LẬP TRÌNH ANDROID

Giảng Viên: Phùng Mạnh Dương

Trường ĐH Công nghệ, ĐHQGHN

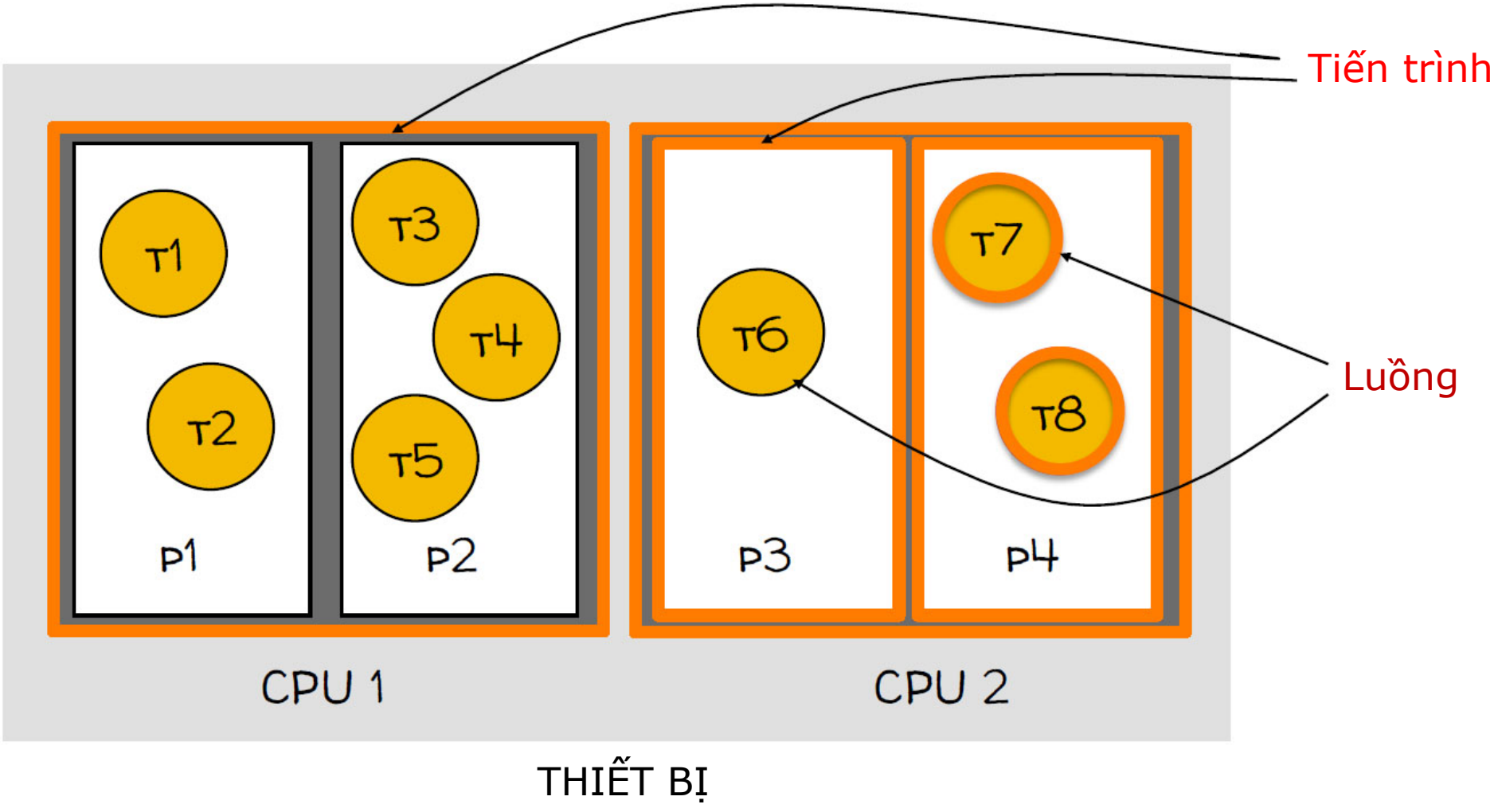
CHƯƠNG 6
Luồng, AsyncTask và Network

- Tổng quan về luồng
- Luồng UI của Android
- Lớp AsyncTask
- Networking

Luồng (Thread)

