

7.2.1 Android XML数据解析

分类 [Android 基础入门教程](#)

Android 基础入门教程(Q群号 : 153836263)

本节引言：

前面两节我们对Android内置的Http请求方式：

HttpURLConnection和HttpClient，本来以为OkHttp 已经集成进来了，然后想讲解下Okhttp的基本用法，后来发现还是要靠第三方，算了，放到进阶部分吧，而本节我们来学习下Android为我们提供的三种解析XML数据的方案！他们分别是：SAX,DOM,PULL三种解析方式，下面我们就来对他们进行学习！

1.XML数据要点介绍

首先我们来看看XML数据的一些要求以及概念：

XML格式数据的简单理解

全名叫做可扩展标记语言,类似于HTML(超文本标记),这里不深究他的定义,只介绍他是用来做什么的!其实我们更多的时候是把xml用来存储数据,可以看作一个微型的数据库,比如我们前面学的SharedPreference就是使用xml文件来保存配置信息的,而我们的SQLite其实底层也是一个xml文件;而在网络应用方面通常作为信息的载体,我们通常把数据包装成xml来传递

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
  <person id = "11">
    <name>Coder-pig</name>
    <age>20</age>
  </person>
  <person id = "13">
    <name>Jay</name>
    <age>20</age>
  </person>
</persons>
```

对应
结构

文档开始
开始元素(persons)
文本结点(空白文本) 开始元素(person) 属性
文本结点(空白文本) 开始元素(name) 文本结点 结束元素
...
结束元素(persons)
文档结束

ps:上面就是简单的定义一个存储person对象的xml文件的编码;要注意一点哦,外面的空白区域也是文本结点哦!

2.三种解析XML方法的比较

- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
 - 1.1 背景相关与系统架构分析
 - 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
 - 1.3 SDK更新不了问题解决
 - 1.4 Genymotion模拟器安装
 - 1.5.1 Git使用教程之本地仓...
 - 1.5.2 Git之使用GitHub搭建...
 - 1.6 .9(九妹)图片怎么玩
 - 1.7 界面原型设计
 - 1.8 工程相关解析(各种文件...
 - 1.9 Android程序签名打包
 - 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
 - 2.3.1 TextView(文本框)详解
 - 2.3.2 EditText(输入框)详解
 - 2.3.3 Button(按钮)与ImageB...
 - 2.3.4 ImageView(图像视图)
 - 2.3.5.RadioButton(单选按钮...
 - 2.3.6 开关按钮ToggleButton...
 - 2.3.7 ProgressBar(进度条)
 - 2.3.8 SeekBar(拖动条)
 - 2.3.9 RatingBar(星级评分条)
 - 2.4.1 ScrollView(滚动条)

SAX, DOM, PULL解析xml的比较

SAX解析XML:

对文档进行顺序扫描,当扫描到文档(document)开始与结束、元素(element)开始与结束、文档(document)结束等地方时通知事件处理函数,由事件处理函数做相应动作,然后继续同样的扫描,直至文档结束。解析速度快,占用的内存也少。每需要解析一类xml,就需要编写新的适合该类XML的处理类,用起来还是有点麻烦的!采用的是流式解析,解析是同步的:读到哪就处理哪!

Dom解析XML:

先把xml文档都读到内存中,然后再用DOM API来访问树形结构,并获取数据。这个写起来很简单,但是很消耗内存,加入读取的数据量大,手机内存不够的话,可能导致手机死机!不建议在Android设备中使用。解析简单的xml文件倒可以!常用的五个接口与类:Document, Element, Node, NodeList, DOMParser。Dom是整个文件解析到内存中,供用户需要的结点信息,支持随机访问。

pull解析XML:

XML pull提供了开始元素和结束元素。当某个元素开始时,我们可以调用parser . nextText()从XML文档中读取所有字符数据。当解析到一个文档结束时,自动生成EndDocument。常用接口和类:XmlPullParser, XmlSerializer, XmlPullParserFactory。和SAX差不多,代码没那么复杂,实现较为简单,非常适合移动设备,Android系统内置pull解析器,而且Android系统内部默认使用pull来解析xml文件,如果需要在J2EE或者其他使用pull需要导入两个包,后面给出下载。

3.SAX解析XML数据

SAX解析xml文件

SAX是一个解析速度快且占用内存少的xml解析器,非常适合用于Android等移动设备;SAX解析xml文件采用的是事件驱动;也就是不需要解析整个文档,而是在解析文档的过程中,判断读到的字符是否符合xml语法的某部分(文件开头,文档结束,或者标签开头与标签结束时)符合的话就会触发事件(回调方法)而这些方法都定义在ContentHandler接口中,而ContentHandler是一个接口,使用起来有些不方便,而Android很贴心地为我们提供了一个帮助类:DefaultHandler,只需要继承这个类,重写对应的方法即可。

可供重写的方法如下:

startDocument():	当读取到文档开始标志时触发,通常在这里完成一些初始化操作
endDocument():	文档结束部分,在这里完成一些善后工作
startElement(names paceURI,localName, qName,atts):	参数依次为命名空间;不带命名空间的前缀标签名;带命名控件前缀名的标签,通过atts可以得到所有的属性名与相应的值;SAX中一个重要的特点就是它的流式处理,当遇到一个标签的时候,它并不会记录下以前所碰到的标签,也就是说,在startElement()方法中,所有你所知道的信息,就是标签的名字和属性,至于标签的嵌套结构,上层标签的名字,是否有子元素等等其它与结构相关的信息,都是不得而知的,都需要你的程序来完成。这使得SAX在编程处理上没有DOM来得那么方便。
endElement(uri,local Name,name):	在遇到结束标签的时候,调用这个方法
characters(ch,start, length):	这个方法用来处理在XML文件中读到的内容,第一个参数用于存放文件的内容,后面两个参数是读到的字符串在这个数组中的起始位置和长度,使用new String(ch,start,length)就可以获取内容。

核心代码:

SAX解析类:SaxHelper.java:

