XPath 官方手册中文版

良少　整理　shendl_s@hotmail.com
http://blog.csdn.net/shendl/

来自于 http://www.zvon.org/

XPATH 指南

实例 1 基本的 XPath 语法类似于在一个文件系统中定位文件,如果路径以斜线 / 开始,那么该路径就表示到一个元素的绝对路径
实例 2 如果路径以双斜线 // 开头, 则表示选择文档中所有满足双斜线//之后规则的元素(无论层级关系)
实例 3 星号 * 表示选择所有由星号之前的路径所定位的元素
实例 4 方块号里的表达式可以进一步的指定元素, 其中数字表示元素在选择集里的位置, 而 last()函数则表示选择集中的最后一个元素.
实例 5
实例 6 属性的值可以被用来作为选择的准则, 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
实例 7 count()函数可以计数所选元素的个数
实例 8 name()函数返回元素的名称, start-with()函数在该函数的第一个参数字符串是以第二个参数字符开始的情况返回 true, contains()函数当其第一个字符串参数包含有第二个字符串参数时返回 true.
实例 9 string-length 函数返回字符串的字符数,你应该用<替代<, 用>代替>
实例 10 多个路径可以用分隔符 | 合并在一起
实例 11 child 轴(axis)包含上下文节点的子元素, 作为默认的轴,可以忽略不写.
实例 12

descendant (后代)轴包含上下文节点的后代,一个后代是指子节点或者子节点的子节点等等, 因此 descendant 轴不会包含属性和命名空间节点.
实例 13 parent 轴(axis)包含上下文节点的父节点, 如果有父节点的话
实例 14 ancestor 轴(axis)包含上下节点的祖先节点, 该祖先节点由其上下文节点的父节点以及父节点的父节点等等诸如此类的节点构成,所以 ancestor 轴总是包含有根节点,除非上下文节点就是根节点本身.
实例 15 following-sibling 轴(axis)包含上下文节点之后的所有兄弟节点
实例 16 preceding-sibling 轴(axis)包含上下文节点之前的所有兄弟节点
实例 17 following 轴(axis)包含同一文档中按文档顺序位于上下文节点之后的所有节点,除了祖先节点,属性节点和命名空间节点
实例 18 following 轴(axis)包含同一文档中按文档顺序位于上下文节点之前的所有节点,除了祖先节点,属性节点和命名空间节点
实例 19 descendant-or-self 轴(axis)包含上下文节点本身和该节点的后代节点
实例 20 ancestor-or-self 轴(axis)包含上下文节点本身和该节点的祖先节点
实例 21 ancestor, descendant, following, preceding 和 self 轴(axis)分割了 XML 文档(忽略属性节点和命名空间节点), 不能交迭, 而一起使用则包含所有节点
实例 22 div 运算符做浮点除法运算, mod 运算符做求余运算, floor 函数返回不大于参数的最大整数(趋近于正无穷), ceiling 返回不小于参数的最小整数(趋近于负无穷)

>> 实例 1 << | 前页 | 后页

基本的 XPath 语法类似于在一个文件系统中定位文件,如果路径以斜线 / 开始,那么该路径就表示到一个元素的绝对路径

/AAA
选择根元素 AAA
<AAA>   <BBB/>   <CCC/>   <BBB/>   <BBB/>   <DDD>   <BBB/>   </DDD>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/AAA/CCC
选择 AAA 的所有 CCC 子元素
<AAA>   <BBB/>   <CCC/>   <BBB/>   <BBB/>   <DDD>   <BBB/>   </DDD>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/AAA/DDD/BBB
选择 AAA 的子元素 DDD 的所有子元素
<AAA>   <BBB/>   <CCC/>

```
    <BBB/>
    <BBB/>
    <DDD>
        <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

The basic XPath syntax is similar to filesystem addressing. If the path starts with the slash / , then it represents an absolute path to the required element.
基本的 XPATH 语法类似于文件系统寻址。
如果以/开头，这表示绝对路径

/AAA

Select the root element AAA

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
        <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

Open the example in XLab. | Tree view (JPG)

/AAA/CCC

Select all elements CCC which are children of the root element AAA

