Bài 09:
CONTENT PROVIDER VÀ ĐA TIẾN TRÌNH TRONG ANDROID

GVGD: ThS. Đặng Thế Hân

Biên soạn: ThS. Giang Hào Côn

Mục tiêu

Cung cấp cho sinh viên kiến thức cơ bản về đa tiến trình trong ứng dụng Android và cách sử dụng Content Provider trong ngôn ngữ lập trình ứng dụng trên Android.

Nội dung

1) Sử dụng Content Proivider trong ứng dụng Android
2) Đa tiến trình trong Android

9.1/ Content Provider trong Android

- Cơ sở dữ liệu **SQLite** trong APP là CSDL riêng của APP đó và các APP khác trong cùng một thiết bị điện thoại không thể truy cập được.
- **Content Provider** cung cấp cơ chế truy cập dữ liệu giữa các ứng dụng khác nhau trên cùng một thiết bị Android.
- Sử dụng Content Provider giống hình thức giao tiếp **client/server** trong đó ứng dụng truy cập dữ liệu đóng vai trò là client và Content Provider đóng vai trò là server.

9.1.1/Truy cập Content Provider

Để truy cập dữ liệu của một ứng dụng khác thì từ ứng dụng của mình chúng chúng ta sử dụng đối tượng ContentResolver. Đối tượng này sẽ cung cấp các chức năng như insert(), update(), delete(), query().

Ví dụ: để lấy nội dung được cung cấp bởi ContentProvider của một ứng dụng khác thì ngay tại ứng dụng của chúng ta sẽ gọi ContentResolver.Query().

```
// Queries the user dictionary and returns results
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,   // The content URI of the words table
    mProjection,                        // The columns to return for each row
    mSelectionClause                    // Selection criteria
    mSelectionArgs,                     // Selection criteria
    mSortOrder);                        // The sort order for the returned rows
```

9.1.2/ Ý nghĩa của phương thức Query

query(uri, projection, selection, selectionArgs, sortOrder)

Tham số query()	Từ khóa	Mô tả
Uri	FROM table_name	Uri sẽ ánh xạ tới bảng có tên là table_name trong ContentProvider
Projection	Col,col,col...	Là một mảng bao gồm các cột cần truy xuất
Selection	WHERE col = value	Điều kiện cột cần truy xuất
SelectionArgs		
sortOrder	ORDER BY col, col,...	Sắp xếp dữ liệu trong Cursor

Một thiết bị Android có thể chứa nhiều Content Providers, do đó, hệ thống phải cung cấp giải pháp để phân biệt các Content Providers. **URI** là một giải pháp cho phép xác định dữ liệu cụ thể với một Content Provider cụ thể.

9.1.2/ Ý nghĩa của phương thức Query

- Để truy vấn ContentProvider chúng ta cần phải chỉ định Uri theo định dạng sau:
  - <prefix>://<authority>/<data_type>/<id>

Part	Mô Tả
Prefix	Luôn luôn là content://
Authority	Chỉ định tên của ContentProvider thông thường sẽ là tên package của ứng dụng muốn truy cập dữ liệu, ví dụ: content://com.gianghaocon.myapp

9.1.2/ Ý nghĩa của phương thức Query

<prefix>://<authority>/<data_type>/<id>

Part	Mô Tả
Data_type	Chỉ định loại dữ liệu sẽ được truy xuất trong ContentProvider, thông thường sẽ là tên bảng. Ví dụ tham chiếu đến bảng student trong ContentProvider: ***com.gianghaocon.myapp/student***
id	Chỉ định dữ liệu yêu cầu truy xuất. ví dụ tham chiếu đến một hàng có giá trị ID là 3 trong bảng student: ***com.gianghaocon.myapp/student/3***

