XML
CONCEPTS AND IMPLEMENTATION

WEEK 04

Murtaza Munawar Fazal

XML Technology

Derived from SGML (Standard Generalized Markup Language)

Text-based format

Describes document structures using markup tags

Useful for describing document formats for Web

It is also useful for describing both structured as well as semi-structured data.

Why XML?

1. Platform-independent
   - Can be run over any operating System
   - Can be processed using any programming language
2. Extensible:
   - No fixed vocabulary
   - Flexible, change the structure by adding new tags
3. Supports Global implementation being fully Unicode
4. No emphasis on display and rendering the XML document

XML Skeleton

Syntax similar to HTML

- Start tag <start>
- end tag </start>
- Case sensitive
- White spaces are preserved
- Elements must be properly nested

XML Infoset

- Information items which are abstract representation of components of an xml document
- Up to 11 information items including Document, element, attribute, processing instructions etc.

Skeleton

Schema Languages

- Used to describe the structure and content of xml document
- During document interchange describes the contract between the producer and consumer application
- DTDs, XDR, XSD

XML APIs

- Tree-model API – (full tree in memory)
  - XmlDocument Class in .NET, DOM, SAX
- Cursor-based API – lens on one node at a time
  - XPathNavigator Class in .NET (only required fields in mem.)
  - XmlCursor class from BEA's XMLBeans toolkit
- Streaming API
  - SAX, XMLPULL,
- Object to XML Mapping API
  - include JAXB, the .NET Framework's XmlSerializer and Castor.

Skeleton

XML Query

- In some cases data extraction from XML documents through available APIs is difficult or all of the data can not be extracted.
- XPath, XQuery

XML Transformation

- XSLT is the primary transformation tool
- There are various vendor tools available for transforming XML to HTML, PDF, WORD, RTF etc.

Uses of XML

Traditional data processing

- XML encodes the data for a program to process

Document-driven programming

- XML documents are containers that build interfaces and applications from existing components

Archiving

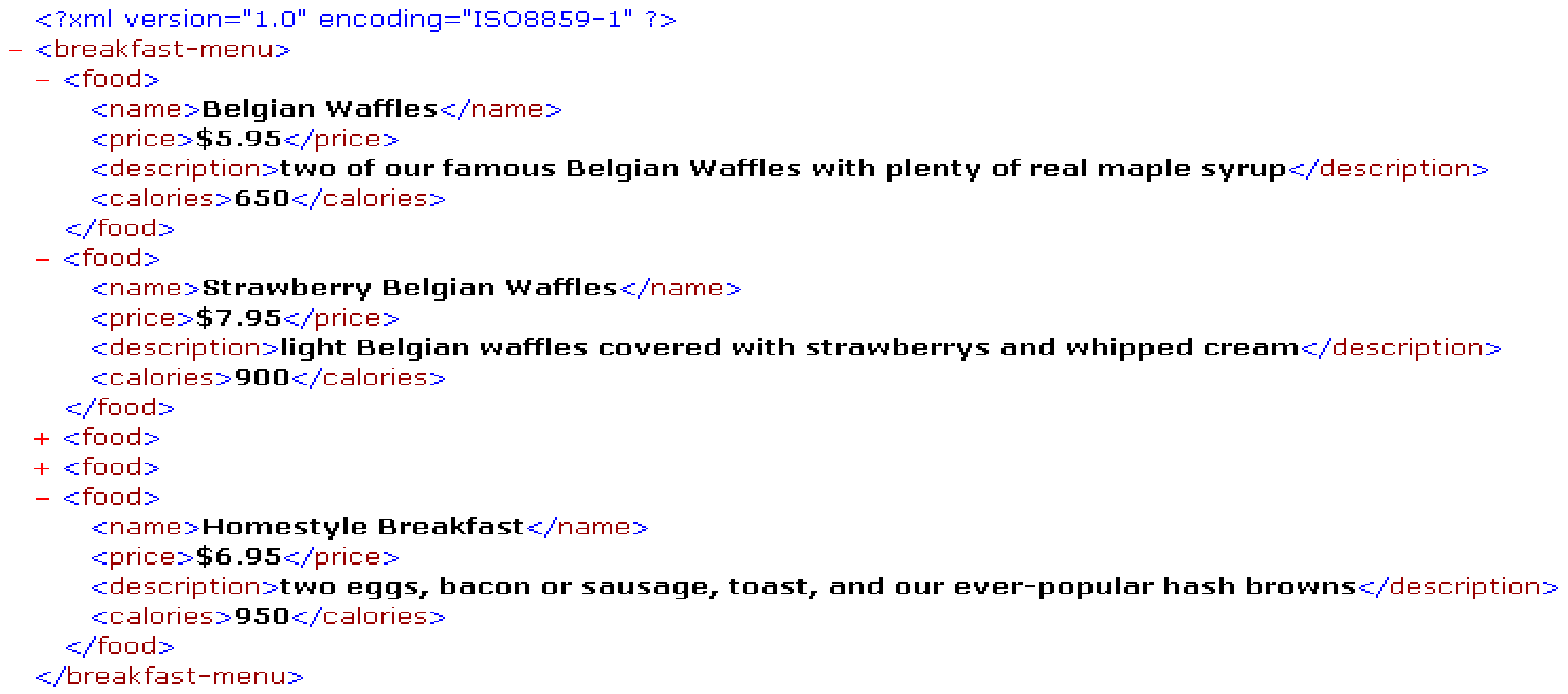
- Foundation for document-driven programming, where the customized version of a component is saved (archived) so it can be used later

Binding

- DTD or schema that defines an XML data structure is used to automatically generate a significant portion of the application that will eventually process that data

