|  |
| --- |
|  |

**3. XPath**

<http://blog.naver.com/PostView.nhn?blogId=cardin9&logNo=20032909370>

// String xPathUrl="/beans/bean[name='factory\_a']/@class";

// String xPathUrl="/beans/bean";

// PathExpression expr = xpath.compile("//book[author='Neal Stephenson']/title/text()");

// String xPathUrl="/beans/bean[constructor='a']" 빈안에 컨스트럭터 엘리먼트가 a인거.;

// String xPathUrl="/beans/bean[@name='Float']/text()";

**3.1 XPath란 무엇인가 ?**

 XPath is XML 문서의 부분을 정의하기 위한 W3C의 표준 문법 규약으로  XML 엘러먼트를 정의하기 위하여 path(경로)를 사용한다. XPath 는 표준함수의 라이브러리를 제공하며 XSLT의 주요 부분이나 XML로 쓰여지지 않았다. 즉, **XPath는 XML 문서의 부분을 정의하기 위한 문법 규약(syntax rule) 의 집합으로써**XPath에 대한 지식이 없이는 XSLT 문서를 생성할  수 없다.  XPath는 XML 문서의 노드를 정의하기 위하여 경로식(path expression)을 사용하며 이러한 경로들은 우리가 파일 시스템에서 사용하였던 표현과 아주 유사하다.

|  |
| --- |
| temp/xpath/sample.jsp |

**XPath 예**

아래의 XML 문서를 보도록 하자.

<?xml version="1.0" encoding="EUC-KR"?>

<catalog>

<cd>

<title>**San DoKaebi**</title>

<artist>**기둥SeulKiDoong**</artist>

<country>**Korea**</country>

<company>**Dongeui**</company>

<price>**8000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Say Yes**</title>

<artist>**Chake&Aska**</artist>

<country>**Japan**</country>

<company>**Sony 레코드**</company>

<price>**12000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Hound dog**</title>

<artist>**Elvis Presley**</artist>

<country>**USA**</country>

<company>**RCA**</company>

<price>**9000**</price>

<year>**1972**</year>

</cd>

<cd>

<title>**Still got the blues**</title>

<artist>**Gary Moore**</artist>

<country>**UK**</country>

<company>**Virgin records**</company>

<price>**10.20**</price>

<year>**1990**</year>

</cd>

<cd>

<title>**Romanza**</title>

<artist>**Andrea Bocelli**</artist>

<country>**EU**</country>

<company>**Polydor**</company>

<price>**15000**</price>

<year>**1996**</year>

</cd>

<cd>

<title>**When a man loves a woman**</title>

<artist>**Percy Sledge**</artist>

<country>**USA**</country>

<company>**Atlantic**</company>

<price>**8000**</price>

<year>**1987**</year>

</cd>

<cd>

<title>**Pavarotti Gala Concert**</title>

<artist>**Luciano Pavarotti**</artist>

<country>**UK**</country>

<company>**DECCA**</company>

<price>**13000**</price>

<year>**1991**</year>

</cd>

 <cd>

<title>**Unchain my heart**</title>

<artist>**Joe Cocker**</artist>

<country>**USA**</country>

<company>**EMI**</company>

<price>**10000**</price>

<year>**1987**</year>

</cd>

</catalog>

아래의 XPath 식(expression)은 루트 엘러먼트인  catalog를 선정한다.

/catalog

아래의 XPath 식은 catalog 엘러먼트 밑의 cd 엘러먼트를 선정한다.

/catalog/cd

아래의 XPath 식은 catalog 엘러먼트 밑의 cd 엘러먼트의 모든 price 엘러먼트를 선정한다.

/catalog/cd/price

**Note: 경로가 /로 시작한다면 엘러먼트에 대한 절대 경로를 의미한다.**

**3.2 XPath 의 구문**

 XPath 는 문자열(string), 수(number), 불리언 표현을 취급하기 위한 표준함수의 라이브러리를 정의한다.  예를 들면 아래의 XPath 식은 price 엘러먼트의 값으로 10000원 이상의 값을 가지는 모든 cd 엘러먼트를 선정한다.

