**XML 基础**

# XML 教程

**XML 指可扩展标记语言**

**XML 被设计用来传输和存储数据。**

**在我们的 XML 教程中，你将了解什么是 XML，以及 XML 与 HTML 之间的差异。**

**XML 很重要，也很容易学习。**

# XML 简介

**什么是 XML?**

* XML 指可扩展标记语言（**EX**tensible **M**arkup **L**anguage）
* 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。

**没有任何行为的 XML**

**XML 是不作为的。**

也许这有点难以理解，但是 XML 不会做任何事情。XML 被设计用来结构化、存储以及传输信息。

下面是 John 写给 George 的便签，存储为 XML：

<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

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

</note>

上面的这条便签具有自我描述性。它拥有标题以及留言，同时包含了发送者和接受者的信息。

但是，这个 XML 文档仍然没有做任何事情。它仅仅是包装在 XML 标签中的纯粹的信息。我们需要编写软件或者程序，才能传送、接收和显示出这个文档。

**XML 仅仅是纯文本**

XML 没什么特别的。它仅仅是纯文本而已。有能力处理纯文本的软件都可以处理 XML。

不过，能够读懂 XML 的应用程序可以有针对性地处理 XML 的标签。标签的功能性意义依赖于应用程序的特性。

**通过 XML 您可以发明自己的标签**

上例中的标签没有在任何 XML 标准中定义过（比如 <to> 和 <from>）。这些标签是由文档的创作者发明的。

这是因为 XML 没有预定义的标签。

在 HTML 中使用的标签（以及 HTML 的结构）是预定义的。HTML 文档只使用在 HTML 标准中定义过的标签（比如 <p> 、<h1> 等等）。

XML 允许创作者定义自己的标签和自己的文档结构。

**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 是 W3C 的推荐标准**

可扩展标记语言 (XML) 于 1998 年 2 月 10 日成为 W3C 的推荐标准。

如需有关 W3C XML 活动的更多信息，请访问我们的 [W3C 教程](http://www.w3school.com.cn/w3c/w3c_xml.asp)。

**XML 无所不在**

当我们看到 XML 标准突飞猛进的开发进度，以及大批的软件开发商采用这个标准的日新月异的速度时，真的是不禁感叹这真是令人叹为观止。

目前，XML 在 Web 中起到的作用不会亚于一直作为 Web 基石的 HTML。

XML 无所不在。XML 是各种应用程序之间进行数据传输的最常用的工具，并且在信息存储和描述领域变得越来越流行。

XML 的用途

**XML 应用于 web 开发的许多方面，常用于简化数据的存储和共享。**

# XML 的用途

**XML 把数据从 HTML 分离**

如果你需要在 HTML 文档中显示动态数据，那么每当数据改变时将花费大量的时间来编辑 HTML。

通过 XML，数据能够存储在独立的 XML 文件中。这样你就可以专注于使用 HTML 进行布局和显示，并确保修改底层数据不再需要对 HTML 进行任何的改变。

通过使用几行 JavaScript，你就可以读取一个外部 XML 文件，然后更新 HTML 中的数据内容。

**提示：**您将在稍后的章节学习更多这方面的内容。

**XML 简化数据共享**

在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据。

XML 数据以纯文本格式进行存储，因此提供了一种独立于软件和硬件的数据存储方法。

这让创建不同应用程序可以共享的数据变得更加容易。

**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 简化平台的变更**

升级到新的系统（硬件或软件平台），总是非常费时的。必须转换大量的数据，不兼容的数据经常会丢失。

XML 数据以文本格式存储。这使得 XML 在不损失数据的情况下，更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

