

# **Web Basics - XML**

## **Lab Book**

## 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
May - 2016	5.1	Anjulata	Refinement

## Table of Contents

<b>Getting Started.....</b>	<b>4</b>
Overview.....	4
Setup Checklist for XML.....	4
Instructions .....	4
<b>Lab 1: Introduction to XML.....</b>	<b>7</b>
1.1: Writing Simple XML File .....	7
1.2: <<TODO>> Create XML files for storing Email data.....	9
1.3: <<TODO>> Create an XML file for storing employee details as given in table .....	9
1.4: Create an XML file to store employee details. ....	10
<b>Lab 2: Writing XML Schemas (XSD).....</b>	<b>12</b>
2.1: Create a XML Schema definition document.....	12
2.2: Create XSD for the XML file created in Lab exercise 1.2 and 1.4. ....	17
2.3: Create XSD for the given book details in “Library.xml” file.....	17
<b>Appendices .....</b>	<b>19</b>
Appendix A: .....	19
Appendix B: .....	36
Appendix C: Table of Figures.....	40
Appendix D: Table of Examples .....	41

## 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**

<b>Goals</b>	<ul style="list-style-type: none"><li>Understanding basics and start of with XML</li></ul>
<b>Time</b>	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**”.

Emplo ye e code	Employee Name		DOJ	Total Experi ence	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>

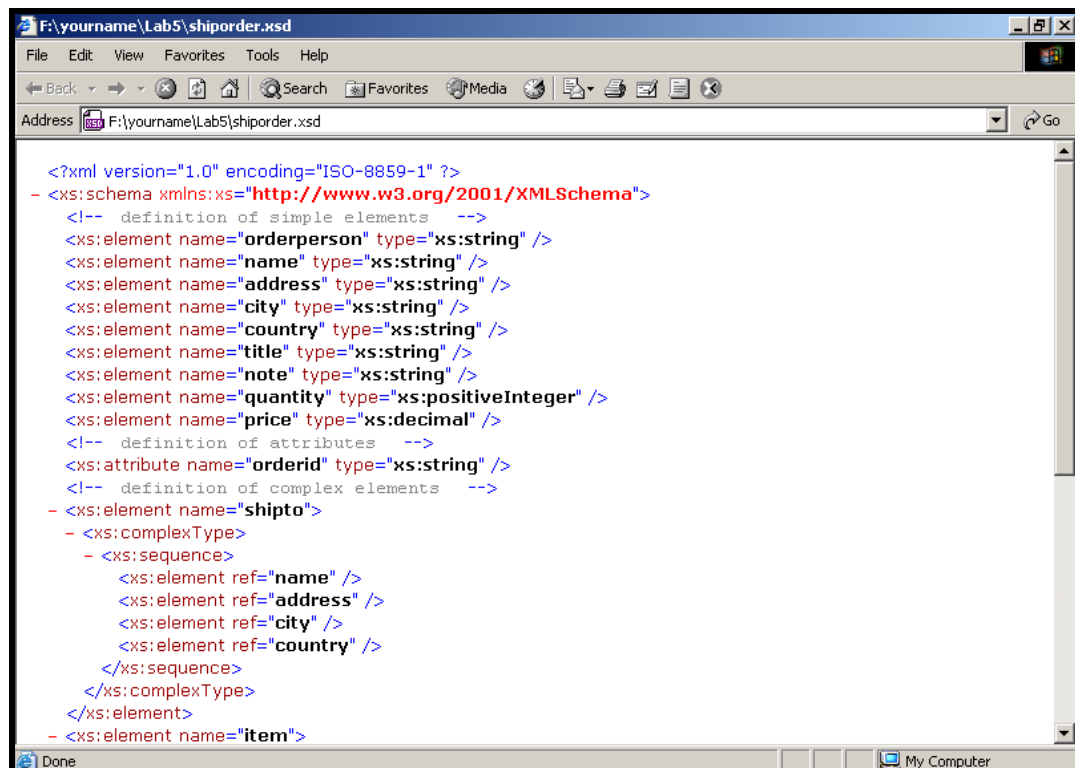
<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
```

```
      <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>

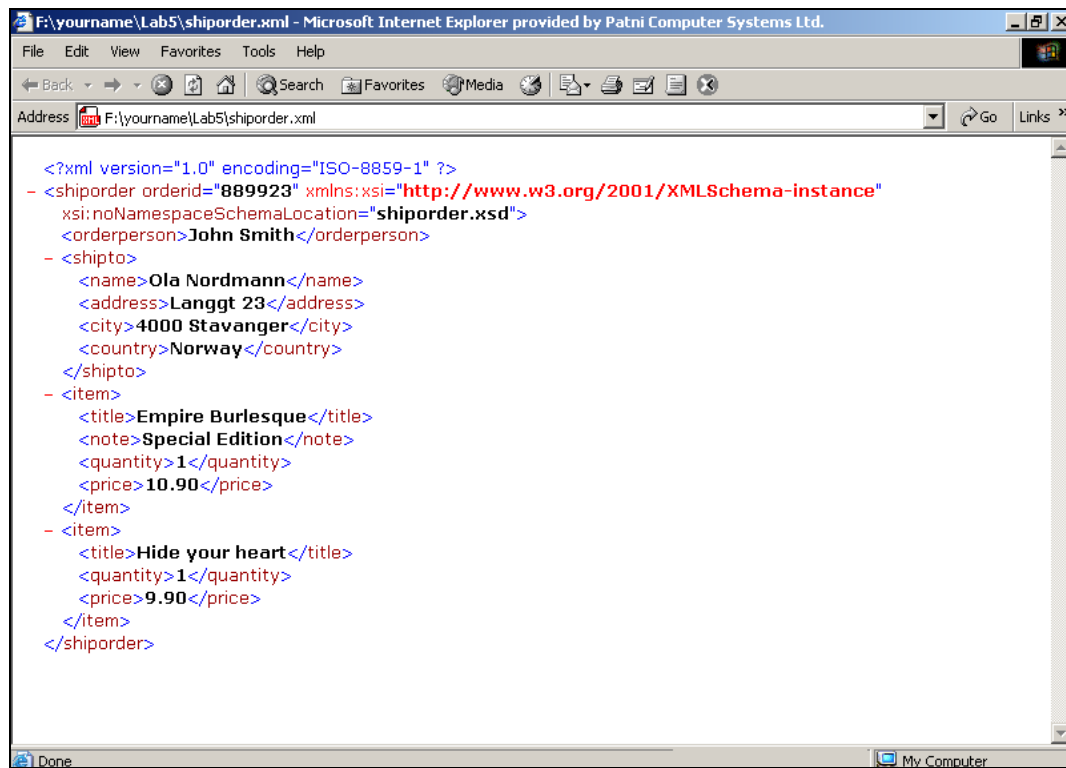
    <name>Ola Nordmann</name>

    <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**

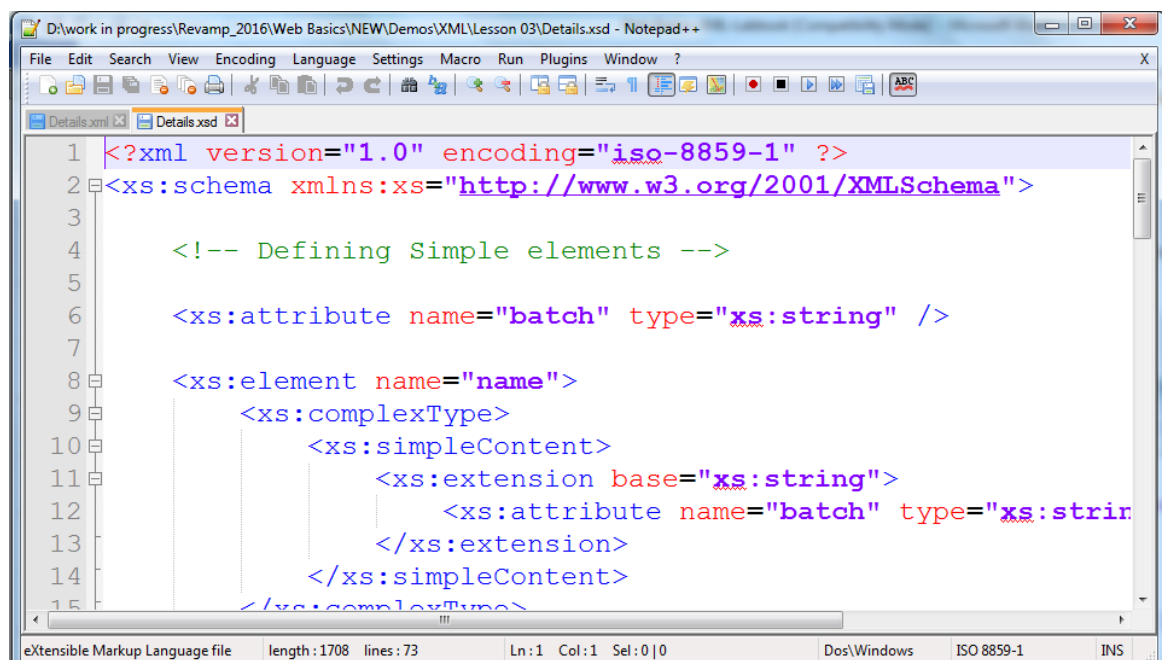
## Appendices

### Appendix A:

#### XML Editors:

1. Notepad++: Notepad++ with XML Tools plugin can be used for validating the XML against the xsd. XML Tools plugin can be downloaded Plugins -> Plugin Manager -> Show Plugin Manager in Notepad++.

