

## Web Basics - XML

---


IGATE  
Speed.Agility.Imagination

Copyright © 2011 IGATE Corporation. All rights reserved. No part of this publication shall be reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior written permission of IGATE Corporation. IGATE Corporation considers information included in this document to be Confidential and Proprietary.

### Document History

Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
11-May-2010	2.0	XML	Tushar Joshi	Revamp/Refinements
11-May-2010	2.0	XML	Anu Mitra	Review
12-May-2010	2.0	XML	CLS Team	Review
18-April-2011	3.0	XML	Anu Mitra	Refinements according to Integrated curriculum
20-May-2013	3.1	XML	Sathiabama R	Revamped according to new curriculum
21-Apr-2015	3.2	XML	Rathnajoithi P	Revamped according to revised curriculum

## Course Goals and Non Goals



- **Course Goals**
  - To learn about how to create XML document
  - To understand the use of XML in web application development
  - To create schema definition
- **Course Non Goals**
  - To learn about how to create XSL and XSLT document
  - To understand DOM implementation
  - To learn XQuery

June 5, 2015

Proprietary and Confidential

- 3 -

IGATE  
Speed. Agility. Imagination.

**Note:**

**Goals:** Participants should be able to know how to create XML document, understand the use of XML in web application development, and creating schema definition.

**Non-Goals:** Participants will not learn about creation of XSL and XSLT file. DOM implementation is beyond the scope of this course. XQuery is not in scope of this course.

## Pre-requisites

- Fair Knowledge of HTML is preferable

June 5, 2015


Proprietary and Confidential

- 4 -

IGATE

Speed. Agility. Imagination.

# Intended Audience



➤ Web Developers

June 5, 2015

Proprietary and Confidential

- 5 -

IGATE

Ignite Your Imagination

Day Wise Schedule

➤ Day 1

Lesson 1: Introduction to XML

Lesson 2: Anatomy of XML Document

➤ Day 1

Lesson 3: XML Schemas

June 5, 2015

Proprietary and Confidential

- 6 -

IGATE

Speed. Agility. Imagination.

## Table of Contents

- **Lesson 1: Introduction to XML**
  - 1.1: Evolution of XML
  - 1.2: Role of XML in web applications
  - 1.3: Different members of XML family
  - 1.4: Introduction to Namespace
  
- **Lesson 2: Anatomy of XML**
  - 2.1: Logical and Physical structure of XML file
  - 2.2: Parts of XML file – Elements, Attributes, Entities, PI's etc.
  
- **Lesson 3: XML Schema Definition**
  - 3.1: Advantages of Schema over DTD
  - 3.2: Method to write a schema definition for an XML file
  - 3.3: Data types used in schemas
  - 3.4: Simple and Complex type of elements
  - 3.5: Indicator – Order, Occurrence, and Group
  - 3.6: Restrictions on XSD elements

## References

---

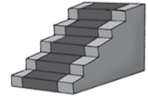
- **Books:**
  - Beginning XML; Wrox Publication
- **Sites:**
  - <http://w3schools.com>





## Next Step Courses (if applicable)

- XQuery
- Using XML with Java/.Net



## Other Parallel Technology Areas

➤ NA

June 5, 2015

Proprietary and Confidential

- 10 -

IGATE

Ignite Your Imagination

Web Basics - XML

Lesson 1: Introduction to XML

June 5, 2015

Proprietary and Confidential

- 1 -

IGATE  
Speed. Agility. Integration.

## Lesson Objectives

➤ **In this lesson, you will learn about:**

- Evolution of XML
- Role of XML in Web Applications
- Different members of XML family
- Introduction to Namespace



1.1: Introduction to SGML and HTML

## The Basics of Markup Language

### ➤ What do we mean by “Markup Language”?

- The term “markup” is used to identify anything put within a document which either adds or provides special meaning (for example, italicized text)
- A markup language is the set of rules
- It also provides a description of document layout and logical structure

June 5, 2015

Proprietary and Confidential

- 3 -

IGATE  
Speed. Agility. Innovation.

### The Basics of Markup Language:

There exist three types of markup:

- **Stylistic:** It determines how a document is presented (for example, the HTML tags <I> for italics, <B> for bold, and <U> for underline).
- **Structural:** It determines how the document is to be structured (for example, the HTML tags <P> for paragraph, <SPAN> for creating ad hoc styles in a document, and <DIV> for grouping structures that are aligned in the same way).
- **Semantic:** It tells about the content of the data (for example, the HTML tags <TITLE> for page title, <HEAD> for page header information, and <SCRIPT> to indicate a JavaScript in a page.)

In XML, the only type of markup that we are concerned with is structural.

## SGML

➤ **SGML stands for Standard Generalized Markup Language**

- SGML was conceptualized in 1974 and adopted as international standard in 1986
- SGML was born out of the basic need to make the data storage-independent
- SGML also does not have any specific document structure, and usage of tag set is not limited
- It does not constrain the potential of creating new document standards

June 5, 2015

Proprietary and Confidential

- 4 -

IGATE  
Speed. Agility. Innovation.

### SGML:

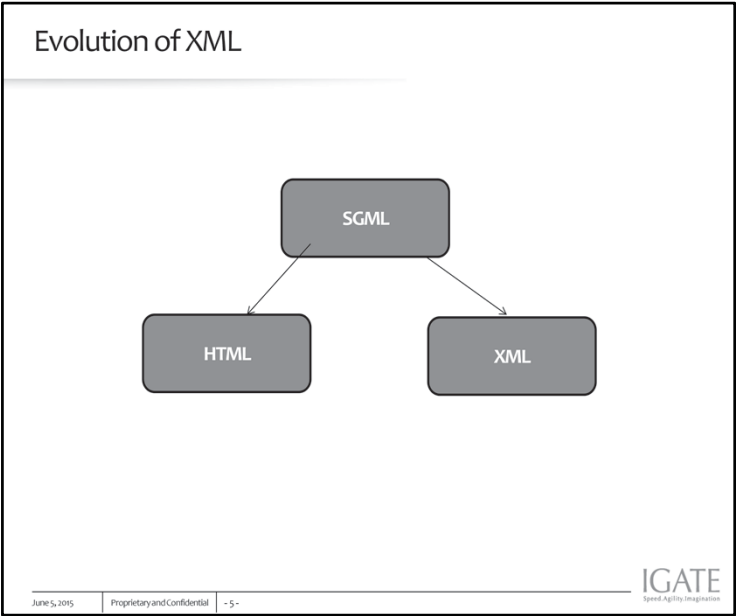
The **Standard Generalized Markup Language (SGML)**, from which XML is derived, has been around in various forms for quite some time now.

SGML may be best described by *what it is not*, as follows:

- SGML does not promote a specific document structure.
- There is not a limited tag set that must be used.
- SGML will not constrain the potential of creating new document standards.

SGML provides the common framework necessary to describe documents and to create new measures of conformance.

- Almost all languages that have been created to manipulate documents can trace at least a portion of their roots back to SGML.
- SGML itself is used by many large organizations, such as the United States Department of Defense, to handle complex electronic document exchanges. Avoiding the presentation features common to other document formats like PDF or even MS Word, SGML concentrates on the structure of the information.
- It does not promote one specific structure. However, it allows for the customized containment of data.



## Why Not Go Back To SGML?

- SGML is an incredibly rich meta language
- SGML is completely configurable
  - For example:
    - You can change the symbols for tagging from angle brackets (<tag>) to curly braces ({tag})
    - You can change the tag name lengths from 8 characters to 88 characters
- There is no style mechanism in SGML
- It is generic, just for generating customized language

June 5, 2005 Proprietary and Confidential - 6 -

IGATE  
Speed. Agility. Integration.

### Why Not Go Back to SGML?

If HTML is limiting and SGML provides a means of describing document structures, why not simply go back to SGML?

SGML is a language for creating languages. With SGML, you can create tag sets.

Example 1: The Air Transport Association used SGML to create a tag set for aircraft maintenance documentation.

Example 2: The Society of Automotive Engineers used SGML to create

a tag set for automotive service manuals.

SGML is an incredibly rich Meta language. It is completely configurable.

Example 1: You can change the symbols for tagging from angle brackets (<tag>) to curly braces ({tag}).

Example 2: You can change the tag name lengths from 8 characters to 88

characters. All this flexibility takes up computing power on the part of tools that interact with SGML. However, for the web, you need **lightweight tools** and processes. As a subset of SGML, in many ways XML serves as a lightweight, web-compatible version of SGML.

Also SGML has some more limitations

No mainstream browser support

No support for styles

Not much support for datatype

Security limitations



## HTML (Hypertext Markup Language)

- **HTML** which is an application of SGML, contains predefined set of tags and it is based on SGML manual
- **HTML** is a markup languages for web pages
- **Similar to SGML**
  - most tags describe meaning, not formatting
  - uses angled bracket convention (<tag></tag>)
  - based on a simple, widely compatible text format
- **Different from SGML**
  - HTML incorporates only one (standard document representation)

June 5, 2005 Proprietary and Confidential - 7 -

IGATE  
Speed. Agility. Intelligence.

### HTML:

The **Hyper Text Markup Language (HTML)**, which is an Application from SGML, contains predefined set of tags. HTML is designed for web publishing intended to describe structure of text.

However, HTML is **inflexible** in that it cannot allow domain-specific tag sets to be created and used without formally introducing them into the HTML DTDs.

Limitations of HTML

Browser specific commands for e.g Netscape-specific tags (e.g., <blink>), Microsoft IE tags (e.g., <marquee>)

Limited no of tags.

Some formatting commands not separating content and presentation e.g CENTER

HTML authors code to the browser's standards, not the W3C standards, therefore

Pages look different in different browsers & HTML validation is difficult

HTML talks of how and not what

Introduction of CSS to separate the look of the document to some extent

Introduction of XHTML to ensure standardization

## Introduction to XML

- What is needed a light-weight form of SGML which can provide well defined syntax for representing and processing document content over the web
- The answer is:
  - XML, the eXtensible Markup Language, is described as a means of structuring data
  - XML provides rules for placing text and other media into structures and allows you to manage and manipulate the results
  - XML standard is a subset of the SGML, developed in 1996 by the SGML working group

June 5, 2005

Proprietary and Confidential

- 8 -

IGATE  
Speed. Agility. Innovation.

### Introduction to SGML and HTML:

XML, the eXtensible Markup Language, is best described as a means of structuring data.

XML provides rules for placing text and other media into structures and

allows you to manage and manipulate the results.

The XML standard is a subset of the **Standard Generalized Markup Language**

**(SGML)**, and was developed in 1996 by the **SGML Editorial Review Board**

under the auspices of **World Wide Web Consortium (W3C)**.

XML is designed for that express purpose. XML provides a mechanism for the

interchange of structured information on the web. This sort of data is

required to transform the web from a publishing media to an application

processing environment

## XML Design Goals

- **The usage of XML was aimed at:-**
  - Should be usable over the Internet
  - Should support a wide variety of applications
  - Should be compatible with SGML and XML documents should be easy to create
- **Also XML can be used for**
  - Data Exchange
  - Store and Retrieve Data

June 5, 2015

Proprietary and Confidential

- 9 -

IGATE  
Speed. Agility. Innovation.

### XML Design Goals:

The design goals for XML were proposed by the World Wide Web Consortium (W3C), and published in January 1998.

The focus of XML is:

To carry the power of SGML through its ability to create user defined information about data, its ability to adapt to specific user needs, and the ability to maintain document changes.

To carry HTML's ease of use by the ability to link, its simplicity in the use of user defined tags, and its portability among different platforms.

Some more design goals are as follows:

It shall be easy to write programs which process XML documents.

The number of optional features in XML should be kept to the absolute minimum, ideally zero.

XML documents should be human-legible and reasonably clear.

The XML design should be prepared quickly.

