LẬP TRÌNH ANDROID

Giảng Viên: Phùng Mạnh Dương

Trường ĐH Công nghệ, ĐHQGHN

CHƯƠNG 9
Cảm biến, Xác định vị trí và Quản lý dữ liệu

Cảm biến

- Cảm biến và SensorManager
- SensorEvent và SensorEventManager
- Lọc giá trị cảm biến
- Ví dụ

Cảm biến

- Cảm biến là thành phần phần cứng dùng để đo các thông số vật lý của môi trường xung quanh thiết bị
- 3 nhóm cảm biến:
  - Chuyển động
  - Vị trí của thiết bị
  - Môi trường: độ sáng, áp suất, độ ẩm
- VD:
  - Đo chuyển động: Cảm biến gia tốc 3 chiều
  - Đo vị trí: Cảm biến từ trường 3 chiều
  - Môi trường: Cảm biến đo áp suất

SensorManager

- Để sử dụng cảm biến, ứng dụng trước hết lấy tham chiếu tới SensorManager.
- SensorManager là một dịch vụ hệ thống để quản lý các cảm biến.

  `getSystemService(Context.SENSOR_SERVICE )`
- Để truy cập vào 1 cảm biến cụ thể, ta sử dụng:

  `SensorManager.getDefaultSensor(int type)`
- Một số hằng số loại cảm biến:
  - Gia tốc: `Sensor.TYPE_ACCELEROMETER`
  - Từ trường: `Sensor.TYPE_MAGNETIC_FIELD`
  - Áp suất: `Sensor.TYPE_PRESSURE`

SensorEventListener

- Muốn nhận thông tin từ một cảm biến, ứng dụng phải cài đặt giao diện SensorEventListener.
- Giao diện này định nghĩa các hàm callback khi có sự thay đổi của cảm biến như độ chính xác, cập nhật giá trị đo mới.

```
void onAccuracyChanged(Sensor sensor, int accuracy)
void onSensorChanged(SensorEvent event)
```

Đăng ký SensorEvent

- Trước khi ứng dụng có thể nhận SensorEvent, nó cần đăng ký SensorEventListener.

```
public boolean registerListener (SensorEventListener
                                 listener, Sensor sensor, int rate)
```

- Sau khi dùng xong, cần hủy đăng ký.

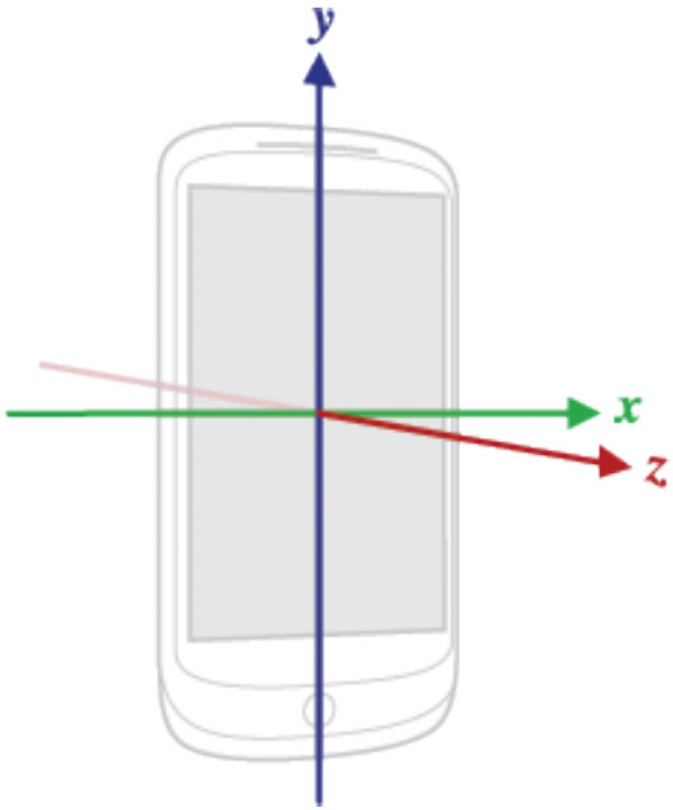
```
public void unregisterListener (SensorEventListener listener,
                                Sensor sensor)
```

