference<ProgressBar> mProBar;
```

- Cập nhật lại phương thức khởi tạo lớp **MyAsyncTask** như sau:

```
// Phương thức khởi tạo
public MyAsyncTask(TextView txt, ProgressBar bar) {
    this.mTextView = new WeakReference<>(txt);
    this.mProBar = new WeakReference<>(bar);
}
```

- Cập nhật giá trị của ProgressBar trong phương thức **onProgressUpdate()**:

```
@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate(values);
    this.mTextView.get().setText(Integer.toString(values[0]));
    this.mProBar.get().setProgress(values[0]);
}
```

- Cập nhật phương thức **startHandle()** trong **MainActivity** *(thêm ProgressBar vào phương thức khởi tạo đối tượng **MyAsyncTask**)*:

```
public void startHandle(View view) {
    int n = 100000;
    TextView textView = findViewById(R.id.text_number);
    ProgressBar pBar = findViewById(R.id.progress_bar);
    pBar.setMax(n);
    // Chạy background thread
    new MyAsyncTask(textView, pBar).execute(n);
}
```

- Chạy ứng dụng: click vào nút START và xem kết quả

Coding Challenge

Thêm TextView bên dưới ProgressBar để hiển thị tỉ lên phần trăm thực hiện tác vụ trong Background Thread