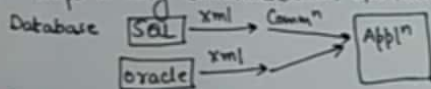


(XML)-①

Easy Engineering Classes – Free YouTube Lectures

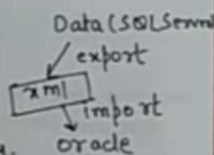
For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML :- extensible Markup Language. ^{you can create your own tags.}
→ used to Store and transport data.
→ Self descriptive.
→ used to Carry data (Not used to display data).
→ Self-defined tags.
→ Platform and Language independent.
→ Helps in easy communication b/w two platforms.



Features and Advantages:-

- i) Separates data from HTML.
- ii) Simplifies data sharing.
- iii) Simplifies data transport.
- iv) Increases data availability.
- v) Simplifies platform change.



XML Example: (college) child college ← root
↳ Hierarchical Structure. ↳ class1 ↳ class2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<college> ← Root Element ↳ declaration
  <class1>
    <Name>Amit</Name>
    <RollNo>1</RollNo>
  </class1>
  <class2>
    <Name>Mohan</Name>
    <RollNo>2</RollNo>
  </class2>
</college>
```

↳ child Element.

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML DTD: XML Document Type Definition/Declaration
 → used to describe XML language precisely.
 → used to define structure of a XML document.
 → Contains list of legal elements.
 → used to perform validation.

DTD SYNTAX: `<!DOCTYPE element DTD Identifiers
 [declaration 1
]> declaration 2`

Types of DTD

Internal

elements are declared within the XML files.

Syntax:

`<!DOCTYPE root-element
 [element-declaration]>`

External

elements are declared outside XML file.

Syntax:

`<!DOCTYPE root-element SYSTEM
 "file-name">`

Internal DTD Example

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE Address [
  <!Element Address Name,
  Company, phone>
  <!Element Name (#PCDATA)
  <!Element Company (#PCDATA)
  ]>
<Address>
  ① <Name>____ </Name>
  ② <Company>____ </Company>
  ③ <Phone>____ </Phone>
</Address>
```

External DTD Example

```
Add.dtd
<!DOCTYPE Address
SYSTEM "Add.dtd">
```

XML File

Root Element



(XML Namespaces)

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML Namespaces: used to avoid element name conflict in XML document.

- It is a set of unique names.
- Identified by URI (Uniform Resource Identifier)
- Attribute name must start with "xmlns"

Syntax: `<element xmlns:prefix="URI">`
element and Attributes Prefix names belongs to URI.

Conflict: Generally conflict occurs when we try to mix XML documents from diff. XML Application.

Eg. of conflict.

| 1.xml | 2.xml |
|-----------------------------|-----------------------------|
| <code><class></code> | <code><class></code> |
| <code><name></code> | <code><name></code> |
| <code></class></code> | <code></class></code> |

Conflict occurs due to same element name.

Example of Namespace: 1.xml

`<?xml version="1.0" encoding="UTF-8"?>`

`<class xmlns:c1="class1...">`

`<c1:name>Aman</c1:name>`

`</c1:class>`

Now there will be no conflict due to namespace.

2.xml

`<c2:class xmlns:c2="class2...">`

`<c2:name>Aman</c2:name>`

`</c2:class>`

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML-Schemas: Commonly known as XML Schema Definition (XSD). It is used to describe & validate the structure and content of XML Data.
 → It is a method of expressing constraints about XML documents.
 → It is like DTD but provides more control on XML structure.

Syntax: `<xs:schema xmlns:xs="...">`

Definition Types

Simple Type
 used only in the content of the text.
 eg: `xs:int`, `xs:string`.
`<xs:element name="Phone" type="xs:int"/>`

Complex Type
 It is the container for other element definitions. Allows you to specify which child elements an element can contain & to provide some structure within your XML documents.

eg: **Complex Type** (Add.xsd)

`<?xml version="1.0" encoding="UTF-8"?>`

`<xs:schema xmlns:xs="Schema1...">`

`<xs:element name="Address">`

`<xs:complexType>` *child elements should appear in sequence.*

`<xs:sequence>`

`<xs:element name="Name" type="xs:string"/>`

`<xs:element name="Phone" type="xs:int"/>`

`</xs:sequence>`

`</xs:complexType>`

`</xs:element>` `</xs:schema>`

(Add.xml)

