

```
chris$ # first let's look at XPath in action
```

```
chris$ ls -l
```

```
total 112
-rw-r--r--  1 chris  staff  50092 Nov 25 10:24 nobel.sql
-rw-r--r--  1 chris  staff    999 Nov 25 10:16 quotes.xml
```

```
chris$ xalan ApplyXPath quotes.xml "/quotelist"
```

```
java ApplyXPath quotes.xml /quotelist
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using /quotelist
<output>
<quotelist>

  <quotation id="q1" style="wise">
    <text>Expect nothing; be ready for everything.</text>
    <source>Samurai chant</source>
  </quotation>

  <quotation id="q2" style="political">
    <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
    <source>Lyndon B. Johnson</source>
  </quotation>

  <quotation id="q3" style="silly">
    <?human laugh?>
    <text>What if the hokey-pokey IS what it's all about?</text>
  </quotation>

  <quotation id="q4" style="wise">
    <text>If they give you ruled paper, write the other way.</text>
    <source>Juan Ramon Jiminez</source>
  </quotation>

  <!-- the checkbook is mightier than the sword? -->
  <quotation id="q5" style="political">
    <text>Banking establishments are more dangerous than standing
    armies.</text>
    <source>Thomas Jefferson</source>
  </quotation>

</quotelist>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml "/quotelist/quotation/source"
```

```
java ApplyXPath quotes.xml /quotelist/quotation/source
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using /quotelist/quotation/source
<output>
<source>Samurai chant</source>
<source>Lyndon B. Johnson</source>
<source>Juan Ramon Jiminez</source>
<source>Thomas Jefferson</source>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml "/*/*/source"
```

```
java ApplyXPath quotes.xml /*/*/source
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using /*/*/source
<output>
<source>Samurai chant</source>
<source>Lyndon B. Johnson</source>
<source>Juan Ramon Jiminez</source>
<source>Thomas Jefferson</source>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml "//source"
```

```
java ApplyXPath quotes.xml //source
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //source
<output>
<source>Samurai chant</source>
<source>Lyndon B. Johnson</source>
<source>Juan Ramon Jiminez</source>
<source>Thomas Jefferson</source>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml "source"
```

```
java ApplyXPath quotes.xml source
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using source
<output>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml "//source"
```

```
java ApplyXPath quotes.xml //source
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //source
<output>
<source>Samurai chant</source>
<source>Lyndon B. Johnson</source>
<source>Juan Ramon Jiminez</source>
<source>Thomas Jefferson</source>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]'
```

```
java ApplyXPath quotes.xml '//quotation[@id="q2"]
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]
<output>
<quotation id="q2" style="political">
  <text>If one morning I walked on top of the water across the Potomac
  River, the headline that afternoon would read "President Can't
  Swim".</text>
  <source>Lyndon B. Johnson</source>
</quotation>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]/child::node()'
```

```
java ApplyXPath quotes.xml //quotation[@id="q2"]/child::node()
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]/child::node()
<output>
```

```
<text>If one morning I walked on top of the water across the Potomac
  River, the headline that afternoon would read "President Can't
  Swim".</text>
```

```
<source>Lyndon B. Johnson</source>
```

```
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]/self::node()'
```

```
java ApplyXPath quotes.xml //quotation[@id="q2"]/self::node()
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]/self::node()
<output>
```

```
<quotation id="q2" style="political">
  <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
  <source>Lyndon B. Johnson</source>
</quotation>
```

```
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]/self::*'
```

```
java ApplyXPath quotes.xml //quotation[@id="q2"]/self::*
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]/self::*
<output>
```

```
<quotation id="q2" style="political">
  <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
  <source>Lyndon B. Johnson</source>
</quotation>
```

```
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]/parent::*'
```

```
java ApplyXPath quotes.xml //quotation[@id="q2"]/parent::*
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]/parent::*
<output>
```

```
<quotelist>
```

```
<quotation id="q1" style="wise">
  <text>Expect nothing; be ready for everything.</text>
  <source>Samurai chant</source>
```

```

</quotation>

<quotation id="q2" style="political">
  <text>If one morning I walked on top of the water across the Potomac
  River, the headline that afternoon would read "President Can't
  Swim".</text>
  <source>Lyndon B. Johnson</source>
</quotation>

<quotation id="q3" style="silly">
  <?human laugh?>
  <text>What if the hokey-pokey IS what it's all about?</text>
</quotation>

<quotation id="q4" style="wise">
  <text>If they give you ruled paper, write the other way.</text>
  <source>Juan Ramon Jiminez</source>
</quotation>

<!-- the checkbook is mightier than the sword? -->
<quotation id="q5" style="political">
  <text>Banking establishments are more dangerous than standing
  armies.</text>
  <source>Thomas Jefferson</source>
</quotation>

</quotelist>
</output>

```

```

chris$ xalan ApplyXPath quotes.xml '//quotation[@id="q2"]/parent::quotation'

```

```

java ApplyXPath quotes.xml //quotation[@id="q2"]/parent::quotation
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id="q2"]/parent::quotation
<output>
</output>

```

```

chris$ xalan ApplyXPath quotes.xml "//quotation[@id='q5']/self::node()/.."

