XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- A Namespace is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to group. The Namespace is identified by URI(Uniform Resource Identifiers).

- eg,.
- <table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
  </table>
- information about a table (a piece of furniture):
- <table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
  </table>
- If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

- Name conflicts in XML can easily be avoided using a name prefix.
- This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Namespace Declaration

- A Namspace is declared using reserved attributes. Such an attribute name must either be xmlns or begin with xmlns: shown as below:
- <element xmlns:name="URL">

Syntax

- The Namespace starts with the keyword xmlns.
- The word name is the Namespace prefix.
- The URL is the Namespace identifier.

Example

- Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<cont:contact xmlns:cont="www.xyz.com/profile">
<cont:name>Tanmay Patil</cont:name>
<cont:company>TutorialsPoint</cont:company>
<cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is cont, and the Namespace identifier (URI) as www.xyz.com/profile. This means, the element names and attribute names with the cont prefix (including the contact element), all belong to the www.xyz.com/profile namespace.

XML - CDATA SECTIONS

CDATA means, Character Data. CDATA are defined as blocks of text that are not parsed by the parser, but are otherwise recognized as markup.

By using CDATA section, you are commanding the parser that the particular section of the document contains no markup and should be treated as regular text.

Syntax

Following is the syntax for CDATA section:

<![CDATA[characters with markup]]>

This section may contain markup characters (<, >, and &), but they are ignored by the XML processor.

Rules are required to be followed for XML CDATA:

CDATA cannot contain the string "]]>" anywhere in the XML document.

Nesting is not allowed in CDATA section.

PCDATA

- PCDATA means parsed character data.
- Think of character data as the text found between the start tag and the end tag of an XML element.
- **PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.**
- Tags inside the text will be treated as markup and entities will be expanded.
- However, parsed character data should not contain any &, <, or > characters; these need to be represented by the & < and > entities, respectively.

Processing instructions (PIs)

- Processing instructions (PIs) can be used to pass information to applications. PIs can appear anywhere in the document outside the markup. They can appear in the prolog, including the document type definition (DTD), in textual content, or after the document

Syntax

Following is the syntax of PI –

<?target instructions?> Where

- **target** – Identifies the application to which the instruction is directed.
- **instruction** – A character that describes the information for the application to process.
- A PI starts with a special tag **<?** and ends with **?>**. Processing of the contents ends immediately after the string **?>** is encountered.

- PIs are rarely used. They are mostly used to link XML document to a style sheet. Following is an example −

<?xml-stylesheet href = “a.css" type = "text/css"?>
Here, the *target* is *xml-stylesheet*. *href=“a.css"* and *type="text/css"* are *data* or *instructions* the target application will use at the time of processing the given XML document.

- In this case, a browser recognizes the target by indicating that the XML should be transformed before being shown;

Processing Instructions Rules

- A PI can contain any data except the combination **?>**, which is interpreted as the closing delimiter.

Example:

- <?display table-view?>
- <?sort alpha-ascending?>
- <?textinfo whitespace is allowed ?>