XML – SAX parser

# SAX parser (Simple API for XML)

Có 2 cách để phân tích dữ liệu XML là DOM và SAX. Cách phân tích bằng DOM cần phải nạp toàn bộ dữ liệu XML vào trong bộ nhớ trước khi có thể sử dụng được nó, cách này sẽ tốn bộ nhớ và tốn thời gian hơn, ưu điểm là viết ít mã.

SAX lại không tốn bộ nhớ và thời gian nên sử dụng tốt hơn cho thiết bị di động. Điểm hay của SAX là nó duyệt qua từng phần tử và thuộc tính, khi đó ta có thể lấy ra và quyết định xem cần lấy gì để đưa vào bộ nhớ nhưng bạn phải viết nhiều code hơn (tùy trường hợp).

# Cách sử dụng SAX

SAX parse dùng callback function (org.xml.sax.helpers.DefaultHandler) để thông báo cho client biết cấu trúc của tài liệu XML. Ta cần phải kế thừa từ lớp DefaultHandler và override lên một số hàm để phân tích dữ liệu XML.

Ta nên override các phương thức sau:

* startDocument() : Phương thức được gọi tại bắt đầu của 1 tài liệu XML.
* endDocument() : Phương thức được gọi tại kết thúc của 1 tài liệu XML.
* startElement() : Phương thức được gọi khi bắt đầu 1 phần tử.
* endElement() : Phương thức được gọi khi kết thúc 1 phần tử.
* characters(): Phương thức được gọi với nội dung text giữa thẻ bắt đầu và thẻ kết thúc của một phần tử trong tài liệu XML.

Thực hành

## Bài 1: Viết chương trình đọc 1 file XML và dùng SAX để phân tích nó.

(sử dụng file dulieu.xml ở bài trước)

### Chuẩn bị:

1. Chép file dulieu.xml lên localhost.
2. Tạo project tên “xmlsaxparser”.
3. Mở file Android ”Manifest.xml” phân quyền cho file như sau:

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

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>

1. Dựa và file xml để xây dựng class. Tạo 1 class tên Item có các thuộc tính giống với xml. (chú ý: nên xây các thuộc tính là private rồi xây các hàm get, set để lấy và gán dữ liệu cho an toàn).

**public** **class** Item {

String id;

String name;

String cost;

String description;

}

### Xây dựng class kế thừa Defaulthandler để điều khiển XML

1. Xây một class mới tên MySaxHandler kế thừa từ lớp DefaultHandler sau đó override lên 3 phương thức là characters, endElement, startElement.

**public** **class** MySaxHandler **extends** DefaultHandler{

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length)

**throws** SAXException {

// **TODO** Auto-generated method stub

**super**.characters(ch, start, length);

}

@Override

**public** **void** endElement(String uri, String localName, String qName)

**throws** SAXException {

// **TODO** Auto-generated method stub

**super**.endElement(uri, localName, qName);

}

@Override

**public** **void** startElement(String uri, String localName, String qName,

Attributes attributes) **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.startElement(uri, localName, qName, attributes);

}

}

1. Xây dựng biến toàn cục

ArrayList<Item> items;

Item item\_tam;

String chuoi\_tam;

1. Xây thêm 2 hàm là hàm tạo để khởi tạo ArrayList items và hàm getItems trả về ArrayList.

**public** MySaxHandler()

{

items=**new** ArrayList<Item>();

}

**public** ArrayList<Item> getItems()

{

**return** items;

}

1. Trong hàm startElement sẽ chạy khi gặp 1 phần tử trong html, ta gán chuoi\_tam về chuỗi rỗng và nếu phần tử này là “item” thì ta sẽ khởi tạo mới “item\_tam” để sẵn sàng đổ dữ liệu vào item\_tam.

@Override

**public** **void** startElement(String uri, String localName, String qName,

Attributes attributes) **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.startElement(uri, localName, qName, attributes);

chuoi\_tam="";

**if**(qName.equalsIgnoreCase("item"))

item\_tam=**new** Item();

}

1. Trong hàm characters là mỗi lần gặp 1 phần tử XML ta sẽ có phần nội dung bên trong phần tử đó. Mỗi khi ta gặp 1 phần tử XML thì ta sẽ lấy nội dung của nó gán vào chuoi\_tam.