Schema

Anatomy of XML document



```
  <?xml version="1.0" encoding="ISO8859-1" ?>
- <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 strawberrys and whipped cream</description>
      <calories>900</calories>
    </food>
  + <food>
  + <food>
  - <food>
      <name>Homestyle Breakfast</name>
      <price>$6.95</price>
      <description>two eggs, bacon or sausage, toast, and our ever-popular hash browns</description>
      <calories>950</calories>
    </food>
  </breakfast-menu>
```

XSD

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

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Section-II Coverage

XML in .NET Framework

XML.NET Architecture

Namespaces and Classes Hierarchy

XmlReader Class

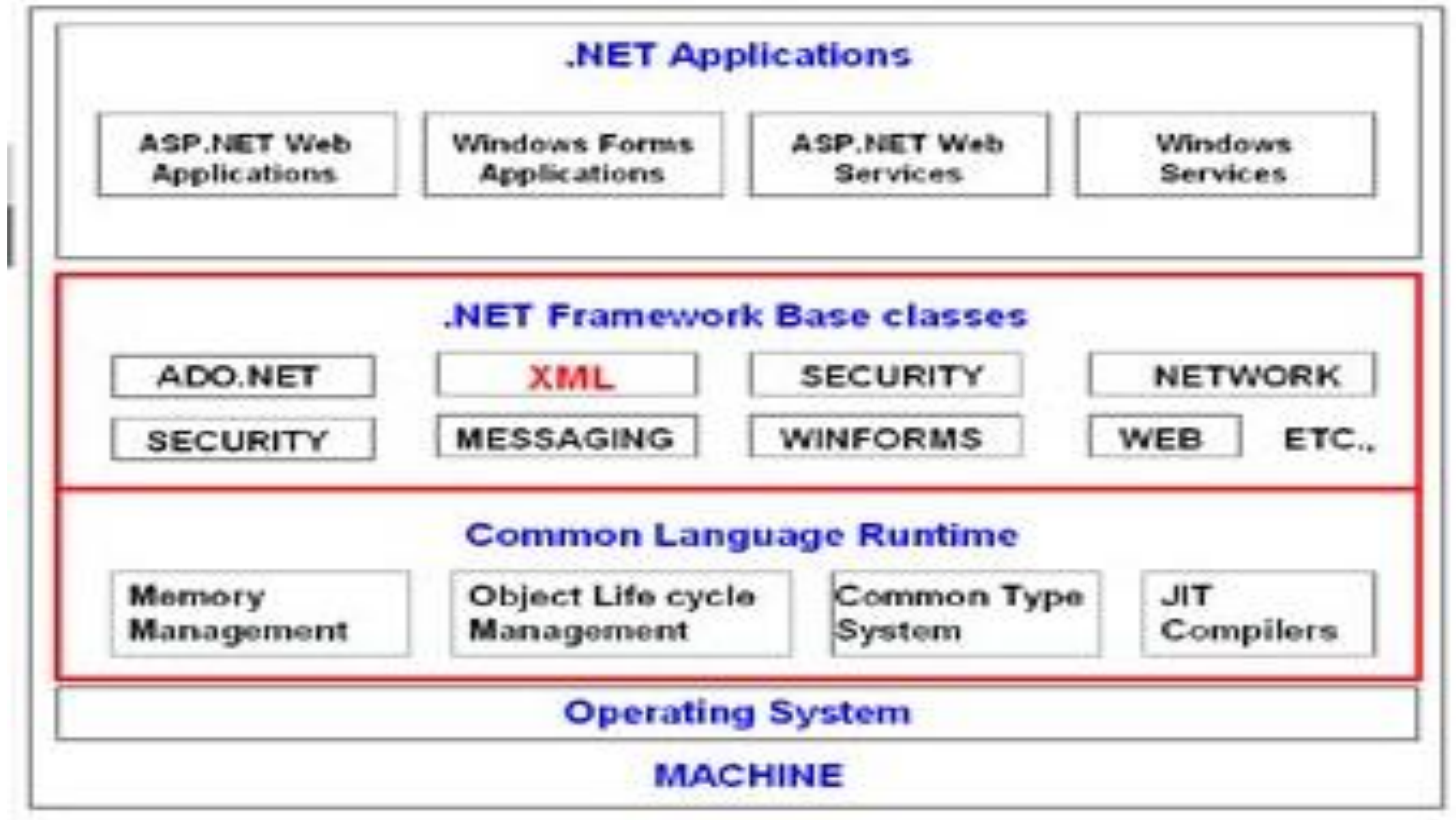
XmlWriter Class

XmlDocument Class

System.Xml.Xsl

Sytem.Xml.XPath

XML in .NET Framework



XML.NET Architecture

XML is extensible in the .NET Framework (extending the existing classes)

In the .NET Framework XML has a pluggable architecture.

Pluggable ~ abstract classes can be easily substituted

Pluggable ~ Data can be streamed between components and new components can be inserted that can alter the processing of the document

Pluggable Architecture

Developer can create new classes by extending the existing ones

Can introduce new features and thus changes the behavior of existing ones

e.g. create MyXmlTextReader extending the XmlTextReader class which converts an attribute-centric document to an element-centric document