Xsd file can be created in Notepad++ as follows:



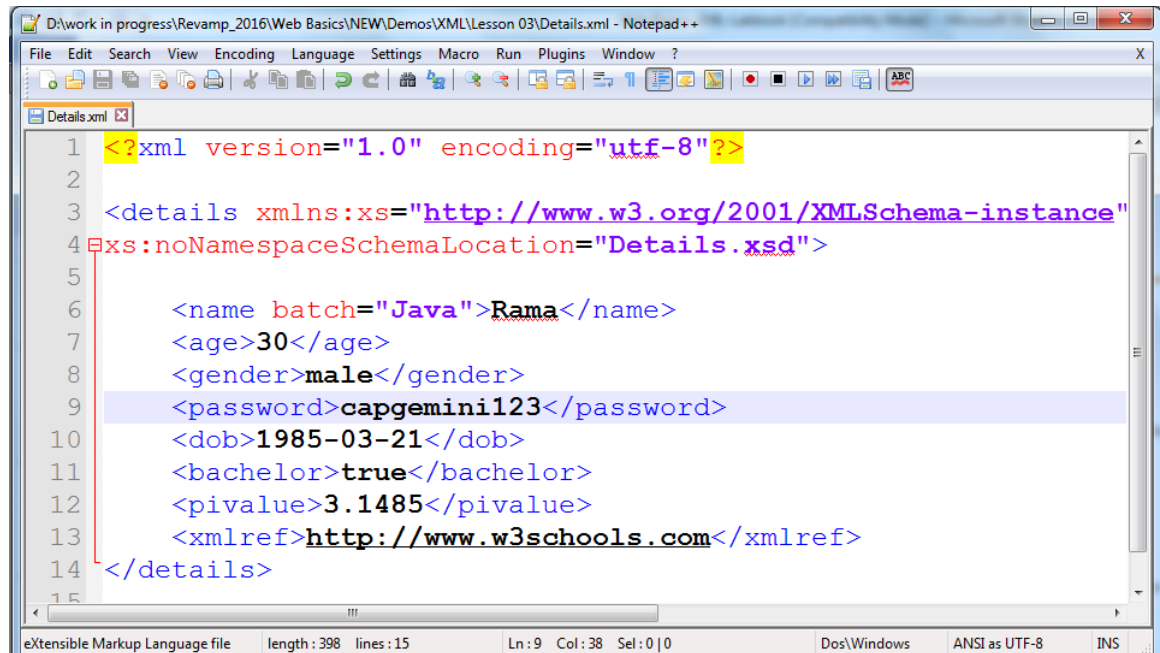
```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Defining Simple elements -->

    <xs:attribute name="batch" type="xs:string" />

    <xs:element name="name">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="batch" type="xs:string" />
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Xml file can be created in Notepad++ as follows:



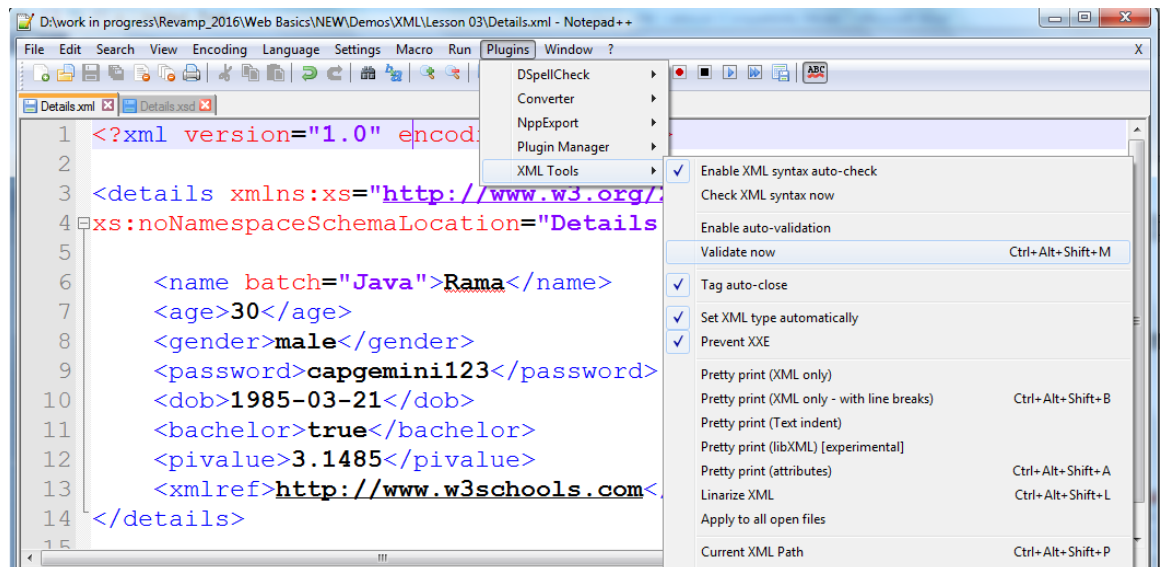
```

1 <?xml version="1.0" encoding="utf-8"?>
2
3 <details xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
4 xs:noNamespaceSchemaLocation="Details.xsd">
5
6     <name batch="Java">Rama</name>
7     <age>30</age>
8     <gender>male</gender>
9     <password>capgemini123</password>
10    <dob>1985-03-21</dob>
11    <bachelor>true</bachelor>
12    <pivalue>3.1485</pivalue>
13    <xmlref>http://www.w3schools.com</xmlref>
14 </details>
15

```

Extensible Markup Language file    length: 398    lines: 15    Ln: 9    Col: 38    Sel: 0 | 0    Dos\Windows    ANSI as UTF-8    INS

Validation of xml can be done as follows:



