用DOM树创建XML

1、思路：**先封装构建一颗DOM树，然后将DOM树转换成XML文件**

2、**三种写DOM树到XML文件的方式**：

1）使用DOM（或DOM4J、JDOM）

2）使用LSSerializer接口

3）使用XSLT（Extensible StyleSheet Language Transformations）可扩展的格式页转换，XSL转换

3、**构建一颗DOM树**

1）不带名称空间

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document doc = builder.newDocument();

//创建标签节点

Element rootElement = doc.**createElement**(rootName);

Element childElement = doc.createElement(childName);

//创建文本节点

Text textNode = doc.**createTextNode**(textContents);

//给文档添加根元素，给父节点添加子节点

doc.**appendChild**(rootElement);

rootElement.appendChild(childElement);

childElement.appendChild(textNode); //添加文本节点

//设置元素属性

rootElement.**setAttribute**(name, value);

2)带名称空间

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

factory.**setNamespaceAware**(true);

DocumentBuilder builder = factory.newDocumentBuilder();

Document doc = builder.newDocument();

//然后使用createElementNS而不是createElement来创建节点

String namespace = <http://www.w3c.org/2000/svg>;

Element rootElement = doc.**createElementNS**(namespace, “svg”); //svg标签名

//如果节点具有命名空间前缀，则xmlns前缀属性会被自动创建

Element svgElement = doc.**createElementNS**(namespace, “svg:svg”); //svg前缀

当该元素被写入XML文件时，它会转变为：

<svg:svg xmlns:svg=”http://www.w3c.org/2000/svg”>

//元素属性有名称空间

rootElement.**setAttributeNS**(namespace, qualifiedName, value);

**//DOM4J**

//1.创建document对象，代表整个xml文档

Document document = DocumentHelper.createDocument();

//2.创建根节点rss

Element rss = document.addElement("rss");

//3.向rss节点中添加version属性

rss.addAttribute("version", "2.0");

//4.生成子节点及节点内容

Element channel = rss.addElement("channel");

Element title = channel.addElement("title");

title.setText("<![CDATA[上海移动互联网产业促进中心正式揭牌 ]]>");

4、**写出树的内容到XML**

**1）DOM原生**

//不对DOM树做任何修改的转换器

Transformer t =TransformerFactory.newInstance().newTransformer();

//为了将DOCTYPE节点纳入输出，还需将SYSTEM和PUBLIC标识符设置为输出属性

t.**setOutputProperty**(OutputKeys.DOCTYPE\_SYSTEM, systemIdentifier);

t.setOutputProperty(OutputKeys.DOCTYPE\_PUBLIC,publicIdentifier);

//设置缩进规则

t.setOutputProperty(OutputKeys.INDENT, “yes”);

t.setOutputProperty(OutputKeys.METHOD, “xml”);

t.setOutputProperty(“{http://xml.apache.org/xslt}indent-amount”, “2”);

//应用不做任何修改的转换操作，开始转换

t.**transform**(new DOMSource(**doc**),new StreamResult(new FileOutputStream(**file**)));

**//DOM4J**

OutputFormat format = OutputFormat.createPrettyPrint();

format.setEncoding("GBK");

// 生成xml文件

File file = new File("rssnews.xml");

XMLWriter writer = new XMLWriter(new FileOutputStream(file), format);

//设置是否转义，默认值是true，代表转义

writer.setEscapeText(false);

writer.write(document);

writer.close();

**2）使用LSSerializer接口**

//获取LSSerializer

DOMImplementation impl = doc.getImplementation();

DOMImplementationLS implLS = (DOMImplementationLS)impl.getFeature(“LS”,”3.0”);

LSSerializer ser = implLS.createLSSerializer();

//需要空格和换行，设置

ser.getDomConfig().setParameter(“format-pretty-print”, true);

//可以将XML文档转换为字符串

String str = ser.writeToString(**doc**);

//如果想将输出直接写入文件中，需要一个LSOutput

LSOutput out = implLS.createLSOutput();

out.setEncoding(“UTF-8”);

out.setByteStream(Files.newOutputStream(path)); //指定输出到哪个文件，不存在就创建

ser.write(**doc**, **out**);

**3)使用XSLT转换**

XSL转换机制可以**指定**将XML文档转换为其他格式的**规则**，例如，txt纯文本、XHTML或其他任何XML格式。

用途：

XSLT通常用来将某种机器可读的XML格式转译为另一种机器可读的XML格式，或者将XML转译为适合人类阅读的表示格式

步骤：

1. 需要提供**XSLT样式表**，它描述了XML文档向某种格式转换的规则
2. XSLT处理器读入**XML文档**和**XSLT样式表**，产生所要的输出

**XSLT样式表：**

<?xml version=”1.0” encoding=”utf-8”?>

<xsl:stylesheet

Xmlns:xsl=<http://www.w3c.org/1999/XSL/Transform>

Version=”1.0”>

<xsl:output method=”html”/> //指定输出的格式为HTML，其他有XML、text

Template1 模板

.