```

```

java ApplyXPath quotes.xml //quotation[@id='q5']/self::node()/..
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //quotation[@id='q5']/self::node()/..
<output>
<quotelist>

```

```

  <quotation id="q1" style="wise">
    <text>Expect nothing; be ready for everything.</text>
    <source>Samurai chant</source>
  </quotation>

```

```

  <quotation id="q2" style="political">
    <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
    <source>Lyndon B. Johnson</source>
  </quotation>

```

```

  <quotation id="q3" style="silly">
    <?human laugh?>

```

```
<text>What if the hokey-pokey IS what it's all about?</text>
</quotation>
```

```
<quotation id="q4" style="wise">
  <text>If they give you ruled paper, write the other way.</text>
  <source>Juan Ramon Jiminez</source>
</quotation>
```

```
<!-- the checkbook is mightier than the sword? -->
<quotation id="q5" style="political">
  <text>Banking establishments are more dangerous than standing
  armies.</text>
  <source>Thomas Jefferson</source>
</quotation>
```

```
</quotelist>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//source[text()="Thomas_Jefferson"]'
```

```
java ApplyXPath quotes.xml //source[text()="Thomas_Jefferson"]
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //source[text()="Thomas_Jefferson"]
<output>
<source>Thomas_Jefferson</source>
</output>
```

```
chris$ xalan ApplyXPath quotes.xml '//source[contains(text(),"J")]'
```

```
java ApplyXPath quotes.xml //source[contains(text(),"J")]
Loading classes, parsing quotes.xml, and setting up serializer
Querying DOM using //source[contains(text(),"J")]
<output>
<source>Lyndon B. Johnson</source>
<source>Juan Ramon Jiminez</source>
<source>Thomas_Jefferson</source>
</output>
```

```
chris$ # now let's look at a new example and compare sql to xpath
```

```
chris$ ls -l
```

```
total 112
-rw-r--r--  1 chris  staff  50092 Nov 25 10:24 nobel.sql
-rw-r--r--  1 chris  staff    999 Nov 25 11:16 quotes.xml
```

```
chris$ psql
```

```
psql (8.4.1)
Type "help" for help.
```

```
chris=# \i ./nobel.sql
```

```
SET
psql:./nobel.sql:7: NOTICE:  table "nobel" does not exist, skipping
DROP TABLE
CREATE TABLE
BEGIN
INSERT 0 1
[skipped...]
INSERT 0 1
COMMIT
```

```
chris=# \d nobel
```

```
Table "public.nobel"
Column |          Type          | Modifiers
-----+-----+-----
 yr    | integer                |
 subject | character varying    |
 winner | character varying    |
```

```
chris=# select * from nobel;
```

```
 yr | subject | winner
-----+-----+-----
1901 | Chemistry | Jacobus H. van 't Hoff
1902 | Chemistry | Emil Fischer
1903 | Chemistry | Svante Arrhenius
1904 | Chemistry | Sir William Ramsay
1905 | Chemistry | Adolf von Baeyer
1906 | Chemistry | Henri Moissan
[skipped...]
```

```
chris=# select table_to_xml('nobel', false, false, '');
```

```
table_to_xml
-----
<nobel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <yr>1901</yr>
    <subject>Chemistry</subject>
    <winner>Jacobus H. van 't Hoff</winner>
  </row>
  <row>
```

```
<yr>1902</yr>
<subject>Chemistry</subject>
<winner>Emil Fischer</winner>
</row>
```

[skipped...]

```
chris=# create table xml (doc xml);
```

CREATE TABLE

```
chris=# select '<a></a>'::xml;
```

```
xml
-----
<a></a>
(1 row)
```

```
chris=# select '<a><a>'::xml;
```

```
ERROR:  invalid XML content
LINE 1: select '<a><a>'::xml;
           ^
```

```
DETAIL:  Entity: line 1: parser error : Premature end of data in tag a line 1
<a><a>
      ^
```

```
Entity: line 1: parser error : Premature end of data in tag a line 1
<a><a>
      ^
```

```
chris=# insert into xml select table_to_xml('nobel', false, false, '');
```

INSERT 0 1

```
chris=# select count(*) from nobel;
```

```
count
-----
  783
(1 row)
```

```
chris=# select * from nobel where subject = 'Chemistry';
```

```
yr | subject | winner
-----+-----+-----
1901 | Chemistry | Jacobus H. van 't Hoff
1902 | Chemistry | Emil Fischer
1903 | Chemistry | Svante Arrhenius
1904 | Chemistry | Sir William Ramsay
1905 | Chemistry | Adolf von Baeyer
1906 | Chemistry | Henri Moissan
1907 | Chemistry | Eduard Buchner
1908 | Chemistry | Ernest Rutherford
[skipped...]
```

```
chris=# select xpath('//subject[text()='Chemistry'],'', doc) from xml;
```

```
{<subject>Chemistry</subject>,<subject>Chemistry</subject>,<subject>Chemistry</sub  
ject>,<skipped...> <subject>Chemistry</subject>}  
(1 row)
```

```
chris=# select xpath('//subject[text()='Chemistry']/self::node()/../winner', doc)  
from xml;
```

```
{"<winner>Jacobus H. van 't Hoff</winner>","<winner>Emil  
Fischer</winner>","<winner>Svante Arrhenius</  
winner>","<winner>Sir William Ramsay</winner>","<winner>Adolf von  
Baeyer</winner>","<winner>Henri Moiss  
an</winner>","<winner>Eduard Buchner</winner>","<winner>Ernest  
Rutherford</winner>","<winner>Wilhelm Os  
twald</winner>","<winner>Otto Wallach</winner>","<winner>Marie  
Curie</winner>","<winner>Victor Grignard  
</winner>","<winner>Paul Sabatier</winner>","<winner>Alfred Werner</winner>","  
<skipped...> "<winner>Richard R. Schrock</winner>"}  
(1 row)
```