XML Classes

XML Sample File

```xml
<?xml version="1.0"?>
<SuperProProductList>
    <Product ID="1" Name="Chair">
        <Price>49.33</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product ID="2" Name="Car">
        <Price>43399.55</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product ID="3" Name="Fresh Fruit Basket">
        <Price>49.99</Price>
        <Available>False</Available>
        <Status>4</Status>
    </Product>
</SuperProProductList>
```

The XML Classes

- .NET support XML classes by System.XML namespace.
  - Reading and writing XML directly using XmlWriter and XmlReader.
  - Dealing with XML as a collection of in-memory objects using the XDocument class.
  - The Xml control to transform XML content to displayable HTML.

The XML Writer

Demo

- XMLWriter class methods like WriteStartDocument() and WriteEndDocument().
- WriteStartElement() and WriteEndElement()
- WrtiteAttributeString()
- WrtiteString() – used to insert text content inside the element.

The XML Reader

A node is a designation that includes comments, whitespace, opening tags, closing tags, content, and even the XML declaration at the top of your file.

The XML Reader

- Read() method used to read the nodes.
- Nodes having properties like NodeType and Name.

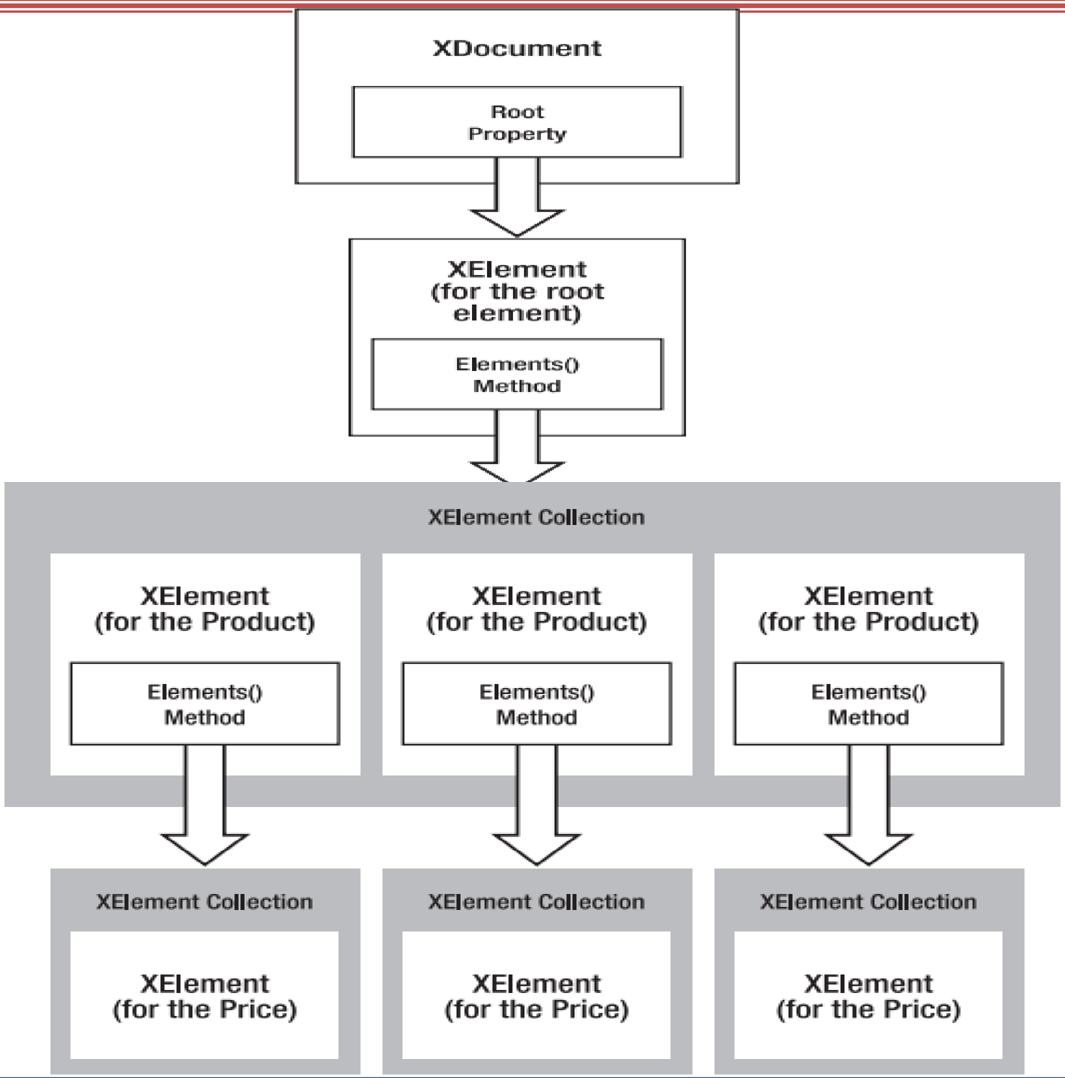
Demo

- The XmlReader and XmlWriter use XML as a backing store.

Working with XML Documents in Memory

- The XDocument class provides a different approach to XML data.
- It provides an in-memory model of an entire XML document.
- You can then browse through the entire document, reading, inserting, or removing nodes at any location.
- For these classes use System.XML.Linq namespace.

Working with XML Documents in Memory



XDocument

- To start building a XML document, you need to create the XDocument, XElement, and XAttribute objects that comprise it.
- Example: XElement element = new XElement("Price", 23.99);

```
//Here’s an example that creates an element with three nested elements and their content
XElement element = new XElement("Product",
        new XElement("ID", 3),
        new XElement("Name", "Fresh Fruit Basket"),
        new XElement("Price", 49.99)
        );
```

Here’s the scrap of XML that this code creates:

```
<Product>
        <ID>3</ID>
        <Name>Fresh Fruit Basket</Name>
        <Price>49.99</Price>
</Product>
```

Demo

Reading an XML Document

- The XDocument makes it easy to read and navigate XML content.
- Xdocument.Load() method to read XML document.

XElement class

```
// Load the document.
XDocument doc = XDocument.Load(file);
// Loop through all the nodes, and create the list of Product objects.
List<Product> products = new List<Product>();
foreach (XElement element in doc.Element("SuperProProductList").Elements("Product"))
{
        Product newProduct = new Product();
        newProduct.ID = (int)element.Attribute("ID");
        newProduct.Name = (string)element.Attribute("Name");
        newProduct.Price = (decimal) element.Element("Price");
        products.Add(newProduct);
}
// Display the results.
gridResults.DataSource = products;
gridResults.DataBind();
```

XML

XML Validation

- XML Schema defines the rules to which a specific XML document should conform, such as the allowable elements and attributes, the order of elements, and the data type of each element.
- You define these requirements in an XML Schema document (XSD).
- XML allows you to create a custom language for storing data, and XSD allows you to define the syntax of the language you create.

XML Namespaces

- Namespaces are disambiguate elements by making it clear what markup language they belong to.
- Example: you could tell the difference between your SuperProProductList standard and another organization's product catalog because the two XML languages would use different namespaces.

END OF LECTURE