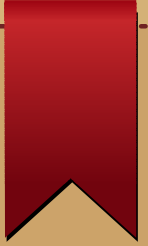


Unit 4



Xml Schema



Introduction to Schema



- DTD is for validating data but it has some limitations
 - DTD syntax is not like XML
 - Can not search information inside DTD's
 - Can not display the content as HTML
- Schema replace most of the features of DTD
- Syntax of Schema is same as XML
 - Schema document is an XML document



Example

- `<!ELEMENT Marks (#PCDATA)>`

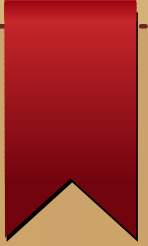
DTD doesn't define Marks can be numeric or not

`<Marks>90</Marks>`

`<Marks>English</Marks>`

- In schema we can specify our element should only contain data
- An XML doc that conforms to the rules of a schema is called as valid

What is XML Schema





- XML schema is a language which is used for expressing constraint about XML documents.
- There are so many schema languages which are used now a days for example Relax- NG and XSD (XML schema definition).
- An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.



Features

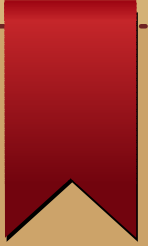
- XML Schema uses XML instance syntax to describe the rules for the grammar of an XML document
- It allows rich variety of data types for element and attributes
- It allows to specify which namespace declarations and definitions belong in, import other namespaces, wildcarding of declaration from other namespaces
- Type definitions and element & attribute declaration is separated from each other
- Allows reuse of type definitions

- 
- Allows to specify constraints on the uniqueness of values of a particular type and relationship between values and other types
- 

DTD vs XSD

- | | | |
|----|--|--|
| 1. | DTD are the declarations that define a document type for SGML. | XSD describes the elements in a XML document. |
| 2. | It doesn't support namespace. | It supports namespace. |
| 3. | It is comparatively harder than XSD. | It is relatively more simpler than DTD. |
| 4. | It doesn't support datatypes. | It supports datatypes. |
| 5. | SGML syntax is used for DTD. | XML is used for writing XSD. |
| 6. | It is not extensible in nature. | It is extensible in nature. |
| 7. | It doesn't give us much control on structure of XML document. | It gives us more control on structure of XML document. |

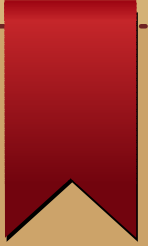
DTD vs XML Schema




1.

- DTD doesn't differentiate data types of an element
- XML differentiates data types of an element
 - Eg: `<!ELEMENT Quantity (#PCDATA)>`
 - `<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'`
`>`
`<xsd:element name='quantity' type='xsd:nonNegativeInteger' />`
`<xsd:/schema>`



- 
- `<Quantity>5</Quantity>`

this is valid xml in both declarations

- `<Quantity>-5</Quantity>`
 - `<Quantity>lots</Quantity>`
 - In XML schema this is invalid
- 

.xml

- `<?xml version="1.0"?>`

```
<Root xmlns:xsi  
="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="file.xsd">
```

Content

```
</Root>
```

.xsd

- `<?xml version="1.0"?>`
`<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">`
`<xsd:element name="Root" type="xsd:datatype"/>`
`</xsd:schema>`

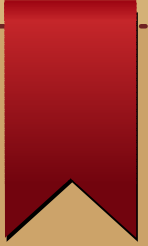
XML Schema types system



- Simple Types: it can contain only text. They can not have sub-elements or attributes. The text can be of various data types(strings, numbers, dates etc)
- Complex Types: it can contain sub-elements, attributes etc. They are made up of one or more simple element.



1. Basics of Simple and complex types



- `<?xml version="1.0"?>`
- `<student xmlns:xsi="`
`http://www.w3.org/2001/XMLSchema-instance`
`xsi:noNamespaceSchemaLocation="student.xsd">`
`<rollno> 132</rollno>`
`<name> ABC </name>`
`<marks> 74 </marks>`
`<result> distinction </result>`
`</student>`
- An XML Schema is not required to have a namespace. To specify the location for an XML Schema that does not have a target namespace, use the `noNamespaceSchemaLocation` attribute

