什么是XML：

* XML 于 1998 年 2 月 10 日成为 W3C 的推荐标准。
* XML 指可扩展标记语言（EXtensible Markup Language）。
* XML 是一种很像HTML的标记语言。
* XML 的设计宗旨是传输数据，而不是显示数据。
* XML 标签没有被预定义。您需要自行定义标签。
* XML 被设计为具有自我描述性。
* XML 是 W3C 的推荐标准。

XML语法规则：

<root> <child> <subchild>.....</subchild> </child> </root>

示例：

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

<!-- This is a comment --> <note> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Don't forget me this weekend!</body> </note>

命名规则：

* 名称可以包含字母、数字以及其他的字符
* 名称不能以数字或者标点符号开始
* 名称不能以字母 xml（或者 XML、Xml 等等）开始
* 名称不能包含空格

特点：

* 标签必须正确嵌套
* 标签必须关闭
* 区分大小写
* 节点中属性的值必须加引号

实体引用：

在 XML 中，一些字符拥有特殊的意义。

|  |  |  |
| --- | --- | --- |
| &lt; | < | 小于 |
| &gt: | > | 大于 |
| &amp: | & | 与 |
| &apos; | ' | 单引号 |
| &quot; | " | 引号 |

XML CDATA：

XML 文档中的所有文本均会被解析器解析。只有 CDATA 区段中的文本会被解析器忽略。

CDATA 部分由 "**<![CDATA[**" 开始，由 "**]]>**" 结束：

**<！[CDATA[** 2>3 </font> **]]>**

DOM 解析：

是将xml文件全部载入，组装成一颗dom树，然后通过节点以及节点之间的关系来解析xml文件或将xml以树结构的方式一次性加载进内存，易用，修改方便，但是更占内存。不适应于大文件，一般情况下用它即可应付大多数应用。 使用DOM创建XML:

/\*\*

\* DOM方式创建XML

\*

\* @throws ParserConfigurationException

\* @throws TransformerFactoryConfigurationError

\* @throws FileNotFoundException

\* @throws TransformerException

\*/

static void fun1() throws ParserConfigurationException, TransformerFactoryConfigurationError, FileNotFoundException,

TransformerException {

// 文档解析器工厂

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// 文档解析器

DocumentBuilder builder = factory.newDocumentBuilder();

// xml文档对象

Document document = builder.newDocument();

// xml文件路径

String path = "src/com/kaige/xml/newdata.xml";

// 根节点

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

// 创建5个学生节点

for (int i = 0; i < 10; i++) {

Element student = document.createElement("Student");

student.setAttribute("id", "stu" + i); // 添加属性

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

name.setAttribute("nameid", "name" + i);// 添加属性

name.appendChild(document.createTextNode("姓名" + i));// 追加子节点

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

age.appendChild(document.createTextNode(18 + i + ""));

// 将age和name加入到student

student.appendChild(name);

student.appendChild(age);

// 将student加入到root

classinfo.appendChild(student);

}

// 将classinfo加入到document

document.appendChild(classinfo);

// 将Document对象转换为DOMSource对象

DOMSource source = new DOMSource(document);

//变压器

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

tf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");// 设置编码格式

tf.setOutputProperty(OutputKeys.INDENT, "yes"); // 设置缩进

// 结果流

StreamResult sr=new StreamResult(filename);

//写入 数据

tf.transform(source, sr);

System.out.println("创建xml成功");

}

使用DOM读取XML:

/\*\*

\* DOM方式读取XML

\*

\* @throws ParserConfigurationException

\* @throws IOException

\* @throws SAXException

\*/

static void fun2() throws ParserConfigurationException, SAXException, IOException {

// 文档解析器工厂

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// 文档解析器

DocumentBuilder builder = factory.newDocumentBuilder();

// 文档对象

Document document = builder.parse("src/com/kaige/xml/newdata.xml");

NodeList nodeList= document.getChildNodes();

//System.out.println("班级数量："+nodeList.getLength());

//根节点

Node classinfo=nodeList.item(0);

//获取学生节点

NodeList studentlist= classinfo.getChildNodes();

//遍历学生节点

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

Node node=studentlist.item(i);

if(node instanceof Text)continue;

System.out.println("学号："+node.getAttributes().item(0).getNodeValue());

NodeList fieldlist= node.getChildNodes();

for (int j = 0; j < fieldlist.getLength(); j++) {

Node fieldNode=fieldlist.item(j);

if(fieldNode instanceof Text)continue;

System.out.println(fieldNode.getNodeName() +":"+fieldNode.getTextContent());

}

}

System.out.println("---------------------------");

}

}