/catalog/cd[price>10000]

XPath 는 XSLT 표준의 주요 부분으로 XSLT를 공부하려면 앞에서 설명한 바있다. 서두에도 이야기 하였듯이 XPath 는 1999년 10월 16일 XML 문서의 어드레싱 부분을 취급하는 언어로써 W3C 권고안으로 탄생하였으며 XSLT, XPointer 및 다른 파싱 소프트웨어에 의해 사용된다.

XPath 는 경로식(path expression)을 사용하여 XML 문서내의 노드를 찾는데 사용된다.

아래와 같은 XML 예제 문서를 보도록 하자.

<?xml version="1.0" encoding="EUC-KR"?>

<catalog>

<cd>

<title>**San DoKaebi**</title>

<artist>**기둥SeulKiDoong**</artist>

<country>**Korea**</country>

<company>**Dongeui**</company>

<price>**8000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Say Yes**</title>

<artist>**Chake&Aska**</artist>

<country>**Japan**</country>

<company>**Sony 레코드**</company>

<price>**12000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Hound dog**</title>

<artist>**Elvis Presley**</artist>

<country>**USA**</country>

<company>**RCA**</company>

<price>**9000**</price>

<year>**1972**</year>

</cd>

<cd>

<title>**Still got the blues**</title>

<artist>**Gary Moore**</artist>

<country>**UK**</country>

<company>**Virgin records**</company>

<price>**10.20**</price>

<year>**1990**</year>

</cd>

<cd>

<title>**Romanza**</title>

<artist>**Andrea Bocelli**</artist>

<country>**EU**</country>

<company>**Polydor**</company>

<price>**15000**</price>

<year>**1996**</year>

</cd>

<cd>

<title>**When a man loves a woman**</title>

<artist>**Percy Sledge**</artist>

<country>**USA**</country>

<company>**Atlantic**</company>

<price>**8000**</price>

<year>**1987**</year>

</cd>

<cd>

<title>**Pavarotti Gala Concert**</title>

<artist>**Luciano Pavarotti**</artist>

<country>**UK**</country>

<company>**DECCA**</company>

<price>**13000**</price>

<year>**1991**</year>

</cd>

 <cd>

<title>**Unchain my heart**</title>

<artist>**Joe Cocker**</artist>

<country>**USA**</country>

<company>**EMI**</company>

<price>**10000**</price>

<year>**1987**</year>

</cd>

</catalog>

**1) 노드 찾기**

 XML 문서는 우리가 컴퓨터의 폴더 구조에서 보는 것처럼 트리 구조로 표현된다.  XPath는 XML 문서 속에서 노드를 찾기 위해 패턴식(pattern expression)을 사용하며 XPath의 패턴은  /로 분리된 자식 엘러먼트의 이름들의 리스트로써 XML 문서에서의 경로를 표현하며  패턴은 경로에 매칭되는 엘러먼트를 선택한다.

아래의 XPath식은 catalog 엘러먼트의 모든 cd 엘러먼트의 price 엘러먼트를 선택한다.

/catalog/cd/price

**Note: /로 시작하는 경로는 엘러먼트에 대한 절대 경로를 나타낸다.**

경로가  // 로 시작하면 경로에 관계없이 문서 속의 모든 엘러먼트에 대하여 주어진 기준(criteria)을 만족하는 엘러먼트를 선택한다.

아래의 XPath 경로는 문서 숙의 모든 cd 엘러먼트를 선택한다.

//cd

**알려지지 않은 엘러먼트 선택하기**

와일드 카드(\*) 는 알려지지 않은 엘러먼트를 선택하기 위해 사용된다.  아래의 XPath 식은 catalog 엘러먼트의 cd 엘러먼트의 모든 자식 노드를 선택하기 위해 사용된다.

/catalog/cd/\*

다음의 XPath 식은 catalog 엘러먼트의 손자 엘러먼트인 모든 price 엘러먼트를 선택한다.

/catalog/\*/price

아래의 XPath 식은 2 레벨의 조상 엘러먼트를 가진 모든 price 엘러먼트를 선택한다.

