# XML 教程

XML 指可扩展标记语言（eXtensible Markup Language）。

XML 被设计用来传输和存储数据。

XML 很重要，也很容易学习。

现在开始学习 XML!

## XML 文档实例

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## XML 实例

通过实例进行学习！使用我们的编辑器，你可以编辑 XML 代码，然后点击测试按钮查看结果。

尝试一下！

XML 简介 – 什么是 XML?



首页 HTML CSS JS 本地书签



# XML 简介

XML 被设计用来传输和存储数据。

HTML 被设计用来显示数据。

## 应该掌握的基础知识

在您继续学习之前，需要对以下知识有基本的了解：

HTML

JavaScript

如果您希望首先学习这些项目，请在我们的 首页 访问这些教程。

---

## 什么是 XML?

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 和 HTML 之间的差异

XML 不是 HTML 的替代。

XML 和 HTML 为不同的目的而设计：

XML 被设计用来传输和存储数据，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 不会做任何事情

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。

下面实例是 Jani 写给 Tove 的便签，存储为 XML：

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

上面的这条便签具有自我描述性。它包含了发送者和接受者的信息，同时拥有标题以及消息主体。

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。

---

## 通过 XML 您可以发明自己的标签

上面实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 XML 文档的创作者发明的。

这是因为 XML 语言没有预定义的标签。

HTML 中使用的标签都是预定义的。HTML 文档只能使用在 HTML 标准中定义过的标签（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

XML 于 1998 年 2 月 10 日成为 W3C 的推荐标准。

如需了解有关 W3C XML 活动的更多信息，请访问我们的 W3C 教程。

---

## XML 无所不在

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 是各种应用程序之间进行数据传输的最常用的工具。

□ XML 教程 XML 用途 □





□ XML 简介 – 什么是 XML? XML 树结构 □

# XML 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 把数据从 HTML 分离

如果您需要在 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。这样您就可以专注于使用 HTML/CSS 进行显示和布局，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 代码，您就可以读取一个外部 XML 文件，并更新您的网页的数据内容。

## 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

对开发人员来说，其中一项最费时的挑战一直是在互联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、新的应用程序或新的浏览器。

## XML 使您的数据更有用

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（掌上计算机、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 用于创建新的互联网语言

很多新的互联网语言是通过 XML 创建的。

这里有一些实例：

XHTML

用于描述可用的 Web 服务 的 WSDL

作为手持设备的标记语言的 WAP 和 WML

用于新闻 feed 的 RSS 语言

描述资本和本体的 RDF 和 OWL

用于描述针针对 Web 的多媒体 的 SMIL

## 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 XML 格式读取彼此的数据，而不需要使用任何的转换程序。

□ XML 简介 – 什么是 XML?　　XML 树结构 □





□ XML 用途　　XML 语法 □

# 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"。

## 一个 XML 文档实例

XML 文档使用简单的具有自我描述性的语法：

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</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（UTF-8 : 万国码, 可显示各种语言）。

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>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

最后一行定义根元素的结尾：

```
</note>
```

您可以假设，从这个实例中，XML 文档包含了一张 Jani 写给 Tove 的便签。

XML 具有出色的自我描述性，您同意吗？

---

## 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。

所有的元素都可以有子元素：

```
<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）。

所有的元素都可以有文本内容和属性（类似 HTML 中）。

---

## 实例：

DOMnode tree

上图表示下面的 XML 中的一本书：

### XML 文档实例

```
<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>
```

实例中的根元素是 <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>。

1 篇笔记
#1

写笔记

XML将数据组织成为一棵树,DOM通过解析 XML 文档,为 XML 文档在逻辑上建立一个树模型,树的节点是一个个的对象。这样通过操作这棵树和这些对象就可以完成对 XML 文档的操作,为处理文档的所有方面提供了一个完美的概念性框架。
如下XML文档:

```
<line id="1"> the <bold>First</bold>line</line>
```

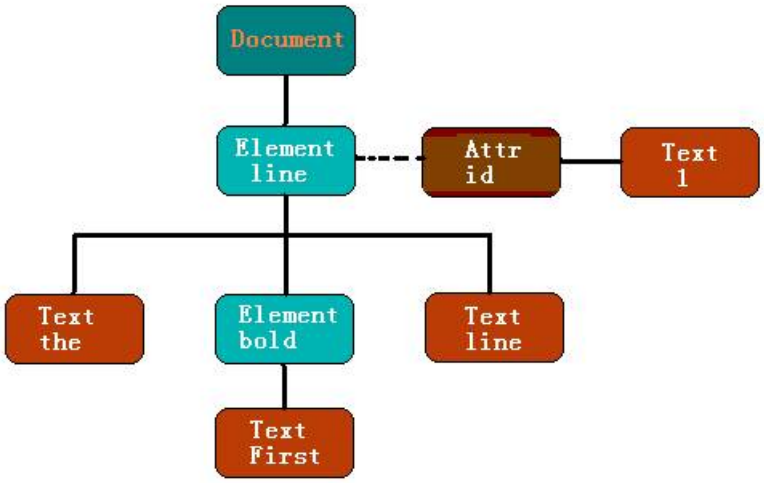


图1 XML 文档的 DOM表示

由于DOM"一切都是节点（everything-is-a-node）"，XML树的每个 Document、Element、Text 、Attr和Comment都是 DOM Node。
由上面例子可知， DOM实质上是一些节点的集合。由于文档中可能包含有不同类型的信息，所以定义了几种不同类型的节点，如：Document、Element、Text、Attr 、CDATASection、ProcessingInstruction、Notation 、EntityReference、Entity、DocumentType、DocumentFragment等。

Yvonne7个月前 (03-14)

反馈/建议





XML 树结构 XML 元素

# 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。

## XML 文档必须有根元素

XML 必须包含根元素，它是所有其他元素的父元素，比如以下实例中 root 就是根元素：

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

以下实例中 note 是根元素：

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## XML 声明

XML 声明文件的可选部分，如果存在需要放在文档的第一行，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
```

> *以上实例包含 XML 版本（*
>
> *UTF-8 也是 HTML5, CSS, JavaScript, PHP, 和 SQL 的默认编码。*

## 所有的 XML 元素都必须有一个关闭标签

在 HTML 中，某些元素不必有一个关闭标签：

```
<p>This is a paragraph.

<br>
```

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>

<br />
```

**注释：**从上面的实例中，您也许已经注意到 XML 声明没有关闭标签。这不是错误。声明不是 XML 文档本身的一部分，它没有关闭标签。

## XML 标签对大小写敏感

XML 标签对大小写敏感。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>
```

