DTD 实例 ➔

DTD 教程

DTD（文档类型定义）的作用是定义 XML 文档的合法构建模块。

DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用。

现在开始学习DTD!

DTD 新闻实例

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

]>
```

DTD 实例 ➔

DTD 实例 DTD 构建模块

DTD 简介

文档类型定义（DTD）可定义合法的XML文档构建模块。它使用一系列合法的元素来定义文档的结构。

DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用。

内部的 DOCTYPE 声明

假如 DTD 被包含在您的 XML 源文件中，它应当通过下面的语法包装在一个 DOCTYPE 声明中：

```
<!DOCTYPE root-element [element-declarations]>
```

带有 DTD 的 XML 文档实例（请在 IE5 以及更高的版本打开，并选择查看源代码）：

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

在您的浏览器中打开此 XML 文件，并选择"查看源代码"命令。

以上 DTD 解释如下：

- **!DOCTYPE note** (第二行)定义此文档是 **note** 类型的文档。
- **!ELEMENT note** (第三行)定义 **note** 元素有四个元素："to、from、heading,、body"
- **!ELEMENT to** (第四行)定义 **to** 元素为 "#PCDATA" 类型
- **!ELEMENT from** (第五行)定义 **from** 元素为 "#PCDATA" 类型
- **!ELEMENT heading** (第六行)定义 **heading** 元素为 "#PCDATA" 类型
- **!ELEMENT body** (第七行)定义 **body** 元素为 "#PCDATA" 类型

外部文档声明

假如 DTD 位于 XML 源文件的外部，那么它应通过下面的语法被封装在一个 DOCTYPE 定义中：

```
<!DOCTYPE root-element SYSTEM "filename">
```

这个 XML 文档和上面的 XML 文档相同，但是拥有一个外部的 DTD:（点击打开该文件，并选择"查看源代码"命令。）

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
```

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

这是包含 DTD 的 "note.dtd" 文件：

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

为什么使用 DTD？

通过 DTD，您的每一个 XML 文件均可携带一个有关其自身格式的描述。

通过 DTD，独立的团体可一致地使用某个标准的 DTD 来交换数据。

而您的应用程序也可使用某个标准的 DTD 来验证从外部接收到的数据。

您还可以使用 DTD 来验证您自身的数据。

← DTD 实例 DTD 构建模块 →

← DTD 简介 DTD 元素 →

DTD - XML 构建模块

XML 和 HTML文档的主要的构建模块是元素标签。

XML 文档构建模块

所有的 XML 文档 (以及 HTML 文档) 均由以下简单的构建模块构成：

- 元素
- 属性
- 实体
- PCDATA
- CDATA

元素

元素是 XML 以及 HTML 文档的**主要构建模块。**

HTML 元素的例子是 "body" 和 "table"。XML 元素的例子是 "note" 和 "message" 。元素可包含文本、其他元素或者是空的。空的 HTML 元素的例子是 "hr"、"br" 以及 "img"。

实例:

```
<body>some text</body>

<message>some text</message>
```

属性可提供**有关元素的额外信息。**

属性总是被置于某元素的开始标签中。属性总是以**名称/值**的形式成对出现的。下面的 "img" 元素拥有关于源文件的额外信息：

```
<img src="computer.gif" />
```

元素的名称是 "img"。属性的名称是 "src"。属性的值是 "computer.gif"。由于元素本身为空，它被一个 " /" 关闭。

实体

实体是用来定义普通文本的变量。实体引用是对实体的引用。

大多数同学都了解这个 HTML 实体引用：" "。这个"无折行空格"实体在 HTML 中被用于在某个文档中插入一个额外的空格。

当文档被 XML 解析器解析时，实体就会被展开。

实体引用	字符
&lt;	<

>	>
&	&
"	"
'	'

PCDATA

PCDATA 的意思是被解析的字符数据（parsed character data）。

可把字符数据想象为 XML 元素的开始标签与结束标签之间的文本。

PCDATA 是会被解析器解析的文本。这些文本将被解析器检查实体以及标记。

文本中的标签会被当作标记来处理，而实体会被展开。

不过，被解析的字符数据不应当包含任何 &、< 或者 > 字符；需要使用 &、< 以及 > 实体来分别替换它们。

CDATA

CDATA 的意思是字符数据（character data）。

CDATA 是不会被解析器解析的文本。在这些文本中的标签不会被当作标记来对待，其中的实体也不会被展开。

← DTD 简介 DTD 元素 →

DTD 构建模块 DTD 属性

DTD - 元素

在一个 DTD 中，元素通过元素声明来进行声明。

声明一个元素

在 DTD 中，XML 元素通过元素声明来进行声明。元素声明使用下面的语法：

```
<!ELEMENT element-name category>
或
<!ELEMENT element-name (element-content)>
```

