XPath 语法规则

一、 XPath 术语：

1. 节点：

   在 XPath 中，有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档（根）节点。XML 文档是被作为节点树来对待的。树的根被称为文档节点或者根节点。

   实例 1：XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book>
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

   上面的 XML 文档中的节点例子：

   <bookstore>（文档节点），<author>J K. Rowling</author>（元素节点），lang="en"（属性节点）。

2. 基本值（或称原子值，Atomic value）

   基本值是无父或无子的节点。基本值的例子：在实例 1 的 xml 文档中 J K. Rowling、"en"

3. 项目（Item）

   项目是基本值或者节点。

4. 节点的关系：

   1）父节点（Parent）：

   每个元素以及属性都有一个父。在上面实例 1 的 xml 文档中，book 元素是 title、author、year 以及 price 元素的父节点。

   2）子节点（Children）：

   元素节点可有零个、一个或多个子。在上面实例 1 的 xml 文档中，title、author、year 以及 price 元素都是 book 元素的子。

   3）同胞（Sibling）：

   拥有相同的父的节点，在上面实例 1 的 xml 文档中，title、author、year 以及 price 元素都是同胞。

   4）先辈（Ancestor）

   某节点的父、父的父，等等。在实例 1 的 xml 文档中，title 元素的先辈是 book 元素和 bookstore 元素。

   5）后代（Descendant）

   某个节点的子，子的子，等等。在实例 1 的 xml 文档中，bookstore 的后代是 book、title、author、year 以及 price 元素。

二、 XPath 语法：

XPath 使用路径表达式来选取 XML 文档中的节点或节点集。节点是通过沿着路径（path）或者步（steps）来选取的。我们将在下面的例子中使用这个 XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <book>
        <title lang="eng">Harry Potter</title>
        <price>29.99</price>
```

```
    </book>
    <book>
        <title lang="eng">Learning XML</title>
        <price>39.95</price>
    </book>
</bookstore>
```

1. 选取节点

XPath 使用路径表达式在 XML 文档中选取节点。节点是通过沿着路径或者 step 来选取的。下面列出了最有用的路径表达式：

表达式	描述
nodename	选取此节点的所有子节点。
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

实例：在下面的表格中，我们已列出了一些路径表达式以及表达式的结果：

路径表达式	结果
bookstore	选取 bookstore 元素的所有子节点。
/bookstore	选取根元素 bookstore。 注释：假如路径起始于正斜杠(/)，则此路径始终代表到某元素的绝对路径！
bookstore/book	选取属于 bookstore 的子元素的所有 book 元素。
//book	选取所有 book 子元素，而不管它们在文档中的位置。
bookstore//book	选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。
//@lang	选取名为 lang 的所有属性。

2. 谓语（Predicates）

谓语用来查找某个特定的节点或者包含某个指定的值的节点，谓语被嵌在方括号中。

实例：在下面的表格中，我们列出了带有谓语的一些路径表达式，以及表达式的结果：

路径表达式	结果
/bookstore/book[1]	选取属于 bookstore 子元素的第一个 book 元素。
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。

/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
/bookstore/book[position()<3]	选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
//title[@lang]	选取所有拥有名为 lang 的属性的 title 元素。
//title[@lang='eng']	选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。
/bookstore/book[price>35.00]	选取 bookstore 元素的所有 book 元素，且其中的 price 元素的值须大于 35.00。
/bookstore/book[price>35.00]/title	选取 bookstore 元素中的 book 元素的所有 title 元素，且其中的 price 元素的值须大于 35.00。

3. 选取未知节点

XPath 通配符可用来选取未知的 XML 元素。

通配符	描述
*	匹配任何元素节点。
@*	匹配任何属性节点。
node()	匹配任何类型的节点。

实例：在下面的表格中，我们列出了一些路径表达式，以及这些表达式的结果：

路径表达式	结果
/bookstore/*	选取 bookstore 元素的所有子元素。
//*	选取文档中的所有元素。
//title[@*]	选取所有带有属性的 title 元素。

4. 选取若干路径

通过在路径表达式中使用“|”运算符，您可以选取若干个路径。

实例：在下面的表格中，我们列出了一些路径表达式，以及这些表达式的结果：

路径表达式	结果
//book/title | //book/price	选取 book 元素的所有 title 和 price 元素。
//title | //price	选取文档中的所有 title 和 price 元素。
/bookstore/book/title | //price	选取属于 bookstore 元素的 book 元素的所有 title 元素,以及文档中所有的 price 元素。

三、 XPath Axes（坐标轴）

我们将在下面的例子中使用此 XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <bookstore>
    <book>
      <title lang="eng">Harry Potter</title>
      <price>29.99</price>
    </book>
    <book>
      <title lang="eng">Learning XML</title>
      <price>39.95</price>
    </book>
</bookstore>
```

