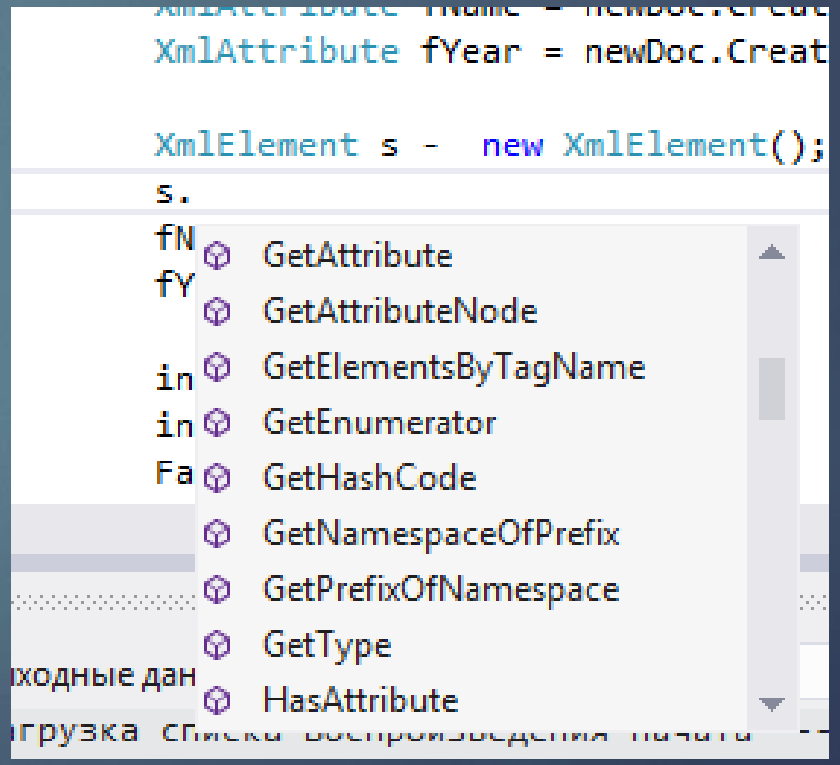
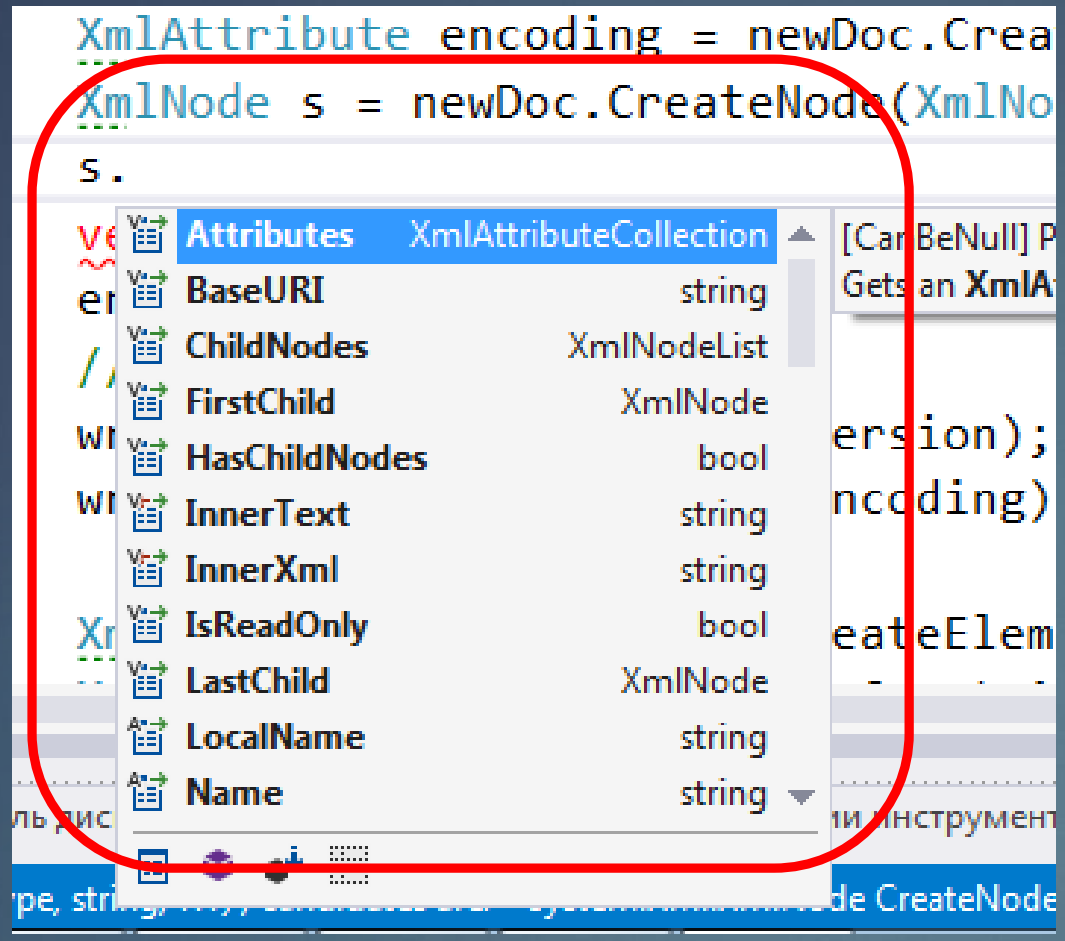
Работа с XML

