## 第12章 HTML

|  |
| --- |
| XPath是一种专门用来在XML文档中查找信息的语言，对XPath的一种较好理解的解释是：如果把XML文档看成是一个文本数据库，XPath就类似于数据库领域的SQL结构化查询语言。本章讨论的问题包括：1. XPath的作用及工作原理；2. XPath的结点类型、表达式、数据类型与函数；3. XPath 2.0的新特性。  本章学习目标：  （1）了解XPath的工作原理；  （2）掌握XPath的定位路径表达式；  （3）熟悉XPath的常用函数和数据类型；  （3）掌握XPath2.0的部分新特性。 |

* 1. **概述**

Web是目前人们获取信息的最为快捷的方式，而Web文档正是由超文本标记语言HTML所编写，本章将对HTML的有关内容作简单介绍，以方便读者对XML的测试。

* + 1. **XPath及其作用**

W3C设计XPath的最初目的是为了满足XSLT和XPointer的需要，为它们操纵XML文档提供公共的句法和语义功能，其主要功能是为XML文档中特定元素、属性等结点信息提供寻址定位支持，同时，XPath还支持对字符串、数字和布尔型数值的操纵处理。XPath采用了一种紧凑的、非XML方式的语法结构，通过XML文档的树状逻辑结构进行位置描述。

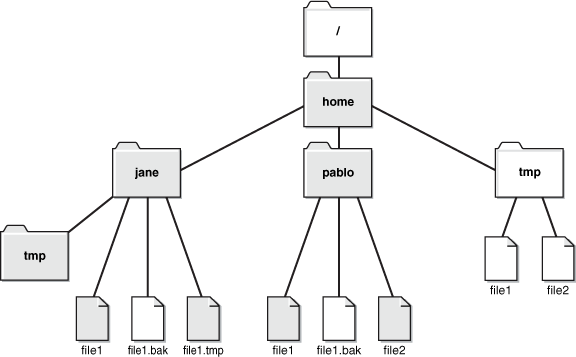


图6-1 操作系统中的文件系统可以看成一棵层次树

如图6-1所示，在计算机文件系统中，所有的文件（文件夹又称为文件目录，可看作是一种特殊类型的文件）构成了一棵具有层次关系的树状结构，此时，文件的定位可以通过层次树中的路径（Path）进行表示；XPath也采用了这一成熟概念，把整个XML文档看成是一棵由结点组成的层次树，通过结点路径进行定位，这正是XPath名称的由来，即XML Path Language。

除了基本的寻址功能之外，XPath还提供了用于匹配结点的内置函数集合，用于测试指定结点是否满足特殊的模式串。最新版本的XPath语言规范是W3C于2010年12月14日发布的XPath 2.0，可以从W3的网站上看到有关该规范的详细信息[[1]](#footnote-1)。XPath 2.0是XPath 1.0的超集，增加了更加丰富的数据类型，并与XPath 1.0保持兼容，在未来的一段时间里，将会有越来越多的解析器对XPath 2.0实现完全支持。

简单说来，XPath的作用有以下几点：

（1）XPath为XML文档提供一种快捷方便并易于使用的寻址功能，实现对XML文档树中指定结点或结点集合的选择定位。

（2）XPath为XML其他相关技术提供核心支持，包括XSLT、XQuery、XPointer、XForm等。

（3）XPath为人们处理XML文档提供了一种标准通用规范，XPath的公共API接口独立于特定语言，使得对XML文档的操纵处理更为方便。

* + 1. **XPath的工作原理**

从前面的章节已经知道，一个XML文档是由标记、属性、处理指令、注释等组成的具有嵌套特点的结构化文档，由简单的文本字符序列组合而成。人们在编写和阅读XML文档时，一般都是采用从左到右、从上到下的方式。例如6-1所示的一个简单的XML文档。

【例6-1】 一个简单的XML文档，保存在“6-1.1.xml”中。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?xml version="1.0" encoding="UTF-8"?>  <order id="TEST-01" >  <?backup dbname="order"?>  <product name="打印纸" price="30" quantity="5"/>  <product name="圆珠笔" price="2" quantity="10"/>  <product name="铅笔" price="1.5" quantity="10"/>  <product name="白板" price="80" quantity="2"/>  </order> |

在例6-1中，我们可以把整个XML文档看成是由“<?xml version=…”组成的一段连续的文本字符序列，从这个方面看，它与普通的文本文件相同，因此我们又可以把它称为序列化XML文档（Serialized XML Document）。

然而，在实际查看和使用一个XML文档时，人们更重视它的逻辑结构特征。为了描述XML文档的逻辑模型，W3C开发出了三个不同的规范，分别是DOM、XPath和XML信息集合（XML Information Set），它们从三个不同的角度来对XML文档进行表示。