XML.NET Classes & Namespaces

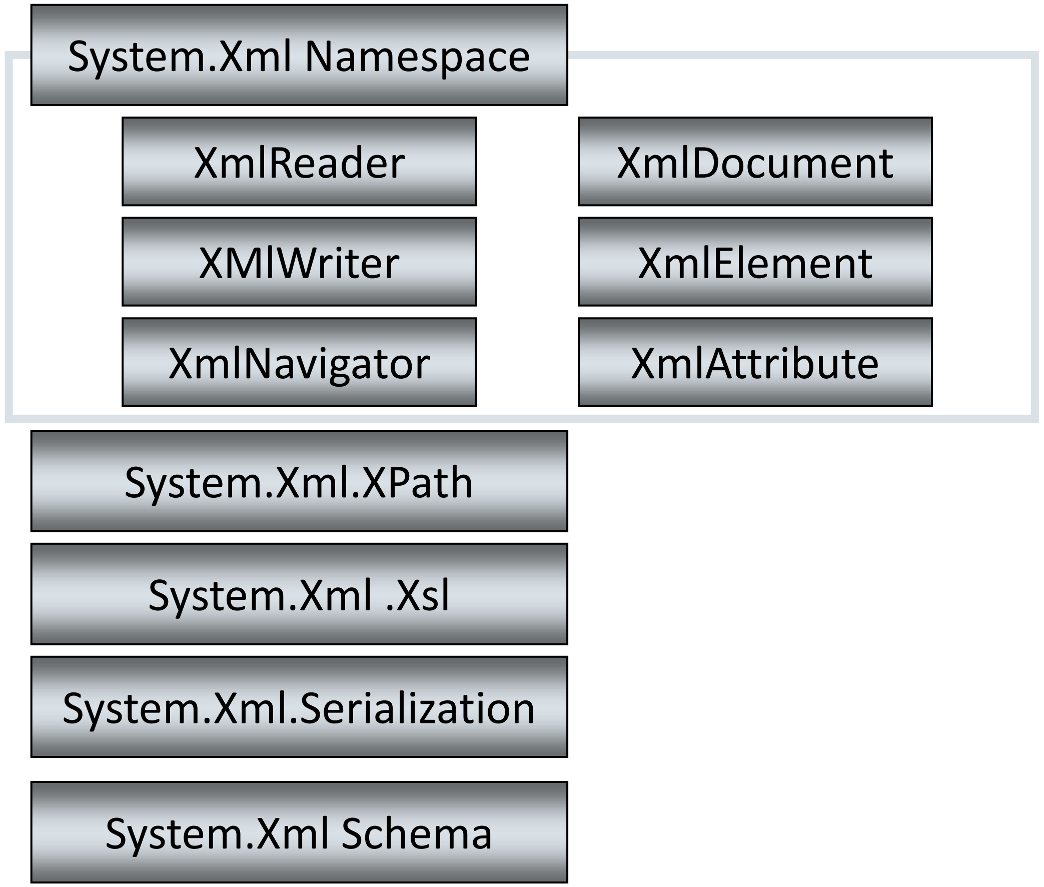
The Framework has several XML classes that allow working with XML documents and data

These classes are the core elements of .NET Framework

Most of the XML classes are contained in the System.Xml namespace.

The term namespace refers to a logical grouping of the classes designed to implement a specific functionality.

Hierarchy

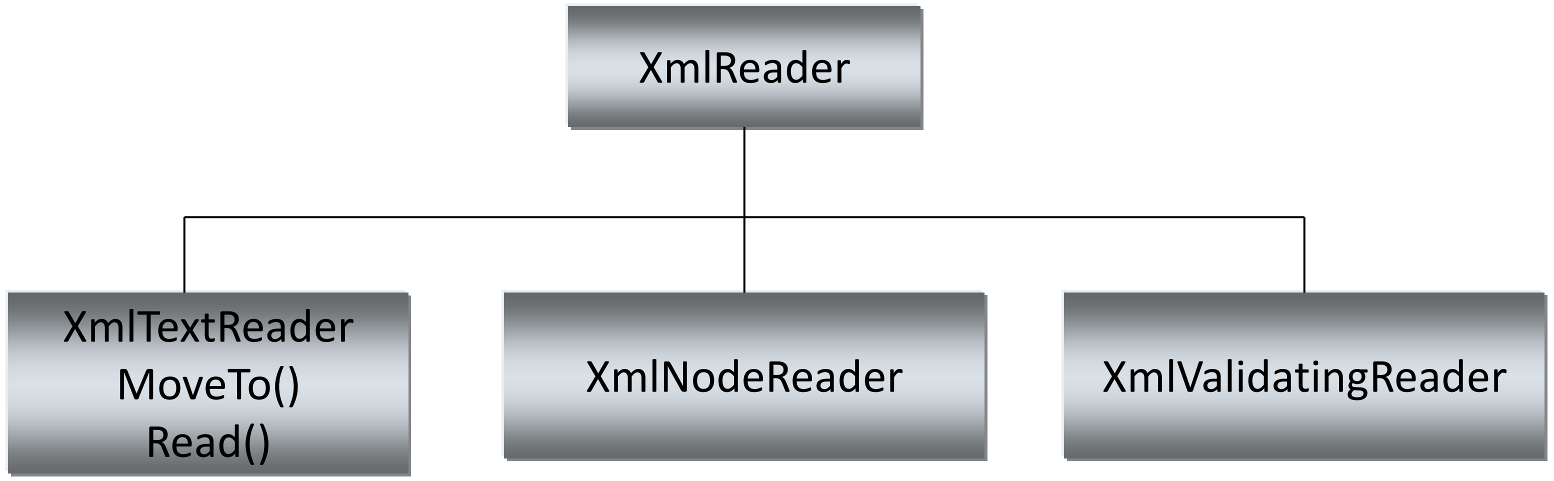

System.Xml Namespace
XmlReader
XmlDocument
XMlWriter
XmlElement
XmlNavigator
XmlAttribute
System.Xml.XPath
System.Xml .Xsl
System.Xml.Serialization
System.Xml Schema

