



教学目标

- 掌握Android中XML文件的解析方法
- 掌握Android中JSON文件的解析方法



目录

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



数据格式引入

- 在开发应用程序的时候经常性的会遇到服务器与客户端通讯，或者不同语言间数据传递与交互的情况。





数据格式引入

- 就这样人们就总结出了统一的数据格式来传递数据，这就是数据格式或叫做数据交换格式。
- 常见的数据交换格式有：
 - XML
 - JSON
 -



目录

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



XML数据格式简介

- XML：可扩展标记语言 (Extensible Markup Language, XML) , 用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。 XML是标准通用标记语言 (SGML) 的子集，非常适合 Web 传输。 XML 提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。
- XML和HTML或其他语言最大的区别是：**XML是用来存储数据的。**



XML数据格式简介

- XML的简单使其易于在任何应用程序中读写数据，这使XML很快成为数据交换的首选公共语言，虽然不同的应用软件也支持其它的数据交换格式，但不久之后它们都支持了XML，那就意味着程序可以更容易的与Windows，Mac OS，Linux以及其它平台下产生的信息结合，可以很容易加载XML数据到程序中，或以XML格式输出结果。



XML语法简介

- 在XML中，采用了如下的语法：
 - 任何的起始标签都必须有一个结束标签。
 - 可以采用另一种简化语法，可以在一个标签中同时表示起始和结束标签。这种语法是在大于符号之前紧跟一个斜线（/），例如`<tag/ >`。XML解析器会将其翻译成`<tag></tag>`。
 - 标签必须按合适的顺序进行嵌套，所以结束标签必须按镜像顺序匹配起始标签。
 - 所有的属性都必须有值。
 - 所有的属性都必须在值的周围加上双引号。



XML语法简介

- <http://www.w3school.com.cn/xml/index.asp>

```
<?xml version="1.0" encoding="utf-8"?>
<students>
    <student>
        <name sex="man">小明</name>
        <nickname>明明</nickname>
    </student>
    <student>
        <name sex="woman">小红</name>
        <nickname>红红</nickname>
    </student>
</students>
```



XML的特点

- XML 被设计用来结构化存储以及传输信息。
- XML 仅仅是纯文本。
- XML 可自定义标签。
- XML 和 HTML 不可相互替代。
- XML 是 W3C 的推荐标准。
- XML 无所不在。



目录

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



Android中解析XML数据

- 在XML操作过程中最常遇见的一个问题就是如何解析它，
Android中有三种常见XML解析方式：
 - DOM方式解析
 - SAX方式解析
 - Pull方式解析



Android中解析XML - DOM

- DOM方式解析XML特点：
 - 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。
 - 一次全部加载，如果对于数据量小的情况下，它的效率还可以，如果XML文件很大的情况下，速度就会慢起来。
 - 直接把文档调入内存中，比较耗内存。



Android中解析XML - DOM

- DOM方式解析XML的步骤：
 1. 首先利用DocumentBuilderFactory创建一个DocumentBuilderFactory实例；
 2. 然后利用DocumentBuilderFactory创建DocumentBuilder；
 3. 然后加载XML文档（ Document ）；
 4. 然后获取文档的根结点（ Element ）；
 5. 然后获取根结点中所有子节点的列表（ NodeList ）；
 6. 然后再获取子节点列表中的需要读取的结点。



Android中解析XML - DOM

```
List<Student> list = new ArrayList<>();  
DocumentBuilderFactory factory  
    = DocumentBuilderFactory.newInstance();  
DocumentBuilder builder = factory.newDocumentBuilder();  
// 获得Document对象  
Document document = builder.parse(is);  
// 获得student的List  
NodeList studentList = document.getElementsByTagName("student");  
// 遍历student标签  
for (int i = 0; i < studentList.getLength(); i++) {  
    // 获得student标签  
    .....  
    // 加到List中  
    list.add(student);  
}
```



Android中解析XML - DOM

```
// 获得student标签
Node node_student = studentList.item(i);
// 获得student标签里面的标签
NodeList childNodes = node_student.getChildNodes();
// 新建student对象
Student student = new Student();
// 遍历student标签里面的标签
for (int j = 0; j < childNodes.getLength(); j++) {
    // 获得name和nickName标签
    .....
}
```