9.1.3/ Các phương thức của ContentProvider

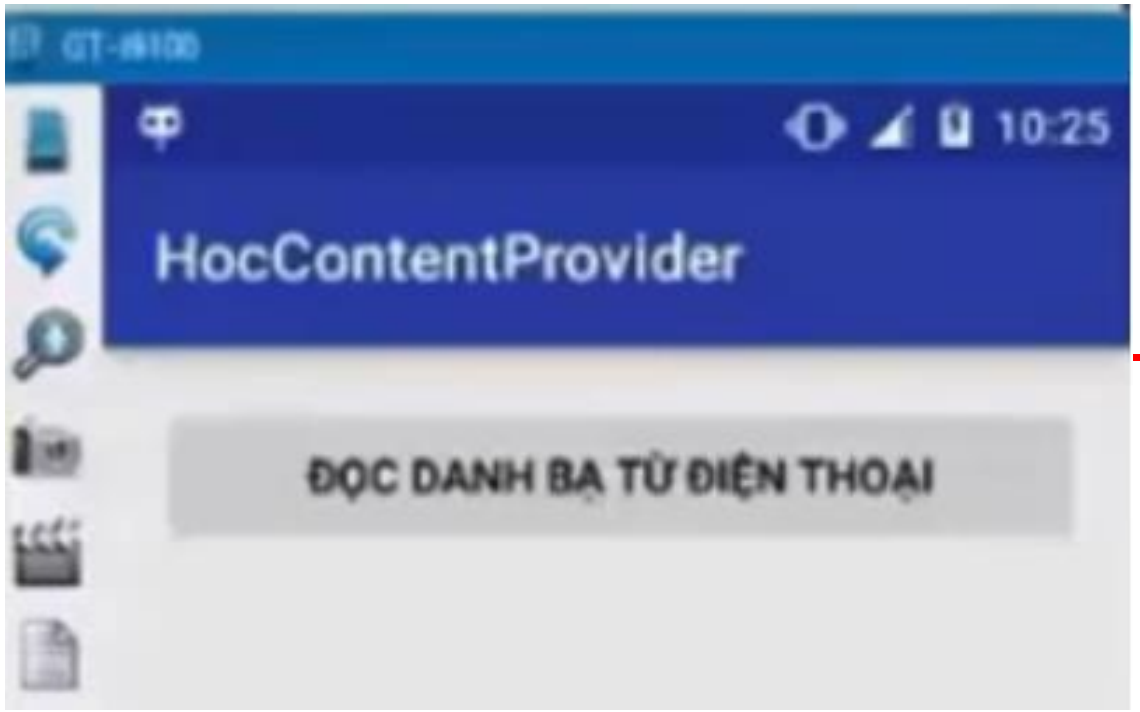
- Có những **URI** có thể mở rộng đến nhiều cấp và lúc này chúng ta có thể dùng lớp **UriMatcher,** và thông qua nó chúng ta sẽ phân tích Uri người dùng truyền vào và sẽ biết được người dùng đang thực hiện hành động nào.
- ContentProvider sẽ override lại 6 phương thức và các phương thức

