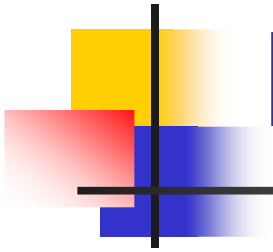
Trường Đại học Bách Khoa Hà Nội
Hanoi University of Science and Technology

Chapter 4. Graphical User Interfaces

Hanoi University of Science and Technology

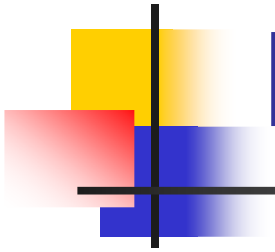
Nguyen Hong Quang
19/8/2012



Mục lục

- ProgressBar
- Tạo luồng Background
- Trao đổi dữ liệu giữa luồng Background và UI Thread
  - runOnUiThread
  - Handle
  - AsyncTask

ProgressBar



ProgressBar

- ProgressBar ngầm định của Activity
- Widget ProgressBar

ProgressBar ngầm định của Activity

- Tạo ProgressBar ngầm định trong hàm onCreate():

```
super.onCreate(savedInstanceState);
requestWindowFeature(Window.FEATURE_PROGRESS);
setContentView(R.layout.main);
```

- Hiển thị ProgressBar:

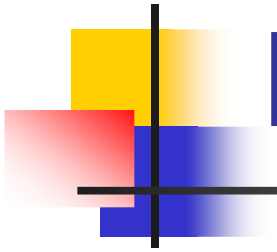
```
setProgressBarVisibility(true);
```

- Thiết lập giá trị cho ProgressBar:

```
setProgress(progress);
```

- Gía trị tối đa ngầm định: 10000

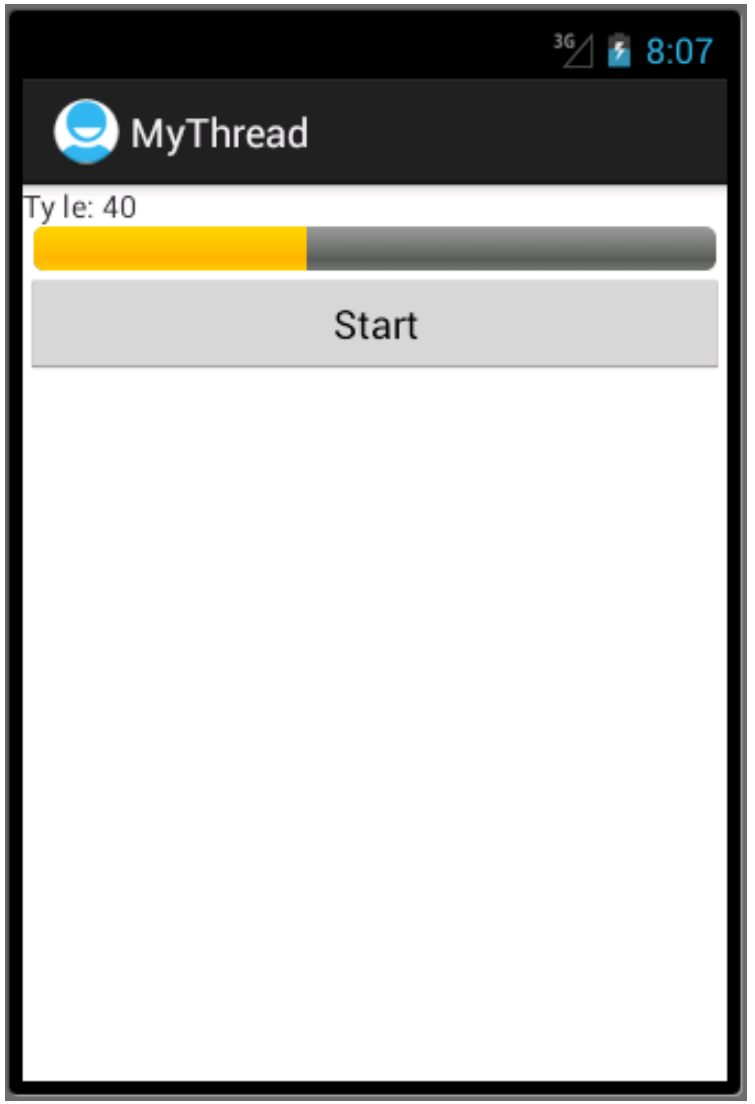
Các phương thức của ProgressBar



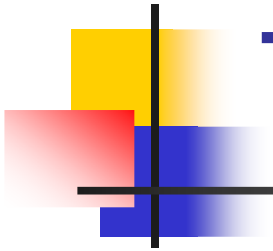
- Method setMax()
- Method getProgress()
- Method incrementProgressBy()

Widget ProgressBar

- Chèn vào file XML
- <ProgressBar android:id="*@+id/progress*"
  - style="*@android:style/Widget.ProgressBar.Horizontal*" />
- Lấy điều khiển và thiết lập giá trị
  - pb = (ProgressBar)findViewById(R.id.*progress);*
  - pb.setProgress(progress);
- Giá trị ngầm định: 100



Tạo luồng Background



Tạo luồng Background

- Khai báo lớp luồng với Runnable:

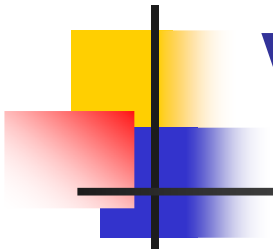
```
private Runnable longTask = new Runnable() {
    public void run() { ... }
}
```