```
/**
 * Created by Jay on 2015/9/8 0008.
 */
public class SaxHelper extends DefaultHandler {
    private Person person;
    private ArrayList<Person> persons;
    //当前解析的元素标签
    private String tagName = null;

    /**
     * 当读取到文档开始标志是触发,通常在这里完成一些初始化操作
     */
    @Override
    public void startDocument() throws SAXException {
```

- 2.4.2 Date & Time组件(上)
- 2.4.3 Date & Time组件(下)
- 2.4.4 Adapter基础讲解
- 2.4.5 ListView简单实用
- 2.4.6 BaseAdapter优化
- 2.4.7ListView的焦点问题
- 2.4.8 ListView之checkbox错...
- 2.4.9 ListView的数据更新问题
- 2.5.0 构建一个可复用的自定...
- 2.5.1 ListView Item多布局的...
- 2.5.2 GridView(网格视图)的...
- 2.5.3 Spinner(列表选项框)...
- 2.5.4 AutoCompleteTextVie...
- 2.5.5 ExpandableListView(...
- 2.5.6 ViewPager(翻转视图)...
- 2.5.7 Toast(吐司)的基本使用
- 2.5.8 Notification(状态栏通...
- 2.5.9 AlertDialog(对话框)详解
- 2.6.0 其他几种常用对话框基...
- 2.6.1 PopupWindow(悬浮框...
- 2.6.2 菜单(Menu)
- 2.6.3 ViewPager的简单使用
- 2.6.4 DrawerLayout(官方侧...
- 3.1.1 基于监听的事件处理机制
- 3.2 基于回调的事件处理机制
- 3.3 Handler消息传递机制浅析
- 3.4 TouchListener PK OnTo...
- 3.5 监听EditText的内容变化
- 3.6 响应系统设置的事件(Co...
- 3.7 AsyncTask异步任务
- 3.8 Gestures(手势)
- 4.1.1 Activity初学乍练
- 4.1.2 Activity初窥门径
- 4.1.3 Activity登堂入室
- 4.2.1 Service初涉
- 4.2.2 Service进阶
- 4.2.3 Service精通
- 4.3.1 BroadcastReceiver牛...
- 4.3.2 BroadcastReceiver庖...
- 4.4.1 ContentProvider初探

```

        this.persons = new ArrayList<Person>();
        Log.i("SAX", "读取到文档头,开始解析xml");
    }

