Application Research of XML Parsing Technology Based on Android

Li Jiang

Department of Software Engineering, Guangzhou University, China Software Engineering Department No. 548, Guangcun South Road, Conghua District, Guangzhou
e-mail: JL@SISE.COM.CN

Abstract. Based on the research of XML core technology, this paper introduces the current mainstream XML parsing methods, XML DOM and SAX methods. For the current android system, which is used in android operating system. XML is researched in the android layout file, which is combined with the current android platform. The paper introduce application of XML pull parsing process, first study the parsing performance of XML, the second through an example to discuss these three methods, the last analysis XML file Size and compares these three methods.

1. Introduction

1.1 Understanding XML
(eXtensibleMarkup Language) XML is a markup language designed specifically for Internet. The focus of XML is on managing the data itself, not on the style of the data, and the display of the data is left to another technique. Although XML is a markup language designed specifically for Internet, it has become the standard for data representation and data exchange on Internet. It is followed by a pile of XML documents. How to effectively manage and quickly query these XML documents is a hot research topic in the academic world, the so-called XML database[1].

2. XML file parsing and its method
The parsing of XML is implemented through the application programming interface API. The application programming interface around the emergence of XML is undoubtedly very important for application development. Application developers can use these standard interfaces to get and set elements, attributes, data content, etc[2], in a document. It is clear that the current mainstream application programming interfaces are DOM and SAX.

2.1 DOM analysis technology
DOM is the document object model, which provides a standard program interface that can be applied to different environments and applications. The interface can be implemented in any language. Modeling documents with objects that describe not only the structure of the document, but also the behavior of objects in the model. The object model is used to describe the logical structure of the document[3]. The object model implements the relationship and interoperability between the behavior of the interface interface used to represent and manipulate the document and the property interface.
The logical structure of a document in XML is similar to a tree. The document, the root, elements, element content, attributes, attribute values, etc. in the document are all represented in the form of an object model[4]. It also contains annotations, processing instructions, document types, entities, entity references, namespaces, events, style sheets, and more. By using, the program developer can dynamically create documents, traverse the document structure, add, modify, delete document content, and change the way the document is displayed.

In general, there are advantages in that the object tree that is randomly accessed by the document is loaded into the memory, the operation method of the object is provided, and the program can access any object of the tree as needed. Programming makes it easy to design object trees in line with object-oriented thinking, and accessing trees is very simple. The developer only needs to call the instructions for building the tree, and then use the tree nodes needed to access the task. It's easy to add and modify elements in the tree[5].

2.2 SAX analysis technology
SAX is an "event-driven" software development package interface. The process is handled in an event-based manner, using two steps that the user first uses to define element markup events, and then enable the parser to parse the document. When the parser traverses the document and encounters a user-defined element tag event[6], it generates an event that will drive the application to call the appropriate program to process that particular element. Otherwise, the parser continues to parse. In contrast, there are the following advantages:

1) can parse files of any size
2) Because the entire file does not need to be loaded into the memory, the memory usage is generally much smaller and does not increase as the file size increases.
3) Simple and fast for reading specific data
4) Very easy to use, if you can get the information you need from a simple sequence of documents, it is almost certainly the fastest way.

3. Android XML application
XML pull provides start and end elements[7]. When an element starts, we can call parser.nextText() to extract all the character data from the XML document. The EndDocument event is automatically generated when interpreted to the end of a document.

3.1. android activity_main.xml
<LinearLayout 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:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_product"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="generate XML" />

    <Button
        android:id="@+id/btn_pull"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Analysis XML" />

</LinearLayout>
3.2 XmlPullParser: XML

3.2.1 XML pull parsing process
The processing of the XML file given above is shown in Figure 1.

![XML pull parsing process diagram]

3.2.2 XmlPullParser: XML pull parser
Android pull parsing has the following process:

1. XmlPullParser.getEventType: mainly gets the node where the current parser is located[8].
2. XmlPullParser.getName: mainly used to get the xml tag name when it is in the START_TAG or END_TAG event.
3. XmlPullParser.getText: mainly used to obtain text content[9].
4. XmlPullParser.next (): Move the parser to the next event, so that all xml files can be traversed.

4. Conclusion
For Android mobile devices, because the resources of the device are more valuable and the memory is limited, we need to choose the appropriate technology to parse the XML, which will help to improve the speed of access.

1. When processing an XML file, the DOM parses the XML file into a tree structure and puts it into memory for processing. When the XML file is small, we can choose DOM because it is simple and intuitive.

2. SAX uses an event as a model for parsing an XML file. It converts an XML file into a series of events, which are determined by different event handlers. When the XML file is large, it is reasonable to choose SAX technology. Although the amount of code is a bit large, it does not require all XML files to be loaded into memory. This is more efficient for limited Android memory, and Android provides a traditional way to use SAX and a handy SAX wrapper. Using the Android.util.Xml class, as you can see from the example, it's easier than using SAX.

3. XML pull parsing does not listen to the end of the element like SAX parsing, but does most of the processing at the beginning. This facilitates early reading of XML files, which can greatly reduce parsing time, which is especially important for slower connected mobile devices. The XML Pull parser is a more efficient method when the XML document is large but only a part of the document is needed.
References
[1] A. Aboulnaga, A. R. Alameldeen, and J. Naughton. Estimating the selectivity of XML Path expressions for Internet-scale applications. In Proceedings of VLDB, Pages 591-600, 2001.
[2] Huang Ying, Yang Huafu. Research on computer storage of compressed documents [J]. Computer Engineering, 2002, 5: 15-18.
[3] Sa Shizhen, Wang Shan. Database System Introduction 4 Beijing Higher Education Press, 2006.
[4] Luo Daofeng, Meng Xiaofeng, et al. OrientStore: Native XML storage method [J]. Computer Science, 2003, 30 (suppl.): 105-110.
[5] Luo Daofeng, Meng Xiaofeng, et al. Update method for precoding of XML data extension [J]. Journal of Software, 2005, 16(5): 810-818.
[6] M Carey, D DeWitt, et al. The BUCKY Object-Relational Benchmark [C]. Proc. of ACM SIGMOD 1997, New York: ACM Press, 1997. 12-21.
[7] Thompson H. xsv: Schema validator. 2002. http://www.w3c.org/2001/03/webdata/xsv
[8] Fuhr N, Großjohann K. XIRQL: A query language for information retrieval in XML documents. In: Croft WB, Harper DJ, Kraft DH, Zobel J, eds. Proc. of the 24th Annual Int'l ACM SIGIR Conf. On Research and Development in Information Retrieval (SIGIR). New Orleans: ACM Press, 2001. 172–180.
[9] Amer-Yahia S, Botev C, Shanmugasundaram J. TeXQuery: A FullText search extension to XQuery. In: Feldman SI, Uretsky M, Najork M, Wills CE, eds. Proc. of the 13th Conf. on World Wide Web (WWW). Manhattan: ACM Press, 2004. 583–594.
[10] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). Baltimore: ACM Press, 2005.