- Luồng là gì?
  - Về mặt khái niệm: Luồng là một tiến trình đơn vị xử lý của máy tính có thể thực hiện một công việc riêng biệt.
  - Về mặt cài đặt:
    - Mỗi luồng có một bộ đếm chương trình (PC) và ngăn xếp riêng
    - Nhưng chia sẻ bộ nhớ Heap và bộ nhớ tĩnh với các luồng khác trong process.
- Multi-thread là khái niệm cho nhiều tiến trình chạy đồng thời.

Luồng (Thread)



Luồng

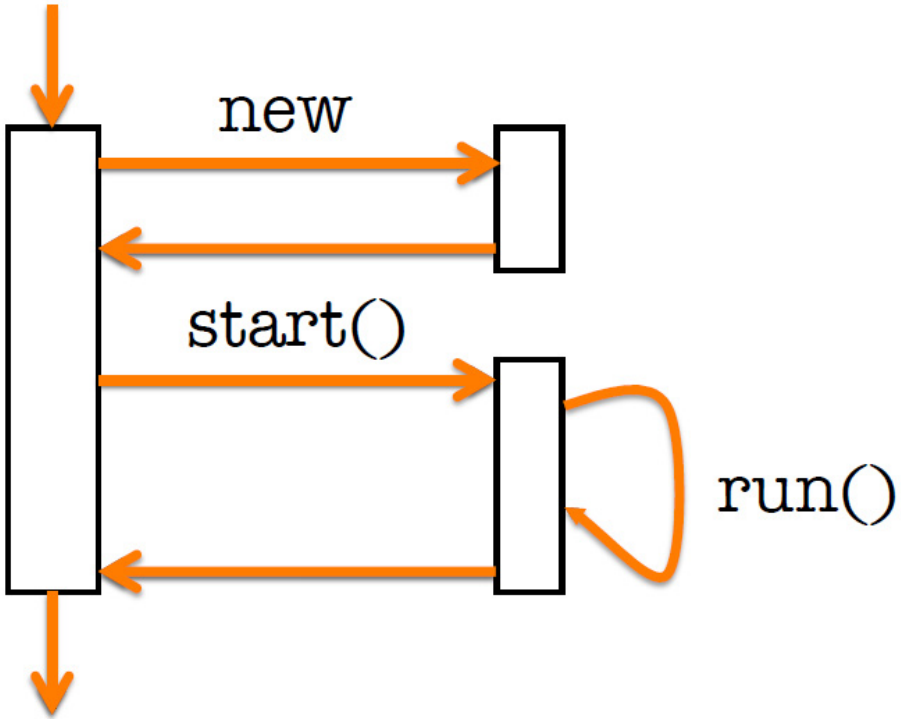
- **Ưu điểm của đa luồng**
  - Mỗi luồng có thể dùng chung và chia sẻ nguồn tài nguyên trong quá trình chạy, nhưng có thể thực hiện một cách độc lập.
  - Ứng dụng trách nhiệm có thể được tách
    - Luồng chính chạy giao diện người dùng
    - Các luồng phụ thực hiện nhiệm vụ gửi đến luồng chính.
- **Nhược điểm của đa luồng:**
  - Càng nhiều luồng thì xử lý càng phức tạp
  - Cần phát hiện tránh các luồng chết, luồng chạy mà không làm gì trong ứng dụng cả

Luồng trong Java

- Được biểu diễn bởi đối tượng kiểu thread trong gói java.lang
- Các luồng trong java được cài đặt giao diện runnable
  - Phải có phương thức public void run()
- Một số phương thức của luồng:
  - void start(): bắt đầu luồng
  - void sleep(): tạm dừng luồng
  - void wait(): luồng hiện tại đợi tới khi một luồng khác gọi notify()

Sử dụng luồng

- Khởi tạo luồng: new Thread()
- Gọi phương thức start() của Thread:
  - Phương thức run() sẽ được gọi.
- Luồng kết thúc khi phương thức run() kết thúc.