`<?xml version="1.0" encoding="UTF-8"?>`

`<Address`

`xmlns:schemaLocation="Add.xsd">` *Path of XML schema def.*

`1 <Name> Aman </Name>`

`2 <Phone> 9810 </Phone>` *Error*

`</Address>` *abcd*

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

| HTML | XML | DTD | XSD |
|---|--|---|--|
| <p>i) Display Data (Look and Feel).</p> <p>ii) Markup language itself</p> <p>iii) Not Case sensitive</p> <p>iv) Predefined Tags</p> <p>v) Static</p> <p>eg:- <code><html></code> } Predefined Tag <code><body></code> <code><p> HTML INTRO</code> <code></p> display</code> <code></body></code> <code></html></code></p> | <p>i) Transport and Store the data.</p> <p>ii) Provide framework to define markup languages.</p> <p>iii) Case-Sensitive</p> <p>iv) Can Create own tags</p> <p>v) Dynamic</p> <p>eg:- <code><College></code> } Custom tags <code><class></code> <code><Name></code> } transport <code></Name></code> <code></class></code> <code></College></code></p> | <p>i) Document Type definition</p> <p>ii) doesn't support <u>data-types</u>.</p> <p>iii) doesn't support name space.</p> <p>iv) Doesn't define Order for child Elements</p> <p>v) Not Extensible</p> <p>eg:- root <code><!DOCTYPE Address[</code> <code><!Element Address (Name)</code> <code><!Element Name (@PCDATA)</code> <code>]]</code></p> | <p>i) XML Schema definition.</p> <p>ii) <u>Supports</u></p> <p>iii) Supports</p> <p>iv) Order Can be defined.</p> <p>v) Extensible → namespace.</p> <p>eg:- <code><xs:element name="Address"</code> <code><xs:complexType></code> <code><xs:sequence></code> } order <code><xs:element name="name" type="xs:string"></code> <code></xs:sequence></code> <code></xs:complexType></code> <code></xs:element></code></p> |

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML with CSS: CSS Stands for Cascading Style Sheets. It can be used to add style and display info to an XML document.

Syntax: `<?xml-stylesheet type="text/css" href="xml.css"?>`

Example: xml.css

College

```
{
  background-color: blue;
}
```

Name

```
{
  color: black;
}
```

Roll

```
{
  color: green;
}
```

<elements>

<College>

<Name>__</Name>

<roll>__</roll>

</College>

<?xml-stylesheet

type="text/css"

href="xml.css"?>

XML with XSLT: "XSL" → Extensible Stylesheet Language. "XSL" is a styling language for XML.

XSLT: XML Transformations.

↳ It is used to transform an XML document into another XML document, or another document type.

What You Can do with XSLT:

- i) Add/ Remove Elements
- ii) Add/ Remove Attributes
- iii) Rearrange/ Sort Elements

Eg. Transforming XML into XHTML using XSLT.

<College>

<class>

<Name> Aman </Name>

<RollNo> 1 </RollNo>

.....

</class>

</College>

| Name | RollNo |
|------|--------|
| Aman | 1 |

Free YouTube Lectures

PU, UPTU and Other Universities, Colleges of India

XML with XSLT: "XSL" → Extensible **St**ylesheet Language. "XSL" is a Styling language for XML.
XSLT: XML Transformations.
 ↳ It is used to transform an XML document into another XML document, or another document type.

What You Can do with XSLT:

- i) Add/Remove Elements
- ii) Add/Remove Attributes
- iii) Rearrange/Sort Elements

Eg. Transforming XML into XHTML using XSLT.

`<?xml-stylesheet type="text/xsl" href="Simple.XSL"?>`
`<College>`
`<class>`
`<Name> Aman </Name>`
`<RollNo> 1 </RollNo>`
`</class>`
`</College>`

| Name | RollNo |
|------|--------|
| Aman | 1 |

Free YouTube Lectures

J and Other Universities, Colleges of India

Simple.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="Simple XSL....">
  <xsl:template match="/">
    <html>
      <body>
        <table>
          <tr>
            <th>Name</th>
            <th>RollNo</th>
            </tr>
            <xsl:for-each select="College/class">
              <tr>
                <td><xsl:value-of select="Name"/></td>
                <td><xsl:value-of select="RollNo"/></td>
              </tr>
            </xsl:for-each>
          </table>
        </body>
      </html>
    </xsl:template>
  </xsl:stylesheet>
```

A loop parsing elements of College/class tag.

fetch Value of element

Element Name

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML-DOM: DOM is Document Object Model.

→ DOM document is a collection of nodes or pieces of info organized in a hierarchy.

→ Tree-Based.

→ Defines a standard way to access and manipulate documents.

→ Programmer can modify/delete their content and can also create new elements.

→ Elements, their content (text & Attributes) are all known as Nodes.

Example: `<h2 id="demo"> </h2>`

HTML DOM. `<button type="button" onclick="document.getElementById('demo').innerHTML='Hello'> click </button>`

`<College>`

✓ `<class>`

✓ `<title>`

✓ `<Gender>`

`College`

`class`

`title`

`Gender`

Tree like hierarchy.
node.

Example of XML DOM:-

`//College.xml`

`<?xml version="1.0" encoding="UTF-8"?>`

`<College>`

`<Branch category="IT">`

`<Students>60</Students>`

`<HOD>ABC</HOD>`

`</Branch>`

`<Branch category="ECE">`

`<Students>100</Students>`

`<HOD>XYZ</HOD>`

`</Branch>`

`</College>`

Get Value of XML element

`t = xmlDoc.getElementsByTagName("Students")`
`[0].childNodes[0].`

nodeValue;

o/p = 60

`[1].childNodes[1].`

nodeValue;

o/p = 100

`[node Value;`

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

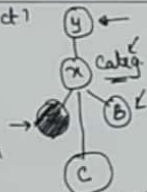
XML DOM Properties (x = node object)

- i) x.nodeName → name of x
- ii) x.nodeValue → Value of x
- iii) x.parentNode → Parent node of x
- iv) x.childNodes → Child nodes of x
- v) x.attributes → attributes of x.

replace name with Tag
name

XML DOM Methods

- i) x.getElementsByTagName(name) → get all elements with "name".
- ii) x.appendChild(node) → insert a child node to x
- iii) x.removeChild(node) → removes a child node from x.



x.removeChild(z);