- Tạo đối tượng luồng:

```
Thread thread;
thread = new Thread(longTask);
```

- Khởi động luồng:

```
thread.start();
```

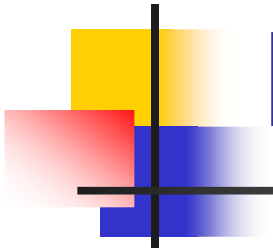


Ví dụ

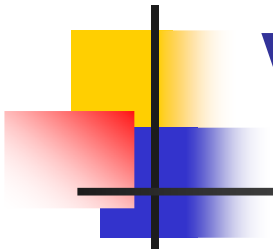
```
private Runnable longTask = new Runnable() {
    public void run() {
        for (int i=0;i<10;i++) {
            SystemClock.sleep(250);
            progress += 10;
            pb.setProgress(progress);
        }
    } // end of run
}; // end of new Runnable()
```

Trao đổi dữ liệu giữa luồng Background và UI Thread

Cập nhật giao diện từ luồng Background



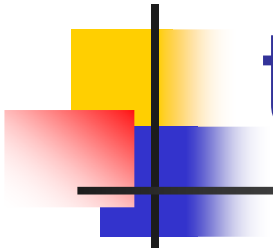
- Cho phép cập nhật ProgressBar (vì ProgressBar được nhìn thấy bởi tất cả các luồng trong ứng dụng)
- Tuy nhiên không cho phép cập nhật các thành phần giao diện khác



Ví dụ

```
private Runnable longTask = new Runnable() {
    public void run() {
        for (int i=0;i<10;i++) {
            SystemClock.sleep(250);
            progress += 10;
            pb.setProgress(progress); ➔ thực hiện được
            String str = "Ty le: " + progress;
            text.setText(str); ➔ không thực hiện được
        }
    }
};
```

Quản lý giao diện đồ họa trong Android

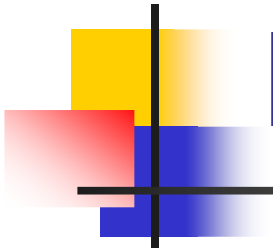


- When you call setText() on a TextView:
  - do not update the screen
  - just pop a message on a queue telling the operating system to update the screen
  - The operating system pops these messages off of this queue and does what the messages require.

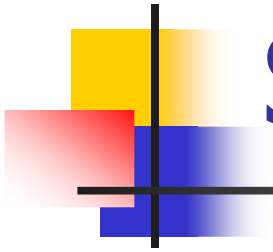
The queue is processed by one thread, variously called the "main application thread" and the "UI thread".