Corrospounding schema file

- `<?xml version="1.0">`

`<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">`

`<xsd:element name="student" type="studenttype"/>`

`<xsd:complexType name="studenttype">`

`<xsd:sequence>`

`<xsd:element name="rollno" type="xsd:string"/>`

`<xsd:element name="name" type="xsd:string"/>`

`<xsd:element name="marks" type="xsd:integer"/>`

`<xsd:element name="result" type="xsd:string"/>`

`</xsd:sequence>`

`</xsd:complexType>`

`</xsd:schema>`

3. Specifying Element Content

We can specify content in element by this:

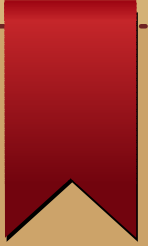
```
<xsd:element name="rollno" type="xsd:string"/>
```

```
<xsd:element name="name" type="xsd:string"/>
```

```
<xsd:element name="marks" type="xsd:integer"/>
```

```
<xsd:element name="result" type="xsd:string"/>
```

Simple Types



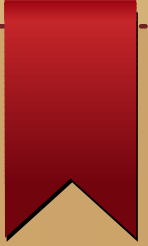
- Numeric
- Time
- XML
- String
- Boolean
- Binary
- URI reference



Numeric data types

Name	Description
byte	A signed 8-bit integer
decimal	A decimal value
int	A signed 32-bit integer
integer	An integer value
long	A signed 64-bit integer
float	32 bit, same as java's float datatype
double	64 bit, same as java's double data type
negativeInteger	An integer containing only negative values (...,-2,-1)
nonNegativeInteger	An integer containing only non-negative values (0,1,2,...)
nonPositiveInteger	An integer containing only non-positive values (...,-2,-1,0)
positiveInteger	An integer containing only positive values (1,2,...)
short	A signed 16-bit integer
unsignedLong	An unsigned 64-bit integer
unsignedInt	An unsigned 32-bit integer
unsignedShort	An unsigned 16-bit integer
unsignedByte	An unsigned 8-bit integer

DateTime Data types



Name	Description
date	Defines a date value
dateTime	Defines a date and time value
duration	Defines a time interval
gDay	Defines a part of a date - the day (DD)
gMonth	Defines a part of a date - the month (MM)
gMonthDay	Defines a part of a date - the month and day (MM-DD)
gYear	Defines a part of a date - the year (YYYY)
gYearMonth	Defines a part of a date - the year and month (YYYY-MM)
time	Defines a time value



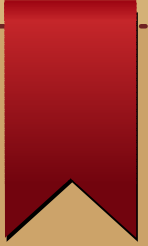
XML Data Types

Name	Description
ENTITY	An XML name, declared as an inparsed entity in a DTD
ENTITIES	List of ENTITY names
ID	A string that represents the ID attribute in XML (only used with schema attributes)
IDREF	A string that represents the IDREF attribute in XML (only used with schema attributes)
IDREFS	Reference to a list of ID names
language	A string that contains a valid language id
Name	A string that contains a valid XML name
NCName	Local names without colons
NMTOKEN	A string that represents the NMTOKEN attribute in XML (only used with schema attributes)
NMTOKENS	A list of NMTOKEN types

XML Data Types

Name	Description
ENTITY	An XML name, declared as an inparsed entity in a DTD
ENTITIES	List of ENTITY names
ID	A string that represents the ID attribute in XML (only used with schema attributes)
IDREF	A string that represents the IDREF attribute in XML (only used with schema attributes)
IDREFS	Reference to a list of ID names
language	A string that contains a valid language id
Name	A string that contains a valid XML name
NCName	Local names without colons
NMTOKEN	A string that represents the NMTOKEN attribute in XML (only used with schema attributes)
NMTOKENS	A list of NMTOKEN types
QName	Prefixed name

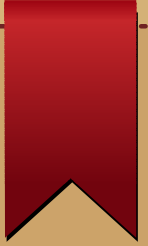
String Data Types



normalizedString	A string that does not contain line feeds, carriage returns, or tabs
string	A string
token	QName



Binary Data Types



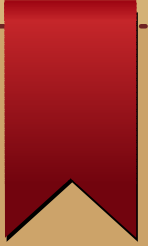
- Binary data can have byte pattern and it is illegal in XML document.
- “Null” have a different meaning and they can not be a part of Xml content
- To encode such content there are 2 standards
- Hexadecimal conversion
- Base-64 encoding (mapping table is used so file size increased by 25%)



