XML总结

XML简介：

XML（EXtensible Markup Language）,可扩展标记语言

特点：

XML与操作系统、编程语言的开发平台都无关【通用的数据交换格式】

实现不同系统之间的数据交换【包括系统的配置文件】

是一种标记语言，没有固定的标记，可以随便发明也可以自己创建

【整个XML是一种树形文件】

是其他技术的基础技术

作用：

数据交互

配置应用程序和网站

Ajax基石

XML文档结构：

<?xml version="1.0" encoding="UTF-8"?> 【声明部分（位于第一行）；这个是满足xml 1.0标准的】

【底下的都是：文档元素描述信息（文档结构），所有带尖括号的都叫标签，也叫元素，都是用树形文件写出来的】

<books> 【根元素，每个XML文件的根元素有且仅有一个】

<!--图书信息 --> 【XML中的注释语法：<!--……-->】

<book id="bk101"> 【books的子元素，包含关系的元素要用缩进来体现】

<author>王珊</author> 【book的子元素】

<title>.NET高级编程</title> 【就像以前的类和属性】

<description>包含C#框架和网络编程等</description>

</book>

<book id="bk102"> 【books的子元素】

<author>李明明</author>

<title>XML基础编程</title>

<description>包含XML基础概念和基本作用</description>

</book>

</books> 【根元素</……>是结束的符号】

【数据库，Java文件，XML文档应当全部一致】

XML文档内容由一系列标签元素组成

<元素名 属性名=“属性值”>元素内容</元素名>

空元素的方法：

<name>（这里是一个空格）</name>

<name></name>

<name/>

语法：

属性值用双引号包裹

一个元素可以有多个属性

属性值中不能直接包含<、“、&（不建议：‘、>）【最好都不要写】

例如：

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

<students>

<!-- 学生信息 -->

<student id="s1">

<name>张三</name>

<age>18</age>

</student>

<student id="s2">

<name>李四</name>

<age>22</age>

</student>

<!-- 这两种写法都可以，但是第一种用的居多 -->

<student id="s1"name="张三"age="18"/>

<student id="s2"name="李四"age="22"/>

</students>

XML编写注意事项：

所有XML元素都必须有结束标签（</……>）

XML标签对大小写敏感

XML必须正确的嵌套

同级标签以缩进对齐

元素名称可以包含字母、数字或其他的字符

元素名称不能以数字或者标点符号开始（<5name>是错误的；<name5>是正确的）

元素名称中不能含空格（<student name>是错误的；<studentname>是正确的的）

推荐的XML编写命名习惯：

编写的元素名称要有描述性。

名字尽量简短些，可以用下划线“\_”，但是不要用中横线“-”、点“.”或者冒号“:”（<book\_title>）。

数据库怎么命名，XML文件就怎么命名，要保持一致。

指出下面XML代码的错误：

<title><name>XML编程</title></name>

嵌套错误，应该是：

<title><name>XML编程</name></title>

当元素中出现很多特殊字符时，可以使用CDATA节，如：

<description><![CDATA[讲解了元素<title>以及</title>的使用]]>

等同于：

<description><讲解了元素&lt;title&gt;以及&lt;/title&gt;的使用> </description>

XML命名空间（类似于Java中的包）：

XML命名空间的作用:

避免命名冲突

解决方法：

使用前缀来避免

例如：

此文档带有某个表格中的信息：

<h:table>

<h:tr>

<h:td>Apples</h:td>

<h:td>Bananas</h:td>

</h:tr>

</h:table>

此 XML 文档携带着有关一件家具的信息：

<f:table>

<f:name>African Coffee Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>

</f:table>

现在，命名冲突不存在了，这是由于两个文档都使用了不同的名称来命名它们的 <table> 元素 (<h:table> 和 <f:table>)。

通过使用前缀，我们创建了两种不同类型的 <table> 元素。

使用命名空间（Namespaces）

例如：

这个 XML 文档携带着某个表格中的信息：

<h:table xmlns:h="http://www.w3.org/TR/html4/">

<h:tr>

<h:td>Apples</h:td>

<h:td>Bananas</h:td>

</h:tr>

</h:table>

此 XML 文档携带着有关一件家具的信息：

<f:table xmlns:f="http://www.w3school.com.cn/furniture">

<f:name>African Coffee Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>

</f:table>

与仅仅使用前缀不同，我们为 <table> 标签添加了一个xmlns属性，里面的网址是统一资源标识符，目的是唯一标识一个命名空间，每一个统一资源标识符都是不一样的，但是没有实际意义，但是一般都能连到一个网页，取值这样就为前缀赋予了一个与某个命名空间相关联的限定名称。