Создание XML-документа DOM-модель

```
using System.Xml;
```

- построить объектную м весь документ или отдельный элемент
- **XmlNode**
- **XmlDocument**
- **XmlElement**
- **XmlAttribute**
- **XmlText**
- **XmlComment**
- **XmlNodeList**




XmlAttribute encoding = newDoc.Crea
XmlNode s = newDoc.CreateNode(XmlNo
s.
Attributes XmlAttributeCollection
BaseURI string
ChildNodes XmlNodeList
FirstChild XmlNode
HasChildNodes bool
InnerText string
InnerXml string
IsReadOnly bool
LastChild XmlNode
LocalName string
Name string
[CanBeNull] P
Gets an XmlA
ersion);
ncoding)
eateElem
de CreateNode

Создание на основе объектной модели документа

```
XmlDocument doc = new XmlDocument();
XmlElement bookElem = doc.CreateElement("Книга");
bookElem.SetAttribute("Год", "2009");
XmlElement titleElem = doc.CreateElement("Название");
titleElem.InnerText = "Программирование";
XmlElement authorElem = doc.CreateElement("Автор");
authorElem.InnerText = "Ник Данилов";
bookElem.AppendChild(titleElem);
bookElem.AppendChild(authorElem);
doc.AppendChild(bookElem);

StringBuilder sb = new StringBuilder();
//может быть любой поток
using (StringWriter sw = new StringWriter(sb))
using (XmlTextWriter xtw = new XmlTextWriter(sw))
{
    xtw.Formatting = Formatting.Indented;
    doc.WriteContentTo(xtw);
}
```

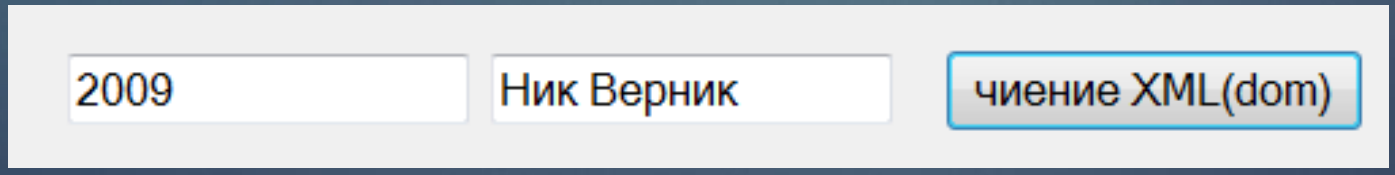
```
XmlDocument newDoc = new XmlDocument();
XmlElement Fam = newDoc.CreateElement("Book");
XmlElement wrapper = newDoc.CreateElement("xml");
//атрибуты
XmlAttribute version = newDoc.CreateAttribute("version");
XmlAttribute encoding = newDoc.CreateAttribute("encoding");

version.Value = "1.0";
encoding.Value = "utf-8";
//добавить
wrapper.Attributes.Append(version);
wrapper.Attributes.Append(encoding);

XmlElement info = newDoc.CreateElement("Bookinfo");
XmlAttribute fName = newDoc.CreateAttribute("Name");
XmlAttribute fYear = newDoc.CreateAttribute("Year");

fName.Value = "Учебник";
fYear.Value = "1989";

info.Attributes.Append(fName)
info.Attributes.Append(fYear)
Fam.AppendChild(info);
wrapper.AppendChild(Fam);

newDoc.AppendChild(wrapper);
newDoc.Save(@"book.xml");
```

1) создаем элемент
2) если элемент сложный, то создаем эти элементы
3) создаем текст
4) все элементы добавляем в основной элемент
5) Затем добавляем в корневой элемент

```
<xml version="1.0" encoding="utf-8">
  <Book>
    <Bookinfo Name="Учебник" Year="1989" />
  </Book>
</xml>
```