```
<AAA>
    <BBB/>
```

```
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
      <BBB/>
    </DDD>
    <CCC/>
  </AAA>
```

Open the example in XLab. | Tree view (JPG)

>> 实例 2 << | 前页 | 后页

如果路径以双斜线 // 开头，则表示选择文档中所有满足双斜线//之后规则的元素(无论层级关系)
==========即，//后的是匹配的通配符，返回所有满足这个路径条件的节点。

//BBB

选择所有 BBB 元素

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
  </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//DDD/BBB
选择所有父元素是 DDD 的 BBB 元素
<AAA>   <BBB/>   <CCC/>   <BBB/>   <DDD>   <BBB/>   </DDD>   <CCC>   <DDD>   <BBB/>   <BBB/>   </DDD>   </CCC> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 3 << | 前页 | 后页

星号 * 表示选择所有由星号之前的路径所定位的元素

/AAA/CCC/DDD/*
选择所有路径依附于/AAA/CCC/DDD 的元素
<AAA>   <XXX>   <DDD>   <BBB/>   <BBB/>   <EEE/>   <FFF/>   </DDD>

```
    </XXX>
    <CCC>
      <DDD>
        <BBB/>
        <BBB/>
        <EEE/>
        <FFF/>
      </DDD>
    </CCC>
    <CCC>
      <BBB>
        <BBB>
          <BBB/>
        </BBB>
      </BBB>
    </CCC>
  </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

/*/*/BBB

选择所有的有 3 个祖先元素的 BBB 元素

```
  <AAA>
    <XXX>
      <DDD>
        <BBB/>
        <BBB/>
        <EEE/>
        <FFF/>
      </DDD>
    </XXX>
    <CCC>
      <DDD>
        <BBB/>
        <BBB/>
        <EEE/>
        <FFF/>
      </DDD>
    </CCC>
    <CCC>
```

```
      <BBB>
        <BBB>
          <BBB/>
        </BBB>
      </BBB>
    </CCC>
  </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*

选择所有元素

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 4 <<| 前页 | 后页

方块号里的表达式可以进一步的指定元素，其中数字表示元素在选择集里的位置，而 last()函数则表示选择集中的最后一个元素.
======类似于数组。但它是从 1 开始，而不是 C,Java 那样从 0 开始。

/AAA/BBB[1]
选择 AAA 的第一个 BBB 子元素
<AAA>   <BBB/>   <BBB/>   <BBB/>   <BBB/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/AAA/BBB[last()]
选择 AAA 的最后一个 BBB 子元素
<AAA>   <BBB/>   <BBB/>   <BBB/>   <BBB/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 5 <<| 前页 | 后页

====@xxx　是选择所有属性为 xxx 的属性

//@id
选择所有的 id 属性======这里返回的是 Attribute 属性对象。而不是节点 Node 对象。
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

//BBB[@id]
选择有 id 属性的 BBB 元素
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

//BBB[@name]
选择有 name 属性的 BBB 元素
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

//BBB[@*]

选择有任意属性的 BBB 元素

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//BBB[not(@*)]

选择没有属性的 BBB 元素

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 6 << | 前页 | 后页

属性的值可以被用来作为选择的准则, 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

=======这里，=表示等于。类似于 SQL 的语法，不同于 Java 和 C++的语法。

//BBB[@id='b1']

选择含有属性 id 且其值为'b1'的 BBB 元素

```
<AAA>
   <BBB id = "b1"/>
```

```
    <BBB name = " bbb "/>
    <BBB name = "bbb"/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//BBB[@name='bbb']

选择含有属性 name 且其值为'bbb'的 BBB 元素

```
<AAA>
    <BBB id = "b1"/>
    <BBB name = " bbb "/>
    <BBB name = "bbb"/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//BBB[normalize-space(@name)='bbb']

选择含有属性 name 且其值(在用 normalize-space 函数去掉前后空格后)为'bbb'的 BBB 元素