Other Data Types

- Xsd: boolean
- Possible values: zero, one, true and false
- Xsd:anyURI to specify URI

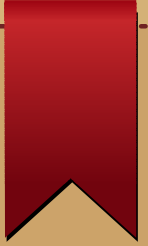
Grouping of Data



- Grouping can be done in 3 different ways:
 - Xsd:sequence grouping mandates that every element in a group must appear exactly once and also in same order.
 - Xsd:choice specifies only one element can appear in a group
 - Xsd:all grouping specifies that all the elements in a group must occur at the most once, but their ordering is not significant



Sequence



```
<?xml version="1.0"?>
```

```
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
```

```
<rollno> 132</rollno>
```

```
<name> ABC </name>
```

```
<marks> 74 </marks>
```

```
<result> distinction </result>
```

```
</student>
```



Schema file

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
<xsd:element name="student" type="studenttype"/>
<xsd:complexType name="studenttype">
  <xsd:sequence>
    <xsd:element name="rollno" type="xsd:string"/>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="marks" type="xsd:integer"/>
    <xsd:element name="result" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Making Choices

```
<?xml version="1.0"?>
```

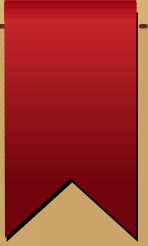
```
<result xmlns:xsi="
```

```
http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="student.xsd">
```

```
    <pass>75%</pass>
```

```
</result>
```



```
<?xml version="1.0"?>
```

```
<xsd:schema  
  xmlns:xs="http://www.w3.org/2001/XMLSchema"  
  xsd:element name="Result">
```

```
<xsd:complexType>
```

```
<xsd:choice>
```

```
  <xsd:element name="Pass" type="xsd:string">
```

```
  <xsd:element name="Fail" type="xsd:string">
```

```
</xsd:choice>
```

```
</xsd:complexType>
```

```
</xsd:schema>
```

Specifying the Frequency:minOccurs and maxOccurs

- The minOccurs attribute specifies the minimum number of occurrences that an element can have. maxOccurs attribute specifies the maximum number of occurrences



```
<xsd:element name="author" type="xsd:string"  
maxOccurs="2"/>
```

default value for both is 1

“unbound” data type specifies it may occur any number of times

Mandating all elements

- It means an element may occur, if it occurs, it must occur only once.
- The order of elements is not necessary
- Eg:
- `<xsd:complexType name="EmpType">`
- `<xsd:element name="name">`
- `<xsd:complexType>`
- `<xsd:all>`
- `<xsd:element name="First_name" type="xsd:string" minOccurs="1" maxOccurs="1"/>`
- `<xsd:element name="Last_name" type="xsd:string" minOccurs="1" maxOccurs="1"/>`
- `</xsd:all></xsd:complexType></xsd:element>`
- `<xsd:complexType>`

- 
- In case of all, maxOccurs and minOccurs can have only value of zero or one, other values are illegal.
- 

Mixed Content

- An XML element, "letter", that contains both text and other elements:



<letter>

Dear Mr.<name>John Smith</name>.

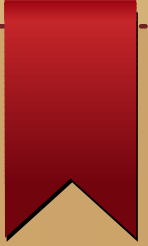
Your order <orderid>1032</orderid>

will be shipped on <shipdate>2001-07-13</shipdate>.

</letter>

- 
- `<xsd:element name="letter">`
 - `<xsd:complexType mixed="true">`
 - `<xsd:sequence>`
 - `<xsd:element name="name" type="xsd:string"/>`
 - `<xsd:element name="orderid" type="xsd:positiveInteger"/>`
 - `<xsd:element name="shipdate" type="xsd:date"/>`
 - `</xsd:sequence>`
 - `</xsd:complexType>`
 - `</xsd:element>`
- 

Deriving Types



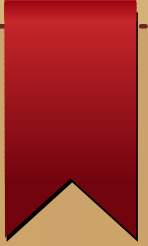
- Deriving simple types in XML schema
 - Restriction
 - Union
 - List



Deriving by Restriction

- Restriction allows us to select a subset of values allowed by the base type
- `<xsd:simpleType name="publishingyear">`
 `<xsd:restriction base="xsd:gYear">`
 `</xsd:restriction>`
`</xsd:simpleType>`

Facets



- String Facets
- The White space facet
- Facets for numbers
- Facet for enumeration
- The pattern facet