**（1）XPath数据模型**

XPath数据模型在表示序列化XML文档时，把每一个部分看成是一个或一组文档树结点。在这个数据模型表示中，XML文档的多数组成部分都可以表示成结点的形式。其中，每个文档都有一个根节点，用于表示文档本身；元素结点用来表示文档中的元素，属性结点用来表示文档中的属性，注释、处理指令等与此相同；文本结点则用来表示元素之间的文本内容。

并不是XML文档中的所有部分都与XPath数据模型中的某个结点相对应。XML文档最开始的XML声明语句、DOCTYPE声明都没有对应的表示方法。除此之外，注释和处理指令虽然可以由注释结点和处理指令结点表示，但对于包含在文档类型声明部分的注释和处理指令信息并没有被XPath数据模型表示出来。

**（2）文档对象模型DOM**

与XPath数据模型类似，文档对象模型DOM把整个XML文档表示成一棵由结点组成的层次树。但是，DOM和XPath在表示结点的方式上有所不同，DOM中的结点包含有丰富的信息，更多的应用于XML编程处理方面。DOM结点的有关内容将在后续章节中详细介绍。

**（3）XML信息集合**

也叫XML信息集，描述了XML文档的抽象表现形式。XML 信息集主要用作各种XML技术使用的定义集，以正式描述需要技术处理的XML文档部分。XML信息集合在表示XML文档时也把它看成是一棵层次树，但采用了与XPath和DOM都不相同的方式。

一个 XML 文档的信息集包含许多信息项，这些信息项是XML文档的抽象表示，每个信息项相当于XPath数据模型中的一个结点。XML信息集是XML文档里的纯信息表示，如果要从XML角度而非文本角度比较两个文件的异同，则信息集是常用的方法，信息集不区分空元素的两种形式，属性所使用的引号类型也不重要，克服了严格的文本字符比较的缺点，因此，例6-2与例6-1中的文档在信息集的角度上是完全等同的。

【例6-2】 文件“6-1.xml”更改后的代码清单，保存在“6-2.xml”中，该文档与“6-1.xml”在信息集合的角度上完全等同。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <order id='TEST-01' >  <?backup dbname="order"?>  <product name='打印纸' price='30' quantity='5'/>  <product name='圆珠笔' price='2' quantity='10'/>  <product name='铅笔' price='1.5' quantity='10'/>  <product name='白板' price='80' quantity='2'/>  </order> |

从以上三种不同的逻辑表示方式可以看出，XPath是一种对XML文档的逻辑描述方式，它把XML文档看成一棵树，通过结点层次关系实现文档内容的寻址和定位。以6-1为例，可把该文档看成是图6-2所示的一棵层次树。

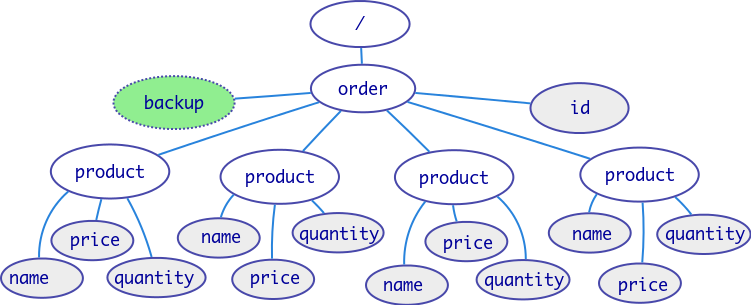


图6-2 “6-1.xml”文档的树状表示

上图中，白色背景的椭圆表示元素结点，灰色背景的椭圆表示属性结点，虚线椭圆则表示处理指令结点。根据这棵层次树，我们可以从根结点开始，定位寻址指定的元素。例如，要寻找第2个product结点，可以顺着第一个根节点“/”开始，在根节点的子结点中选择“order”结点，再找第2个“person”结点。XPath的基本原理就是利用这些结点之间的层次关系来进行定位的，它以元素或属性的标记名称代表结点，结点之间通过“/”进行分割，以这样形成的字符串表示XML指定部分的位置。因此，如果要定位第2个product元素的name属性， XPath表达式就可以写成：

/order/product[2]/@name

XPath表达式要求在属性名称前加上“@”符号，以区别于元素。

* + 1. **XPath的表达式与操作符**

XPath是一组语法和语义规范，用于引用XML文档的指定部分，如果之前未曾用过XPath，你或许会认为XPath的语法是基于XML格式的，但事实并非如此。XPath表达式由一组文本块组成，并通过一些特殊符号组合到一起。下面总结了XPath中支持的表达式类型，用来构建表达式的操作符以优先级递增的顺序从上到下、从左到右给出，也可以利用圆括号显式地控制操作符之间的优先级。

