XML Parser - DOM

# XML - DOM

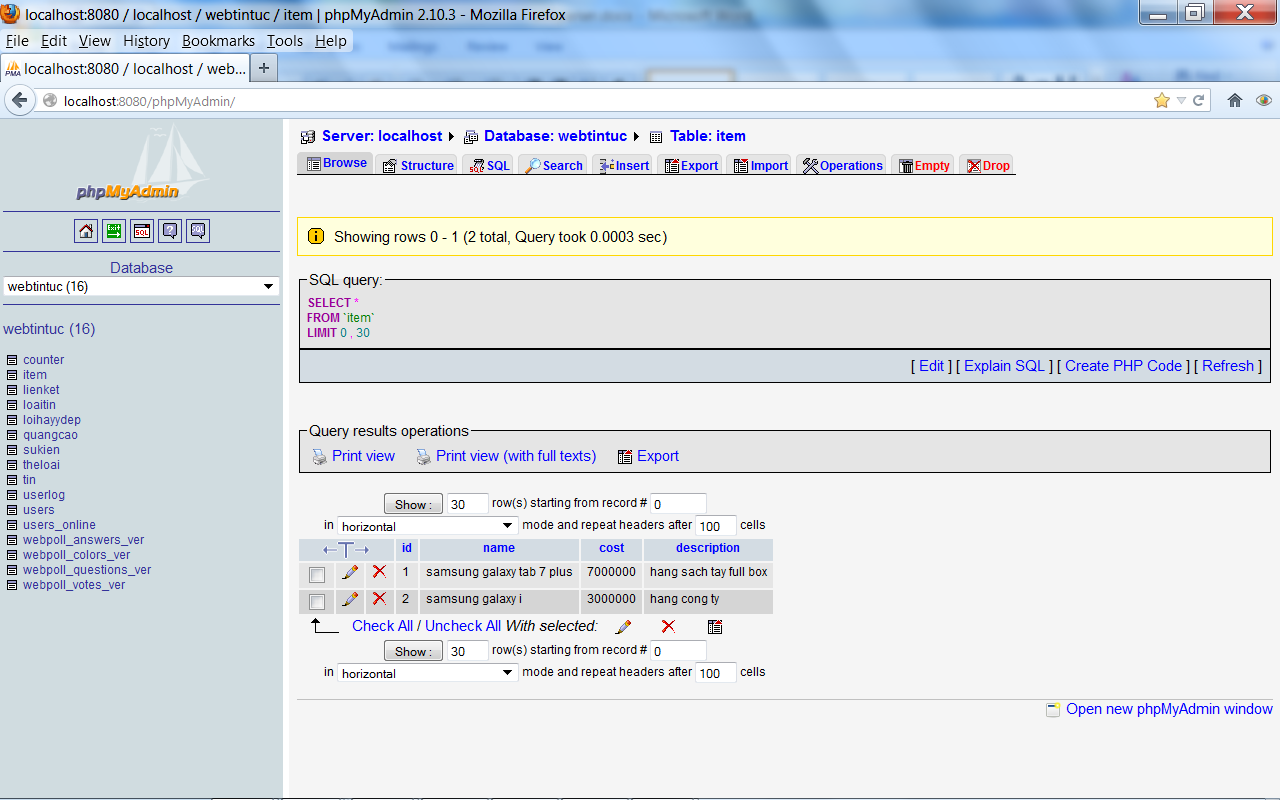
Viết tắt eXtensible Markup Language (Ngôn ngữ đánh dấu mở rộng). Mục đích chính của XML là đơn giản hóa việc chia sẽ dữ liệu giữa các hệ thống khác nhau, đặt biệt là các hệ thống kết nối qua Internet.

Về cấu trúc nó nhìn hơi giống với HTML là cũng cấu trúc từ các thẻ, trong thẻ có thuộc tính và trong thẻ có thể có các thẻ con nhưng khác HTML là các thẻ là do người dùng tự định nghĩa. Mục đích của nó cũng khác HTML là nó dùng để biểu diễn dữ liệu có cấu trúc chứ không phải dùng hiển thị dữ liệu như HTML.

Nói đơn giản nó dùng để lưu trữ dữ liệu thuần văn bản nhưng có cấu trúc để giúp giao tiếp dữ liệu giữa các nền khác nhau,các ngôn ngữ khác nhau.

