

PDM Avanzado Procesar arquivos XML

Sumario

- 1 Introducción
- 2 Procesando arquivos XML
- 3 Caso práctico
 - ◆ 3.1 Preparación
 - ◆ 3.2 Creamos a activity

Introdución

O XML é unha linguaxe de marcas en formato de texto moi útil para o intercambio de información entre diferentes plataformas.

- Máis información: http://es.wikipedia.org/wiki/Extensible_Markup_Language
- Información para aprender a facer arquivos xml: <http://www.w3schools.com/xml/>

Un exemplo de arquivo XML pode ser o seguinte:

```
<contactos>
  <contacto>
    <nome>Angel</nome>
    <dir>C/ de Ala nº 12</dir>
    <tel tipo="fixo">981111111</tel>
    <tel tipo="mobil">691111111</tel>
  </contacto>
  <contacto>
    <nome>Luis</nome>
    <dir>C/ de Acola nº 55</dir>
    <tel tipo="fixo">982222222</tel>
    <tel tipo="mobil">692222222</tel>
  </contacto>
</contactos>
```

Nesta unidade imos aprender como podemos ler o contido do arquivo e trasladar os datos ó noso programa (poden ser elementos gráficos coma listas ou bases de datos, como xa vimos).

Procesando arquivos XML

Para ler un arquivo XML temos dúas aproximacións:

- DOM: Document Object Model. Ten coma principal desvantaxe que é necesario gardar todo o documento en memoria e coma principal vantaxe que podemos acceder a calquera parte do documento indo cara adiante ou cara atrás.
- SAX, que ven a ser usar XmlPullParser en Android: non podemos movernos pola estrutura xml e só gardamos en memoria o que imos lendo.

Os pasos que temos que dar para ler e procesar un arquivo XML son:

- **Paso 1:** Crear un obxecto da clase `XmlPullParser` e asinarlle o `InputStream` do arquivo que queremos ler.

```
InputStream is = getAssets().open("ArquivoXML.xml");

XmlPullParser parser = Xml.newPullParser();
parser.setInput(is, "UTF-8");
```

Xa vimos nas unidades anteriores como obter un `InputStream`.

O método `setInput` espera recibir o `InputStream` como primeiro parámetro e o xogo de caracteres como segundo. Pode producir unha excepción '`XmlPullParserException`' polo que a engadimos as excepcións que pode producir a execución do método (throws ou try).

- **Paso 2:** Agora necesitamos avanzar polo 'árbol' xml. Neste punto podemos utilizar os seguintes métodos:

◊ Chamando ó método **next()**. Cada vez que chamamos a dito método avanzamos de elemento e ó mesmo tempo 'producimos' eventos como son: principio de documento, fin de documento, apertura de etiqueta, pecha de etiqueta,....

Aviso: Poderíamos ler todo o documento utilizando este método, pero hai que ter coidado xa que o documento xml non debería ter saltos de liña (como está no exemplo) xa que eses caracteres tamén son lidos.

Para saber o evento que se produce facemos o seguinte:

```
int evento = parser.next();
```

Se queremos percorrer todo a árbore ata chegar ó fin do documento:

```
int evento = parser.next();
while(evento != XmlPullParser.END_DOCUMENT) {

    evento = parser.next();

}
```

◊ Chamando ó método **nextTag()**: Neste caso chama a next() e devolve un evento se estamos situados nun START_TAG ou END_TAG. En caso contrario devolve unha excepción.

◊ Chamando ó método **nextText()**: Se estamos situados nun START_TAG, ó chamar a dito método devolve a cadea (o texto) que se atopa entre o START_TAG e o END_TAG.

• **Paso 3:** Neste intre, dependendo da flexibilidade que queiramos ter á hora de ler o arquivo, faremos o programa máis ou menos complexo.

A idea e ir lendo pola árbore e analizando cando chegamos a un principio de etiqueta (por exemplo <nome>) Nese intre temos que avanzar na árbore e obter o texto (Angel).