The design of XML shall be formal and concise.

## XML Today

- The primary functional purpose of XML is to transfer structured text and data among systems in multiple organizations
- XML, unlike HTML, does not have a fixed format
  - There are no pre-defined tags; you create your own
- Like HTML, XML uses tags. Tags are always enclosed within angled-brackets (< >)
  - XML tags define the meta information and are distributed throughout the document
- XML 1.0 is the most widely used version

June 5, 2015

Proprietary and Confidential

- 10 -

**IGATE**  
 Speed. Agility. Innovation.

About XML:

XML versus HTML:

Both are based on SGML – the International Standard for structured information.

In HTML:

<p>P200 Laptop

<br>Friendly Computer Shop

<model>P200 Laptop</model>

<br>\$1438

Shop</dealer>

In XML:

<product>

<model>P200 Laptop</model>

<dealer>Friendly Computer

<price>\$1438</price>

</product>

Both XML and HTML may appear the same in your browser. However, the XML data is smart data. HTML tells how the data should look, but XML tells you what it means.

With XML, your browser knows there is a product, and it knows the model, dealer, and price. From a group of these, it can show you the cheapest product or closest dealer without going back to the server. Unlike HTML, with XML you create your own tags, so they describe exactly what you need to know. As a result, your client-side applications can access data sources anywhere on the Web, and in any format. New “middle-tier” servers sit between the data sources and the client, translating everything into your own task-specific XML.

## XML versus HTML

➤ **<table>**  
    **<tr>**  
        **<td>Apples</td>**  
        **<td>Bananas</td>**  
    **</tr>**  
**</table>**

**<table>** tag in HTML is predefined & used for creating tabular display

➤ **<table>**  
    **<name>African Coffee Table</name>**  
    **<width>80</width>**  
    **<length>120</length>**  
**</table>**

**<table>** tag in XML could mean anything e.g. its a coffee table which is a furniture

1.2: The Role of XML

XML and the Web

➤ **XML deals with what the data is about and how to specify the data structure**

- XML represents data formats on web for the following:
  - Books
  - Financial transactions (EDI)
  - Technical manuals
  - Chemical formulae
  - Medical records
  - Museum catalog records
  - Chess games
  - Encyclopedia entries

June 5, 2005

Proprietary and Confidential

- 12 -

IGATE  
Speed. Agility. Integration.

### The Role of XML:

XML will be most interesting to people and organizations who have:

Information resources that do not fit into the HTML mold, and

Resources that they want to make available over the web

XML was created to structure, store, and transport information. It is just plain text. Software that can handle plain text can also handle XML. However, XML-aware applications can specially handle the XML tags. The functional meaning of the tags depends on the nature of the application.

XML is as important for the web, as HTML was to the foundation of the web.

XML is everywhere. It is the most common tool for data transmissions between all sorts of applications, and is popular in the area of storing and describing information.

Some examples:

Books

Financial transactions (EDI)

Technical manuals

Chemical formulae

Medical records

Museum catalog records

Chess games

Encyclopedia entries

1.3: Introducing XML and Its Relatives

A Family of Standards

➤ XML is a group of technologies

➤ It consists of the following specifications:

- Extensible Style Language (XSL)
- XML Linking Language (including Xpath, Xlink, and Xpointer)
- XML Namespaces

June 5, 2015

Proprietary and Confidential

- 13 -

IGATE  
Speed. Agility. Innovation.

Introducing XML and its Relatives:

XML is a group of technologies.

It consists of the following specifications:

eXtensible Style Language: XSL works with the XML data in a way similar to the manner in which CSS works with HTML.

XML Linking: XML linking and addressing mechanisms are specified in three W3C Working Draft documents:

XML Path Language (Xpath): The primary purpose of Xpath is to do the actual addressing of parts rather than the whole XML document.

XML Linking Language (Xlink): It uses XML syntax to create structures to describe both “simple unidirectional links” of today’s HTML as well as more “sophisticated multidirectional links”. The important part of Xlink is that it defines the relationship between two or more data objects (or portion of objects) as opposed to a whole document.

XML Pointer Language (Xpointer): Xpointer builds on Xpath to support addressing into the internal structures of XML documents. Thus you can use the XML markup to link to specific parts of another document without supplying an ID reference.

XML Namespaces

## Extensible Style Language (XSL)

- **Cascading Style Sheets (CSS)** makes it possible for the same HTML content to be easily formatted in multiple ways
- **Extensible Style Language (XSL)** works with XML data in a way similar to that CSS works with HTML
  - The rules created with the style language – the style sheet – should define how the content will be displayed
  - Formatting should not appear in the content itself



## XML Namespaces

- XML Namespaces provide a way of assigning unique names to document constructs so that the software can operate correctly and avoid collisions
- XML Namespaces allow context to be given to the element names
  - This allows them to remain unique and thus process able

June 5, 2015

Proprietary and Confidential

- 15 -

IGATE  
Speed. Agility. Innovation.

### XML Namespaces:

XML Namespaces are a way of assigning “unique names” to document constructs so that software can operate correctly and avoid collisions.

Namespaces allow context to be given to element names, which allow them to remain unique and thus processable.

## Summary

➤ **In this lesson, you have learnt that:**

- SGML is the Standard Generalized Markup Language
- HTML is the Hypertext Markup Language and XML is the Extensible Markup Language (meta-markup language)
- XML is not a replacement for HTML
- HTML tags do not say anything about the structure of the information
- HTML lacks in link management, is not reusable, is not Object Oriented, and so on
- Looking at the future of electronic commerce, HTML has limitations
- XML is a project of w3c and its implementation is in the developing stage
- XML uses features of SGML but it is easy compare to it
- Markup Language created using XML are called XML vocabularies or XML applications



June 5, 2005

Proprietary and Confidential

- 16 -

IGATE  
Speed. Agility. Innovation.

### Summary: Why XML?

Many of the most influential companies in the software industry are promoting XML as the next step in the evolution of the web. How can they be so confident about something so new?

We can all safely bet on XML because the central ideas in this new technology are in fact very old and have been proven correct across several decades and thousands of projects.

#### Database Publishing:

One particularly popular application of XML will surely be the publishing of databases to the web.

Consider for instance a product database, used by the internal ordering system of a toy manufacturer. The manufacturer might want the database to be available on the web so that potential clients know the toys that are available and their price.

Rather than having someone in the web design department to mark up the data again, they can build a connection between their web server and their database using the features typically built into web servers that allows those sorts of data pipes. The designer can then make the product list beautiful using a style sheet. Pictures of the toys can be supplied by the database. In essence, the web site will be merely a view on the data in the database. As toys get added and removed from the database, they will appear and disappear from the view on the website.

Summary:

Database Publishing (contd.):

XML is also expected to become an important tool for interchange of database information. They are the “documents produced by and for computer software”.

Databases have typically interchanged information using simple file formats such as one-record per line with semi-colons between the fields. This is not sufficient for the new object-oriented information being produced by databases. Objects must have internal structure and links between them. XML can represent this using elements and attributes to provide a common format for transferring database records between databases.

Today’s web model is a “client/server” model. Queries from the customer go to the server, and resulting responses are shipped back to the customer for viewing in HTML. Unfortunately, a web server can handle only a limited number of connections at one time.

Today, XML has enabled a new breed of web server software, one that allows the web developer to add a new “middle-tier” server to the web model.

Summary:

Database Publishing (contd.):

In the old web model, the customer using browser such as IE or Netscape on the client interacted directly with data sources on remote servers. The client maintained its connection throughout the interactive session. Each query was sent a response in HTML, which could be directly viewed by the client browser. Maintaining the connection between the client and server was critical.

In the new three-tier web model, the information that fits the profile of the customer is retrieved at once from the remote databases by software on the middle tier, either as XML documents or through an ODBC or similar database connection. From that point, continued interaction with the remote databases is no longer required. The connection to the remote servers can be, and is, terminated.

Once all information that fits the customer profile has been assembled by software on the middle tier, it is sent in XML to the client. Now the requirement for further interaction between the client and the middle tier server is eliminated as well.

Rich XML data, directly usable by client applications and scripting languages like JavaScript, has been delivered to the client. The connection between the client and the middle tier server can now be terminated. At this point, all computing becomes client-based, resulting in a much more efficient use of the web and a much more satisfying customer experience.

## Review Question

- **Question 1:** Which of the following is/are true about SGML?
  - Option 1: makes Data storage independent
  - Option 2: usage of tag set is unlimited
  - Option 3: both the above
- **Question 2:** \_\_\_\_ allows to apply style to XML
- **Question 3:** XML namespace provides \_\_\_\_



Web Basics - XML

Lesson 2: Anatomy of an XML Document

June 5, 2015

Proprietary and Confidential

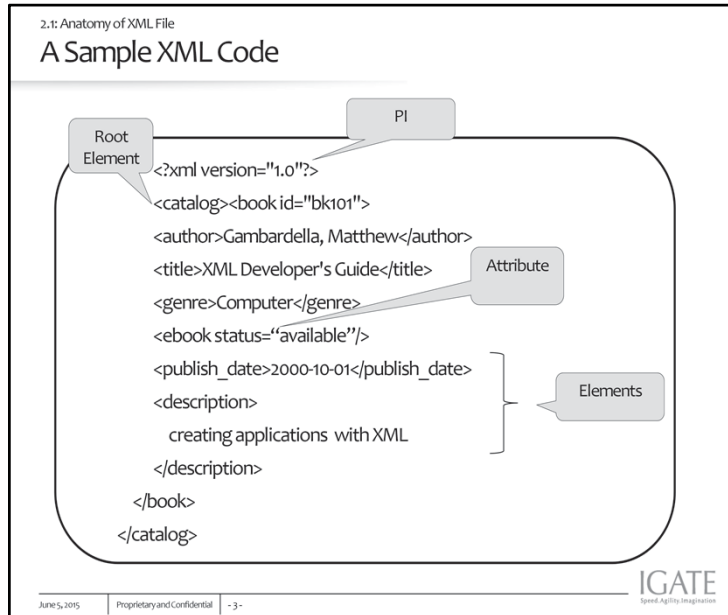
- 1 -

IGATE  
Ignite. Adapt. Inspire.

## Lesson Objectives

- In this lesson, you will learn:
- Logical and physical structure of an XML file
- Parts of XML file like:
  - Elements
  - Attributes
  - Entities
- Processing instructions
- Creating DTD





Elements can be simple, empty, mixed

Simple : `<data>value</data>`

Mixed : `<data> value`

`<sub>sub1</sub1>`

`</data>`

Empty : `<data></data>`



## Understanding the Sample XML Code

➤ Let us now understand the different parts of the XML file:

- XML Declaration
- Root Element
- An Empty Element
- Attributes

June 5, 2015

Proprietary and Confidential

- 4 -

IGATE  
Speed. Agility. Imagination.

### Understanding the Sample XML Code:

#### XML Declaration:

It is a processing instruction (identified by the ? at its start and end).

#### Root Element:

Each XML document must have only one root element, all the other elements must be completely enclosed in that element.

Line 2 (in example) identifies the start element (the start tag), and line 12 identifies the end of the element (the end tag).

#### Empty Elements:

Empty elements have no content and are marked up as either of the following:

`<empty_element/>`

`<empty_element></empty_element>`

#### Attribute Markup:

Attributes are used to attach information to the information contained in an element. The general form for using an attribute is as follows:

`<element-name property="value">`

2.4: XML Markup

## Using XML Markup

- Tags carry the smallest unit of meaning signifying structure, format, or style of the data
- They are always enclosed within angled brackets, that is "< >". Tags are case-sensitive
- The tags <book>, <Book>, and <BOOK> carry different meanings and cannot be used interchangeably
- All tags must be paired so that they have a start <book> and an end </book>
- Tags when combined with data form elements

June 5, 2015

Proprietary and Confidential

- 5 -

IGATE  
Intellectual Property Management

### Using XML Markup:

XML is concerned with element markup.