空元素

空元素通过类别关键词EMPTY进行声明：

```
<!ELEMENT element-name EMPTY>

实例:

<!ELEMENT br EMPTY>

XML example:

<br />
```

只有 PCDATA 的元素

只有 PCDATA 的元素通过圆括号中的 #PCDATA 进行声明：

```
<!ELEMENT element-name (#PCDATA)>

实例:

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

带有任何内容的元素

通过类别关键词 ANY 声明的元素，可包含任何可解析数据的组合：

```
<!ELEMENT element-name ANY>

实例:

<!ELEMENT note ANY>
```

带有子元素（序列）的元素

带有一个或多个子元素的元素通过圆括号中的子元素名进行声明：

```
<!ELEMENT element-name (child1)>
或
<!ELEMENT element-name (child1,child2,...)>

实例：

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

当子元素按照由逗号分隔开的序列进行声明时，这些子元素必须按照相同的顺序出现在文档中。在一个完整的声明中，子元素也必须被声明，同时子元素也可拥有子元素。"note" 元素的完整声明是：

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

声明只出现一次的元素

```
<!ELEMENT element-name (child-name)>

实例：

<!ELEMENT note (message)>
```

上面的例子声明了：message 子元素必须出现一次，并且必须只在 "note" 元素中出现一次。

声明最少出现一次的元素

```
<!ELEMENT element-name (child-name+)>

实例：

<!ELEMENT note (message+)>
```

上面的例子中的加号（+）声明了：message 子元素必须在 "note" 元素内出现至少一次。

声明出现零次或多次的元素

```
<!ELEMENT element-name (child-name*)>

实例：

<!ELEMENT note (message*)>
```

上面的例子中的星号（*）声明了：子元素 message 可在 "note" 元素内出现零次或多次。

声明出现零次或一次的元素

```
<!ELEMENT element-name (child-name?)>

实例：

<!ELEMENT note (message?)>
```

上面的例子中的问号(?)声明了：子元素 message 可在 "note" 元素内出现零次或一次。

声明"非.../即..."类型的内容

实例：

```
<!ELEMENT note (to,from,header,(message|body))>
```

上面的例子声明了：“note" 元素必须包含 "to" 元素、"from" 元素、"header" 元素，以及非 "message" 元素既 "body" 元素。

声明混合型的内容

实例：

```
<!ELEMENT note (#PCDATA|to|from|header|message)*>
```

上面的例子声明了：“note" 元素可包含出现零次或多次的 PCDATA、"to"、"from"、"header" 或者 "message"。

← DTD 构建模块 DTD 属性 →

DTD 元素 XML 元素和属性比较

DTD - 属性

在 DTD 中，属性通过 ATTLIST 声明来进行声明。

声明属性

属性声明使用下列语法：

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>

DTD 实例:

<!ATTLIST payment type CDATA "check">

XML 实例:

<payment type="check" />
```

以下是 **属性类型**的选项：

类型	描述
CDATA	值为字符数据 (character data)
(*en1*|*en2*|..)	此值是枚举列表中的一个值
ID	值为唯一的 id
IDREF	值为另外一个元素的 id
IDREFS	值为其他 id 的列表
NMTOKEN	值为合法的 XML 名称
NMTOKENS	值为合法的 XML 名称的列表
ENTITY	值是一个实体
ENTITIES	值是一个实体列表
NOTATION	此值是符号的名称
xml:	值是一个预定义的 XML 值

默认**属性值**可使用下列值：

值	解释
值	属性的默认值

#REQUIRED	属性值是必需的
#IMPLIED	属性不是必需的
#FIXED value	属性值是固定的

默认属性值

```
DTD:
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">

合法的 XML:
<square width="100" />
```

在上面的例子中，"square" 被定义为带有 CDATA 类型的 "width" 属性的空元素。如果宽度没有被设定，其默认值为0 。

#REQUIRED

语法

```
<!ATTLIST element-name attribute-name attribute-type #REQUIRED>
```

实例

```
DTD:
<!ATTLIST person number CDATA #REQUIRED>

合法的 XML:
<person number="5677" />

非法的 XML:
<person />
```

假如您没有默认值选项，但是仍然希望强制作者提交属性的话，请使用关键词 #REQUIRED。

#IMPLIED

语法

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

实例

