XML ?

eXtensible Markup Language

X M L

XML ?

- It is used for storing and transferring data.
- It is a markup language much like HTML.
- Unlike HTML, XML is case-sensitive, requires each tag is closed properly, and preserves whitespace.
- XML Tags need not to be predefined like HTML (therefore known as extensible).

XML ?

- A markup language is a language that annotates text so that the computer can manipulate that text.
- Most markup languages are human-readable because the annotations are written in a way to distinguish them from the text itself.
- It was designed to be self-descriptive.

XML in Android

- We create XML layouts in Android, and later alter them using Java logic.

- Resources are the additional files and static content an application needs, such as animations, color schemes, layouts, menu layouts.

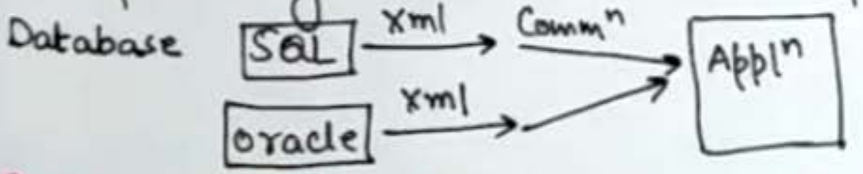
- Each layout file must contain one (and only one!) root element.

Easy Engineering Classes – Free YouTube Lectures

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

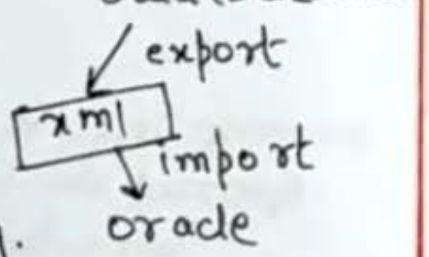
XML :- eXtensible Markup Language. (you can create your own tags.)

- → 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)

↳ Hierarchial Structure. (college ← root; child ↳ class1, class2)

```
<?xml version="1.0" encoding="ISO-8859-1"?>   ↳ declaration
<college>   ← Root Element
  <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 Identifier
  [ 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>
```

DTD

XML

Root Element

Root Element

External DTD Example

Add.dtd

XML File

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

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:name = "URL" >
name → Prefix
URL → URI

element and Attributes names belongs to URL.

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

Eg. of Conflict.

1.xml

```
<class>
  <name></name>
</class>
```

2.xml

```
<class>
  <name></name>
</class>
```

Conflict occurs due to same element name.

Example of Namespace: 1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<c1:class xmlns:c1="class1 ....">
  <c1:name> Aman </c1:name>
</c1:class>
```

UTF-8 → URI
c1 → Prefix

2.xml

```
<c2:class xmlns:c2="class2 ...">
  <c2:name> Aman </c2:name>
</c2:class>
```

Now there will be no conflict due to namespace.

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:- of Complex Type (Add.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="Schema1....">
 <xs:element name="Address">
  <xs:ComplexType>
  <xs:sequence>
   <xs:element name="1Name" type="xs:string"/>
   <xs:element name="2Phone" type="xs:int"/>
  </xs:sequence>
 </xs:Complex type>
</xs:element>
</xs:schema>
```

Child elements should appear in sequence.

(Add.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<Address
  xsi:schemaLocation="____Add.xsd">
  1<Name> Aman </Name>
  2<Phone> 9810 </Phone>
</Address>
```

Path of Xml schema def^n.

123 → Error

abcd → Error

Easy Engineering Classes – Free YouTube Lectures

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

HTML

i) Display Data (Look and feel).

ii) Markup language itself

iii) Not Case sensitive

iv) Predefined Tags

v) Static

eg:-

```
<html>
  <body>
    <p> HTML INTRO </p>
  </body>
</html>
```

Predefined Tag

display

XML

i) Transport and Store the data.

ii) Provide framework to define markup languages.

iii) Case-Sensitive

iv) Can Create own tags

v) Dynamic

eg:-

```
<College>
  <Class>
    <Name> </Name>
  </Class>
</College>
```

Custom tags

transport

DTD

i) Document Type definition

ii) doesn't support data-types.

iii) doesn't support name space.

iv) Doesn't define Order for child Elements

v) Not Extensible

eg:-

```
<!DOCTYPE Address[
<!Element Address (Name)>
<!Element Name (#PCDATA)>
]>
```

root

=

=

=

XSD

i) XML Schema definition.

ii) Supports

iii) Supports

iv) Order Can be defined.

v) Extensible → namespace.

eg:-

```
<xs: element name= "Address">
  <xs: Complex Type>
    <xs: sequence>
      <xs: elemenent name=
      "name" type= "xs:string" />
    </xs: sequence>
  </xs: Complex type>
</xs: element>
```

order

View

- It represents a rectangular area of the screen, and is responsible for **displaying information or content**, and event handling.

- Text, images, and buttons are all Views in Android.

ViewGroup

- It is essentially an **'invisible container'** that holds multiple Views or ViewGroups together, and defines their layout properties.

- Common View Groups
  - A List View displays a list of scrollable items.
  - A Grid View displays items in a two-dimensional scrollable grid.
  - A Table Layout groups views into rows and columns

Root View

- It is **Root** Element of XML Layout file.

- Common View Groups
  - A Linear Layout aligns its contents into a single direction, whether vertical or horizontal.
  - A Relative Layout displays its child content in positions relative to the parent.
  - A Frame Layout is a placeholder on a screen that display only a single view (Fragments)

XML Layout

Root View

activity_main.xml | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

Attributes

View(Child)

LinearLayout