在上面的实例中，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 属性值必须加引号

与 HTML 类似，XML 元素也可拥有属性（名称/值的对）。

在 XML 中，XML 的属性值必须加引号。

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=12/11/2007>

<to>Tove</to>

<from>Jani</from>

</note>
```

```
<note date="12/11/2007">

<to>Tove</to>

<from>Jani</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。

如果您把字符 "<" 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; | < | less than |
| &gt; | > | greater than |
| &amp; | & | ampersand |

| | | |
|---|---|---|
| ' | ' | apostrophe |
| " | " | quotation mark |

**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 Tove |
| Output: | Hello Tove |

在 XML 中，文档中的空格不会被删减。

## 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）。

在 Unix 和 Mac OSX 中，使用 LF 来存储新行。

在旧的 Mac 系统中，使用 CR 来存储新行。

XML 以 LF 存储换行。

□ XML 树结构 XML 元素 □

□

**1 篇笔记**

#1

□写笔记

□

**XML语法-属性**

```
<student id="100">

    <name>Tom</name>

</student>
```

(1)属性值用双引号 " 或单引号 ' 分隔，如果属性值中有单引号，则用双引号分隔；如果有双引号，则用单引号分隔。那么如果属性值中既有单引号还有双引号怎么办？这种要使用实体（转义字符，类似于html中的空格符），XML 有 5 个预定义的实体字符，如下：

| | | |
|---|---|---|
| < | < | 小于 |
| > | > | 大于 |
| & | & | 和号 |
| ' | ' | 单引号 |
| " | " | 引号 |

(2)一个元素可以有多个属性，它的基本格式为：

```
<元素名 属性名1="属性值1" 属性名2="属性值2">
```

(3)特定的属性名称在同一个元素标记中只能出现一次

(4)属性值不能包括 <,>,&，如果一定要包含，也要使用实体

# 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。

一个元素可以包含：

- 其他元素
- 文本
- 属性
- 或混合以上所有...

```
<bookstore>
<book category="CHILDREN">
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="WEB">
<title>Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

在上面的实例中，<bookstore> 和 <book> 都有**元素内容**，因为他们包含其他元素。<book> 元素也有**属性**（category="CHILDREN"）。<title>、<author>、<year> 和 <price> 有**文本内容**，因为他们包含文本。

## XML 命名规则

XML 元素必须遵循以下命名规则：

- 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

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：<first_name>、<last_name>。

名称应简短和简单，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"，一些软件会认为您想要从 first 里边减去 name。

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，即使用数据库的命名规则来命名 XML 文档中的元素。

在 XML 中，éòá 等非英语字母是完全合法的，不过需要留意，您的软件供应商不支持这些字符时可能出现的问题。

## XML 元素是可扩展的

XML 元素是可扩展，以携带更多的信息。

请看下面的 XML 实例：

```
<note>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

让我们设想一下，我们创建了一个应用程序，可将 <to>、<from> 以及 <body> 元素从 XML 文档中提取出来，并产生以下的输出：

**MESSAGE**

**To:** Tove

**From:** Jani

Don't forget me this weekend!

想象一下，XML 文档的作者添加的一些额外信息：

```
<note>
<date>2008-01-10</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

那么这个应用程序会中断或崩溃吗？

不会。这个应用程序仍然可以找到 XML 文档中的 <to>、<from> 以及 <body> 元素，并产生同样的输出。

XML 的优势之一，就是可以在不中断应用程序的情况下进行扩展。

□ XML 语法 XML 属性 □

□点我分享笔记

反馈/建议



首页 HTML CSS JS 本地书签

□ XML 元素 XML DTD □

# XML 属性

XML元素具有属性，类似 HTML。

属性（Attribute）提供有关元素的额外信息。

## XML 属性

在 HTML 中，属性提供有关元素的额外信息：

```
<img src="computer.gif">
<a href="demo.html">
```

属性通常提供不属于数据组成部分的信息。在下面的实例中，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 元素可以这样写：

```
<person sex="female">
```

或者这样也可以：

```
<person sex='female'>
```

如果属性值本身包含双引号，您可以使用单引号，就像这个实例：

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

或者您可以使用字符实体：

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

## XML 元素 vs. 属性

请看这些实例：

```
<person sex="female">
<firstname>Anna</firstname>
<lastname>Smith</lastname>
</person>
```

```
<person>
<sex>female</sex>
<firstname>Anna</firstname>
<lastname>Smith</lastname>
</person>
```

在第一个实例中，sex 是一个属性。在第二个实例中，sex 是一个元素。这两个实例都提供相同的信息。

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，那么请使用元素吧。

## 我最喜欢的方式

下面的三个 XML 文档包含完全相同的信息：

第一个实例中使用了 date 属性：

```
<note date="10/01/2008">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

第二个实例中使用了 date 元素：

```
<note>
<date>10/01/2008</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

第三个实例中使用了扩展的 date 元素（这是我的最爱）：

```
<note>
<date>
<day>10</day>
<month>01</month>
<year>2008</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## 避免 XML 属性？

因使用属性而引起的一些问题：

属性不能包含多个值（元素可以）

属性不能包含树结构（元素可以）

属性不容易扩展（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。

不要做这样的蠢事（这不是 XML 应该被使用的方式）：

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

## 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。这个实例向我们演示了这种情况：

```
<messages>
<note id="501">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
<note id="502">
<to>Jani</to>
<from>Tove</from>
<heading>Re: Reminder</heading>
<body>I will not</body>
</note>
</messages>
```

上面的 id 属性仅仅是一个标识符，用于标识不同的便签。它并不是便签数据的组成部分。

在此我们极力向您传递的理念是：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 元素 XML DTD

# 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"的 XML 文档拥有正确的语法。

在前面的章节描述的语法规则：

- XML 文档必须有一个根元素
- XML元素都必须有一个关闭标签
- XML 标签对大小写敏感
- XML 元素必须被正确的嵌套
- XML 属性值必须加引号

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## 验证 XML 文档

合法的 XML 文档是"形式良好"的 XML 文档，这也符合文档类型定义（DTD）的规则：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

在上面的实例中，DOCTYPE 声明是对外部 DTD 文件的引用。下面的段落展示了这个文件的内容。

## XML DTD

DTD 的目的是定义 XML 文档的结构。它使用一系列合法的元素来定义文档结构：

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

如果您想要学习 DTD，请在我们的首页查找 DTD 教程。

## XML Schema

W3C 支持一种基于 XML 的 DTD 代替者，它名为 XML Schema：

```
<xs:element name="note">

<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

