RSS

# RSS

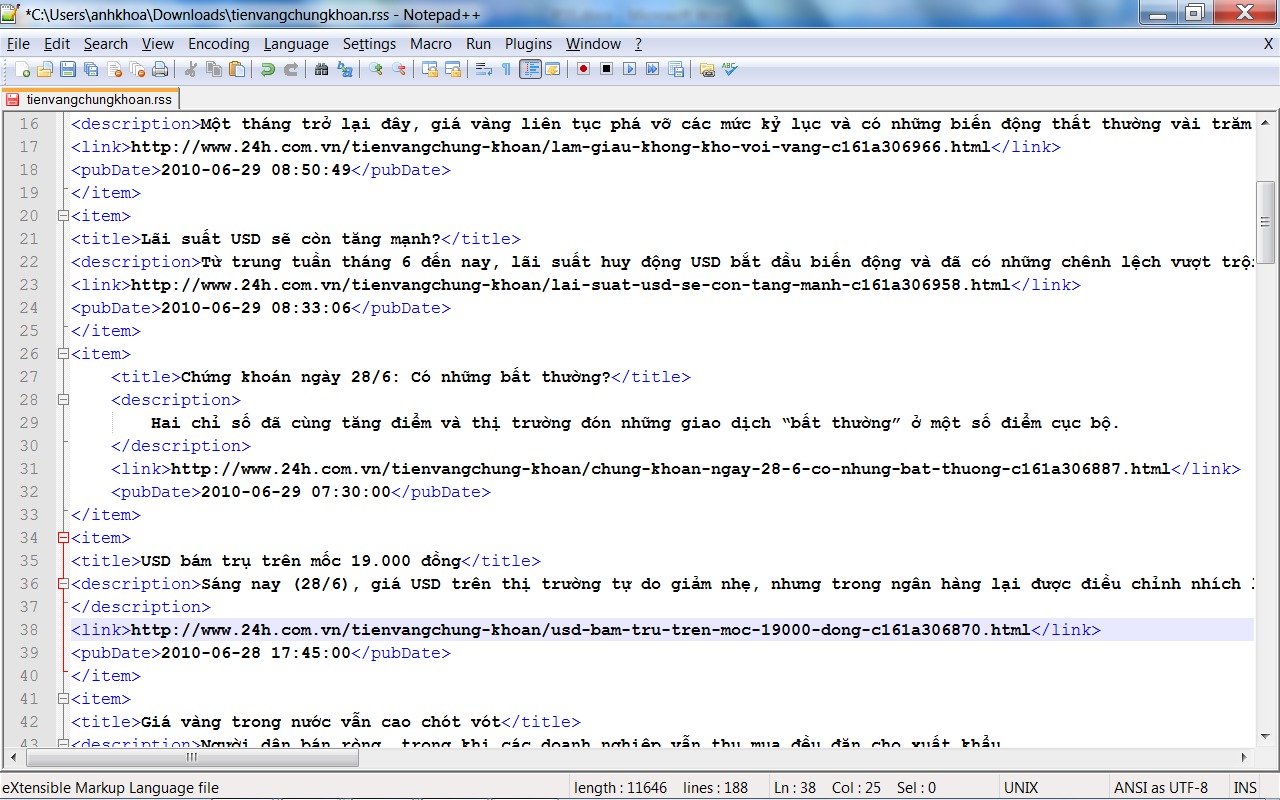
Là một định dạng tập tin thuộc họ XML dùng để chia sẻ tin tức Web được dùng bởi nhiều website tin tức và webblog.

Chữ RSS là dùng để chỉ các chuẩn sau:

* Rich Site Summary (RSS 0.91)
* RDF Site Summary (RSS 0.9 and 1.0)
* Really Simple Syndication (RSS 2.0.0)

Một file RSS là một file XML chứa danh sách các item (các bài tin). Mỗi item thường chứa title là tiêu đề của bài tin, description mô tả bài tin, link chứa liên kết đến bài tin gốc, pubDate ngày xuất bản bài tin. Nó còn gọi là RSS feeds.

Vd: 1 item của web 24h.com



Các công cụ đọc tin phía client thường được xây dựng thành một chương trình đọc lập hoặc một phần mở rộng của 1 chương trình lớn. Các chương trình đọc tin dạng này cũng có mặt trên nhiều hệ điều hành khác nhau và có cả trên các thiết bị di động.

Trên các trang web, RSS feeds thường được liên kết bằng một hình chữ nhật màu cam [Feed-icon.svg](http://vi.wikipedia.org/w/index.php?title=T%E1%BA%ADp_tin:Feed-icon.svg&page=1), có thể kèm theo các kí tự XML [XML.gif](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:XML.gif)hay RSS [RSS.gif](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:RSS.gif).