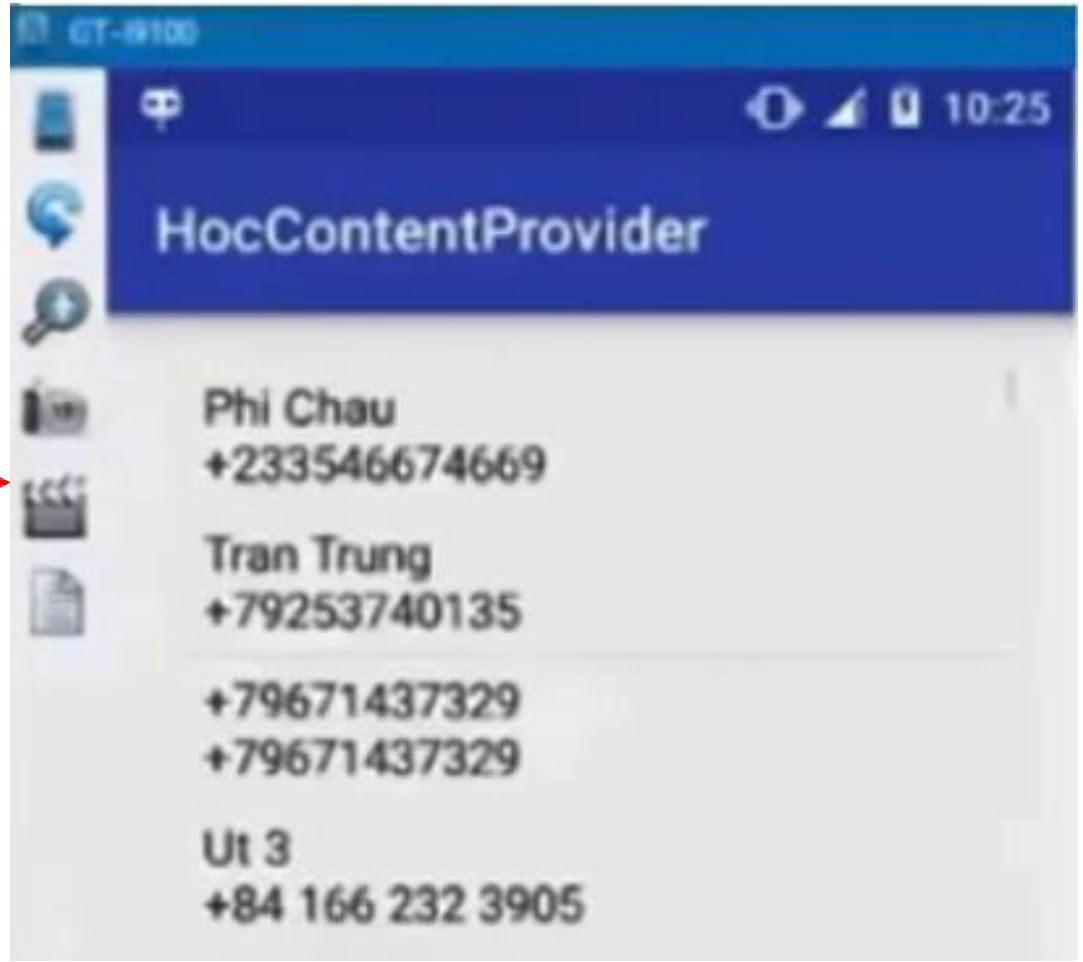
Phương thức	Chức năng
Query()	Truy xuất dữ liệu từ ContentProvider. Sử dụng các tham số để truy vấn dữ liệu và trả về Cursor.

9.1.3/ Các phương thức của ContentProvider

Phương thức	Chức năng
onCreate()	Khởi tạo ContentProvider. ContentProvider sẽ không được khởi tạo cho đến khi ContentResolver truy cập.
insert()	Phương thức này được gọi khi một hàng mới được thêm vào cơ sở dữ liệu Provider.
update()	Phương thức này được gọi khi cập nhật vào cơ sở dữ liệu Provider.
delete()	Phương thức này được gọi khi xoá các hàng trong cơ sở dữ liệu Provider.

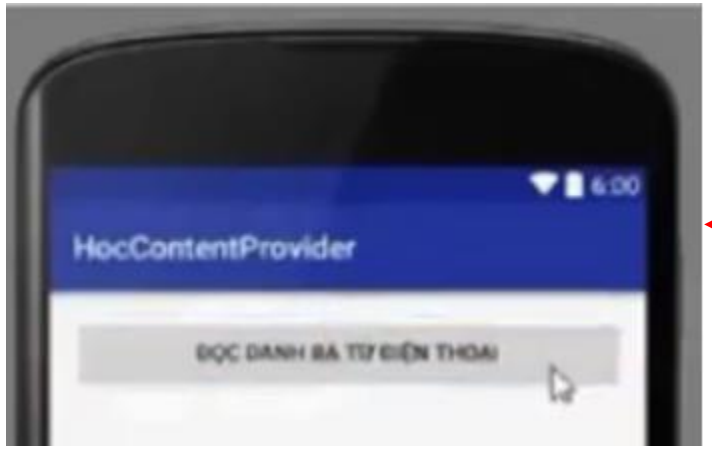
Ví dụ sử dụng ContentProvider





Ví dụ sử dụng ContentProvider

Bước 01: tạo 1 project mới và thiết kế giao diện như sau:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...
    ...
    android:orientation="vertical"
    tools:context="com.gianghaocon.sudungcontentprovider">