SensorEvent

- Kết quả đọc dữ liệu cảm biến sẽ được biểu diễn bởi lớp SensorEvent.
- Dữ liệu cụ thể phụ thuộc vào loại cảm biến tạo ra dữ liệu. Nhưng thông thường sẽ bao gồm:
  - Loại cảm biến
  - Thời điểm đo
  - Độ chính xác của dữ liệu đo
  - Giá trị đo

Hệ tọa độ cảm biến

- Để sử dụng dữ liệu cảm biến, ta cần biết dữ liệu đo được tính như thế nào với mỗi cảm biến.
- Nhiều cảm biến sử dụng hệ tọa độ 3 chiều như hình vẽ.
- Hệ tọa độ không đổi khi hướng của thiết bị thay đổi.



- VD: SensorRawAccelerometer

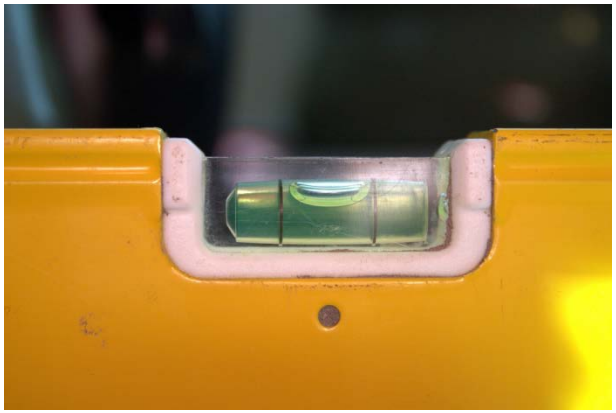
Giá trị gia tốc

- Nếu thiết bị được giữ thẳng, giá trị gia tốc lý tưởng sẽ là:
  - X ≈ 0 m/s2
  - Y ≈ 9.81 m/s2
  - Z ≈ 0 m/s2
- Nhưng thực tế, các giá trị này thường bị dao động do chuyển động, bề mặt không đặt phẳng, nhiễu...

Lọc giá trị cảm biến

- Khi tạo ứng dụng sử dụng cảm biến, ta thường thực hiện các kỹ thuật xử lý để làm mịn dữ liệu.
- 2 kỹ thuật phổng biến là:
  - Bộ lọc thông thấp
  - Bộ lọc thông cao
- Bộ lọc thông thấp:
  - Không quan tâm tới những thay đổi nhỏ
  - Nhấn mạnh vào sự thay đổi chậm và đều đặn
- Bộ lọc thông thấp được sử dụng khi ứng dụng quan tâm tới lực hấp dẫn không đổi, thay vì những nhiễu động nhỏ do rung tay.

Lọc giá trị cảm biến



- VD bộ lọc thông thấp: đo độ song song của thiết bị với mặt đất.

- Bộ lọc thông cao:
  - Quan tâm tới những thay đổi nhỏ, quá độ, nhanh.
  - Không nhấn mạnh vào những thay đổi chậm và đều đặn
- Bộ lọc thông cao được sử dụng khi ứng dụng quan tâm chuyển động của người dùng thay vì lực hấp dẫn.
- VD: Nhạc cụ xúc xắc
- VD: SensorFilteredAccelerometer

SensorCompass

- Sử dụng cảm biến gia tốc và cảm biến từ trường để tạo thành 1 cái la bàn.