Vd: Trên trang 24h.com.vn



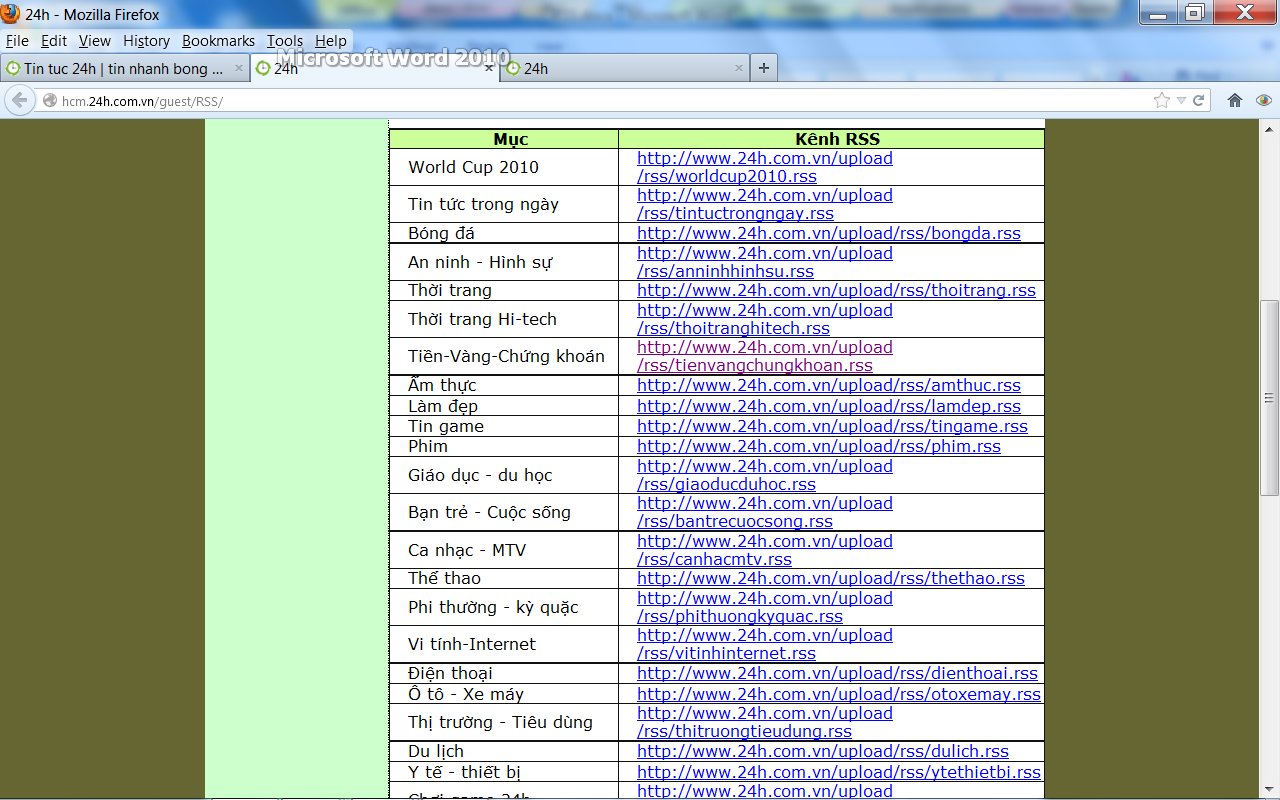
Từ định dạng RSS ta có thể tạo ra phần mềm android đọc tin tức từ các báo.

Thực hành

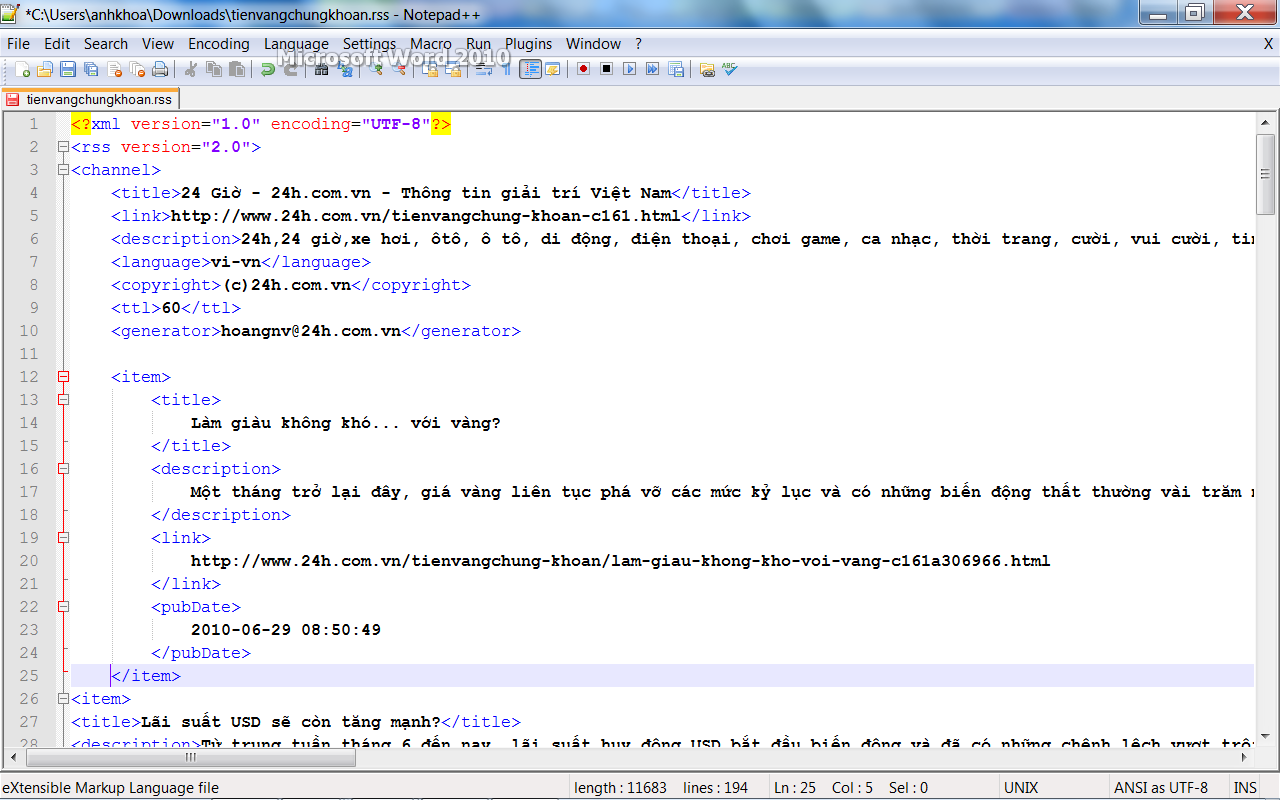
# Bài 1 Viết chương trình đọc báo từ trang 24h.com.vn

## Khảo xát cấu trúc RSS của 24h.com.vn

1. Truy cập website <http://www.24h.com.vn> rồi nhấn vào biểu tượng RSS trên giao diện chính.

1. Phân tích cấu trúc của file RSS. Chú ý thẻ title có đến 2 loại khác nhau.



## Tạo project, activity hiển thị loại tin.

1. Tạo project tên vd: “rss24hdemo1”.
2. 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. Trong giao diện chính kéo vào 1 listview (dùng để hiển thị các category của 24h). Mở file java chính khai báo và ánh xạ listview.
2. Khai báo thêm 2 biến toàn cục.