   <Button
        android:Layout_width="match_parent"
        android:Layout_height="wrap_content"
        android:text="Đọc danh bạ từ điện thoại"
        android:id="@+id/btnDocDanhBa"
        android:onClick="xulyMoManhinhDocDanhBa"
    />

</LinearLayout>
```

Ví dụ sử dụng ContentProvider

Bước 02: tạo Activity mới tên là DanhBaActivity, đặt 1 Listview và đặt tên là lvDanhBa:


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...
    ...
    android:orientation="vertical"
    tools:context="com.gianghaocon.sudungcontentprovider">

    <ListView
        android:Layout_width="match_parent"
        android:Layout_height="wrap_content"
        ...
        android:id="@+id/lvDanhBa"
    />
</LinearLayout>
```

Ví dụ sử dụng ContentProvider

Bước 03: mở file MainActivity.java và code như sau:

```
package com.gianghaocon.sudungcontentprovider;

import ...

public class MainActivity extends AppCompatActivity {

        @Override
        protected void onCreate(Bundle saveInstanceState) {
            super.onCreate(saveInstanceState)
            setContentView(R.Layout.activity_main)
        }

        public void xulyMoManHinhDocDanhBa(View view) {
           Intent intent = new Intent(MainActivity.this, DanhbaActivity.class)
           startActivity(intent)
        }
}
```

Ví dụ sử dụng ContentProvider

Bước 04: Tạo một Java Class tên Contact và cài đặt getter, setter cho 2 thuộc tính trên và 2 contructor có đối số và không đối số

```
package com.gianghaocon.sudungcontentprovider;

public class Contact implements Serialization {
     private String name;
     private String phone;

    public Contact(){
    }

    public Contact(String name, String phone) {
       this.name = name;
       this.phone= phone;
    }
    ...
}
```

Ví dụ sử dụng ContentProvider

Bước 05: Mở file DanhBaActivity.java

```
package com.gianghaocon.sudungcontentprovider;

import ...

public class DanhBaActivity extends AppCompactACtivity {

    ListView lvDanhBa;
    ArrayList<Contact>dsDanhBa;
    ArrayAdapter<Contact>adapterDanhBa;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_danhba);

        anhxa();
        lietkeTatDanhBaTuThietBi();
    }

}
```

lietketatDanhBaTuThietBi()

```
private void anhxa() {
    lvDanhBa = (ListView) findViewById(R.id.lvDanhBa);
    dsDanhBa = new ArrayList<>();
    adapterDanhBa = new ArrayAdapter<Contact>(
        DanhBaActivity.this, android.R.layout.simple_list_item_1,
        dsDanhBa);
    lvDanhBa.setAdapter(adapterDanhBa);
}
```

Ví dụ sử dụng ContentProvider

```
private void lietkeTatdanhBaTuThietBi(){
  Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
  Cursor cursor = getContentResolver().query(uri,null,null,null,null);
  dsDanhBa.clear();
  while (cursor.moveToNext())
  {
     String tenCotName = ContactsContract.Contacts.DISPLAY_NAME;
     String tencotPhone = ContactsContract.Common.DataKinds.Phone.NUMBER;

     int vtTenCotName = cursor.getColumnIndex(tenCotName);
     int vtTenCotPhone = cursor.getColumnIndex(tenCotPhone);

     String name = cursor.getString(vtTenCotName);
     String phone = cursor.getString(vtTenCotPhone);

     Contact contact = new Contact(name, phone);
     dsDanhBa.add(contact);
  }
  apdapterDanhBa.notifyDataSetChanged();
}
```

Ví dụ sử dụng ContentProvider

Bước 06: Mở file AndroidManifest.xml xin cấp quyền truy xuất danh bạ

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="vn.edu.topica.hoccontentprovider">
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HocContentProvider"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DanhBaActivity"></activity>
    </application>