Instead of XML's tags being markers that indicate where a style should change or a new line should begin, XML's element markup is composed of three parts:

- The start tag
- The content
- The end tag

### Elements:

- Elements contain information or content and can also contain other elements.
- There is one element that contains all the other elements called the "root element".
- Tags show the beginning and end of an element.
- XML documents are divided into containers called "elements". People who are familiar with HTML, know that <p> ... </p>, <form> ... </form>, <br> are all elements.

## Using XML Markup

### ➤ Attribute Markup:

- It is used to attach information to the information contained in an element.
- General form for using an attribute is as follows:
- `<element-name property="value">`
- An attribute value must be enclosed in quotation marks.
- You can either use single quote or double quote. However, you cannot mix the two in the same specification.

June 5, 2005

Proprietary and Confidential

- 6 -

IGATE  
Speed. Agility. Simplicity.

### Using XML Markup: Attributes:

Attributes are element modifiers. They provide additional and more specific information about an element and its content.

Normally in HTML, attributes are used most often to provide the browser with a suggestion for formatting the display of the elements content by a browser.

For example: bgcolor attribute of <body> element or align attribute normally are used with almost all elements.

However, the same is not true with XML. The attributes are used to provide further information about the element itself. This is because the main purpose of XML is to separate markup from display, so you will rarely see formatting attributes in XML DTDs.

## Using XML Markup

### ➤ Naming Rules:

- A name consists of at least one letter: a to z or A to Z
- If the name consists of more than one character, then it may start with an underscore ( \_ ) or a colon ( : )
- The initial letter can be followed by one or more letters, digits, hyphens, underscores, or full stops

June 5, 2015

Proprietary and Confidential

- 7 -

IGATE  
Speed. Agility. Innovation.

### Using XML Markup:

#### Naming Rules:

A name consists of at least one letter: a to z or A to Z.

If the name consists of more than one character, then it may start with an underscore ( \_ ) or a colon ( : )

The initial letter can be followed by one or more letters, digits, hyphens, underscores, and full stops.

## Using XML Markup

### ➤ Comments:

- Comments have the following form:
  - `<!--This is comment text-->`
- Use the comment start tag and end tag correctly.
- Everything in the comment text will be completely ignored by the XML processor
- Following comment is therefore quite safe:
  - `<!-- These are the declaration for the <title> and <body>-->`

June 5, 2015

Proprietary and Confidential

- 8 -

IGATE  
Speed. Agility. Innovation.

### Using XML Markup: Comments

In keeping with the design constraint of keeping XML simple, its comment facilities are also simple. Comments have the following form:

`<!--This is comment text-->`

Provided that you use the comment start tag and end tag correctly, everything in the comment text will be completely ignored by the XML processor. The following comment is therefore quite safe:

`<!-- These are the declaration for the <title> and <body>-->`

There is only one restriction on what you can place in your comment text: the string `--` is not allowed. This keeps XML backward compatible with SGML.

### Using XML Markup

➤ **Predefined Entities:**

Character	Replacement
&	&amp; or &#38; #38
'	&apos; or &#39
>	&gt; or &#62
<	&lt; or &#60; #60
"	&quot; or &#34

June 5, 2015

Proprietary and Confidential

- 9 -

IGATE  
Ignored Rights Management

Using XML Markup:  
Predefined Entities:

The special characters for quote ("), apostrophe ('), less-than (<), greater-than (>), and ampersand (&) are used for punctuation in XML, and are represented with predefined entities: &quot;, &apos;, &lt;, &gt;, and &amp;.

Notice that the semicolon is part of the entity. You cannot use "<" or "&" in attributes or elements.

## A Well-formed XML document

- A well-formed XML document simply includes markup pages with descriptive tags
- A well-formed XML does not need a DTD, but should conform to XML syntax
- If all tags are correctly formed and follow XML guidelines, then the document is a well-formed XML

June 5, 2015

Proprietary and Confidential

- 10 -

IGATE  
Speed. Agility. Imagination.

The XML syntax is discussed on the next slide.

## Syntax Rules for XML

### ➤ An XML document

- Is case sensitive
- Has a single root element
- Has all matching tags
- XML Elements should be properly nested
- All attributes are quoted
- White spaces are not ignored
- May or may not have a (DTD) Document Type Description to describe the document



## Demo

➤ **A sample XML Document:**

- Example1: Note.xml
- Example2: Greeting.xml
- Example3: musicians.xml



## Summary

➤ **In this lesson, you have learnt the following:**

- XML has specific naming rules which describes names you can use for its markup objects, that is elements



## Review Question

- Question 1: XML document must have one \_\_\_\_.
- Question 2: A comment in XML document is written as:
  - Option 1: <!-- ... -->
  - Option 2: /\* .....\*/
  - Option 3: //
- Question 3: \_\_\_\_ are storage units in the XML document.



# Web Basics - XML

## Lesson 3: XML Schema Definition

June 5, 2015

Proprietary and Confidential

- 1 -

IGATE  
Speed. Agility. Imagination.

## Lesson Objectives

➤ **In this lesson, you will learn about:**

- Advantages of Schema over DTD
- Method to write a schema definition for an XML file
- Data types used in schemas
- Simple and Complex type of elements
- Indicator – Order, Occurrence, and Group
- Restrictions on XSD elements



4.1: Advantages of Schemas over DTD

## Introduction to XML Schema

- The XML Schema Definition Language is an XML language for describing and constraining the content of XML documents
- XML Schema is a W3C recommendation
- XML Schema defines what it means for an XML document to be valid
- XML Schema are a radical departure from Document Type Definitions (DTDs), the existing schema mechanism inherited from SGML

June 5, 2015

Proprietary and Confidential

- 3 -

IGATE  
Speed. Agility. Imagination.

### What You Should Already Know

Before you continue, you should have a basic understanding of the following:

HTML/ XHTML

XML and XML Namespaces

A basic understanding of DTD

### Introduction to Schema:

An XML document is essentially a structured medium for storing information. In order to assess the validity of a XML document, you need to establish exactly to which structure the information within the document must adhere. This is accomplished with **schema**.

A schema describes the arrangement of **markup** and **character data** within a valid XML document. It describes the grammar, vocabulary, structure, datatypes, etc. of a XML document. A traditional schema solution is DTD. We are already familiar with DTD and XML namespaces.

## XML Schemas

### ➤ Drawbacks of DTD:

- Use of non-XML syntax
- No support for data typing
- Non-extensibility

June 5, 2015

Proprietary and Confidential

- 4 -

IGATE  
Speed. Agility. Imagination.

### Advantages of Schemas over DTD:

#### XML Data Modeling with DTDs:

We are familiar with DTD for XML document. DTDs originated from SGML and provide a standard mechanism for validating SGML documents.

XML is subset of SGML. Hence it also uses the same approach. However, DTDs have some drawbacks. So a new approach for modeling XML data has come up. It is called as **XML schema**.

DTDs rely on a specialized syntax for describing the structure of XML vocabularies. This is a drawback of DTD. The question arises, why is it necessary to learn a specialized syntax, when XML itself provides a suitable syntax for describing data of any kind.

#### Disadvantages of DTD:

Good DTDs are difficult to write.

There is no provision for inheritance from one DTD to another.

DTDs do not provide support for namespaces.

It is limited in its descriptive powers.

## Why Use XML Schemas?

### ➤ XML Schemas

- support data types
- use XML syntax
- secure data communication
- are extensible
- Well-Formed is not enough

June 5, 2015

Proprietary and Confidential

- 5 -

**IGATE**  
Speed. Agility. Imagination.

### Why Use XML Schemas?

As mentioned earlier XML Schemas have advantages over using DTD.

Let us now see some more reasons why we should use XML Schemas.

**Support Data Types:** XML Schemas have support for datatypes. This makes it simple to describe allowable document content, to validate the data correctness. It is also easy to work with data from database, defining restrictions on data and/or data formats. It also allows conversion of data between different data types.

**Use of XML Syntax:** When writing XML Schemas you follow XML syntax. Hence you can use the XML Editors and Parsers to work with the Schema files. In addition to this you also do not need to learn a different language.

**Secure Data Communication:** During data transfer it is essential that both dispatcher and receiver of the data have the same understanding about the transferred content. The dispatcher will have to depict the data in such a way that it is understood by the receiver.

**Are extensible:** XML schemas can be inherited i.e one XML schema can be extended by another XML schema. You can also create your own datatypes from standard datatypes.



## XML Schema

➤ **An XML Schema defines:**

- Elements that can appear in a document
- Attributes that can appear in a document
- The elements that are child elements
- The order of child elements
- The number of child elements
- The criteria whether an element is empty or can include text
- Data types for elements and attributes
- Default and fixed values for elements and attributes

June 5, 2015

Proprietary and Confidential - 6 -

IGATE  
Speed. Agility. Imagination.

### Advantages of Schemas over DTD:

#### Example:

Consider the following example:

```
<!ELEMENT pin-code #PCDATA>
```

Now, consider the following statement:

```
<pin-code>ABC-123444-hhh</pin-code>
```

It is both well-formed and valid even though ABC-123444-hhh certainly does not represent a pin code in any form.

The data-type constraints available in schemas can allow the schema designer to limit the content of the pin-code element to a six digits number, for example, 400090.

XML Schemas are the successors of DTDs.

## Namespaces

- **XML Namespaces provide a method to avoid element name conflicts**
  - Name Conflicts: In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications
  - XML Namespaces provides a method to avoid element name conflicts

## Namespaces

The diagram shows three rounded rectangular boxes containing XML code snippets. The top-left box contains a standard table element. The top-right box contains a table element with attributes. The bottom-left box contains a container element 'tables' with two table elements inside. A callout bubble points from the bottom-right of the 'tables' box to the text 'How do you differentiate between these table?'.

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

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

```
<tables>
  <table> ....</table>
  <table> ....</table>
</tables>
```

How do you differentiate between these table?

June 5, 2015    Proprietary and Confidential    - 8 -

IGATE  
Speed. Agility. Imagination

### Namespaces:

If the XML fragments in the above slide were added together, then there would be a name conflict. Both contain a `<table>` element, but the elements have different content and meaning.

An XML parser will not know how to handle these differences.