Чтение XML-файла с помощью класса XmlDocument

```
const string sourceXml =
        "<книга год=\"2009\">"+
        "<название>Программирование</название>"+
        "<автор>Ник Верник</автор>"+
        "</книга>";

        XmlDocument doc = new XmlDocument();
        doc.LoadXml(sourceXml);

        textBox1.Text = doc.GetElementsByTagName("книга")[0].Attributes["год"].Value;
        textBox2.Text = doc.GetElementsByTagName("автор")[0].InnerText;
    }
```



```
class Book
    {
        public string id;
        public string title;
        public string author;
    }
        List<Book> books = new List<Book>();

        void readxmlDom()
        {
            XmlDocument doc = new XmlDocument();
            doc.Load(@"booksforRead.xml");
            XmlNodeList nodes = doc.DocumentElement.SelectNodes("/catalog/book");

            foreach (XmlNode node in nodes)
            {
                Book book = new Book();

                book.author = node.SelectSingleNode("author").InnerText;
                book.title = node.SelectSingleNode("title").InnerText;
                book.id = node.Attributes["id"].Value;

                books.Add(book);
            }

        }
```

```
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    ...
  </book>
  <book id="bk102">
    <author>Steve Jin</author>
    <title>TBD</title>
    <genre>Computer</genre>
    <price>34</price>
    <publish_date>2015-09-01</publish_date>
    <description>
     something
    </description>
 </book>
</catalog>
```

SAX
Simple API to Xml

DOM	SAX
Формальная спецификация DOM	Нет формальной спецификации SAX
Запись xml дерева должна быть в памяти до парсинга	Не требует загрузки в память, может работать с максимальной глубиной xml дерева
Потоковое чтение с диска невозможно	Возможно потоковое чтение с диска
Меньше проблем с XML валидацией	Больше проблем с XML валидацией

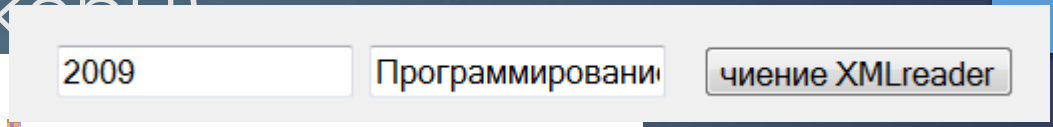
```
XmlTextWriter writer = new XmlTextWriter(@"bookSax.xml", Encoding.UTF8);

      writer.WriteStartDocument(true);

      writer.WriteComment("This xml file done using SAX");

      writer.WriteStartElement("Book");


      writer.WriteStartElement("Bookinfo");
      writer.WriteStartAttribute("Name");
      writer.WriteValue("Учебник");
      writer.WriteEndAttribute();

      writer.WriteStartAttribute("Year");
      writer.WriteValue(1998);
      writer.WriteEndAttribute();
      writer.WriteEndElement();

      writer.WriteEndElement();
      writer.WriteEndDocument();
      writer.Flush();
      writer.Close();
```

```
<xml version="1.0" encoding="utf-8">
  <Book>
    <Bookinfo Name="Учебник" Year="1989" />
  </Book>
</xml>
```

Создание на основе XmlWriter

```
StringBuilder sb1 = new StringBuilder();
using (StringWriter sw1 = new StringWriter(sb1))
using (XmlTextWriter xtw = new XmlTextWriter(sw1))
{
    xtw.Formatting = Formatting.Indented;

    xtw.WriteStartElement("Книга");
    xtw.WriteAttributeString("Год", "2009");

    xtw.WriteStartElement("Название");
        xtw.WriteString("Программирование");
    xtw.WriteEndElement();

    xtw.WriteStartElement("Автор");
        xtw.WriteString("Ник Данилов");
    xtw.WriteEndElement();

    xtw.WriteEndElement();
}
```

Чтение XML-файла с помощью класса XmlTextReader (марк



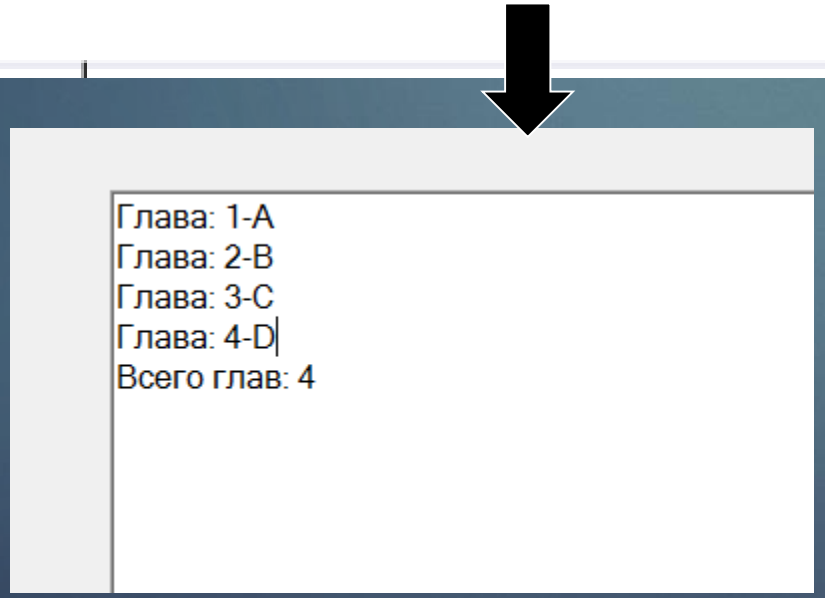

```
const string sourceXml =
          "<книга год=\"2009\">" +
          "<название>Программирование</название>" +
          "<автор>Ник Верник</автор>" +
          "</книга>";
string Year = null, author = null;
using (StringReader reader = new StringReader(sourceXml))
using (XmlTextReader xmlReader = new XmlTextReader(reader))
{
    while (xmlReader.Read())
    {
        if (xmlReader.NodeType == XmlNodeType.Element)
        {
            if (xmlReader.Name == "книга")
            {
                if (xmlReader.MoveToAttribute("год"))
                {
                    Year = xmlReader.Value;
                }
            }
            else if (xmlReader.Name == "название")
            {
                xmlReader.Read();
                author = xmlReader.Value;
            }
```

