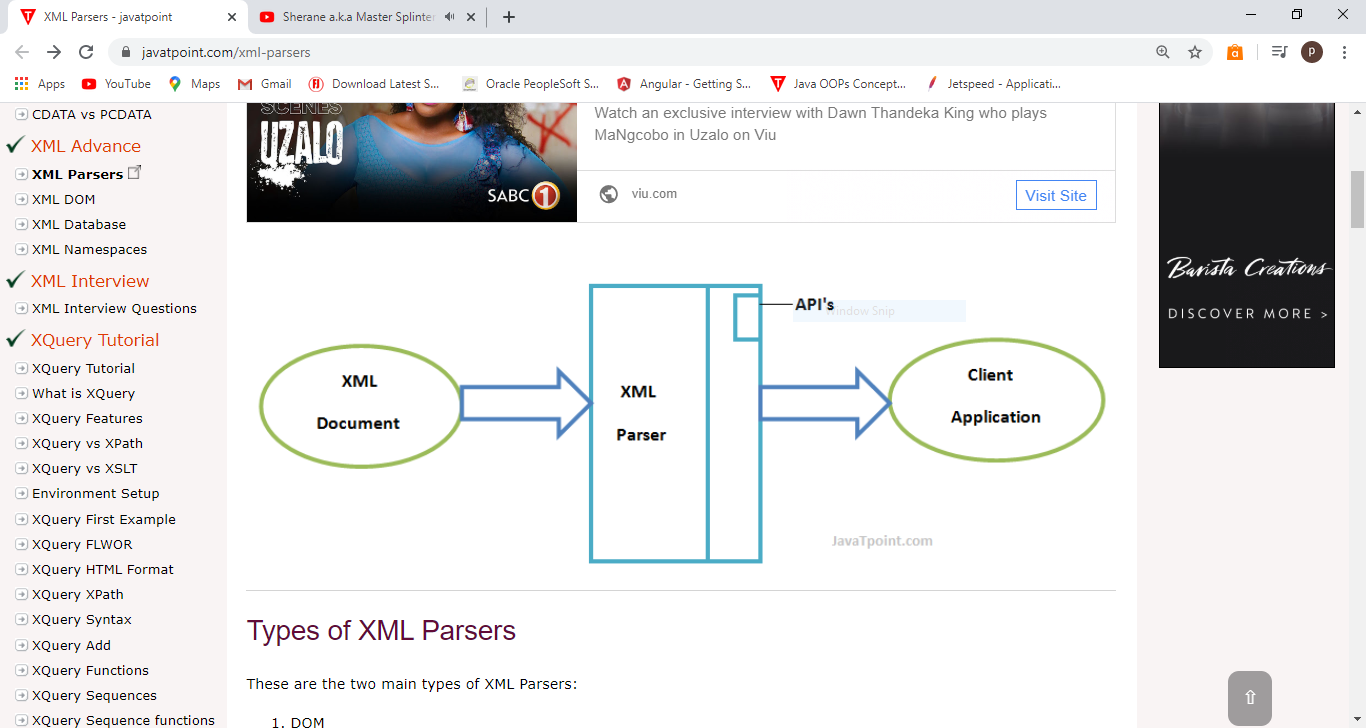
XML Parsers

This is a software library/ package that provides interfaces for client applications to work with an XML document.

Parsers are designed to read XML and create ways for programs to use XML.

Validates the document and checks that the document is well formatted.

Visual representation



Types of XML parsers include:

* DOM
* SAX

DOM/Document Object Model

This is an object which contains all the information of an XML document.

The DOM parser implements the DOM API.

SAX/Simple API for XML

This is an event based API.

Applications using SAX receive event notifications about the XML document being processed an element, attribute (basically a top-down approach).

Tokens<> are processed in the same order as they appear in the document.

When to use SAX

Processing the document in a linear fashion from top down.

Document isn’t deeply nested.

XPath

Can be used to navigate through elements and attributes in a XML document.

XPath Nodes include:

* Elements
* Attributes
* Text
* Namespace
* Processing- instructions
* Comment
* Document nodes

Selecting Nodes include

Nodename (Selects all node with the name “nodename”.)

/ (Selects from the root node)

// (Selects nodes in the document from the current node that match the selection no matter where they are.)

. (Selects the current node.)

.. (Selects the parent of the current node.)

@ (Selects the attributes.)

Predicates

These are used to find a specific node or a node that contains a specific value.

|  |  |
| --- | --- |
| Path 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 |
| /class/student[last()<3] | Selects the first two student elements that are children of the class element |
| //student[@id] | Selects all the student elements that have an attribute named id |
| //student[@id = “393”] | Selects all the student elements that have a "id" attribute with a value of "393" |
| /class/student[marks>35] | Selects all the student elements of the class element that have a mark element with a value greater than 35 |
| /bookstore/book[price>35.00]/title | Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00 |

Selecting Unknown Nodes

|  |  |
| --- | --- |
| Wildcard | Description |
| \* | Matches any element node |
| @\* | Matches any attribute node |
| Node() | Matches any node of any kind |

|  |  |
| --- | --- |
| Path Expression | Result |
| /class/\* | Selects all the child element nodes of the class element |
| //\* | Selects all elements in the document |
| //title[@\*] | Selects all title elements which have at least one attribute of any kind |

Selecting multiple paths

|  |  |
| --- | --- |
| Path Expression | Result |
| //book/title | //book/price | Selects all the title AND price elements of all book elements |
| //title | //price | Selects all the title AND price elements in the document |
| /bookstore/book/title | //price | Selects all the title elements of the book element of the bookstore element AND all the price elements in the document |

Location Path Expression

A path can be absolute or relative.

An absolute path starts with a (/) and relative location path does not.

Syntax example

An absolute location path:

/step/step/...

A relative location path:

step/step/...

Each step s evaluated against nodes in the current node-set.

A step consists of:

An axis (This is a 3 way relationship between the selected nodes and current node.)

A node test (Identifies the node within the an axis)

Zero or more predicates (To further refine the selected node-set.)

Syntax for a location step is:

axisname:: nodetest[predicate]

Examples include:

|  |  |
| --- | --- |
| **Example** | **Result** |
| child::book | Selects all book nodes that are children of the current node |
| attribute::lang | Selects the lang attribute of the current node |
| child::\* | Selects all element children of the current node |
| attribute::\* | Selects all attributes of the current node |
| child::text() | Selects all text node children of the current node |
| child::node() | Selects all children of the current node |
| descendant::book | Selects all book descendants of the current node |
| ancestor::book | Selects all book ancestors of the current node |
| ancestor-or-self::book | Selects all book ancestors of the current node - and the current as well if it is a book node |
| child::\*/child::price | Selects all price grandchildren of the current node |