## Namespaces

- The namespace attribute is placed in the start tag of an element and has the following syntax:  
`xmlns:namespace-  
prefix="namespace"`
- The W3C namespace specification states that the namespace itself should be an Uniform Resource Identifier (URI)
- When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace

## Solving the Name Conflict Using a Prefix

### ➤ Code Snippet

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
      <h:td>Apples</h:td><h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table xmlns:f="http://www.w3schools.com/furniture">
```

June 5, 2015

Proprietary and Confidential

- 10 -

**IGATE**  
Speed. Agility. Imagination.

### Solving the Name Conflict using a Prefix:

In the example in the above slide, there will be no conflict because the two <table> elements have different names. This XML carries information about an HTML table, and a piece of furniture.

When using prefixes in XML, a so-called **namespace** for the prefix must be defined.

The namespace is defined by the **xmlns attribute** in the start tag of an element.

The namespace declaration has the following syntax. xmlns:prefix="URI".

In the example on the above slide, the xmlns attribute in the <table> tag gives the h: and f: prefixes a qualified namespace.

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

## Solving the Name Conflict Using a Prefix

### ➤ Code Snippet continued

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

4.2: Writing a Schema Definition for an XML File

**Illustration(Message.xsd)**

➤ Let us see an example on writing a schema definition:

```

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<xs:element name="message">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>

```

June 5, 2015

Proprietary and Confidential

- 12 -

**IGATE**  
 Speed. Agility. Imagination
**Creating Schema Document:**

The <schema> element is the root element of every XML Schema. The <schema> element contains some attributes. A schema declaration often looks like something as shown in the above slide.

xmlns:xs= http://www.w3.org/2001/XMLSchema

It implies that the elements and data types used in the schema are from "http://www.w3.org/2001/XMLSchema" namespace. It also signifies that any elements and datatypes referred from here should have the prefix "xs".

Some more optional attributes:

targetNamespace="patniNamespace"

This value is a unique identifier. The value could be anything. Place this attribute at the top of the XSD means all entities are part of the namespace

xmlns=" http://www.w3.org/2001/XMLSchema "

It indicates that the default namespace is "http://www.w3.org/2001/XMLSchema".

elementFormDefault="qualified"

It signifies that any elements used by the XML instance document that were declared in this schema must be qualified by namespace.

## 4.2: Writing a Schema Definition for an XML File

## Illustration(Message.xsd)

## ➤ Code Snippet continued

```
<xs:element name="subject" type="xs:string"/>
<xs:element name="text" type="xs:string"/>
<xs:attribute name="priority" type="xs:string" use="required"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```



## Using XSD in XML Document

### ➤ Example:

```
<note xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="message.xsd">
```

- Include the above in the XML document, to reference the XML Schema definition

June 5, 2015

Proprietary and Confidential - 14 -

**IGATE**  
Speed. Agility. Imagination.

### Writing a Schema Definition for an XML File:

#### Using XSD in XML Document:

xmlns=" http://www.w3.org/2001/XMLSchema" indicates the default namespace declaration which tells the schema-validator that all the elements used in this XML document are declared in the "http://www.w3.org/2001/XMLSchema" namespace.

When you have the XML Schema Instance namespace available, that is "http://www.w3.org/2001/XMLSchema-instance ", you can use the schemaLocation attribute. This attribute has two values:

The first value is the namespace to use.

The second value is the location of the XML schema to use for that namespace  
"xsi:schemaLocation="message.xsd "

## XML-Schema Definition

### ➤ Simple Element:

- `<xs:element name="title" type="xs:string"/>`
  - where "title" is the name of the element and "xs:string" is the data type of the element

### ➤ Specifying default or fixed values:

- `<xs:element name="title" type="xs:string" default="No Title"/>`
- `<xs:element name="category" type="xs:string" fixed="Common"/>`

June 5, 2015

Proprietary and Confidential

- 15 -

**IGATE**  
 Speed. Agility. Innovation

### Writing a Schema Definition for an XML File:

#### What is a Simple Element?

A simple element is an XML element that can contain only text. It cannot contain any other element or attribute. The text can be of many different types. It can be one of the types that are included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions to a data type in order to limit its content, and you can make it mandatory for the data to match a defined pattern.

#### Default and Fixed Values for Simple Elements:

Simple elements may have a default value or a fixed value specified.

A default value is automatically assigned to the element when no other value is specified. In the above example, default value is "No Title".

A fixed value is also automatically assigned to the element, and you cannot specify another value. In the above example, the fixed value is "common".

Here are some XML elements:

```
<lastname>Refsnes</lastname>
      <age>36</age>
<dateborn>1970-03-27</dateborn>
```

Here are the corresponding simple element definitions:

```
<xs:element          name="lastname"          type="xs:string"/>
<xs:element          name="age"               type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

## XML Schema Data Types

➤ **XML Schema Data Types belongs to following categories:**

- XSD String: String data types are used for values that contains character strings.
- XSD Date: Date and time data types are used for values that contain date and time.
- XSD Numeric: Numeric data types are used for numeric values
- XSD Misc: Other miscellaneous data types like boolean, base64Binary, hexBinary, float, double, etc.

June 5, 2015

Proprietary and Confidential - 16 -

IGATE  
Speed. Agility. Imagination

### XML Schema Vocabulary:

XML Schema has support for data types.

With the support for data types it is easier:

- to describe permissible document content.
- to validate the correctness of data.
- to work with data from a database.
- to define data patterns (data formats).
- to convert data between different data types.

Apart from the above advantages, XML schemas are **extensible**. It implies that you can reuse your Schema in other Schemas and also reference multiple schemas from the same document.

## String Data Types

### ➤ String Data Type:

- `<xs:element name="Author" type="xs:string"/>`  
`<Author>John Smith</Author>`

### ➤ NormalizedString Data Type:

- `<xs:element name="Author" type="xs:normalizedString"/>`  
`<Author>John Smith</Author>`

### ➤ Token Data Type:

- `<xs:element name="Author" type="xs:token"/>`  
`<Author>John Smith</Author>`

June 5, 2015

Proprietary and Confidential - 17 -


 IGate  
 Speed. Agility. Imagination

### String Data Types:

#### String Data Type:

The string data type can contain characters, line feeds, carriage returns, and tab characters.

#### NormalizedString Data Type

The normalizedString data type is derived from the String data type.

The normalizedString data type also contains characters, but the XML processor will remove line feeds, carriage returns, and tab characters.

#### Token Data Type

The token data type is also derived from the String data type.

The token data type also contains characters, but the XML processor will remove line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces.

#### Restrictions on String Data Types

Restrictions that can be used with String data types:

- enumeration
- length
- maxLength
- minLength
- pattern
- whiteSpace

## Date and Time Data Types

### ➤ Date Data Type:

```
<xs:element name="publishdate" type="xs:date"/>
<publishdate>2002-09-24</publishdate>
```

### ➤ Time Data Type:

```
<xs:element name="publishtime" type="xs:time"/>
<publishtime>09:00:00</publishtime>
```

### ➤ DateTime Data Type:

```
<xs:element name="publishdatetime" type="xs:dateTime"/>
<publishdatetime>2002-05-30T09:00:00</publishdatetime>
```

June 5, 2015

Proprietary and Confidential - 18 -


 IGate  
Speed. Agility. Imagination.

### Date and Time Data Types:

#### Date Data Type:

The date data type is used to specify a date.

The date is specified in the following form "YYYY-MM-DD", where YYYY indicates the year, MM indicates the month, and DD indicates the day.

#### Time Data Type:

The time data type is used to specify a time.

The time is specified in the following form "hh:mm:ss", where hh indicates the hour, mm indicates the minute, and ss indicates the second.

#### DateTime Data Type:

The dateTime data type is used to specify a date and a time.

The dateTime is specified in the following form "YYYY-MM-DDThh:mm:ss", where YYYY indicates the year, MM indicates the month, DD indicates the day, T indicates the start of the required time section, hh indicates the hour, mm indicates the minute, and ss indicates the second.

Note: All components are required in all categories!

#### Restrictions on Date Data Types

Restrictions that can be used with Date data types:

Enumeration, maxExclusive, maxInclusive, minExclusive, minInclusive, pattern, whiteSpace

## Numeric Data Types

### ➤ Decimal Data Type:

- `<xs:element name="price" type="xs:decimal"/>`  
`<price>999.50</price>` or  
`<price>+999.5450</price>` or  
`<price>-999.5230</price>`

### ➤ Integer Data Type:

- `<xs:element name="price" type="xs:integer"/>`  
`<price>999</price>` Or  
`<price>+999</price>` Or  
`<price>-999</price>`

June 5, 2015

Proprietary and Confidential

- 19 -

**IGATE**  
 Speed. Agility. Imagination

### Numeric Data Types:

**Decimal Data Type:** The decimal data type is used to specify a numeric value.

**Integer Data Type:** The integer data type is used to specify a numeric value without a fractional component.

**Numeric Data Types:** All the data types below derive from the Decimal data type (except for decimal itself)!

**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

**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

## Miscellaneous Data Types

➤ **Boolean Data Type:**

- `<xs:element name="disabled" type="xs:boolean"/>`  
`<disabled>true</disabled>`

➤ **Binary Data Types:**

- `<xs:element name="blobsrc" type="xs:hexBinary"/>`

➤ **AnyURI Data Type:**

- `<xs:element name="PicSrc" type="xs:anyURI"/>`  
`<PicSrc>"http://www.w3schools.com/images/smiley.gif" </ PicSrc >`

June 5, 2015

Proprietary and Confidential

- 20 -

IGATE  
Speed. Agility. Imagination.

### Miscellaneous Data Types:

**Boolean Data Type:** The boolean data type is used to specify a true or false value.

**Binary Data Types:** Binary data types are used to express binary-formatted data. We have two binary data types:

- base64Binary (Base64-encoded binary data)
- hexBinary (hexadecimal-encoded binary data)

**AnyURI Data Type:** The anyURI data type is used to specify a URI.

## Attribute in XSD

### ➤ Defining an Attribute:

- `<xs:attribute name="AuthorID" type="xs:string"/>`
  - where "AuthorID" is the name of the attribute and "xs:string" specifies the data type of the attribute.

### ➤ Creating Optional and Required Attributes:

- `<xs:attribute name="btype" type="xs:string" use="required"/>`

### ➤ Attributes are optional by default

June 5, 2015

Proprietary and Confidential

- 21 -

**IGATE**  
 Speed. Agility. Imagination

### Attribute in XSD:

#### What is an Attribute?

Simple elements cannot have attributes. If an element has attributes, it is considered to be of complex type. However, the attribute itself is always declared as a simple type. This means that an element with attributes always has a complex type definition.

#### Example:

```
<Author AuthorID="A001">Smith</Author>
```

Following is a corresponding simple attribute definition:

```
<xs:attribute name="AuthorID" type="xs:string"/>
```

#### Default and Fixed Values for Attributes

Attributes may have a default value or a fixed value specified.

A default value is automatically assigned to the attribute when no other value is specified.

In the following example the default value is "UnKnown":

```
<xs:attribute name="AuthorID" type="xs:string"
  default="UnKnown"/>
```

#### Optional and Required Attributes

Attributes are optional, by default. To specify that the attribute is required, use the "use" attribute:

```
<xs:attribute name="AuthorID" type="xs:string"
  use="required"/>
```



## 4.3: Simple and Complex Type of Elements

## Complex Element

## ➤ Illustration:

```

<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

June 5, 2015

Proprietary and Confidential

- 22 -

**IGATE**  
 Speed. Agility. Imagination
Complex Element:

## Complex Element:

A complex element is an XML element that contains other elements and/or attributes.

There are four kinds of complex elements:

- Empty elements

- Elements that contain only other elements

- Elements that contain both – other elements and text

- Elements that contain text

## Examples of Complex XML Elements:

## Example 1:

Consider a complex XML element, “product”, which is empty:

➤ <product pid="1345"/>

It can be declared as:

```

<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>

```

Restrictions on XSD Elements:

Restrictions on Content:

When an XML element or attribute has a datatype associated with , it puts a restriction on the element’s or attribute’s content. If an XML element is of type “xs:integer” and contains a string value “Nice Day”, then the element will not validate.

With XML Schemas, you can also add your own restrictions to your XML elements and attributes. These restrictions are called **facets**.

Some restrictions that apply on XSD elements are as follows:

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 the manner in which white space (line feeds, tabs, spaces, and carriage returns) is handled

## Restriction on Values

### ➤ Example

```
<xs:element name="Quantity"><xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="500"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

June 5, 2015

Proprietary and Confidential

- 24 -

**IGATE**  
Speed. Agility. Imagination.

### Restrictions on XSD Elements:

#### Restrictions on Values:

The example in the above slide defines an element called “Quantity” with a restriction. The value of book “Quantity” cannot be lower than 0 or greater than 500.

## Restriction on Set Values

### ➤ Example

```
<xs:element name="Category">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Dot Net"/>
      <xs:enumeration value="BI"/>
      <xs:enumeration value="RDBMS"/>
      <xs:enumeration value="J2EE"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- The “Category” element is a simple type with a restriction.
- The acceptable values are Dot Net, BI, RDBMS, and J2EE

IGATE  
Speed Agility Imagination

June 5, 2015

Proprietary and Confidential

- 25 -

### Restriction on Set of Values:

According to the example shown on the above slide, the Element category can have only four possible values which are Dot Net, BI, RDBMs, and J2EE.