String [] ten\_loai={"thể thao","vi tinh - internet","dien thoai","du lich"};

String [] rss\_loai={"http://www.24h.com.vn/upload/rss/thethao.rss",

"http://www.24h.com.vn/upload/rss/vitinhinternet.rss",

"http://www.24h.com.vn/upload/rss/dienthoai.rss",

"http://www.24h.com.vn/upload/rss/dulich.rss"

};

1. Trong hàm onCreate tạo ra Adapter và gán dữ liệu từ mảng “ten\_loai” vào.

ArrayAdapter<String> adapter=**new**

ArrayAdapter<String>(**this**,android.R.layout.*simple\_list\_item\_1*,ten\_loai);

lv.setAdapter(adapter);

1. Chạy chương trình để xem kết quả hiển thị trên listview.
2. Bắt sự kiện click vào một mục trên listview.

lv.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2,

**long** arg3) {

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

//arg2 chính là vị trí (index) được nhấn

}

});

## Tạo activity “tintheoloai” để nhận dữ liệu (nhận địa chỉ RSS).

1. Tạo một file layout mới tên “layout\_tintheoloai.xml” có root là LinearLayout. Kéo vào đại 1 TextView.
2. Tạo một file java mới tên “tintheoloai.java”, cho class này kê1 thừa Activity. Override lên hàm onCreate và gán layout cho nó.

**public** **class** tintheoloai **extends** Activity{

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*layout\_tintheoloai*);

}

}

1. Mở Manifest và thêm vào khai báo activity như sau:

<activity

android:name=*".tintheoloai"*

android:label=*"@string/title\_activity\_main"* >

</activity>

1. Tiếp theo ta sẽ gởi dữ liệu từ activity chính qua activity “tintheoloai”. Mở file java của activity chính. Trong sự kiện click vào một item của listview ta viết thêm như sau.

lv.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2,

**long** arg3) {

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

//arg2 chính là vị trí (index) được nhấn

Intent intent=**new** Intent(getApplicationContext(),tintheoloai.**class**);

intent.putExtra("diachi\_rss", rss\_loai[arg2]);

startActivity(intent);

}

});

1. Mở file “tintheoloai.java” ta khai báo thêm 1 biến toàn cục tên “diachi\_rss”. Trong onCreate ta lấy dữ liệu từ intent và Toast ra thử xem đúng chưa.

**public** **class** tintheoloai **extends** Activity{

String diachi\_rss;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*layout\_tintheoloai*);

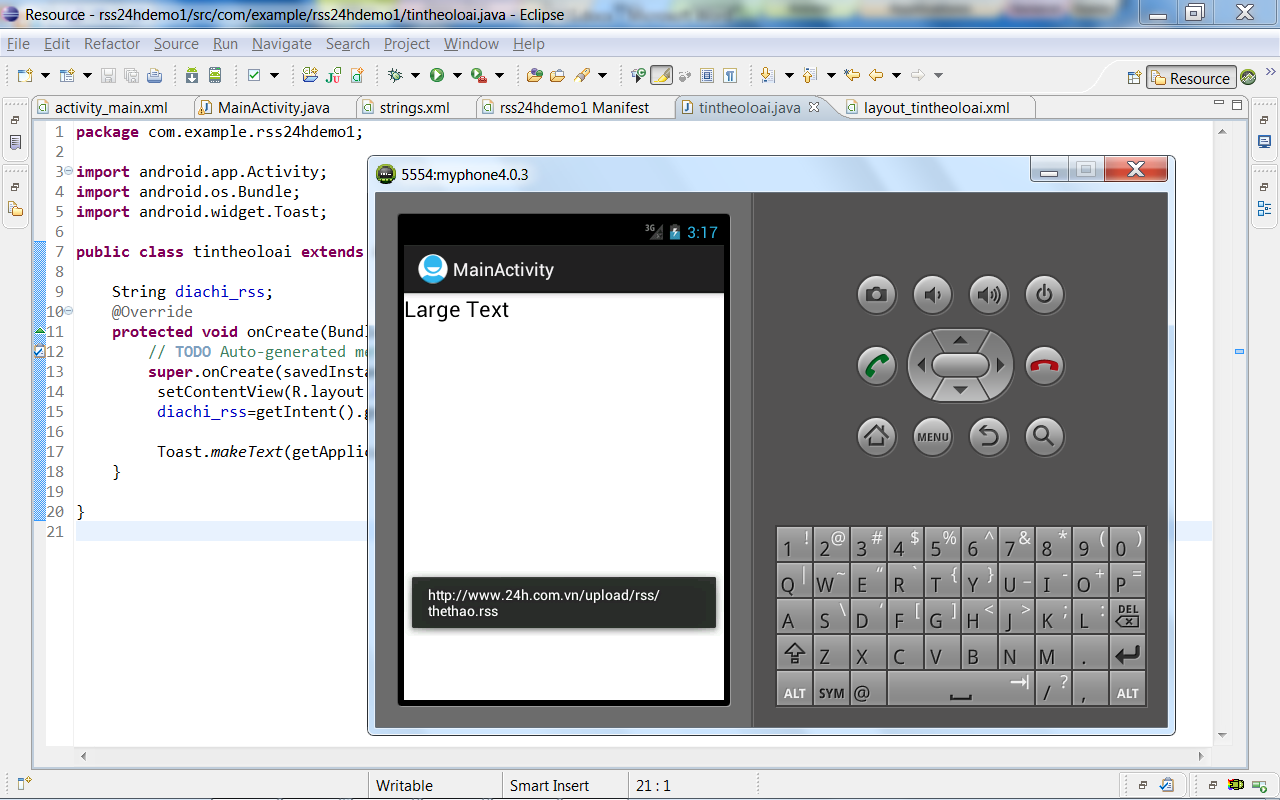
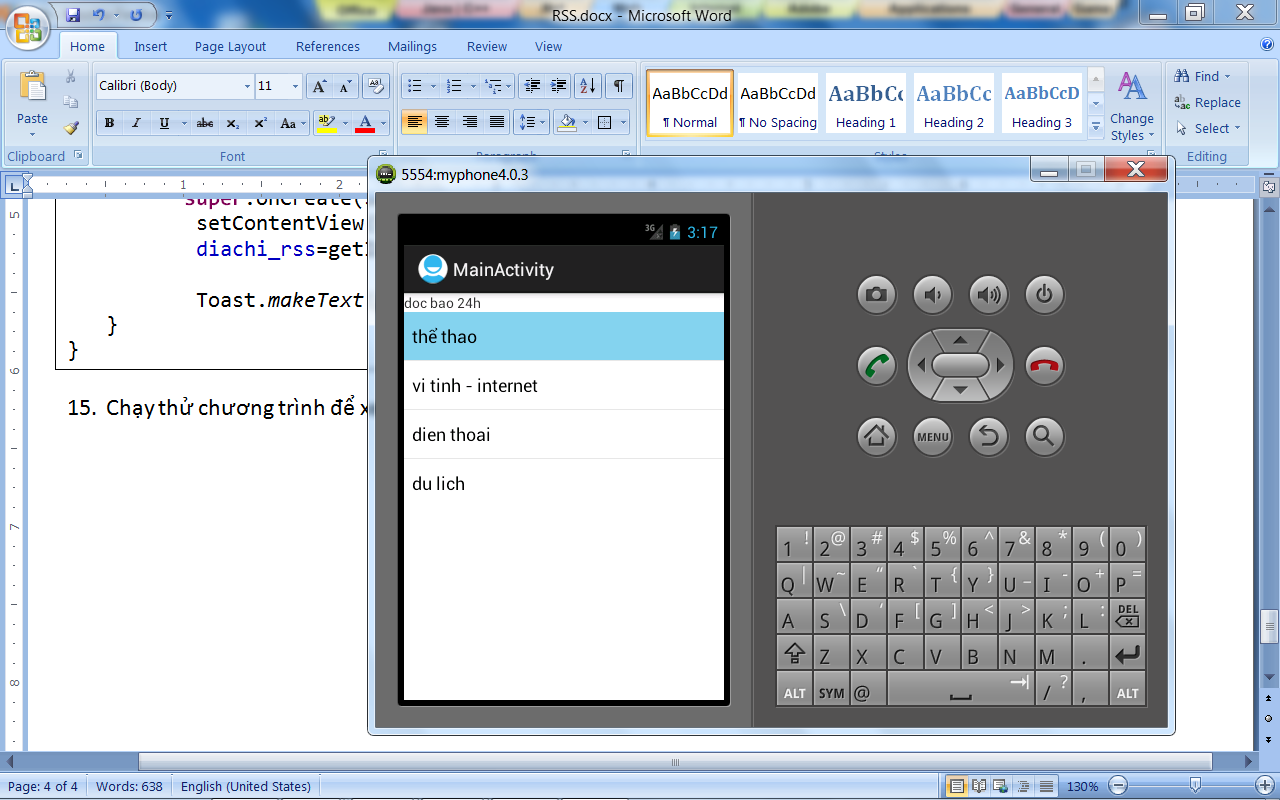
diachi\_rss=getIntent().getExtras().getString("diachi\_rss");