</manifest>
```

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

9.2/ Đa tiến trình trong Android

Giới thiệu về Thread trong Android

- **Thread** là một **tiến trình đơn vị** xử lý của máy tính có thể thực hiện một công việc riêng biệt.
- **Mutil-Thread** là khái niệm cho nhiều tiến trình chạy đồng thời, một ứng dụng Java ngoài luồng chính ra có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh hơn và hiệu quả hơn.
- Trong Android sẽ chia làm 2 loại Thread:
  - **Worker Thread** : Là một dạng khác của Thread nhưng không thể tương tác với giao diện người dùng. (**không cho cập nhật giao diện**)

9.2/Đa tiến trình trong Android

Giới thiệu về Thread trong Android

- Trong Android sẽ chia làm 2 loại Thread:
  - **Worker Thread**: Là một dạng khác của Thread nhưng không thể tương tác với giao diện người dùng. (không cho cập nhật giao diện).
  - **MainThread**: Là một dạng Thread nhưng có thể tương tác với giao diện người dùng. (cho phép cập nhập giao diện người dùng):
    - Handler
    - AsynTask
    - BroadCast Receiver
    - Service

9.2/ Đa tiến trình trong Android

Giới thiệu về Thread trong Android

- **Ưu điểm của đa luồng** :
  - Mỗi luồng có thể dùng chung và chia sẻ nguồn tài nguyên trong quá trình chạy, nhưng có thể thực hiện một cách độc lập.