Выдача запроса к XML-документу с по

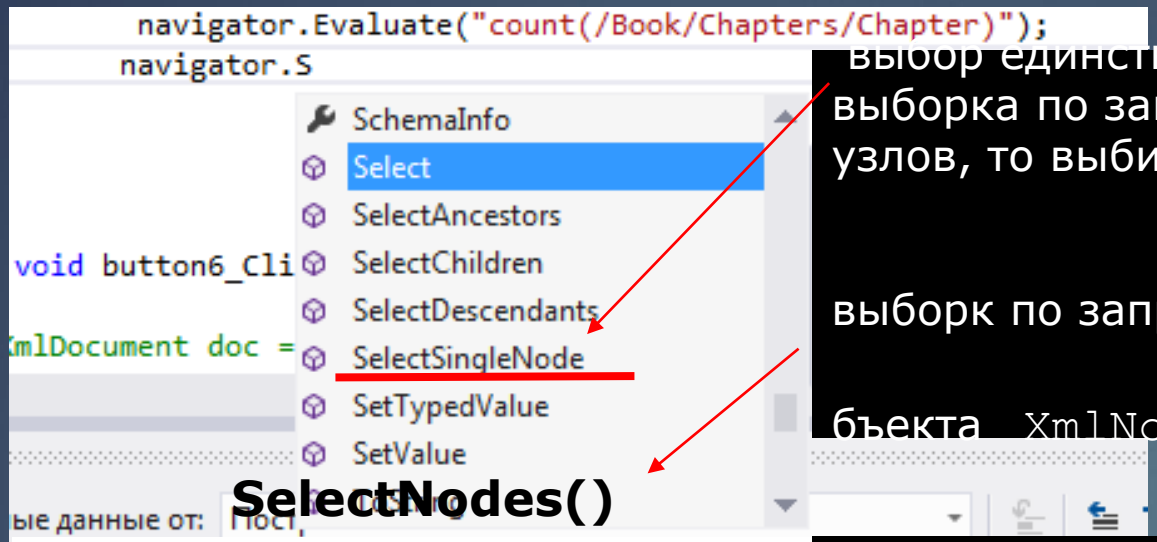
```
using System.Xml.XPath;
<?xml version="1.0" encoding="utf-16" ?>
<Book>
    <Title>Programmer</Title>
    <Author>Ivan Ivanov</Author>
    <Source Retrieved="2017-02-14T00:00:00">
    < URL >http://www.gutenberg.org/files/135/135.txt</URL>
    </Source>
    <Chapters>
        <Chapter>1-A</Chapter>
        <Chapter>2-B</Chapter>
         <Chapter>3-C</Chapter>
        <Chapter>4-D</Chapter>
      </Chapters>
 </Book>
```

```
private void button5_Click(object sender, EventArgs e)
{    richTextBox1.Text="";
     XPathDocument doc = new XPathDocument("my.xml");
     XPathNavigator navigator = doc.CreateNavigator();
     XPathNodeIterator iter = navigator.Select("/Book/Chapters/Chapter");
     while (iter.MoveNext())
     {
        richTextBox1.Text +="Глава: "+ iter.Current.Value+ Environment.NewLine;
     }
      richTextBox1.Text +="Всего глав: "+
                navigator.Evaluate("count(/Book/Chapters/Chapter)");

}
```

выбор в документе всех узлов с именем Chapter, которые находятся в элементах ….



позволяет выбирать элементы, соответствующие определенному селектору



выбор единственного узла из выборки. Если выборка по запросу содержит несколько узлов, то выбирается первый

выборк по запросу коллекции узлов в виде

бъекта XmlNodeList

SelectNodes()

Все дочерние узлы текущего узла

выбор всех узлов с определенным именем

выбор атрибута текущего узла

```
//----------------------
    navigator.Select("*");
    // выбор всех дочерних узлов
    XPathNodeIterator iter1 = navigator.Select("*");
   //атрибуты элемента
    XPathNodeIterator iter2 = navigator.Select("Source");
    foreach (XmlNode n in iter2)
        Console.WriteLine(n.SelectSingleNode("@Retrieved").Value);
    //вложенный элемент  имеет определенное значение
    XPathItem iter3 = navigator.SelectSingleNode("Book[Chaper='1']");
```