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length)

**throws** SAXException {

// **TODO** Auto-generated method stub

**super**.characters(ch, start, length);

chuoi\_tam=**new** String(ch,start,length);

}

1. Hàm endElement là khi kết thúc 1 phần tử. Khi kết 1 phần tử đó là id, name, cost, description thì ta sẽ gán chuoi\_tam vào cho thuộc tính của item\_tam tương ứng. Item\_tam được tạo nhờ hàm startElement và dữ liệu bên trong nút được tạo nhờ hàm characters. Cuối cùng phát xét 1 trường hợp quan trọng là khi kết thúc phần tử “item” tức là khi đã có đầy đủ dữ liệu của 1 item, ta sẽ add item đó vào ArrayList.

@Override

**public** **void** endElement(String uri, String localName, String qName)

**throws** SAXException {

// **TODO** Auto-generated method stub

**super**.endElement(uri, localName, qName);

**if**(qName.equalsIgnoreCase("item"))

items.add(item\_tam);

**else** **if**(qName.equalsIgnoreCase("id"))

item\_tam.id=chuoi\_tam;

**else** **if**(qName.equalsIgnoreCase("name"))

item\_tam.name=chuoi\_tam;

**else** **if**(qName.equalsIgnoreCase("cost"))

item\_tam.cost=chuoi\_tam;

**else** **if**(qName.equalsIgnoreCase("description"))

item\_tam.description=chuoi\_tam;

}

### Xây class MySaxParser

1. Tạo một class mới tên MySaxParser. Class này có 1 hàm tên xmlParser (có thể cho hàm này static vì có 1 hàm duy nhất) nhận vào một InputStream và trả về một List là danh sách các Item.

**public** **class** MySaxParser {

**public** List<Item> xmlParser(InputStream is)

{

List<Item> items=**null**;

**try**{

//tao xmlreader tu xmlparser

XMLReader xmlreader=SAXParserFactory.*newInstance*().newSAXParser().getXMLReader();

//tao saxhandler

MySaxHandler saxhandler=**new** MySaxHandler();

//luu handler vao xmlreader

xmlreader.setContentHandler(saxhandler);

xmlreader.parse(**new** InputSource(is));

//lay danh sach cac item bo vao items

items=saxhandler.getItems();

}

**catch**(Exception e)

{

Log.*d*("loi","lay khong duoc "+ e.toString());

}

**return** items;

}

}

### Sử dụng ở class chính

1. Quay lại activity chính. Để dễ trên layout kéo vào 1 TextView, qua file java khai báo và ánh xạ TextView.
2. Trong file java khai báo các biến toàn cục gồm: items là một ArrayList dùng để chứa các item có được khi parser. Biến path chứa đường dẫn đến file xml. Biến InputStream vì class MySaxParser nhận vào một InputStream.

List<Item> items=**new** ArrayList<Item>();

String path="http://10.0.2.2:8080/testandroid/dulieu.xml";

InputStream is;

String chuoi="";

TextView tv;

1. Do vấn đề phiên bản mới nên làm việc với network không nằm trên mainthread được do đó ta phải xây một class kế thừ từ AsyncTask. Trong doInBackground ta tạo ra URL rồi dùng URLConnection để kết nối và lấy dữ liệu cho InputStream. Từ InputStream ta dùng lớp MySaxParser để phân tích dữ liệu xml và trả về ArrayList items. (chú ý: for nơi đây không cần thiết, khi đã có ArrayList items ta có thể dùng Adapter để đổ dữ liệu vào ListView, Spinner,..... ở đây để đơn giản ta duyệt arraylist để đưa vào String và đổ lên TextView cho dễ).