```
<AAA>
    <BBB id = "b1"/>
    <BBB name = " bbb "/>
    <BBB name = "bbb"/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

count()函数可以计数所选元素的个数

//*[count(BBB)=2]

选择含有 2 个 BBB 子元素的元素

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[count(*)=2]

选择含有 2 个子元素的元素

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[count(*)=3]
选择含有 3 个子元素的元素
<AAA>   <CCC>   <BBB/>   <BBB/>   <BBB/>   </CCC>   <DDD>   <BBB/>   <BBB/>   </DDD>   <EEE>   <CCC/>   <DDD/>   </EEE> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 8 << | 前页 | 后页

name()函数返回元素的名称, start-with()函数在该函数的第一个参数字符串是以第二个参数字符开始的情况返回 true, contains()函数当其第一个字符串参数包含有第二个字符串参数时返回 true.

//*[name()='BBB']
选择所有名称为 BBB 的元素(这里等价于//BBB)
<AAA>   <BCC>   <BBB/>   <BBB/>

```
      <BBB/>
    </BCC>
    <DDB>
      <BBB/>
      <BBB/>
    </DDB>
    <BEC>
      <CCC/>
      <DBD/>
    </BEC>
  </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[starts-with(name(),'B')]

选择所有名称以"B"起始的元素

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[contains(name(),'C')]

选择所有名称包含"C"的元素

```
<AAA>
  <BCC>
```

```
        <BBB/>
        <BBB/>
        <BBB/>
    </BCC>
    <DDB>
        <BBB/>
        <BBB/>
    </DDB>
    <BEC>
        <CCC/>
        <DBD/>
    </BEC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 9 <<| 前页 | 后页

string-length 函数返回字符串的字符数,你应该用<替代<, 用>代替>

//*[string-length(name()) = 3]

选择名字长度为 3 的元素

```
<AAA>
    <Q/>
    <SSSS/>
    <BB/>
    <CCC/>
    <DDDDDDDD/>
    <EEEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[string-length(name()) < 3]

选择名字长度小于 3 的元素

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//*[string-length(name()) > 3]

选择名字长度大于 3 的元素

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

多个路径可以用分隔符 | 合并在一起
=======用|表示或者。不同于 java 中用||表示或者。

//CCC | //BBB

选择所有的 CCC 和 BBB 元素

```
<AAA>
```

```
    <BBB/>
    <CCC/>
    <DDD>
       <CCC/>
    </DDD>
    <EEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

/AAA/EEE | //BBB

选择所有的 BBB 元素和所有是 AAA 的子元素的 EEE 元素

```
<AAA>
    <BBB/>
    <CCC/>
    <DDD>
       <CCC/>
    </DDD>
    <EEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

/AAA/EEE | //DDD/CCC | /AAA | //BBB

可以合并的路径数目没有限制

```
<AAA>
    <BBB/>
    <CCC/>
    <DDD>
       <CCC/>
    </DDD>
    <EEE/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 11 <<

child 轴(axis)包含上下文节点的子元素，作为默认的轴,可以忽略不写.
======child::前缀，表示只包含该节点（就是返回它自己），不包括该节点的子、孙节点。这是默认的。写不写都一样！

/AAA
等价于 /child::AAA
<AAA>   <BBB/>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/child::AAA
等价于/AAA
<AAA>   <BBB/>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/AAA/BBB
等价于/child::AAA/child::BBB
<AAA>   <BBB/>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/child::AAA/child::BBB
等价于/AAA/BBB
<AAA>   <BBB/>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

/child::AAA/BBB
二者都可以被合并
<AAA>   <BBB/>   <CCC/> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 12 << | 前页 | 后页

descendant (后代)轴包含上下文节点的后代,一个后代是指子节点或者子节点的子节点等等, 因此 descendant 轴不会包含属性和命名空间节点.

