**dom4j**

　　dom4j是一个Java 的XML API，类似于jdom，用来读写XML文件的。dom4j是一个非常非常优秀的Java XML API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件，可以在SourceForge上找到它。在IBM developerWorks上面可以找到一篇文章，对主流的Java XML API进行的性能、功能和易用性的评测，dom4j无论在那个方面都是非常出色的。我早在将近两年之前就开始使用dom4j，直到现在。如今你可以看到越 来越多的Java软件都在使用dom4j来读写XML，特别值得一提的是连Sun的JAXM也在用dom4j。这是必须使用的jar包， Hibernate用它来读写配置文件

　　dom4j需要导入dom4j-full.jar包

　　DOM4J是dom4j.org出品的一个开源XML解析包，它的网站中这样定义：

　　Dom4j is an easy to use, open source library for working with XML, XPath and XSLT on the Java platform using the Java Collections Framework and with full support for DOM, SAX and JAXP.

　　Dom4j是一个易用的、开源的库，用于XML，XPath和XSLT。它应用于Java平台，采用了Java集合框架并完全支持DOM，SAX和JAXP。

　　DOM4J使用起来非常简单。只要你了解基本的XML-DOM模型，就能使用。然而他自己带的 指南只有短短一页（html），不过说的到挺全。国内的中文资料很少。因而俺写这个短小的教程方便大家使用，这篇文章仅谈及基本的用法，如需深入的使用， 请……自己摸索或查找别的资料。

　　之前看过IBM developer社区的文章（参见附录），提到一些XML解析包的性能比较，其中DOM4J的性能非常出色，在多项测试中名列前茅。（事实上DOM4J的官方文档中也引用了这个比较）所以这次的项目中我采用了DOM4J作为XML解析工具。

　　在国内比较流行的是使用JDOM作为解析器，两者各擅其长，但DOM4J最大的特色是使用大量 的接口，这也是它被认为比JDOM灵活的主要原因。大师不是说过么，“面向接口编程”。目前使用DOM4J的已经越来越多。如果你善于使用JDOM，不妨 继续用下去，只看看本篇文章作为了解与比较，如果你正要采用一种解析器，不如就用DOM4J吧。

　　它的主要接口都在org.dom4j这个包里定义：

　　Attribute Attribute

定义了XML的属性

　　Branch

Branch为能够包含子节点的节点如XML元素(Element)和文档(Docuemnts)定义了一个公共的行为，

　　CDATA

CDATA 定义了XML CDATA 区域

　　CharacterData

CharacterData是一个标识借口，标识基于字符的节点。如CDATA，Comment, Text.

　　Comment

Comment 定义了XML注释的行为

　　Document

定义了XML文档

　　DocumentType

DocumentType 定义XML DOCTYPE声明

　　Element

Element定义XML 元素

　　ElementHandler

ElementHandler定义了 Element 对象的处理器

　　ElementPath 被 ElementHandler 使用，用于取得当前正在处理的路径层次信息

　　Entity

Entity定义 XML entity

　　Node

Node为所有的dom4j中XML节点定义了多态行为

　　NodeFilter

NodeFilter 定义了在dom4j节点中产生的一个滤镜或谓词的行为（predicate）

　　ProcessingInstruction

ProcessingInstruction 定义 XML 处理指令.

　　Text

Text 定义XML 文本节点.

　　Visitor

Visitor 用于实现Visitor模式.

　　XPath

XPath 在分析一个字符串后会提供一个XPath 表达式

　　看名字大致就知道它们的涵义如何了。

　　要想弄懂这套接口，关键的是要明白接口的继承关系：

　　interface java.lang.Cloneable

　　interface org.dom4j.Node

　　interface org.dom4j.Attribute

　　interface org.dom4j.Branch

　　interface org.dom4j.Document

　　interface org.dom4j.Element

　　interface org.dom4j.CharacterData

　　interface org.dom4j.CDATA

　　interface org.dom4j.Comment

　　interface org.dom4j.Text

　　interface org.dom4j.DocumentType

　　interface org.dom4j.Entity

　　interface org.dom4j.ProcessingInstruction

　　一目了然，很多事情都清楚了。大部分都是由Node继承来的。知道这些关系，将来写程序就不会出现ClassCastException了。

　　下面给出一些例子（部分摘自DOM4J自带的文档），简单说一下如何使用。

　　１． 读取并解析XML文档：

　　读写XML文档主要依赖于org.dom4j.io包，其中提供DOMReader和SAXReader两类不同方式，而调用方式是一样的。这就是依靠接口的好处。

　　// 从文件读取XML，输入文件名，返回XML文档

　　public Document read(String fileName) throws MalformedURLException, DocumentException {