```
</xs:element>
```

如果您想要学习 XML Schema，请在我们的首页查找 Schema 教程。

## 一个通用的 XML 验证器

为了帮助您检查 XML 文件的语法，我们创建了 XML 验证器，以便您对任何 XML 文件进行语法检查。

请看下一章。

XML 属性 XML 验证器





XML DTD 查看 XML 文件

# XML 验证器

使用我们的 XML 验证器来对您的 XML 文件进行语法检查。

## XML 错误会终止您的程序

XML 文档中的错误会终止您的 XML 应用程序。

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（比如您忘记了结束标签）。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。**

## 对您的 XML 进行语法检查

为了帮助您对 XML 进行语法检查，我们创建了一个 XML 验证器。

把您的 XML 粘贴到下面的文本框中，然后点击"验证"按钮来进行语法检查。

语法正确实例：

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

验证

语法错误实例：

```
<?xml version="1.0" ?>
<note>
<to>Tove</to>
<from>Jani</Ffrom>
<heading>Reminder</heading>
```

验证

**注释：**只会检查您的 XML 是否"形式良好"。如果您想根据 DTD 来验证 XML，请参阅此页面上的最后一段。

## 根据 DTD 来验证 XML

如果您运行 **Internet Explorer**，您可以在下面的文本区域中根据 DTD 验证您的 XML。

只要把 DOCTYPE 声明（带有 DTD）添加到您的 XML 中 <xml> 元素后，然后点击"验证"按钮即可：

```
<?xml version="1.0" ?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

验证（只限 IE!）

XML DTD　　查看 XML 文件





XML 验证器　　XML 和 CSS

# 查看 XML 文件

在所有主流的浏览器中，均能够查看原始的 XML 文件。

不要指望 XML 文件会直接显示为 HTML 页面。

## 查看 XML 文件

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

查看这个 XML 文件：note.xml

XML 文档将显示为代码颜色化的根以及子元素。通过点击元素左侧的加号（+）或减号（ - ），可以展开或收起元素的结构。要查看原始的 XML 源（不包括 + 和 - 符号），选择"查看页面源代码"或从浏览器菜单"查看源文件"。

**注释：**在 Safari 中，只有元素的文本将被显示。要查看原始的 XML，您必须右键单击页面，选择"查看源文件"。

## 查看无效的 XML 文件

如果一个错误的XML文件被打开，浏览器会报告错误。

请查看这个 XML 文件：note_error.xml

## 其他 XML 实例

请查看这些 XML 文档，这会有助于您建立对 XML 的感性认识。

一个 XML 的 CD 目录

这是一个 CD 集，存储为 XML 数据。

一个 XML 的植物目录

这是一个来自植物店的植物目录，存储为 XML 数据。

一个简单的食物菜单

这是一个来自餐馆的早餐菜单，存储为 XML 数据。

## 为什么 XML 显示这个样子？

XML 文档不会携带有关如何显示数据的信息。

由于 XML 标签由 XML 文档的作者"发明"，浏览器无法确定像 <table> 这样一个标签究竟描述一个 HTML 表格还是一个餐桌。

在没有任何有关如何显示数据的信息的情况下，大多数的浏览器都会仅仅把 XML 文档显示为源代码。

在下面的章节，我们会了解几个有关这个显示问题的解决方案，其中会使用 CSS、XSLT 和 JavaScript。

□ XML 验证器　　XML 和 CSS □





□ 查看 XML 文件　　XML 和 XSLT □

# 使用 CSS 显示 XML

通过使用 CSS（Cascading Style Sheets 层叠样式表），您可以添加显示信息到 XML 文档中。

## 使用 CSS 显示您的 XML?

使用 CSS 来格式化 XML 文档是有可能的。

下面的实例就是关于如何使用 CSS 样式表来格式化 XML 文档：

请看这个 XML 文件：CD 目录

然后看这个样式表：CSS 文件

最后，请查看：使用 CSS 文件格式化的 CD 目录

下面是 XML 文件的一小部分。第二行把 XML 文件链接到 CSS 文件：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
<CD>
<TITLE>Empire Burlesque</TITLE>
<ARTIST>Bob Dylan</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>Columbia</COMPANY>
<PRICE>10.90</PRICE>
<YEAR>1985</YEAR>
</CD>
<CD>
<TITLE>Hide your heart</TITLE>
<ARTIST>Bonnie Tyler</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>CBS Records</COMPANY>
<PRICE>9.90</PRICE>
<YEAR>1988</YEAR>
</CD>
.
.
.
</CATALOG>
```

使用 CSS 格式化 XML 不是常用的方法。

W3C 推荐使用 XSLT，请看下一章。

□ 查看 XML 文件　　XML 和 XSLT □





□ XML 和 CSS　　XMLHttpRequest 对象 □

# 使用 XSLT 显示 XML

通过使用 XSLT，您可以把 XML 文档转换成 HTML 格式。

## 使用 XSLT 显示 XML

XSLT 是首选的 XML 样式表语言。

XSLT（eXtensible Stylesheet Language Transformations）远比 CSS 更加完善。

XSLT 是在浏览器显示 XML 文件之前，先把它转换为 HTML：

使用 XSLT 显示 XML

如果您想要学习有关 XSLT 的知识，请在我们的首页查找 XSLT 教程。

## 在服务器上通过 XSLT 转换 XML

在上面的实例中，当浏览器读取 XML 文件时，XSLT 转换是由浏览器完成的。

在使用 XSLT 来转换 XML 时，不同的浏览器可能会产生不同结果。为了减少这种问题，可以在服务器上进行 XSLT 转换。

查看结果。

□ XML 和 CSS　　XMLHttpRequest 对象 □

□ 点我分享笔记

反馈/建议



□



□ XML 和 XSLT　　XML 解析器 □

# XMLHttpRequest 对象

## 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

如需学习更多关于 XMLHttpRequest 对象的知识，请学习我们的 XML DOM 教程。

## XMLHttpRequest 实例

当你在下面的输入字段中键入一个字符，一个 XMLHttpRequest 发送到服务器 - 返回名称的建议（从服务器上的文件）：

在输入框中键入一个字母：

首字母

建议:

## 创建一个 XMLHttpRequest 对象

所有现代浏览器（IE7+、Firefox、Chrome、Safari 和 Opera）都有内建的 XMLHttpRequest 对象。

创建 XMLHttpRequest 对象的语法：