  - Phân tách các luồng: luồng chính thì chạy trên giao diện người dùng, các luồng phụ làm nhiệm vụ riêng gửi đến luồng chính.
- **Nhược điểm của đa luồng** :
  - Càng nhiều luồng thì xử lý càng phức tạp.
  - Nhiều luồng chạy thì sẽ khó quản lý, dễ gây ra lỗi hoặc luồng chết.

9.2/ Đa tiến trình trong Android

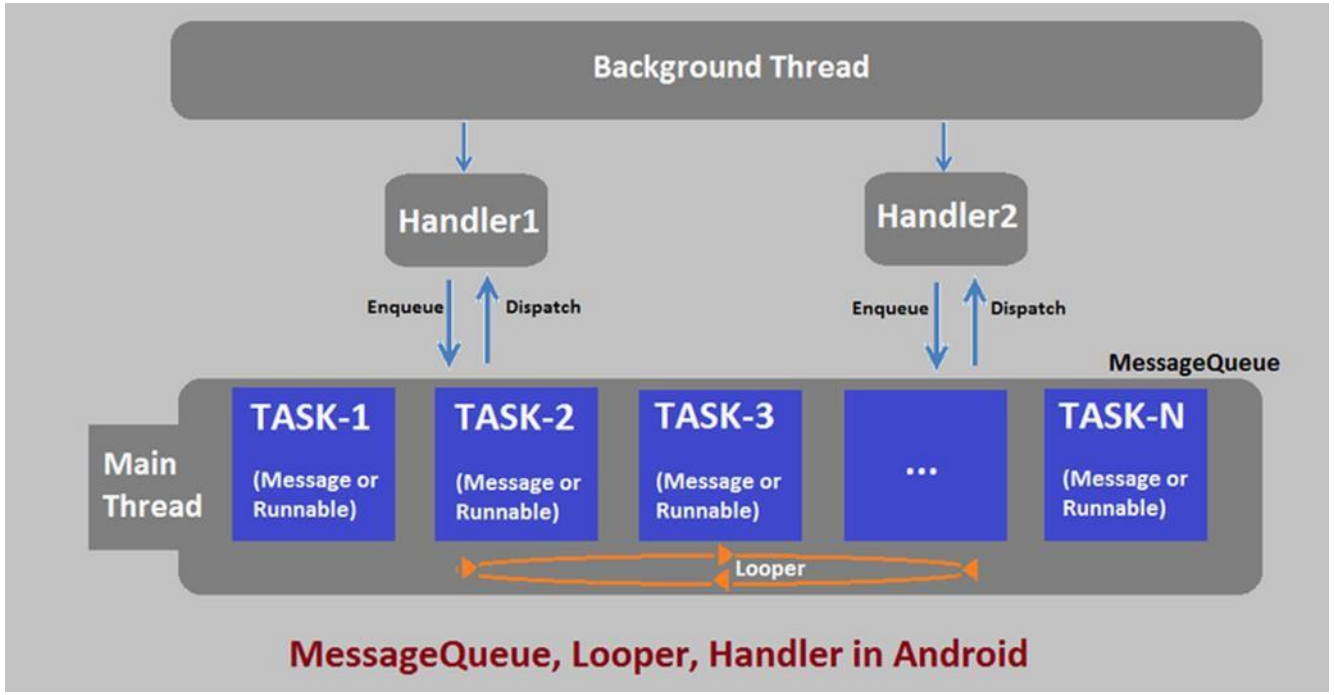
HANDLER là gì ?

- Handler là một đối tượng Android cung cấp dùng để liên kết, trao đổi giữa các Thread với nhau, là trao đổi giữa Thread sinh ra Handler và các Thread khác. Thường là **Main Thread** (UI Thread) với các **Worker Thread** (Background Thread).
- Handler có nhiệm vụ gửi và thực thi các Message hoặc Runnable tới Message Queue của Thread sinh ra nó (Handler). Handler luôn được gắn kết với một Thread (Thread sinh ra nó) cũng với Message Queue (của Thread đó). Các Message và Runnable sẽ được thực thi khi đi ra khỏi Message Queue. Có 2 nhiệm vụ mà Handler thường làm đó là:

Đa tiến trình trong Android

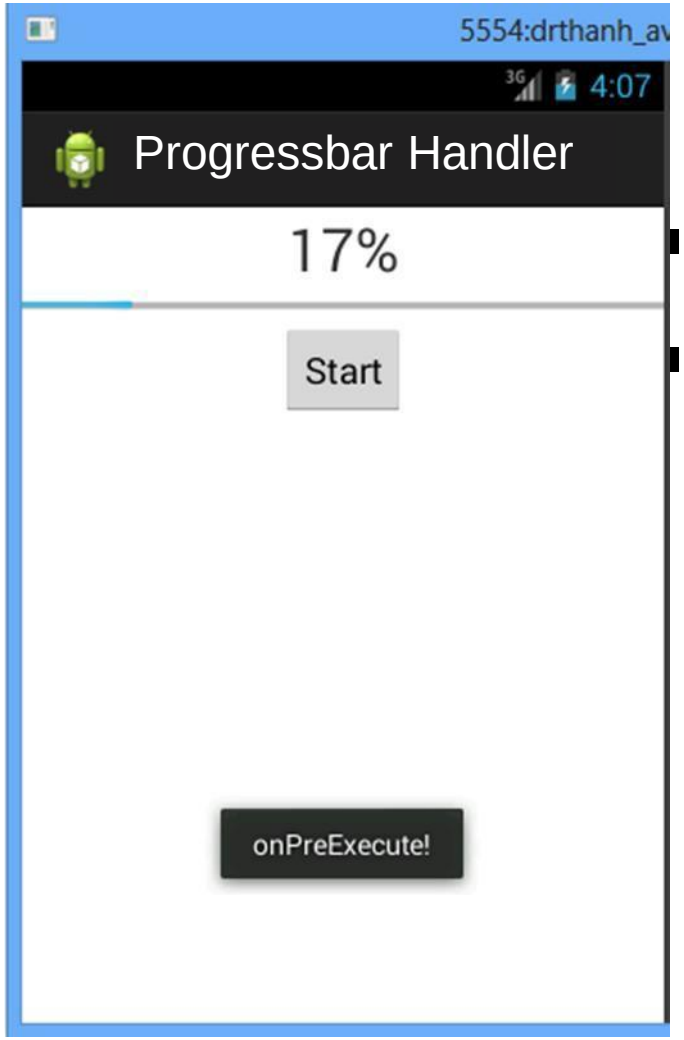
HANDLER là gì ?

1. Lên lịch thực thi các Message và Runnable ở các thời điểm trong tương lai.
2. Sắp xếp một hành động được thực hiện trong một Thread khác.



