# NET.C#.09
# XML. Работа с XML в .Net

# XML Основы

**XML**

**eXtensible Markup Language**

**Что такое XML ?**

**XML это язык разметки (как HTML)**

**XML был спроектирован для хранения данных (не отображения)**

**XML – теги не предопределены. Вы можете определять свои собственные теги**

**XML является самоописываемым языком**

# XML Основы

## XML Ничего Не Делает

**XML был создан для структурирования, хранения и передачи информации.**

```
//сообщение (message)
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

**Здесь есть «отправитель» сообщения, «получатель» , «заголовок» и «тело» сообщения**

# XML Основы

**В XML Вы можете изобретать свои теги**

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

**Теги <to> и <from> не определены ни в одном XML-стандарте. Эти теги введены самим автором XML-документа.**

**XML позволяет автору определять свои собственные теги и структуру документа**

# XML Основы

**XML есть везде**

**Как использовать XML?**

XML Simplifies Data Sharing

**XML-данные хранятся в простом текстовом формате. Это позволяет обеспечить независимый от программно-аппаратного обеспечения способ хранения даных.**

**This makes it much easier to create data that can be shared by different applications.**

**XML упрощает передачу данных**

# Пример XML-документа

**XML использует простой синтаксис:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

**Первая строка - это XML declaration. Определяет XML-версию (1.0) и кодировку символов.**

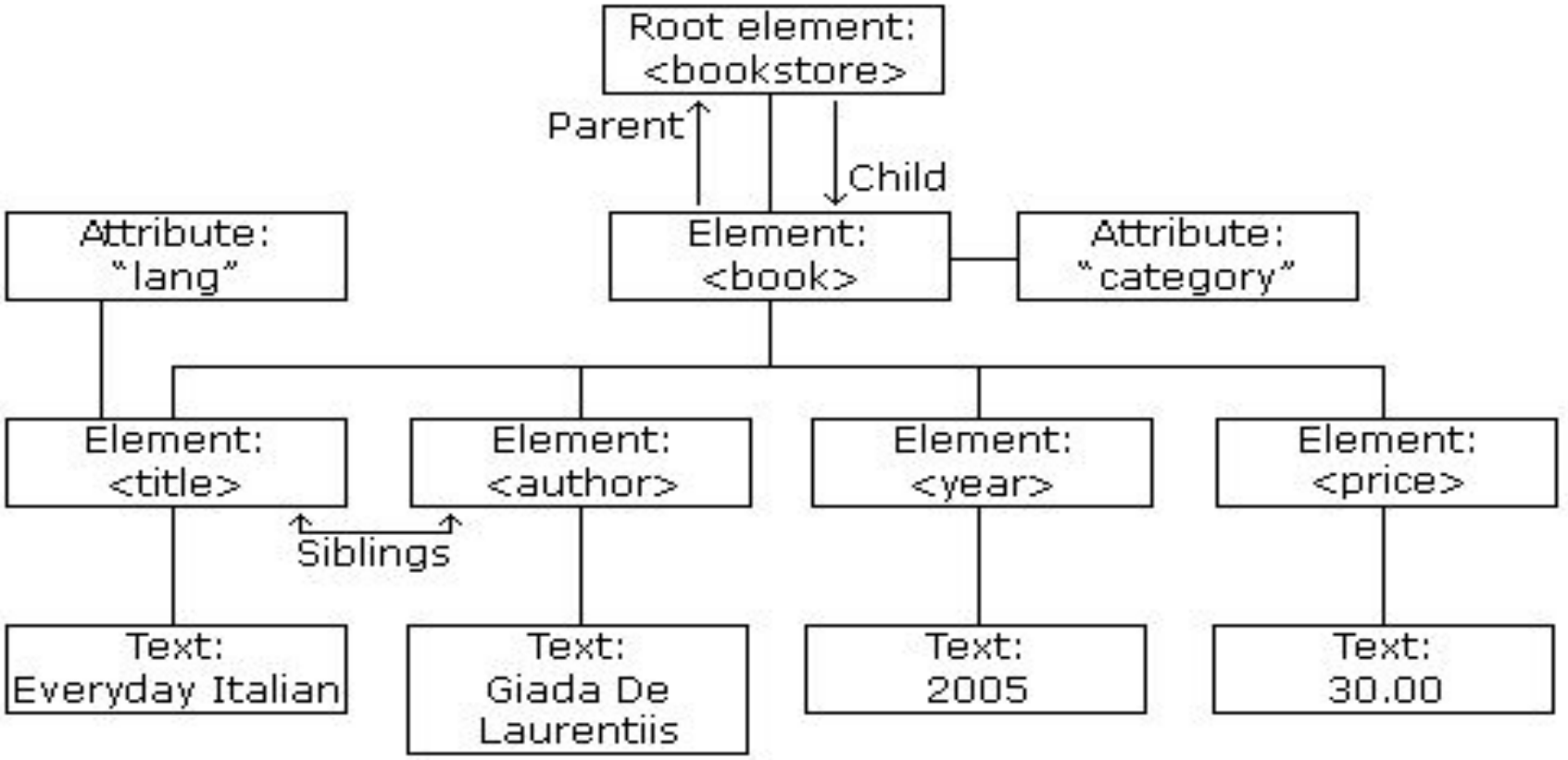
**Следующая строка - root element (корневой элемент) документа (это как сказать, что этот документ – есть «записка» (note)**

**Следующие 4 строки - это дочерние элементы (child elements) корня (to, from, heading, body)**

# XML Documents. Структура

## Структура XML-документа древовидная (Tree Structure)

**XML документ должен содержать корневой элемент, который является родителем всех остальных элементов. Все элементы могут иметь под-элементы (child elements):**