/\*/\*/price

아래의 XPath 식은 경로에 관계없이 문서 속의 엘러먼트를 선택한다.

//\*

**2) 보다 자세하게 엘러먼트 선정하기**

XPath 속의 대괄호[]를 이용하여 선택하고자 하는 엘러먼트를 보다 세밀하게 표현할 수 있다.

아래의 XPath 표현은 catalog 엘러먼트의 첫 번째 cd 자식을 선택한다.

/catalog/cd[1]

아래의 XPath 표현은 catalog 엘러먼트의 마지막 cd 자식을 선택한다(그러나 first()라는 함수는 존재하지 않는다).

/catalog/cd[last()]

아래의 XPath 표현은 price 엘러먼트를 가진 catalog 엘러먼트의 모든 cd 엘러먼트를 선택한다.

/catalog/cd[price]

아래의 XPath 표현은 price 엘러먼트의 값으로 9000을 가진 catalog 엘러먼트의 모든 cd 엘러먼트를 선택한다.

/catalog/cd[price=9000]

아래의 XPath 표현은 price 엘러먼트의 값으로 9000을 가진 catalog 엘러먼트의 cd 엘러먼트의 모든 price 엘러먼트를 선택한다.

/catalog/cd[price=90000]/price

**3) 여러 path를 선택하기**

 | 연산자를 사용하며 여러 경로를 선택할 수 있다. 다음의 XPath 식은 catalog 엘러먼트의 cd 엘러먼트의 모든 title과 artist 엘러먼트를 선택한다.

/catalog/cd/title | /catalog/cd/artist

다음의 XPath 식은 문서 속에 있는 모든 title과 artist 엘러먼트를 선택한다.

//title | //artist

다음의 XPath 식은 문서 속에 있는 모든 title, artist, price 엘러먼트를 선택한다.

//title | //artist | //price

다음의 XPath 식은 문서 속에 있는 모든 artist 엘러먼트와 catalog 엘러먼트의 cd 엘러먼트의 모든 title 엘러먼트를 선택한다.

/catalog/cd/title | //artist

**3.3 속성 선정하기**

XPath에서는 모든 속성이 @로 명시되며  아래의 예는 country라는 이름을 가진 속성을 선택한다.

//@country

아래의 XPath 식은 country라는 속성을 가진 모든 cd 엘러먼트를 선택한다.

//cd[@country]

아래의 XPath 식은 어떠한 속성이라도 속성이 존재하는 cd 엘러먼트를 선택한다.

//cd[@\*]

아래의 XPath 식은 속성 country의 값으로 UK를 가지는 모든 cd 엘러먼트를 선택한다.

//cd[@country='UK']

**3.4 XPath의  위치 경로**

**1) 위치 경로 식(Location Path Expression)**

위치 경로는 절대 또는 상대 경로 둘 중 하나이다. 절대 경로는 /로 시작하며 상대 위치 경로는 그렇지 않다.  그러나 두 방식 다 /로 구분된 하나 또는 그 이상의 위치 단계로 구성되어 있다.

|  |
| --- |
| An absolute location path:  /step/step/...  A relative location path:  step/step/... |

위치 단계는 왼쪽에서 오른쪽으로 한번에 한 단계씩 체크한다. 즉, 각 단계에서 현재의 노드셋(node-set)에 속해 있는 각 노드를 대상으로 하여 체크한다. 만약 위치 경로가 절대 경로이면 현재의 노드셋은 루트 노드로 구성된다.  만약 위치 경로가 상대 경로이면 현재의 노드셋은 식에 사용된 노드로 구성된다.

위치 단계는 다음과 같은 요소로 구성된다.