Android中解析XML - DOM

```
// 获得name和nickName标签
Node childNode = childNodes.item(j);
// 判断是name还是nickName
if ("name".equals(childNode.getNodeName())) {
    String name = childNode.getTextContent();
    student.setName(name);
    // 获取name的属性
    NamedNodeMap nnm = childNode.getAttributes();
    Node n = nnm.item(0); // 获取sex属性，由于只有一个属性，所以取0
    student.setSex(n.getTextContent());
} else if ("nickname".equals(childNode.getNodeName())) {
    String nickName = childNode.getTextContent();
    student.setNickName(nickname);
}
```



Android中解析XML - SAX

- SAX即是：Simple API for XML。
- SAX是基于事件驱动的。当然Android的事件机制是基于回调函数的，在用SAX解析XML文档时候，在读取到文档开始和结束标签时候就会回调一个事件，在读取到其它节点与内容时候也会回调一个事件。



Android中解析XML - SAX



- 事件源：
 - org.xml.sax包中的XMLReader，它通过parser()方法来解析XML文档，并产生事件。
- 事件处理器：
 - org.xml.sax包中ContentHandler、DTDHandler、ErrorHandler，以及EntityResolver这4个接口。



Android中解析XML - SAX

- XML Reader通过相应的事件处理器注册方法setXX()来完成与这4个接口的连接。

处理器名称	处理事件	XMLReader注册方法
ContentHandler	跟文档内容有关的事件 (1) 文档的开始与结束 (2) XML元素的开始与结束 (3) 可忽略的实体 (4) 名称空间前缀映射开始和结束 (5) 处理指令 (6) 字符数据和可忽略的空格	setContentHandler(ContentHandler h)
ErrorHandler	处理XML文档时产生的错误	setErrorHandler(ErrorHandler h)
DTDHandler	处理对文档的DTD进行解析时产生的相应事件	setDTDHandler(DTDHandler h)
EntityResolver	处理外部实体	setEntityResolver(EntityResolver e)



Android中解析XML - SAX

- 但是无需都实现这4个接口，SDK为我们提供了DefaultHandler类来处理，DefaultHandler类的一些主要事件回调方法如下：

方法名称	描述
setDocumentLocator(Locator locator)	设置一个可以定位文档内容事件发生位置的定位对象
startDocument()	处理文档内容开始事件
startElement(String uri, String localName, String qName, Attributes attrs)	处理元素开始事件
characters(char[] ch, int start, int length)	处理元素的字符内容
endElement(String uri, String localName, String qName)	处理元素结束事件
endDocument()	处理文档解析的结束事件



Android中解析XML - SAX

- 使用SAX方式解析XML需要XMLReader 以及 DefaultHandler来配合，解析步骤如下：
 1. 创建SAXParserFactory对象。
 2. 根据SAXParserFactory.newSAXParser()方法返回一个 SAXParser解析器。
 3. 根据SAXParser解析器获取事件源对象XMLReader。
 4. 实例化一个DefaultHandler对象。
 5. 连接事件源对象XMLReader到事件处理类DefaultHandler。
 6. 调用XMLReader的parse方法从输入源中获取xml数据。
 7. 通过DefaultHandler返回我们需要的数据集合。



Android中解析XML - SAX

- 自定义解析处理器MyHandler，继承自DefaultHandler，并重写其事件回调方法。

```
// 解析到文档开始调用，一般做初始化操作
@Override
public void startDocument() throws SAXException {
    list = new ArrayList<>();
}

// 解析到文档末尾调用，一般做回收操作
@Override
public void endDocument() throws SAXException {
    super.endDocument();
}
```



Android中解析XML - SAX

```
// 每读到一个元素就调用该方法
@Override
public void startElement(String uri, String localName, String
qName, Attributes attributes) throws SAXException {
    if ("student".equals(qName)) {
        // 读到student标签
        student = new Student();
    } else if ("name".equals(qName)) {
        // 获取name里面的属性
        String sex = attributes.getValue("sex");
        student.setSex(sex);
    }
    super.startElement(uri, localName, qName, attributes);
}
```