**XML 使您的数据更有用**

由于 XML 独立于硬件、软件以及应用程序，XML 使您的数据更可用，也更有用。

不同的应用程序都能够访问您的数据，不仅仅在 HTML 页中，也可以从 XML 数据源中进行访问。

通过 XML，您的数据可供各种阅读设备使用（手持的计算机、语音设备、新闻阅读器等），还可以供盲人或其他残障人士使用。

**XML 用于创建新的 Internet 语言**

很多新的 Internet 语言是通过 XML 创建的：

其中的例子包括：

* XHTML - 最新的 HTML 版本
* WSDL - 用于描述可用的 web service
* WAP 和 WML - 用于手持设备的标记语言
* RSS - 用于 RSS feed 的语言
* RDF 和 OWL - 用于描述资源和本体
* SMIL - 用于描述针针对 web 的多媒体

**假如开发人员都是理性的**

**假如他们都是理性的，就让未来的应用程序使用 XML 来交换数据吧。**

未来也许会出现某种字处理软件、电子表格程序以及数据库，它们可以使用纯文本格式读取彼此的数据，而不需要使用任何的转换程序。

我们现在能做的只有祈祷微软公司和所有其他的软件开发商在这一方面取得一致了。

XML 树结构

**XML 文档形成了一种树结构，它从“根部”开始，然后扩展到“枝叶”。**

# XML 树结构

**一个 XML 文档实例**

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 具有出色的自我描述性，你同意吗？

**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>

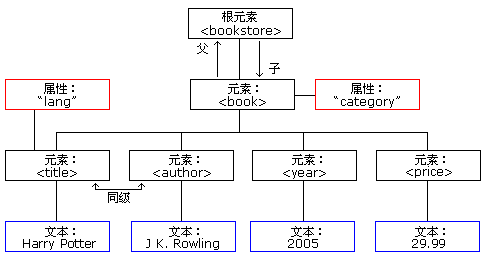
</child>

</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）。

**实例**



上图表示下面的 XML 中的一本书：

<bookstore>

<book category="COOKING">

<title lang="en">Everyday Italian</title>

<author>Giada De Laurentiis</author>

<year>2005</year>

<price>30.00</price>

</book>

<book category="CHILDREN">

<title lang="en">Harry Potter</title>

<author>J K. Rowling</author>

<year>2005</year>

<price>29.99</price>

</book>

<book category="WEB">

<title lang="en">Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price>39.95</price>

</book>

</bookstore>

例子中的根元素是 <bookstore>。文档中的所有 <book> 元素都被包含在 <bookstore> 中。

<book> 元素有 4 个子元素：<title>、< author>、<year>、<price>。

# 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：

<p>This is a paragraph

<p>This is another paragraph

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：

<p>This is a paragraph</p>

<p>This is another paragraph</p>

**注释：**您也许已经注意到 XML 声明没有关闭标签。这不是错误。声明不属于XML本身的组成部分。它不是 XML 元素，也不需要关闭标签。

**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>

**注释：**打开标签和关闭标签通常被称为开始标签和结束标签。不论您喜欢哪种术语，它们的概念都是相同的。

**XML 必须正确地嵌套**

在 HTML 中，常会看到没有正确嵌套的元素：

<b><i>This text is bold and italic</b></i>

在 XML 中，所有元素都**必须**彼此正确地嵌套：

<b><i>This text is bold and italic</i></b>

在上例中，正确嵌套的意思是：由于 <i> 元素是在 <b> 元素内打开的，那么它必须在 <b> 元素内关闭。

**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**。

<root>

<child>

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

</child>

</root>

**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。请研究下面的两个 XML 文档。第一个是错误的，第二个是正确的：

<note date=08/08/2008>

<to>George</to>

<from>John</from>

</note>

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

<to>George</to>

<from>John</from>

</note>

在第一个文档中的错误是，note 元素中的 date 属性没有加引号。

**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 -->

**在 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 以 LF 存储换行**

在 Windows 应用程序中，换行通常以一对字符来存储：回车符 (CR) 和换行符 (LF)。这对字符与打字机设置新行的动作有相似之处。在 Unix 应用程序中，新行以 LF 字符存储。而 Macintosh 应用程序使用 CR 来存储新行。