    /**
     * 读到一个开始标签时调用,第二个参数为标签名,最后一个参数为属性数组
     */
    @Override
    public void startElement(String uri, String localName, String qName,
                            Attributes attributes) throws SAXException {
        if (localName.equals("person")) {
            person = new Person();
            person.setId(Integer.parseInt(attributes.getValue("id")));
            Log.i("SAX", "开始处理person元素~");
        }
        this.tagName = localName;
    }

    /**
     * 读到内容,第一个参数为字符串内容,后面依次为起始位置与长度
     */
    @Override
    public void characters(char[] ch, int start, int length)
        throws SAXException {
        //判断当前标签是否有效
        if (this.tagName != null) {
            String data = new String(ch, start, length);
            //读取标签中的内容
            if (this.tagName.equals("name")) {
                this.person.setName(data);
                Log.i("SAX", "处理name元素内容");
            } else if (this.tagName.equals("age")) {
                this.person.setAge(Integer.parseInt(data));
                Log.i("SAX", "处理age元素内容");
            }
        }
    }

    /**
     * 处理元素结束时触发,这里将对象添加到结合中
     */
    @Override
    public void endElement(String uri, String localName, String qName)
        throws SAXException {

```

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScrip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

```

        if (localName.equals("person")) {
            this.persons.add(person);
            person = null;
            Log.i("SAX", "处理person元素结束~");
        }
        this.tagName = null;
    }

    /**
     * 读取到文档结尾时触发,
     */
    @Override
    public void endDocument() throws SAXException {
        super.endDocument();
        Log.i("SAX", "读取到文档尾,xml解析结束");
    }

    //获取persons集合
    public ArrayList<Person> getPersons() {
        return persons;
    }
}

```

然后我们在MainActivity.java中写上写上这样一个方法，然后要解析XML的时候调用下 就好了~

```

private ArrayList<Person> readxmlForSAX() throws Exception {
    //获取文件资源建立输入流对象
    InputStream is = getAssets().open("person1.xml");
    //①创建XML解析处理器
    SaxHelper ss = new SaxHelper();
    //②得到SAX解析工厂
    SAXParserFactory factory = SAXParserFactory.newInstance();
    //③创建SAX解析器
    SAXParser parser = factory.newSAXParser();
    //④将xml解析处理器分配给解析器,对文档进行解析,将事件发送给处理器
    parser.parse(is, ss);
    is.close();
    return ss.getPersons();
}

```

一些其他的话：

嗯，对了，忘记给大家说下我们是定义下面这样一个person1.xml文件，然后放到assets目录下的！文件内容如下：**person1.xml**

```

<?xml version="1.0" encoding="UTF-8"?>
<persons>
    <person id = "11">
        <name>SAX解析</name>
    </person>
</persons>

```

8.3.3 Paint API之—— Mask...

8.3.4 Paint API之—— Xferm...

8.3.5 Paint API之—— Xferm...

8.3.6 Paint API之—— Xferm...

8.3.7 Paint API之—— Xferm...

8.3.8 Paint API之—— Xferm...

8.3.9 Paint API之—— Color...

8.3.10 Paint API之—— Colo...

8.3.11 Paint API之—— Colo...

8.3.12 Paint API之—— Path...

8.3.13 Paint API之—— Sha...

8.3.14 Paint几个枚举/常量值...

8.3.15 Paint API之——Type...

8.3.16 Canvas API详解(Part 1)

8.3.17 Canvas API详解(Part...

8.3.18 Canvas API详解(Part...

8.4.1 Android动画合集之帧...

8.4.2 Android动画合集之补...

8.4.3 Android动画合集之属...

8.4.4 Android动画合集之属...

9.1 使用SoundPool播放音...

9.2 MediaPlayer播放音频与...

9.3 使用Camera拍照

9.4 使用MediaRecord录音

10.1 TelephonyManager(电...

10.2 SmsManager(短信管理...

10.3 AudioManager(音频管...

10.4 Vibrator(振动器)

10.5 AlarmManager(闹钟服务)

10.6 PowerManager(电源服...

10.7 WindowManager(窗口...

10.8 LayoutInflater(布局服务)

10.9 WallpaperManager(壁...

10.10 传感器专题(1)——相...

10.11 传感器专题(2)——方...

10.12 传感器专题(3)——加...

10.12 传感器专题(4)——其...

10.14 Android GPS初涉

11.0 《2015最新Android基...