Android中解析XML - SAX

```
// 读到属性内容调用
@Override
public void characters(char[] ch, int start, int length)
    throws SAXException {
    tempString = new String(ch, start, length);
    super.characters(ch, start, length);
}
```



Android中解析XML - SAX

```
// 读到元素的结尾调用
@Override
public void endElement(String uri, String localName, String
qName) throws SAXException {
    if ("student".equals(qName)) {
        list.add(student);
    }
    if ("name".equals(qName)) {
        student.setName(tempString);
    } else if ("nickname".equals(qName)) {
        student.setNickName(tempString);
    }
    super.endElement(uri, localName, qName);
}
```



Android中解析XML - SAX

- 使用SAX方式解析XML需要XMLReader与DefaultHandler配合使用。

```
// 创建SAXParserFactory对象
SAXParserFactory spf = SAXParserFactory.newInstance();
// 初始化Sax解析器
SAXParser sp = spf.newSAXParser();
// 新建解析处理器
MyHandler handler = new MyHandler();
// 根据SAXParser解析器获取事件源对象XMLReader
XMLReader xmlReader = sp.getXMLReader();
xmlReader.setContentHandler(handler);
xmlReader.parse(new InputSource(is));
```



Android中解析XML - PULL

- Android的事件机制是基于回调函数的。PULL方式也是基于事件驱动的，与SAX方式一样。只不过PULL方式解析XML的方法返回的是数字。
 - 读取到XML的开始返回 START_DOCUMENT；
 - 读取到XML的结束返回 END_DOCUMENT；
 - 读取到XML的开始标签返回 START_TAG；
 - 读取到XML的结束标签返回 END_TAG；
 - 读取到XML的文本返回 TEXT。



Android中解析XML - PULL

- PULL方式解析XML基本处理方式：
 - 当导航到XmlPullParser.START_DOCUMENT，可以不做处理，当然你可以实例化集合对象等等。
 - 当导航到XmlPullParser.START_TAG，则判断是否是数据标签，如果是，则实例化数据对象，并调用getAttributeValue方法获取标签中属性值。
 - 当导航到其他标签，使用nextText方法来获取文本节点内容。
 - 导航到XmlPullParser.END_TAG的，有开始就要有结束。在这里我们就需要判断是否是数据结束标签，如果是，则把数据对象存进list集合中。



Android中解析XML - PULL

```
// 创建xmlPullParser解析器
XmlPullParser parser = Xml.newPullParser();
// 初始化xmlPullParser解析器
parser.setInput(is, "UTF-8");
// 读取文件的类型
int type = parser.getEventType();
// 无限判断文件类型进行读取
while (type != XmlPullParser.END_DOCUMENT) {
    // 继续往下读取标签类型
    type = parser.next();
}
```



Android中解析XML - PULL

```
switch (type) {  
    // 开始标签  
    case XmlPullParser.START_TAG:  
        if ("students".equals(parser.getName())) {  
            list = new ArrayList<>();  
        } else if ("student".equals(parser.getName())) {  
            student = new Student();  
        } else if ("name".equals(parser.getName())) {  
            // 获取sex属性  
            String sex = parser.getAttributeValue(null, "sex");  
            student.setSex(sex);  
            // 获取name值  
            .....  
        }  
        .....  
    break;  
}
```



Android中解析XML - PULL

```
// 结束标签
case XmlPullParser.END_TAG:
    if ("student".equals(parser.getName())) {
        list.add(student);
    }
    break;
```



目录

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



JSON数据格式简介

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于JavaScript (Standard ECMA-262 3rd Edition - December 1999) 的一个子集。JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。这些特性使JSON成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成。



JSON数据格式简介

- JSON 可以将 JavaScript 对象中表示的一组数据转换为字符串，然后就可以在函数之间轻松地传递这个字符串，或者在异步应用程序中将字符串从客户端传递给服务器端程序。这个字符串看起来有点儿古怪，但是 JavaScript 很容易解释它，而且 JSON 可以表示比"名称 / 值对"更复杂的结构。例如，可以表示数组和复杂的对象，而不仅仅是键和值的简单列表。



