SMVS
Page No.
Date : / /

XPath and XQuery

* XPath:

XPath is a Query Language used to navigate through element and attributes in an XML document.

It Provides a way to Locate specific elements or attributes

A string of the Syntax used to define a path to a pertular node in an XML document.

A Sequences of steps that navigate through the ~~hier~~ hierarchy of an XML Document.

=> XPath Expression:

XPath Expression includes Following XPath Component.

1 Path: Used to Give starting node.

SMVS
Page No.
Date : / /

(ii) Absolute Path : Starts from the root node in XML Document.

Syntax : /root/element

(iii) Relative Path : Starts from the current node

Syntax : element/subelement

2 Axes : An Axes define the tree relationship between nodes selected by current node.

(i) child : Selects children of the current node.

(ii) parent : Selects the parent of the current node

Syntax : parent::node()

(iii) descendant : Selects all descendant (child, grandchild etc) of current node.

Syntax : 'descendant::node()'

(iv) ancestor : Selects all ancestor (parent, grandpare.) of current node.

(v) Following-Sibiling : Select all sibiling after the current node.

(vi) precending-sibiling : Select all sibiling before the current node.

(vii) self : Selects the current node

(viii) attribute : Selects attributes of current node

(ix) desendant-or-self : Current node + Desendant

(x) ancestor-or-self : Current node + Ancestor

3 Node Tests : Node Tests specify which nodes to select based on their type, name etc.

(a) Element Node : Selects nodes with a specific name

(b) Wildcard : Selects nodes of any name

(c) Attribute Node : Selects Attributes nodes with a specific name.

(d) Node Type : Select Node Based on their Type.

(i) node() : Select any node
(ii) text() : Select text node
(iii) comment() : Select Comment node

(e) Namespace : Select node based on their namespace URL.

4 Predicates : Predicates in XPath are used to filter nodes based on specific condition or criteria.

(a) Filtering Nodes : Filter nodes based on their position, value or other condition.

(b) Position Based Filtering : Selects node based on their position in sequence.

(c) Value-Based Filtering : Selects Node

based on their attribute or child element value.

(f) Logical Condition: Multiple condition can be combined using 'and' and 'or', 'not' operators.

(g) String Function: Function like 'contains()', 'starts-with()' and 'substring()' can be used.

(h) Numeric Function: Function like 'number()', 'Floor()' can be used.

=> Evaluation of Path Expression:

Evaluating XPath expression gives ~~nag~~ navigation path in XML Document tree and select the node.

1 Context Node:

The Evaluation starts at the Context Node in which it can

SMVS
Page No.
Date : / /

Root node or Relative Node.

2 Evaluating the Axis:

The Axis is defines relationships between the nodes.

3 Applying Node Test:

The node test checks iff the node matches the specific criteria.

4 Filtering with Predicates:

Predicates are applied to filter nodes further.

Ex. /bookstore/book[price>30]/title

starts from Root element '/' 'book-store.

Selects 'book' element with 'price' child element is > 30.

From those 'book' elements, selects the 'title' child elements.

SMVS
Page No.
Date : / /

=> Abberviations: XPath provides serveral abberviation to make expressions more concise.

(a) Child Axis Abbervation: The child axis is the default axis.

(b) Attribute Axis: It can be abbreviated with the '@' symbol.

(c) Current Node: It can be abbreviated with the a single dot '.'.

(d) Parent Node: It can be abbreviated with the two dots '..'.

(e) Decendant Axis: It can be abbreviated with the '//' symbol.

Ex. /bookstore/book/@category:

SMVS
Page No.
Date : / /

* Define Features of XPath 2.0 :

=> XPath 2.0 is improvements version of the XPath 1.0 which provide better support for XML Schema.

-> Features :

1 Data Types : Provides wide range of the data Types.

Ex. String, number, date, time, dataTime etc.

2 Expression Language Enhancements :

XPath 2.0 supports more complex path expression and allow to use predicates more effectively.

3 Function and Operators :

XPath 2.0 includes varioues new Function and operators.

Function : String, Number, Date and Time, Sequence, Regular Expression

SMVS
Page No.
Date : / /

4 Sequences: Sequences can include nodes, atomic value or mix of both.

5 Axis and Node Tests:

Enhanced node tests include testing for specific types or values.

6 Improved Boolean Logic:

XPath 2.0 introduces more robust boolean logic and supports complex boolean expression.

7 Schema-Aware XPath:

This allows for querying based on type information and validating nodes against XML schema types.

SMVS
Page No.
Date : / /

* XQuery Feature :

=> XQuery offers several advanced features that enhance capability to query and manipulate XML data effectively.

-> Features :

1 XPath Expression :

XPath Expression are used to locate elements, attributes and value based on their criteria.

2 FLWOR Expression :

FLWOR expression allow complex querying and transformations.

3 XQuery Functions :

XQuery Supports user-defined function, which allow you to encapsulate reusable logic.

4 Types and Schema Validation :

XQuery Supports type checking and

Validation against XML Schema definations.

5 Path Expression and Predicated:

Path Expression and Predicates are used to traverse and Filter XML document.

6 Join Operations:

XQuery supports various types of Join Operation to combine data from multiple source.

7 Grouping and Aggregation:

XQuery allows grouping and Aggregation of data to perform operation like counting, summing etc.

8 Data Types:

XQuery Support various data types including string, dates, time etc.

SMVS
Page No.
Date : / /

* FLWOR expression in XQuery :

=> FLWOR Expression are powerful Features in XQuery.

Using FLWOR Expression, We can Perform this Five Expression,

For, Let, Where, Order by, Return

1 For Clause :

The 'For' clause is used to iterate over the sequence of item.

In 'For' clause, each item in the sequence is bound to the specified variable.

Syntax : For $variable in $sequence

Multiple 'For' clauses can be used to handle nested sequence.

'For' clauses can be combine with all the FLWOR expression.

Example:

```
for $note in doc("note.xml")
//book
return $note/title
```

Gives Title of each note.

2 'let' clause:

Used to Binds a value to variable, which can letter used in XQuery.

The 'let' clause creates a variable and bind it to the result expression.

Syntax: let $variable := $expression

Enables use of values or expression r without recalculating them.

Variable defined with 'let' are available within FLWOR expression.

Ex:

```
let $tp := sum(doc("notes.xml")
        //book/price)
return $tp
```

Gives, Total Prices of all Notes.

SMVS
Page No.
Date : / /

3 'where' Clause :

Used to filters the sequence of items based on a specified condition.

The 'where' clause filter the item when that meet the specific condition.

Syntax : where $condition.

The condition must evaluate to a boolean value.

Example :

```
For $note in doc("note.xml")
  //note
  where $note/price >20
    return $note/title.
```

4 'order by' clause :

Specifies the sorting order of the sequence based on one or more expression

User can specify the ascending or descending order.

Syntax:
order by $expression [ascending descending]

The default order is ascending.

Ex.
```
For $note in doc("notes.xml")
//not
order by $note/title
return $note/title.
```

5 'return' clause:

Specifies the output of the query and constructing final result.

Syntax:
return $expression

The Result can be XML element, a text node or any other element.

Ex.
```
For $note in doc("note.xml")//note
return <book>
        <title>($book/title)
        </title>
</book>
```