```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Корневой элемент - <bookstore>

Все элементы <book> содержатся внутри элемента <bookstore>

У элемента <book> есть 4 children: <title>, <author>, <year>, <price>

# XML. Правила синтаксиса

Все XML элементы должны иметь закрывающий тег

```
<message>This is correct</message>
```

XML-теги чувствительны к регистру. Тег <Letter> и тег <letter> являются различными тегами.

XML-элементы должны быть вложенными

```
<b><i>This text is bold and italic</i></b>
```

XML-документ должен иметь Root Element (корневой элемент)

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

# XML. Правила синтаксиса

XML-атрибуты заключаются в кавычки

```
//неверно
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

```
//верно
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Комментарии пишутся также, как в HTML.

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

# XML-атрибуты

XML-элементы могут иметь атрибуты. Атрибуты предоставляют дополнительную информацию об элементе

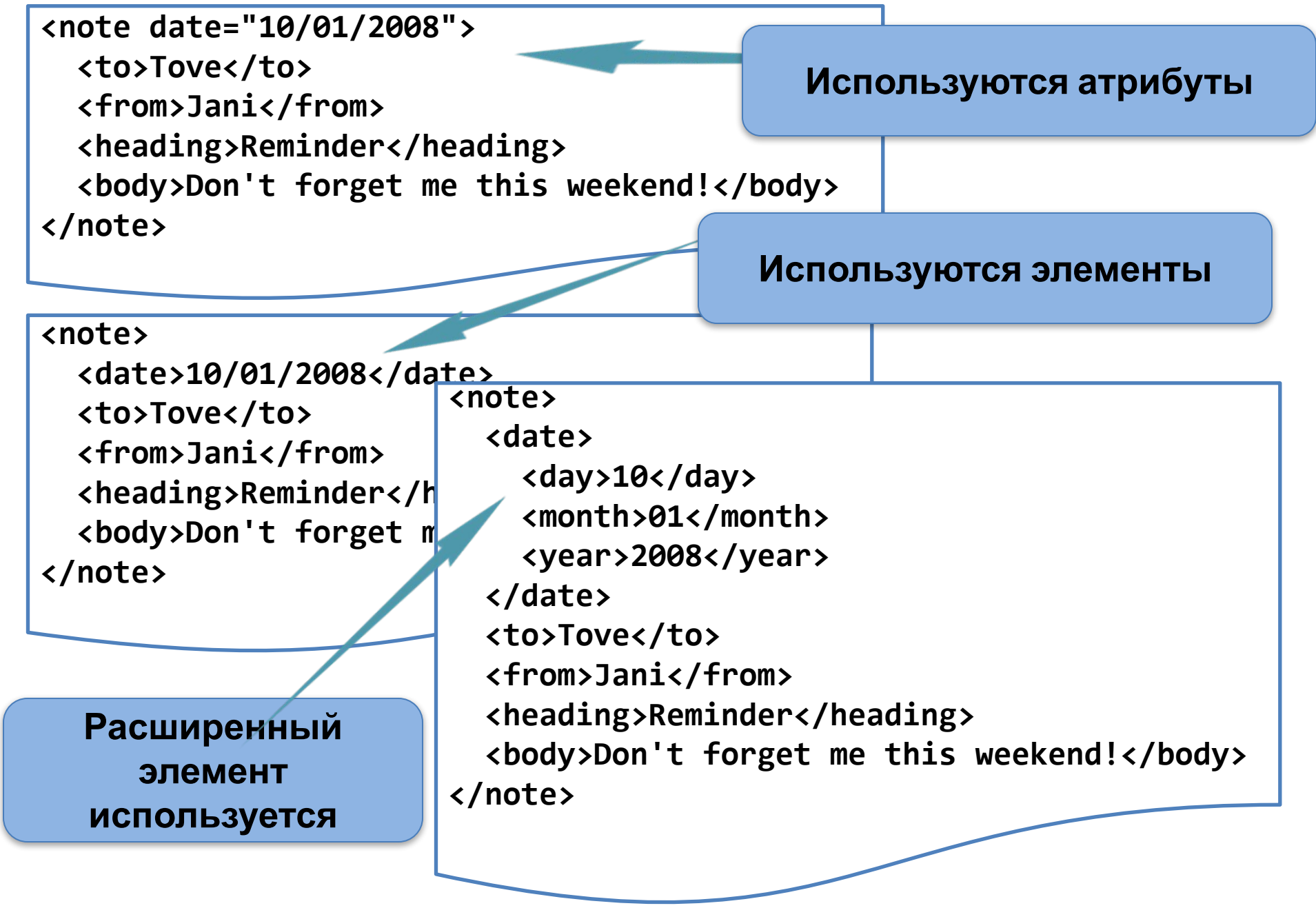
```
<file type="gif">computer.gif</file>

<person sex="female">
```