JSON基础结构

- JSON基本的结构有两种：
 - “名称/值” 对的集合（ a collection of name/value pairs ）。不同的语言中，它被理解为对象（ object ），记录（ record ），结构（ struct ），字典（ dictionary ），哈希表（ hash table ），有键列表（ keyed list ），或者关联数组（ associative array ）。
 - 值的有序列表（ an ordered list of values ）。在大部分语言中，它被理解为数组（ array ）。



JSON数据格式示例

- JSON 表示名称 / 值对的方式

```
{  
    "name": "zhangsan",  
    "sex": "man",  
    "age": 18  
}
```



JSON数据格式示例

- JSON 表示数组的方式：

```
{  
    "person": [  
        {  
            "name": "zhang",  
            "sex": "man",  
            "age": 18  
        },  
        {  
            "name": "lily",  
            "sex": "woman",  
            "age": 18  
        }  
    ]  
}
```



XML和JSON的比较

- 可读性
 - JSON和XML的可读性可谓不相上下，一边是简易的语法，一边是规范的标签形式，很难分出胜负。
- 可扩展性
 - XML天生有很好的扩展性，JSON当然也有。不过JSON可以存储JavaScript复合对象，有着XML不可比拟的优势。



XML和JSON的比较

- 编码难度
 - XML有丰富的编码工具，比如Dom4j、JDom等，JSON也有提供的工具。无工具的情况下，相信熟练的开发人员一样能很快的写出想要的XML文档和JSON字符串，不过，XML文档要多很多结构上的字符。
- 解码难度
 - 难度相当，XML存在时间较长，技术较成熟。
 - JSON是较新的数据结构，但有方便的解析类库。
- JSON对数据的描述性比XML较差。
- JSON的速度要远远快于XML。



目录

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



Android中解析JSON数据的方法

- Android提供JSON解析类，都在包org.Json下：
 - **JsonObject**：可以看作是一个JSON对象，这是系统中有关JSON定义的基本单元，其包含一对儿(Key/Value)数值。它对外部(External)：应用toString()方法输出的数值)调用的响应体现为一个标准的字符串。Value的类型包括：Boolean、JSONArray、JsonObject、Number、String或者默认值JsonObject.NULL object。



Android中解析JSON数据的方法

– **JsonStringer** : JSON文本构建类，这个类可以帮助快速和便捷的创建JSON text。其最大的优点在于可以减少由于格式的错误导致程序异常，引用这个类可以自动严格按照JSON语法规则（syntax rules）创建JSON text。每个JsonStringer实体只能对应创建一个JSON text。



Android中解析JSON数据的方法

- **JSONArray** : 它代表一组有序的数值。将其转换为String输出(toString)所表现的形式是用方括号包裹，数值以逗号分隔（例如：[value1,value2,value3]）。这个类的内部同样具有查询行为，get()和opt()两种方法都可以通过index索引返回指定的数值，put()方法用来添加或者替换数值。同样这个类的value类型可以包括：Boolean、JSONArray、JSONObject、Number、String或者默认值JSONObject.NULL object。
- **JsonTokener** : Json解析类。
- **JsonException** : Json中用到的异常。



Android中解析JSON数据的方法

```
// 创建JSONObject的实例  
JSONObject jsonObject = new JSONObject();  
// 调用put方法将user对象的数据  
// 采用key/value的形式放入JSONObject对象中  
jsonObject.put("name", user.getName())  
    .put("sex", user.getSex())  
    .put("age", user.getAge());  
String jsonStr = jsonObject.toString();
```



Android中解析JSON数据的方法

```
// 通过JSON字符串创建一个JSONObject实例  
JSONObject object = new JSONObject(jsonStr);  
User user = new User();  
// 从JSONObject对象中取出值赋给user对象的属性  
user.setName(object.getString("name"));  
user.setAge(object.getInt("age"));  
user.setSex(object.getString("sex"));
```



内容回顾

1 数据格式引入

2 XML数据格式简介

3 解析XML数据的方法

4 JSON数据格式简介

5 解析JSON数据的方法



Thank you!

