# Xml学习笔记

## 什么是 XML?

* XML 指可扩展标记语言（EXtensible Markup Language）
* XML 是一种标记语言，很类似 HTML
* XML 的设计宗旨是传输数据，而非显示数据
* 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 与 HTML 的主要差异

XML 不是 HTML 的替代。

XML 和 HTML 为不同的目的而设计：

XML 被设计为传输和存储数据，其焦点是数据的内容。

HTML 被设计用来显示数据，其焦点是数据的外观。

HTML 旨在显示信息，而 XML 旨在传输信息。

## XML 不是对 HTML 的替代

XML 是对 HTML 的补充。

XML 不会替代 HTML，理解这一点很重要。在大多数 web 应用程序中，XML 用于传输数据，而 HTML 用于格式化并显示数据。

对 XML 最好的描述是：

XML 是独立于软件和硬件的信息传输工具。

## XML 简化数据传输

通过 XML，可以在不兼容的系统之间轻松地交换数据。

对开发人员来说，其中一项最费时的挑战一直是在因特网上的不兼容系统之间交换数据。

由于可以通过各种不兼容的应用程序来读取数据，以 XML 交换数据降低了这种复杂性。

# XML 树结构

XML 使用简单的具有自我描述性的语法：

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

<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

</note>

第一行是 XML 声明。它定义 XML 的版本 (1.0) 和所使用的编码 (ISO-8859-1 = Latin-1/西欧字符集)。

下一行描述文档的根元素（像在说：“本文档是一个便签”）：

<note>

接下来 4 行描述根的 4 个子元素（to, from, heading 以及 body）：

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

最后一行定义根元素的结尾：

</note>

从本例可以设想，该 XML 文档包含了 John 给 George 的一张便签。

## XML 文档形成一种树结构

XML 文档必须包含根元素。该元素是所有其他元素的父元素。

XML 文档中的元素形成了一棵文档树。这棵树从根部开始，并扩展到树的最底端。

所有元素均可拥有子元素：

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

父、子以及同胞等术语用于描述元素之间的关系。父元素拥有子元素。相同层级上的子元素成为同胞（兄弟或姐妹）。

所有元素均可拥有文本内容和属性（类似 HTML 中）。

# XML 语法规则

**XML 的语法规则很简单，且很有逻辑。这些规则很容易学习，也很容易使用。**

## 所有 XML 元素都须有关闭标签

在 HTML，经常会看到没有关闭标签的元素：

在 XML 中，省略关闭标签是非法的。所有元素都必须有关闭标签：允许有单标签。如下：



如果标签中的大段内容有许多特殊符号，就用<!CDATA[]>来表示



## XML 标签对大小写敏感

XML 元素使用 XML 标签进行定义。

XML 标签对大小写敏感。在 XML 中，标签 <Letter> 与标签 <letter> 是不同的。

必须使用相同的大小写来编写打开标签和关闭标签：

<Message>这是错误的。</message>

<message>这是正确的。</message>

## XML 必须正确地嵌套

## XML 文档必须有根元素

XML 文档必须有一个元素是所有其他元素的父元素。该元素称为根元素。

## XML 的属性值须加引号

与 HTML 类似，XML 也可拥有属性（名称/值的对）。

在 XML 中，XML 的属性值须加引号。例如：

<note date="08/08/2008">

<to>George</to>

<from>John</from>

</note>

**Note 属性值需要引号。**

## 实体引用

在 XML 中，一些字符拥有特殊的意义。

如果你把字符 "<" 放在 XML 元素中，会发生错误，这是因为解析器会把它当作新元素的开始。

这样会产生 XML 错误：

<message>if salary < 1000 then</message>

为了避免这个错误，请用实体引用来代替 "<" 字符：

<message>if salary &lt; 1000 then</message>

在 XML 中，有 5 个预定义的实体引用：

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

注释：在 XML 中，只有字符 "<" 和 "&" 确实是非法的。大于号是合法的，但是用实体引用来代替它是一个好习惯。

## XML 中的注释

在 XML 中编写注释的语法与 HTML 的语法很相似：

<!-- This is a comment -->

## XML 中，空格会被保留

HTML 会把多个连续的空格字符裁减（合并）为一个：

HTML: Hello my name is David.

输出: Hello my name is David.