```

1 <?xml version="1.0" encoding="utf-8"?>
2
3 <details xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
4 xs:noNamespaceSchemaLocation="Details.xsd">
5
6     <name batch="Java">Rama</name>
7     <age>30</age>
8     <gender>male</gender>
9     <password>capgemini123</password>
10    <dob>1985-03-21</dob>
11    <bachelor>true</bachelor>
12    <pivalue>3.1485</pivalue>
13    <xmlref>http://www.w3schools.com</xmlref>
14 </details>
15

```

XML Tools

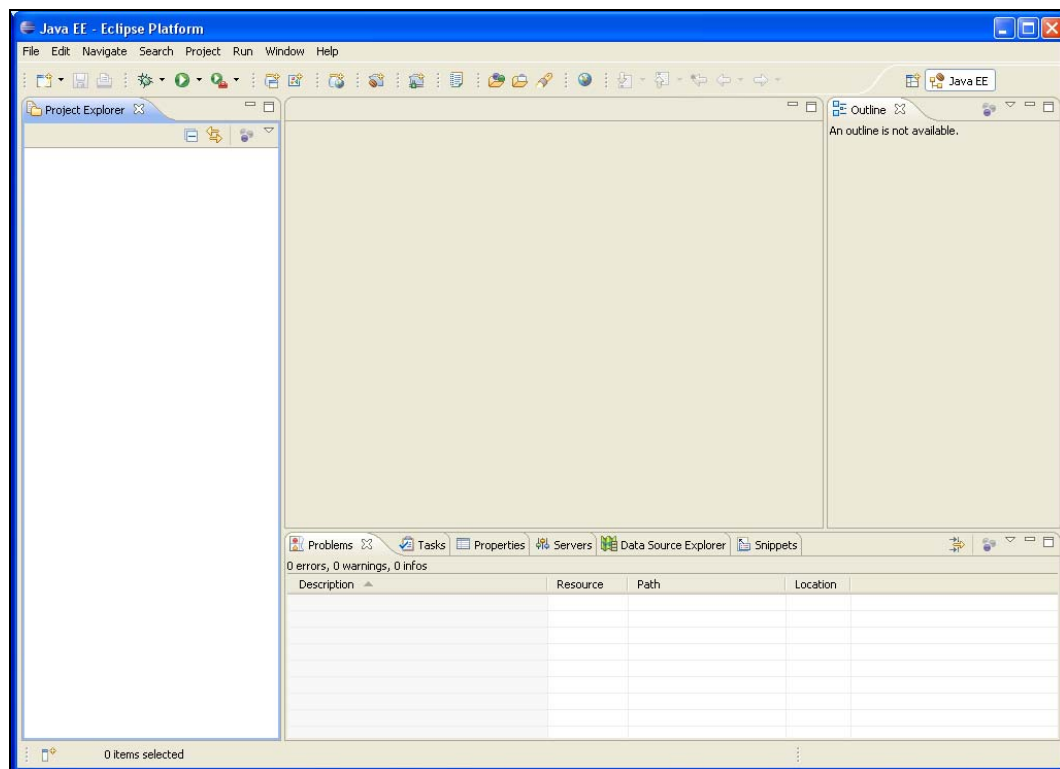
- ☒ Enable XML syntax auto-check
- Check XML syntax now
- Enable auto-validation
- Validate now    Ctrl+Alt+Shift+M
- ☒ Tag auto-close
- ☒ Set XML type automatically
- ☒ Prevent XXE
- Pretty print (XML only)
- Pretty print (XML only - with line breaks)    Ctrl+Alt+Shift+B