Ví dụ: nếu bạn tạo new Handler instance trong phương thức onCreate() của activity, nó có thể được dùng để đẩy dữ liệu lên main thread. Dữ liệu có thể được đẩy qua lớp Handler có thể là một instance của lớp Message hoặc Runnable. Handler rất hữu ích nếu bạn muốn đẩy nhiều lần dữ liệu vào mainThread.

Ví dụ sử dụng Handler trong Android



- Bước 01: Tạo Project mới.
- Bước 02: Thiết kế giao diện. Activity_Main.xml
- Bước 03: Mở file MainActivity.java

9.2/ Đa tiến trình trong Android

AsyncTask

- **AsyncTask** là phương tiện khác để xử lý công việc sử dụng background thread và giao tiếp với UI thread mà không dùng Thread hay Handler.
- Trong **AsyncTask<Params, Progress, Result>** có 3 đối số Generic Type:

Params	Là giá trị ((biến) được truyền vào khi gọi thực thi tiến trình và nó sẽ được truyền vào **doInBackground**
Progress	Là giá trị (biến) dùng để update giao diện diện lúc tiến trình thực thi, biến này sẽ được truyền vào hàm **onProgressUpdate**
Result	Là biến dùng để lưu trữ kết quả trả về sau khi tiến trình thực hiện xong

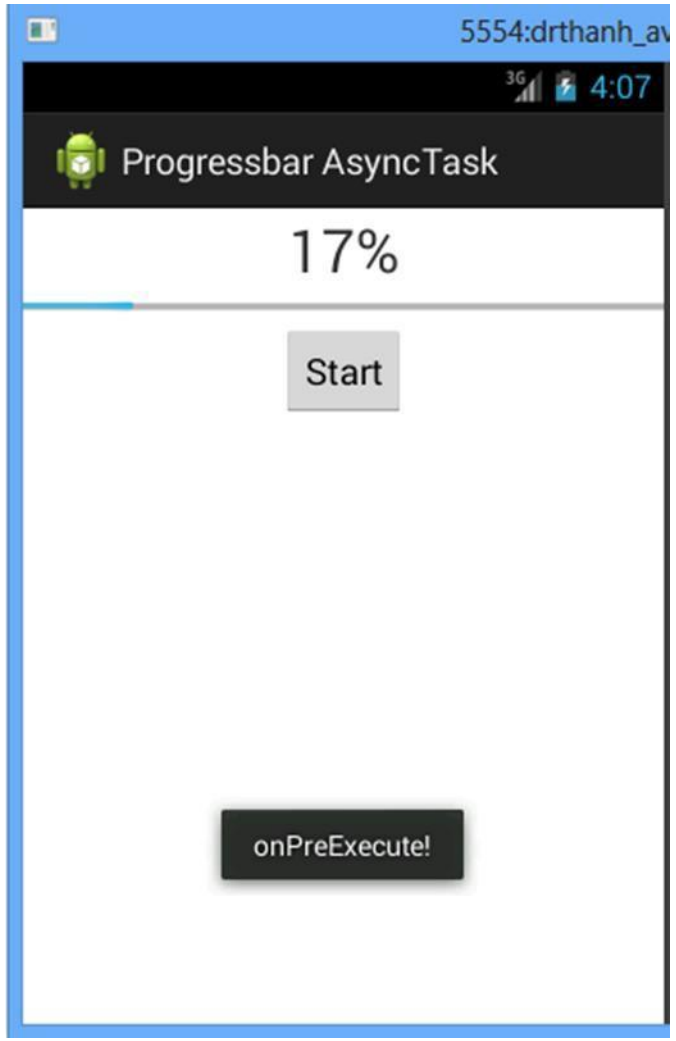
9.2/ Đa tiến trình trong Android

AsyncTask

Thông thường trong 1 AsyncTask sẽ chứa 4 hàm, đó là :

onPreExecute()	Tự động được gọi đầu tiên khi tiến trình được kích hoạt.