1. XPath 轴

   轴可定义相对于当前节点的节点集。

轴名称	结果
ancestor	选取当前节点的所有先辈（父、祖父等）。
ancestor-or-self	选取当前节点的所有先辈（父、祖父等）以及当前节点本身。
attribute	选取当前节点的所有属性。
child	选取当前节点的所有子元素。
descendant	选取当前节点的所有后代元素（子、孙等）。
descendant-or-self	选取当前节点的所有后代元素（子、孙等）以及当前节点本身。
following	选取文档中当前节点的结束标签之后的所有节点。
namespace	选取当前节点的所有命名空间节点。
parent	选取当前节点的父节点。
preceding	选取文档中当前节点的开始标签之前的所有节点。
preceding-sibling	选取当前节点之前的所有同级节点。
self	选取当前节点。

2. 位置路径表达式

   位置路径可以是绝对的，也可以是相对的。绝对路径起始于正斜杠(/)，而相对路径不会这样。在两种情况中，位置路径均包括一个或多个步，每个步均被斜杠分割：

   绝对位置路径：/step/step/...

   相对位置路径：step/step/...

   每个步均根据当前节点集之中的节点来进行计算。

3. 步（step）包括：

轴（axis）：定义所选节点与当前节点之间的树关系。
节点测试（node-test）：识别某个轴内部的节点。
零个或者更多谓语（predicate）：更深入地提炼所选的节点集。
步的语法：轴名称::节点测试[谓语]。

实例：

例子	结果
child::book	选取所有属于当前节点的子元素的 book 节点。
attribute::lang	选取当前节点的 lang 属性。
child::*	选取当前节点的所有子元素。
attribute::*	选取当前节点的所有属性。
child::text()	选取当前节点的所有文本子节点。
child::node()	选取当前节点的所有子节点。
descendant::book	选取当前节点的所有 book 后代。
ancestor::book	选择当前节点的所有 book 先辈。
ancestor-or-self::book	选取当前节点的所有 book 先辈以及当前节点（如果此节点是 book 节点）
child::*/child::price	选取当前节点的所有 price 孙节点。

4. 常用函数：

last()　　节点数组的最后一个
position()　　下标索引，从 1 开始
text()　　节点的文本内容
contains(@attr, 'value')　　模糊匹配
starts-with(@attr, 'value')　　是否以指定字符开头
count(@attr)　　匹配次数
string-length()　　返回字符串的字符数,你应该用<替代<, 用>代替>
normalize-space()　　函数删除了前部和尾部的空格, 并且把连续的空格串替换为一个单一的空格

四、 XPath 运算符：

运算符	描述	实例	返回值
|	计算两个节点集	//book | //cd	返回所有拥有 book 和 cd 元素的节点集
+	加法	6 + 4	10
-	减法	6 - 4	2
*	乘法	6 * 4	24
div	除法	8 div 4	2

=	等于	price=9.80	如果 price 是 9.80，则返回 true。如果 price 是 9.90，则返回 false。
!=	不等于	price!=9.80	如果 price 是 9.90，则返回 true。如果 price 是 9.80，则返回 false。
<	小于	price<9.80	如果 price 是 9.00，则返回 true。如果 price 是 9.90，则返回 false。
<=	小于或等于	price<=9.80	如果 price 是 9.00，则返回 true。如果 price 是 9.90，则返回 false。
>	大于	price>9.80	如果 price 是 9.90，则返回 true。如果 price 是 9.80，则返回 false。
>=	大于或等于	price>=9.80	如果 price 是 9.90，则返回 true。如果 price 是 9.70，则返回 false。
or	或	price=9.80 or price=9.70	如果 price 是 9.80，则返回 true。如果 price 是 9.50，则返回 false。
and	与	price>9.00 and price<9.90	如果 price 是 9.80，则返回 true。如果 price 是 8.50，则返回 false。
mod	计算除法的余数	5 mod 2	1

五、 XPath 实例

实例一：

我们将在下面的例子中使用这个 XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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">XQuery Kick Start</title>
      <author>James McGovern</author>
      <author>Per Bothner</author>
      <author>Kurt Cagle</author>
      <author>James Linn</author>
      <author>Vaidyanathan Nagarajan</author>
```

