7.2.1 Android XML数据解析

分类 **Android 基础入门教程**

本节引言：

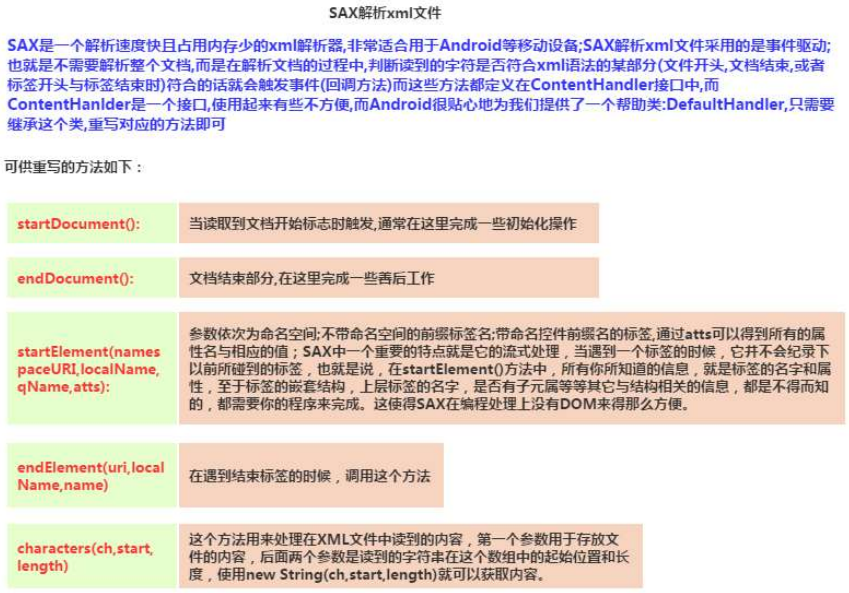
前面两节我们对Android内置的Http请求方式：HttpURLConnection和HttpClient，本来以为OkHttp 已经集成进来了，然后想讲解下Okhttp的基本用法，后来发现还是要导第三方，算了，放到进阶部分 吧，而本节我们来学习下Android为我们提供的三种解析XML数据的方案！他们分别是： SAX,DOM,PULL三种解析方式，下面我们就来对他们进行学习！

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：

2.三种解析XML方法的比较

3.SAX解析XML数据



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 {
```

```
        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, Stri
ng qName,
                             Attributes attributes) throws SAXE
xception {
        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 {
```

```
        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>
```

```
        <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

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 = DocumentBuilderF
actory.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DomImplemention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!

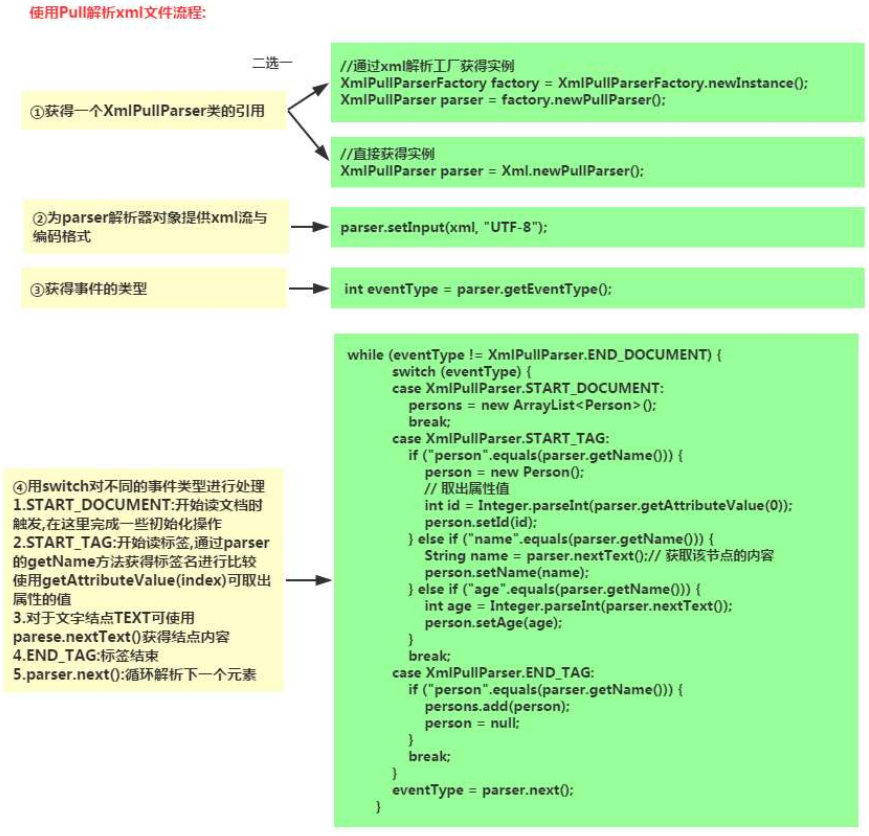
5.PULL解析XML数据

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,通常是使用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：



核心代码：

```
public static ArrayList<Person> getPersons(InputStream xml)thro
ws 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:

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：

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 →

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