Trao đổi dữ liệu giữa luồng Background và UI Thread - Handle

Communication with the Handler

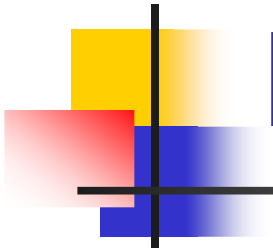


- Your background thread can communicate with the Handler, which will do all of its work on the activity's UI thread.
- Two options for communicating with the Handler:
  - messages and Runnable objects.



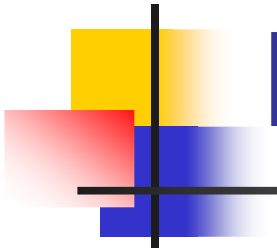
Send a Message to a Handler

- Invoke obtainMessage() to get the Message object
- Send the Message to the Handler:
  - sendMessage()
  - sendMessageAtFrontOfQueue()
  - sendMessageAtTime()
  - sendMessageDelayed()
  - sendEmptyMessage()



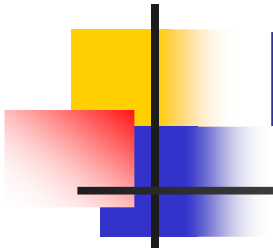
Process these messages

- Handler needs to implement handleMessage(), which will be called with each message that appears on the message queue.
- There, the handler can update the UI as needed.



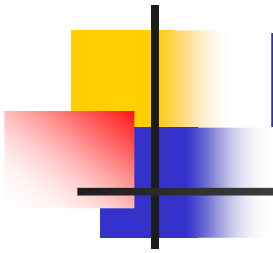
Example – Handler object

```
Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        //bar.incrementProgressBy(5);
        String str = "Ty le: " + progress;
        text.setText(str);
    }
};
```



Runnable object

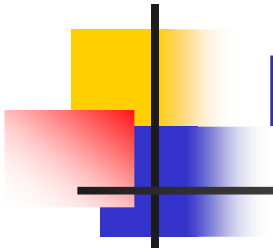
```
private Runnable longTask = new Runnable() {
    public void run() {
        for (int i=0;i<10;i++) {
            SystemClock.sleep(250);
            progress += 10;
            Message msg = handler.obtainMessage();
            handler.sendMessage(msg);
        }
    }
};
```



Khởi động luồng

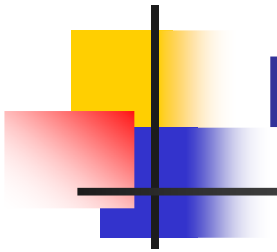
```
thread = new Thread(longTask);
thread.start();
```

Trao đổi dữ liệu giữa luồng Background và UI Thread - runOnUiThread



runOnUiThread

- This works similarly to the post() methods on Handler and View, in that it queues up a Runnable to run on the UI thread... if you are not on the UI thread right now.
- If you already are on the UI thread, it invokes the Runnable immediately.
- This gives you the best of both worlds: no delay if you are on the UI thread, yet safety in case you are not.