выбор элементов с определенным значением вложенного элемента

Преобразование информации из базы данных в XML-документ

```
XmlDocument doc = new XmlDocument();
DataSet dataSet = new DataSet("Books");
using (SqlConnection conn = new SqlConnection(GetConnectionString()))
using (SqlCommand cmd = new SqlCommand("SELECT * FROM Books", conn))
using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))
{
    adapter.Fill(dataSet);
}

StringBuilder sb = new StringBuilder();
using (StringWriter sw = new StringWriter(sb))
{
    dataSet.WriteXml(sw);
}
```

Преобразование XML-документа в HTML-документ

Таблица стилей для преобразования XSLTFile1.xslt w3c

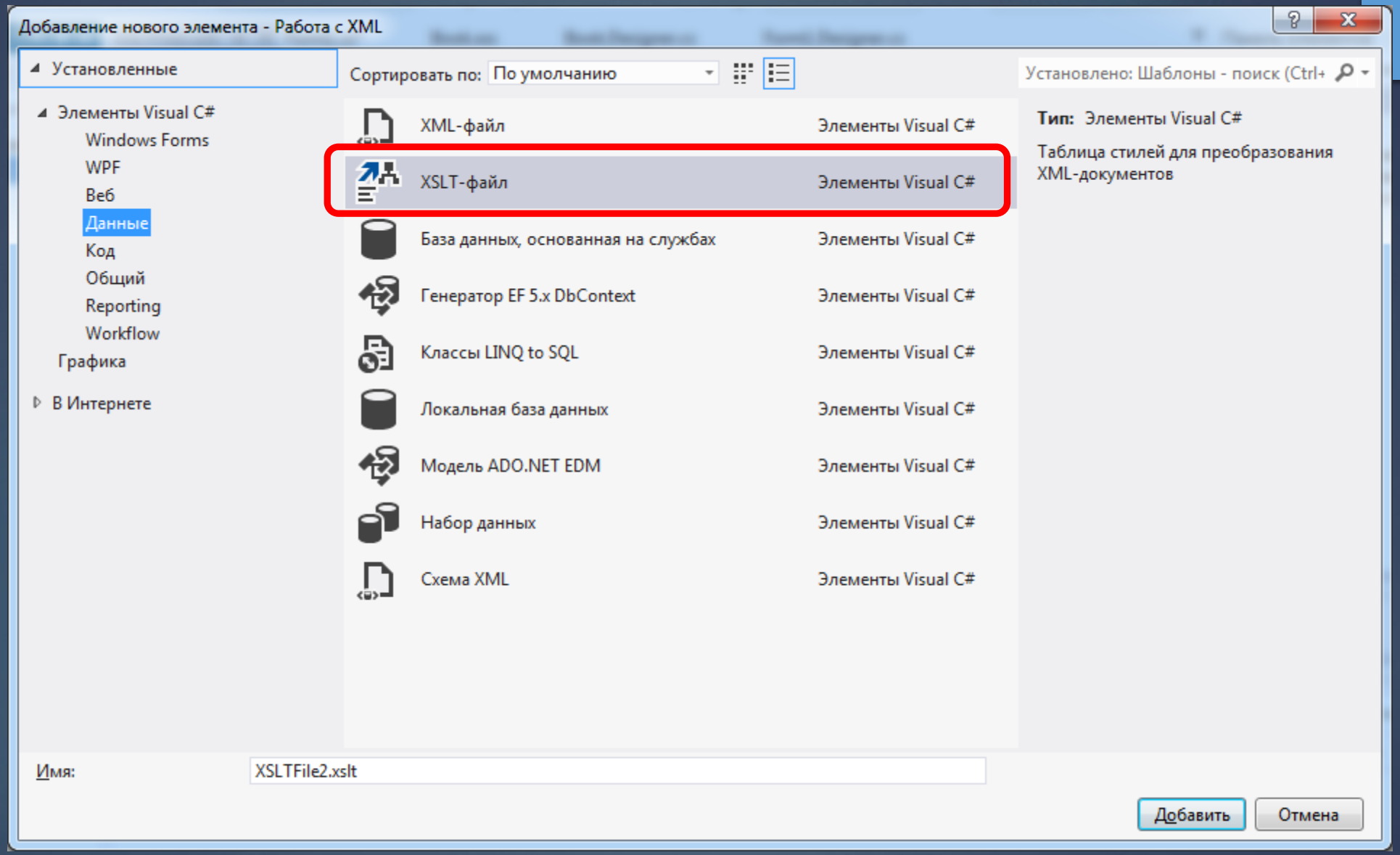
http://www.codenet.ru/webmast/xml/xslt/w3c.php