## Restrictions on Series of Values

- To limit the content of an XML element to define a series of numbers or letters that can be used, we can use the pattern constraint.

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- The only acceptable value is ONE of the LOWERCASE letters from a to z
- The “Category” element is a simple type with a restriction.
- The acceptable values are Dot Net, BI, RDBMS, and J2EE

## Restrictions on Series of Values

➤ Some more examples of Pattern

[a-zA-Z][a-zA-Z][a-zA-Z]	THREE of the LOWERCASE OR UPPERCASE letters from a to z
[0-9]{10}	Any 10 digit number
[A-Z][0-9]{3}	1 uppercase letter followed by 3 digits
[0-9][0-9][0-9][a-zA-Z]*	3digits followed by any number of uppercase or lowercase letters
EMP[#_!]	'EMP' followed by 1 # or ! Or _

- The “Category” element is a simple type with a restriction.
- The acceptable values are Dot Net, BI, RDBMS, and J2EE

June 5, 2015

Proprietary and Confidential - 27 -

IGATE  
Speed Agility Imagination

### Restrictions on Series of Values

The above table shows how to give patterns in restrictions.  
The next example defines an element called “gender” with a restriction. The only acceptable value is male or female:

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The next example defines an element called “password” with a restriction. There must be exactly eight characters in a row and those characters must be lowercase or uppercase letters from a to z, or a number from 0 to 9:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

## 4.5: Indicator – Order, Occurrence, and Group

## Types of Indicators

➤ **We have seven types of indicators:**

- Order indicators:
  - All
  - Choice
  - Sequence
- Occurrence indicators:
  - maxOccurs
  - minOccurs
- Group indicators:
  - Group name
  - attributeGroup name

June 5, 2015

Proprietary and Confidential - 28 -

**IGATE**  
Speed. Agility. Imagination.

### Types of Indicators:

Indicators are specially used to control the occurrences of elements in different orders. Sometimes, we may want certain elements to occur only once, or certain elements may not be in a particular order, or certain elements may not be necessary at all (optional) and so on.

We can handle these kinds of issues by using **indicators**.

#### Order Indicators:

Order indicators are used to define how elements should occur.

## All Indicator

- The `<all>` indicator specifies, by default, that the child elements can appear in any order and that each child element must occur once and only once

```
<xs:element name="book">
  <xs:complexType>
    <xs:all>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

June 5, 2015

Proprietary and Confidential

- 29 -

**IGATE**  
Speed. Agility. Imagination.

### Indicator – Order, Occurrence, and Group:

In the example shown above, “book” element can have only “title & author” child elements which can occur in any sequence.

#### All Indicator:

When using the `<all>` indicator you can set the `<minOccurs>` indicator to 0 or 1 and the `<maxOccurs>` indicator can only be set to 1.



## Choice Indicator

- The `<choice>` indicator specifies that either one child element or another can occur

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

## Sequence Indicator

- The `<sequence>` indicator specifies that the child elements must appear in a specific order

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

## maxOccurs Indicator

- The `<maxOccurs>` indicator specifies the maximum number of times an element can occur:

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="vendor" type="xs:string" maxOccurs="2"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

June 5, 2015

Proprietary and Confidential - 32 -

**IGATE**  
Speed. Agility. Imagination.

### Indicator – Order, Occurrence, and Group:

#### Occurrence Indicators:

Occurrence indicators are used to define how often an element can occur.

**Note:** For all “Order” and “Group” indicators (any, all, choice, sequence, group name, and group reference), the default value for `maxOccurs` and `minOccurs` is 1.

#### `maxOccurs` Indicator:

The `<maxOccurs>` indicator specifies the maximum number of times an element can occur.

## minOccurs Indicator

- The `<minOccurs>` indicator specifies the minimum number of times an element can occur:

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="vendor" type="xs:string" minOccurs="2"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

June 5, 2015

Proprietary and Confidential - 33 -

**IGATE**  
Speed. Agility. Imagination.

### Indicator – Order, Occurrence, and Group:

#### minOccurs Indicator:

The example in the above slide indicates that the “vendor” element can occur a minimum of zero times and a maximum of two times in the “book” element.

**Tip:** To allow an element to appear for an unlimited number of times, use the `maxOccurs="unbounded"` statement.

## Group Indicators

➤ **Group Indicators:**

- Group indicators are used to define related sets of elements

➤ **Element Groups:**

- Element groups are defined with the group declaration, as shown below:

```
<xs:group name="groupname"> ... </xs:group>
```

### Indicator – Order, Occurrence, and Group:

#### Group Indicators:

You must define an all, choice, or sequence element inside the group declaration.

## Group Indicators (Contd)

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence> </xs:group>
```

```
<xs:element name="person" type="personinfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

IGATE  
Speed. Agility. Imagination.

June 5, 2015

Proprietary and Confidential - 35 -

### Group Indicators (Contd):

The example in the above slide defines a group named “persongroup”, that defines a group of elements that must occur in an exact sequence.

After you have defined a group, you can reference it in another group or complex type definition, as shown above.

## Demo on XML-Schema Definition

- **Demo on:**
- **Run Validator**
  - Code to be validated
  - Schema file
  - Shiporder.xsd (schema File)
  - Shiporder.xml (xml Document)



## Summary

➤ **In this lesson, you have learnt:**

- A schema describes the arrangement of markup and character data within a valid XML document
- The purpose of XML Schema and XML DTD is same
- Currently, only Microsoft IE5 supports XML Schema
- XML Schema vocabulary defines different elements





## Review Question

- **Question 1:** List any four valid datatypes in XML Schema:  
\_\_\_\_, \_\_\_\_, \_\_\_\_, and \_\_\_\_.
- **Question 2:** The elements defined in a schema come from this namespace:
  - Option 1 : sourceNamespace
  - Option 2: targetNamespace
  - Option 3: cannot be specified
- **Question 3:** Choice is an Occurrence indicator.
  - True/False



# Web Basics - XML

## Lab Book

---

Copyright © 2011 IGATE Corporation. All rights reserved. No part of this publication shall be reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior written permission of IGATE Corporation.

IGATE Corporation considers information included in this document to be Confidential and Proprietary.

Document Revision History

Date	Revision No.	Author	Summary of Changes
30-Sep-2009	0.1 D	Pradnya Jagtap	Content Creation
05-Oct-2009	0.1 D	CLS Team	Review
11-May-2010	2.0	Tushar Joshi	Revamp/Refinement
11-May-2010	2.0	Anu Mitra	Review
21-April-2011	3.0	Anu Mitra	Integration Changes
20-May-2013	4.0	Sathiabama R	Revamp/Refinement
21-Apr-2015	5.0	Rathnajothi P	Revamp/Refinement as per revised TOC

---

## Table of Contents

---

Document Revision History .....	2
Table of Contents .....	3
Getting Started .....	4
Overview .....	4
Setup Checklist for XML .....	4
Instructions .....	4
Lab 1. Introduction to XML .....	6
1.1: Writing Simple XML File .....	6
1.2: <<TODO>> Create XML files for storing Email data .....	7
1.3: <<TODO>> Create an XML file for storing employee details as given in table .....	7
1.4: Create an XML file to store employee details. ....	8
Lab 2. Writing XML Schemas (XSD) .....	9
2.1: Create a XML Schema definition document .....	9
2.2: Create XSD for the XML file created in Lab exercise 1.2 and 1.4 .....	12
2.3: Create XSD for the given book details in "Library.xml" file .....	12
Appendices .....	14
Appendix A: .....	14
Appendix B: .....	23
Appendix C: Table of Figures .....	28
Appendix D: Table of Examples .....	29

---

## Getting Started

---

### Overview

This Labbook will guide you through XML. It will help you learn and write your own XML Documents. The Labbook contains solved examples and also the **To Do** assignments. Follow the steps to go through the solved examples and then work out on **To Do** assignments.

### Setup Checklist for XML

Here is what is expected on your machine in order for the lab to work.

#### Minimum System Requirements

- Hardware: Networked PCs with minimum 64 MB RAM and 60 MB HDD.
- Software:
  - Window based Operating System having the latest version of Internet Explorer (IE) or Netscape Navigator installed.
  - Eclipse Luna or Visual Studio 2008

### Instructions

#### Steps to write a XML document in eclipse:

**Step 1:** Run **eclipse.exe** file.

**Step 2:** For writing the code for XML first create the Project.

Go to **File** → **New** → **Project** → **General-Project** → **Project** → Click **Next** button → Give the project name → Click **Finish**.

**Step 3:** Right click Project and go to **New** → **Other** → **XML** → **XML File** → **Next** button → Write the file name → click **Next** button → select one of the options → Click **Finish**.

- a. If you want to create xml file from existing DTD, then select the first option.
- b. If want to create from existing schemas then select second option.
- c. If you want a simple one, then you can select third option.

**Step 4:** Write the code for XML

**Step 5:** Save your file with a suitable filename and .xml extension

**Steps to View the output of XML Document:**

**Step 1:** Right click XML file. Select **Open With** → **Other** → it will prompt a window → select **Internal Web Browser** → click **OK**.

**Step 2:** Your output is visible in the browser window.

**Note:**

Create a directory by your name in drive <drive>. In this directory create subdirectory XML\_Assgn.

**For example:** d:\5001\XML\_Assgn\, use this as a workspace for eclipse project. Create separate projects for each lab within the workspace.

**Refer to Appendix A for Steps to create an XML file in an XML document editor**

---

## Lab 1. Introduction to XML

---

Goals	<ul style="list-style-type: none"><li>Understanding basics and start of with XML</li></ul>
Time	60 minutes

### 1.1: Writing Simple XML File

#### Solution:

**Step 1:** Write the code as follows by using the Eclipse\VS2008 editor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>two of our famous Belgian Waffles with plenty of real maple
syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>light Belgian waffles covered with strawberries and whipped
cream</description>
    <calories>900</calories>
  </food>
  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>light Belgian waffles covered with an assortment of fresh
berries and whipped cream</description>
    <calories>900</calories>
  </food>
</breakfast_menu>
```