VD đơn giản: có 1 database lưu trên server bằng MySQL, giờ muốn gởi dữ liệu của nó qua internet vào lưu xuống database trên máy android dùng SQLite. Ta thấy 2 database khác nhau nằm trên 2 nền khác nhau làm sao truyền dữ liệu qua lại đây?. Khi đó ta dùng XML. Ta sẽ đọc dữ liệu từ server trong MySQL và đưa về dạng XML sau đó truyền qua mạng, đến máy Android sẽ đọc XML xuống và phân tích để lấy dữ liệu ra đưa vào SQLite. Vì XML là thuần văn bản nên đọc được trên các nền khác nhau, nhưng lại có cấu trúc nên có thể đưa về dạng database.

Ví dụ 1 bảng trong database MySql như hình, có thể đưa về cấu trúc XML:



<?xml version="1.0" encoding="utf-8" ?>

<menu>

<!-- Table item -->

<item>

<id>1</id>

<name>samsung galaxy tab 7 plus</name>

<cost>7000000</cost>

<description>hang sach tay full box</description>

</item>

<item>

<id>2</id>

<name>samsung galaxy i</name>

<cost>3000000</cost>

<description>hang cong ty</description>

</item>

</menu>

# XML DOM (Document Object Model) Parser

Là việc lấy file XML từ server về và phân tích để có được dữ liệu theo cấu trúc bên trong của nó.

DOM(Mô hình đối tượng tài liệu) là một cách để phân tích cấu trúc của tài liệu XML (cách khác là dùng SAX). Nó sẽ load toàn bộ dữ liệu vào bộ nhớ và sau đó phân tích tài liệu dựa vào các Node.

Để có thể lấy được file XML từ server ta cũng dùng các thao tác HTTP, ví dụ dùng DefaultHttpClient, HttpPost, và dùng .execute() để lấy chuỗi XML (chuỗi XML chính là nội dung trong file XML).

Sau khi đã có chuỗi XML ta sẽ

### Tạo ra 1 đối tượng DocumentBuilderFactory.

Đối tượng này định nghĩa một factory API cho phép chương trình có thể phân tích một cây DOM object từ tài liệu HTML.

DocumentBuilderFactory dbf=DocumentBuilderFactory.*newInstance*();

### Tạo đối tượng DocumentBuilder

Tạo đối tượng này từ hàm newDocumentBuilder() của đối tượng DocumentBuilderFactory

Đối tượng này sẽ cho phép ta phân tích cú pháp từ nhiều dạng đối tượng đầu vào. Các nguồn đầu vào gồm InputStream, Files, URLs và SAX InputSources.

DocumentBuilder db=dbf.newDocumentBuilder();

### Tạo đối tượng đầu vào InputSource

Nó cho phép đóng gói thông tin của nguồn đầu vào trong một đối tượng duy nhất, nó sẽ gồm một public identifier (định danh công cộng), system identifer (định danh hệ thống), một byte stream hoặc một character stream.

Sau đó ta dùng hàm setCharacterStream và đưa chuỗi xml vào để xem chuỗi xml như một Character stream cho đối tượng đầu vào InputSource.

InputSource is=**new** InputSource();

is.setCharacterStream(**new** StringReader(xml));

### Tạo đối tượng Document

Gọi hàm parse() của đối tượng DocumentBuilder và truyền vào InputSource để tạo ra đối tượng Document. Từ đối tượng document ta có thể lấy ra node, tập hợp các node trong cây DOM.

Document document=db.parse(is);

Thực Hành

# Bài 1: Viết chương trình cho phép đọc 1 file XML từ trên server (localhost) và phân tích dữ liệu XML.

Thực hiện:

1. Chép file “dulieu.xml” lên localhost.
2. Tạo 1 project mới tên “docfilexml”. Chọn máy ảo version 1.6 .
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. Để tiện cho việc phát triển về sau ta sẽ xây 1 class mới có tên “XMLParser.xml”. Trong file ta xây dựng hàm đầu tiên tên getXmlFromUrl. Hàm này có chức năng đưa vào chuỗi đường dẫn đến file XML trên server, hàm này sẽ trả về chuỗi dữ liệu xml của file đưa vào.

Hàm này đầu tiên khởi tạo đối tượng DefaultHttpClient, khởi tạo đối tượng httppost để lấy dữ liệu theo dạng HTTP-POST.