Toast.*makeText*(getApplicationContext(), diachi\_rss, Toast.*LENGTH\_SHORT*).show();

}

}

1. Chạy thử chương trình để xem đúng chưa.



## Xây dựng activity hiển thị danh sách tin theo loại

Ta đã gởi được địa chỉ của rss qua activity “tintheoloai” tiếp theo ta sẽ xây dựng activity “tintheoloai” để hiển thị danh sách tiêu đề các bài tin mới nhất dựa vào rss mà activity đã nhận được. Phần này ta phải xây một số class hỗ trợ gồm:

* Một class tên “Item” giúp tạo ra các đối tượng có thuộc tính giống với các nút trong file RSS.
* Một class tên MySaxHandler kế thừa từ class DefaultHandler để điều khiển việc đọc và phân tích XML.
* Một class tên MySaxParser đảm nhận việc đọc file RSS-xml từ network và đưa vào MySacHandler để phân tích.
* Một class chính (là activity “tintheoloai”) để hiển thị danh sách các tin.

### Tạo class Item

1. Dựa và file xml để xây dựng class. Tạo 1 class tên Item có các thuộc tính giống với xml (dùng private rồi tạo get/set cho chuẩn hơn).

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

**private** String title;

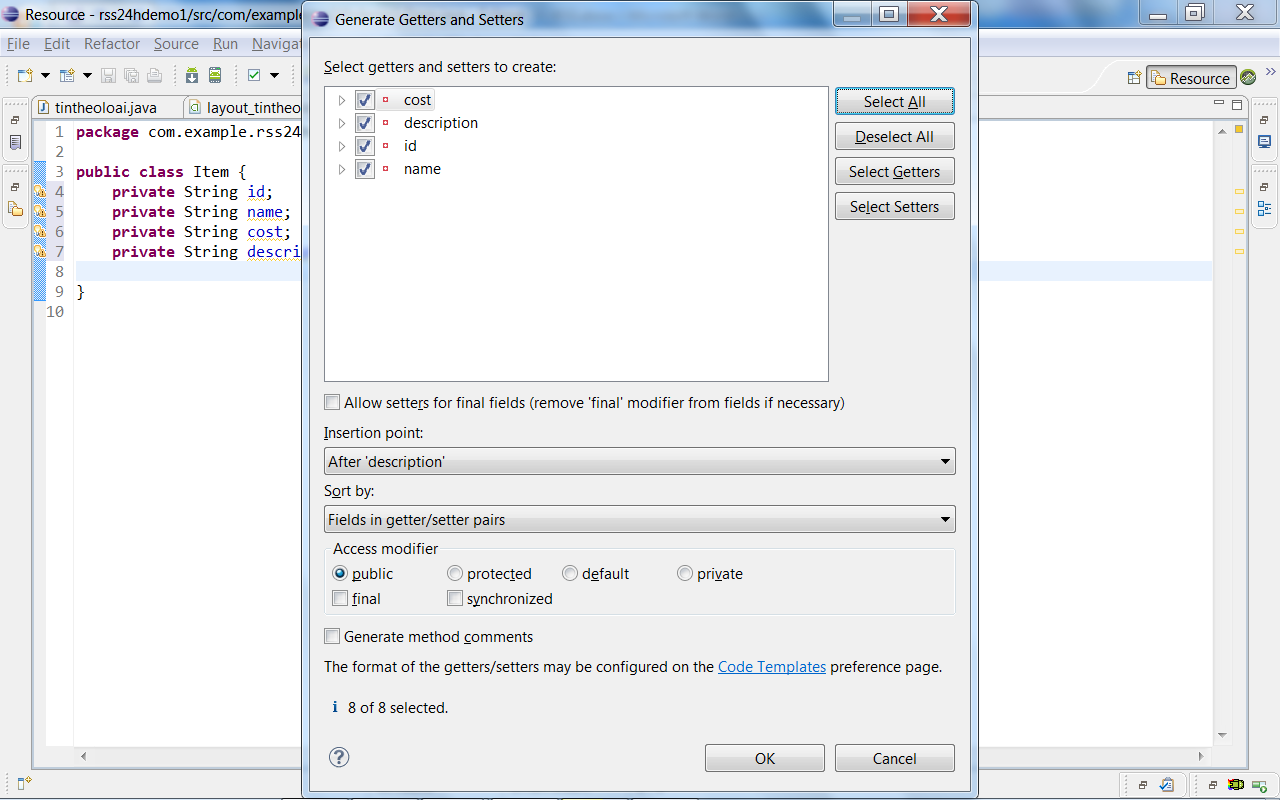
**private** String description;