```
DTD:
<!ATTLIST contact fax CDATA #IMPLIED>

合法的 XML:
<contact fax="555-667788" />

合法的 XML:
<contact />
```

假如您不希望强制作者包含属性，并且您没有默认值选项的话，请使用关键词 #IMPLIED。

#FIXED

语法

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

实例

```
DTD:
<!ATTLIST sender company CDATA #FIXED "Microsoft">

合法的 XML:
<sender company="Microsoft" />

非法的 XML:
<sender company="W3Schools" />
```

如果您希望属性拥有固定的值，并不允许作者改变这个值，请使用 #FIXED 关键词。如果作者使用了不同的值，XML 解析器会返回错误。

列举属性值

语法

```
<!ATTLIST element-name attribute-name (en1|en2|..) default-value>
```

实例

```
DTD:
<!ATTLIST payment type (check|cash) "cash">

XML 例子:
<payment type="check" />
或
<payment type="cash" />
```

如果您希望属性值为一系列固定的合法值之一，请使用列举属性值。

← DTD 元素 XML 元素和属性比较 →

DTD 属性 DTD 实体

XML 元素 vs. 属性

在XML中，并没有规定何时使用属性，以及何时使用子元素。

使用元素 vs. 属性

数据可以存储在子元素或属性。

让我们来看下这些实例:

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

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

在第一个例子中"sex"是一个属性。在后面一个例子中，"sex"是一个子元素。但是两者都提供了相同的信息。

没有特别规定何时使用属性，以及何时使用子元素。我的经验是在 HTML 中多使用属性，但在XML中，使用子元素，会感觉更像数据信息。

我喜欢的方式

我喜欢在子元素中存储数据

下面的三个XML文档包含完全相同的信息：

本例中使用"date"属性：

```
<note date="12/11/2002">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

本例中使用"date"元素：

```
<note>
  <date>12/11/2002</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

本例中使用了扩展的"date" 元素: (这是我最喜欢的方式):

```
<note>
  <date>
    <day>12</day>
    <month>11</month>
    <year>2002</year>
```

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

避免使用属性?

你应该避免使用属性?

一些属性具有以下问题:

- 属性不能包含多个值（子元素可以）
- 属性不容易扩展（为以后需求的变化）
- 属性无法描述结构（子元素可以）
- 属性更难以操纵程序代码
- 属性值是不容易测试，针对DTD

如果您使用属性作为数据容器，最终的XML文档将难以阅读和维护。 尝试使用**元素**来描述数据。只有在提供的数据是不相关信息时我们才建议使用属性。

不要这个样子结束（这不是XML应该使用的）：

```
<note day="12" month="11" year="2002"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

一个属性规则的例外

规则总是有另外的

关于属性的规则我有一个例外情况。

有时我指定的 ID 应用了元素。这些 ID 应用可在HTML中的很多相同的情况下可作为 NAME 或者 ID 属性来访问 XML 元素。以下实例展示了这种方式：

```
<messages>
<note id="p501">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

<note id="p502">
  <to>Jani</to>
  <from>Tove</from>
  <heading>Re: Reminder</heading>
  <body>I will not!</body>
</note>
</messages>
```

以上实例的XML文件中，ID是只是一个计数器，或一个唯一的标识符，来识别不同的音符，而不是作为数据的一部分。

在这里我想说的是，元数据（关于数据的数据）应当存储为属性，而数据本身应当存储为元素。

DTD 属性 DTD 实体

XML 元素和属性比较 DTD 验证

DTD - 实体

实体是用于定义引用普通文本或特殊字符的快捷方式的变量。

- 实体引用是对实体的引用。
- 实体可在内部或外部进行声明。

一个内部实体声明

语法

```
<!ENTITY entity-name "entity-value">
```

实例

```
DTD 实例:

<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright runoob.com">

XML 实例:

<author>&writer;&copyright;</author>
```

注意： 一个实体由三部分构成: 一个和号 (&), 一个实体名称, 以及一个分号 (;)。

一个外部实体声明

语法

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

实例

```
DTD 实例:

<!ENTITY writer SYSTEM "http://www.runoob.com/entities.dtd">
<!ENTITY copyright SYSTEM "http://www.runoob.com/entities.dtd">

XML example:

<author>&writer;&copyright;</author>
```