　　SAXReader reader = new SAXReader();

　　Document document = reader.read(new File(fileName));

　　return document;

　　}

　　其中，reader的read方法是重载的，可以从InputStream, File, Url等多种不同的源来读取。得到的Document对象就带表了整个XML。

　　根据本人自己的经验，读取的字符编码是按照XML文件头定义的编码来转换。如果遇到乱码问题，注意要把各处的编码名称保持一致即可。

　　２． 取得Root节点

　　读取后的第二步，就是得到Root节点。熟悉XML的人都知道，一切XML分析都是从Root元素开始的。

　　public Element getRootElement(Document doc){

　　return doc.getRootElement();

　　}

　　３． 遍历XML树

　　DOM4J提供至少3种遍历节点的方法：

　　1) 枚举(Iterator)

　　// 枚举所有子节点

　　for ( Iterator i = root.elementIterator(); i.hasNext(); ) {

　　Element element = (Element) i.next();

　　// do something

　　}

　　// 枚举名称为foo的节点

　　for ( Iterator i = root.elementIterator(foo); i.hasNext();) {

　　Element foo = (Element) i.next();

　　// do something

　　}

　　// 枚举属性

　　for ( Iterator i = root.attributeIterator(); i.hasNext(); ) {

　　Attribute attribute = (Attribute) i.next();

　　// do something

　　}

　　2)递归

　　递归也可以采用Iterator作为枚举手段，但文档中提供了另外的做法

　　public void treeWalk() {

　　treeWalk(getRootElement());

　　}

　　public void treeWalk(Element element) {

　　for (int i = 0, size = element.nodeCount(); i < size; i++) {

　　Node node = element.node(i);

　　if (node instanceof Element) {

　　treeWalk((Element) node);

　　} else { // do something....

　　}

　　}

　　}

　　3) Visitor模式

　　最令人兴奋的是DOM4J对Visitor的支持，这样可以大大缩减代码量，并且清楚易懂。了 解设计模式的人都知道，Visitor是GOF设计模式之一。其主要原理就是两种类互相保有对方的引用，并且一种作为Visitor去访问许多 Visitable。我们来看DOM4J中的Visitor模式(快速文档中没有提供)

　　只需要自定一个类实现Visitor接口即可。

　　public class MyVisitor extends VisitorSupport {

　　public void visit(Element element){

　　System.out.println(element.getName());

　　}

　　public void visit(Attribute attr){

　　System.out.println(attr.getName());

　　}

　　}

　　调用： root.accept(new MyVisitor())

　　Visitor接口提供多种Visit()的重载，根据XML不同的对象，将采用不同的方式来 访问。上面是给出的Element和Attribute的简单实现，一般比较常用的就是这两个。VisitorSupport是DOM4J提供的默认适配 器，Visitor接口的Default Adapter模式，这个模式给出了各种visit(\*)的空实现，以便简化代码。

　　注意，这个Visitor是自动遍历所有子节点的。如果是root.accept(MyVisitor)，将遍历子节点。我第一次用的时候，认为是需要自己遍历，便在递归中调用Visitor，结果可想而知。

　　4. XPath支持

　　DOM4J对XPath有良好的支持，如访问一个节点，可直接用XPath选择。

　　public void bar(Document document) {

　　List list = document.selectNodes( //foo/bar );

　　Node node = document.selectSingleNode(//foo/bar/author);

　　String name = node.valueOf( @name );

　　}

　　例如，如果你想查找XHTML文档中所有的超链接，下面的代码可以实现：

　　public void findLinks(Document document) throws DocumentException {

　　List list = document.selectNodes( //a/@href );

　　for (Iterator iter = list.iterator(); iter.hasNext(); ) {

　　Attribute attribute = (Attribute) iter.next();

　　String url = attribute.getValue();

　　}

　　}

　　5. 字符串与XML的转换

　　有时候经常要用到字符串转换为XML或反之，

　　// XML转字符串

　　Document document = ...;

　　String text = document.asXML();

　　// 字符串转XML

　　String text = <name>James</name> </person>;