在 XML 中，文档中的空格不会被删节。

# XML 元素

**XML 文档包含 XML 元素。**

## 什么是 XML 元素？

XML 元素指的是从（且包括）开始标签直到（且包括）结束标签的部分。

元素可包含其他元素、文本或者两者的混合物。元素也可以拥有属性。

有的标签包含元素内容，有的标签只有文本，只包含文本内容。

## XML 命名规则

XML 元素必须遵循以下命名规则：

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

可使用任何名称，没有保留的字词。

## 最佳命名习惯

使名称具有描述性。使用下划线的名称也很不错。

名称应当比较简短，比如：<book\_title>，而不是：<the\_title\_of\_the\_book>。

避免 "-" 字符。如果您按照这样的方式进行命名："first-name"，一些软件会认为你需要提取第一个单词。

避免 "." 字符。如果您按照这样的方式进行命名："first.name"，一些软件会认为 "name" 是对象 "first" 的属性。

避免 ":" 字符。冒号会被转换为命名空间来使用（稍后介绍）。

XML 文档经常有一个对应的数据库，其中的字段会对应 XML 文档中的元素。有一个实用的经验，即使用数据库的名称规则来命名 XML 文档中的元素。

## XML 元素是可扩展的

XML 元素是可扩展，以携带更多的信息。

# XML 属性

**XML 元素可以在开始标签中包含属性，类似 HTML。**

**属性 (Attribute) 提供关于元素的额外（附加）信息。**

## XML 属性

从 HTML，你会回忆起这个：<img src="computer.gif">。"src" 属性提供有关 <img> 元素的额外信息。

在 HTML 中（以及在 XML 中），属性提供有关元素的额外信息：

<img src="computer.gif">

<a href="demo.asp">

属性通常提供不属于数据组成部分的信息。在下面的例子中，文件类型与数据无关，但是对需要处理这个元素的软件来说却很重要：

<file type="gif">computer.gif</file>

## XML 属性必须加引号

属性值必须被引号包围，不过单引号和双引号均可使用。比如一个人的性别，person 标签可以这样写：

<person sex="female">

或者这样也可以：

<person sex='female'>

注释：如果属性值本身包含双引号，那么有必要使用单引号包围它，就像这个例子：

<gangster name='George "Shotgun" Ziegler'>

或者可以使用实体引用：

<gangster name="George &quot;Shotgun&quot; Ziegler">

## XML 元素 vs. 属性

请看这些例子：

<person sex="female">

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

<person>

<sex>female</sex>

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

在第一个例子中，sex 是一个属性。在第二个例子中，sex 则是一个子元素。两个例子均可提供相同的信息。

没有什么规矩可以告诉我们什么时候该使用属性，而什么时候该使用子元素。我的经验是在 HTML 中，属性用起来很便利，但是在 XML 中，您应该尽量避免使用属性。如果信息感觉起来很像数据，那么请使用子元素吧。

## 避免 XML 属性？

因使用属性而引起的一些问题：

* 属性无法包含多重的值（元素可以）
* 属性无法描述树结构（元素可以）
* 属性不易扩展（为未来的变化）
* 属性难以阅读和维护

请尽量使用元素来描述数据。而仅仅使用属性来提供与数据无关的信息。

## 针对元数据的 XML 属性

有时候会向元素分配 ID 引用。这些 ID 索引可用于标识 XML 元素，它起作用的方式与 HTML 中 ID 属性是一样的。

元数据（有关数据的数据）应当存储为属性，而数据本身应当存储为元素。

# XML 验证

**拥有正确语法的 XML 被称为“形式良好”的 XML。**

**通过 DTD 验证的 XML 是“合法”的 XML。**

## 形式良好的 XML 文档

**“形式良好”或“结构良好”的 XML 文档拥有正确的语法。**

“形式良好”（Well Formed）的 XML 文档会遵守前几章介绍过的 XML 语法规则：

* XML 文档必须有根元素
* XML 文档必须有关闭标签
* XML 标签对大小写敏感
* XML 元素必须被正确的嵌套
* XML 属性必须加引号

# XML 验证器

## XML 错误会终止您的程序

XML 文档中的错误会终止你的 XML 程序。

W3C 的 XML 规范声明：如果 XML 文档存在错误，那么程序就不应当继续处理这个文档。理由是，XML 软件应当轻巧，快速，具有良好的兼容性。

如果使用 HTML，创建包含大量错误的文档是有可能的（比如你忘记了结束标签）。其中一个主要的原因是 HTML 浏览器相当臃肿，兼容性也很差，并且它们有自己的方式来确定当发现错误时文档应该显示为什么样子。

使用 XML 时，这种情况不应当存在

***DOCTYPE 声明：***

**<!DOCTYPE note [**

**<!ELEMENT note (to,from,heading,body)>**

**<!ELEMENT to (#PCDATA)>**

**<!ELEMENT from (#PCDATA)>**

**<!ELEMENT heading (#PCDATA)>**

**<!ELEMENT body (#PCDATA)>**

**]>**

**添加到xml中：**

**<?xml version="1.0" ?>**

**<!DOCTYPE note [**

**<!ELEMENT note (to,from,heading,body)>**

**<!ELEMENT to (#PCDATA)>**

**<!ELEMENT from (#PCDATA)>**

**<!ELEMENT heading (#PCDATA)>**

**<!ELEMENT body (#PCDATA)>**

**]>**

**<note>**

**<to>George</to>**

**<from>John</from>**

**<heading>Reminder</heading>**

**<body>Don't forget the meeting!</body>**

**</note>**

# 使用 CSS 显示 XML

**通过使用 CSS，可为 XML 文档添加显示信息。**

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

<?xml-stylesheet type="text/css" href="cd\_catalog.css"?>

# XMLHttpRequest 对象

## 什么是 XMLHttpRequest 对象？

XMLHttpRequest 对象用于在后台与服务器交换数据。

XMLHttpRequest 对象是开发者的梦想，因为您能够：

* 在不重新加载页面的情况下更新网页
* 在页面已加载后从服务器请求数据
* 在页面已加载后从服务器接收数据
* 在后台向服务器发送数据

所有现代的浏览器都支持 XMLHttpRequest 对象。

## 解析 XML 文档

## 创建 XMLHttpRequest 对象

所有现代浏览器 (IE7+、Firefox、Chrome、Safari 以及 Opera) 都内建了 XMLHttpRequest 对象。

通过一行简单的 JavaScript 代码，我们就可以创建 XMLHttpRequest 对象。

### 创建 XMLHttpRequest 对象的语法：

xmlhttp=new XMLHttpRequest();

### 老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

下面的代码片段把 XML 文档解析到 XML DOM 对象中：

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.open("GET","books.xml",false);

xmlhttp.send();

xmlDoc=xmlhttp.responseXML;