表6-1 XPath的表达式和操作符

|  |  |
| --- | --- |
| 表达式类型 | 操作符 |
| 定位路径表达式 | /, //, | |
| 布尔表达式 | or, and |
| 等式表达式 | =, != |
| 关系表达式 | <=, <, >=, > |
| 数值表达式 | +, -, div, mod, \*, -(unary) |

结点和结点集是XPath最为重要的概念之一。那到底什么是结点呢？结点可以看成是能够用于XPath定位步中用来实现定位功能的离散的、逻辑性的东西，XML文档中的任意一个元素、属性、处理指令、注释等，都可以是一个结点。

* + 1. **结点的基本属性**

对于一个XML文档来说，其结点有各种不同的属性或者特征，例如，在XPath字符串类型部分，我们已经提到XPath元素内容可以包括任意合法的字符串文本，除此之外， 与结点有关的属性还包括结点名称、文档结点顺序以及结点之间的“家族关系”。

* + - 1. **结点名称**

大部分结点（并不是所有的结点，根结点就没有名称）都有一个名称，为便于理解结点名称，我们还需要了解以下三个术语：限定名称、本地名称和扩展名称。

* 限定名称（Qualified name）

其缩写形式为QName，该术语直接取自于W3C的XML命名空间规范部分（http://www.w3.org/TR/REC-xml-names）。通常说来，结点的QName是XML文档实例中该结点的唯一标识符，它包括结点的命名空间前缀部分。

例如：<product>元素的QName为product；如果元素标记带有命名空间前缀，如<common:product>，那么它的QName就是整个的common:product。

* 本地名称（Local-name）

一个结点的本地名称是QName去除命名空间前缀后剩余的内容。例如，元素的QName是common:product，那么它的本地名称就是product。对于一个给定的结点，如果没有指定命名空间，其规范名称和本地名称则完全相同。

* 扩展名称（Expended-name）

当结点与特定的命名空间相关联时，它的扩展名称是由与命名空间关联的URI与本地名称共同构成的。扩展名称不关心命名空间前缀的具体名称，当两个元素的QName不同时，它们的扩展名称有可能相同——只要这两个元素的命名空间关联的URI相同，同时它们的本地名称也相同。

* 1. **HTML的元素与标签**

XPath1.0可以满足人们在处理XML方面的最基本需求，但随着XML应用的不断深入，人们需要XPath能够具备更为强大的功能，因此，W3C自20年开始，着手制定XPath2.0，并于2007年1月23日正式成为推荐标准，最新版本可从W3C官网获取。本节主要介绍XPath2.0中一些较为有用的新特性，供读者参考。

* + 1. **支持XML Schema的数据类型**

XPath1.0所支持的表达式只有结点集、布尔、数值和字符串四种类型，没有考虑日期等其他类型数据的处理。因此，XPath2.0引入了对XML Schema的19种简单类型的直接支持，并提供了与之配套的函数和操作符。

例如，在许多XML的应用当中，都需要有时间和日期数据，如创建时间、购买日期等，XPath2.0中所增加的W3C Schema中的日期和时间数据类型，具备自动验证日期时间数据是否有效的能力，可用于此类数据的表示，相比于XPath 1.0，具有明显优势。

与XPath1.0一样，XPath 2.0中也保留了7种不同的结点类型，不过，XPath1.0中的根结点（Root Node）在2.0中拥有了更加贴切的名称——文档节点（Document Node）。

* + 1. **更为丰富的处理函数**

XPath2.0中提供了更为丰富的处理函数，例如在字符串处理方面提供的函数有：

* tokenize()：拆分字符串，
* matches()：根据正则表达式测试字符串是否匹配。

在日期和时间处理方面，也提供了专门的处理函数，包括:

* current-date()
* current-DateTime()
* current-time()
* day-from-date()
* hour-from-time()
* years-from-yearMonthDuration()
  + 1. **支持序列**

XPath2.0通过for操作符，可以创建新的序列或处理已有的序列。例如，在XPath 2.0的测试输入框中，输入如下表达式：

for $i in 1 to 6 return $i\*$i

该表达式会生成序列（1,4,9,16,25,36），建议读者利用XML编辑器进行测试验证。另外，在XPath2.0中，可以通过“$”符号加合法的字符串创建变量，如以上表达式中的“$i”。

【例6-5】XPath2.0的序列支持示例

1. 通过oXygen XML Editor或XML Spy编辑工具打开“6-1.xml”。
2. 在XML工具的XPath输入框中，输入如下表达式，并观察结果。