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:output method="html" />
  <xsl:template match="/">
    <html>
      <head>
        <title>
          <xsl:value-of select="/Book/Title"/>
        </title>
      </head>
      <body>
        <b>Автор  :</b><xsl:value-of select="/Book/Author"/><br></br>
        Главы:
        <table border="1">
          <xsl:for-each select="/Book/Chapters/Chapter"
            <tr>
              <td>
                <xsl:value-of select="."/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
       <p> Количество глав <xsl:value-of select="count(/Book/Chapters/Chapter)"/></p>
      </body>
    </html>
  </xsl:template>
```

XSLT создавался для применения в XSL, языке стилей для XML

XSLT - язык преобразований XML документов в другие XML документы

Используемая в XSLT модель данных аналогична используемой в XPath с определенными дополнениями


Добавление нового элемента - Работа с XML
Установленные
Элементы Visual C#
Windows Forms
WPF
Веб
Данные
Код
Общий
Reporting
Workflow
Графика
В Интернете
Сортировать по: По умолчанию
Установлено: Шаблоны - поиск (Ctrl+
XML-файл Элементы Visual C#
XSLT-файл Элементы Visual C#
База данных, основанная на службах Элементы Visual C#
Генератор EF 5.x DbContext Элементы Visual C#
Классы LINQ to SQL Элементы Visual C#
Локальная база данных Элементы Visual C#
Модель ADO.NET EDM Элементы Visual C#
Набор данных Элементы Visual C#
Схема XML Элементы Visual C#
Тип: Элементы Visual C#
Таблица стилей для преобразования XML-документов
Имя:
XSLTFile2.xslt
Добавить
Отмена

```
using System.Xml.Xsl;
using System.Diagnostics;
```

```
XslCompiledTransform transform = new XslCompiledTransform();
transform.Load("XSLTFile1.xslt");
transform.Transform("my.xml", "my.html");

Process.Start("my.html");
```

Автор :Ivan Ivanov
Главы:

1-A
2-B
3-C
4-D

Количество глав 4

LINQ to XML