Sử dụng luồng

- VD: ThreadingNoThreading
- VD: ThreadingSimple

Sử dụng luồng

- Android không cho phép luồng phụ hiển thị các view.
- Tất cả ứng dụng Android có 1 luồng chính còn gọi là luồng UI.
- Các thành phần ứng dụng trong cùng 1 tiến trình đều sử dụng cùng luồng UI.
- Các tương tác người dùng, phương thức lifecycle, lời gọi hệ thống... được thực thi trong luồng UI.
- Luồng UI không thread-safe: Nếu luồng UI bị chặn bởi tác vụ tốn thời gian sẽ dẫn đến giao diện không thể tương tác.
- Các luồng không phải UI sẽ không thể thiết đặt các View.

Sử dụng luồng

- Để cập nhật giao diện, Android cung cấp 1 số phương thức luôn chạy trong luồng UI
  - boolean View.post (Runnable action)
  - void Activity.runOnUiThread(Runnable action)
- VD:
  - ThreadingViewPost
  - ThreadingRunOnUiThread

Lớp AsyncTask

- Cung cấp một cách thức hiệu quả để quản lý các tác vụ thực hiện ở luồng background và luồng UI.
  - Luồng background:
    - Thực hiện tác vụ cần nhiều thời gian
    - Báo cáo tiến độ thực hiện
  - Luồng UI:
    - Thiết lập ban đầu cho tác vụ trong background
    - Cập nhật tiến độ thực hiện
    - Sử dụng kết quả

Lớp AsyncTask

- Lớp AsyncTask

```
class AsyncTask<Params, Progress, Result> {
}
```

- Có 3 đối số là kiểu Generic:
  - 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
  - Những đối số nào không sử dụng trong quá trình thực thi tiến trình thì ta thay bằng Void.

Các bước cài đặt AsyncTask

Thông thường trong 1 AsyncTask sẽ chứa 4 hàm

- void onPreExecute()
  - Tự động được gọi đầu tiên khi tiến trình được kích hoạt
  - Được thực hiện trong luồng UI
  - Thực hiện cài đặt cho công việc cần nhiều thời gian...
- Result doInBackground (Params...params):
  - Thực hiện tác vụ trong luồng background
  - Có thể gọi: void publishProgress(Progress... values) để báo cáo tiến độ thực hiện.
- void onProgressUpdate (Progress... values): được gọi để trả lời publishProgress và cập nhật giao diện

Các bước cài đặt AsyncTask

- void onPostExecute (Result result)
  - Khi tiến trình kết thúc thì hàm này sẽ tự động được gọi
  - Được thực hiện trong luồng UI để xử lý kết quả.
- Trong 4 hàm trên thì:
  - Hàm doInBackground() bắt buộc phải tồn tại
  - Các hàm khác có thể khuyết, nhưng nên tận dụng đầy đủ 4 hàm đã nêu.
- Cài đặt AsyncTask:
  - Tạo một lớp kế thừa từ AsyncTask
  - Sau đó từ MainActivity ta gọi hàm execute() của tiến trình này.
- VD: ThreadingAsyncTask

Networking

- Nhiều ứng dụng sử dụng dữ liệu và dịch vụ thông qua mạng Internet.
- Android cung cấp nhiều lớp để hỗ trợ mạng:
  - Java.net: socket, url
  - Org.apache: HttpRequest, HttpResponse
  - Android.net: URI, AndroidHttpClient, AudioStream
- Ứng dụng minh họa:
  - Gửi yêu cầu tới máy chủ dịch vụ để lấy dữ liệu về động đất
  - Hiển thị kết quả nhận được:
    - Dữ liệu nguyên thủy (chưa xử lý)
    - Dữ liệu đã được xử lý để lấy thông tin cần