Vị trí và Bản đồ

- Thông tin vị trí là gì?
- Các lớp hỗ trợ lấy thông tin vị trí
- Bản đồ
- Các lớp hỗ trợ hiển thị và tùy biến bản đồ

Các dịch vụ xác định ví trí

- Các ứng dụng di động có thể được hưởng lợi từ thông tin vị trí tại mỗi thời điểm.
- Vì vậy, Android cho phép các ứng dụng có thể xác định và xử lý thông tin vị trí.
- VD về sử dụng thông tin vị trí:
  - Tìm các cửa hàng gần vị trí người dùng hiện tại.
  - Tìm đường đi từ vị trí hiện tại tới cửa hàng
  - Tạo bản đồ địa lý

Vị trí

- Android cung cấp lớp Location hỗ trợ xác định vị trí.
- Vị trí bao gồm thông tin:
  - Latitude
  - Longtitude
  - Thời điểm
  - Tùy chọn: độ chính xác, độ cao, tốc độ, hướng

LocationProvider

- Thông tin vị trí được lấy từ LocationProvider.
- Thiết bị có thể truy cập vào nhiều LocationProvider
  - Vệ tinh GPS
  - Trạm BTS của mạng di động
  - Trạm thu phát wifi
- Các loại LocationProvider:
  - Network: bao gồm trạm BTS và wifi
  - GPS: Vệ tinh
  - Passive: lấy thông tin vị trí nhờ các ứng dụng khác

LocationProvider

- Network Provider:
  - Xác định vị trí dựa trên các trạm BTS của mạng di động hoặc điểm truy cập wifi.
  - Cần khai báo:

android.permission.ACCESS_COARSE_LOCATION

android.permission.ACCESS_FINE_LOCATION

- GPS Provider:
  - Xác định vị trí sử dụng các vệ tinh
  - Cần khai báo:

android.permission.ACCESS_FINE_LOCATION

- Passive Provider:
  - Xác định vị trí nhờ các ứng dụng khác
  - Cần khai báo:

android.permission.ACCESS_FINE_LOCATION

LocationProvider

- Mỗi loại Provider có ưu nhược điểm khác nhau về chi phí, độ chính xác, độ sẵn có và thời gian đáp ứng.
  - GPS: chính xác, đáp ứng chậm, ngoài trời
  - Network: kém chính xác, đáp ứng nhanh, có sẵn khi trong tầm phủ sóng di động và wifi.
  - Passive: đáp ứng nhanh nhưng không phải luôn có sẵn

LocationManager

- Lớp LocationManager là một dịch vụ hệ thống giúp truy cập thông tin vị trí.

  `getSystemService(Context.LOCATION_SERVICE)`

- Sau khi lấy tham chiếu có thể:
  - Đọc thông tin vị trí được cập nhật gần đây nhất.
  - Đăng ký để nhận cập nhật thay đổi vị trí.
  - Đăng ký để nhận Intent khi thiết bị tới gần hoặc ra xa một địa điểm/khu vực nào đó.

LocationListener

- Định nghĩa các hàm callback được gọi khi vị trí hoặc locationProvider thay đổi.
- Một giao diện LocationListener bao gồm các hàm sau:

```
void onLocationChanged(Location location)
void onProviderDisabled(String provider)
void onProviderEnabled(String provider)
void onStatusChanged(String provider,int status,
                     Bundle extras)
```

Lấy thông tin vị trí

- Để lấy thông tin vị trí, ta thực hiện các bước sau:
  - Đăng ký LocationListener để lắng nghe cập nhật vị trí từ location provider
  - Cập nhật vị trí "current best estimate"
  - Khi vị trí đủ chính xác yêu cầu, ngừng cập nhật vị trí bằng cách hủy đăng ký LocationListener
  - Sử dụng "current best estimate" làm vị trí hiện tại
- Để xác định vị trí là đủ chính xác có thể dựa vào:
  - Khoảng thời gian đo cho phép bao lâu
  - Cần độ chính xác bao nhiêu
  - Mức độ tiêu tốn pin
- VD: LocationGetLocation