```

        <age>18</age>
    </person>
    <person id = "13">
        <name>XML1</name>
        <age>43</age>
    </person>
</persons>

```

我们是把三种解析方式都糅合到一个demo中，所以最后才贴全部的效果图，这里的话，贴下打印的Log，相信大家会对SAX解析XML流程更加明了：

```

I/SAX : 读取到文档头, 开始解析xml
I/SAX : 开始处理person元素~
I/SAX : 处理name元素内容
I/SAX : 处理age元素内容
I/SAX : 处理person元素结束~
I/SAX : 开始处理person元素~
I/SAX : 处理name元素内容
I/SAX : 处理age元素内容
I/SAX : 处理person元素结束~
I/SAX : 读取到文档尾, xml解析结束

```

另外，外面的空白文本也是文本节点哦！解析的时候也会走这些节点！

4.DOM解析XML数据

Dom解析XML文件

Dom解析XML文件时会把XML文件的所有内容以文档树的形式存储到内存中,然后允许您使用DOM API遍历XML树、检索所需的数据。使用DOM操作XML的代码看起来是比较直观的,并且在编码方面比基于SAX的实现更加简单。但是,因为DOM需要将XML文件的所有内容以文档树方式存放在内存中,所以内存的消耗比较大,特别对于运行Android的移动设备来说,因为设备的资源比较宝贵,所以建议还是采用SAX来解析XML文件,当然,如果XML文件的内容比较小采用DOM也是可行的。

先了解一下Dom的一些api

DocumentBuilderFactory (解析器工厂类)	创建方法: DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder (解析器类)	创建方法: 通过解析器工厂类来获得 DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
Document (文档树模型)	将要解析的xml文件读入Dom解析器 Document doc = dbBuilder.parse(context.getAssets().open("persons2.xml"));
ps:Document对象代表了一个xml文档的模型树,所有的其他Node都以一定的顺序包含在Document对象之内,排列成一个树状结构,以后对XML文档的所有操作都与解析器无关,	
NodeList (结点列表类)	代表一个包含一个或者多个Node的列表,可看作数组,有以下两个方法: item(index):返回集合的第index个Node项 getLength():列表的节点数
Node (结点类)	Dom中最基本的对象,代表文档树中的抽象结点,很少会直接使用的;通常是调用他的子对象Element,Attr,Text等
Element (元素类)	Node最主要的子对象,再元素中可以包含属性,因此有取属性的方法: getAttribute()获得属性值;getTagName():元素的名称;
Attr (属性类)	代表某个元素的属性,虽然Attr继承自Node接口,但因为Attr是包含在Element中的,但不能将其看做是Element的子对象,因为Attr并不是DOM树的一部分

核心代码：

DomHelper.java

```

/**
 * Created by Jay on 2015/9/8 0008.
 */
public class DomHelper {
    public static ArrayList<Person> queryXML(Context context)
    {
        ArrayList<Person> Persons = new ArrayList<Person>();
        try {
            //①获得DOM解析器的工厂示例:
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory
            .newInstance();
            //②从Dom工厂中获得dom解析器
            DocumentBuilder dbBuilder = dbFactory.newDocumentBu
            ilder();
            //③把要解析的xml文件读入Dom解析器
            Document doc = dbBuilder.parse(context.getAssets().
            open("person2.xml"));
            System.out.println("处理该文档的DomImplementation对象="
            + doc.getImplementation());
            //④得到文档中名称为person的元素的结点列表
            NodeList nList = doc.getElementsByTagName("person")
            ;
            //⑤遍历该集合,显示集合中的元素以及子元素的名字
            for(int i = 0;i < nList.getLength();i++)
            {
                //先从Person元素开始解析
                Element personElement = (Element) nList.item(i)
                ;
                Person p = new Person();
                p.setId(Integer.valueOf(personElement.getAttrib
                ute("id")));
                //获取person下的name和age的Note集合
                NodeList childNoList = personElement.getChildNo
                des();
                for(int j = 0;j < childNoList.getLength();j++)
                {
                    Node childNode = childNoList.item(j);
                    //判断子note类型是否为元素Note
                    if(childNode.getNodeType() == Node.ELEMENT_
                    NODE)
                    {
                        Element childElement = (Element) childN
                        ode;
                        if("name".equals(childElement.getNodeNa
                        me()))
                            p.setName(childElement.getFirstChil
                        d().getNodeValue());
                        else if("age".equals(childElement.getNo
                        deName()))
                            p.setAge(Integer.valueOf(childEleme
                        nt.getFirstChild().getNodeValue()));
                    }
                }
            }
        }
    }
}

```

