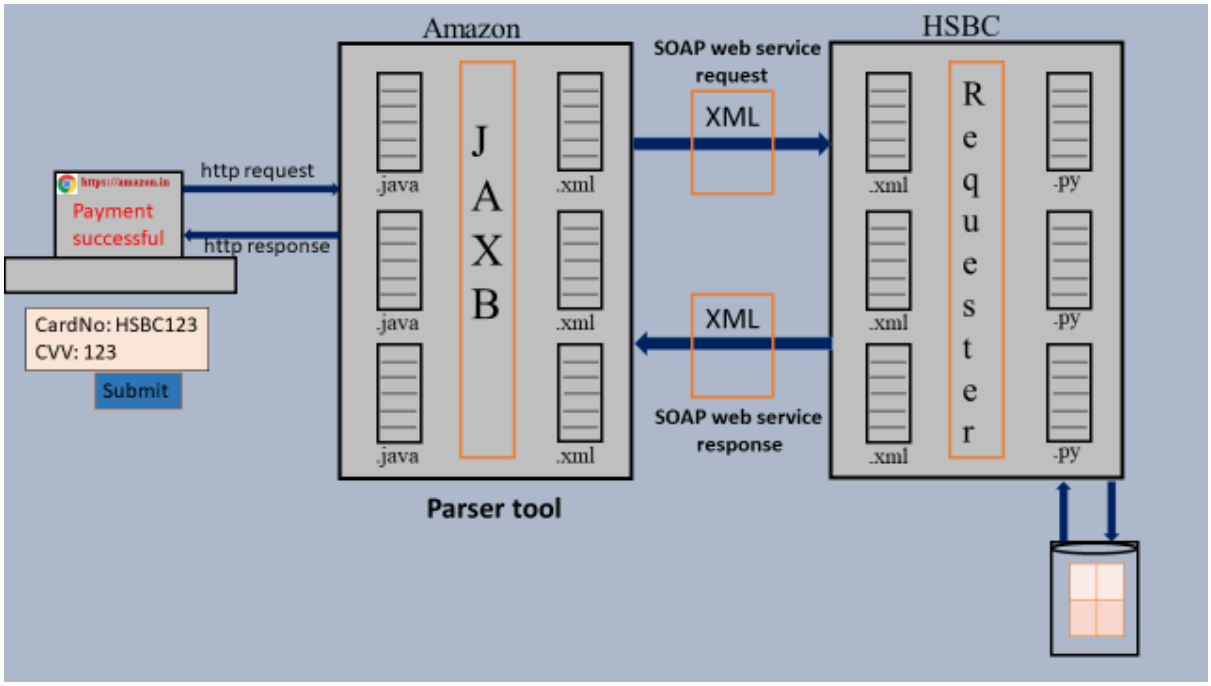
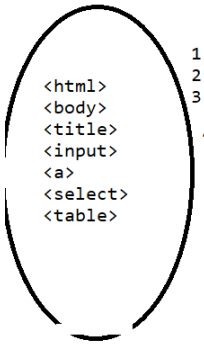
Extensible Mark-up Language (XML)

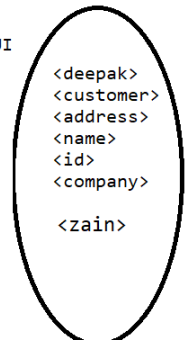


- XML is “Markup Language & Platform Independent Language" which helps to store and transport data
- Different Applications which are developed using different technologies or same technologies can transfer the data among themselves with the help of XML
- As the name implies it's an extension of HTML & hence XML looks similar to HTML but it’s not a HTML
- XML has User-defined(custom) tags.
- XML tags are also called as "elements “

HTML vs XML



1. used to devlop WebPages or GUI
2. collection of Inbuilt tages
3. Case Insensitive
4. not a strictly types lang



1. used to store & transfer the data between 2 application when runing in same platform/ different platform
2. collection of custom tag is called XML lang
3. very strictly typed lang
4. case sensitive lang
5. Platform Independent Language

XML Syntax

- XML is "Strictly Typed" Language hence case-sensitive
- They cannot contain spaces
- Every opening element should have corresponding closing element and also XML elements must be properly nested/closed
- They must start with a letter or underscore
- They cannot start with the letters like xml or XML or Xml etc.
- MIME type (Content Type) of XML is "application/xml “
- File extension of XML is ".xml"

Rule 1: XML Prolog:

Below line is called as "XML prolog", which is optional. If it exists, it must be the First Line of XML

EG :