XML解析器：

非验证解析器【检查文档格式是否良好，不能验证是够有效】

验证解析器【使用DTD（Document Type Define是一种验证机制）检查文档的有效性】

解析XML技术：

DOM：

基于XML文档树结构的解析（解析成DOM树放内存里）

适用于多次访问的XML文档

特点：比较消耗资源

SAX：

基于事件的解析

适用于大数据量的XML文档

特点：占用资源少，内存消耗小

DOM4J（把DOM解析进行了一次封装，应用更多）：

非常优秀的Java XML API

性能优异、功能强大

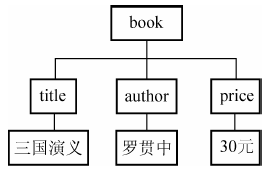
开放源代码

DOM解析XML：

DOM介绍

文档对象模型（Document Object Model）

DOM把XML文档映射成一个倒挂的树



示例：

<book id=”1234”>

<title>三国演义</title>

<author>罗贯中</author>

<price>30元</price>

</book>

其中所有带尖括号的叫元素节点，只有文本文字的叫属性节点，id=”1234”叫做属性节点

变成：

更易于增删改查的应用

常用接口介绍：

Document：表示整个 XML 文档

NodeList getElementsByTagName(String Tag)

按文档顺序返回文档中指定标记名称的所有元素集合

Element createElement(String tagName)

创建指定标记名称的元素

Node：该文档树中的单个节点

NodeList getChildNodes()

获取该元素的所有子节点，返回节点集合

Element：XML 文档中的一个元素（是一种特殊的Node）

String getTagName()

获取元素名称

DOM解析XML文件步骤：

创建解析器工厂对象

解析器工厂对象创建解析器对象

解析器对象指定XML文件创建Document对象

以Document对象为起点操作DOM树

DOM4J：

开源，易用，应用于Java平台上的一种解析XML的技术，提供第三方的jar包使用了大量的接口。

其中：

Document：定义XML文档（和DOM一样）

Element：定义XML元素

Text：定义XML文本节点

Attribute：定义了XML 的属性

第一步：拖进来jar包选择build path之后add build path，就会变成奶瓶。

然后再写开始初始化信息：

public Document parse(String url) throws DocumentException {

SAXReader reader = new SAXReader();

Document document = reader.read(url);

return document;

解析文件：

try {

parse("animals.xml");

} catch (DocumentException e) {

System.out.println("文件找不到");

}

对文件进行操作：

//先得到根节点

Element root=doc.getRootElement();

//遍历根元素的所有子节点

Iterator it= root.elementIterator();

while(it.hasNext()){

//得到每个user的所有子节点

Element user=(Element)it.next();

//其他操作

String XXX=user.getText();//得到子节点的文本

String XXX=user.attributeValue("id");//得到子节点的属性

user.addElement("name").addText(name);//给子节点添加文本

user.addAttribute("id", id);//添加属性

root.remove(user);//删除节点服务器上的 XML

XML 文件是类似 HTML 文件的纯文本文件。

XML 能够通过标准的 Web 服务器轻松地存储和生成。

在服务器上存储 XML 文件

XML 文件在 Internet 服务器上进行存储的方式与 HTML 文件完全相同。

启动 Windows 记事本，并写入以下行：

<?xml version="1.0" encoding="ISO-8859-1"?>  
<note>  
<from>Jani</from>  
<to>Tove</to>  
<message>Remember me this weekend</message>  
</note>

然后用适当的文件名，比如 "note.xml"，在 Web 服务器上保存这个文件。

通过 ASP 生成 XML

XML 可在不安装任何 XML 软件的情况下在服务器端生成。

如需从服务器生成 XML 响应 - 只需简单地编写以下代码并在 Web 服务器上把它保存为一个 ASP 文件：

<%  
response.ContentType="text/xml"  
response.Write("<?xml version='1.0' encoding='ISO-8859-1'?>")  
response.Write("<note>")  
response.Write("<from>Jani</from>")  
response.Write("<to>Tove</to>")  
response.Write("<message>Remember me this weekend</message>")  
response.Write("</note>")  
%>

通过 PHP 生成 XML

如需使用 PHP 从服务器上生成 XML 响应，请使用下面的代码：

<?php  
header("Content-type: text/xml");  
echo "<?xml version='1.0' encoding='ISO-8859-1'?>";  
echo "<note>";  
echo "<from>Jani</from>";  
echo "<to>Tove</to>";  
echo "<message>Remember me this weekend</message>";  
echo "</note>";  
?>

请注意，响应头部的内容类型必须设置为 "text/xml"。从数据库生成 XML

XML 可在不安装任何 XML 软件的情况下从数据库生成。

如需从服务器生成 XML 数据库响应，只需简单地编写以下代码，并把它在 Web 服务器上保存为 ASP 文件：

<%  
response.ContentType = "text/xml"  
set conn=Server.CreateObject("ADODB.Connection")  
conn.provider="Microsoft.Jet.OLEDB.4.0;"  
conn.open server.mappath("/db/database.mdb")  
  
sql="select fname,lname from tblGuestBook"  
set rs=Conn.Execute(sql)  
  
response.write("<?xml version='1.0' encoding='ISO-8859-1'?>")  
response.write("<guestbook>")  
while (not rs.EOF)  
response.write("<guest>")  
response.write("<fname>" & rs("fname") & "</fname>")  
response.write("<lname>" & rs("lname") & "</lname>")  
response.write("</guest>")  
rs.MoveNext()  
wend  
  
rs.close()  
conn.close()  
response.write("</guestbook>")  
%>

在服务器上通过 XSLT 转换 XML

下面的 ASP 代码在服务器上把 XML 文件转换为 XHTML：

<%  
'Load XML  
set xml = Server.CreateObject("Microsoft.XMLDOM")  
xml.async = false  
xml.load(Server.MapPath("simple.xml"))  
  