```

    }
    Persons.add(p);
}
} catch (Exception e) {e.printStackTrace();}
return Persons;
}
}

```

代码分析：

从代码我们就可以看出DOM解析XML的流程，先整个文件读入Dom解析器，然后形成一棵树，然后我们可以遍历节点列表获取我们需要的数据！

5.PULL解析XML数据

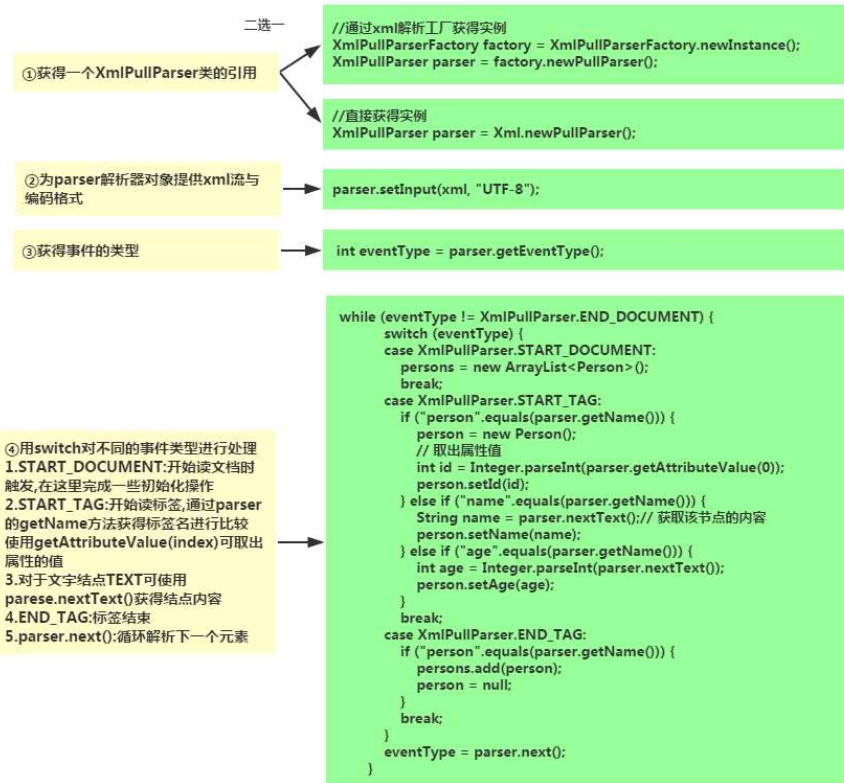
使用Pull解析xml

简单介绍

除了前面的SAX和DOM解析XML文件外,Android系统其实内置了Pull解析器用来解析xml文件,比如我们前面学到的SharedPreference就是使用的内置pull解析配置文件的!他的使用和SAX相似,都是采用事件驱动来完成xml的解析的;而pull的编码较为简单,只需处理开始与结束事件,通常使用switch语句,根据事件的不同类型,匹配不同的处理方式,有五种事件:START_DOCUMENT;START_TAG;TEXT;END_TAG;END_DOCUMENT XML pull提供了开始元素和结束元素。当某个元素开始时,可以调用parser.nextText从XML文档中提取所有字符数据。当解析到一个文档结束时,自动生成EndDocument事件。在PULL解析过程中返回的是数字,且我们需要自己获取产生的事件然后做相应的操作,而不像SAX那样由处理器触发一种事件的方法,执行我们的代码:读取得到xml的声明返回 START_DOCUMENT; 结束返回 END_DOCUMENT; 开始标签返回 START_TAG; 结束标签返回 END_TAG; 文本返回 TEXT。

使用PULL解析XML数据的流程：

使用Pull解析xml文件流程:



核心代码：

```

public static ArrayList<Person> getPersons(InputStream xml) throws Exception

```