Нет правил, определяющих когда использовать атрибуты, а когда элементы

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

Оба примера несут одну и ту же информацию


<note date="10/01/2008">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
Используются атрибуты
<note>
<date>10/01/2008</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</h
<body>Don't forget m
</note>
Используются элементы
<note>
<date>
<day>10</day>
<month>01</month>
<year>2008</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
Расширенный элемент используется

# Некторые проблемы использования атрибутов

Атрибуты не могут быть многозначными (элементы могут)

Атрибуты не могут образовывать tree-структуру (элементы могут)

Атрибуты не просто расширить (при изменениях)

**Атрибуты сложно читать и поддерживать.**
**Используйте элементы для данных.**
**Используйте атрибуты для информации, которая не относится к данным.**

# Работа с XML в C#. XML DOM

.NET framework предлагает классы для работы с XML documents (System.Xml namespace) .
.NET использует XML Document Object Model (DOM), которяа представлена набором классов, представляющих различные части XML document ( и сам документ).

| Class | **Описание** |
|---|---|
| XmlNode | Представляет узел в дереве XML document. (элементы, атрибуты, комментарии). |
| XmlDocument | Представляет XML document. Может использоваться для считывания XML-файла или сохранения. |
| XmlElement | Представляет XML element. |
| XmlAttribute | Представляет XML attribute. |
| XmlText | Представляет текст внутри XML element. |
| XmlComment | Представляет XML comment. |

# XML DOM

XmlDocument class представляет XML document и все его содержимое

| Method | Description |
|---|---|
| CreateAttribute | Создает XmlAttribute. |
| CreateElement | Создает XmlElement. |
| CreateTextNode | Создает XmlText с соответствующим текстом. |
| Load | Считывает XML document из файла . |
| Save | Сохраняет содержимое XmlDocument в файл. |

XmlElement class представляет один XML element.

| Method | Description |
|---|---|
| GetAttribute | Возвращает значение атрибута. |
| GetAttributeNode | Возвращает XmlAttribute |