```
using System.Xml.Linq
```

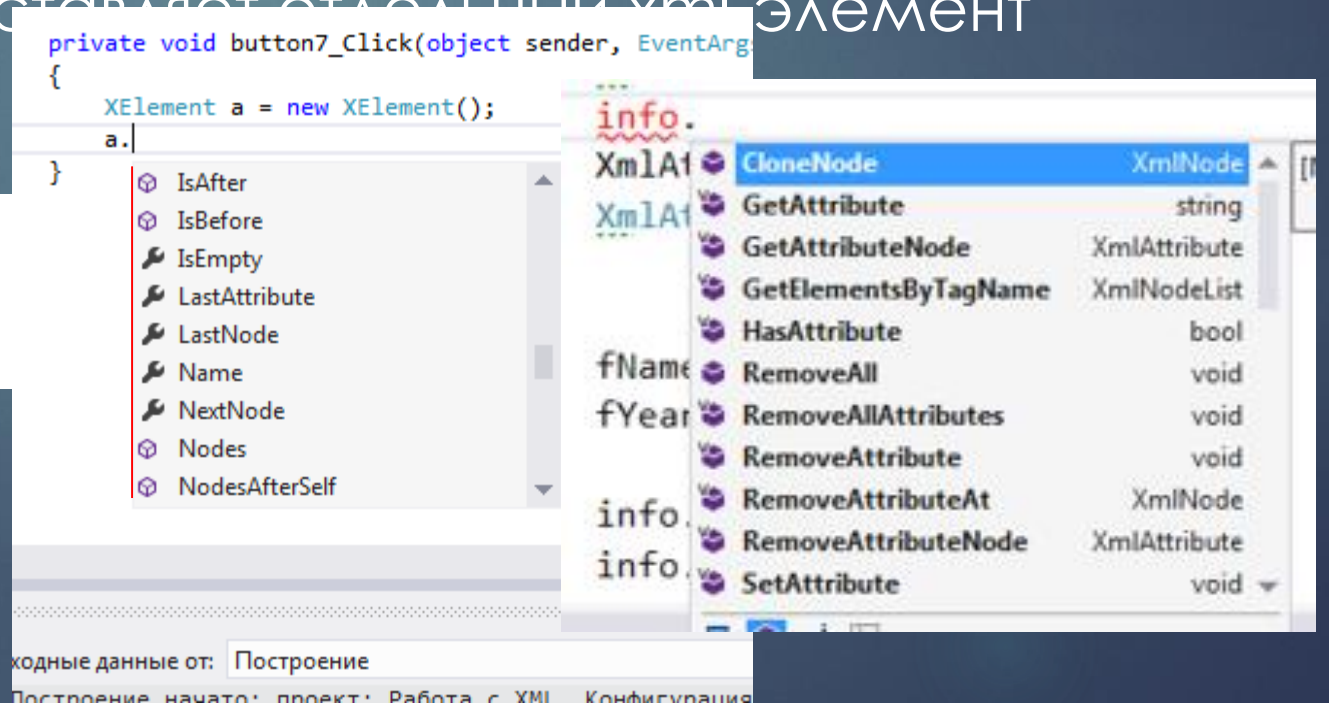
XAttribute: представляет атрибут xml-элемента

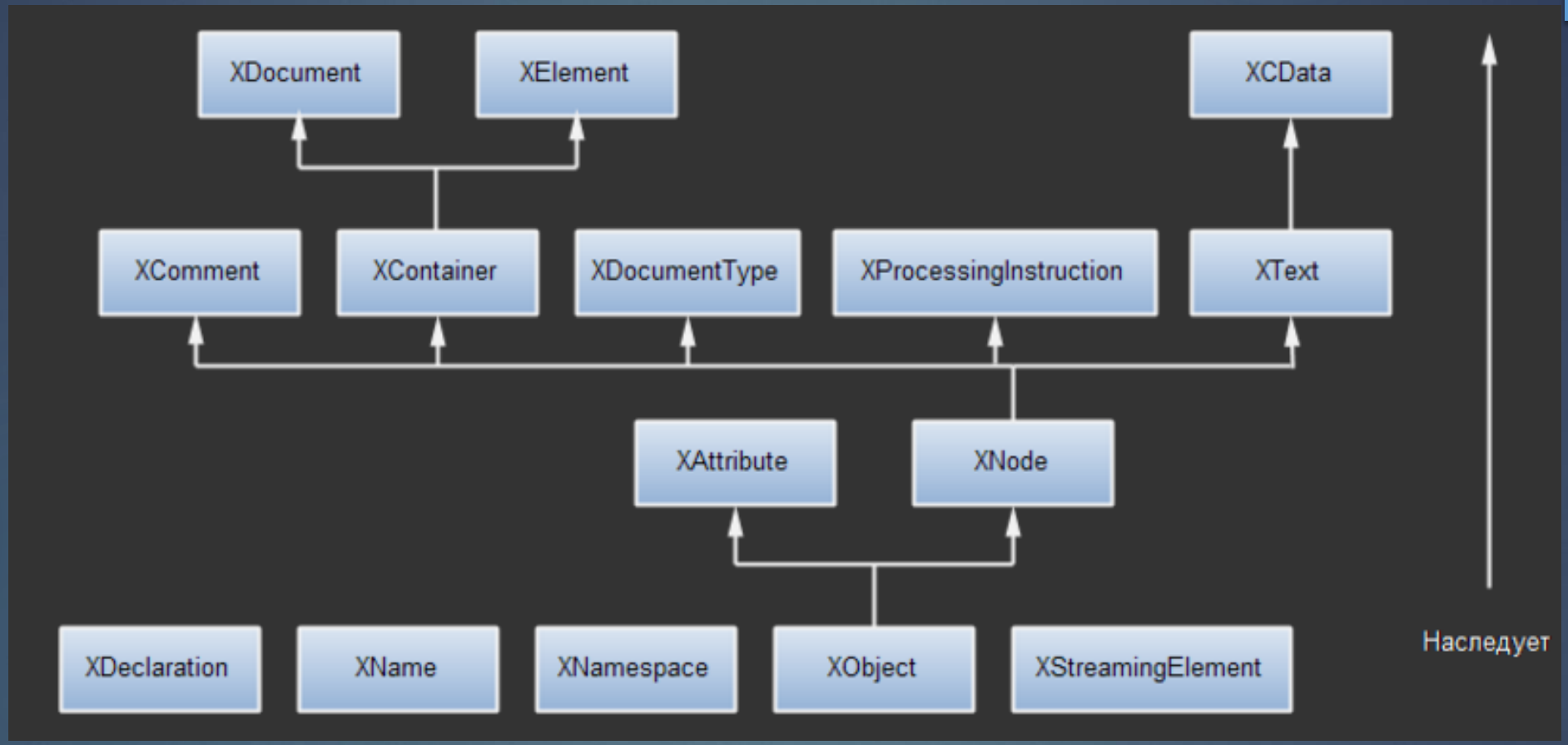
XComment: представляет комментарий

XDocument: представляет весь xml-документ

XElement: представляет отдельный xml-элемент

позволяет получать вложенные элементы управлять ими



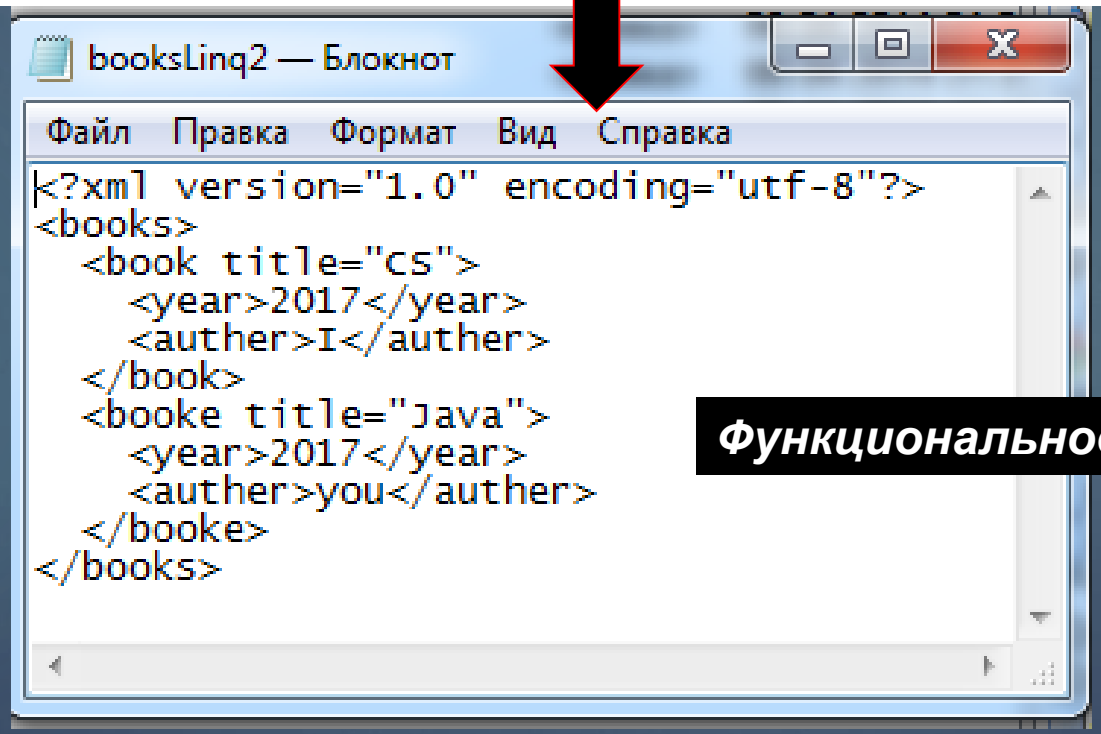

XDocument
XElement
XCData
XComment
XContainer
XDocumentType
XProcessingInstruction
XText
XAttribute
XNode
XDeclaration
XName
XNamespace
XObject
XStreamingElement
Наследует

Используя функциональность LINQ to XML создадать новый XML-документ