XML Parsing

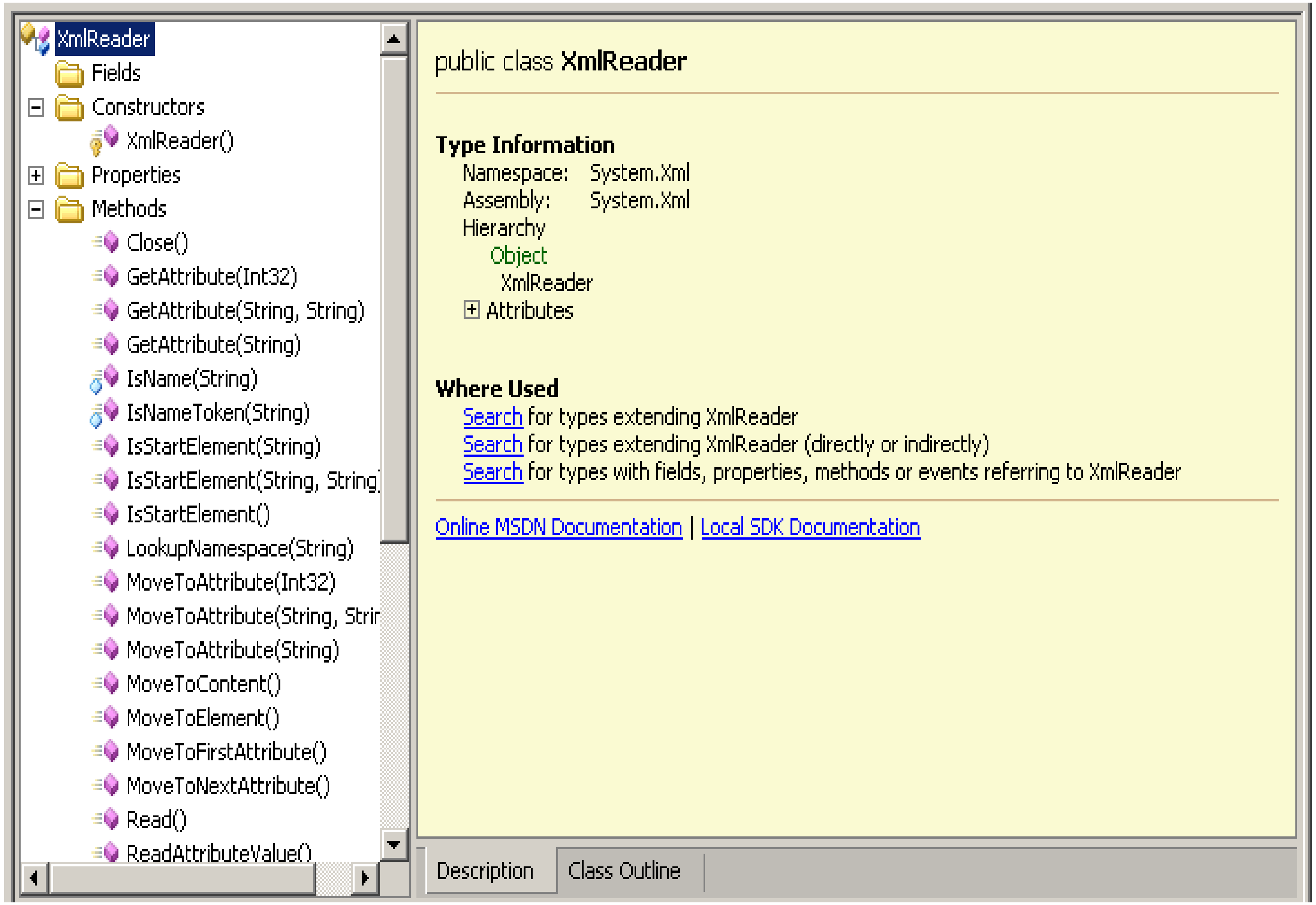
Parsing in .NET

- Pull Model
- DOM

XmlReader Class



- XmlReader – Abstract Class
- XmlTextReader – Reads text based stream, non-cached, read-only
- XmlNodeReader – Reads in memory DOM tree
- XmlValidatingReader – validates with DTD, XDR XSD schemas



XmlReader
Fields
Constructors
XmlReader()
Properties
Methods
Close()
GetAttribute(Int32)
GetAttribute(String, String)
GetAttribute(String)
IsName(String)
IsNameToken(String)
IsStartElement(String)
IsStartElement(String, String)
IsStartElement()
LookupNamespace(String)
MoveToAttribute(Int32)
MoveToAttribute(String, Strir
MoveToAttribute(String)
MoveToContent()
MoveToElement()
MoveToFirstAttribute()
MoveToNextAttribute()
Read()
ReadAttributeValue()

public class **XmlReader**

Type Information
Namespace: System.Xml
Assembly: System.Xml
Hierarchy
Object
XmlReader
Attributes

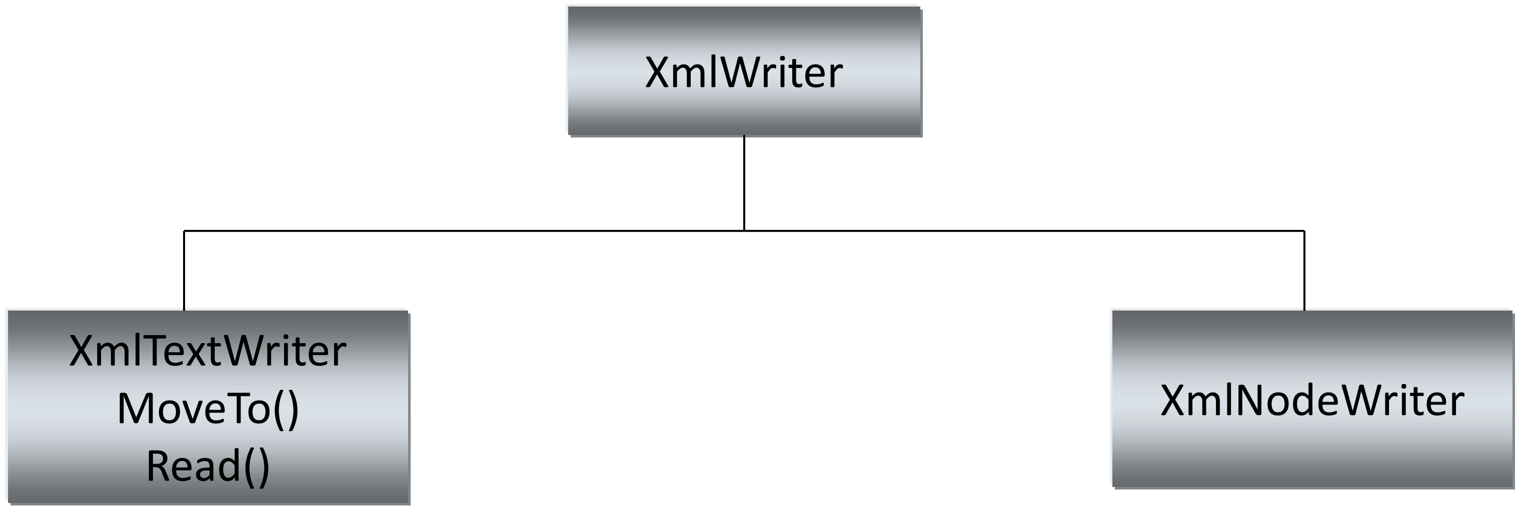
Where Used
Search for types extending XmlReader
Search for types extending XmlReader (directly or indirectly)
Search for types with fields, properties, methods or events referring to XmlReader