축 : 위치 단계에 의해 선정된 노드와 현재 노드와의 트리 관계를 나타냄

노드 테스트 : 위치 단계에 의해 선정된 노드의 확장명과 노드 타입을 나타냄

0개 이상의 서술부 : 위치 단계에 의해 선정된 노드의 집합을 보다 상술하는 식

위치 단계의 문법은 다음과 같다.

axisname::nodetest[predicate]

예:

child::price[price=9.90]

**2) 축과 노드 테스트(Axes and Node Tests)**

축은 현재 노드와 관련 있는 노드 셋을 정의하고 노드 테스트는 축 안에 있는 노드를 정의 하며 노드에 대한 테스트를 이름 또는 유형으로 할 수 있다.

|  |  |
| --- | --- |
| **축 이름** | **Description** |
| ancestor | 현재 노드의 모든 조상 노드를 포함  **Note: 이 축은 현재 노드가 루트인 경우를 제외하고는 항상 루트를 포함하고 있다.** |
| ancestor-or-self | 현재 노드와  현재 노드의 모든 조상 노드를 포함 |
| attribute | 현재 노드의 모든 속성을 포함 |
| child | 현재 노드의 모든 자식 노드를 포함 |
| descendant | 현재 노드의 모든 후손 노드를 포함  **Note: 이 축은 이름 공간이나 속성은 포함하지 않는다.** |
| descendant-or-self | 현재 노드 및 현재노드의 후손 노드를 포함 |
| following | 현재 노드의 태그를 종료한 뒤에 있는 문서 속에 있는 모든 것을 포함 |
| following-sibling | 현재 노드의 모든 형제 노드를 포함  **Note:** 현재의 노드가 속성 노드 또는 이름 공간 노드라면 이 축은 비어있음 |
| namespace | 현재 노드의 모든 이름 공간 노드를 포함 |
| parent | 현재 노드의 부모 노드를 포함 |
| preceding | 문서 속에서 현재 노드의 시작 태그 앞에 존재하는 모든 것을 포함 |
| preceding-sibling | 현재 노드 앞의 모든 형제 노드를 포함  **Note:** 현재 노드가 속성 노드 또는 이름공간 노드일 경우 이 축은 비어있음 |
| self | 현재 노드를 포함 |

**예**

|  |  |
| --- | --- |
| **Example** | **Result** |
| child::cd | 현재노드의 자식 노드인 모든 cd 엘러먼트를 선택한다.(만약 현재 노드가 자식이 없으면 빈 노드셋을 선택한다) |
| attribute::src | 현재 노드의 src 속성을 선택한다.(만약 현재 노드가 src 속성을 가지고 있지 않다면 빈 노드셋을 선택한다) |
| child::\* | 현재 노드의 모든 자식 엘러먼트를 선택한다 |
| attribute::\* | 현재 노드의 모든 속성을 선택한다 |
| child::text() | 현재 노드의 택스트 노드 자식을 선택한다 |
| child::node() | 현재 노드의 모든 자식을 선택한다 |
| descendant::cd | 현재 노드의 모든 cd 엘러먼트 자손을 선택한다. |
| ancestor::cd | 현재 노드의 모든 cd 조상들을 선택한다. |
| ancestor-or-self::cd | 현재 노드의 모든 cd 조상들을 선택한다. 만약 현재 노드가 cd 엘러먼트이면 현재 노드도 포함한다 |
| child::\*/child::price | 현재 노드의 모든 price 손자 노드를 선택한다 |
| / | 문서의 루트를 선택한다 |

**3) 서술 부(Predicates)**

서술부는 [] 속에 놓여지며 노드셋을 필터링하여 새로운 노드 셋을 만들어낸다.

|  |  |
| --- | --- |
| **예** | **결과** |
| child::price[price=9000] | 현재 노드의 모든 price 자식 엘러먼트중 값이 9000인 price 엘러먼트를 선택 |
| child::cd[position()=1] | 현재 노드의 첫 cd 엘러먼트를 선택 |
| child::cd[position()=last()] | 현재 노드의 마지막 cd 엘러먼트를 선택 |
| child::cd[position()=last()-1] | 현재 노드의 cd 자식 노드 중 마지막에서 두 번째 cd 자식 엘러먼트를 선택 |
| child::cd[position()<6] | 현재 노드의 첫 다섯 cd 자식 노드를 선택 |
| /descendant::cd[position()=7] | 문서 속의 7번째 cd 엘러먼트를 선택 |
| child::cd[attribute::type="classic"] | 현재 노드의 cd 엘러먼트 중 type 속성으로 classic이라는 값을 가지는 엘러먼트 선택 |