Gửi yêu cầu HTTP dùng Socket

- Để thực hiện được ứng dụng minh họa, chương trình cần:
  - Tạo HTTP request
  - Gửi nó tới server
  - Nhận và xử lý kết quả
  - Hiển thị kết quả
- Android cung cấp một số lớp để thực hiện:
  - Lớp Socket
  - Lớp HttpURLConnection
  - Lớp AndroidHttpClient
- VD: NetworkingSocket

Lớp HttpURLConnection

- Nhược điểm của socket?
- Cung cấp khả năng tương tác cao và chi tiết hơn lớp socket.
- Tuy nhiên API vẫn kém linh động hơn so với lớp HttpAndroidClient.
- VD: NetworkingURL

AndroidHttpClient

- Dựa trên mã nguồn dụ án của dự án Apapche's DefaultHttpClient
- Chia quá trình tương tác HTTP thành 2 đối tượng:
  - Đối tượng yêu cầu
  - Đối tượng trả lời
- VD: NetworkingAndroidHttpClient

Xử lý dữ liệu phản hồi HTTP

- Dữ liệu phải hồi yêu cầu HTTP thường theo định dạng không phù hợp để người dùng đọc trực tiếp.
- 2 định dạng phổ biến:
  - JSON: JavaScript Object Notation
  - XML: eXtensable Markup Language

JSON

- Hướng tới định dạng dữ liệu trao đổi nhẹ, đơn giản.
- Dữ liệu được đóng gói trong 2 kiểu định dạng:
  - Bản đồ các cặp Key/Value
  - Danh sách có thứ tự các giá trị.
- Xem thêm: http://www.json.org/

Dữ liệu động đất (định dạng JSON)

- http://api.geonames.org/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo


```
{"earthquakes”: [
{"eqid":"c0001xgp","magnitude":8.8,"lng":142.369, "
"src":"us”, "datetime":"2011-03-11 04:46:23","depth":"
24.4,"lat":38.322}
{"eqid":"2007hear","magnitude":8.4,"lng":101.3815,"
"src":"us”,"datetime":"2007-09-12 09:10:26","depth": "
30,"lat":-4.5172},
...
{"eqid":"2010xkbv","magnitude":7.5,"lng":91.9379,"
"src":"us”,"datetime":"2010-06-12 17:26:50","depth":"
35,"lat":7.7477}
]
}
```


VD: NetworkingAndroidHttpClientJSON

XML

- XML bao gồm phần đánh dấu và phần nội dung.
  - Phần đánh dấu thể hiện mô tả về cấu trúc layout và logic
  - Nội dung: dữ liệu thực tế
- Xem thêm: http://www.w3.org/TR/xml

Dữ liệu động đất (định dạng XML)

- http://api.geonames.org/earthquakes?north=44.1&south=-9.9&east=-22.4&west=55.2& username=demo

```
<geonames>
 <earthquake>
  <src>us</src>
  <eqid>c0001xgp</eqid>
  <datetime>2011-03-11 04:46:23</datetime>
  <lat>38.322</lat>
  <lng>142.369</lng>
  <magnitude>8.8</magnitude>
  <depth>24.4</depth>
 </earthquake>
...
</geonames>
```

Xử lý dữ liệu XML

- DOM:
  - Chuyển dữ liệu thành một cây các node.
  - Tốn nhiều bộ nhớ nhưng cho phép xử lý đa luồng
- SAX:
  - Đọc dữ liệu XML như một stream và gọi hàm callback của ứng dụng để xử lý.
  - Tốn ít bộ nhớ hơn nhưng chỉ xử lý lần lượt từng luồng dữ liệu
- PULL:
  - Lặp qua các thẻ XML
  - Tốn ít bộ nhớ hơn DOM và linh hoạt trong xử lý hơn SAX
- VD: NetworkingAndroidHttpClientXML