Sau đó sẽ gọi httpclient.execute() để lấy dữ liệu trên server và đổ vào chuỗi xml rồi tra chuỗi về.

Chú ý phần nằm trong các dấu // là một cách viết khác có thể tham khảo thêm.

**public** String getXmlFromUrl(String url)

{

String xml=**null**;

DefaultHttpClient httpclient=**new** DefaultHttpClient();

HttpPost httppost=**new** HttpPost(url);

**try** {

///dung qua doi tuong EntityUtils de lay ve httpentity hoac tao ra ResponseHanler deu duoc

//HttpResponse httpresponse=httpclient.execute(httppost);

//HttpEntity httpentity=httpresponse.getEntity();

//xml=EntityUtils.toString(httpentity);

//xml=httpclient.execute(httppost);

/////////////////

ResponseHandler<String> handler = **new** BasicResponseHandler();

xml = httpclient.execute(httppost, handler);

} **catch** (Exception e)

{

Log.*d*("loi",e.toString());

}

**return** xml;

}

1. Tiếp theo ta xây tiếp 1 hàm tên getDomElement. Hàm này nhận vào 1 chuỗi (dạng XML) và trả về một đối tượng Document.

Trong hàm đầu tiên ta tạo ra một đối tượng từ lớp DocumentBuilderFactory.

Sau đó tạo ra đối tượng DocumentBuilder từ đối tượng DocumentBuilderFactory.

Sao đó ta khởi tạo 1 đối tượng đầu vào (InputSource) và dùng hàm setCharacterStream để gán dữ liệu xml vào dữ liệu đầu vào như 1 dạng luồng kí tự Character Stream.

Sao đó gọi hàm parser của đối tượng DocumentBuilder và truyền vào InputSource và ta nhận được đối tượng kiểu Document. Cuối cùng trả về đối tượng Document.

**public** Document getDomElement(String xml)

{

Document document=**null**;

DocumentBuilderFactory dbf=DocumentBuilderFactory.*newInstance*();

**try**

{

DocumentBuilder db=dbf.newDocumentBuilder();

InputSource is=**new** InputSource();

is.setCharacterStream(**new** StringReader(xml));

document=db.parse(is);

}**catch**(Exception e)

{

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

}

**return** document;

}

1. Tiếp theo xây hàm getValue hàm này nhận vào 1 element, và một chuỗi tên. Nó tạo ra một danh sách các nút (NodeList) có tên truyền vào. Sau đó lấy ra giá trị của nút đầu tiên (cũng là nút duy nhất theo bài).

**public** String getValue(Element element,String str)//import org.w3c.dom.Element

{

NodeList nl=element.getElementsByTagName(str).item(0).getChildNodes();

Node nvalue=(Node)nl.item(0);

**return** nvalue.getNodeValue();

}

1. Quay về file java chính. Đầu tiên khai báo các biến toàn cục gồm chuỗi địa chỉ truy đến file XML, các biến bắt đầu bằng từ KEY\_ để lưu tên của các nút có trong file xml, các biến nên được lưu trước để sau này có sửa thì sửa cho dể, ngoài ra cũng nên dùng từ khóa final để không cho phép các hàm sửa nội dung của nó.

**static** **final** String *url*="http://10.0.2.2:8080/testandroid/dulieu.xml";

**final** String KET\_ITEM="item";

**final** String KEY\_NAME="name";

**final** String KEY\_COST="cost";

**final** String KEY\_DESCRIPTION="description";

String dulieu="";

TextView tv;

1. Trong hàm onCreate ánh xạ textview và viết thêm. Đầu tiên tạo đối tượng XMLParser, đối tượng này ta đã tự viết class cho nó ở trên. Tiếp theo khai báo chuỗi vào gọi hàm getXmlFromUrl() để lấy về toàn bộ chuỗi XML. Sao đó tạo đối tượng Document để lấy DOM. Tạo NodeList để lấy tất cả các nút tên “item”.

XMLParser xmlparser=**new** XMLParser();

String xml=xmlparser.getXmlFromUrl(*url*); //lay xml

Log.*d*("xml",xml);

Document doc=xmlparser.getDomElement(xml); //lay dom element

NodeList nl=doc.getElementsByTagName(KET\_ITEM);

1. Sau đó dùng vòng lặp for duyệt các nút và lấy dữ liệu con của nó gồm name, cost và description. Hàm .getValue ta đã xây dựng sẵn.