Example 1: foodmenu.xml file

**Step 2:** Save this file as foodmenu.xml.

**Step 3:** View the output in Browser.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Edited by Tushar@ -->
-<breakfast_menu>
-  <food>
-    <name>Belgian Waffles</name>
-    <price>$5.95</price>
-    <description>two of our famous Belgian Waffles with plenty of real maple syrup</description>
-    <calories>650</calories>
-  </food>
-  <food>
-    <name>Strawberry Belgian Waffles</name>
-    <price>$7.95</price>
-    <description>light Belgian waffles covered with strawberries and whipped cream</description>
-    <calories>900</calories>
-  </food>
-  <food>
-    <name>Berry-Berry Belgian Waffles</name>
-    <price>$8.95</price>
-    <description>light Belgian waffles covered with an assortment of fresh berries and whipped cream</description>
-    <calories>900</calories>
-  </food>
-</breakfast_menu>

```

Figure 1: Output of foodmenu.xml

### 1.2: <<TODO>> Create XML files for storing Email data

Data need to be stored are:

- To
- From
- CC
- BCC
- Subject
- Body

**Solution:**

**Step 1:** Write your code using Visual Web Developer 2008\Eclipse.

**Step 2:** Save this file as **email.xml**.

**Step 3:** View the output in Internet Explorer

### 1.3: <<TODO>> Create an XML file for storing employee details as given in table

The following table describes the employee details.

The employee details should be stored inside the root element “EmployeeDetails”.

Employee code	Employee Name		DOJ	Total Experience	Department		Designation					Sal	Grade			
	First Name	Last name			Unit	loc	1	2	3	4	5		a	b	c	d
1111	Seema	Dalvi	09/08/97	12	3d	Mumbai				*		21000	*			
222	Bharati	Mantri	12/01/99	5	4a	Pune		*				12000	*			



**1.4: Create an XML file to store employee details.**

Payroll department has decided to use XML files as a rich data source, so the existing EMP table has to be converted into XML data source i.e. XML file need to be created for the below EMP table representing entire data (To Do)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	1100	200	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	24820	30
7521	WARD	SALESMAN	7698	22-FEB-81	1375	19700	30
7566	JONES	MANAGER	7839	02-APR-81	3175		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1375	20600	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	2000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1650	23000	30
876	ADAMS	CLERK	7788	23-MAY-87	1430	200	20
7900	JAMES	CLERK	7698	03-DEC-81	1045	200	30
7902	FORD	ANALYST	7566	03-DEC-81	3200		20
7934	MILLER	CLERK	7782	23-JAN-82	1430	200	10

## Lab 2. Writing XML Schemas (XSD)

<b>Goals</b>	<ul style="list-style-type: none"> <li>Write XML Schema Definition (XSD).</li> </ul>
<b>Time</b>	180 minutes

### 2.1: Create a XML Schema definition document

Create a XML Schema definition document. We will then write a XML Document based on this XSD. Validate your XML against the XSD. (Refer Appendix B)

#### Solution:

**Step 1:** Write the following code and save it as “**Shiporder.xsd**”.

The Schema document looks like a typical XML document. The output can be seen in Internet Explorer.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definition of simple elements -->
  <xs:element name="orderperson" type="xs:string"/>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="address" type="xs:string"/>
  <xs:element name="city" type="xs:string"/>
  <xs:element name="country" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="note" type="xs:string"/>
  <xs:element name="quantity" type="xs:positiveInteger"/>
  <xs:element name="price" type="xs:decimal"/>
  <!-- definition of attributes -->
  <xs:attribute name="orderid" type="xs:string"/>
  <!-- definition of complex elements -->
  <xs:element name="shipto">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="address"/>
        <xs:element ref="city"/>
        <xs:element ref="country"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
```

```

        <xs:element ref="title"/>
        <xs:element ref="note" minOccurs="0"/>
        <xs:element ref="quantity"/>
        <xs:element ref="price"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="shiporder">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="orderperson"/>
            <xs:element ref="shipto"/>
            <xs:element ref="item" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="orderid" use="required"/>
    </xs:complexType>
</xs:element>
</xs:schema>

```

Example 2: shiporder.xsd

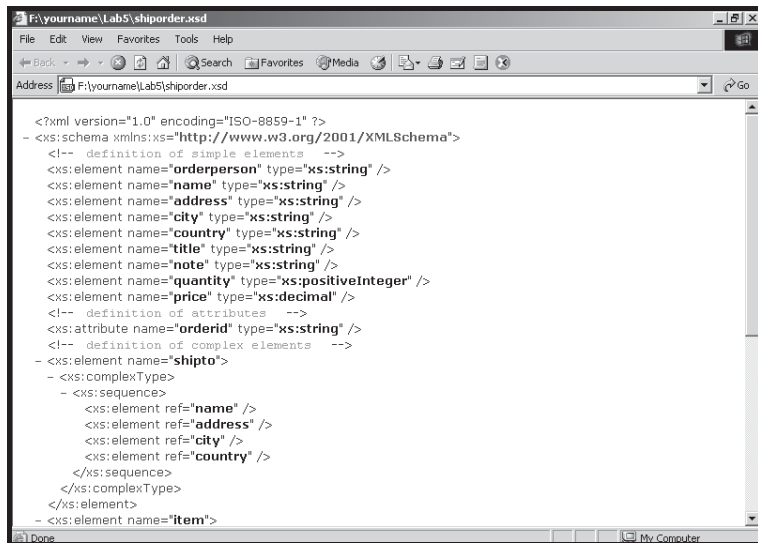


Figure 2: Output of Shiporder.xsd

**Step 2:** Now write an XML Document using the above XML Schema Definition.

Save this file as “**Shiporder.xml**”. View the output in Internet Explorer.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

**Example 3: Shiporder.xml**

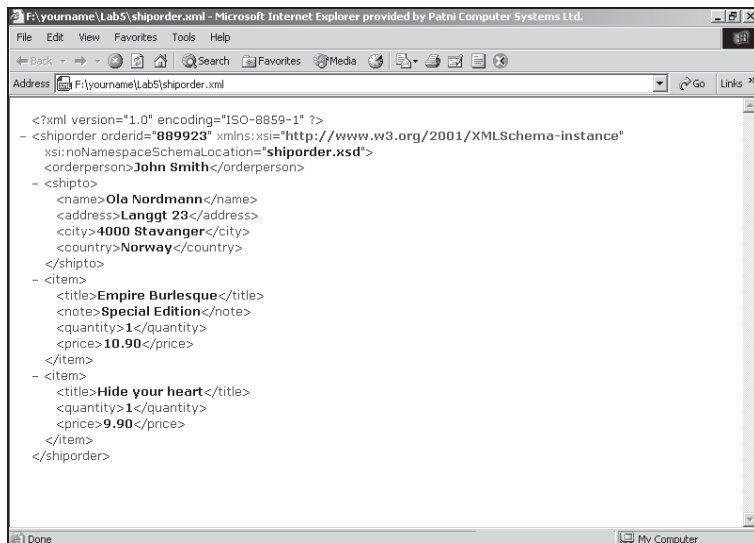


Figure 3: Output of Shiporder.xml

2.2: Create XSD for the XML file created in Lab exercise 1.2 and 1.4.

2.3: Create XSD for the given book details in “Library.xml” file.

Consider the following data exists in “Library.xml” file and create XSD for the same.

```
<Library>
  <Book>
    <Title>Dead to the World </Title>
    <Author>Charlaine Harris </Author>
    <Publisher>ABC Publishing company</Publisher>
    <Cover cover_type="Hardbook"></Cover>
    <Category cat_type="Horror"></Category>
    <ISBN isbn_num="Z1"></ISBN>
    <Rating rate_Val="5"></Rating>
    <Price>$13.97 </Price>
    <Comments>A good book</Comments>
  </Book>
  <Book>

    <Title>Gardens of the Moon </Title>
    <Author>Steven Erikson</Author>
```

```
<Publisher>XYZ Publishers</Publisher>
<Cover cover_type="Hardbook"></Cover>
<Category cat_type="SCI-FI"></Category>
<ISBN isbn_num="Z3"></ISBN>
<Rating rate_Val="4"></Rating>
<Price>$17.65</Price>
<Comments>A wonderful book</Comments>
</Book>
</Library>
```

**Example 4: Library.xml**

---

## Appendices

---

### Appendix A:

#### XML Editors:

1. **Eclipse Luna:** XML editor comes with a full range of features you would expect from an eclipse editor. It includes IntelliSense, color-coding, brace matching, and formatting.

You can download **eclipse** from the following url:

<http://www.eclipse.org/downloads/>

- a. When you click on eclipse.exe file, the screen for Creating new XML file in eclipse is displayed as follows:

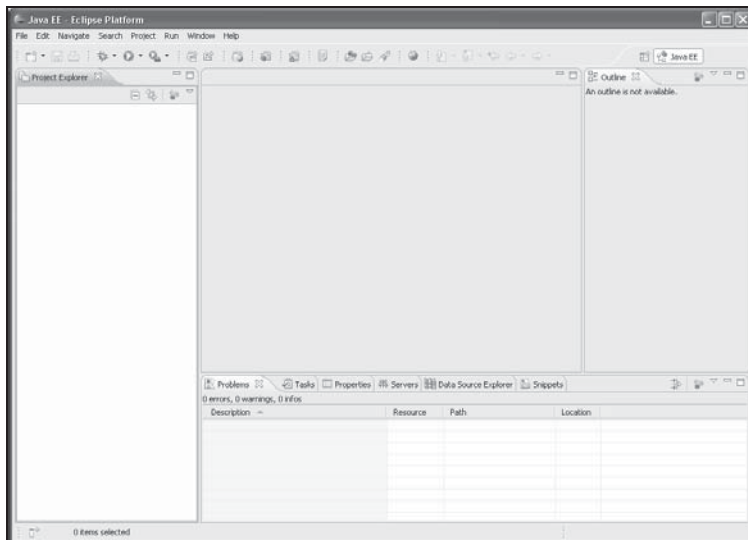


Figure 4: Opening eclipse Luna

b. For writing XML file, we need to create **General Project** as follows:

(i) Select **New → Project**.

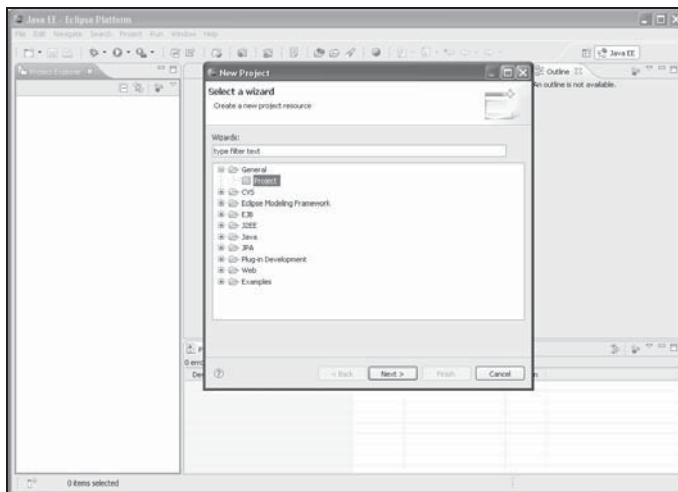
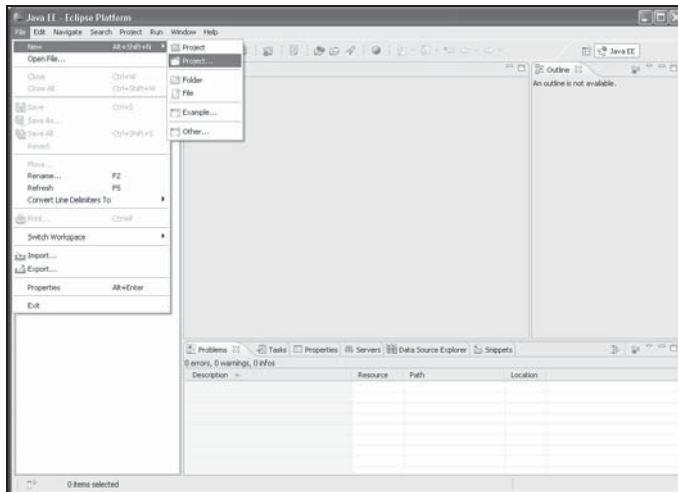
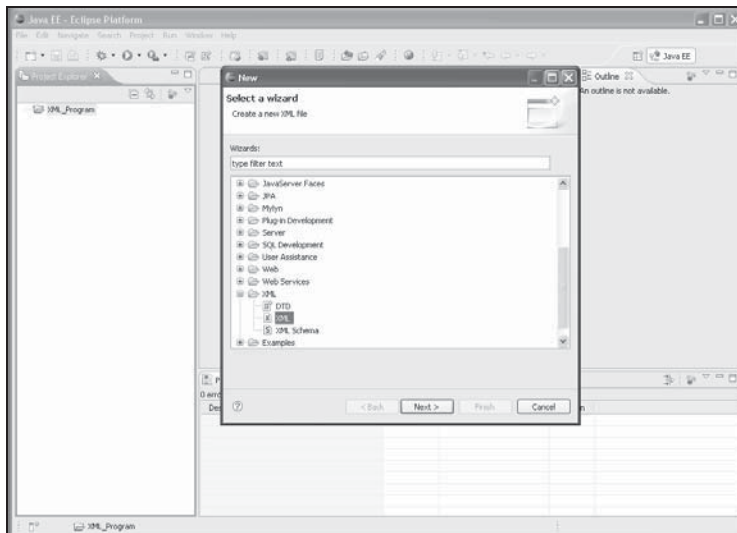
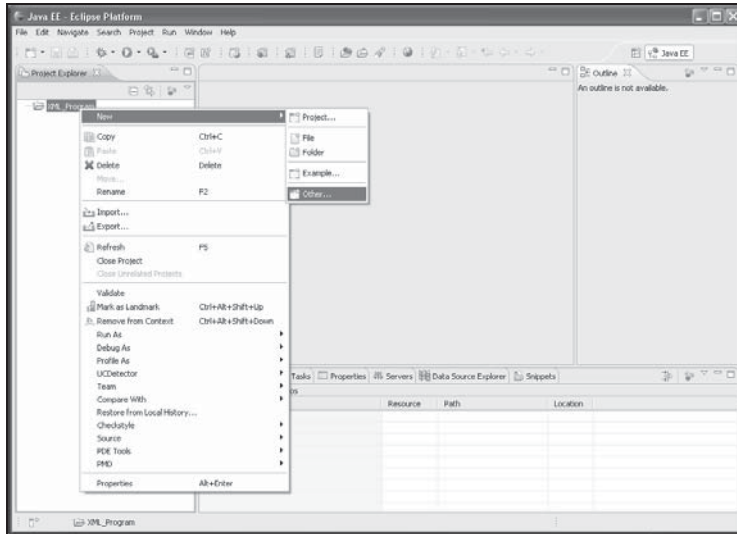