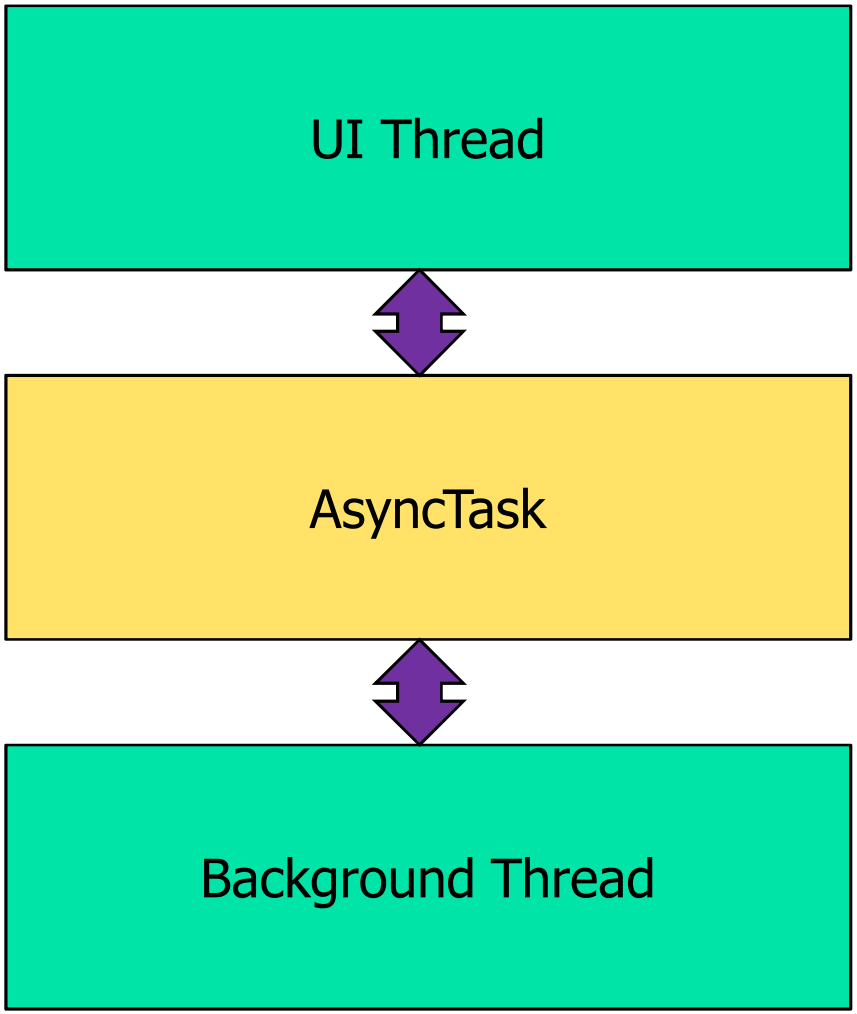


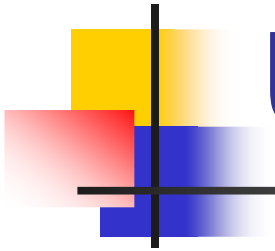
runOnUiThread

```
runOnUiThread(new Runnable() {
    public void run() {
        String str = "Ty le: " + progress;
        text.setText(str);
    }
});
```

Trao đổi dữ liệu giữa luồng Background và UI Thread - AsyncTask

Vai trò của AsyncTask

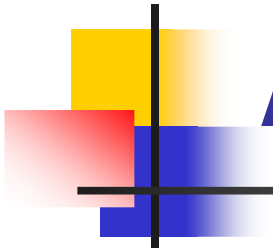




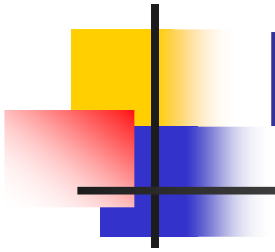
Use AsyncTask

- Create a subclass of AsyncTask
  - Inner class
  - Override one or more AsyncTask methods:
    - accomplish the background work
    - work associated with the task that needs to be done on the UI thread (e.g., update progress)
- Create an instance of the AsyncTask subclass and call execute()

Three data types using by AsyncTask



- The type of information that is needed to process the task (e.g., URLs to download)
- The type of information that is passed within the task to indicate progress
- The type of information that is passed when the task is completed to the post-task code

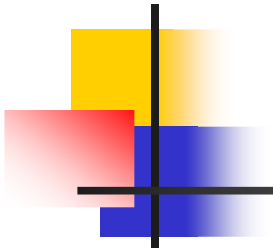


Methods of AsyncTask

- doInBackground()
- onPreExecute()
- onPostExecute()

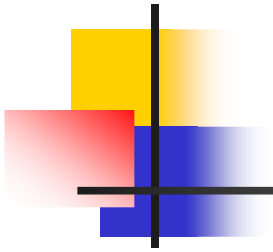
- onProgressUpdate()
- publishProgress()

Methods of AsyncTask doInBackground()

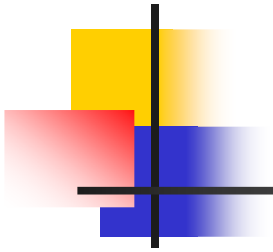


- This method will be called by AsyncTask on a background thread.
- The doInBackground() method will receive, as parameters, a varargs array of the first of the three data types listed above – the data needed to process the task.
- So, if your task's mission is to download a collection of URLs, doInBackground() will receive those URLs to process.