Lời khuyên để tiết kiệm pin

- Luôn kiểm tra giá trị đo gần đây nhất. Nếu đủ tốt -> không cần thực hiện phép đo mới.
- Tần suất cập nhật thấp nhất có thể đồng thời giới hạn thời gian đo.
- Sử dụng độ chính xác thấp nhất cho phép. Chỉ sử dụng GPS nếu thật sự cần thiết.
- Ngừng cập nhật vị trí trong onPause()

Bản đồ

- Android hỗ trợ Bản đồ thông qua Google Maps Android v2 API.
- API này cung cấp 1 số loại bản đồ:
  - Thông thường: bản đồ đường phố truyền thống
  - Vệ tinh: Hiển thị không ảnh của 1 khu vực
  - Lai: Thông thường + Vệ tinh
  - Địa hình: Chi tiết về địa hình bề mặt

Tùy biến bản đồ

- Android cho phép tùy biến bản đồ theo 1 số cách:
  - Thay đổi vị trí Camera
  - Thêm Icon đánh dấu hay chồng ảnh lên trên bản đồ
  - Đáp ứng với cử chỉ
  - Hiển thị vị trí hiện tại

Một số lớp bản đồ

- Để hiển thị bản đồ, ta dùng 1 số lớp sau:
  - GoogleMap: biểu diễn và quản lý bản đồ
  - MapFragment: Hiển thị GoogleMap trong 1 fragment
  - Camera: định nghĩa phần bản đồ hiển thị trên màn hình và xác định điểm nhìn của người dùng.
  - Marker: biểu diễn các biểu tượng hay cửa sổ pop-up hiển thị thông tin chi tiết về 1 vị trí.

Map Permission

- Để sử dụng bản đồ, ta cần 1 số quyền sau:

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

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

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

<uses-permission android:name="com.google.android.providers.
gsf.permission.READ_GSERVICES"/>

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

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

Triển khai 1 ứng dụng bản đồ

- Các bước thực hiện:
  - Cấu hình Google Play Service
  - Đăng ký 1 API key
  - Thiết lập quyền, các cài đặt và API key trong file Manifest.xml
  - Chèn bản đồ vào trong ứng dụng
- VD: MapEarthQuakeMap

LƯU TRỮ DỮ LIỆU

- Lưu trữ dữ liệu với SharedPreferences
- Thao tác file với bộ nhớ trong
- Thao tác file với bộ nhớ ngoài
- Làm việc với cơ sở dữ liệu SQLite

Giới thiệu về lưu trữ dữ liệu trên Android

- Các ứng dụng di động có thể cần phải tạo và xử lý lượng dữ liệu ở nhiều cấp độ khác nhau.
- Android cung cấp các lớp hỗ trợ quản lý dữ liệu giữa các phiên làm việc của ứng dụng:
  - SharedPreferences: Lưu trữ và xử lý dữ liệu nhỏ, nguyên thủy. VD: Username.
  - Bộ nhớ trong: Lưu trữ dữ liệu nhỏ và dành riêng cho từng ứng dụng. VD: các file tạm sinh ra bởi ứng dụng.
  - Bộ nhớ ngoài: Lưu trữ dữ liệu lớn và chung cho các ứng dụng như nhạc hay video.
  - Cơ sở dữ liệu: Lưu trữ dữ liệu có cấu trúc và riêng cho từng ứng dụng.

SharedPreferences

- Lưu trữ các loại dữ liệu đơn giản theo định dạng tên-giá trị. VD: String, Float
- Dữ liệu lưu trữ tồn tại vĩnh viễn qua các phiên làm của ứng dụng.

→ Cho phép người dùng tạo dữ liệu trong 1 phiên làm việc. Rồi sau đó tắt ứng dụng. Bật lại ứng dụng mà vẫn có thể truy cập vào dữ liệu đã tạo ra ở phiên trước.

