Java XPath Parser - Overview

The XPath is an official recommendation of the World Wide Web Consortium (W3C). It defines a language to find information in an XML file. It is used to traverse elements and attributes of an XML document. XPath provides various type of expressions which can be used to enquire relevant information from the XML document.

## What is XPath?

* **Structure Definations** - XPath defines the parts of an XML document like element, attribute, text, namespace, processing-instruction, comment, and document nodes
* **Path Expressions** XPath provides powerful path expressions select nodes or list of nodes in XML documents.
* **Standard Functions**XPath provides a rich library of standard functions for manipulation of string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values etc.
* **Major part of XSLT**XPath is one of the major element in XSLT standard and is must have knowledge in order to work with XSLT documents.
* **W3C recommendation**XPath is official recommendation of World Wide Web Consortium (W3C).

**Here is the input text file we need to parse:**

<?xml version="1.0"?>

<class>

<student rollno="393">

<firstname>dinkar</firstname>

<lastname>kad</lastname>

<nickname>dinkar</nickname>

<marks>85</marks>

</student>

<student rollno="493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>vinni</nickname>

<marks>95</marks>

</student>

<student rollno="593">

<firstname>jasvir</firstname>

<lastname>singn</lastname>

<nickname>jazz</nickname>

<marks>90</marks>

</student>

</class>

## XPath Expressions

XPath uses a path expression to select node or list of nodes from an xml document. Following is the list of useful paths and expression to select any node/ list of nodes from an xml document.

|  |  |
| --- | --- |
| **Expression** | **Description** |
| node-name | Select all nodes with the given name "nodename" |
| / | Selection starts from the root node |
| // | Selection starts from the current node that match the selection |
| . | Selects the current node |
| .. | Selects the parent of the current node |
| @ | Selects attributes |
| student | Example: Selects all nodes with the name "student" |
| class/student | Example: Selects all student elements that are children of class |
| //student | Selects all student elements no matter where they are in the document |

## Predicates

Predicate are used to find specifi node or a node containing specific value and are defined using [...] .

|  |  |
| --- | --- |
| **Expression** | **Result** |
| /class/student[1] | Selects the first student element that is the child of the class element. |
| /class/student[last()] | Selects the last student element that is the child of the class element. |
| /class/student[last()-1] | Selects the last but one student element that is the child of the class element. |
| //student[@rollno='493'] | Selects all the student elements that have an attribute named rollno with a value of '493' |

# Java XPath Parser - Parse XML Document

## Steps to Using XPath

Following are the steps used while parsing a document using XPath Parser.

* Import XML-related packages.
* Create a DocumentBuilder
* Create a Document from a file or stream
* Create an Xpath object and an XPath path expression
* Compile the XPath expression using XPath.compile() and get a list of nodes by evaluating the compiled expression via XPath.evaluate()
* Iterate over the list of nodes.
* Examine attributes
* Examine sub-elements

**Import XML-related packages**

import org.w3c.dom.\*;

import org.xml.sax.\*;

import javax.xml.parsers.\*;

import javax.xml.xpath.\*;

import java.io.\*;

**Create a DocumentBuilder**

DocumentBuilderFactory factory =

DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

**Create a Document from a file or stream**

StringBuilder xmlStringBuilder = new StringBuilder();

xmlStringBuilder.append("<?xml version="1.0"?> <class> </class>");

ByteArrayInputStream input = new ByteArrayInputStream(

xmlStringBuilder.toString().getBytes("UTF-8"));

Document doc = builder.parse(input);

**Build XPath**

XPath xPath = XPathFactory.newInstance().newXPath();

**Prepare Path expression and evaluate it**

String expression = "/class/student";

NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(doc, XPathConstants.NODESET);

**Iterate over NodeList**

for (int i = 0; i < nodeList.getLength(); i++) {

Node nNode = nodeList.item(i);

...

}

**Examine attributes**

//returns specific attribute

getAttribute("attributeName");

//returns a Map (table) of names/values

getAttributes();

**Examine sub-elements**

//returns a list of subelements of specified name

getElementsByTagName("subelementName");

//returns a list of all child nodes

getChildNodes();

## Demo Example

**Here is the input text file we need to parse:**

<?xml version="1.0"?>

<class>

<student rollno="393">

<firstname>dinkar</firstname>

<lastname>kad</lastname>

<nickname>dinkar</nickname>

<marks>85</marks>

</student>

<student rollno="493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>vinni</nickname>