**class** MyAsyncTask **extends** AsyncTask<Void,Void,Void>

{

@Override

**protected** Void doInBackground(Void... arg0) {

// **TODO** Auto-generated method stub

**try** {

URL url=**new** URL(path);

URLConnection connection=url.openConnection();

InputStream is=connection.getInputStream();

MySaxParser mysaxparser=**new** MySaxParser();

items=mysaxparser.xmlParser(is);

**for**(**int** i=0;i<items.size();i++)

{

chuoi+=items.get(i).name+" ";

chuoi+=items.get(i).cost+ "\n";

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** **null**;

}

@Override

**protected** **void** onPostExecute(Void result) {

// **TODO** Auto-generated method stub

**super**.onPostExecute(result);

tv.setText(chuoi);

}

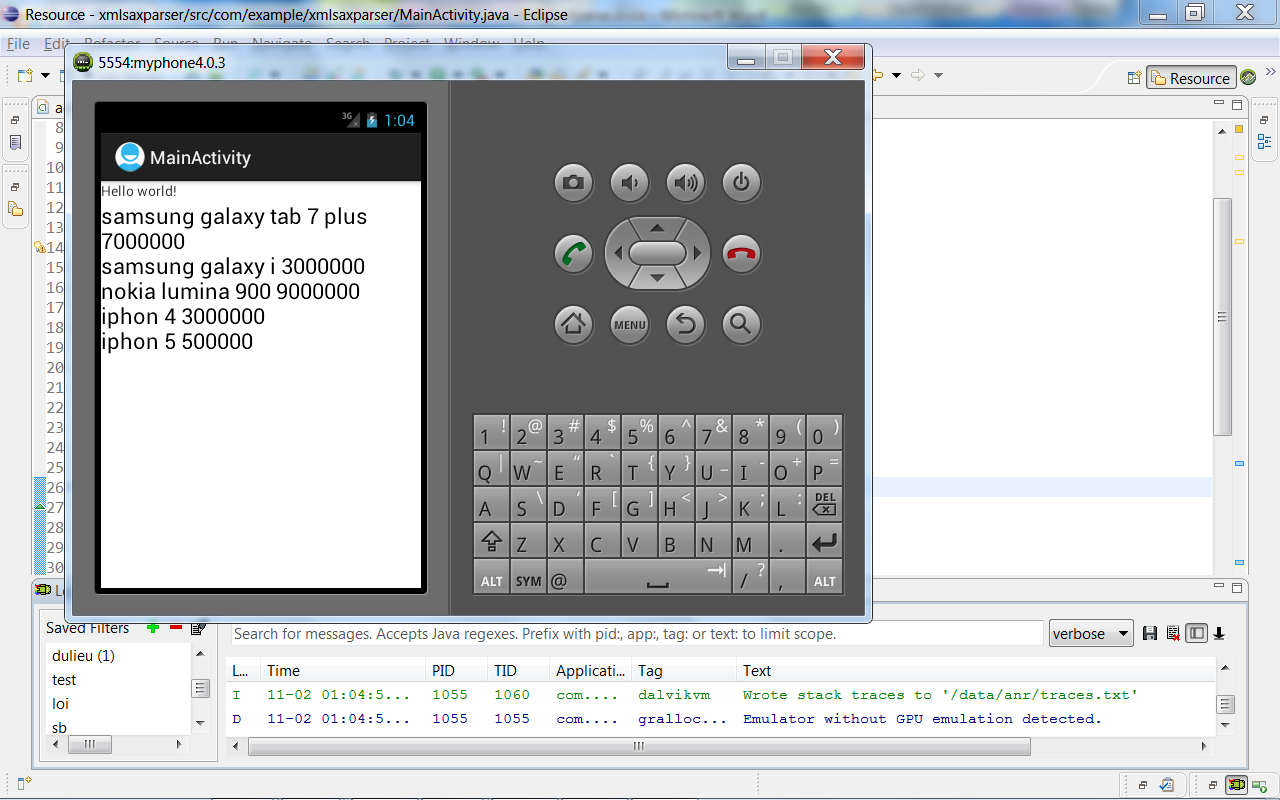
}

1. Cuối cùng trong hàm onCreate ta tạo đối tượng MyAsyncTask và chạy nó.

MyAsyncTask a=**new** MyAsyncTask();

a.execute();

16. Chạy chương trình để xem kết quả.



# Bài 2 : Sửa bài 1 lại để đổ dữ liệu lên ListView. Không đọc dữ liệu từ web mà hãy chép file xml vào thư mục Assets.

Gợi ý: Đọc xml từ Assets.

items = mysaxparser.parse(getAssets().open("dulieu.xml"));