Figure 5: Creating General Project

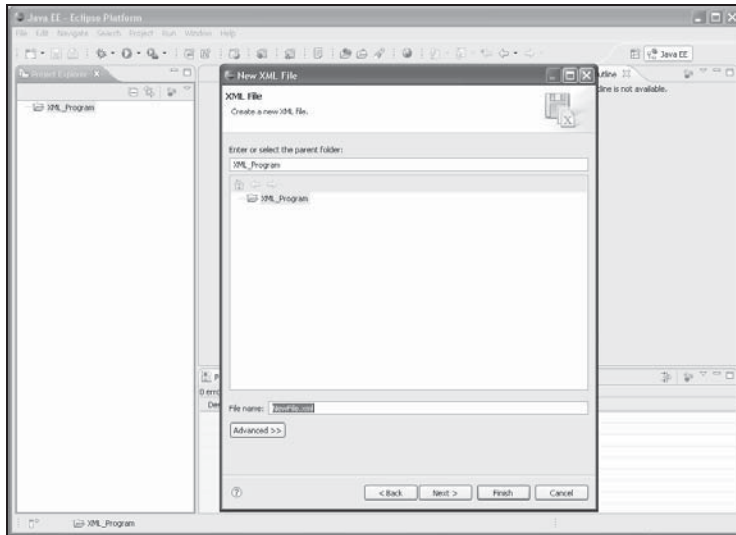
(ii) Then click **Next** button. Type the project name and click **Finish**.



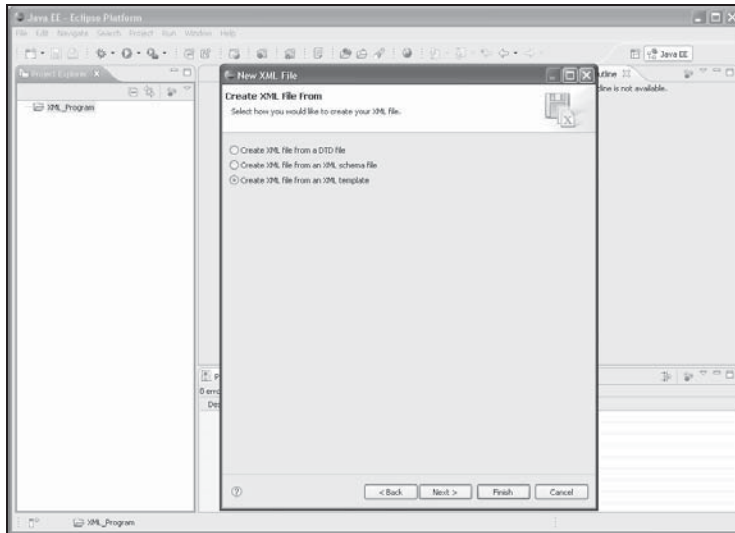
c. Then for creating new XML file, follow the steps shown in the screenshots:



(i) Select XML file.



(ii) Give the file name, and click **Next** button.



**Fig 4 : Creating XML file**

(iii) Select any one of the option as per the requirement. Then click **Finish** button.

e. For creating Schemas, follow the steps given below:

(i) Go to **File → New → Other → XML → XML Schema → Next**

(ii) Write the file name, and click **Finish**.

f. For viewing the output, follow the steps given below:

(i) Right click **File**, and select **Open With → Other → select Internet Web Browser → OK**.

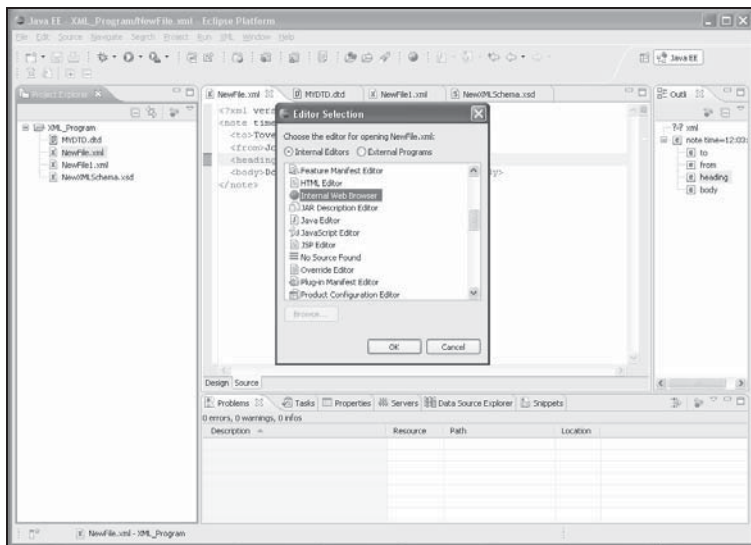
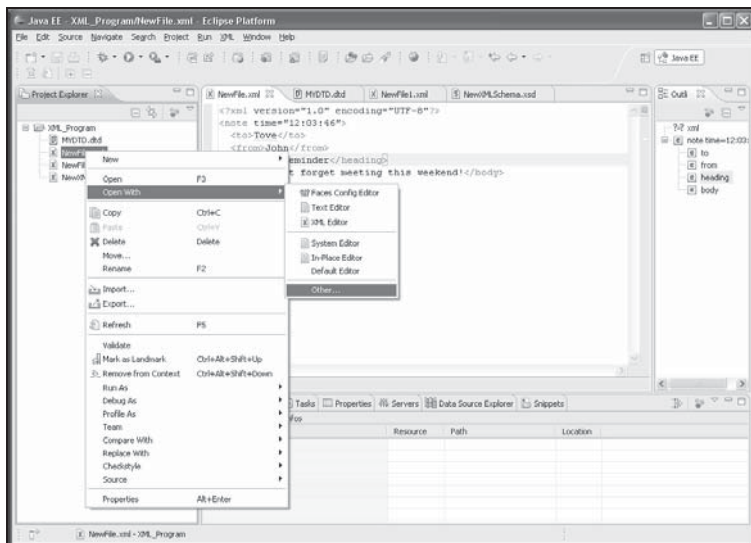
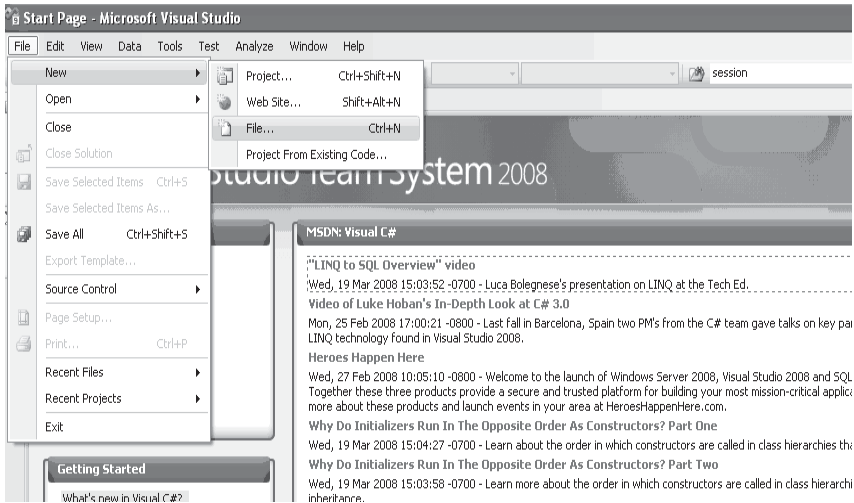


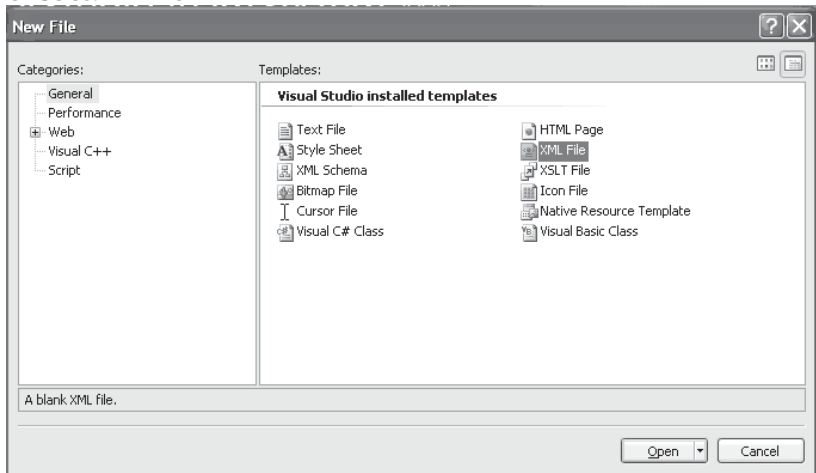
Figure 6: For viewing output

## 2. Creating XML file using Visual Studio 2008

- A. Go to Visual Studio Team System 2008
- B. Go to File->New->File take new XML file