```
    <year>2003</year>
    <price>49.99</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>
```

1. 选取所有 title
   /bookstore/book/title/text()
2. 选取第一个 book 的 title
   /bookstore/book[1]/title/text()
3. 选取所有价格
   /bookstore/book/price/text()
4. 选取价格高于 35 的 price 节点
   /bookstore/book[price>35]/price
5. 选取价格高于 35 的 title 节点
   /bookstore/book[price>35]/title
6. 选取所有的 book 节点
   /bookstore/book
7. 选取第一个 book 节点
   /bookstore/book[0]/text()
8. 选取 price
   /bookstore/book/price/text()
9. 选取价格高于 35 的 price 价格
   /bookstore/book[price>35]/price
10. 选取价格高于 35 的 title 节点
    /bookstore/book[price>35]/title
11. 选取属于 bookstore 子元素的最后一个 book 元素。
    /bookstore/book[last()]
12. 选取属于 bookstore 子元素的倒数第二个 book 元素。
    /bookstore/book[last()-1]
13. 选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
    /bookstore/book[position()<3]
14. 选取 ID 为 1，标题中含有 Java 的 book 元素.
    /bookstore/book[@id='1' and contains(@title, 'Java')]

实例二：

我们将利用下面的 XML 文档描述 XPath 语法

```
<?xml version="1.0" encoding="GB2312"?>
<order>
  <item catalog="parts">
    <itemNumber>C2688-67037</itemNumber>
    <description>LCD 液晶显示器</description>
    <quantity>1</quantity>
    <price>358.00</price>
  </item>
```

```
  <item catalog="parts">
    <itemNumber>C2688-67061</itemNumber>
    <description>音箱</description>
    <quantity>1</quantity>
    <price>16.50</price>
  </item>
  <item catalog="parts">
    <itemNumber>C2688-67010</itemNumber>
    <description>鼠标</description>
    <quantity>1</quantity>
    <price>8.50</price>
  </item>
</order>
```

节点定位：一个 XPath 的模式是使用反斜杠“/”分开子元素名称描述路径

下面的 XPath 表达式选择元素 order 下元素 item 中的所有 price 元素

```
/order/item/price
```

下面的 XPath 表达式选择文档中所有的 item 元素

```
//item
```

选择未知元素:通配符 “*”可用于选择未知 XML 元素

下面的 XPath 表达式选择元素 order 中的所有 item 元素所属的子元素

```
/order/item/*
```

下面的 XPath 表达式选择元素 order 下所有孙子辈的 price 元素

```
/order/*/price
```

下面的 XPath 表达式选择所有具有两个祖先的 price 元素

```
/*/*/price
```

下面的 XPath 表达式选择文档所有元素

```
//*
```

选择分支:使用方括号[]可以指定特定的元素

下面的 XPath 表达式选择元素 order 中的第一个 item 的子元素

```
/order/item[1]
```

下面的 XPath 表达式选择元素 order 中的最后一个 item 的子元素

```
/order/item[last()]
```

下面的 XPath 表达式选择元素 order 中具有 price 元素的 item 元素

```
/order/item[price]
```

下面的 XPath 表达式,从元素 order 中选择具有 price 等于 12.60 元素的 item 元素

```
/order/item[price=16.50]
```

下面的 XPath 表达式,从隶属于元素 order 的 item 元素中选择具有 price 等于 12.60 元素的 price 元素

```
/order/item[price=16.50]/price
```

选择几个路径:

下面的 XPath 表达式,从隶属于 order 的 item 元素中选择所有 itemNumber 和 description 元素

```
/order/item/itemNumber | /order/item/description
```

下面的 XPath 表达式,从文档中选择所有 itemNumber 和 description 元素

```
//itemNumber | //description
```

下面的 XPath 表达式,从文档中选择所有 itemNumber ， description 和 price 元素

```
//itemNumber | //description | //price
```

下面的 XPath 表达式,选取属于 order 中 item 下所有 itemNumber 元素和从文档中选择所有 description 元素

```
/order/item/itemnumber | //description
```

选择属性:在 XPath 中，所有属性使用@前缀

下面的 XPath 表达式,选取所有名为 catalog 的属性

//@catalog
下面的 XPath 表达式,选取所有具有 catalog 属性的 item 元素
//item[@catalog]
下面的 XPath 表达式,选取所有具有任何属性的 item 元素
//item[@*]
下面的 XPath 表达式,选取所有具有 catalog 等于"parts"属性的 item 元素
//item[@catalog="parts"]

%USERPROFILE%\Local Settings\Application Data\Microsoft\VisualStudio\10.0\Extensions\Whole Tomato Software\Visual Assist X\10.6.1845.0