**4) 위치 경로의 단축된 문법**

경로 위치를 표현하기 위해 압축된 문법을 사용하며 가장 중요한 압축된 표현은 child:: 로써 위치 단계에서 생략될 수 있다.

|  |  |  |
| --- | --- | --- |
| **약칭** | **의미** | **예제** |
| 없음 | child:: | cd 는 child::cd의 줄인 표현 |
| @ | attribute:: | cd[@type="classic"]은 child::cd[attribute::type="classic"]의 줄인 표현 |
| . | self::node() | .//cd 는 self::node()/descendant-or-self::node()/child::cd 의 줄인 표현 |
| .. | parent::node() | ../cd 는 parent::node()/child::cd 의 줄인 표현 |
| // | /descendant-or-self::node()/ | //cd 는 /descendant-or-self::node()/child::cd 의 줄인 표현 |

|  |  |
| --- | --- |
| **예** | **결과** |
| cd | 현재 노드의 모든 자식 cd 엘러먼트를 선택 |
| \* | 현재 노드의 모든 자식 엘러먼트를 선택 |
| text() | 현재 노드의 모든 텍스트 노드 자식을 선택 |
| @src | 현재 노드의 모든 src 속성을 선택 |
| @\* | 현재 노드의 모든 속성을 선택 |
| cd[1] | 현재 노드의 첫 번째 자식을 선택 |
| cd[last()] | 현재 노드의 마지막 자식을 선택 |
| \*/cd | 현재 노드의 모든 손자 노드를 선택 |
| /book/chapter[3]/para[1] | book의 chapter[3]의 첫 para를 선택 |
| //cd | 문서의 루트의 모든 cd 자손들을 선택하고 나서 해당 문서를 현재 노드로 하여 모든 cd 엘러먼트를 선택 |
| . | 현재 노드 선택 |
| .//cd | 현재 노드의 cd 엘러먼트 자손들을 선택 |
| .. | 현재 노드의 부모를 선택 |
| ../@src | 현재 노드의 부모의 src 속성을 선택 |
| cd[@type="classic"] | 현재 노드의 모든 cd 자식 노드중 type 속성으로 classic을 갖는 자식을 선택 |
| cd[@type="classic"][5] | 현재 노드의 cd 자식 노드중 type 속성으로 classic을 갖는 자식중 5번째 자식을 선택 |
| cd[5][@type="classic"] | 현재 노드의 5번째 cd 자식 노드가 ype 속성으로 classic을 갖는 다면 선택 |
| cd[@type and @country] | 현재 노드의 모든 cd 자식 중 속성으로 type과 country를 가지는 자식을 선택 |

**3.5 XPath 식**

**XPath 는 수치, 등식, 관계 및 불리언 연산을 지원한다.**

**1) 수치 표현식(Numerical Expressions)**

수치 표현식은 수치에 대하여 산술연산을 시행하기 위하여 사용된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **연산자** | **설명** | **예제** | **결과** |
| + | 더하기 | 6 + 4 | 10 |
| - | 빼기 | 6 - 4 | 2 |
| \* | 곱하기 | 6 \* 4 | 24 |
| div | 나누기 | 8 div 4 | 2 |
| mod | Modulus (나누기의 나머지) | 5 mod 2 | 1 |

**Note:**XPath 는 산술 연산을 시행하기 전에 각 피연산자를 수치로 바꾼 다음 계산한다.

**2) 등호 연산(Equality Expressions)**

등호 연산은 두 값의 동일 여부를 테스트하기 위해 사용한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **연산자** | **설명** | **예제** | **결과** |
| = | 같음 | price=9.80 | 참 (만약 price is 9000 이면) |
| != | 같지 않음 | price!=9.80 | 거짓 |

**3) 노드 셋에 대한 테스트**

만약 test value 가 노드 셋에 대한 등호 연산을 시행하였다면 만약 해당 노드 셋에 test value를 만족하는 값이 하나라도 있다면 결과는 참이 된다.