Online MSDN Documentation | Local SDK Documentation

XmlReader Example

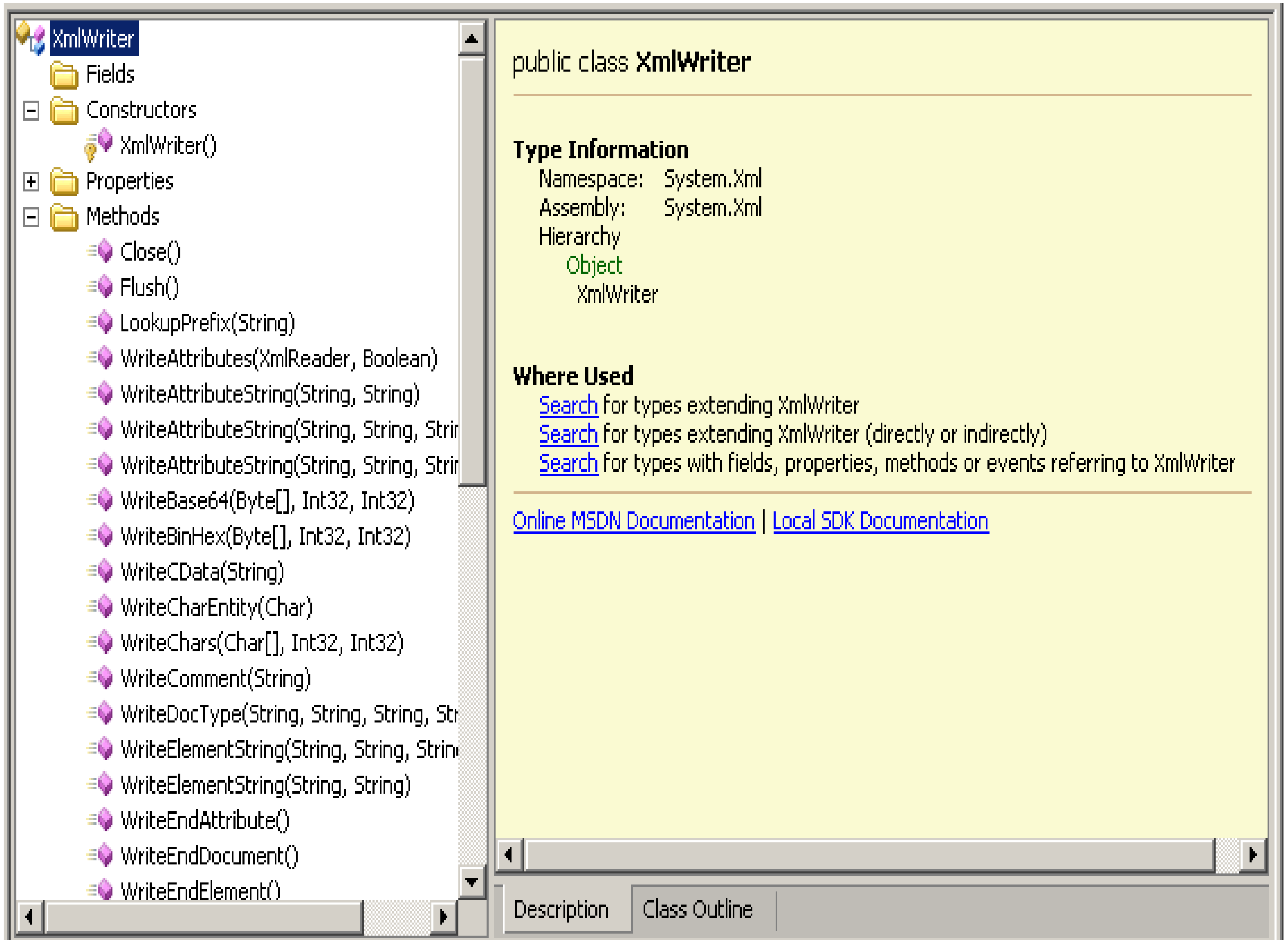
```
XmlReader reader;
Reader = new XmlTextReader(“test.xml”);
While(reader.Read()){
/*
Your processing code here
*/
}
```

XmlWriter Class



```
XmlWriter writer = new XmlTextWriter();
writer.WriteStartDocument();
Writer.WriteStartElement(“name”, “Ahmed”);
```