- Pretty print (Text indent)
- Pretty print (libXML) [experimental]
- Pretty print (attributes)    Ctrl+Alt+Shift+A
- Linarize XML    Ctrl+Alt+Shift+L
- Apply to all open files
- Current XML Path    Ctrl+Alt+Shift+P

2. **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:

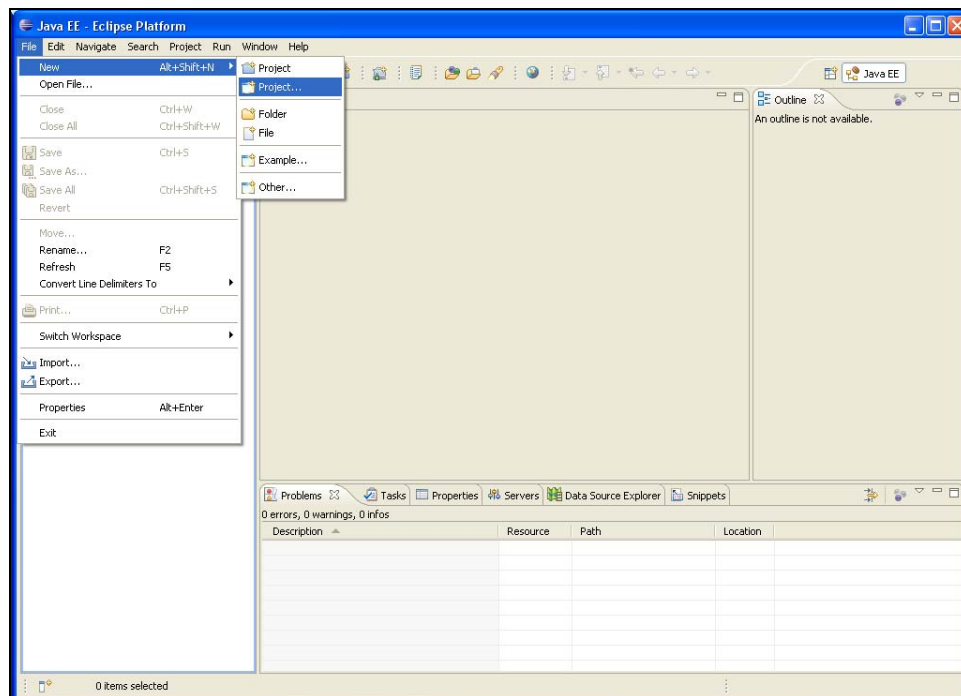
<http://www.eclipse.org/downloads/>

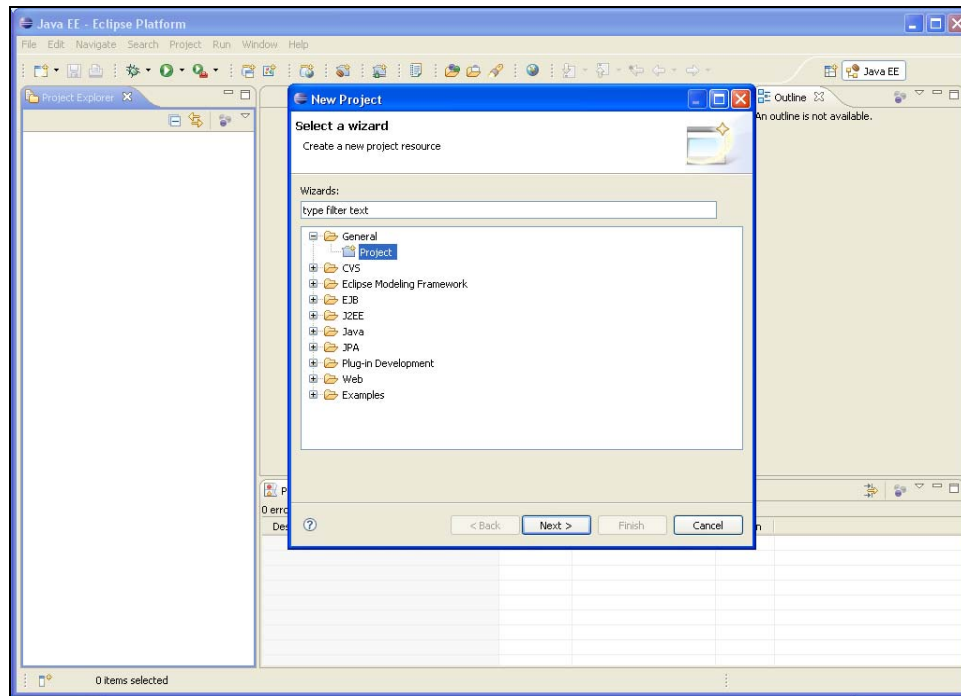
- a. When you click on eclipse.exe file, the screen for Creating new XML file in eclipse is displayed as follows:



**Figure 4: Opening eclipse Luna**

- b. For writing XML file, we need to create General Project as follows:
- (i) Select New → Project.

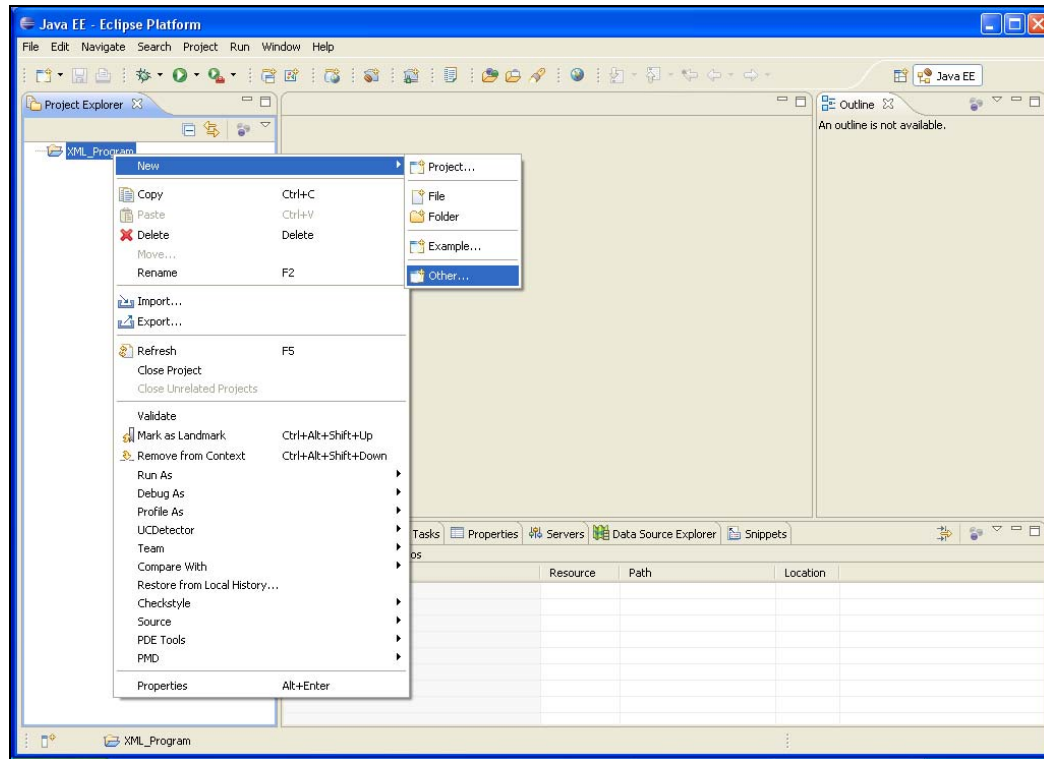




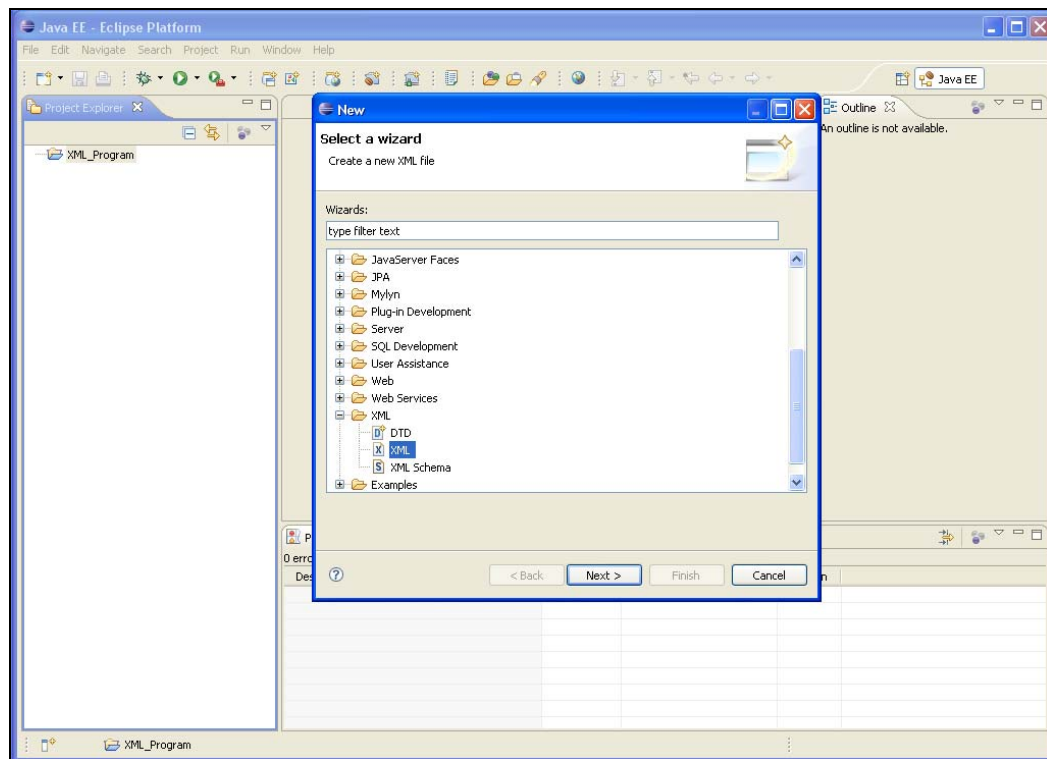