만약 test value가 노드 셋에 대하여 부등호 연산을 시행하였다면 만약 해당 노드셋에 test value를 만족하지 않는 것이 하나라도 있다면 결과는 거짓이 된다.

**결론은 노드 셋이 동시에 참이 될 수 도 있고 거짓이 될 수 도 있다는 것이다.**

**4) 관계식(Relational Expressions)**

관계식은 두 값을 비교하기 위해 사용된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **연산자** | **설명** | **예제** | **결과** |
| < | 작음 | price<8000 | 거짓(만약 price가 9000이면) |
| <= | 작거나 같음 | price<=9000 | 참 |
| > | 큼 | price>9000 | 거짓 |
| >= | 크거나 같음 | price>=9000 | 참 |

**Note:**XPath 는 연산을 시행하기 전에 각 피연산자를 수치로 바꾼 다음 계산한다.

**5) 불리언식(Boolean Expressions)**

불리언 식은 두 값을 비교하기 위하여 사용된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **연산자** | **설명** | **예제** | **결과** |
| or | or | price=9000 or price=8000 | 참 (만약 price is 8000이면) |
| and | and | price<=9000 and price>=8000 | 거짓e |

**6) XPath 함수**

XPath 는 데이터 변환을 위한 함수 라이브러리를 포함하고 있다.

**XPath 함수 라이브러리**

XPath 함수 라이브러리는 데이터를 변환, 번역하기 위한 중요 함수를 포함하고 있다.

**노드 셋 함수(Node Set Functions)**

|  |  |
| --- | --- |
| **이름** | **설명** |
| count() | 선택된 엘러먼트의 수를 반환 |
| id() | Selects 엘러먼트들을 그들 고유의 ID로 선택 |
| last() | 식 평가 컨텍스트(expression evaluation context)로부터 컨텍스트의 크기에 해당하는 수를 반환 |
| local-name() | 문서 순서에서 첫 번째인 인수 노드셋(argument node-set)의 노드의 확장명의 local part를 반환 |
| name() | 엘러먼트의 이름을 반환 |
| namespace-uri() | 문서 순서에서 첫 번째인 인수 노드셋(argument node-set)의 노드의 확장명의 이름 공간 URI를 반환 |
| position() | 식 평가 컨텍스트(expression evaluation context)로부터 컨텍스트의 포지션에 해당하는 수를 반환 |

**문자열 함수(String Functions)**

|  |  |  |
| --- | --- | --- |
| **이름** | **설명** | **예제 및 결과** |
| concat() | Returns 인수의 연접을 반환 | concat('The',' ','XML')  결과: 'The XML' |
| contains() | 첫 번째 문자열이 두 번째 문자열을 포함하고 있으면 참이며 그 이외에는 거짓 | contains('XML','X')  결과: true |
| normalize-space() | 앞뒤의 공백을 제거 | normalize-space(' The XML')  결과: 'The XML' |
| starts-with() | 첫 번째 문자열이 두 번째 문자열로 시작하면 참이며 그 이외에는 거짓 | starts-with('XML','X')  결과: true |
| string() | 객체를 문자열로 변환 | string(3.14)  결과: '3,14' |
| string-length() | 문자열 속의 문자의 개수를 반환 | string-length('Beatles')  결과: 7 |
| substring() | 부분 문자열을 반환 | substring('Beatles',1,4)  결과: 'Beat' |
| substring-after() | 부분 문자열 뒤의 부분 문자열을 반환 | substring-after('12/10','/')  결과: '10' |
| substring-before() | 부분 문자열 앞의 부분 문자열을 반환 | substring-before('12/10','/')  결과: '12' |
| translate() | 문자를 문자열로 변환 | translate('12:30',':','!')  결과: '12!30' |