A variable 'evento' pode valer (entre outros valores):

◊ XmlPullParser.START_TAG: Comezo de etiqueta.

◊ XmlPullParser.END_TAG: Fin de etiqueta.

◊ XmlPullParser.TEXT: Texto

◊ XmlPullParser.END_DOCUMENT: Fin de documento.

No caso que nos ocupa, se queremos ler esta parte do documento xml:

```
<contacto>
  <nome>Angel</nome>
  <dir>C/ de Ala nº 12</dir>
  <tel tipo="fixo">981111111</tel>
  <tel tipo="mobil">691111111</tel>
</contacto>
```

Unha posible forma de facelo sería:

```
if (parser.getName().equals("contacto")) { // Un novo contacto
    evento = parser.nextTag(); // Pasamos a <nome>
    Log.i("XML", "NOME:" + parser.nextText());
    evento = parser.nextTag(); // Pasamos a <dir>
    Log.i("XML", "DIR:" + parser.nextText());
    evento = parser.nextTag(); // Pasamos a <tel>
    Log.i("XML", "Tel fixo:" + parser.nextText());
    evento = parser.nextTag(); // Pasamos a <tel>
    Log.i("XML", "Tel móbil:" + parser.nextText());

}
```

Fixarse como estamos a ler un único contacto. Si isto o poñemos dentro do while indicado anteriormente, no que lemos o documento ata que remate, leríamos todos dando como resultado:

```

12-11 18:54:59.481: I/XML(14112): NOME:Angel
12-11 18:54:59.481: I/XML(14112): DIR:C/ de Ala nº 12
12-11 18:54:59.481: I/XML(14112): Tel fixo:981111111
12-11 18:54:59.481: I/XML(14112): Tel mobil:691111111
12-11 18:54:59.481: I/XML(14112): NOME:Luis
12-11 18:54:59.481: I/XML(14112): DIR:C/ de Acola nº 55
12-11 18:54:59.491: I/XML(14112): Tel fixo:982222222
12-11 18:54:59.491: I/XML(14112): Tel mobil:692222222

```

Aquí entra en xogo o comentado anteriormente da flexibilidade. Esta forma de ler o documento funciona sempre que enviemos os datos de nome, dir, tel1, tel2, pero, que pasa se un ven sen teléfono móbil ? Ó analizar o arquivo xml daría un erro ou traeríamos datos erróneos.

Polo tanto teremos que ter en conta as posibilidades que poda traer o arquivo xml para realizar unha análise máis completa e complexa.

Se queremos obter os atributos dentro dunha etiqueta teremos que facer uso do métodos:

- ◊ `getAttributeCount`: Devolve o número de atributos
- ◊ `getAttributeName(int count)`: Accede ó nome do atributo indicado por count.
- ◊ `getAttributeValue(int count)`: Obtemos o valor do atributo indicando por count.

No exemplo anterior, temos un atributo 'tipo' na liña do teléfono:

```

<tel tipo="fixo">981111111</tel>
<tel tipo="mobil">691111111</tel>

```

Unha forma de obter o valor deses atributos:

```

if (evento == XmlPullParser.START_TAG) {
    if (parser.getName().equals("contacto")) { // Un novo contacto
        evento = parser.nextTag(); // Pasamos a <nome>
        Log.i("XML", "NOME:" + parser.nextText());
        evento = parser.nextTag(); // Pasamos a <dir>
        Log.i("XML", "DIR:" + parser.nextText());
        evento = parser.nextTag(); // Pasamos a <tel>
        // Estamos no tag <tel>=> obtemos o nome e valor do atributo tipo
        Log.i("XML", "Tel:" + parser.getAttributeName(0) + "-->" + parser.getAttributeValue(0) + ":" + parser.nextText());
        evento = parser.nextTag(); // Pasamos a <tel móbil>
        Log.i("XML", "Tel:" + parser.getAttributeName(0) + "-->" + parser.getAttributeValue(0) + ":" + parser.nextText());
    }
}