<marks>95</marks>

</student>

<student rollno="593">

<firstname>jasvir</firstname>

<lastname>singh</lastname>

<nickname>jazz</nickname>

<marks>90</marks>

</student>

</class>

**Demo Example:**

*XPathParserDemo.java*

package com.tutorialspoint.xml;

import java.io.File;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.xpath.XPath;

import javax.xml.xpath.XPathConstants;

import javax.xml.xpath.XPathExpressionException;

import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;

import org.w3c.dom.NodeList;

import org.w3c.dom.Node;

import org.w3c.dom.Element;

import org.xml.sax.SAXException;

public class XPathParserDemo {

public static void main(String[] args) {

try {

File inputFile = new File("input.txt");

DocumentBuilderFactory dbFactory

= DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder;

dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(inputFile);

doc.getDocumentElement().normalize();

XPath xPath = XPathFactory.newInstance().newXPath();

String expression = "/class/student";

NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(doc, XPathConstants.NODESET);

for (int i = 0; i < nodeList.getLength(); i++) {

Node nNode = nodeList.item(i);

System.out.println("\nCurrent Element :"

+ nNode.getNodeName());

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

System.out.println("Student roll no : "

+ eElement.getAttribute("rollno"));

System.out.println("First Name : "

+ eElement

.getElementsByTagName("firstname")

.item(0)

.getTextContent());

System.out.println("Last Name : "

+ eElement

.getElementsByTagName("lastname")

.item(0)

.getTextContent());

System.out.println("Nick Name : "

+ eElement

.getElementsByTagName("nickname")

.item(0)

.getTextContent());

System.out.println("Marks : "

+ eElement

.getElementsByTagName("marks")

.item(0)

.getTextContent());

}

}

} catch (ParserConfigurationException e) {

e.printStackTrace();

} catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (XPathExpressionException e) {

e.printStackTrace();

}

}

}

This would produce the following result:

Current Element :student

Student roll no : 393

First Name : dinkar

Last Name : kad

Nick Name : dinkar

Marks : 85

Current Element :student

Student roll no : 493

First Name : Vaneet

Last Name : Gupta

Nick Name : vinni

Marks : 95

Current Element :student

Student roll no : 593

First Name : jasvir

Last Name : singh

Nick Name : jazz

Marks : 90

# Java XPath Parser - Query XML Document

## Demo Example

**Here is the input text file we need to query:**

<?xml version="1.0"?>

<class>

<student rollno="393">

<firstname>dinkar</firstname>

<lastname>kad</lastname>

<nickname>dinkar</nickname>

<marks>85</marks>

</student>

<student rollno="493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>vinni</nickname>

<marks>95</marks>

</student>

<student rollno="593">

<firstname>jasvir</firstname>

<lastname>singn</lastname>

<nickname>jazz</nickname>

<marks>90</marks>

</student>

</class>

**Demo Example:**

*XPathParserDemo.java*

package com.tutorialspoint.xml;

import java.io.File;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.xpath.XPath;

import javax.xml.xpath.XPathConstants;

import javax.xml.xpath.XPathExpressionException;

import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;

import org.w3c.dom.NodeList;

import org.w3c.dom.Node;

import org.w3c.dom.Element;

import org.xml.sax.SAXException;

public class XPathParserDemo {

public static void main(String[] args) {

try {

File inputFile = new File("input.txt");

DocumentBuilderFactory dbFactory

= DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder;

dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(inputFile);

doc.getDocumentElement().normalize();

XPath xPath = XPathFactory.newInstance().newXPath();

String expression = "/class/student[@rollno='493']";

NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(doc, XPathConstants.NODESET);

for (int i = 0; i < nodeList.getLength(); i++) {

Node nNode = nodeList.item(i);

System.out.println("\nCurrent Element :"

+ nNode.getNodeName());

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

System.out.println("Student roll no : "

+ eElement.getAttribute("rollno"));

System.out.println("First Name : "

+ eElement

.getElementsByTagName("firstname")

.item(0)

.getTextContent());

System.out.println("Last Name : "

+ eElement

.getElementsByTagName("lastname")

.item(0)

.getTextContent());

System.out.println("Nick Name : "

+ eElement

.getElementsByTagName("nickname")

.item(0)

.getTextContent());

System.out.println("Marks : "

+ eElement

.getElementsByTagName("marks")

.item(0)

.getTextContent());

}

}

} catch (ParserConfigurationException e) {

e.printStackTrace();

} catch (SAXException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (XPathExpressionException e) {

e.printStackTrace();

}

}

}

This would produce the following result:

Current Element :student

Student roll no : 493

First Name : Vaneet

Last Name : Gupta

Nick Name : vinni

Marks : 95

# Java XPath Parser - Create XML Document

**XPath parser is used to to navigate XML Document only. It is better to use DOM parser for creating XML. Please refer the Java DOM Parser section for the same.**

# Java XPath Parser - Modify XML Document

**XPath parser is used to to navigate XML Document only. It is better to use DOM parser for modifying XML. Please refer the Java DOM Parser section for the same.**