Methods of AsyncTask doInBackground()



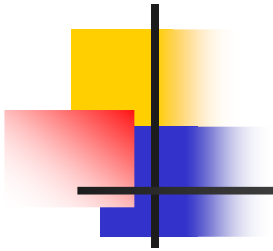
- The doInBackground() method must return a value of the third data type listed above – the result of the background work.



Methods of AsyncTask onPreExecute()

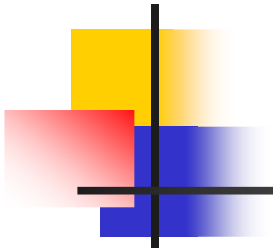
- This method is called, from the UI thread, before the background thread executes doInBackground().
- Here, you might initialize a ProgressBar or otherwise indicate that background work is commencing.

Methods of AsyncTask onPostExecute()



- This method is called, from the UI thread, after doInBackground() completes.
- It receives, as a parameter, the value returned by doInBackground() (e.g., success or failure flag).
- Here, you might dismiss the ProgressBar and make use of the work done in the background, such as updating the contents of a list.

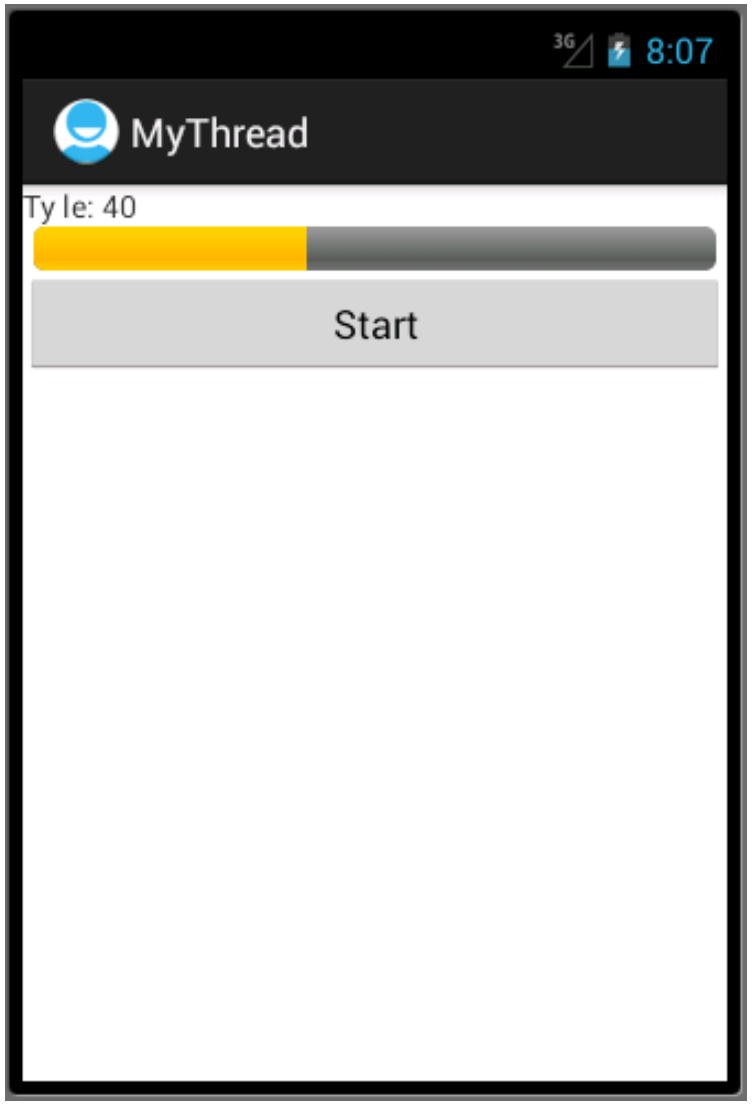
Methods of AsyncTask onProgressUpdate()



- If doInBackground() calls the task's publishProgress() method, the object(s) passed to that method are provided to onProgressUpdate(), but in the UI thread.
- That way, onProgressUpdate() can alert the user as to the progress that has been made on the background work, such as updating a ProgressBar or continuing an animation.
- The onProgressUpdate() method will receive a varargs of the second data type from the above list – the data published by doInBackground() via publishProgress().