```

Dando como resultado:

```

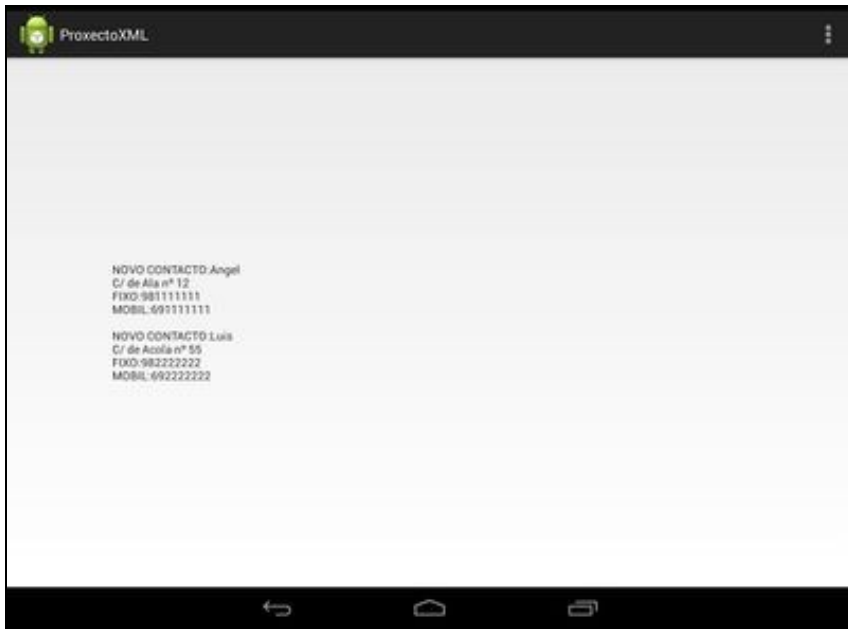
12-11 18:57:26.641: I/XML(14198): NOME:Angel
12-11 18:57:26.641: I/XML(14198): DIR:C/ de Ala nº 12
12-11 18:57:26.641: I/XML(14198): Tel:tipo-->fixo:981111111
12-11 18:57:26.641: I/XML(14198): Tel:tipo-->mobil:691111111
12-11 18:57:26.641: I/XML(14198): NOME:Luis
12-11 18:57:26.641: I/XML(14198): DIR:C/ de Acola nº 55
12-11 18:57:26.641: I/XML(14198): Tel:tipo-->fixo:982222222
12-11 18:57:26.641: I/XML(14198): Tel:tipo-->mobil:692222222

```

Caso práctico

O obxectivo desta práctica é ver un exemplo de procesamento dun arquivo XML.

O arquivo terá de contido unha lista de contactos como vimos ó principio desta unidade. O que fará o activity será ler ese contido, pasar os datos lidos a obxectos dunha clase Contactos e amosará o contido do array de Contactos creados nun TextView.



Por que introducimos o modelo de clases neste apartado ?

Lembrar que isto xa o usamos na [unidade de bases de datos](#).

Normalmente os datos lidos dende un arquivo XML os imos querer gardar nunha base de datos. Como xa vimos como facelo utilizando clases que modelizan os datos, por iso o facemos así.

Preparación

- **Media:Ud4_01_arquivo.xml**: Gardade este arquivo (todo o nome con letras minúsculas) no cartafol /assets/ do voso proxecto Android.

Creamos a activity

- Nome do proxecto: **UD4_01_XML**
- Nome da activity: **UD4_01_XML.java**

Código do layout xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".UD4_01_XML" >

    <TextView
        android:id="@+id/UD4_01_txtDatos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        />

</RelativeLayout>
```

Código da classe UD4_01_XML

Obxectivo: Ler un arquivo XML e procesalo.