```

{
    //XmlPullParserFactory pullPaser = XmlPullParserFactory.new
Instance();
    ArrayList<Person> persons = null;
    Person person = null;
    // 创建一个xml解析的工厂
    XmlPullParserFactory factory = XmlPullParserFactory.newInst
ance();
    // 获得xml解析类的引用
    XmlPullParser parser = factory.newPullParser();
    parser.setInput(xml, "UTF-8");
    // 获得事件的类型
    int eventType = parser.getEventType();
    while (eventType != XmlPullParser.END_DOCUMENT) {
        switch (eventType) {
            case XmlPullParser.START_DOCUMENT:
                persons = new ArrayList<Person>();
                break;
            case XmlPullParser.START_TAG:
                if ("person".equals(parser.getName())) {
                    person = new Person();
                    // 取出属性值
                    int id = Integer.parseInt(parser.getAttributeVa
lue(0));
                    person.setId(id);
                } else if ("name".equals(parser.getName())) {
                    String name = parser.nextText(); // 获取该节点的内
容
                    person.setName(name);
                } else if ("age".equals(parser.getName())) {
                    int age = Integer.parseInt(parser.nextText());
                    person.setAge(age);
                }
                break;
            case XmlPullParser.END_TAG:
                if ("person".equals(parser.getName())) {
                    persons.add(person);
                    person = null;
                }
                break;
        }
        eventType = parser.next();
    }
    return persons;
}

```

使用Pull生成xml数据的流程:

使用Pull生成xml文件流程:



核心代码：

```

public static void save(List<Person> persons, OutputStream out)
throws Exception {
    XmlSerializer serializer = Xml.newSerializer();
    serializer.setOutput(out, "UTF-8");
    serializer.startDocument("UTF-8", true);
    serializer.startTag(null, "persons");
    for (Person p : persons) {
        serializer.startTag(null, "person");
        serializer.attribute(null, "id", p.getId() + "");
        serializer.startTag(null, "name");
        serializer.text(p.getName());
        serializer.endTag(null, "name");
        serializer.startTag(null, "age");
        serializer.text(p.getAge() + "");
        serializer.endTag(null, "age");
        serializer.endTag(null, "person");
    }

    serializer.endTag(null, "persons");
    serializer.endDocument();
    out.flush();
    out.close();
}

```

6.代码示例下载：

运行效果图：



```
jay.xml
<?xml version='1.0' encoding='UTF-8'
standalone='yes' ?><persons><person
id="21"><name>逗比1</name><age>70</age></
person><person id="31"><name>逗比2</
name><age>50</age></person><person id="11"><name>
逗比3</name><age>30</age></person></persons>
```

^

反馈



代码下载：XMLParseDemo.zip：[下载 XMLParseDemo.zip](#)

本节小结：

本节介绍了Android中三种常用的XML解析方式，DOM，SAX和PULL，移动端我们建议用后面这两种，而PULL用起来更加简单，这里就不多说了，代码是最好的老师~本节就到这里，下节我们 来学习

Android为我们提供的扣脚Json解析方式！谢谢~

← 7.1.4 Android HTTP请求方式:HttpClient

7.2.2 Android JSON数据解析 →



坚持1个月 看美剧不用字幕

每天45分钟 30天见证你的英语奇迹

立即行动 >

在线实例

- [HTML 实例](#)
- [CSS 实例](#)
- [JavaScript 实例](#)
- [Ajax 实例](#)
- [jQuery 实例](#)
- [XML 实例](#)
- [Java 实例](#)

字符集&工具

- [HTML 字符集设置](#)
- [HTML ASCII 字符集](#)
- [HTML ISO-8859-1](#)
- [HTML 实体符号](#)
- [HTML 拾色器](#)
- [JSON 格式化工具](#)

最新更新

- [JavaScript 查找...](#)
- [JavaScript 判断...](#)
- [设置 SSH 通过密...](#)
- [CSS all 属性](#)
- [Px、Em 换算工具](#)
- [px,pt,em换算表](#)
- [px、em、rem 区别...](#)

站点信息

- [意见反馈](#)
- [免责声明](#)
- [关于我们](#)
- [文章归档](#)

关注微信



Copyright © 2013-2015 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1