- Thường được sử dụng để lưu trữ lâu dài dữ liệu tùy biến của ứng dụng. VD:
  - Tên tài khoản
  - Các mạng wifi hay sử dụng
  - Các lựa chọn tùy biến của người dùng

Sử dụng SharedPreferences

- Lấy tham chiếu:
  - Tới đối tượng SharedPreference của một Activity: Activity.getPreferences (int mode)
    - MODE_PRIVATE: Dữ liệu là riêng cho ứng dụng hiện tại
  - Tới đối tượng SharedPreference thông qua tên:

Context.getSharedPreferences (String name, int mode)

- Name: Tên của SharedPreference
- Mode: MODE_PRIVATE

Sử dụng SharedPreferences

- Ghi vào đối tượng SharedPreference:
  - Gọi: SharedPreferences.edit()

    Trả về: Một thể hiện của SharedPreferences.Editor
  - Thêm hoặc thay đổi giá trị:
    - putInt(String key, int value)
    - putString(String key, String value)
    - remove(String key)
  - Ghi lại thay đổi:
    - SharedPreferences.Editor.commit()

Sử dụng SharedPreferences

- Đọc dữ liệu lưu bởi SharedPreference:

  Sử dụng các phương thức
  - getAll()
  - getBoolean(String key, ...)
  - getString(String key, ...)

- VD: DataManagementSharedPreferences

File

- Trong Android, phần lưu trữ được chia làm 2: Bộ nhớ trong (Internal) và bộ nhớ ngoài (External)
  - Bộ nhớ trong: Thường là bộ nhớ flash đi kèm với thiết bị.
  - Bộ nhớ ngoài: Thẻ nhớ có thể tháo rời.
- Bộ nhớ trong: Lưu trữ các tập dữ liệu nhỏ, riêng cho từng ứng dụng.
- Bộ nhớ ngoài: Lưu trữ dữ liệu lớn và chung cho các ứng dụng.

File API cho bộ nhớ trong

- Lớp biểu diễn một thực thể file hệ thống xác định bởi một đường dẫn.
- Mở file để đọc ghi (Tạo nếu file chưa tồn tại):

FileOutputStream openFileOutput (String name, int mode)

- Mở file để đọc:

FileInputStream openFileInput (String name)

- VD: DataManagementFileInternalMemory

Sử dụng File với bộ nhớ ngoài

- Android hỗ trợ thao tác với bộ nhớ ngoài.
- Bước đầu tiên là kiểm tra trạng thái của bộ nhớ:

String Environment.getExternalStorageState()

  - MEDIA_MOUNTED – Tồn tại bộ nhớ. Có thể đọc/ghi.
  - MEDIA_MOUNTED_READ_ONLY - Tồn tại bộ nhớ. Chỉ đọc.
  - MEDIA_REMOVED – Không tồn tại bộ nhớ.

- Để ghi vào bộ nhớ ngoài, cần được cấp quyền:

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

- VD: DataManagementFileExternalMemory

Cơ sở dữ liệu

- Android cung cấp SQLite cho phép ứng dụng tạo và sử dụng cơ sở dữ liệu lưu trữ ở bộ nhớ trong.
- SQLite: http://vi.wikipedia.org/wiki/SQLite
  - Hệ thống cơ sở dữ liệu quan hệ nhỏ gọn, hoàn chỉnh, có thể cài đặt bên trong các trình ứng dụng khá
  - Kích thước chương trình gọn nhẹ, với cấu hình đầy đủ chỉ không đầy 300 kB
  - Tuân theo chuẩn SQL92

Sử dụng cơ sở dữ liệu

- Sử dụng các phương thức dựa trên lớp SQLiteOpenHelper:
  - Tạo lớp con của SQLiteOpenHelper
  - Gọi super() trong phương thức khởi tạo của lớp con để khởi tạo cơ sở dữ liệu
  - Override onCreate(): Thường tạo table (CREATE TABLE)
  - Override onUpgrade()
  - Sử dụng các phương thức của SQLiteOpenHelper để mở và sử dụng CSDL.
  - Thực thi các thao tác trên CSDL.
- Database lưu tại: /data/data/package_name/databases
- Quản lý: http://sqlitebrowser.org/
- VD: DataManagementSQL