XmlWriter Class Overview



XmlWriter
Fields
Constructors
XmlWriter()
Properties
Methods
Close()
Flush()
LookupPrefix(String)
WriteAttributes(XmlReader, Boolean)
WriteAttributeString(String, String)
WriteAttributeString(String, String, Strir
WriteAttributeString(String, String, Strir
WriteBase64(Byte[], Int32, Int32)
WriteBinHex(Byte[], Int32, Int32)
WriteCData(String)
WriteCharEntity(Char)
WriteChars(Char[], Int32, Int32)
WriteComment(String)
WriteDocType(String, String, String, Str
WriteElementString(String, String, Strin
WriteElementString(String, String)
WriteEndAttribute()
WriteEndDocument()
WriteEndElement()

public class **XmlWriter**

Type Information

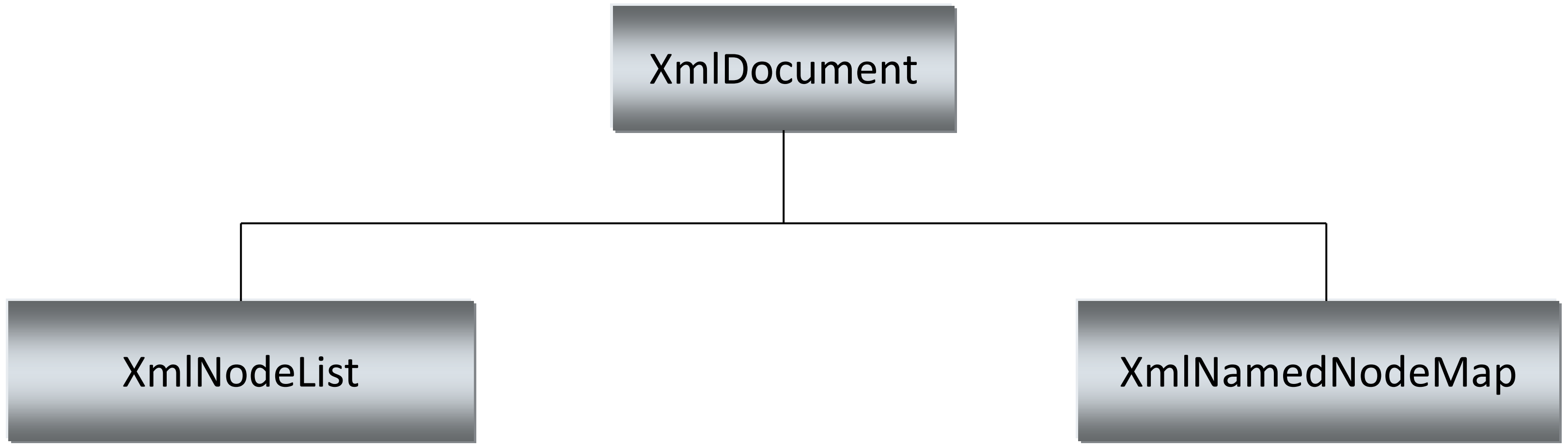
Namespace: System.Xml
Assembly: System.Xml
Hierarchy
Object
XmlWriter

Where Used

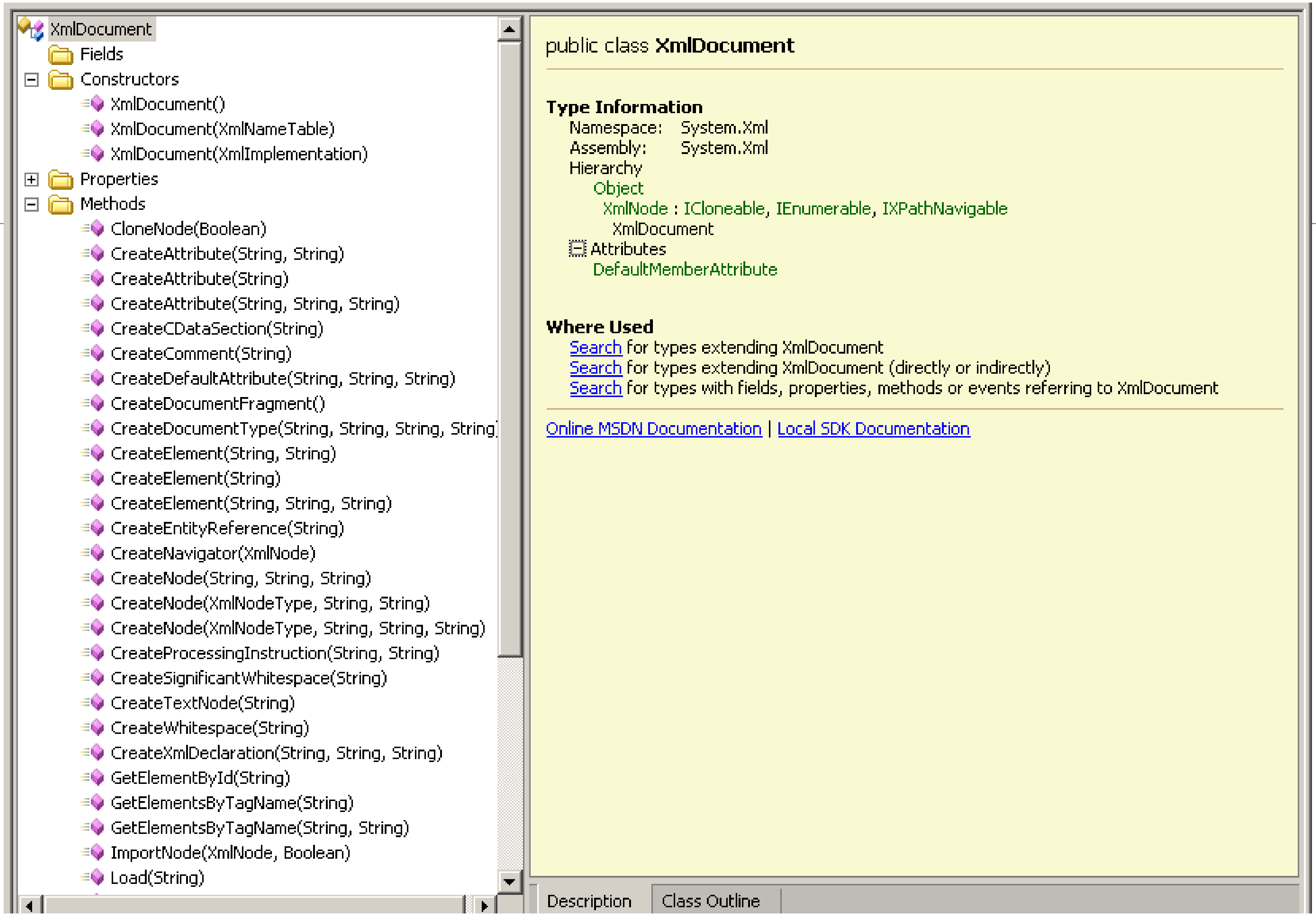
Search for types extending XmlWriter
Search for types extending XmlWriter (directly or indirectly)
Search for types with fields, properties, methods or events referring to XmlWriter