**for**(**int** i=0;i<nl.getLength();i++)

{

Element e=(Element)nl.item(i);

String name=xmlparser.getValue(e,KEY\_NAME);

String cost=xmlparser.getValue(e,KEY\_COST);

String description=xmlparser.getValue(e,KEY\_DESCRIPTION);

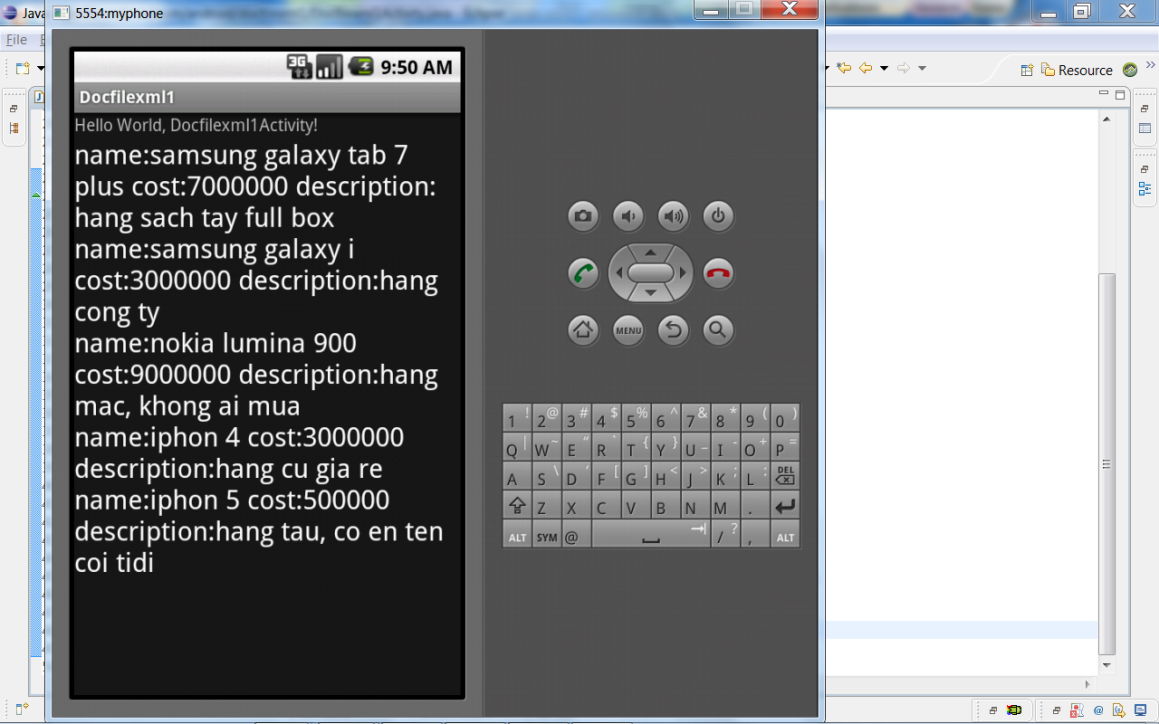
Log.*d*("dulieu", "name:" + name + " cost:"+ cost + " description:"+ description);

dulieu+="name:" + name + " cost:"+ cost + " description:"+ description+"\n";