Restrictions for Datatypes

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Example

- `<xsd:simpleType name="publishingYear">`
 `<xsd:restriction base="xsd:gYear">`
 `<xsd:minInclusive value="1980"/>`
 `<xsd:restriction>`
 `<xsd:restriction>`

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

1. String Facets

- Xsd:length
- Xsd:minLength
- Xsd:maxLength

Example

- `<xsd:simpleType name="salutationType">`

`<xsd:restriction base="xsd:string">`

`<xsd:minLength value="2"/>`

`<xsd:maxLength value="3"/>`

`</xsd:restriction>`

`</xsd:simpleType>`

`<xsd:element name="person" type="personType"/>`

`<xsd:complexType name="personType">`

`<xsd:sequence>`

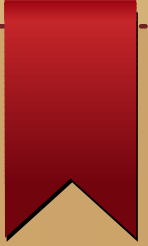
`<xsd:element name="salutation" type="salutationType"/>`

`.....</xsd:sequence>`

`</xsd:complexType>`


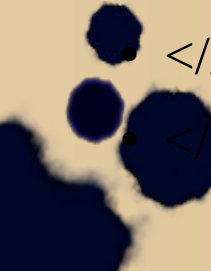
`</xsd:schema>`

2. The white space facet



- Xsd:whiteSpace facet allows to specify how to deal with white space
- Preserve: white space should be kept. This is default
- Replace: for replacing every tab, line feed and carriage return character
- Collapse: condenses multiple consecutive spaces in to single space





- 
- `<?xml version="1.0"?>`
 - `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
 - `<xsd:simpleType name="contentsType">`
 - `<xsd:restriction base="xsd:string">`
 - `<xsd:whiteSpace value="collapse"></xsd:whiteSpace>`
 - `</xsd:restriction>`
 - `</xsd:simpleType>`
 - `<xsd:element name="Poem" type="poemType"/>`
 - `<xsd:complexType name="poemType">`
 - `<xsd:sequence>`
 - `<xsd:element name="name" type="xsd:string"/>`
 - `<xsd:element name="contents" type="contentsType"/>`
 - `</xsd:sequence>`
 - `</xsd:complexType>`
 - `</xsd:schema>`
- 



<?xml version="1.0"?>

- <Poem xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xsd:noNamespaceSchemaLocation="whitespace.xsd">
- <name>Mr. s</name>
- <contents>khbjk ljhojhnj
-
- kjbnjlkbn;jkf jnhj
- ljn timer ihjb jhv timer jhv
-
- jn timer
-
- knjlk timer khbkj
- </contents>
- </Poem>
-

- 
- `<Poem
 xsd:noNamespaceSchemaLocation="whitespace.xsd"
 ><name>Mr. s</name><contents>`
 - `khbjk lhojhnj kjbni lkbn;jkf jnhj l jnhjn ihjb jhvjhvjh
 jhvh jnhnj knjlk nj khbkjk`
 - `</contents></Poem>`
- 

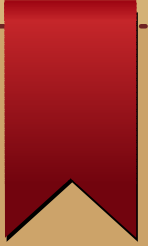
Facets for numbers

- Xsd:totalDigits
- Xsd:fractionDigits

Facets for enumeration

- `<?xml version="1.0"?>`
- `<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
- `<xsd:simpleType name="BookType">`
- `<xsd:restriction base="xsd:string">`
- `<xsd:enumeration value="JAVA"/>`
- `<xsd:enumeration value="XML"/>`
- `<xsd:enumeration value="C++"/>`
- `<xsd:enumeration value="AJAX"/>`
- `</xsd:restriction>`
- `</xsd:complexType>`
- `</xsd:schema>`

Unions



- Unions allow to combine simple types to create new simple types.
- Student can be identified uniquely either by roll number or name.
- New simple type which can have either rollno or name



Example

- `<?xml version="1.0"?>`
- `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
- `<xsd:element name="Student" type="RollnoOrName">`
- `<xsd:simpleType name="RollnoOrName">`
- `<xsd:union>`
- `<xsd:simpleType>`
 - `<xsd:restriction base="xsd:decimal"/>`
- `</xsd:simpleType>`
- `<xsd:simpleType>`
 - `<xsd:restriction base="xsd:string"/>`
- `</xsd:simpleType>`
- `</xsd:union>`
- `</xsd:simpleType>`
- `</xsd:schema>`