<?xml encoding=’UTF-8’ version=1.1 schema=http://testing.xom…>

Rule 2: Xml Comments

- The syntax of XML comment is similar to that of HTML

  Ex:

  <!--This is a comment -->

Rule 3: XML Structure

- Like HTML, XML follows a Tree Structure
- An XML tree starts at a "root element" and branches from

"Root element" will have "child elements “

- XML Consists of "Only One" root element which is parent of all other element’s child elements
- child can have "sub elements / child elements

<root>

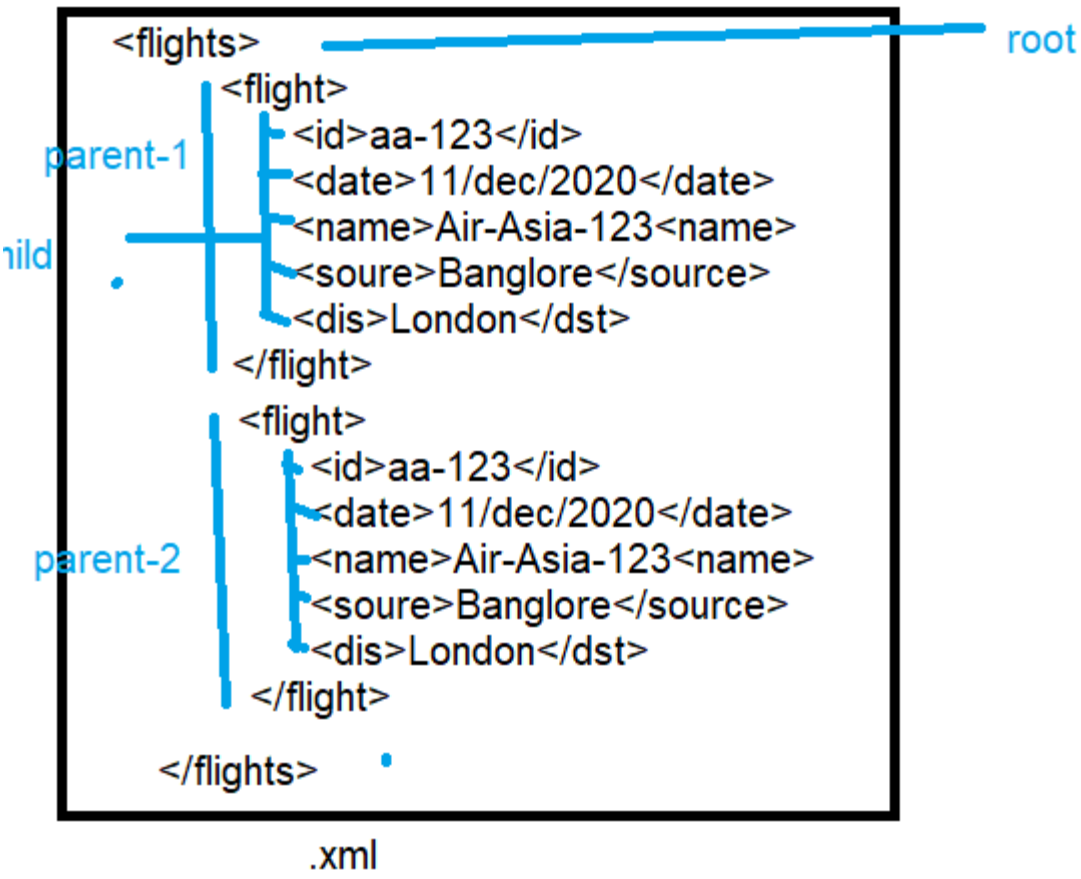
<child>

<subchild>.....</subchild>

</child>

</root>

```
<employee>
        <name>Deepak</name>
</employee>
```



Rule 4: XML-Entity References

- Some characters have a special meaning in XML.
- If you place a character like "<" inside an XML element, it will generate an error because it represents the start of a new element

Ex:<message>salary<1000</message>

- To avoid this error, we can replace the "<" character with an "entity reference" as shown below

<message>salary <1000</message>

There are 5 pre-defined entity references in XML:

< < less than

> > greater than

& & ampersand

' ‘ apostrophe

" " quotation mark

Rule 5: XML PCDATA: Parsed Character Data

- Text between start-element and end-element is called as PCDATA which will be examined by the parser.