}

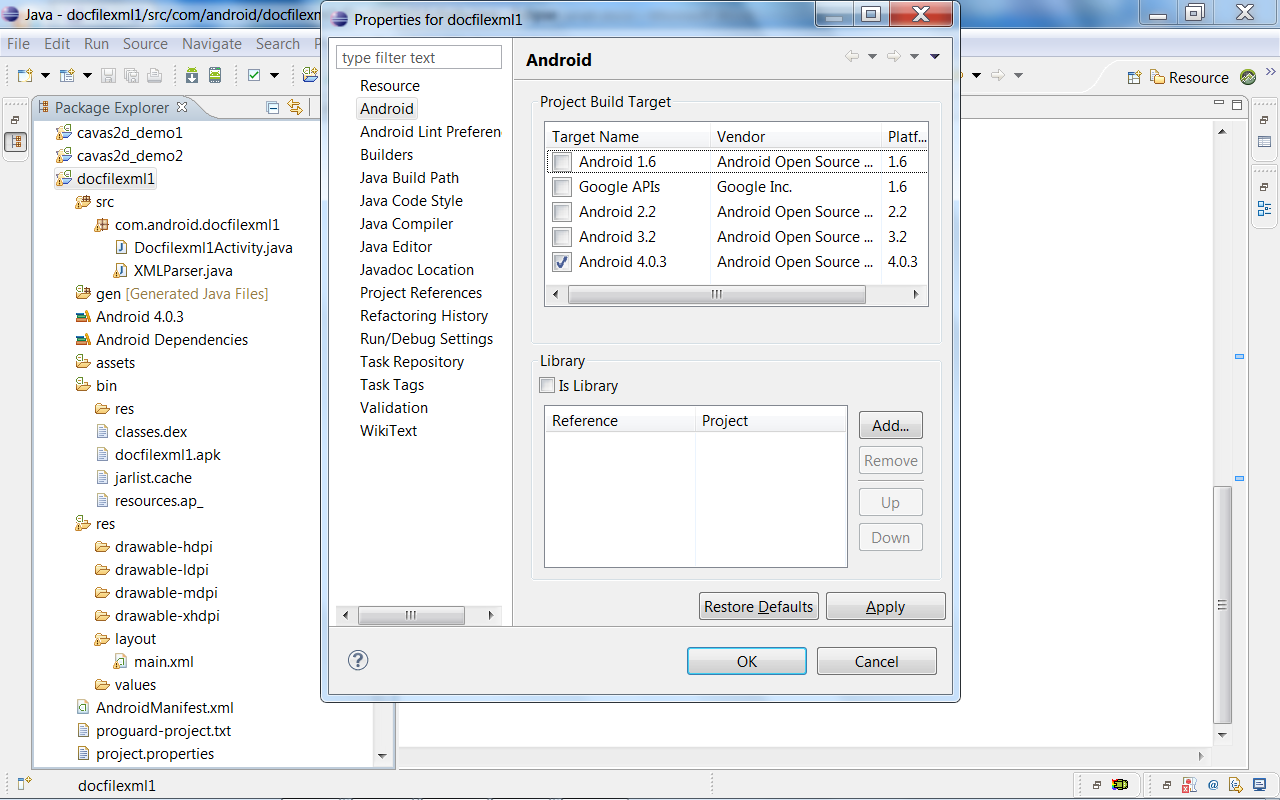
tv.setText(dulieu);

1. Chạy chương trình và xem kết quả.



# Bài 2 : Chuyển bài 1 lên version cao hơn.

1. Click phải project chọn “Property” chọn tab Android, check vào version cao rồi nhấn OK.



1. Mở tập tin Manifest và sửa version lên 15.

<uses-sdk android:minSdkVersion=*"15"* />

1. Mở file java chính lên. Khai báo 1 biến handle toàn cục

Handler handler;

1. Cũng trong class chính, sau hàm onCreate ta tạo 1 Thread mới truyền vào Runnable trong runnable ta override hàm run().

Thread mythread=**new** Thread(**new** Runnable(){

**public** **void** run() {

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

}

});

1. Cut mã lệnh phần truy vấn XML và paste vào hàm run.

XMLParser xmlparser=**new** XMLParser();

String xml=xmlparser.getXmlFromUrl(*url*); //lay xml

Log.*d*("xml",xml);

Document doc=xmlparser.getDomElement(xml); //lay dom element

……………………………………………………………………………………………………………………..

tv.setText(dulieu);

1. Bỏ lệnh tv.setText(dulieu) ở cuối và thay bằng các lệnh sau:

//tv.setText(dulieu);

Bundle bundle=**new** Bundle();

bundle.putString("dulieu", dulieu);

Message me=**new** Message();

me.setData(bundle);

handler.sendMessage(me);

1. Cuối cùng trong onCreate ta thêm các lệnh sau:

handler=**new** Handler()

{

@Override

**public** **void** handleMessage(Message msg) {

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

**super**.handleMessage(msg);

Bundle bundle=**new** Bundle();

bundle=msg.getData();

String chuoi=bundle.getString("dulieu");

tv.setText(chuoi);

}

};

mythread.start();

1. Chạy chương trình để thấy nó chạy tốt trên version cao.

# Bài 3: Phát triển bài 2 để đưa dữ liệu xml vào 1 ListView, khi chọn 1 mục sẽ chuyển sang 1 Activity khác với ListView được chọn.