```
xmlhttp=new XMLHttpRequest();
```

旧版本的Internet Explorer（IE5和IE6）中使用 ActiveX 对象：

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

在下一章中，我们将使用 XMLHttpRequest 对象从服务器取回 XML 信息。

□ XML 和 XSLT　　XML 解析器 □



□ XMLHttpRequest 对象　　XML DOM □

# XML Parser

所有现代浏览器都有内建的 XML 解析器。

XML 解析器把 XML 文档转换为 XML DOM 对象 - 可通过 JavaScript 操作的对象。

## 解析 XML 文档

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;
```

## 解析 XML 字符串

下面的代码片段把 XML 字符串解析到 XML DOM 对象中：

```
txt="<bookstore><book>";
txt=txt+"<title>Everyday Italian</title>";
txt=txt+"<author>Giada De Laurentiis</author>";
txt=txt+"<year>2005</year>";
txt=txt+"</book></bookstore>";

if (window.DOMParser)
{
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
}
else // Internet Explorer
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async=false;
xmlDoc.loadXML(txt);
}
```

**注释：** Internet Explorer 使用 loadXML() 方法来解析 XML 字符串，而其他浏览器使用 DOMParser 对象。

## 跨域访问

出于安全方面的原因，现代的浏览器不允许跨域的访问。

这意味着，网页以及它试图加载的 XML 文件，都必须位于相同的服务器上。

## XML DOM

在下一章中，您将学习如何访问 XML DOM 对象并取回数据。

□ XMLHttpRequest 对象　　XML DOM □





□ XML 解析器　　在 HTML 页面中显示 XML 数据 □

# XML DOM

DOM（Document Object Model 文档对象模型）定义了访问和操作文档的标准方法。

## XML DOM

XML DOM（XML Document Object Model）定义了访问和操作 XML 文档的标准方法。

XML DOM 把 XML 文档作为树结构来查看。

所有元素可以通过 DOM 树来访问。可以修改或删除它们的内容，并创建新的元素。元素，它们的文本，以及它们的属性，都被认为是节点。

在我们的 XML DOM 教程中，您可以学习更多有关 XML DOM 的知识。

## HTML DOM

HTML DOM 定义了访问和操作 HTML 文档的标准方法。

所有 HTML 元素可以通过 HTML DOM 来访问。

在我们的 HTML DOM 教程中，您可以学习更多有关 HTML DOM 的知识。.

## 加载一个 XML 文件 - 跨浏览器实例

下面的实例把 XML 文档（"note.xml"）解析到 XML DOM 对象中，然后通过 JavaScript 提取一些信息：

### 实例

```
<html>
<body>
<h1>W3Schools Internal Note</h1>
<div>
<b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</div>
```

```
<script>
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","note.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

尝试一下 »

## 重要注释！

如需从上面的 XML 文件（"note.xml"）的 <to> 元素中提取文本 "Tove"，语法是：

```
getElementsByTagName("to")[0].childNodes[0].nodeValue
```

请注意，即使 XML 文件只包含一个 <to> 元素，您仍然必须指定数组索引 [0]。这是因为 getElementsByTagName() 方法返回一个数组。

## 加载一个 XML 字符串 - 跨浏览器实例

下面的实例把 XML 字符串解析到 XML DOM 对象中，然后通过 JavaScript 提取一些信息：

### 实例

```
<html>
<body>
<h1>W3Schools Internal Note</h1>
<div>
<b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</div>

<script>
txt="<note>";
txt=txt+"<to>Tove</to>";
txt=txt+"<from>Jani</from>";
txt=txt+"<heading>Reminder</heading>";
txt=txt+"<body>Don't forget me this weekend!</body>";
txt=txt+"</note>";

if (window.DOMParser)
{
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
}
else // Internet Explorer
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async=false;
xmlDoc.loadXML(txt);
}

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
```

```
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>
</body>
</html>
```

尝试一下 »

XML 解析器 在 HTML 页面中显示 XML 数据

点我分享笔记

反馈/建议





XML DOM XML 应用程序

# HTML 页面显示 XML 数据

## 在 HTML 页面中显示 XML 数据

在下面的实例中，我们打开一个 XML 文件（"cd_catalog.xml"），然后遍历每个 CD 元素，并显示HTML 表格中的 ARTIST 元素和 TITLE 元素的值：

### 实例

```
<html>
<body>

<script>
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","cd_catalog.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

document.write("<table border='1'>");
var x=xmlDoc.getElementsByTagName("CD");
for (i=0;i<x.length;i++)
{
document.write("<tr><td>");
document.write(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
document.write("</td><td>");
document.write(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
document.write("</td></tr>");
```

```
}
document.write("</table>");
</script>

</body>
</html>
```

尝试一下 »

如需了解更多关于使用 JavaScript 和 XML DOM 的信息，请访问我们的 XML DOM 教程。

XML DOM　　XML 应用程序

点我分享笔记

反馈/建议





在 HTML 页面中显示 XML 数据　　XML 命名空间

# XML 应用程序

本章演示一些基于 XML, HTML, XML DOM 和 JavaScript 构建的小型 XML 应用程序。

## XML 文档实例

在本应用程序中，我们将使用 "cd_catalog.xml" 文件。

## 在 HTML div 元素中显示第一个 CD

下面的实例从第一个 CD 元素中获取 XML 数据，然后在 id="showCD" 的 HTML 元素中显示数据。displayCD() 函数在页面加载时调用：

### 实例

```
x=xmlDoc.getElementsByTagName("CD");
i=0;

function displayCD()
{
artist=(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
title=(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
year=(x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue);
txt="Artist: " + artist + "<br />Title: " + title + "<br />Year: "+ year;
document.getElementById("showCD").innerHTML=txt;
}
```

尝试一下 »

## 添加导航脚本

为了向上面的实例添加导航（功能），需要创建 next() 和 previous() 两个函数：

## 实例

```
function next()
{ // display the next CD, unless you are on the last CD
if (i<x.length-1)
{
i++;
displayCD();
}
}

function previous()
{ // displays the previous CD, unless you are on the first CD
if (i>0)
{
i--;
displayCD();
}
}
```

尝试一下 »

## 当点击 CD 时显示专辑信息

最后的实例展示如何在用户点击某个 CD 项目时显示专辑信息：

尝试一下。

如需了解更多关于使用 JavaScript 和 XML DOM 的信息，请访问我们的 XML DOM 教程。

在 HTML 页面中显示 XML 数据　　XML 命名空间





XML 应用程序　　XML CDATA

# XML 命名空间

XML 命名空间提供避免元素命名冲突的方法。

## 命名冲突

在 XML 中，元素名称是由开发者定义的，当两个不同的文档使用相同的元素名时，就会发生命名冲突。

这个 XML 携带 HTML 表格的信息：