/descendant::*
选择文档根元素的所有后代.即所有的元素被选择
<AAA>   <BBB>   <DDD>   <CCC>

```
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

/AAA/BBB/descendant::*

选择/AAA/BBB 的所有后代元素

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//CCC/descendant::*

选择在祖先元素中有 CCC 的所有元素

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//CCC/descendant::DDD

选择所有以 CCC 为祖先元素的 DDD 元素

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
```

```
            <EEE>
               <DDD>
                  <FFF/>
               </DDD>
            </EEE>
         </DDD>
      </CCC>
   </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 13 << | 前页 | 后页

parent 轴(axis)包含上下文节点的父节点，如果有父节点的话

//DDD/parent::*

选择 DDD 元素的所有父节点

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 14 <<| 前页 | 后页

Ancestor(祖先)轴(axis)包含上下节点的祖先节点，该祖先节点由其上下文节点的父节点以及父节点的父节点等等诸如此类的节点构成,所以 ancestor 轴总是包含有根节点,除非上下文节点就是根节点本身.
=====注意，和父节点一样，把包含自身。
只有 child 是包含自身，其他前缀都不包含自身。

/AAA/BBB/DDD/CCC/EEE/ancestor::*

选择一个绝对路径上的所有节点

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//FFF/ancestor::*

选择 FFF 元素的祖先节点

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 15 << | 前页 | 后页

following-sibling 轴(axis)包含上下文节点之后的所有兄弟节点

/AAA/BBB/following-sibling::*

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
```

```
    <XXX>
      <DDD>
        <EEE/>
        <DDD/>
        <CCC/>
        <FFF/>
        <FFF>
          <GGG/>
        </FFF>
      </DDD>
    </XXX>
    <CCC>
      <DDD/>
    </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//CCC/following-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 16 <<| 前页 | 后页

preceding-sibling 轴(axis)包含上下文节点之前的所有兄弟节点

/AAA/XXX/preceding-sibling::*
<AAA> <BBB> <CCC/> <DDD/> </BBB> <XXX> <DDD> <EEE/> <DDD/> <CCC/> <FFF/> <FFF> <GGG/> </FFF> </DDD> </XXX> <CCC> <DDD/> </CCC> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

//CCC/preceding-sibling::*

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

following 轴(axis)包含同一文档中按文档顺序位于上下文节点之后的所有节点,除了祖先节点,属性节点和命名空间节点

/AAA/XXX/following::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
         <DDD>
```

```
            <EEE/>
         </DDD>
      </ZZZ>
      <FFF>
         <GGG/>
      </FFF>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//ZZZ/following::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
         <DDD>
            <EEE/>
         </DDD>
      </ZZZ>
      <FFF>
         <GGG/>
      </FFF>
   </BBB>
```

```
    <XXX>
      <DDD>
        <EEE/>
        <DDD/>
        <CCC/>
        <FFF/>
        <FFF>
          <GGG/>
        </FFF>
      </DDD>
    </XXX>
    <CCC>
      <DDD/>
    </CCC>
  </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 18 <<

| 前页 | 后页

preceding 轴(axis)包含同一文档中按文档顺序位于上下文节点之前的所有节点，除了祖先节点,属性节点和命名空间节点

/AAA/XXX/preceding::*

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
```

```
            <CCC/>
            <FFF/>
            <FFF>
               <GGG/>
            </FFF>
         </DDD>
      </XXX>
      <CCC>
         <DDD/>
      </CCC>
   </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//GGG/preceding::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
      </ZZZ>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 19 <<| 前页 | 后页

descendant-or-self 轴(axis)包含上下文节点本身和该节点的后代节点

/AAA/XXX/descendant-or-self::*
<AAA>   <BBB>   <CCC/>   <ZZZ>   <DDD/>   </ZZZ>   </BBB>   <XXX>   <DDD>   <EEE/>   <DDD/>   <CCC/>   <FFF/>   <FFF>   <GGG/>   </FFF>   </DDD>   </XXX>   <CCC>   <DDD/>   </CCC> </AAA>
在 XLab 中打开实例 | 树视图 (JPG)

//CCC/descendant-or-self::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
      </ZZZ>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

ancestor-or-self 轴(axis)包含上下文节点本身和该节点的祖先节点

/AAA/XXX/DDD/EEE/ancestor-or-self::*

