JAXP的DOM解析方式

1. **JAXP（Java API for XMLProcessing，意为XML处理的Java API）是Java XML程序设计的应用程序接口之一，它提供解析和验证XML文档的能力。**

**JAXP解析XML的三种基本接口为：**

**（1）文档对象模型解析接口或DOM接口**

**（2）XML简单API解析接口或SAX接口**

**（3）XML流API或StAX接口（是JDK 6的一部分，为JDK 5提供单独的包）。**

**除了解析接口，JAXP还提供了XSLT接口用来对XML文档进行数据和结构的转换。**

1. **JAXP的DOM解析方式的基本使用**



示例：

DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();

DocumentBuilder db = dbf.newDocumentBuilder();

Document document = (Document) db.parse(**new** File("stu3.xml"));

Element rootElement = document.getDocumentElement();

1. java已经把每个节点类型都定为一个short值，且由于数值比较难记，因此就把每个节点的值封装成属性（Field）。
2. 在利用**Document的getElementById方法**获取对象时，jvm不认可xml中的人为定义的**id属性**，必须利用Element的**setIdAttribute方法**进行设置某个属性作为**JVM认可的Id属性**。

**示例1：设置id为Id**

NodeList nodeList = rootElement.getElementsByTagName("user");

//设置dd属性为ID

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

((Element) nodeList.item(i)).setIdAttribute("id", **true**);

}

//通过ID属性获取Element对象

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

System.***out***.println(document.getElementById("00"+i).getAttribute("dd"));}

**示例2：设置dd为Id**

NodeList nodeList = rootElement.getElementsByTagName("user");

//设置dd属性为ID

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

((Element) nodeList.item(i)).setIdAttribute("dd", **true**);

}

//通过ID属性获取Element对象

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

System.***out***.println(document.getElementById(Character.*toString*((**char**)('k'+i))).getAttribute("id")); }

1. 添加一个Element对象：

**private** **static** **void** addUser(Document document) {

Element user = document.createElement("user");

Element age = document.createElement("age");

Element name = document.createElement("name");

Element sex = document.createElement("sex");

user.setAttribute("id", "006");

user.setAttribute("dd", "q");

age.setTextContent("88");

name.setTextContent("xixixxix");

sex.setTextContent("MALE");

user.appendChild(name);

user.appendChild(age);

user.appendChild(sex);

document.getDocumentElement().appendChild(user);

}

1. 删除一个Element对象：

前提是设置Id。、

NodeList nodeList = rootElement.getElementsByTagName("user");

// 设置dd属性为ID

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

((Element) nodeList.item(i)).setIdAttribute("id", **true**);

}

private static void deleteElement(Document document, String id) {

Element delete = document.getElementById(id);

Element root = document.getDocumentElement();

root.removeChild(delete);

}

1. 把修改后的**DOM树**写到指定文件中去，或者直接更新原文件。

**private** **static** **void** writeToFile(Document document, File destFile)

**throws** Exception{

TransformerFactory tf = TransformerFactory.*newInstance*();

Transformer transformer = tf.newTransformer();

//使用转换对象的transform方法，需要传入两个对象，

//一个是DOMSource，另一个是StreamResult

//DOMSource是把document封装一下，而StreamResult是把目的地封装一下。

DOMSource domsource = **new** DOMSource(document);

StreamResult sr = **new** StreamResult(**new** FileWriter(destFile));

transformer.setOutputProperty(OutputKeys.***METHOD***,"xml");

transformer.setOutputProperty(OutputKeys.***INDENT***,"yes");

transformer.transform(domsource, sr);

}

1. JAXP的SAX解析方式：