doInBackground()	Được thực thi trong quá trình tiến trình chạy nền, thông qua hàm này để ta gọi hàm onProgressUpdate để cập nhật giao diện (gọi lệnh publishProgress). Ta không thể cập nhật giao diện trong hàm doInBackground().
onProgressUpdate()	Dùng để cập nhật giao diện lúc runtime.
onPostExecute()	Sau khi tiến trình kết thúc thì hàm này sẽ tự động xảy ra. Ta có thể lấy được kết quả trả về sau khi thực hiện tiến trình kết thúc ở đây.

Ví dụ sử dụng AsyncTask trong Android



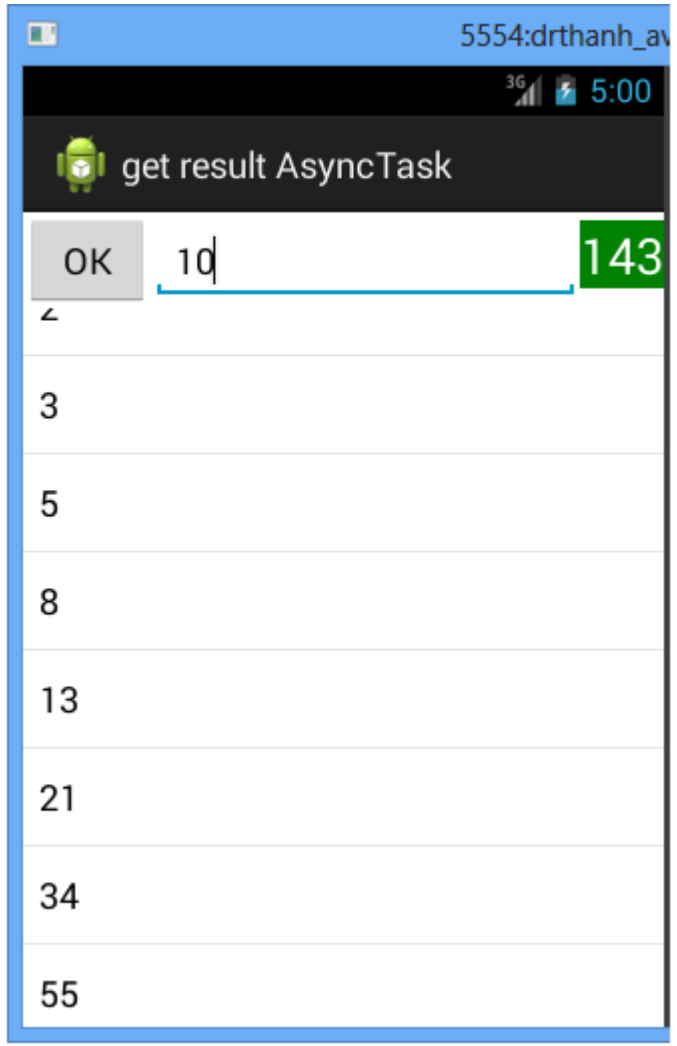
Bước 01: Tạo Project mới.

Bước 02: Thiết kế giao diện. Activity_Main.xml

Bước 03: Tạo class MyAsyncTask kế thừa AsyncTask

Bước 04: Xử lý code trong MainActivity.java

Ví dụ sử dụng AsyncTask trong Android



Bước 01: Tạo Project mới.

Bước 02: Thiết kế giao diện. Activity_Main.xml

Bước 03: Tạo class MyAsyncTask kế thừa AsyncTask

Bước 04: Xử lý code trong MainActivity.java

Câu hỏi thảo luận

1. Trình bày mục đích sử dụng Content Provider. Content Provider có những phương thức nào và trình bày ý nghĩa của từng phương thức đó.
2. Trình bày mục đích, ưu điểm và nhược điểm của Thread, Handler và AsyncTask trong Android.
3. Hãy cho biết sự khác nhau của Thread, Handler và AsyncTask ?