XML 元素和属性比较 DTD 验证

DTD 实体 DTD 总结

DTD 验证

使用 Internet Explorer 可根据某个 DTD 来验证您的 XML。

通过 XML 解析器进行验证

当您试图打开某个 XML 文档时，XML 解析器有可能会产生错误。通过访问 parseError 对象，就可以取回引起错误的确切代码、文本甚至所在的行。

注意： load() 方法用于文件，而 loadXML() 方法用于字符串。

实例

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.validateOnParse="true";
xmlDoc.load("note_dtd_error.xml");

document.write("<br />Error Code: ");
document.write(xmlDoc.parseError.errorCode);
document.write("<br />Error Reason: ");
document.write(xmlDoc.parseError.reason);
document.write("<br />Error Line: ");
document.write(xmlDoc.parseError.line);
```

尝试一下 »

查看xml文件

关闭验证

通过把 XML 解析器的 validateOnParse 设置为 "false"，就可以关闭验证。

实例

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.validateOnParse="false";
xmlDoc.load("note_dtd_error.xml");

document.write("<br />Error Code: ");
document.write(xmlDoc.parseError.errorCode);
document.write("<br />Error Reason: ");
document.write(xmlDoc.parseError.reason);
document.write("<br />Error Line: ");
document.write(xmlDoc.parseError.line);
```

尝试一下 »

通用的 XML 验证器

为了帮助您验证 XML 文件，我们创建了此 链接，这样你就可以验证任何 XML 文件了。

parseError 对象

您可以在我们的《XML DOM 教程》中阅读更多有关 parseError 对象的信息。

← DTD 实体 DTD 总结 →

DTD 教程 DTD 简介

DTD - 来自网络的实例

电视节目表 DTD

由 David Moisan 创造。拷贝自： http://www.davidmoisan.org/

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

报纸文章 DTD

拷贝自：http://www.vervet.com/

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic Press">

]>
```

产品目录 DTD

拷贝自： http://www.vervet.com/

```
<!DOCTYPE CATALOG [

<!ENTITY AUTHOR "John Doe">
<!ENTITY COMPANY "JD Power Tools, Inc.">
<!ENTITY EMAIL "jd@jd-tools.com">

<!ELEMENT CATALOG (PRODUCT+)>

<!ELEMENT PRODUCT
(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
<!ATTLIST PRODUCT
NAME CDATA #IMPLIED
CATEGORY (HandTool|Table|Shop-Professional) "HandTool"
PARTNUM CDATA #IMPLIED
PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
INVENTORY (InStock|Backordered|Discontinued) "InStock">

<!ELEMENT SPECIFICATIONS (#PCDATA)>
<!ATTLIST SPECIFICATIONS
WEIGHT CDATA #IMPLIED
POWER CDATA #IMPLIED>

<!ELEMENT OPTIONS (#PCDATA)>
<!ATTLIST OPTIONS
FINISH (Metal|Polished|Matte) "Matte"
ADAPTER (Included|Optional|NotApplicable) "Included"
CASE (HardShell|Soft|NotApplicable) "HardShell">

<!ELEMENT PRICE (#PCDATA)>
<!ATTLIST PRICE
MSRP CDATA #IMPLIED
WHOLESALE CDATA #IMPLIED
STREET CDATA #IMPLIED
SHIPPING CDATA #IMPLIED>

<!ELEMENT NOTES (#PCDATA)>

]>
```

← DTD 教程 DTD 简介 →

DTD 验证

您已经学习了 DTD，下一步呢？

DTD 总结

本教程已经向您讲解了如何描述 XML 文档的结构。

您学习到了如何使用 DTD 来定义一个 XML 文档的合法元素，以及如何在您的 XML 内部或者作为一个外部引用来声明 DTD。

您已经学习了如何为 XML 文档声明合法的元素、属性、实体以及 CDATA 部分。

您也看到了如何根据某个 DTD 来验证一个 XML 文档。

您已经学习了 DTD，下一步学习什么内容呢？

下一步应当学习 XML Schema。

XML Schema 用于定义 XML 文档的合法元素，类似 DTD。我们认为 XML Schema 很快会将 DTD 取而代之，被用在大部分的网络应用程序中。

XML Schema 是基于 XML 的 DTD 替代物。

与 DTD 不同，XML Schema 支持数据类型和命名空间。

如果您希望学习更多有关 XML Schema 的知识，请访问我们的《XML Schema 教程》。

DTD 验证