XML document



```
<?xml version="1.0"?>
```

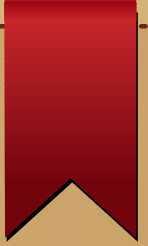
```
<student xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"  
xsd:noNamespaceSchemaLocation="union.xsd">
```

Rahul

```
</student>
```



Lists



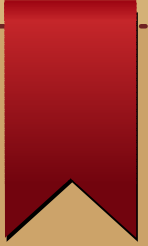
- A list type allows the creation of a list of a particular simple type
- ItemType attribute is used with `xsd:list` which allows to specify the type of each of the items in the list.



Lists

- `<?xml version="1.0"?>`
- `<xsd:schema xmlns:xsd="`
`http://www.w3.org/2001/XMLSchema">`
- `<xsd:element name="emp_no" type="EmpList">`
- `<xsd:simpleType name="EmpList">`
`<xsd:list itemType="xsd:int"/>`
- `</xsd:simpleType>`
- `</xsd:schema>`

XML document



```
<?xml version="1.0"?>
```

```
<emp_no xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"  
xsd:noNamespaceSchemaLocation="list.xsd">
```

```
123 3554 6567
```

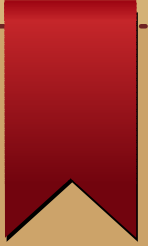
```
</emp_no>
```



Lists

- `<?xml version="1.0"?>`
- `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
- `<xsd:element name="emp_no" type="FiveEmp">`
- `<xsd:simpleType name="FiveEmp">`
 - `<xsd:restriction base="EmpList">`
 - `<xsd:length value="5"/>`
 - `</xsd:restriction>`
 - `<xsd:simpleType name="EmpList">`
 - `<xsd:list itemType="xsd:int"/>`
- `</xsd:simpleType>`
- `</xsd:schema>`

Lists



```
<?xml version="1.0"?>
```

```
<emp_no xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"  
xsd:noNamespaceSchemaLocation="list.xsd">
```


```
123 3554 6567 5454
```

```
</emp_no>
```



Empty Elements



- Can not have child elements or its own data
 - `<xsd:complexType name="empty1">`
 - `</xsd:complexType>`
- 

Attributes

- Use attribute to declare attribute

```
<attribute name="designation" type="xsd:string"  
use="required"/>
```

```
<attribute name="designation" type="xsd:string"  
use="optional"/>
```

```
<attribute name="designation" type="xsd:string"  
use="prohibited"/>
```

- Cannot use minOccurs or maxOccurs

Attributes

- It can also be fixed or default

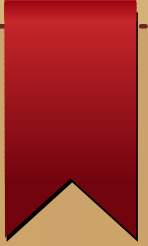
```
<attribute name="designation" type="xsd:string"  
use="required" fixed="Manager"/>
```

```
<attribute name="designation" type="xsd:string"  
use="optional" default="Manager"/>
```

Add attribute to element

- `<?xml version="1.0"?>`
- `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
- `<xsd:element name="employee" type="EmpType">`
- `<xsd:complexType name="EmpType">`
 - `<xsd:sequence>`
 - `<xsd:element name="emp_no" type="xsd:int"/>`
 - `<xsd:element name="emp_name" type="xsd:string"/>`
 - `</xsd:sequence>`
 - `<xsd:attribute name="emp_conf" type="xsd:boolean"/>`
- `</xsd:complexType>`
- `</xsd:schema>`

XML document



```
<?xml version="1.0"?>
```

```
<employee emp_conf="true"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema-  
instance" xsd:noNamespaceSchemaLocation="emp.xsd">
```

```
<emp_no>32</emp_no>
```

```
<emp_name>ram</emp_name>
```

```
</employee>
```



Grouping Attribute

- `<?xml version="1.0"?>`
- `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`
- `<xsd:element name="employee">`
- `<xsd:complexType>`
 - `<xsd:attributegroup ref="emp_details"/>`
 - `</xsd:complexType>`
- `<xsd:attributeGroup name="emp_details">`
 - `<xsd:attribute name="emp_no" use="required" type="xsd:int"/>`
 - `<xsd:attribute name="emp_name" use="required" type="xsd:string"/>`
 - `</xsd:attributeGroup>`
- `</xsd:complexType>`
- `</xsd:schema>`