**수치 함수(Number Functions)**

|  |  |  |
| --- | --- | --- |
| **이름** | **설명** | **예제 및 결과** |
| ceiling() | 인수보다 크지 않은 작은 정수 반환 | ceiling(3.14)  결과: 4 |
| floor() | 인수보다 크기 않은 가장 큰 정수 반환 | floor(3.14)  결과: 3 |
| number() | 인수를 수로 변환 | number(price) |
| round() | 인수에 가장 근접한 정수를 반환 | round(3.14)  결과: 3 |
| sum() | 인수 노드 셋의 각 노드에 대한 노드의 문자열 값을 수로 변환한 결과의 합을 반환 | sum(/cd/price) |

**불리언 함수(Boolean Functions)**

|  |  |  |
| --- | --- | --- |
| **이름** | **설명** | **예제 및 결과** |
| boolean() | 인수를 불리언으로 변환 |  |
| false() | 거짓 반환 | number(false())  결과: 0 |
| lang() |  |  |
| not() | 인수가 참이면 거짓 반환, 나머지의 경우는 거짓 | not(false()) |
| true() | 참 반환 | number(true())  결과 : 1 |

**3.5 XPath 예제**

**지금 까지 배운 내용을 앞의 예제를 사용하여 설명한다.**

**collection.xml**

<?xml version="1.0" encoding="EUC-KR"?>

<catalog>

<cd>

<title>**San DoKaebi**</title>

<artist>**기둥SeulKiDoong**</artist>

<country>**Korea**</country>

<company>**Dongeui**</company>

<price>**8000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Say Yes**</title>

<artist>**Chake&Aska**</artist>

<country>**Japan**</country>

<company>**Sony 레코드**</company>

<price>**12000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Hound dog**</title>

<artist>**Elvis Presley**</artist>

<country>**USA**</country>

<company>**RCA**</company>

<price>**9000**</price>

<year>**1972**</year>

</cd>

<cd>

<title>**Still got the blues**</title>

<artist>**Gary Moore**</artist>

<country>**UK**</country>

<company>**Virgin records**</company>

<price>**10.20**</price>

<year>**1990**</year>

</cd>

<cd>

<title>**Romanza**</title>

<artist>**Andrea Bocelli**</artist>

<country>**EU**</country>

<company>**Polydor**</company>

<price>**15000**</price>

<year>**1996**</year>

</cd>

<cd>

<title>**When a man loves a woman**</title>

<artist>**Percy Sledge**</artist>

<country>**USA**</country>

<company>**Atlantic**</company>

<price>**8000**</price>

<year>**1987**</year>

</cd>

<cd>

<title>**Pavarotti Gala Concert**</title>

<artist>**Luciano Pavarotti**</artist>

<country>**UK**</country>

<company>**DECCA**</company>

<price>**13000**</price>

<year>**1991**</year>

</cd>

 <cd>

<title>**Unchain my heart**</title>

<artist>**Joe Cocker**</artist>

<country>**USA**</country>

<company>**EMI**</company>

<price>**10000**</price>

<year>**1987**</year>

</cd>

</catalog>

**노드 선정**

지금부터는 XML 문서로부터 어떻게 노드들을 추출하는지를 인터넷 익스플로러에서 selectNode 함수를 사용하여 보여준다.  이 함수는 경로식을 인자로써 사용할 것이다.

xmlobject.selectNodes(XPath expression)

**cd 노드 선택**

아래의 예제는 catalog의 모든 cd를 선택하는 예제이다.

xmlDoc.selectNodes("/catalog/cd")

코드 :

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

<cd>

<title>**San DoKaebi**</title>

<artist>**기둥SeulKiDoong**</artist>

<country>**Korea**</country>

<company>**Dongeui**</company>

<price>**8000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Say Yes**</title>

<artist>**Chake&Aska**</artist>

<country>**Japan**</country>

<company>**Sony 레코드**</company>

<price>**12000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Hound dog**</title>