Ví dụ với AsyncTask

```
button.setOnClickListener(new
    View.OnClickListener() {
            public void onClick(View v) {
                // Perform action on click
                new MyAsyncTask().execute();
            }
    }
);
```



Ví dụ với AsyncTask doInBackground method

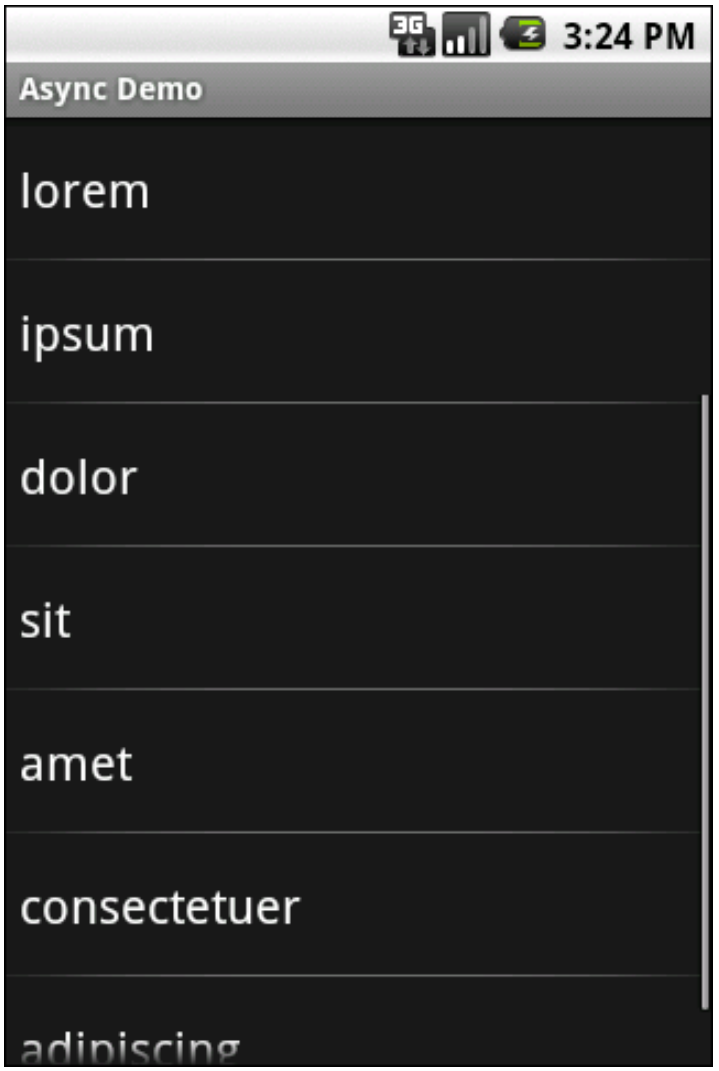
```
class MyAsyncTask extends AsyncTask<Void, Integer, Void> {
    int progress = 0;
      @Override
      protected Void doInBackground(Void... unused) {
            // TODO Auto-generated method stub
            for (int i=0;i<10;i++) {
                  progress += 10;
                  publishProgress(progress);
                  SystemClock.sleep(250);
            }
            return null;
      } // end of "doInBackground" method
```

Ví dụ với AsyncTask onProgressUpdate method

```
@Override
protected void onProgressUpdate(Integer... values) {
    // TODO Auto-generated method stub
    int progress = values[0].intValue();
    pb.setProgress(progress);
    String str = "Ty le : " + progress;
    text.setText(str);
    super.onProgressUpdate(values);
}
```

Bài tập

- Viết chương trình bổ sung lần lượt từng xâu ký tự vào ListView với khoảng thời gian 100ms, sử dụng các kỹ thuật :
  - Handler
  - runOnUiThread
  - AsyncTask



End of Lecture

Q&A