**private** String link;

**private** String pubdate;

}

1. Tạo get/set cho các thuộc tính. Trong mã lệnh click vào trong class rồi click phải chọn “Source”->”Generate getters and setters”. Hộp thoại mở ra chọn các thuộc tính cần phát sinh get/set rồi ok.



Mã lệnh sẽ phát sinh thêm:

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getLink() {

**return** link;

}

**public** **void** setLink(String link) {

**this**.link = link;

}

**public** String getPubdate() {

**return** pubdate;

}

**public** **void** setPubdate(String pubdate) {

**this**.pubdate = pubdate;

}

### Tạo class MySaxHandler để đọc và phân tích XML

1. Xây dựng một class mới tên MySaxHandler kế thừa từ DefaultHandler và override lên 3 phương thức là charcters, 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. Khai báo các biến toàn cục. items là list chứa các item, chuoi\_tam để đọc dữ liệu trong thẻ XML, item\_tam để khi chạy gán dữ liệu vào nó và khi hoàn tất việc gán sẽ add nó vào items. Có một biến lạ là “vaoitem” được gán true.

ArrayList<Item> items;

Item item\_tam;

String chuoi\_tam;

**boolean** vaoitem=**false**;

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 viết thêm như sau( chú ý dùng equalsIgnoreCase và vao\_item=true):

@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);

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

{

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

vao\_item=**true**;

}

}

1. Trong hàm characters viết thêm như sau:

@Override

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

**throws** SAXException {

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

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

**if**(vao\_item==**true**)

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

}

1. Trong hàm endElement viết thêm như sau:

@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**(vao\_item==**true**)

{

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

item\_tam.setTitle(chuoi\_tam);

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

item\_tam.setDescription(chuoi\_tam);

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

item\_tam.setLink(chuoi\_tam);

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

item\_tam.setPubdate(chuoi\_tam);

}

}

### Xây dựng tiếp activity “tintheoloai” để test.

1. Quay lại file “tintheoloai.java”. Khai báo các biến toàn cục. (chú ý: biến dia\_chi\_rss đã làm rồi và gán dữ liệu cho nó ở trong hàm tạo rồi, biến chuỗi ở đây chỉ dùng để test dữ liệu)

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

InputStream is;

String chuoi="";

String diachi\_rss;

1. Xây 1 class nội tên MyAsyncTask kế thừa từ AsyncTask (chú ý phần for để đưa dữ liệu vào chuỗi và xuất chuỗi ra ở Logcat trong onPostExecute, nó chỉ nhằm mục đích test dữ liệu, ta sẽ bỏ nó ở các bước sau).

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

{

@Override

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

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

**try** {

URL url=**new** URL(diachi\_rss);

URLConnection connection=url.openConnection();

InputStream is=connection.getInputStream();

items=(ArrayList<Item>) MySaxParser.*xmlParser*(is);

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

{

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

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

Log.*d*("dulieu", chuoi);