.

.

Template2

</xsl:stylesheet>

**模板示例:**

<xsl:template match=”/staf/employee”>

<tr><xsl:apply-templates/></tr>

</xsl:template>

match属性的值是一个XPath表达式

该模板表示：每当看懂啊XPath集/staff/employee中的一个节点时

1. 产生字符串<tr>
2. 对于要处理的**子元素**继续应用模板（会递归！）
3. 当处理完所有所有子元素后，产生字符串</tr>

把属性值复制到输出中的模板：

<xsl:template match=”/staff/employee/hiredate”>

<td><xsl:**value-of** **select**=”@year”/>-<xsl:value-of select=”@month”/>-<xsl:value-of select=”@day”/></td>

</xsl:template>

这里xsl:value-of语句用于计算，由select的XPath值指定的**节点集（属性节点）**的，字符串值

**XML转HTML示例：**

<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

version="1.0">

<xsl:output method="html"/> //指定输出为html格式

<xsl:template match="/staff"> //使用XPath来分别定位

<table border="1"><xsl:apply-templates/></table>

</xsl:template>

<xsl:template match="/staff/employee"> //使用XPath来分别定位

<tr><xsl:apply-templates/></tr>

</xsl:template>

<xsl:template match="/staff/employee/name"> //使用XPath来分别定位

<td><xsl:apply-templates/></td>

</xsl:template>

<xsl:template match="/staff/employee/salary"> //使用XPath来分别定位

<td>$<xsl:apply-templates/></td>

</xsl:template>

<xsl:template match="/staff/employee/hiredate"> //使用XPath来分别定位

<td><xsl:value-of select="@year"/>-<xsl:value-of

select="@month"/>-<xsl:value-of select="@day"/></td>

</xsl:template>

</xsl:stylesheet>

**转纯文本示例（Properties类键值对形式）:**

<?xml version="1.0"?>

<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

version="1.0">

<xsl:output method="text" omit-xml-declaration="yes"/>

<xsl:template match="/staff/employee">

employee.<xsl:value-of select="position()" //注意中间生造了文本employee.

/>.name=<xsl:value-of select="name/text()"/>

employee.<xsl:value-of select="position()"

/>.salary=<xsl:value-of select="salary/text()"/>

employee.<xsl:value-of select="position()"

/>.hiredate=<xsl:value-of select="hiredate/@year"

/>-<xsl:value-of select="hiredate/@month"

/>-<xsl:value-of select="hiredate/@day"/>

</xsl:template>

</xsl:stylesheet>

**Java平台下的XSL转换（XSLT）：**

（1）为每个样式表设置一个转换器工厂，得到一个转换器对象，并让它把一个**源**转换成结果

File styleSheet = new File(fileName); //某个xsl文件

StreamSource styleSource = new StreamSource(styleSheet);

Transformer t = TransformerFactory.newInstance().newTransformer(styleSource);

t.transform(source, result);

（2）谈谈source和 result

Source接口有四个实现类：DOMSource、SAXSource、StAXSource、StreamSource，这些类使我们能通过文件、流、Reader、URL、DOM树节点来传入一个source

比如：来自DOM树节点的DOMSource（传入Document就相当于整个树）、

Result接口有三个实现类：DOMResult、SAXResult、StreamResult

如果要把转换结果存储到DOM树中，那就再创建一个新的文档节点包装到DOMResult中：

Document doc = builder.newDocument();

t.transform(source, new DOMResult(doc));

如果要把转换结果保存到文件中，可以使用StreamResult

t.transform(source, new StreamResult(file));

**题外话： 将非XML的遗留数据转换成XML的一个小技巧**

Source中的SAXSource可以 从XMLReader接口的SAX实现获得！可以不必从一个现有XML文件开始工作。

先上代码：

t.transform(new SAXSource(new EmployeeReader(), new InputSource(new FileInputStream(filename))), result);

其中EmployeeReader就是XMLReader的SAX实现类，InputSource就是文件来源，result决定保存的形式