```
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

import android.app.Activity;
import android.os.Bundle;
import android.util.Xml;
import android.widget.TextView;
import android.widget.Toast;

public class UD4_01_XML extends Activity {

    private ArrayList<Contacto>contactos = new ArrayList<Contacto>();

    private void lerArquivo() throws IOException, XmlPullParserException {

        InputStream is = getAssets().open("ud4_01_arquivo.xml");

        XmlPullParser parser = Xml.newPullParser();
        parser.setInput(is, "UTF-8");

        int evento = parser.nextTag();
        Contacto contacto = null;

        while(evento != XmlPullParser.END_DOCUMENT) {
            if(evento == XmlPullParser.START_TAG) {
                if (parser.getName().equals("contacto")) { // Un novo contacto
                    contacto = new Contacto();
                    evento = parser.nextTag(); // Pasamos a <nome>
                    contacto.setNome(parser.nextText());
                    evento = parser.nextTag(); // Pasamos a <dir>
                    contacto.setDireccion(parser.nextText());
                    evento = parser.nextTag(); // Pasamos a <tel>
                    contacto.setTelefono_fijo(parser.nextText());
                    evento = parser.nextTag(); // Pasamos a <tel móvil>
                    contacto.setTelefono_mobil(parser.nextText());
                }
                if(evento == XmlPullParser.END_TAG) {
                    if (parser.getName().equals("contacto")) { // Un novo contacto
                        contactos.add(contacto);
                    }
                }

                evento = parser.next();
            }
        }

        is.close();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ud4_01__xml);

        try {
            lerArquivo();

            TextView caixa = (TextView)findViewById(R.id.UD4_01_txtDatos);
            for (Contacto contacto : contactos){

                caixa.append("\nNOVO CONTACTO:");
                caixa.append(contacto.getNome()+"\n");
                caixa.append(contacto.getDireccion()+"\n");
            }
        }
    }
}
```

```

caixa.append("FIXO:" + contacto.getTelefono_fijo()+"\n");
caixa.append("MOBIL:" + contacto.getTelefono_mobil()+"\n");
}
} catch (IOException e) {
    // TODO Auto-generated catch block
e.printStackTrace();
Toast.makeText(this, "ERRO:" + e.getMessage(), Toast.LENGTH_LONG).show();
} catch (XmlPullParserException e) {
    // TODO Auto-generated catch block
e.printStackTrace();
Toast.makeText(this, "ERRO:" + e.getMessage(), Toast.LENGTH_LONG).show();
}
}

}

```

- Liña 16: Definimos o array que vai gardar os contactos do arquivo XML.
- Liñas 19-54: Método que vai ler o arquivo XML e gardara no array anterior a lista de contactos.

◊ Liña 32: Comprobamos se o inicio de etiqueta é <contacto>. Nese intre se crea un novo obxecto Contacto para engadir ó array.
 ◊ Liñas 33-40: Lemos o contido do xml e imos enchendo o obxecto da clase Contacto creado previamente.
 ◊ Liñas 44-45: Se atopamos o fin de etiqueta </contacto> engadimos o novo contacto ó array.

- Liña 61: Chamamos ó método que le o arquivo.
- Liñas 63-70: Percorremos o array de contactos e imos engadindo ó TextView os datos.

- Creamos a clase que vai modelizar os Contactos:

Código da clase Contacto

```

public class Contacto {
    private String nome;
    private String direccion;
    private String telefono_fijo;
    private String telefono_mobil;

    public Contacto(String nome, String direccion, String tel_fijo, String tel_mobil){
        this.nome=nome;
        this.direccion=direccion;
        this.telefono_fijo=tel_fijo;
        this.telefono_mobil=tel_mobil;
    }

    public Contacto(){

    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public String getTelefono_fijo() {
        return telefono_fijo;
    }
}

```

```
public void setTelefono_fixo(String telefono_fixo) {  
    this.telefono_fixo = telefono_fixo;  
}  
  
public String getTelefono_mobil() {  
    return telefono_mobil;  
}  
  
public void setTelefono_mobil(String telefono_mobil) {  
    this.telefono_mobil = telefono_mobil;  
}  
  
}
```

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2014).