Online MSDN Documentation | Local SDK Documentation

XmlDocument (DOM)



- XmlNodeList – Collection of Different Xml Nodes
- XmlNamedNodeMap – collection of Attributes
- Methods
  - Load
  - LoadXml
  - Save



XmlDocument
- Fields
- Constructors
  - XmlDocument()
  - XmlDocument(XmlNameTable)
  - XmlDocument(XmlImplementation)
- Properties
- Methods
  - CloneNode(Boolean)
  - CreateAttribute(String, String)
  - CreateAttribute(String)
  - CreateAttribute(String, String, String)
  - CreateCDataSection(String)
  - CreateComment(String)
  - CreateDefaultAttribute(String, String, String)
  - CreateDocumentFragment()
  - CreateDocumentType(String, String, String, String)
  - CreateElement(String, String)
  - CreateElement(String)
  - CreateElement(String, String, String)
  - CreateEntityReference(String)
  - CreateNavigator(XmlNode)
  - CreateNode(String, String, String)
  - CreateNode(XmlNodeType, String, String)
  - CreateNode(XmlNodeType, String, String, String)
  - CreateProcessingInstruction(String, String)
  - CreateSignificantWhitespace(String)
  - CreateTextNode(String)
  - CreateWhitespace(String)
  - CreateXmlDeclaration(String, String, String)
  - GetElementById(String)
  - GetElementsByTagName(String)
  - GetElementsByTagName(String, String)
  - ImportNode(XmlNode, Boolean)
  - Load(String)

public class **XmlDocument**

Type Information

Namespace: System.Xml
Assembly: System.Xml
Hierarchy
  Object
    XmlNode : ICloneable, IEnumerable, IXPathNavigable
      XmlDocument
Attributes
  DefaultMemberAttribute

Where Used

Search for types extending XmlDocument
Search for types extending XmlDocument (directly or indirectly)
Search for types with fields, properties, methods or events referring to XmlDocument

Online MSDN Documentation | Local SDK Documentation

Xpath Query in .NET

Used to specify query expressions to locate nodes in XML document

Used in XSLT stylesheets to locate and apply transformation to specific nodes in an XML document

Used in DOM code to locate and process specific nodes in an XML document

XML and ADO.NET

In .NET Framework tight integration has been introduced between XML classes and ADO.NET

The DataSet components are able to read and write XML using XmlRead and XmlWrite Classes

There is a synchronization between XmlDocument and DataSet, which enables automatic update

XmlDataDocument (extension of XmlDocument) provides a bridge between relational and hierarchical data

Serialization

What is Serialization?