　　Document document = DocumentHelper.parseText(text);

　　6 用XSLT转换XML

　　public Document styleDocument(

　　Document document,

　　String stylesheet

　　) throws Exception {

　　// load the transformer using JAXP

　　TransformerFactory factory = TransformerFactory.newInstance();

　　Transformer transformer = factory.newTransformer(

　　new StreamSource( stylesheet )

　　);

　　// now lets style the given document

　　DocumentSource source = new DocumentSource( document );

　　DocumentResult result = new DocumentResult();

　　transformer.transform( source, result );

　　// return the transformed document

　　Document transformedDoc = result.getDocument();

　　return transformedDoc;

　　}

　　7. 创建XML

　　一般创建XML是写文件前的工作，这就像StringBuffer一样容易。

　　public Document createDocument() {

　　Document document = DocumentHelper.createDocument();

　　Element root = document.addElement(root);

　　Element author1 =

　　root

　　.addElement(author)

　　.addAttribute(name, James)

　　.addAttribute(location, UK)

　　.addText(James Strachan);

　　Element author2 =

　　root

　　.addElement(author)

　　.addAttribute(name, Bob)

　　.addAttribute(location, US)

　　.addText(Bob McWhirter);

　　return document;

　　}

　　8. 文件输出

　　一个简单的输出方法是将一个Document或任何的Node通过write方法输出

　　FileWriter out = new FileWriter( foo.xml );

　　document.write(out);

　　如果你想改变输出的格式，比如美化输出或缩减格式，可以用XMLWriter类

　　public void write(Document document) throws IOException {

　　// 指定文件

　　XMLWriter writer = new XMLWriter(

　　new FileWriter( output.xml )

　　);

　　writer.write( document );

　　writer.close();

　　// 美化格式

　　OutputFormat format = OutputFormat.createPrettyPrint();

　　writer = new XMLWriter( System.out, format );

　　writer.write( document );

　　// 缩减格式

　　format = OutputFormat.createCompactFormat();

　　writer = new XMLWriter( System.out, format );

　　writer.write( document );

　　}

　　如何，DOM4J够简单吧，当然，还有一些复杂的应用没有提到，如ElementHandler等。如果你动心了，那就一起来用DOM4J.

　　DOM4J官方网站：(我老连不上)

　　http://www.dom4j.org/

　　DOM4J下载(SourceForge)，最新版本为1.4

　　http://sourceforge.net/projects/dom4j

　　用Dom4j解析XML及中文问题

　　发表于 2004年9月27日 20:21

　　本文主要讨论了用dom4j解析XML的基础问题，包括建立XML文档，添加、修改、删除节点，以及格式化（美化）输出和中文问题。可作为dom4j的入门资料。

　　转载自:http://jalorsoft.com/holen/

　　作者：陈光（holen@263.net）

　　时间：2004-09-11

　　本文主要讨论了用dom4j解析XML的基础问题，包括建立XML文档，添加、修改、删除节点，以及格式化（美化）输出和中文问题。可作为dom4j的入门资料。

　　1． 下载与安装

　　dom4j是sourceforge.net上的一个开源项目，主要用于对XML的解析。从2001年7月发布第一版以来，已陆续推出多个版本，目前最高版本为1.5。

　　dom4j专门针对Java开发，使用起来非常简单、直观，在Java界，dom4j正迅速普及。

　　可以到http://sourceforge.net/projects/dom4j下载其最新版。

　　dom4j1.5的完整版大约13M，是一个名为dom4j-1.5.zip的压缩包，解压后 有一个dom4j-1.5.jar文件，这就是应用时需要引入的类包，另外还有一个jaxen-1.1-beta-4.jar文件，一般也需要引入，否则 执行时可能抛java.lang.NoClassDefFoundError: org/jaxen/JaxenException异常，其他的包可以选择用之。

　　2． 示例XML文档（holen.xml）

　　为了述说方便，先看一个XML文档，之后的操作均以此文档为基础。

　　holen.xml

　　<?xml version="1.0" encoding="UTF-8"?>

　　<books>

　　<!--This is a test for dom4j, holen, 2004.9.11-->

　　<book show="yes">

　　<title>Dom4j Tutorials</title>

　　</book>

　　<book show="yes">

　　<title>Lucene Studing</title>

　　</book>

　　<book show="no">

　　<title>Lucene in Action</title>

　　</book>

　　<owner>O'Reilly</owner>

　　</books>

　　这是一个很简单的XML文档，场景是一个网上书店，有很多书，每本书有两个属性，一个是书名，一个为是否展示[show]，最后还有一项是这些书的拥有者[owner]信息。