'Load XSL  
set xsl = Server.CreateObject("Microsoft.XMLDOM")  
xsl.async = false  
xsl.load(Server.MapPath("simple.xsl"))  
  
'Transform file  
Response.Write(xml.transformNode(xsl))  
%>

实例解释

* 第一个代码块创建微软 XML 解析器的实例（XMLDOM），并把 XML 文件载入内存。
* 第二个代码块创建解析器的另一个实例，并把 XSL 文件载入内存。
* 最后一个代码使用 XSL 文档来转换 XML 文档，并把结果以 XHTML 发送到您的

## 通过 ASP 把 XML 保存为文件

这个 ASP 实例会创建一个简单的 XML 文档，并把该文档保存到服务器上：

<%  
text="<note>"  
text=text & "<to>Tove</to>"  
text=text & "<from>Jani</from>"  
text=text & "<heading>Reminder</heading>"  
text=text & "<body>Don't forget me this weekend!</body>"  
text=text & "</note>"  
  
set xmlDoc=Server.CreateObject("Microsoft.XMLDOM")  
xmlDoc.async=false  
xmlDoc.loadXML(text)  
  
xmlDoc.Save("test.xml")  
%>

**实例：XML 气象服务**

XML 国家气象服务案例，来自 NOAA（National Oceanic and Atmospheric Administration）：

<?xml version="1.0" encoding="ISO-8859-1" ?>  
<current\_observation>  
  
<credit>NOAA's National Weather Service</credit>  
<credit\_URL>http://weather.gov/</credit\_URL>  
  
<image>  
<url>http://weather.gov/images/xml\_logo.gif</url>  
<title>NOAA's National Weather Service</title>  
<link>http://weather.gov</link>  
</image>  
  
<location>New York/John F. Kennedy Intl Airport, NY</location>  
<station\_id>KJFK</station\_id>  
<latitude>40.66</latitude>  
<longitude>-73.78</longitude>  
<observation\_time\_rfc822>Mon, 11 Feb 2008 06:51:00 -0500 EST  
</observation\_time\_rfc822>  
  
<weather>A Few Clouds</weather>  
<temp\_f>11</temp\_f>  
<temp\_c>-12</temp\_c>  
<relative\_humidity>36</relative\_humidity>  
<wind\_dir>West</wind\_dir>  
<wind\_degrees>280</wind\_degrees>  
<wind\_mph>18.4</wind\_mph>  
<wind\_gust\_mph>29</wind\_gust\_mph>  
<pressure\_mb>1023.6</pressure\_mb>  
<pressure\_in>30.23</pressure\_in>  
<dewpoint\_f>-11</dewpoint\_f>  
<dewpoint\_c>-24</dewpoint\_c>  
<windchill\_f>-7</windchill\_f>  
<windchill\_c>-22</windchill\_c>  
<visibility\_mi>10.00</visibility\_mi>  
  
<icon\_url\_base>http://weather.gov/weather/images/fcicons/</icon\_url\_base>  
<icon\_url\_name>nfew.jpg</icon\_url\_name>  
<disclaimer\_url>http://weather.gov/disclaimer.html</disclaimer\_url>  
<copyright\_url>http://weather.gov/disclaimer.html</copyright\_url>  
  
</current\_observation>

总结：

XML 可用于交换、共享和存储数据。

XML 文档形成 [树状结构](https://www.runoob.com/xml/xml-tree.html)，在"根"和"叶子"的分支机构开始的。

XML 有非常简单的 [语法规则](https://www.runoob.com/xml/xml-syntax.html)。带有正确语法的 XML 是"形式良好"的。有效的 XML 是针对 [DTD](https://www.runoob.com/xml/xml-dtd.html) 进行验证的。

[XSLT](https://www.runoob.com/xml/xml-xsl.html) 用于把 XML 转换为其他格式，比如 HTML。

所有现代的浏览器有一个内建的 [XML 解析器](https://www.runoob.com/xml/xml-parser.html)，可读取和操作 XML。

[DOM](https://www.runoob.com/xml/xml-dom.html)（Document Object Model）定义了一个访问 XML 的标准方式。

[XMLHttpRequest](https://www.runoob.com/xml/xml-http.html) 对象提供了一个网页加载后与服务器进行通信的方式。

[XML 命名空间](https://www.runoob.com/xml/xml-namespaces.html)提供了一种避免元素命名冲突的方法。

[CDATA](https://www.runoob.com/xml/xml-cdata.html) 区域内的文本会被解析器忽略。