```
<table>
<tr>
<td>Apples</td>
<td>Bananas</td>
```

```
</tr>
</table>
```

这个 XML 文档携带有关桌子的信息（一件家具）：

```
<table>
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
```

假如这两个 XML 文档被一起使用，由于两个文档都包含带有不同内容和定义的 <table> 元素，就会发生命名冲突。

XML 解析器无法确定如何处理这类冲突。

---

## 使用前缀来避免命名冲突

在 XML 中的命名冲突可以通过使用名称前缀从而容易地避免。

该 XML 携带某个 HTML 表格和某件家具的信息：

```
<h:table>
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>

<f:table>
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>
```

在上面的实例中，不会有冲突，因为两个 <table> 元素有不同的名称。

---

## XML 命名空间 - xmlns 属性

当在 XML 中使用前缀时，一个所谓的用于前缀的**命名空间**必须被定义。

命名空间是在元素的开始标签的 **xmlns 属性**中定义的。

命名空间声明的语法如下。xmlns:*前缀*="*URI*"。

```
<root>

<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>

<f:table xmlns:f="http://www.w3cschool.cc/furniture">
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>

</root>
```

在上面的实例中，<table> 标签的 xmlns 属性定义了 h: 和 f: 前缀的合格命名空间。

当命名空间被定义在元素的开始标签中时，所有带有相同前缀的子元素都会与同一个命名空间相关联。

命名空间，可以在他们被使用的元素中或者在 XML 根元素中声明：

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3cschool.cc/furniture">

<h:table>
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>

<f:table>
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>

</root>
```

**注释：**命名空间 URI 不会被解析器用于查找信息。

其目的是赋予命名空间一个惟一的名称。不过，很多公司常常会作为指针来使用命名空间指向实际存在的网页，这个网页包含关于命名空间的信息。

请访问 http://www.w3.org/TR/html4/。

## 统一资源标识符（URI，全称 Uniform Resource Identifier）

**统一资源标识符**（URI）是一串可以标识因特网资源的字符。

最常用的 URI 是用来标识因特网域名地址的**统一资源定位器**（URL）。另一个不那么常用的 URI 是**统一资源命名**（URN）。

在我们的实例中，我们仅使用 URL。

## 默认的命名空间

为元素定义默认的命名空间可以让我们省去在所有的子元素中使用前缀的工作。它的语法如下：

```
xmlns="namespaceURI"
```

这个 XML 携带 HTML 表格的信息：

```
<table xmlns="http://www.w3.org/TR/html4/">
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

这个XML携带有关一件家具的信息：

```
<table xmlns="http://www.w3schools.com/furniture">
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
```

## 实际使用中的命名空间

XSLT 是一种用于把 XML 文档转换为其他格式的 XML 语言，比如 HTML。

在下面的 XSLT 文档中，您可以看到，大多数的标签是 HTML 标签。

非 HTML 的标签都有前缀 xsl，并由此命名空间标识：xmlns:xsl="http://www.w3.org/1999/XSL/Transform"：

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr>
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

如果您想要学习有关 XSLT 的知识，请在我们的首页查找 XSLT 教程。

# 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。

## PCDATA - 被解析的字符数据

XML 解析器通常会解析 XML 文档中所有的文本。

当某个 XML 元素被解析时，其标签之间的文本也会被解析：

```
<message>This text is also parsed</message>
```

解析器之所以这么做是因为 XML 元素可包含其他元素，就像这个实例中，其中的 <name> 元素包含着另外的两个元素（first 和 last）：

```
<name><first>Bill</first><last>Gates</last></name>
```

而解析器会把它分解为像这样的子元素：

```
<name>
<first>Bill</first>
<last>Gates</last>
</name>
```

解析字符数据（PCDATA）是 XML 解析器解析的文本数据使用的一个术语。

## CDATA - （未解析）字符数据

术语 CDATA 是不应该由 XML 解析器解析的文本数据。

像 "<" 和 "&" 字符在 XML 元素中都是非法的。

"<" 会产生错误，因为解析器会把该字符解释为新元素的开始。

"&" 会产生错误，因为解析器会把该字符解释为字符实体的开始。

某些文本，比如 JavaScript 代码，包含大量 "<" 或 "&" 字符。为了避免错误，可以将脚本代码定义为 CDATA。

CDATA 部分中的所有内容都会被解析器忽略。

CDATA 部分由 "<![CDATA[" 开始，由 "]]>" 结束：

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
{
return 1;
}
else
{
return 0;
}
}
]]>
</script>
```

在上面的实例中，解析器会忽略 CDATA 部分中的所有内容。

**关于 CDATA 部分的注释：**

CDATA 部分不能包含字符串 "]]>"。也不允许嵌套的 CDATA 部分。

标记 CDATA 部分结尾的 "]]>" 不能包含空格或换行。

□ XML 命名空间 XML 编码 □





# XML 编码

XML 文档可以包含非 ASCII 字符，比如挪威语 æ ø å，或者法语 ê è é。

为了避免错误，需要规定 XML 编码，或者将 XML 文件存为 Unicode。

## XML 编码错误

如果您载入一个 XML 文档，您可以得到两个不同的错误，表示编码问题：

**在文本内容中发现无效字符。**

如果您的 XML 中包含非 ASCII 字符，且文件保存为没有指定编码的单字节 ANSI（或 ASCII），您会得到一个错误。

单字节编码属性的 XML 文件。

相同的单字节没有编码属性的 XML 文件。

**将当前编码切换为不被支持的指定编码**

如果您的 XML 文件保存为带有指定的单字节编码（WINDOWS-1252、ISO-8859-1、UTF-8）的双字节 Unicode（或 UTF-16），您会得到一个错误。

如果您的 XML 文件保存为带有指定的双字节编码（UTF-16）的单字节 ANSI（或 ASCII），您也会得到一个错误。

双字节没有编码的 XML 文件。

相同的双字节具有单字节编码的 XML 文件。

## Windows 记事本

Windows 记事本默认会将文件保存为单字节的 ANSI（ASCII）。

如果您选择 "另存为..."，就可以指定 ANSI、UTF-8、Unicode（UTF-16）或 Unicode Big。

将下面的 XML 保存为 ANSI、UTF-8 和 Unicode（注意文档不包含任何编码属性）。

```
<?xml version="1.0"?>
<note>
<from>Jani</from>
<to>Tove</to>
<message>Norwegian: æøå. French: êèé</message>
</note>
```