表达式：for $p in //product return $p/@price\*$p/@quantity

返回结果：由价格与数量的乘机构成的数字序列，即（150.0，20.0， 15.0， 160.0）。

1. 利用sum()函数，对上一序列的结果求和。

表达式：sum(for $p in //product return $p/@price\*$p/@quantity)

返回结果：345.0

在上例中，人们会觉得第2步返回的是一个序列，而第3步返回的则是一个数值。事实上，XPath2.0中所有表达式的返回值都是序列，例子中第3步通过sum函数返回的总和也是一个序列，只不过该序列仅由一个成员构成。在平常交流过程中，人们经常会说表达式返回一个“数字”或“字符串”，其本质是指“一个数值构成的序列”或“一个字符串构成的序列”，两种说法在此场景下都可接受，并无本质区别。总之，XPath2.0中，一切皆为序列。

另外，XPath2.0中的序列还拥有如下特点：

* 序列很浅，不存在序列之序列的情况，如果表达式指定了嵌套序列，XPath2.0会自动平铺展开。

例如：序列( (1, 2), (3, 4), (5, 6) )，经XPath变换后，等价于序列(1, 2, 3, 4, 5, 6)。

* 序列中的成员拥有顺序关系。
* 序列中的成员可以重复出现。
  + 1. **支持逻辑判断**

XPath 2.0支持if/else的逻辑判断结构。

例如，在例6-5的基础上，如果要根据订单的总额，输出当前订单是否为大额订单信息，可以采用如下的XPath表达式：

if( sum(for $p in //product return $p/@price\*$p/@quantity) > 200) then "YES" else "NO"

针对“6-1.xml”文档来说，上述表达式将输出由一个"YES"字符串构成的序列。

* + 1. **更多的结点测试**

在XPath2.0中，可以根据结点的类型选择结点，例如，可以通过//element()选取文档中的全部元素结点，通过//attribute()选择所有的属性结点。

仍以“6-1.xml”文档，输入XPath表达式“//element()”，此时将返回由5个元素结点构成的序列，其中序列中的第一个结点为order元素，后续四个结点分别对应order元素下的4个product元素。

* + 1. **调用自定义函数**

XPath2.0还可以调用自定义函数。以电子商务中的订单处理为例，一个订单会包含客户所购买的多个商品，假设计算商品价格总和作为订单总额的逻辑处理较为复杂，需要考虑不同会员的折扣情况，我们可以把它定义为一个函数，令函数名为getOrderFee(orderId)，orderId代表订单编号，此时就可以像调用XPath本身的内置函数一样调用该函数，例如：

**getOrderFee(order/@orderId)**

需要注意的是，XPath2.0本身并不能建立自定义函数，我们只能在支持XPath的XSLT、XQuery等应用程序中建立自定义函数，供XPath调用。

* 1. **HTML的元素属性**
  2. **小结**

本章讨论了XPath1.0有关的结点与结点集的基本概念，XPath包含7中不同的结点类型，其中根结点在XPath2.0中又称之为文档结点。XPath主要通过定位路径表达式实现定位，一个定位路径表达式由若干个定位步骤组成，每一个定位步骤又由轴、结点测试和谓词构成。另外，XPath还提供了许多简写形式，以简化定位路径的编写。

除定位路径表达式之外，XPath1.0还支持布尔、等式、关系和数值表达式，并支持字符串、数值、布尔和结点集四种数据类型；本章还给出了与四种数据类型配套的常用函数。

最后，介绍了XPath2.0的部分新特性，包括：对XML Schema的支持、逻辑判断、序列支持、函数增强、结点测试和自定义函数。

* 1. **习题**

1. 简述XML文档三种不同的数据表示模型。
2. XPath的结点类型有哪些？
3. 各类编程语言的XML处理包中经常遇到QName的名字，请问QName在XML中代表什么含义？
4. XPath的定位路径表达式由哪些部分构成，并简述各部分的作用。
5. 假设下图左侧为一个XML文档构成的树，椭圆形表示元素结点，并假设当前上下文结点为F，请在右侧的表中根据轴的类型，填入轴所选中的结点代号。

|  |  |  |
| --- | --- | --- |
|  | 轴 | 选中的结点 |
| child | G, I |
| parent | B |
| ancestor | B, A, root |
| ancestor-or-self | F, B, A, root |
| descendant | G, H, I |
| descendant-or-self | F, G, H, I |
| following | J, K, L, M, N, … |
| following-sibling | J, K |
| preceding | E, D, C |
| preceding-sibling | E, C |
| self | F |

1. 请问XPath的数值类型能否支持科学计数法？
2. XPath2.0与1.0相比，有哪些新特点？
3. 对于缩写形式的定位路径表达式“//product[2]/@name”，请给出其完整写法。

1. http://www.w3.org/TR/xpath20/ [↑](#footnote-ref-1)