**Figure 5: Creating General Project**

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

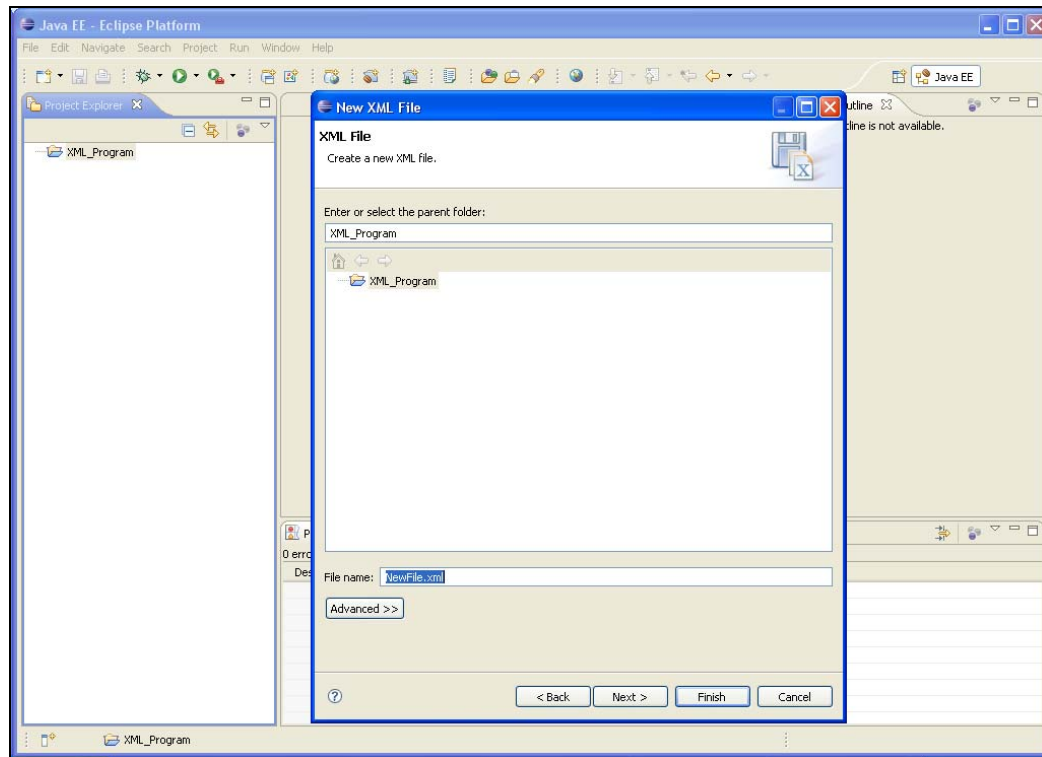
- c. Then for creating new XML file, follow the steps shown in the screenshots:



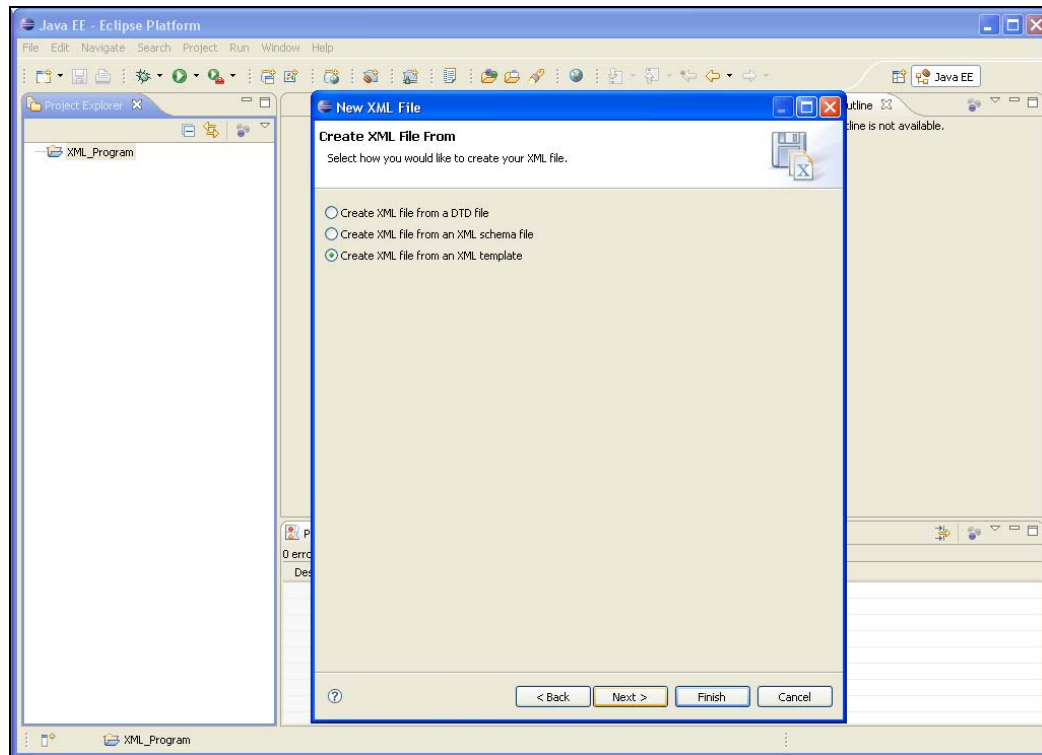




(i) Select **XML** file.

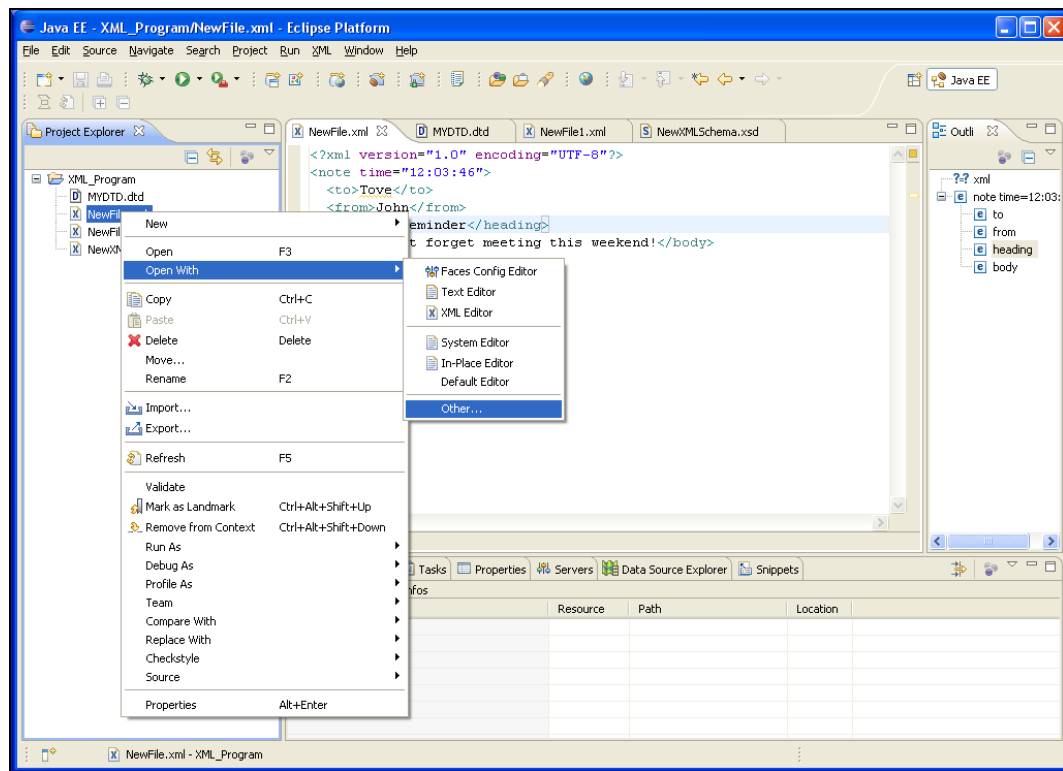


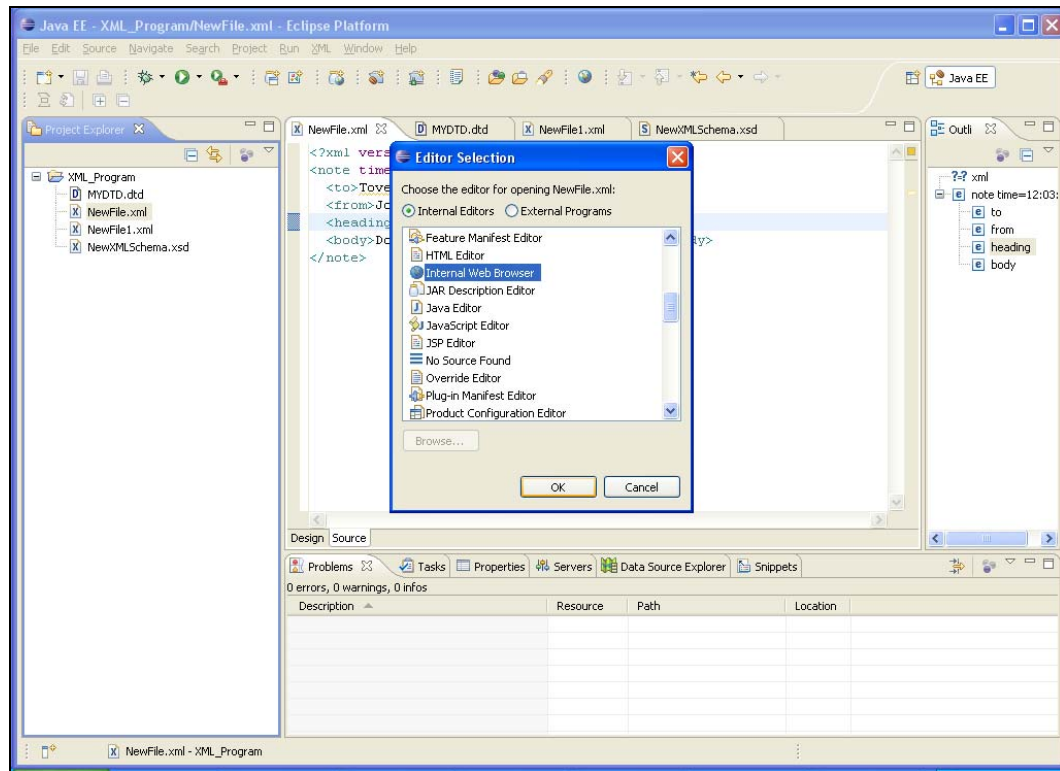
- (ii) Give the file name, and click **Next** button.



**Fig 4 : Creating XML file**

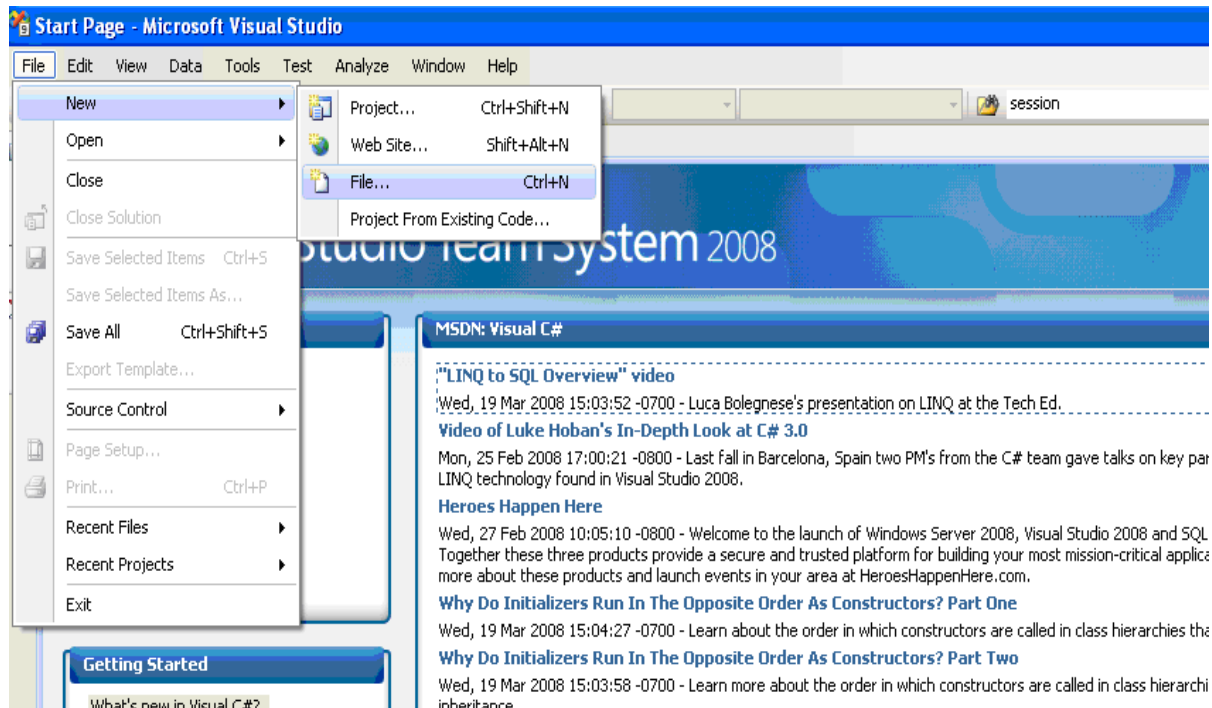
- (iii) Select any one of the option as per the requirement. Then click **Finish** button.
- d. For creating Schemas, follow the steps given below:
- (i) Go to **File → New → Other → XML → XML Schema → Next**
  - (ii) Write the file name, and click **Finish**.