尝试将文件拖到您的浏览器，并查看结果。不同的浏览器会显示不同的结果。

不同编码的体验：

```
<?xml version="1.0" encoding="us-ascii"?>
<?xml version="1.0" encoding="windows-1252"?>
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?xml version="1.0" encoding="UTF-16"?>
```

请尝试：

带有正确编码的保存

带有错误编码的保存

## 结论

始终使用编码属性

使用支持编码的编辑器

确保您知道编辑器使用什么编码

在您的编码属性中使用相同的编码

XML CDATA　　服务器上的 XML





XML 编码　　XML DOM 高级

# 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>")
%>
```

请注意，此响应的内容类型必须设置为 "text/xml"。

查看这个 ASP 文件如何从服务器返回。

如果您想要学习 ASP，请在我们的首页查找 ASP 教程。

---

## 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"。

查看这个 PHP 文件如何从服务器返回。

如果您想要学习 PHP，请在我们的首页查找 PHP 教程。

---

## 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>")
%>
```

查看以上 ASP 文件的实际数据库输出。

上面的实例使用了带有 ADO 的 ASP。

如果您想要学习 ASP 和 ADO，请在我们的首页查找相关教程。

---

## 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 发送到您的浏览器。

看看上面的代码怎么运行。

## 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")
%>
```

□ XML 编码　　XML DOM 高级 □





□ 服务器上的 XML　　XML 注意事项 □

# XML DOM 高级

## XML DOM - 高级

在本教程的较早章节中，我们介绍了 XML DOM，并使用了 XML DOM 的 getElementsByTagName() 方法从 XML 文档中取回数据。

在本章中我们将结合一些其他重要的 XML DOM 方法。

您可以在我们的 XML DOM 教程 中学习更多有关 XML DOM 的知识。

## 获取元素的值

下面的实例中使用的 XML 文件：books.xml。

下面的实例检索第一个 <title> 元素的文本值：

### 实例

```
txt=xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

尝试一下 »

## 获取属性的值

下面的实例检索第一个 <title> 元素的 "lang" 属性的文本值：

### 实例

```
txt=xmlDoc.getElementsByTagName("title")[0].getAttribute("lang");
```

尝试一下 »

## 改变元素的值

下面的实例改变第一个 <title> 元素的文本值：

### 实例

```
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];
x.nodeValue="Easy Cooking";
```

尝试一下 »

## 创建新的属性

XML DOM 的 setAttribute() 方法可用于改变现有的属性值，或创建一个新的属性。

下面的实例创建了一个新的属性（edition="first"），然后把它添加到每一个 <book> 元素中：

### 实例

```
x=xmlDoc.getElementsByTagName("book");

for(i=0;i<x.length;i++)
{
x[i].setAttribute("edition","first");
}
```

尝试一下 »

## 创建元素

XML DOM 的 createElement() 方法创建一个新的元素节点。

XML DOM 的 createTextNode() 方法创建一个新的文本节点。

XML DOM 的 appendChild() 方法向节点添加子节点（在最后一个子节点之后）。

如需创建带有文本内容的新元素，需要同时创建元一个新的元素节点和一个新的文本节点，然后把他追加到现有的节点。

下面的实例创建了一个新的元素（<edition>），带有如下文本：First，然后把它添加到第一个 <book> 元素：

### 实例

```
newel=xmlDoc.createElement("edition");
newtext=xmlDoc.createTextNode("First");
newel.appendChild(newtext);

x=xmlDoc.getElementsByTagName("book");
x[0].appendChild(newel);
```

尝试一下 »

实例解释

创建一个 <edition> 元素

创建值为 "First" 的文本节点

把这个文本节点追加到新的 <edition> 元素

把 <edition> 元素追加到第一个 <book> 元素

## 删除元素

下面的实例删除第一个 <book> 元素的第一个节点：

### 实例

```
x=xmlDoc.getElementsByTagName("book")[0];
x.removeChild(x.childNodes[0]);
```

尝试一下 »

**注释：** 上面实例的结果可能会根据所用的浏览器而不同。Firefox 把新行字符当作空的文本节点，而 Internet Explorer 不是这样。您可以在我们的 XML DOM 教程 中阅读到更多有关这个问题以及如何避免它的知识。

□ 服务器上的 XML　　XML 注意事项 □





□ XML DOM 高级　　XML 技术 □

# XML 注意事项

这里列出了您在使用 XML 时应该尽量避免使用的技术。

## Internet Explorer - XML 数据岛

**它是什么？** XML 数据岛是嵌入到 HTML 页面中的 XML 数据。

**为什么要避免使用它？** XML 数据岛只在 Internet Explorer 浏览器中有效。

**用什么代替它？** 您应当在 HTML 中使用 JavaScript 和 XML DOM 来解析并显示 XML。

如需更多有关 JavaScript 和 XML DOM 的信息，请访问我们的 XML DOM 教程。

## XML 数据岛实例

本例使用 XML 文档 "cd_catalog.xml"。

把 XML 文档绑定到 HTML 文档中的一个 <xml> 标签。id 属性定义数据岛的标识符，而 src 属性指向 XML 文件：

### 实例

本实例只适用于 IE 浏览器

```
<html>
<body>

<xml id="cdcat" src="cd_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>

</body>
</html>
```

尝试一下 »

<table> 标签的 datasrc 属性把 HTML 表格绑定到 XML 数据岛。

 标签允许 datafld 属性引用要显示的 XML 元素。在这个实例中，要引用的是 "ARTIST" 和 "TITLE"。当读取 XML 时，会为每个 <CD> 元素创建相应的表格行。

## Internet Explorer - 行为

**它是什么？** Internet Explorer 5 引入了行为。行为是通过使用 CSS 样式向 XML （或 HTML ）元素添加行为的一种方法。

**为什么要避免使用它？** 只有 Internet Explorer 支持 behavior 属性。

**使用什么代替它？** 使用 JavaScript 和 XML DOM（或 HTML DOM）来代替它。

### 实例 1 - 鼠标悬停突出

下面的 HTML 文件中的 <style> 元素为 <h1> 元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
h1 { behavior: url(behave.htc) }
</style>
</head>
<body>

<h1>Mouse over me!!!</h1>

</body>
</html>
```

下面显示的是 XML 文档 "behave.htc"（该文件包含了一段 JavaScript 和针对元素的事件句柄）：

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script>
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

尝试一下 »

### 实例 2 - 打字机模拟

下面的 HTML 文件中的 <style> 元素为 id 为 "typing" 的元素定义了一个行为：

