## Xpath使用详解

### (一).XPath简介

XPath是W3C的一个标准。它最主要的目的是为了在XML1.0或XML1.1文档节点树中定位节点所设计。目前有XPath1.0和XPath2.0两个版本。其中Xpath1.0是1999年成为W3C标准，而XPath2.0标准的确立是在2007年。W3C关于XPath的英文详细文档请见：<http://www.w3.org/TR/xpath20/> 。

XPath是一种表达式语言，它的返回值可能是节点，节点集合，原子值，以及节点和原子值的混合等。XPath2.0是XPath1.0的超集。它是对XPath1.0的扩展，它可以支持更加丰富的数据类型，并且XPath2.0保持了对XPath1.0的相对很好的向后兼容性，几乎所有的XPath2.0的返回结果都可以和XPath1.0保持一样。另外XPath2.0也是XSLT2.0和XQuery1.0的用于查询定位节点的主表达式语言。

### (二).XPath路径表达式

#### 1). 路径表达式语法：

**绝对位置路径：**绝对路径起始于正斜杠( / )。 例如：/step/step/...  
**相对位置路径：**相对路径step/step/...   
**步进表达式**：轴::节点测试[谓词]

说明： 其中轴表示步进表达式选择的节点和当前上下文节点间的树状关系（层次关系），节点测试指定步进表达式选择的节点名称扩展名，谓词即相当于过滤表达式以进一步过滤细化节点集。 谓词可以是0个或多个。多个多个谓词用逻辑操作符and， or连接。取逻辑非用not()函数。

典型的XPath查询表达式举例：/messages/message//child::node()[@id=0]，其中/messages/message是路径（绝对路径以"/"开始），child::是轴表示在子节点下选择，node()是节点测试表示选择所有的节点。[@id=0]是谓词，表示选择所有有属性id并且值为0的节点。

#### 2). 谓词和轴的概念

XPath的谓词即筛选表达式，类似于SQL的where子句.

|  |  |
| --- | --- |
| **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 | 直到所有这个节点的父辈节点，顺序选择每个父辈节点前的所有同级节点 |
| preceding-sibling | 选取当前节点之前的所有同级节点。 |
| self | 选取当前节点。 |

#### 3). 运算符

|  |  |
| --- | --- |
| **运算符/特殊字符** | **说明** |
| / | 此路径运算符出现在模式开头时，表示应从根节点选择。 |
| // | 从当前节点开始递归下降，此路径运算符出现在模式开头时，表示应从根节点递归下降。 |
| . | 当前上下文。 |
| .. | 当前上下文节点父级。 |
| \* | 通配符；选择所有元素节点与元素名无关。（不包括文本，注释，指令等节点，如果也要包含这些节点请用node()函数） |
| @ | 属性名的前缀。 |
| @\* | 选择所有属性，与名称无关。 |
| : | 命名空间分隔符；将命名空间前缀与元素名或属性名分隔。 |
| ( ) | 括号运算符(优先级最高)，强制运算优先级。 |
| [ ] | 应用筛选模式（即谓词，包括"过滤表达式"和"轴（向前/向后）"）。 |
| [ ] | 下标运算符；用于在集合中编制索引。 |
| | | 两个节点集合的联合，如：//messages/message/to | //messages/message/cc |
| - | 减法。 |
| div， | 浮点除法。 |
| and, or | 逻辑运算。 |
| mod | 求余。 |
| not() | 逻辑非 |
| = | 等于 |
| ！= | 不等于 |
| 特殊比较运算符 | < 或者 &lt;  <= 或者 &lt;=  > 或者 &gt;  >= 或者 &gt;= |

#### 4). 函数及说明：

XPath函数和XSLT，XQuery等共享函数库，函数库为我们提供了功能丰富的各种函数的调用，我们也可以自定义自己的函数。这里不再对每个函数的用法逐一说明，英文意义参看W3C关于XPath函数的介绍吧W3：<http://www.w3.org/TR/xquery-operW3ators> 。中文的可以参考这个网站, <http://www.w3school.com.cn/xpath/xpath_functions.asp>

#### 5). 常用表达式实例：

|  |  |
| --- | --- |
| / | Document Root文档根. |
| /\* | 选择文档根下面的所有元素节点 |
| /node() | 根元素下所有的节点（包括文本节点，注释节点等） |
| /text() | 查找文档根节点下的所有文本节点 |
| /messages/message | messages节点下的所有message节点 |
| /messages/message[1] | messages节点下的第一个message节点 |
| /messages/message[1]/node() | 第一个message节点下的所有子节点 |
| /messages/message[1]/\*[last()] | 第一个message节点的最后一个子节点 |
| /messages/message[1]/node()[last()] | 第一个message节点的最后一个子节点 |
| /messages/message[1]//text() | 第一个message节点下递归下降查找所有的文本节点（无限深度） |
| /messages/message[1] /child::node()  /messages/message[1] /node()  /messages/message[position()=1]/node()  //message[@id=1] /node() | 第一个message节点下的所有子节点 |
| //message[@id=1] //child::node() | 递归所有子节点（无限深度） |
| //message[@id=0]/ancestor::\* | Ancestor轴表示所有的祖辈，父，祖父等。 向上递归 |
| //message[@id=1]/datetime/@date | 查找id=1的message节点的datetime节点的date属性 |
| //message[datetime] | 查找所有含有datetime节点的message节点 |
| //message/datetime[attribute::\*]  //message/datetime[attribute::node()]  //message/datetime[@\*]  //message/datetime[@node()] | 选择所有含有属性的datetime节点 |
| //message[@id=0]//books/\*[local-name()='book' and namespace-uri()='http://www.amazon.com/books/schema'] | 选择books下的所有的book节点，(节点名和命名空间都匹配) |