　　3． 建立一个XML文档

　　/\*\*

　　\* 建立一个XML文档,文档名由输入属性决定

　　\* @param filename 需建立的文件名

　　\* @return 返回操作结果, 0表失败, 1表成功

　　\*/

　　public int createXMLFile(String filename){

　　/\*\* 返回操作结果, 0表失败, 1表成功 \*/

　　int returnValue = 0;

　　/\*\* 建立document对象 \*/

　　Document document = DocumentHelper.createDocument();

　　/\*\* 建立XML文档的根books \*/

　　Element booksElement = document.addElement("books");

　　/\*\* 加入一行注释 \*/

　　booksElement.addComment("This is a test for dom4j, holen, 2004.9.11");

　　/\*\* 加入第一个book节点 \*/

　　Element bookElement = booksElement.addElement("book");

　　/\*\* 加入show属性内容 \*/

　　bookElement.addAttribute("show","yes");

　　/\*\* 加入title节点 \*/

　　Element titleElement = bookElement.addElement("title");

　　/\*\* 为title设置内容 \*/

　　titleElement.setText("Dom4j Tutorials");

　　/\*\* 类似的完成后两个book \*/

　　bookElement = booksElement.addElement("book");

　　bookElement.addAttribute("show","yes");

　　titleElement = bookElement.addElement("title");

　　titleElement.setText("Lucene Studing");

　　bookElement = booksElement.addElement("book");

　　bookElement.addAttribute("show","no");

　　titleElement = bookElement.addElement("title");

　　titleElement.setText("Lucene in Action");

　　/\*\* 加入owner节点 \*/

　　Element ownerElement = booksElement.addElement("owner");

　　ownerElement.setText("O'Reilly");

　　try{

　　/\*\* 将document中的内容写入文件中 \*/

　　XMLWriter writer = new XMLWriter(new FileWriter(new File(filename)));

　　writer.write(document);

　　writer.close();

　　/\*\* 执行成功,需返回1 \*/

　　returnValue = 1;

　　}catch(Exception ex){

　　ex.printStackTrace();

　　}

　　return returnValue;

　　}

　　说明：

　　Document document = DocumentHelper.createDocument();

　　通过这句定义一个XML文档对象。

　　Element booksElement = document.addElement("books");

　　通过这句定义一个XML元素，这里添加的是根节点。

　　Element有几个重要的方法：

　　l addComment：添加注释

　　l addAttribute：添加属性

　　l addElement：添加子元素

　　最后通过XMLWriter生成物理文件，默认生成的XML文件排版格式比较乱，可以通过 OutputFormat类的createCompactFormat()方法或createPrettyPrint()方法格式化输出，默认采用 createCompactFormat()方法，显示比较紧凑，这点将在后面详细谈到。

　　生成后的holen.xml文件内容如下：

　　<?xml version="1.0" encoding="UTF-8"?>

　　<books><!--This is a test for dom4j, holen, 2004.9.11--><book show="yes"><title>Dom4j Tutorials</title></book><book show="yes"><title>Lucene Studing</title></book><book show="no"><title>Lucene in Action</title></book><owner>O'Reilly</owner></books>

　　4． 修改XML文档

　　有三项修改任务，依次为：

　　l 如果book节点中show属性的内容为yes,则修改成no

　　l 把owner项内容改为Tshinghua，并添加date节点

　　l 若title内容为Dom4j Tutorials,则删除该节点

　　/\*\*

　　\* 修改XML文件中内容,并另存为一个新文件

　　\* 重点掌握dom4j中如何添加节点,修改节点,删除节点

　　\* @param filename 修改对象文件

　　\* @param newfilename 修改后另存为该文件

　　\* @return 返回操作结果, 0表失败, 1表成功

　　\*/

　　public int ModiXMLFile(String filename,String newfilename){

　　int returnValue = 0;

　　try{

　　SAXReader saxReader = new SAXReader();

　　Document document = saxReader.read(new File(filename));

　　/\*\* 修改内容之一: 如果book节点中show属性的内容为yes,则修改成no \*/

　　/\*\* 先用xpath查找对象 \*/

　　List list = document.selectNodes("/books/book/@show" );