}

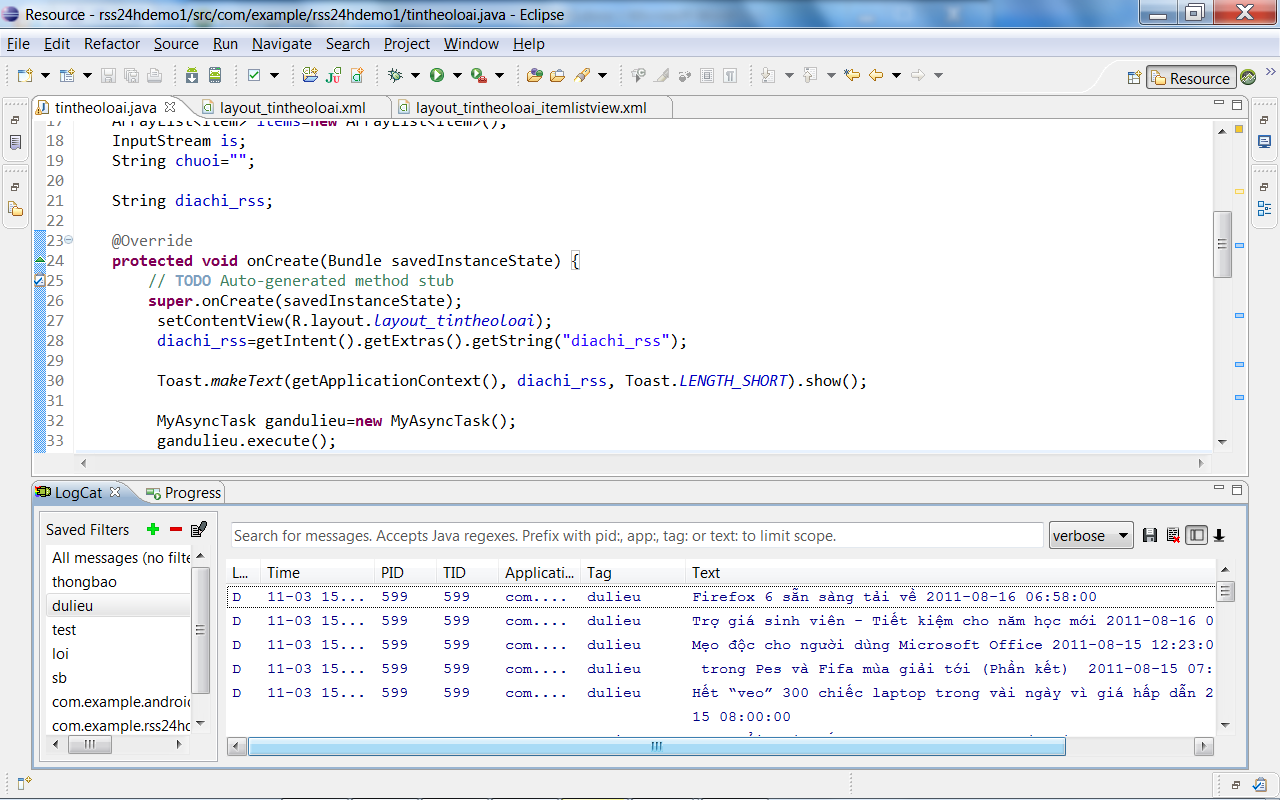
}

1. Trong hàm onCreate viết thêm như sau:

MyAsyncTask gandulieu=**new** MyAsyncTask();

gandulieu.execute();

1. Chạy chương trình, nhìn LogCat để thấy kết quả có lấy ra được hay chưa.



### Tạo ListView để load danh sách các tin theo loai.

1. Sau khi đã test trên logcat và thấy dữ liệu đã lấy được ta bỏ lệnh log.d , bỏ biến “chuoi” và bỏ vòng lặp for trong hàm doInBackGround.
2. Tiếp theo ta sẽ tạo ListView để đưa danh sách vào. Trên file “layout\_tintheoloai.xml” kéo vào 1 ListView. Qua file “tintheoloai.java” khai báo và ánh xạ ListView.
3. Đầu tiên tạo layout cho 1 mục con của ListView. Tạo 1 layout mới đặt tên “layout\_tintheoloai\_itemlistview.xml” trong layout vừa tạo ta tạo ra 3 textView (nhớ đặt lại tên là title, description,pubdate cho dễ):

|  |  |
| --- | --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:orientation=*"vertical"* >  <TextView  android:id=*"@+id/title"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"Medium Text"*  android:textColor=*"#ff0000"*  android:textAppearance=*"?android:attr/textAppearanceMedium"* />  <TextView  android:id=*"@+id/description"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"TextView"*  android:textColor=*"#000000"*  />  <TextView  android:id=*"@+id/pubdate"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"Small Text"*  android:textAppearance=*"?android:attr/textAppearanceSmall"* />  </LinearLayout> |  |

1. Quay lại file java “tintheoloai.java” xây 1 thêm 1 class để tạo adapter như sau:

**class** MyAdapter **extends** ArrayAdapter<Item>{

Context context;

ArrayList<Item> items;

**public** MyAdapter(Context context, **int** textViewResourceId, ArrayList<Item> objects) {

**super**(context, textViewResourceId, objects);

// **TODO** Auto-generated constructor stub

**this**.context=context;

**this**.items=objects;

}

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

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

LayoutInflater inf=(LayoutInflater)

context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);

View rowview=inf.inflate(R.layout.*layout\_tintheoloai\_itemlistview*,parent, **false**);

TextView tv\_title=(TextView)rowview.findViewById(R.id.*title*);

TextView tv\_description=(TextView)rowview.findViewById(R.id.*description*);

TextView tv\_pubdate=(TextView)rowview.findViewById(R.id.*pubdate*);

tv\_title.setText(items.get(position).getTitle().toString());

tv\_description.setText(items.get(position).getDescription().toString());

tv\_pubdate.setText(items.get(position).getPubdate().toString());

**return** rowview;

}

1. Trong hàm onPostExecute viết thêm như sau:

@Override

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

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

**super**.onPostExecute(result);

**try**{

MyAdapter adapter=**new** MyAdapter(getApplicationContext(),R.layout.*layout\_tintheoloai\_itemlistview*,items);

lv.setAdapter(adapter);

}**catch**(Exception e)