```
<AAA>
   <BBB>
      <CCC/>
```

```
        <ZZZ>
          <DDD/>
        </ZZZ>
      </BBB>
      <XXX>
        <DDD>
          <EEE/>
          <DDD/>
          <CCC/>
          <FFF/>
          <FFF>
            <GGG/>
          </FFF>
        </DDD>
      </XXX>
      <CCC>
        <DDD/>
      </CCC>
    </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 21 <<| 前页 | 后页

ancestor, descendant, following, preceding 和 self 轴(axis)分割了 XML 文档(忽略属性节点和命名空间节点)，不能交迭，而一起使用则包含所有节点

注意：除了 child::放在节点名的前面（等于不放），其他轴都是放在节点名的后面。这样的形式：

节点名/轴::*

//GGG/ancestor::*

```
<AAA>
```

```
      <BBB>
         <CCC/>
         <ZZZ/>
      </BBB>
      <XXX>
         <DDD>
            <EEE/>
            <FFF>
               <HHH/>
               <GGG>
                  <JJJ>
                     <QQQ/>
                  </JJJ>
                  <JJJ/>
               </GGG>
               <HHH/>
            </FFF>
         </DDD>
      </XXX>
      <CCC>
         <DDD/>
      </CCC>
   </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//GGG/descendant::*

```
   <AAA>
      <BBB>
         <CCC/>
         <ZZZ/>
      </BBB>
      <XXX>
         <DDD>
            <EEE/>
            <FFF>
               <HHH/>
               <GGG>
                  <JJJ>
```

```
                    <QQQ/>
                </JJJ>
                <JJJ/>
            </GGG>
            <HHH/>
        </FFF>
    </DDD>
</XXX>
<CCC>
    <DDD/>
</CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//GGG/following::*

```
<AAA>
    <BBB>
        <CCC/>
        <ZZZ/>
    </BBB>
    <XXX>
        <DDD>
            <EEE/>
            <FFF>
                <HHH/>
                <GGG>
                    <JJJ>
                        <QQQ/>
                    </JJJ>
                    <JJJ/>
                </GGG>
                <HHH/>
            </FFF>
        </DDD>
    </XXX>
    <CCC>
        <DDD/>
    </CCC>
```

```
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//GGG/preceding::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <FFF>
            <HHH/>
            <GGG>
               <JJJ>
                  <QQQ/>
               </JJJ>
               <JJJ/>
            </GGG>
            <HHH/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

注意：
/child::AAA 与/AAA/self::*一样！

//GGG/self::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <FFF>
            <HHH/>
            <GGG>
               <JJJ>
                  <QQQ/>
               </JJJ>
               <JJJ/>
            </GGG>
            <HHH/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

```
//GGG/ancestor::* | //GGG/descendant::* |
//GGG/following::* | //GGG/preceding::* |
                    //GGG/self::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
```

```
            <FFF>
               <HHH/>
               <GGG>
                  <JJJ>
                     <QQQ/>
                  </JJJ>
                  <JJJ/>
               </GGG>
               <HHH/>
            </FFF>
         </DDD>
      </XXX>
      <CCC>
         <DDD/>
      </CCC>
   </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

>> 实例 22 << | 前页 | 后页

div 运算符做浮点除法运算, mod 运算符做求余运算, floor 函数返回不大于参数的最大整数(趋近于正无穷), ceiling 返回不小于参数的最小整数(趋近于负无穷)

//BBB[position() mod 2 = 0]

选择偶数位置的 BBB 元素

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <CCC/>
```

```
    <CCC/>
    <CCC/>
  </AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//BBB[position() = floor(last() div 2 + 0.5) or position() = ceiling(last() div 2 + 0.5)]

选择中间的 BBB 元素

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <CCC/>
   <CCC/>
   <CCC/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)

//CCC[position() = floor(last() div 2 + 0.5) or position() = ceiling(last() div 2 + 0.5)]

选择中间的 CCC 元素

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <CCC/>
```

```
    <CCC/>
    <CCC/>
</AAA>
```

在 XLab 中打开实例 | 树视图 (JPG)