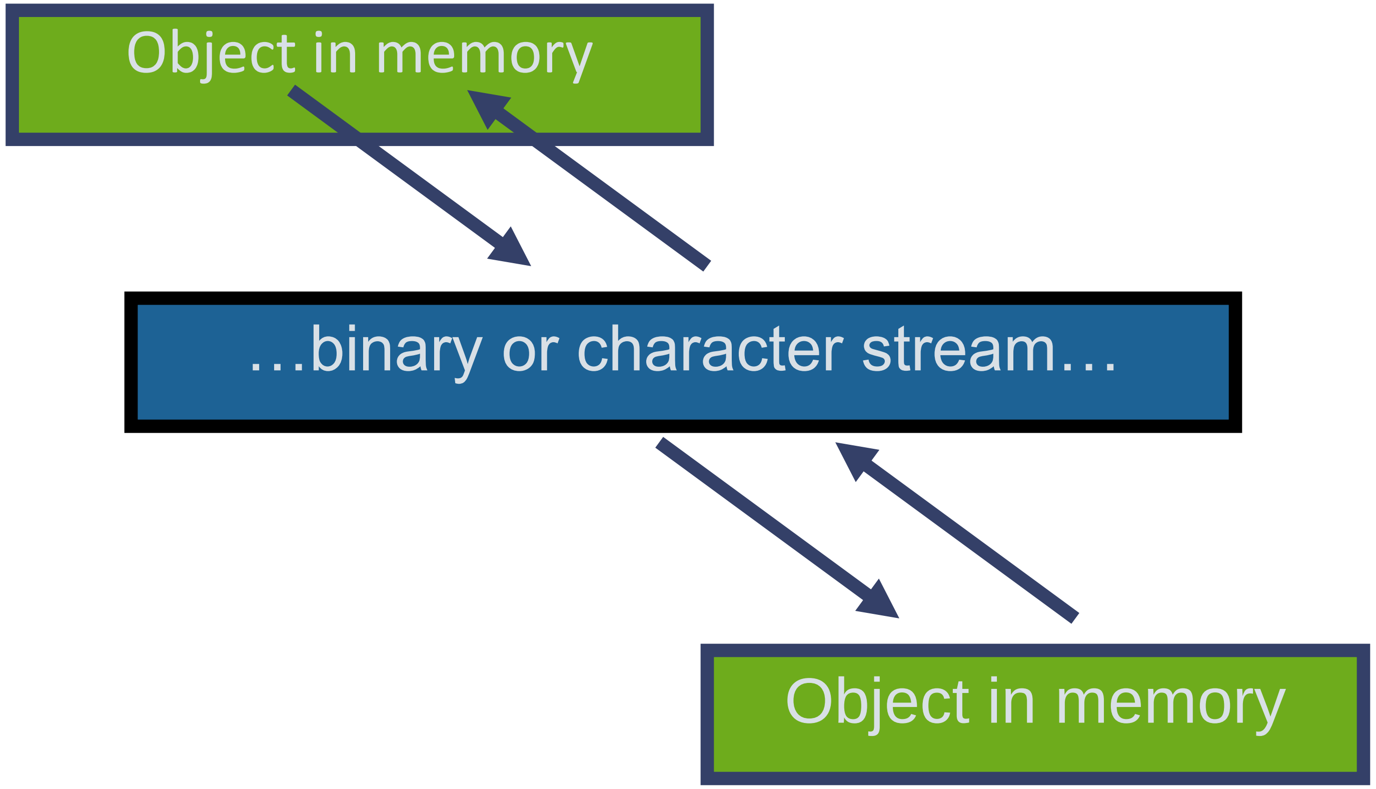
System.Serialization

Scenarios in Serialization

Basic Serialization

Custom Serialization

Serialization/Deserialization



Serialization

What is Serialization

Serialization is the process of converting an object, or a connected graph of objects, stored within computer memory, into a linear sequence of bytes

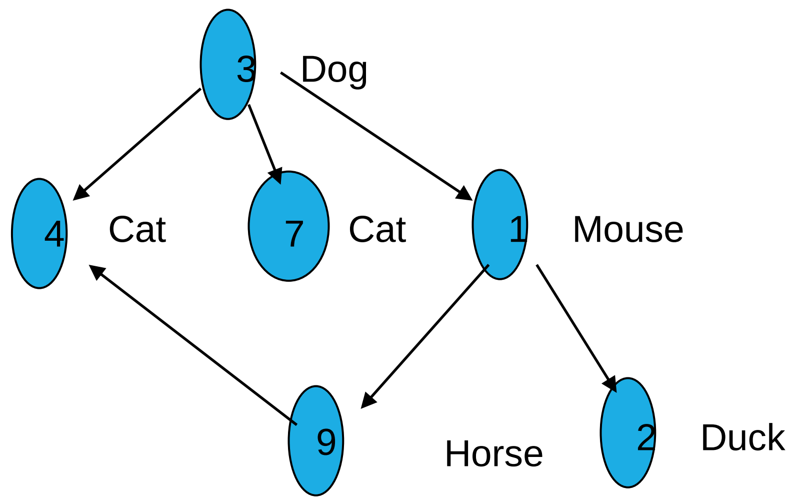
Use the sequence of bytes in several ways:

- Send it to another process
- Send it to the clipboard, to be browsed or used by another application
- Send it to another machine
- Send it to a file on disk

Serialization
Object Graph

What is an object graph?

- An object graph is a set of objects with some set of references to each other
- The most obvious problem is how to represent the links between the objects in the Serialized stream



Serialization
How Serialization Works

Because run-time metadata 'knows' about each object's layout in memory, and its field and property definitions, you can serialize objects automatically, without having to write code to serialize each field

The serialized stream might be encoded using XML, or a compact binary representation

The format is decided by the the Formatter object that you call:

- Binary
- SOAP
- Custom

Serializaiton

FileStream Example

```
class SerializeExample{

public static void Main(String[] args)
{
  ArrayList l = new ArrayList();
  for (int x=0; x< 100; x++) {
       l.Add (x);
  } // create the object graph
  FileStream s = File.Create("foo.bin"); // create the
  filestream BinaryFormatter b = new BinaryFormatter();
  // create the BinaryFormatter
  b.Serialize(s, l);
  // serialize the graph to the stream
} // end main

} // end class
```

Serializaiton

Deserialize Example

```
using System; using System.IO;
using System.Collections;
using System.Serialization;
using System.Serialization.Formatters.Binary;

class DeSerialize
{
  public static void Main(String[] args) {
  FileStream s = File.Open("foo.bin"); // open the
  filestream BinaryFormatter b = new BinaryFormatter(); //
  create the formatter ArrayList p = (ArrayList)
  b.Deserialize(s); // deserialize
  p.ToString(); // print out the new object graph
} // end Main
} // end Class DeSerialize
```

Serialization
Basic Serialization

A Type is NOT Serializable unless Type is specifically marked as Serializable

The Serializable Attribute

```
[Serializable] public class MyClass {}
```

The Non-Serializable Attribute

```
[Serializable] public class MyClass {
  [NotSerialized] int _cashSize;
}
```

.NET Serialization Facilities

Take an extremely simple C# class:

```
public class InitialConfiguration
{
 public enum Difficulty {hard, medium, easy};
 public InitialConfiguration() { }
 public Difficulty starting = Difficulty.medium;
}
```

.NET Serialization Facilities

Use .NET library functions to serialize it:

```
InitialConfiguration conf = new InitialConfiguration();

XmlSerializer ser
 = new XmlSerializer(typeof(InitialConfiguration));

XmlTextWriter writer = new XmlTextWriter(
 stream, System.Text.Encoding.UTF8);

ser.Serialize(writer, conf);
```

.NET Serialization Facilities

Get XML:

```
<?xml version="1.0" encoding=“utf-8"?>
<InitialConfiguration
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Starting>medium</starting>
</InitialConfiguration>
```