Example: -

<employee>Ram</employee>

The string "Ram" is considered as PCDATA

PCData: will be always considered as String

Rule 6: XML-CDATA: Character Data

- If XML data contain many special characters, it is cumbersome to replace all of them. Instead, we can use "CDATA (character data) section “

Example: -

<employee>emplyeSal >1000 & sal < 10 </employee> : Wrong

<![CDATA[<emplyeSal >1000 & sal < 10 </employee>]]>: Correct

Rule 7: XML-Elements & Attributes

- XML element is everything from (including) the element's start tag to (including) the element's end tag
- An element can contain:

1. data

2. Attributes

3. other elements

4. All of the above

➔**XML-Attributes**

- Like HTML, XML elements can also have attributes, but **attributes can't easily expand like elements**
- XML Attributes Must be Quoted either single or double quotes can be use

```
EG :1                                   EG:2
     <person gender = 'male'>               <person>
         <name>deepak</name>                   <gender>male</gender>
     </person>                                 <name>deepak</name>
                                            </person>
```

➢

Example 1 gender is an attribute

Example 2 gender is an element

➔XML Elements

Will go for elements when data is to be extended

```
<?xml version=1.0 , Encoding="UTF-8 scema='http://resource.com.dtd'>
<person>
        <id>sk-01</id>
        <name>deepak</name>
        <gender>male</gender>
        <mobileNum>9886662262</mobileNum>
<person>
```

```
<person id="sk-01"  gender="male">
   <name>deepak</name>
   <mobilenum>9886662262</mobilenum>
</person>
```

or

```
<person id="sk-01"  gender="male">
   <name>
         <fname>deepak</fname>
         <lname>gowda</lname>
    </name>
   <mobilenum>9886662262</mobilenum>
</person>
```

Rule 8: XML Schema's

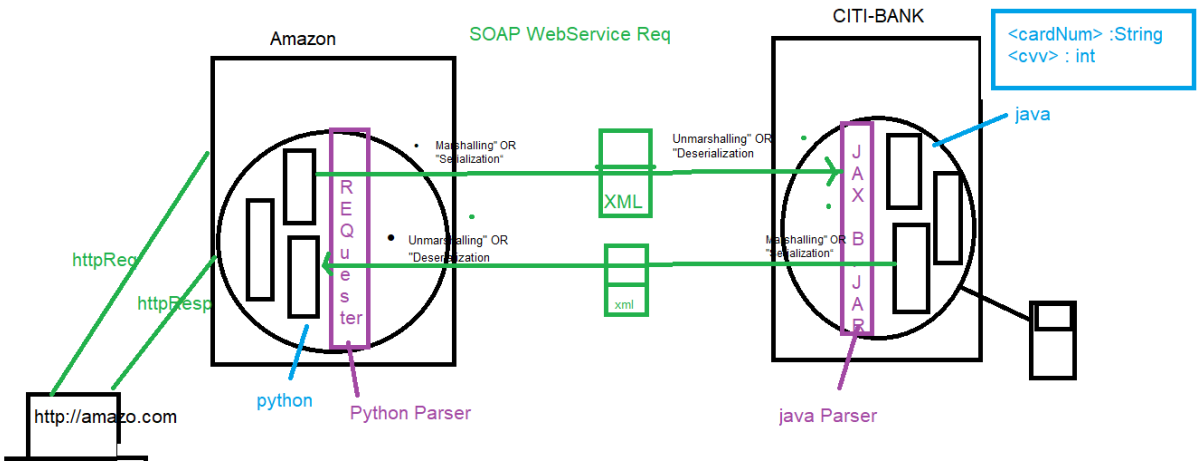
- W.K.T XML helps us to store & transfer the data.
- When sending data from one application to an another, it is essential that both applications have the same "expectations / agreement" about the content/data.
- for example, A date like "03-11-2004" -in some countries, be interpreted as 3rd November and -in other countries as 11th March.

There are two ways to define a Schema for XML

1. Document Type Definition (DTD)
2. XML Schema Definition (XSD)

XML-PARSAR(JAXB)

- JAXB is a Java API helps us to convert Java Object to XML & vice-versa
- The Process of converting Java Object to XML is called as "Marshalling" OR "Serialization “
- The Process of converting XML to Java Object is called as "Unmarshalling" OR "Deserialization”


Amazon
SOAP WebService Req
CITI-BANK
<cardNum> :String
<cvv> : int
java
Marshalling" OR
"Serialization"
Unmarshalling" OR
"Deserialization
XML
Unmarshalling" OR
"Deserialization
Marshalling" OR
"Serialization"
xml
httpReq
httpResp
http://amazo.com
python
Python Parser
java Parser