- e. For viewing the output, follow the steps given below:
- (i) Right click **File**, and select **Open With → Other → select Internet Web Browser → OK**.



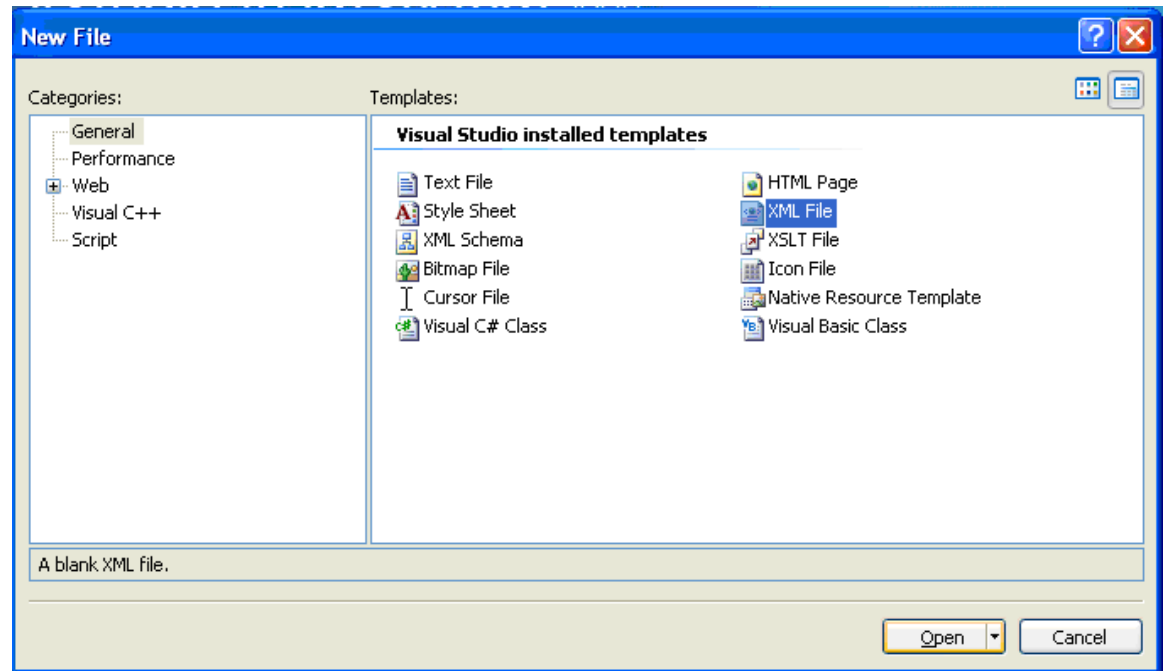


**Figure 6: For viewing output**

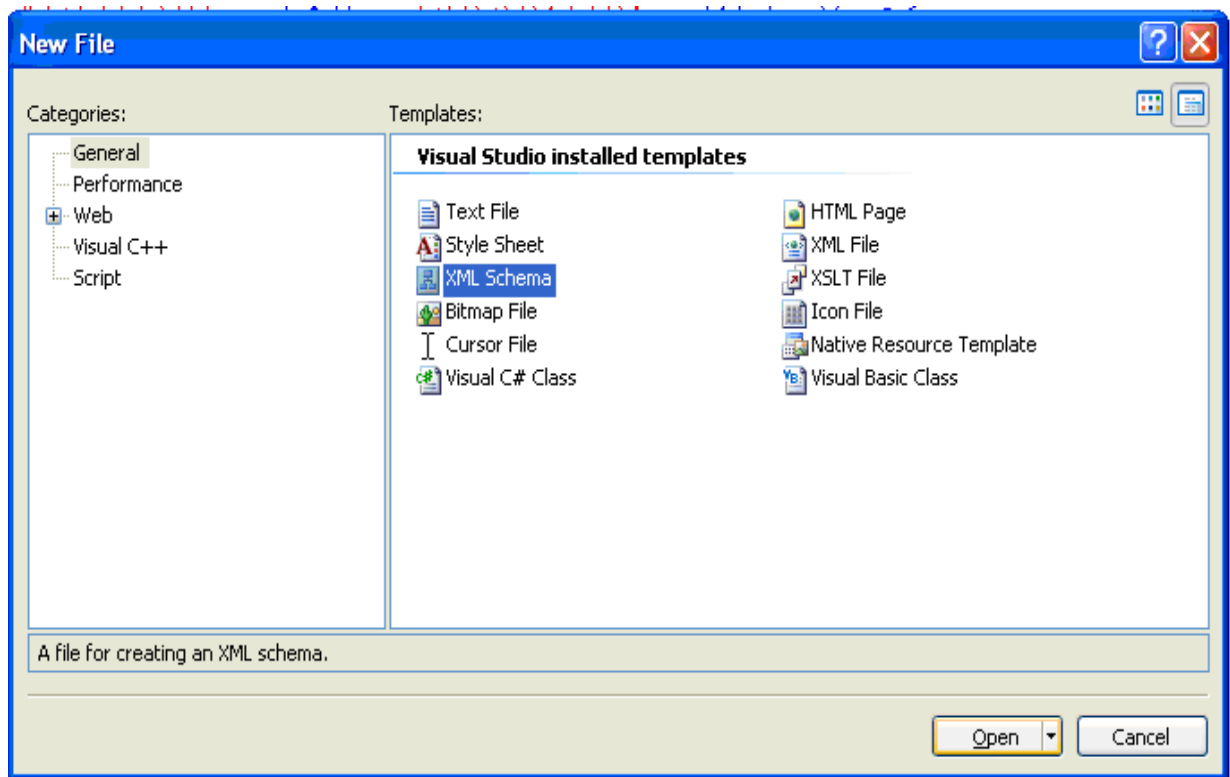
3. Visual Studio 2008
  - a. Go to Visual Studio Team System 2008
  - b. Go to File->New->File take new XML file



c. Select XML File



4. 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

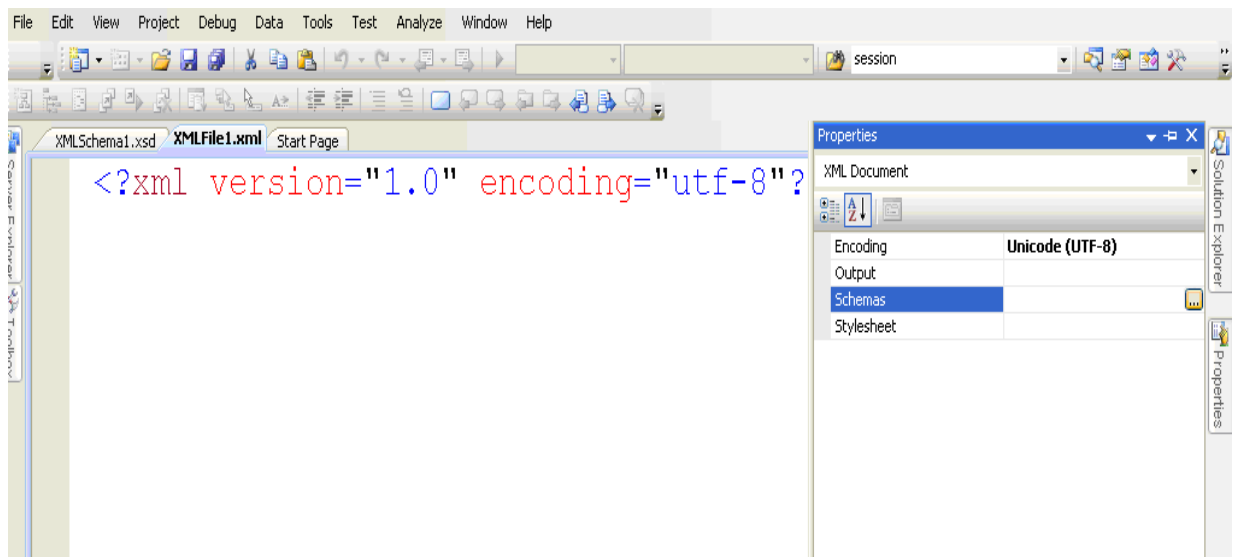


5. Attaching Schema file to XML file for validation

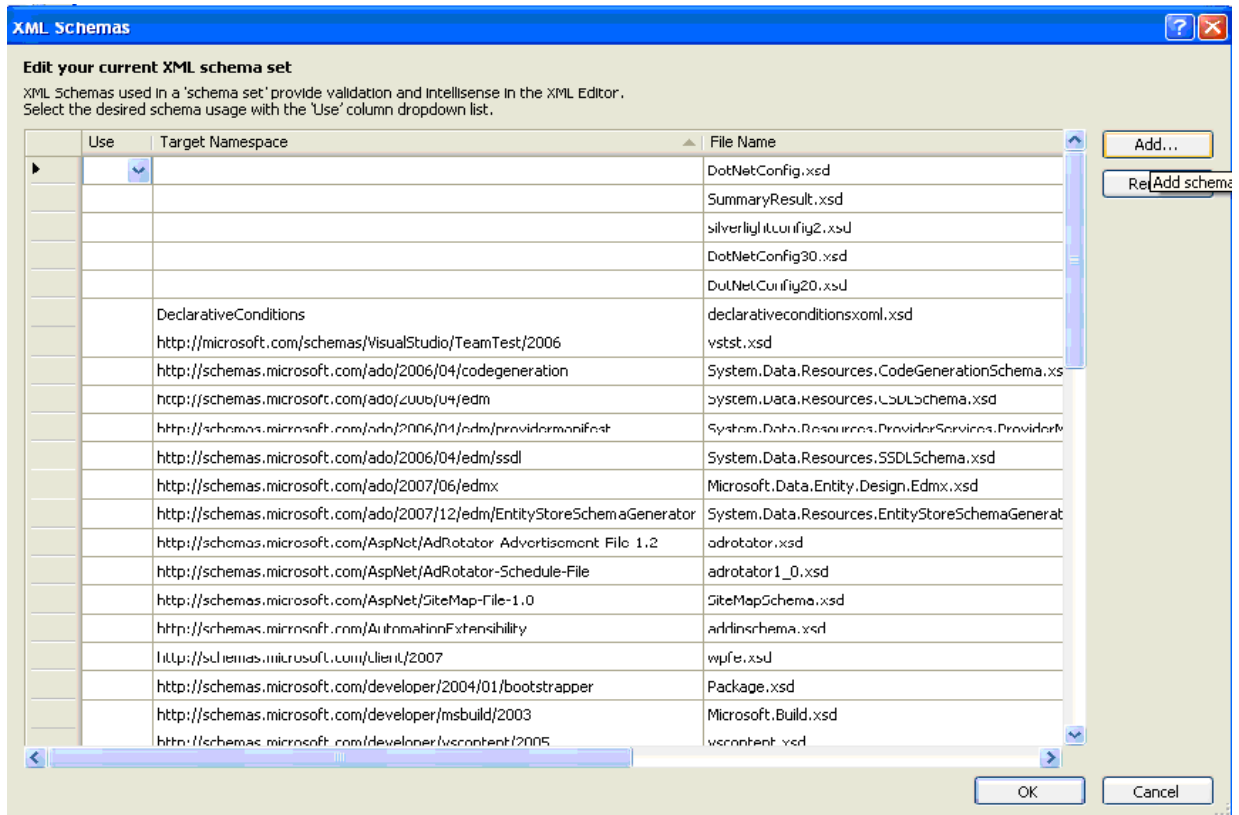
Follow the steps given in Appendix A.2 to create XML file then follow the following steps to attach schema file.

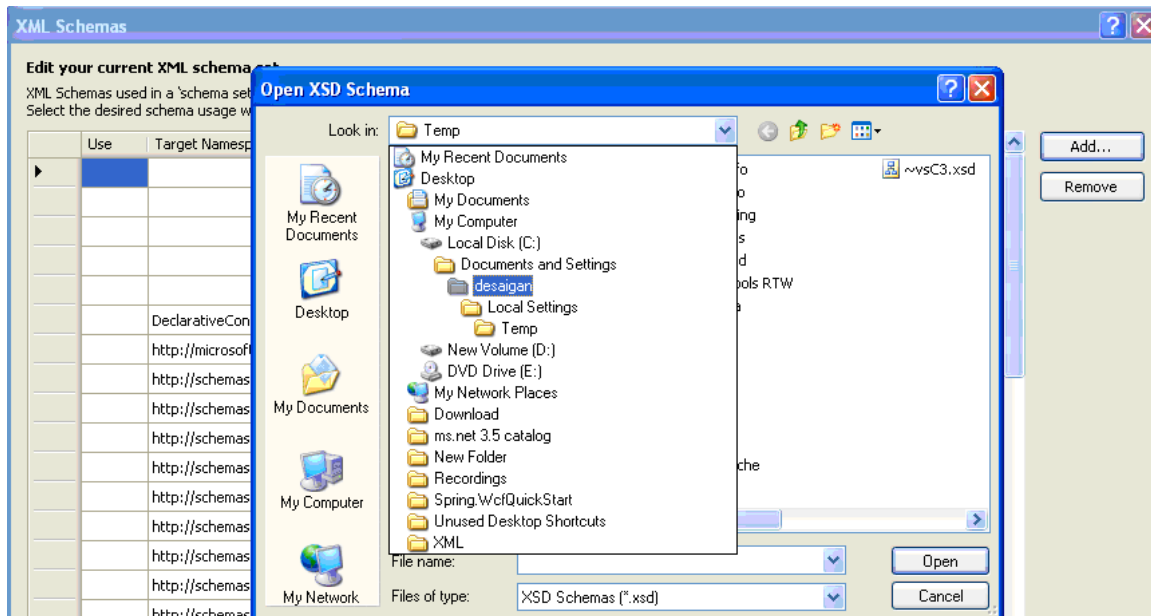