　　Iterator iter = list.iterator();

　　while(iter.hasNext()){

　　Attribute attribute = (Attribute)iter.next();

　　if(attribute.getValue().equals("yes")){

　　attribute.setValue("no");

　　}

　　}

　　/\*\*

　　\* 修改内容之二: 把owner项内容改为Tshinghua

　　\* 并在owner节点中加入date节点,date节点的内容为2004-09-11,还为date节点添加一个属性type

　　\*/

　　list = document.selectNodes("/books/owner" );

　　iter = list.iterator();

　　if(iter.hasNext()){

　　Element ownerElement = (Element)iter.next();

　　ownerElement.setText("Tshinghua");

　　Element dateElement = ownerElement.addElement("date");

　　dateElement.setText("2004-09-11");

　　dateElement.addAttribute("type","Gregorian calendar");

　　}

　　/\*\* 修改内容之三: 若title内容为Dom4j Tutorials,则删除该节点 \*/

　　list = document.selectNodes("/books/book");

　　iter = list.iterator();

　　while(iter.hasNext()){

　　Element bookElement = (Element)iter.next();

　　Iterator iterator = bookElement.elementIterator("title");

　　while(iterator.hasNext()){

　　Element titleElement=(Element)iterator.next();

　　if(titleElement.getText().equals("Dom4j Tutorials")){

　　bookElement.remove(titleElement);

　　}

　　}

　　}

　　try{

　　/\*\* 将document中的内容写入文件中 \*/

　　XMLWriter writer = new XMLWriter(new FileWriter(new File(newfilename)));

　　writer.write(document);

　　writer.close();

　　/\*\* 执行成功,需返回1 \*/

　　returnValue = 1;

　　}catch(Exception ex){

　　ex.printStackTrace();

　　}

　　}catch(Exception ex){

　　ex.printStackTrace();

　　}

　　return returnValue;

　　}

　　说明：

　　List list = document.selectNodes("/books/book/@show" );

　　list = document.selectNodes("/books/book");

　　上述代码通过xpath查找到相应内容。

　　通过setValue()、setText()修改节点内容。

　　通过remove()删除节点或属性。

　　5． 格式化输出和指定编码

　　默认的输出方式为紧凑方式，默认编码为UTF-8，但对于我们的应用而言，一般都要用到中文，并且希望显示时按自动缩进的方式的显示，这就需用到OutputFormat类。

　　/\*\*

　　\* 格式化XML文档,并解决中文问题

　　\* @param filename

　　\* @return

　　\*/

　　public int formatXMLFile(String filename){

　　int returnValue = 0;

　　try{

　　SAXReader saxReader = new SAXReader();

　　Document document = saxReader.read(new File(filename));

　　XMLWriter writer = null;

　　/\*\* 格式化输出,类型IE浏览一样 \*/

　　OutputFormat format = OutputFormat.createPrettyPrint();

　　/\*\* 指定XML编码 \*/

　　format.setEncoding("GBK");

　　writer= new XMLWriter(new FileWriter(new File(filename)),format);

　　writer.write(document);

　　writer.close();

　　/\*\* 执行成功,需返回1 \*/

　　returnValue = 1;

　　}catch(Exception ex){

　　ex.printStackTrace();

　　}

　　return returnValue;

　　}

　　说明：

　　OutputFormat format = OutputFormat.createPrettyPrint();

　　这句指定了格式化的方式为缩进式，则非紧凑式。

　　format.setEncoding("GBK");

　　指定编码为GBK。

　　XMLWriter writer = new XMLWriter(new FileWriter(new File(filename)),format);

　　这与前面两个方法相比，多加了一个OutputFormat对象，用于指定显示和编码方式。

　　6． 完整的类代码

　　前面提出的方法都是零散的，下面给出完整类代码。

　　Dom4jDemo.java

　　package com.holen.dom4j;

　　import java.io.File;

　　import java.io.FileWriter;

　　import java.util.Iterator;

　　import java.util.List;

　　import org.dom4j.Attribute;

　　import org.dom4j.Document;

　　import org.dom4j.DocumentHelper;

　　import org.dom4j.Element;

　　import org.dom4j.io.OutputFormat;

　　import org.dom4j.io.SAXReader;

　　import org.dom4j.io.XMLWriter;