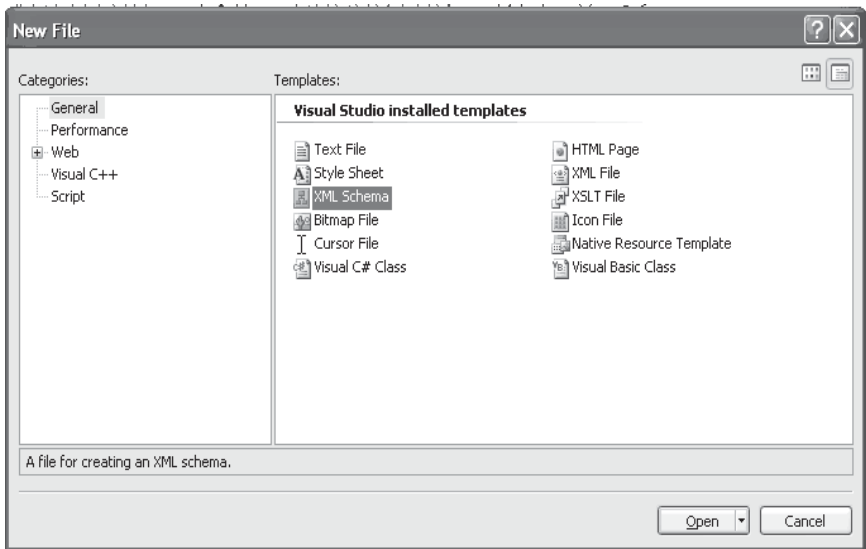


### C. Select XML File



### 3. Creating Schema file using Visual Studio Team System 2008

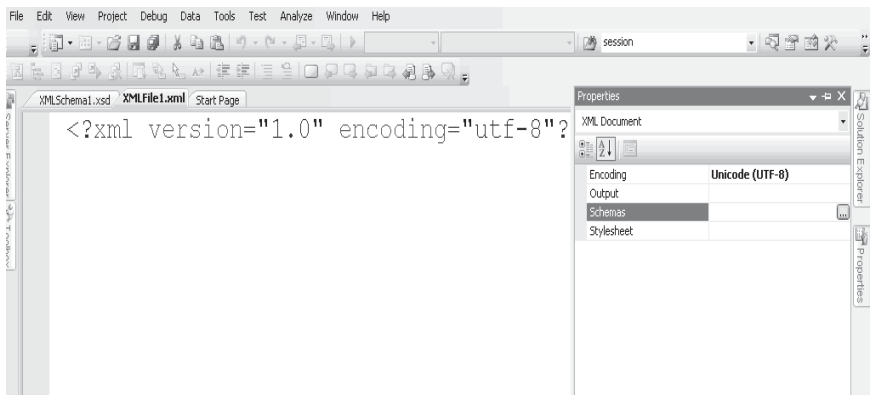
- A. Go to Visual Studio Team System 2008
- B. Go to File->New->File take new XML Schema



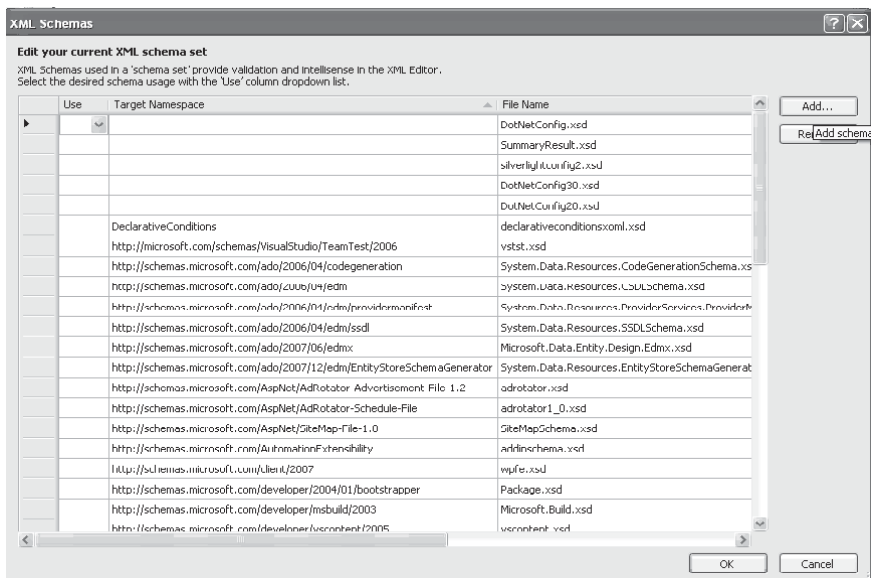
#### 4. Attaching Schema file to XML file for validation

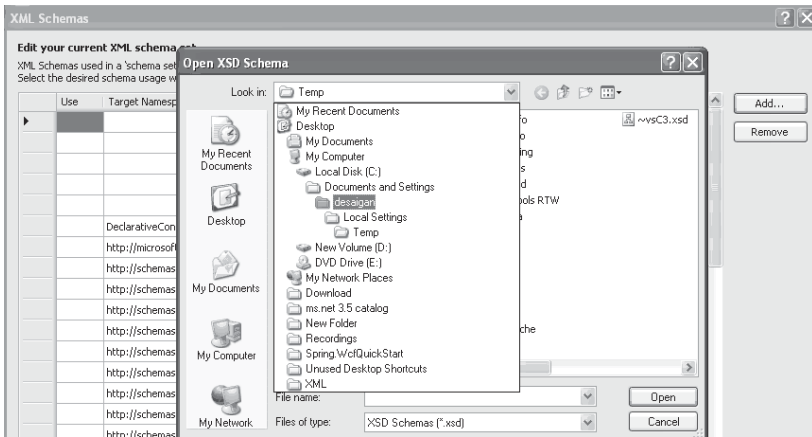
Follow the steps given in Appendix A.2 to create XML file then follow the following steps to attach schema file.

- A. Go to properties window (press F4) and Select Schema tab.



B. Click on Add & browse your Schema file -> Click on OK





C. Validating your XML file against Schema file will be taken care by Editor itself & necessary errors will be shown.

## Appendix B:

1. Steps for creating XML file from the existing DTD and Schema.
  - a. Go to File → New → Other → XML → DTD.
  - b. Write the file name, and click Next button.
  - c. Select “Create XML file from a DTD file” option.



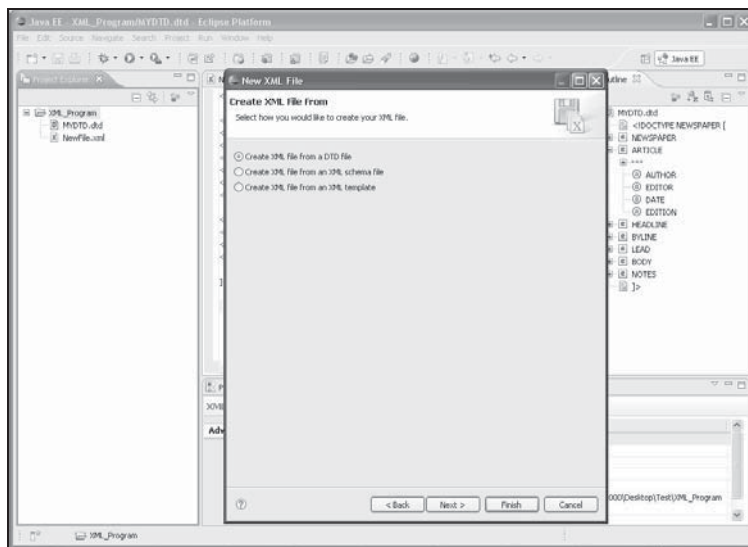


Figure 7: For creating XML file from a DTD file

- d. Click **Next** button, select **DTD file name**, and click **Next**.

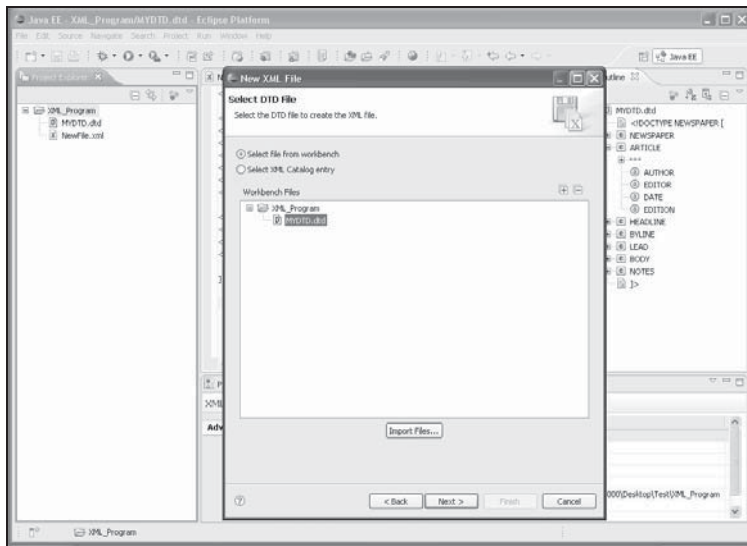


Figure 8: Selection of a DTD file

- e. Now click **Next**, select the root element, and click **Finish**.

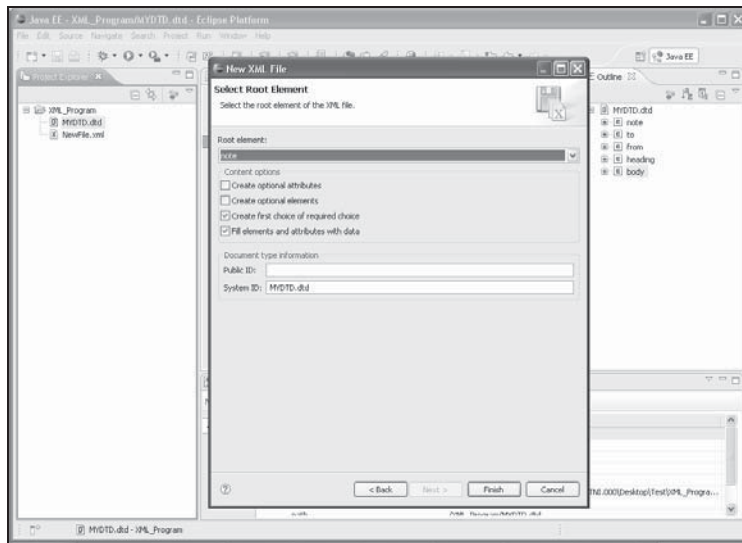


Figure 9: Select Root element for a XML file

- f. For creating XML file from existing schema file, you should follow the same steps, except select “**Create XML file from an XML Schema**” option.

## 2. Validating a XML document against a DTD or Schemas:

When we create a XML document, we need to check for it “Validity” and its “Well Formedness”. Eclipse allows us to validate the of the XML document against a DTD or Schema.

Right click the **XML file**, and select **Validate** option. If there is any problem, then you can view it in **Problem** tab.

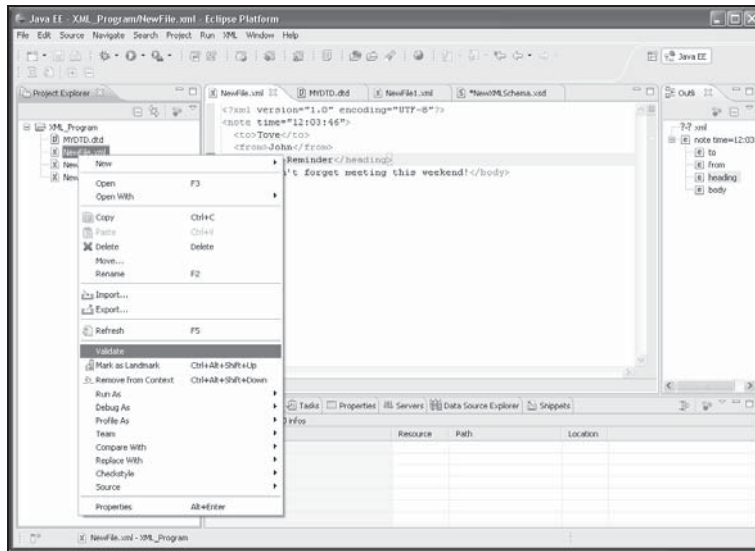


Figure 10: For Validating a XML file

---

**Appendix C: Table of Figures**

Figure 1: Output of foodmenu.xml .....	7
Figure 2: Output of Shiporder.xsd .....	10
Figure 3: Output of Shiporder.xml .....	12
Figure 5: Opening eclipse Luna .....	14
Figure 6: Creating General Project .....	15
Figure 8: For viewing output .....	19
Figure 9: For creating XML file from a DTD file .....	24
Figure 10: Selection of a DTD file .....	25
Figure 11: Select Root element for a XML file .....	26
Figure 12: For Validating a XML file .....	27

---

**Appendix D: Table of Examples**

Example 1: foodmenu.xml file.....	6
Example 2: shiporder.xsd.....	10
Example 3: Shiporder.xml.....	11
Example 4: Library.xml .....	13