- a. Go to properties window (press F4) and Select Schema tab.





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

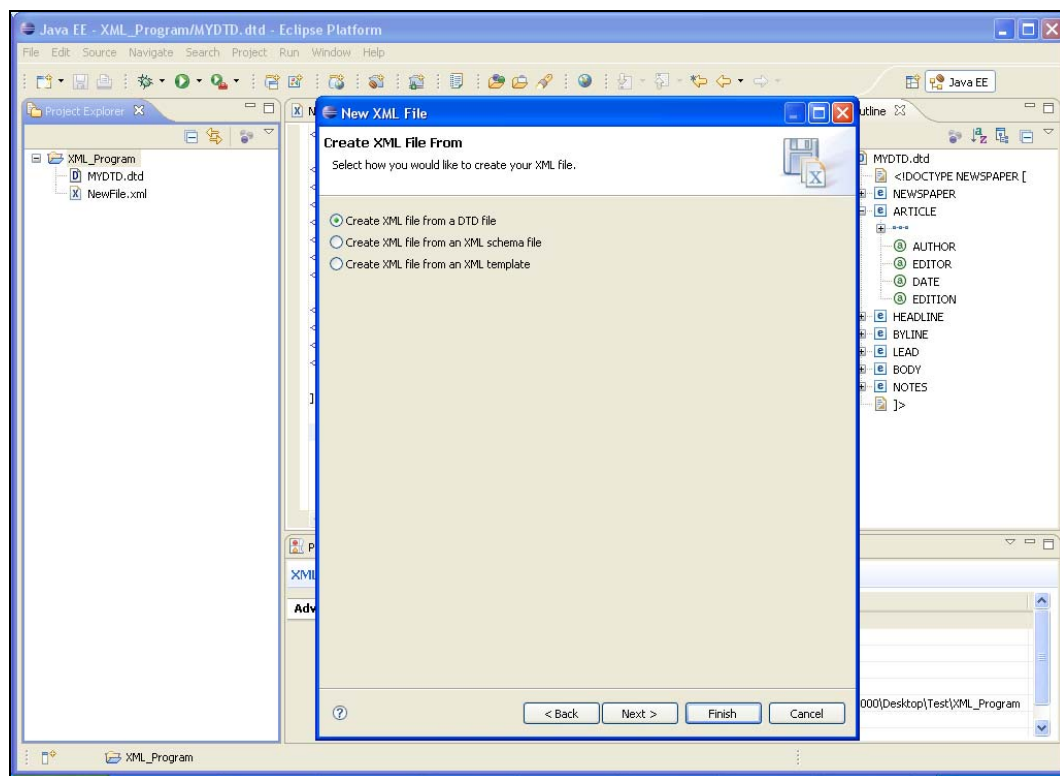




Validating your XML file against Schema file will be taken care by Editor itself & necessary errors will be shown.

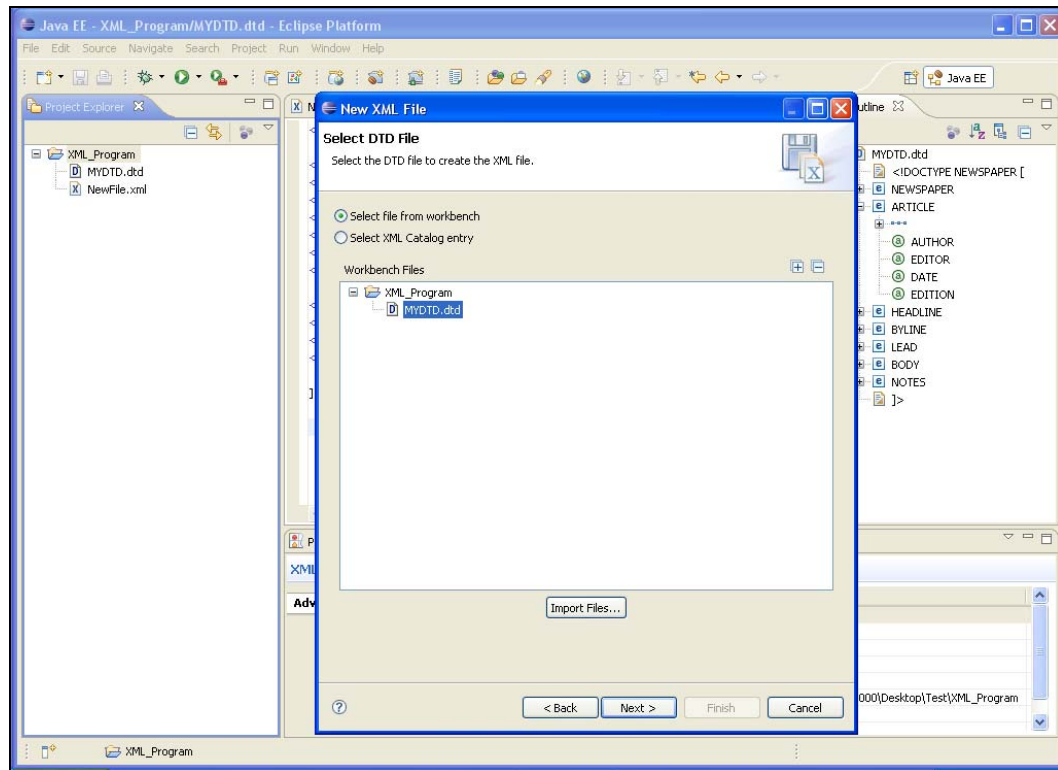
## Appendix B:

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.



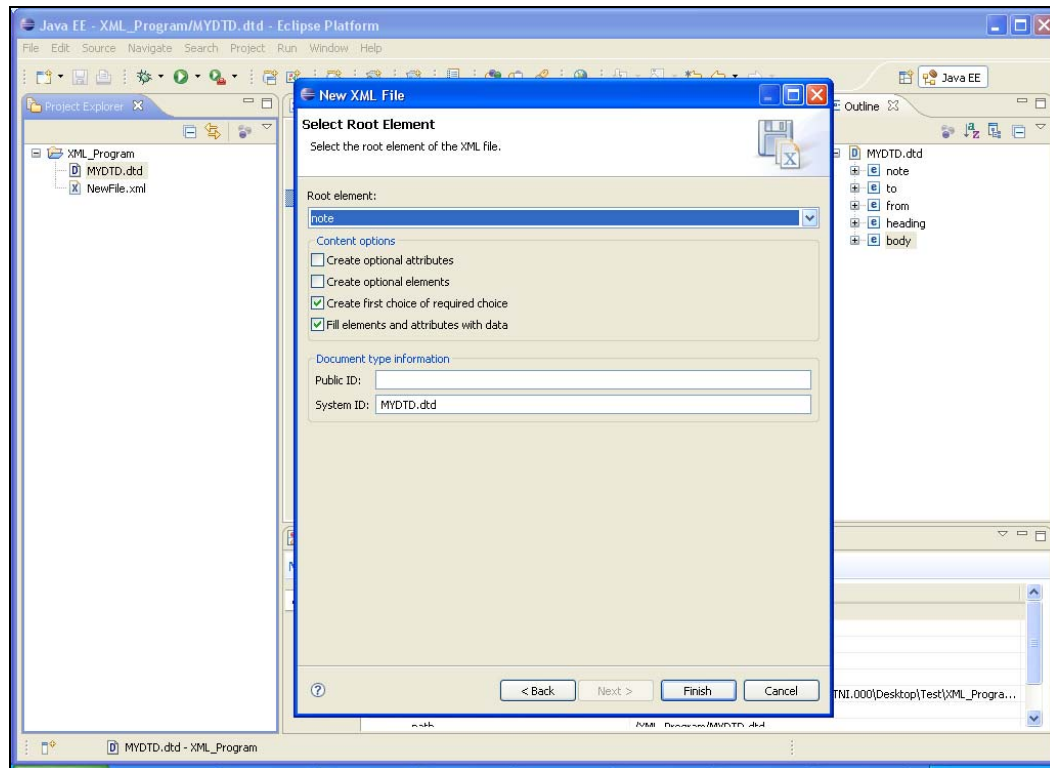
**Figure 7: For creating XML file from a DTD file**

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



**Figure 8: Selection of a DTD file**

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



**Figure 9: Select Root element for a XML file**

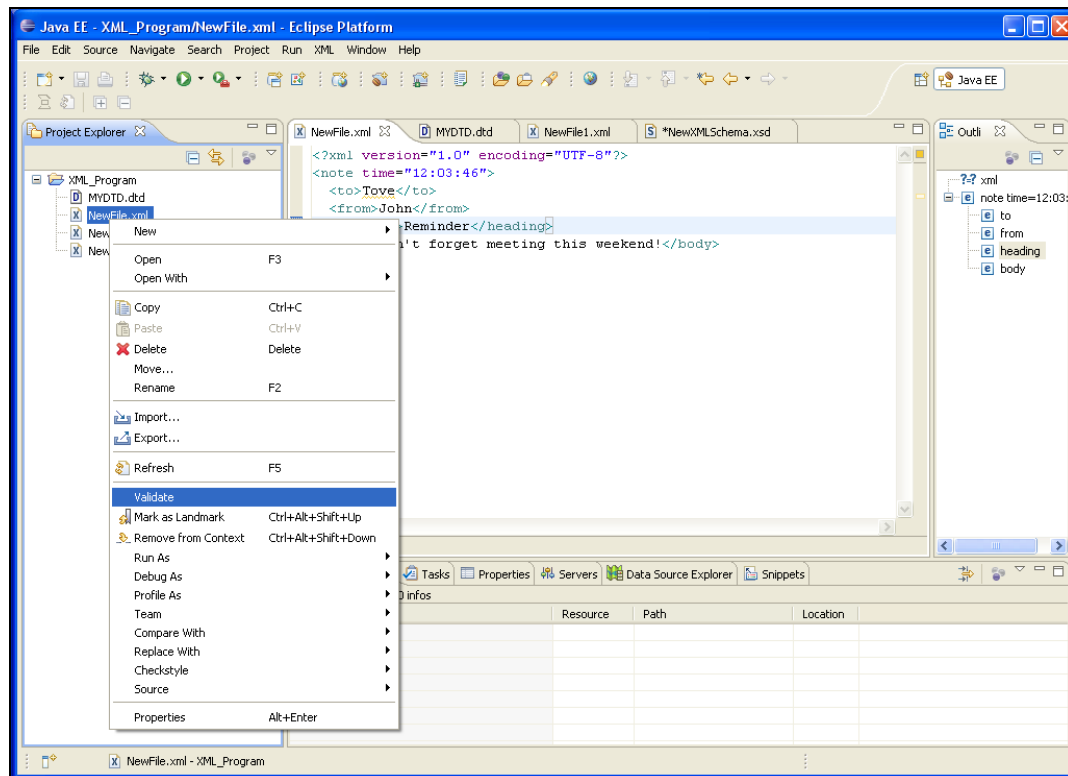
- a. For creating XML file from existing schema file, you should follow the same steps, except select **“Create XML file from an XML Schema”** option.

### 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 .....	8
Figure 2: Output of Shiporder.xsd .....	15
Figure 3: Output of Shiporder.xml .....	17
Figure 4: Opening eclipse Luna.....	21
Figure 5: Creating General Project.....	23
Figure 6: For viewing output .....	29
Figure 7: For creating XML file from a DTD file.....	36
Figure 8: Selection of a DTD file.....	37
Figure 9: Select Root element for a XML file.....	38
Figure 10: For Validating a XML file .....	39



**Appendix D: Table of Examples**

Example 1: foodmenu.xml file .....	8
Example 2: shiporder.xsd .....	14
Example 3: Shiporder.xml .....	16
Example 4: Library.xml .....	18