```
<html>
<head>
```

```
<style type="text/css">
#typing
{
behavior:url(typing.htc);
font-family:'courier new';
}
</style>
</head>
<body>

<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>v </span>

</body>
</html>
```

下面显示的是 XML 文档 "typing.htc"：

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script>
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
{
clearInterval(t);
}
}
</script>
```

尝试一下 »

XML 注意事项 现实生活中的 XML

# XML 相关技术

下面是一个 XML 技术的列表。

XHTML (可扩展 HTML)

更严格更纯净的基于 XML 的 HTML 版本。

XML DOM (XML 文档对象模型)

访问和操作 XML 的标准文档模型。

XSL (可扩展样式表语言) XSL 包含三个部分：

- XSLT (XSL 转换) - 把 XML 转换为其他格式，比如 HTML
- XSL-FO (XSL 格式化对象)- 用于格式化 XML 文档的语言
- XPath - 用于导航 XML 文档的语言

XQuery (XML 查询语言)

基于 XML 的用于查询 XML 数据的语言。

DTD (文档类型定义)

用于定义 XML 文档中的合法元素的标准。

XSD (XML 架构)

基于 XML 的 DTD 替代物。

XLink (XML 链接语言)

在 XML 文档中创建超级链接的语言。

XPointer (XML 指针语言)

允许 XLink 超级链接指向 XML 文档中更多具体的部分。

SOAP (简单对象访问协议)

允许应用程序在 HTTP 之上交换信息的基于 XML 的协议。

WSDL (Web 服务描述语言)

用于描述网络服务的基于 XML 的语言。

RDF (资源描述框架)

用于描述网络资源的基于 XML 的语言。

RSS (真正简易聚合)

聚合新闻以及类新闻站点内容的格式。

SVG (可伸缩矢量图形)

定义 XML 格式的图形。

XML 注意事项 现实生活中的 XML

点我分享笔记

反馈/建议

# 现实生活中的 XML

如何使用 XML 来交换信息的一些实例。

## 实例：XML 新闻

**XMLNews 是用于交换新闻和其他信息的规范。**

对新闻的供求双方来说，通过使用这种标准，可以使各种类型的新闻信息通过不同软硬件以及编程语言进行的制作、接收和存档更加容易：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nitf>
<head>
<title>Colombia Earthquake</title>
</head>
<body>
<headline>
<hl1>143 Dead in Colombia Earthquake</hl1>
</headline>
<byline>
<bytag>By Jared Kotler, Associated Press Writer</bytag>
</byline>
<dateline>
<location>Bogota, Colombia</location>
<date>Monday January 25 1999 7:28 ET</date>
</dateline>
</body>
</nitf>
```

## 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>
```

□ XML 技术 XML 编辑器 □





□ 现实生活中的 XML XML E4X □

# XML 编辑器

如果您希望极认真地学习和使用 XML，那么您一定会从一款专业的 XML 编辑器的使用上受益。

## XML 是基于文本的

XML 是基于文本的标记语言。

关于 XML 的一件很重要的事情是，XML 可被类似记事本这样的简单的文本编辑器来创建和编辑。

不过，在您开始使用 XML 进行工作时，您很快会发现，使用一款专业的 XML 编辑器来编辑 XML 文档会更好。

## 为什么不使用记事本?

许多 Web 开发人员使用记事本来编辑 HTML 和 XML 文档，这是因为最常用的操作系统都带有记事本，而且它很容易使用。从个人来讲，我经常使用记事本来快速地编辑某些简单的 HTML、CSS 以及 XML 文件。

但是，如果您将记事本用于 XML 编辑，可能很快会发现不少问题。

记事本不能确定您编辑的文档类型，所以也就无法辅助您的工作。

## 为什么使用 XML 编辑器?

当今，XML 是非常重要的技术，并且开发项目正在使用这些基于 XML 的技术：

- 用 XML Schema 定义 XML 的结构和数据类型
- 用 XSLT 来转换 XML 数据
- 用 SOAP 来交换应用程序之间的 XML 数据
- 用 WSDL 来描述网络服务
- 用 RDF 来描述网络资源
- 用 XPath 和 XQuery 来访问 XML 数据
- 用 SMIL 来定义图形

为了能够编写出无错的 XML 文档，您需要一款智能的 XML 编辑器！

## XML 编辑器

专业的 XML 编辑器会帮助您编写无错的 XML 文档，根据某种 DTD 或者 schema 来验证 XML，以及强制您创建合法的 XML 结构。

XML 编辑器应该能够：

- 为开始标签自动添加结束标签
- 强制您编写合法的 XML
- 根据某种 DTD 来验证 XML
- 根据某种 Schema 来验证 XML
- 对您的 XML 语法进行代码的颜色化

---

在菜鸟教程，我们多年来一直使用 XMLSpy。 XMLSpy 是我们最喜爱的 XML 编辑器。这里是我们特别喜欢的一些特点：

- 在 32 位和 64 位版本中可用
- 使用方便
- 上下文敏感的人们帮手
- 语法着色和漂亮的印刷
- 智能修复验证与自动校正错误
- 文本视图和网格视图之间轻松切换
- 图形化的 XML Schema 编辑器
- 所有主流数据库的数据库导入导出
- SharePoint® 服务器支持
- 内置许多 XML 文档类型的模板
- 显示 XML 数据的图表创建
- XPath 1.0/2.0 的智能自动完成
- XSLT 1.0/2.0 编辑器、分析器和调试器
- XQuery 编辑器、分析器和调试器
- SOAP 客户端和调试器
- 图像化的 WSDL 1.1/2.0 编辑器
- XBRL 验证 & 分类编辑
- 支持 Office 2007 / OOXML
- Java、C++ 和 C# 的代码生成
- HTML5 和 CSS3 支持

了解更多关于 XMLSpy

XMLSpy 是 Altova MissionKit® 的 XML 软件套件的六个工具之一。

了解更多用于 XML 开发的 Altova MissionKit。

□ 现实生活中的 XML　　XML E4X □

□点我分享笔记

反馈/建议

# XML - E4X

E4X 向 JavaScript 添加了对 XML 的直接支持。

## E4X 实例

```
var employees=
<employees>
<person>
<name>Tove</name>
<age>32</age>
</person>
<person>
<name>Jani</name>
<age>26</age>
</person>
</employees>;

document.write(employees.person.(name == "Tove").age);
```

这个实例仅适用于 **Firefox**!