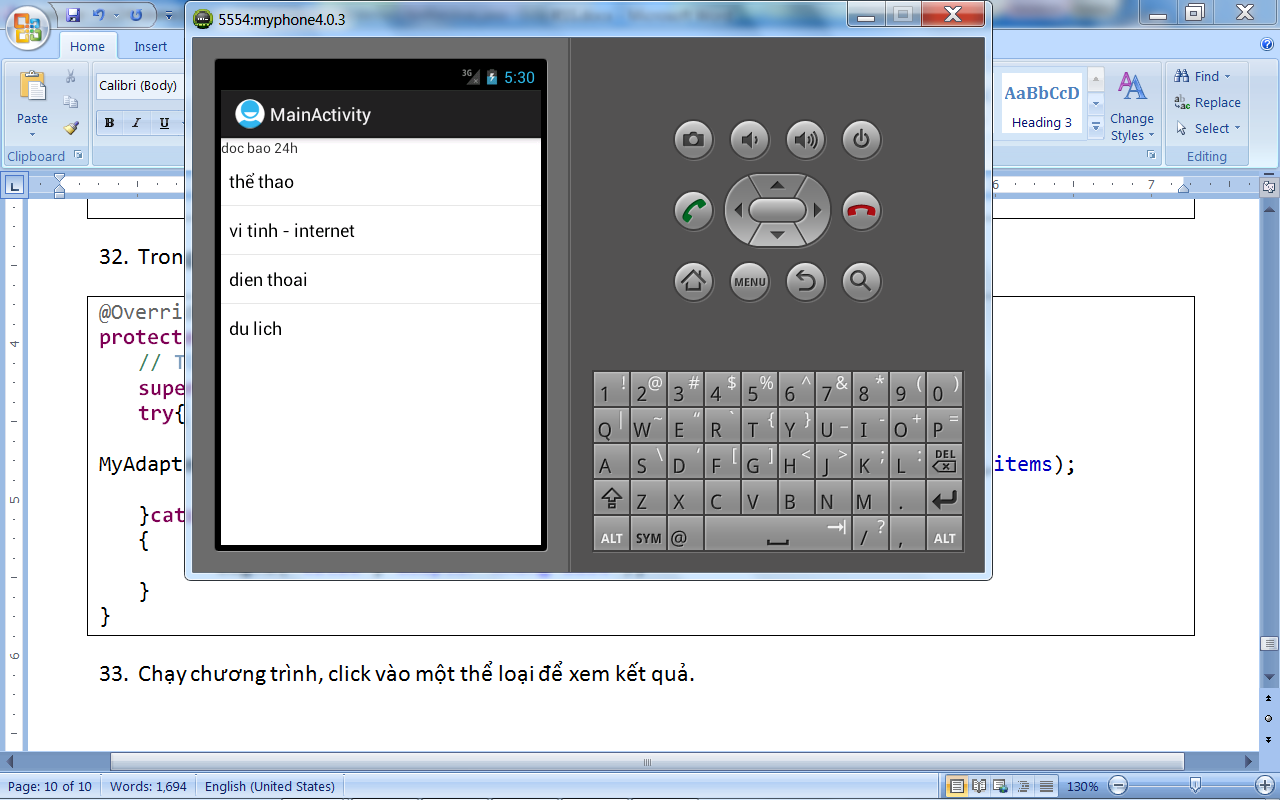
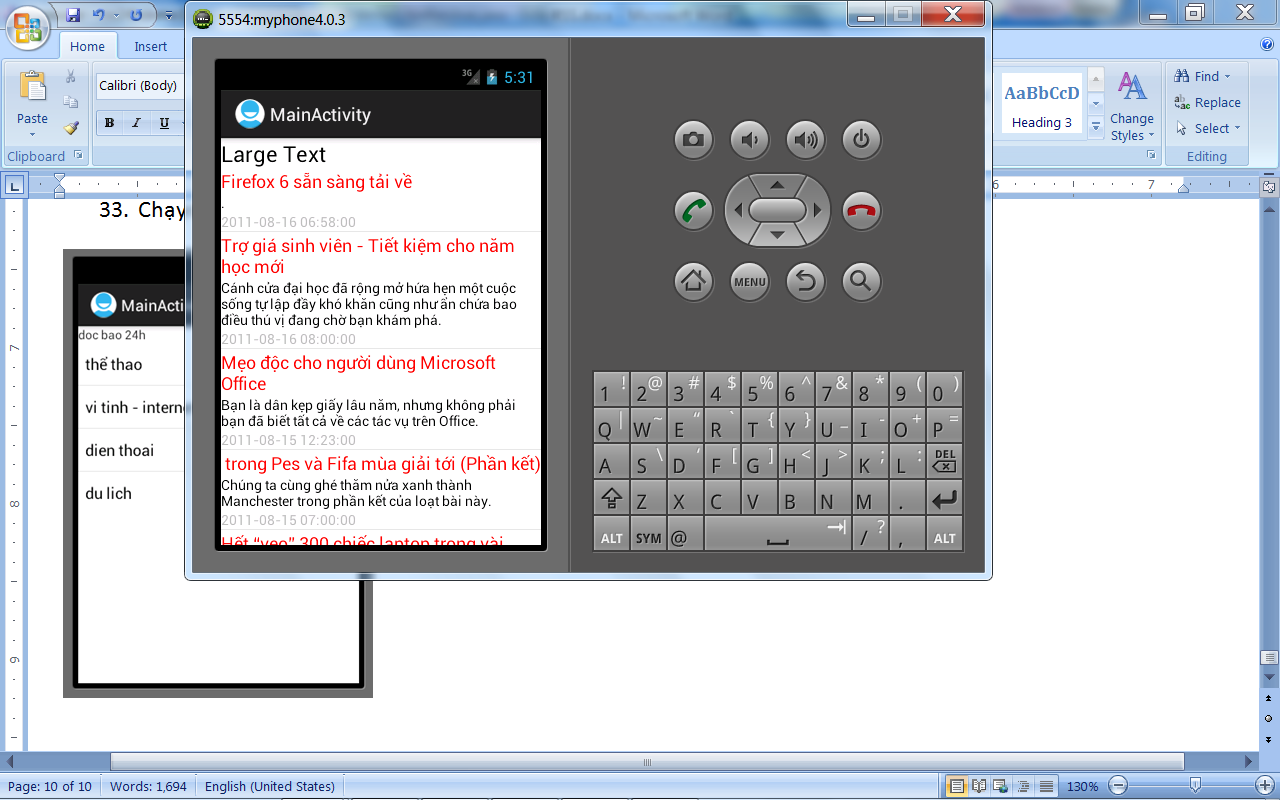
{

Log.*d*("title","adapter khong duoc");

}

}

1. Chạy chương trình, click vào một thể loại để xem kết quả.

### Thêm ProgressDialog

Do khi click vào sẽ load dữ liệu trên mạng nên ta có thể dùng thêm ProgressDialog cho đẹp.