```
XDocument xdoc = new XDocument();
 // создаем первый элемент
 XElement book = new XElement("book");
 // создаем атрибут
 XAttribute bookNameAttr = new XAttribute("name", "CS");
 XElement bookPriceElem = new XElement("price", "40000");
 // добавляем атрибут и элементы в первый элемент
 book.Add(bookNameAttr);
 book.Add(bookPriceElem);

 // создаем корневой элемент
 XElement books = new XElement("books");
 // добавляем в корневой элемент
 books.Add(book);
// добавляем корневой элемент в документ
 xdoc.Add(books);
 //сохраняем документ
 xdoc.Save("booksLinq.xml");
```

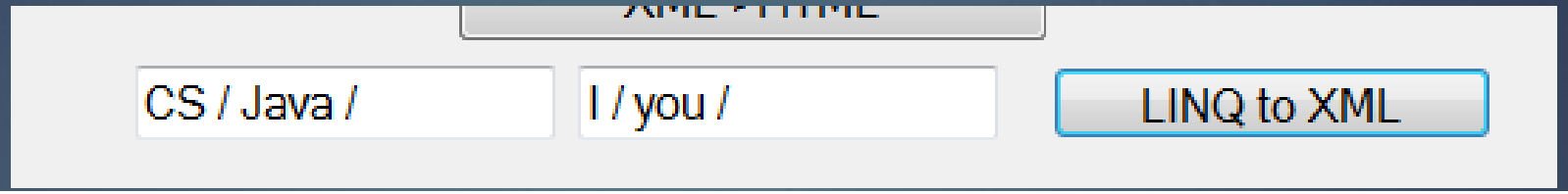
```
<?xml version="1.0" encoding="utf-8"?>
<books>
  <book name="CS">
    <price>40000</price>
  </book>
</books>
```

```
XDocument xdoc2 = new XDocument(new XElement("books",
                  new XElement("book",
                      new XAttribute("title", "CS"),
                      new XElement("year", "2017"),
                      new XElement("auther", "I")),
                  new XElement("booke",
                      new XAttribute("title", "Java"),
                      new XElement("year", "2017"),
                      new XElement("auther", "you"))));
                          xdoc2.Save("booksLinq2.xml");
```



Функциональное конструирование

Выборка элементов в LINQ to XML




```
XDocument xdoc3 = XDocument.Load("booksLinq2.xml");
            var result = from xe in xdoc3.Descendants("book")
                         where xe.Element("year").Value == "2017"
                         select new
                         {
                             title = xe.Attribute("title").Value,
                             author = xe.Element("author").Value,
                         };

            foreach (var item in result)
            {
                this.textBox5.Text += item.title;
                this.textBox6.Text += item.author;
                this.textBox5.Text += " / ";
                this.textBox6.Text += " / ";
            }
```

Переберем элементы и выведем их значения