尝试一下 »

## 作为一个 JavaScript 对象的 XML

E4X 是正式的 JavaScript 标准，增加了对 XML 的直接支持。

使用 E4X，您可以用声明 Date 或 Array 对象变量的方式声明 XML 对象变量：

```
var x = new XML()

var y = new Date()

var z = new Array()
```

## E4X 是一个 ECMAScript（JavaScript）标准

ECMAScript 是 JavaScript 的正式名称。ECMA-262（JavaScript 1.3）是在 1999 年 12 月标准化的。

E4X 是 JavaScript 的扩展，增加了对 XML 的直接支持。ECMA-357（E4X）是在 2004 年 6 月标准化的。

ECMA 组织（成立于 1961 年），是专门用于信息和通信技术（ICT）和消费电子（CE）的标准化。 ECMA 制定的标准为：

- JavaScript
- C# 语言
- 国际字符集
- 光盘
- 磁带
- 数据压缩
- 数据通信
- 等等...

## 没有使用 E4X

下面的实例是一个跨浏览器的实例，实例加载一个现有的 XML 文档（"note.xml"）到 XML 解析器，并显示消息说明：

## 实例

```
var xmlDoc;
//code for Internet Explorer
if (window.ActiveXObject)
{
xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async=false;
xmlDoc.load("note.xml");
displaymessage();
}
// code for Mozilla, Firefox, etc.
else (document.implementation && document.implementation.createDocument)
{
xmlDoc= document.implementation.createDocument("","",null);
xmlDoc.load("note.xml");
xmlDoc.onload=displaymessage;
}

function displaymessage()
{
document.write(xmlDoc.getElementsByTagName("body")[0].firstChild.nodeValue);
}
```

尝试一下 »

## 使用 E4X

下面的实例是上面的实例相同，但是使用了 E4X:

```
var xmlDoc=new XML();
xmlDoc.load("note.xml");
document.write(xmlDoc.body);
```

简单多了，是不是？

## 浏览器支持

**Firefox** 是目前唯一对 E4X 的支持比较好的浏览器。

目前还没有支持 E4X 的有 **Opera**、**Chrome** 或 **Safari**。

到目前为止，没有迹象显示在 **Internet Explorer** 中对 E4X 的支持。

## E4X 的未来

E4X 没有得到广泛的支持。也许它提供的实用功能太少，尚未被其他的解决方案涉及：

对于完整的 XML 处理，您还需要学习 XML DOM 和 XPath

对于访问 XMLHttpRequests，JSON 是首选的格式。

对于简单的文档处理，JQuery 选择更容易。

XML 编辑器 XML 总结

# XML 总结 下一步学习什么呢？

## XML 总结

XML 可用于交换、共享和存储数据。

XML 文档形成 树状结构，在"根"和"叶子"的分支机构开始的。

XML 有非常简单的 语法规则。带有正确语法的 XML 是"形式良好"的。有效的 XML 是针对 DTD 进行验证的。

XSLT 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，可读取和操作 XML。

DOM（Document Object Model）定义了一个访问 XML 的标准方式。

XMLHttpRequest 对象提供了一个网页加载后与服务器进行通信的方式。

XML 命名空间提供了一种避免元素命名冲突的方法。

CDATA 区域内的文本会被解析器忽略。

我们的 XML 实例也代表了这个 XML 教程总结。

## 下一步学习什么呢？

我们推荐学习 XML DOM 和 XSLT。

如果您想要学习有关验证 XML 的知识，我们推荐学习 DTD 和 XML Schema。

下面是每个主题的一个简短描述。

## XML DOM（Document Object Model）

XML DOM 定义了一种访问和处理 XML 文档的标准方式。

XML DOM 是平台和语言独立的，可用于任何编程语言，如 Java、JavaScript 和 VBScript。

如果您想要学习更多有关 DOM 的知识，请访问我们的 XML DOM 教程。

## XSLT（XML 样式表语言转换）

XSLT 是 XML 文件的样式表语言。

通过使用 XSLT，可以把 XML 文档转换为其他格式，比如 XHTML。

如果您想要学习更多有关 XSLT 的知识，请访问我们的 XSLT 教程。

## XML DTD（文档类型定义）

DTD 的目的是定义 XML 文档中合法的元素、属性和实体。

通过使用 DTD，每个 XML 文件可以随身携带它自己的格式的描述。

DTD 可以被用来确认您收到的数据和您自己的数据是否有效。

如果您想要学习更多有关 DTD 的知识，请访问我们的 DTD 教程。

## XML Schema

XML Schema 是一种基于 XML 的 DTD 替代。

不像 DTD，XML Schema 支持数据类型，且使用 XML 语法。

如果您想要学习更多有关 XML Schema 的知识，请访问我们的 XML Schema 教程。

# XML 实例

这些例子演示了 XML 文件、XML 格式化和 XML 转换（XSLT）。

还演示了与 XML 一起使用的 JavaScript（AJAX）。

**查看 XML 文件**

查看一个简单的 XML 文件（note.xml）

查看一个带有错误的相同的 XML 文件

查看一个 XML 的 CD 目录

查看一个 XML 的植物目录

查看一个 XML 的食物菜单

**实例解释**

**XML 和 CSS**

查看一个 XML 的 CD 目录

查看对应的 CSS 文件

显示由 CSS 文件格式化的 CD 目录

**实例解释**

**XML 和 XSLT**

查看一个 XML 的食物菜单

查看对应的 XSLT 样式表

显示由 XSLT 样式表格式化的食物菜单

**实例解释**

**解析 XML 和 XML DOM**

查看一个简单的 XML 文件（note.xml）

解析 XML 文件 - 跨浏览器实例

解析 XML 字符串 - 跨浏览器实例

**实例解释**

**XML to HTML**

查看一个 XML 的 CD 目录

把 XML 数据显示为 HTML 表格

**实例解释**

**XML 应用程序**

查看一个 XML 的 CD 目录

在 div 元素中显示 XML 数据

XML 节点导航

一个简单的 CD 目录应用程序

实例解释

**来自服务器的 XML 输出**

查看 ASP 如何返回 XML

查看 PHP 如何返回 XML

查看来自数据库的 XML 输出

实例解释

**XML DOM 高级**

获取 XML 元素的值

获取 XML 属性的值

更改 XML 元素的值

把一个新的属性添加到 XML 元素

创建一个新的 XML 元素

删除一个 XML 元素

实例解释

□ XML 总结

□ 点我分享笔记

反馈/建议