1. Trong Class MyAsyncTask ta thêm vào một ProgressDialog và override lên hàm onPreExecute để show nó như sau (chú ý truyền vào hàm show phải dùng tên Activity hiện tại .this chứ không dùng getApplicationContext được):

ProgressDialog dialog;

@Override

**protected** **void** onPreExecute() {

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

**super**.onPreExecute();

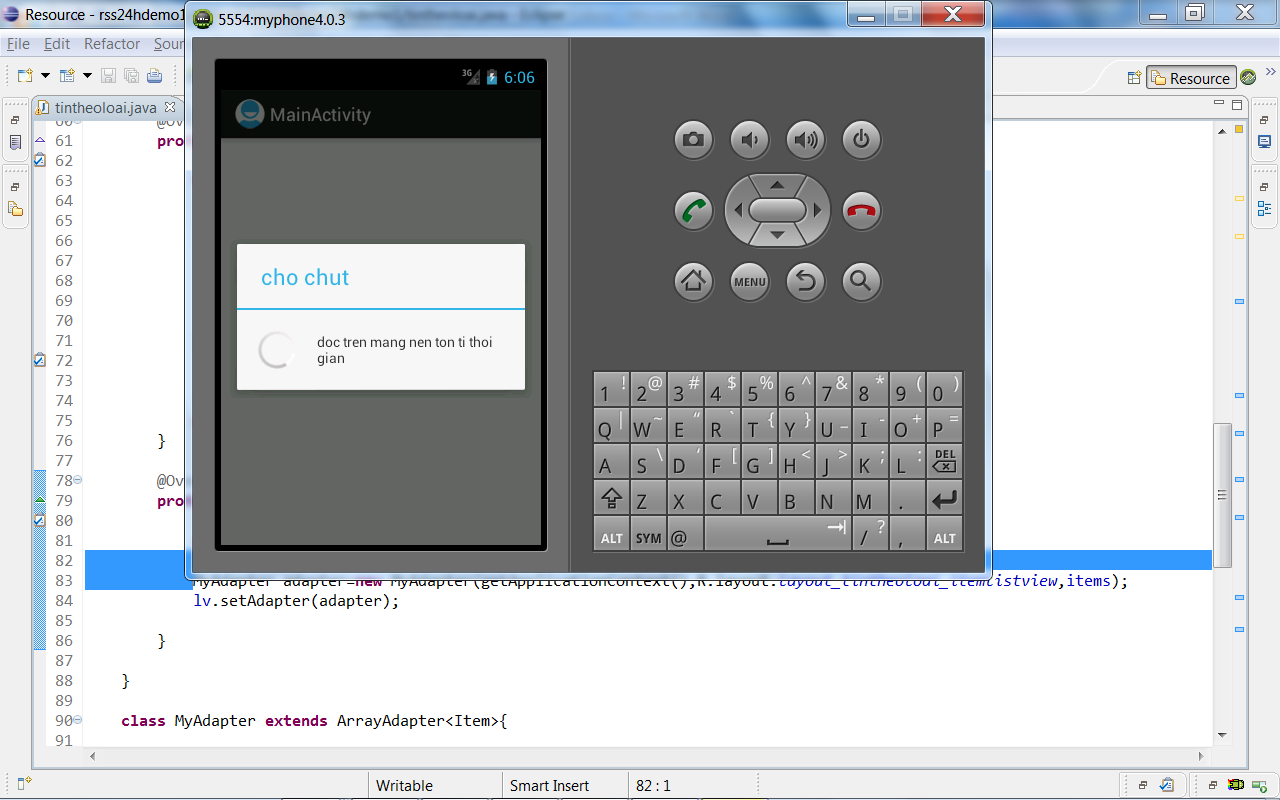
dialog=ProgressDialog.*show*(tintheoloai.**this**, "cho chut", "doc tren mang nen ton ti thoi gian");

}

1. Trong hàm onPostExecute thêm lệnh để tắt dialog

dialog.dismiss();

1. Chạy và test dialog khi click vào một “loaitin”.



## Tạo activity để xem tin

1. Tạo một layout mới tên “layout\_xemtin.xml”. Trong source thêm vào 1 webview như sau:

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<WebView

android:id=*"@+id/webView1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"* />

</LinearLayout>

1. Tạo một file java mới tên “XemTin.java”. Override lên onCreate gọi setContextView để gán layout cho file này là “layout\_xemtin”. Khai báo một biến toàn cục WebView, trong onCreate ánh xạ nó vào.

**public** **class** XemTin **extends** Activity{

WebView wv;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*layout\_xemtin*);

wv=(WebView)findViewById(R.id.*webView1*);

}

}

1. Mở manifest thêm activity như sau:

<activity

android:name=*".XemTin"*

android:label=*"@string/title\_activity\_main"* >

</activity>

1. Quay lại file “tintheoloai.java” để gởi link qua. Mở file tintheoloai.java. Trong hàm tạo bắt thêm sự kiện click cho ListView.

lv.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2,

**long** arg3) {

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

String link=items.get(arg2).getLink();

//Toast.makeText(getApplicationContext(),link, Toast.LENGTH\_SHORT).show();

Intent intent=**new** Intent(getApplicationContext(),XemTin.**class**);

intent.putExtra("link", link);

startActivity(intent);

}

});

1. Quay về file “XemTin.java”. Khai báo các biến toàn cục.

WebView wv;

String link;

ProgressDialog progressbar;

1. Trong hàm onCreate thêm phần để lấy biến link ra rồi gọi hàm loadUrl, đưa vào link là có thể chạy được. Không cần phải tạo thread để xử lý network vì đây là webview.

link=getIntent().getExtras().getString("link");

wv.loadUrl(link);

1. Để thêm phần màu mè nên ta sẽ thêm ProgressBar. Chú ý ở đây ta không dùng ProgressBar như ở phần trước cho WebView. Ta phải dùng thông qua WebViewClient. Ta sẽ thêm phần hiện thị ProgressBar và phần setWebViewClient giữa 2 câu lệnh ở bước 42 cho hợp lý thôi (thêm sau vẫn được).

link=getIntent().getExtras().getString("link");

progressbar=ProgressDialog.*show*(**this**, "cho chut", "loading.....");

wv.setWebViewClient(**new** WebViewClient(){

@Override

**public** **void** onPageFinished(WebView view, String url) {

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

**super**.onPageFinished(view, url);

**if**(progressbar.isShowing())

progressbar.dismiss();

}

@Override

**public** **boolean** shouldOverrideUrlLoading(WebView view, String url) {

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

wv.loadUrl(url);

**return** **super**.shouldOverrideUrlLoading(view, url);

}

});

wv.loadUrl(link);

1. Xem thêm một số thuộc tính của WebView để canh chỉnh cho đẹp hơn.
2. Chạy chương trình để xem kết quả.