<artist>**Elvis Presley**</artist>

<country>**USA**</country>

<company>**RCA**</company>

<price>**9000**</price>

<year>**1972**</year>

</cd>

<cd>

<title>**Still got the blues**</title>

<artist>**Gary Moore**</artist>

<country>**UK**</country>

<company>**Virgin records**</company>

<price>**12000**</price>

<year>**1990**</year>

</cd>

<cd>

<title>**Romanza**</title>

<artist>**Andrea Bocelli**</artist>

<country>**EU**</country>

<company>**Polydor**</company>

<price>**15000**</price>

<year>**1996**</year>

</cd>

<cd>

<title>**When a man loves a woman**</title>

<artist>**Percy Sledge**</artist>

<country>**USA**</country>

<company>**Atlantic**</company>

<price>**8000**</price>

<year>**1987**</year>

</cd>

<cd>

<title>**Pavarotti Gala Concert**</title>

<artist>**Luciano Pavarotti**</artist>

<country>**UK**</country>

<company>**DECCA**</company>

<price>**13000**</price>

<year>**1991**</year>

</cd>

 <cd>

<title>**Unchain my heart**</title>

<artist>**Joe Cocker**</artist>

<country>**USA**</country>

<company>**EMI**</company>

<price>**10000**</price>

<year>**1987**</year>

</cd>

**첫 번째 노드 선택하기**

아래의 예제는 cd catalog의 첫 번째 cd 노드만을 선택한다.

mlDoc.selectNodes("/catalog/cd[0]")

코드 :

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd[0]"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

<cd>

<title>**San DoKaebi**</title>

<artist>**기둥SeulKiDoong**</artist>

<country>**Korea**</country>

<company>**Dongeui**</company>

<price>**8000**</price>

<year>**1992**</year>

</cd>

**Note:** IE 5 에서는 [0]

**price 노드 선택하기**

아래의 코드는 cd catalog 로부터 모든 price 노드를 선택하는 예제이다.

xmlDoc.selectNodes("/catalog/cd/price")

코드 :

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd/price"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

<price> 8000 </price>

<price> 12000 </price>

<price> 9000 </price>

<price> 12000 </price>

<price> 15000 </price>

<price> 8000 </price>

<price> 13000 </price>

<price> 10000 </price>

**price 텍스트 노드 선택하기**

어래의 코드는 price 노드로부터 오직 텍스트만을 선택한다.

xmlDoc.selectNodes("/catalog/cd/price/text()")

코드 :

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd/price/text()"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

8000

12000

9000

12000

15000

8000

13000

10000

**price가 1200 이상인 cd 노드 선택**

아래의 예제는 price가 12000이상인 모든 cd 노드를 선택한다.

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd[price>10.80]"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

<cd>

<title>**Say Yes**</title>

<artist>**Chake&Aska**</artist>

<country>**Japan**</country>

<company>**Sony 레코드**</company>

<price>**12000**</price>

<year>**1992**</year>

</cd>

<cd>

<title>**Still got the blues**</title>

<artist>**Gary Moore**</artist>

<country>**UK**</country>

<company>**Virgin records**</company>

<price>**12000**</price>

<year>**1990**</year>

</cd>

<cd>

<title>**Romanza**</title>

<artist>**Andrea Bocelli**</artist>

<country>**EU**</country>

<company>**Polydor**</company>

<price>**15000**</price>

<year>**1996**</year>

</cd>

<cd>

<title>**Pavarotti Gala Concert**</title>

<artist>**Luciano Pavarotti**</artist>

<country>**UK**</country>

<company>**DECCA**</company>

<price>**13000**</price>

<year>**1991**</year>

</cd>

**price가 1200 이상인 price 노드 선택**

아레의 예제는 price가 12000이상인 모든 price 노드를 선택한다.

xmlDoc.selectNodes("/catalog/cd[price>10.80]/price")

코드 :

<html>

<body>

<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("cdcatalog.xml")

path="/catalog/cd[price>10.80]/price"

set nodes=xmlDoc.selectNodes(path)

for each x in nodes

  document.write("<xmp>")

  document.write(x.xml)

  document.write("</xmp>")

next

</script>

</body>

</html>

결과 :

<price>**12000**</price>

<price>**12000**</price>

<price>**15000**</price>

<price>**13000**</price>

**[출처]** [XPath 문법](http://blog.naver.com/cardin9/20032909370)|**작성자** [개구리](http://blog.naver.com/cardin9)