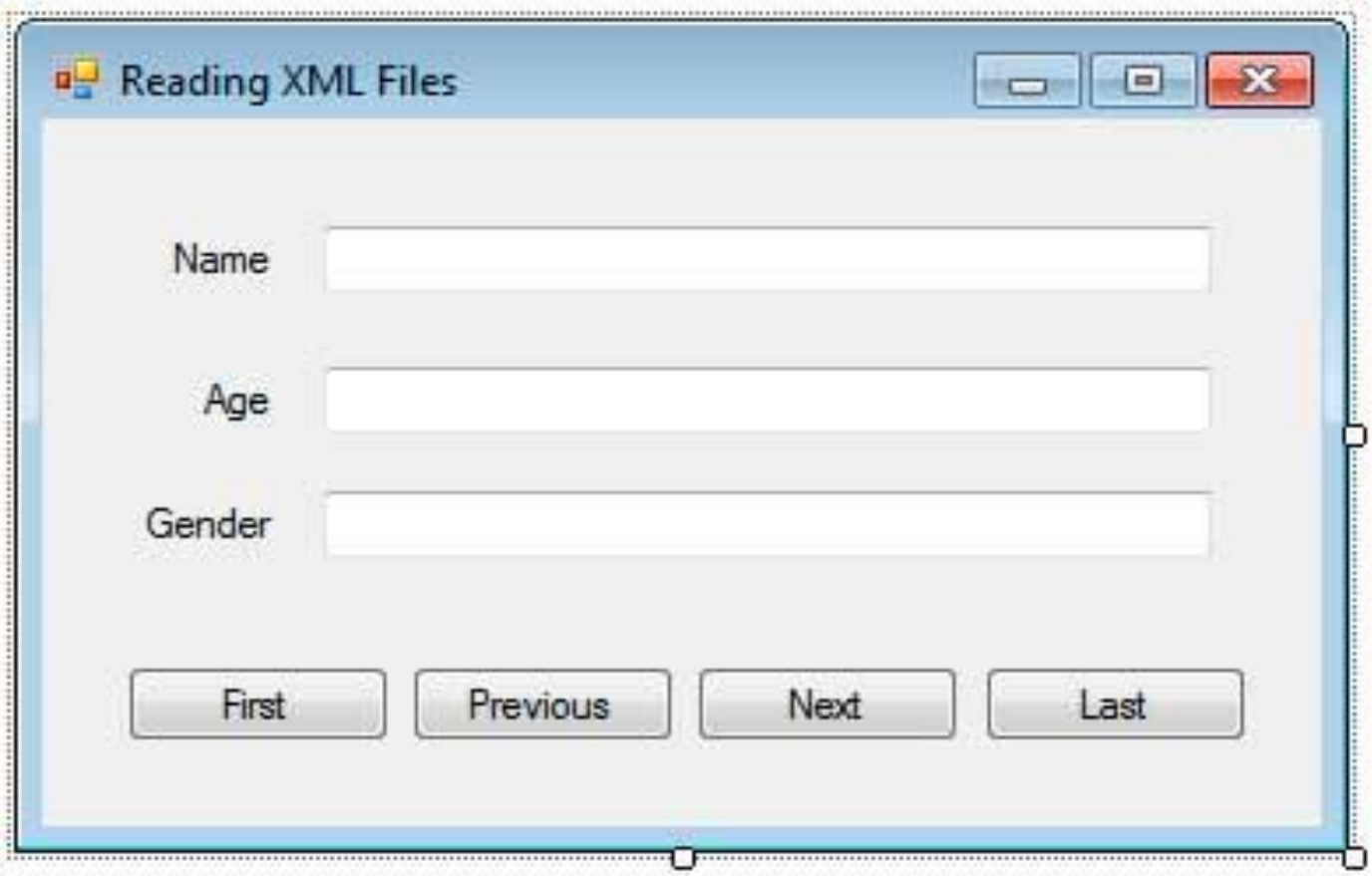
```
using System.Xml;
//add a private XmlDocument member and a string indicating the path
// of the XML file
private XmlDocument doc;
private const string PATH = @"C:\sample.xml";
//create an xml document
doc = new XmlDocument();
//If there is no current file, then create a new one
//Create neccessary nodes
XmlDeclaration declaration = doc.CreateXmlDeclaration("1.0",
   "UTF-8", "yes");
XmlComment comment = doc.CreateComment("This is an XML Generated
   File");
 XmlElement root = doc.CreateElement("Persons");
 XmlElement person = doc.CreateElement("Person");
 XmlAttribute name = doc.CreateAttribute("name");
 XmlElement age = doc.CreateElement("Age");
 XmlElement gender = doc.CreateElement("Gender");
```

```
//Add the values for each nodes
name.Value = textBoxName.Text;
age.InnerText = textBoxAge.Text;
gender.InnerText = textBoxGender.Text;
//Construct the document
doc.AppendChild(declaration);
doc.AppendChild(comment);
doc.AppendChild(root);
root.AppendChild(person);
person.Attributes.Append(name);
person.AppendChild(age);
person.AppendChild(gender);
doc.Save(PATH);
```

```
//If there is already a file
//Load the XML File
doc.Load(PATH);
//Get the root element
XmlElement root = doc.DocumentElement;
XmlElement person = doc.CreateElement("Person");
XmlAttribute name = doc.CreateAttribute("name");
XmlElement age = doc.CreateElement("Age");
XmlElement gender = doc.CreateElement("Gender");
//Add the values for each nodes
name.Value = textBoxName.Text;
age.InnerText = textBoxAge.Text;
gender.InnerText = textBoxGender.Text;
//Construct the Person element
person.Attributes.Append(name); person.AppendChild(age);
person.AppendChild(gender);
//Add the New person element to the end of the root element
root.AppendChild(person);
//Save the document
doc.Save(PATH);
```

# Чтение из XML-файла

**Рассмотрим пример приложения, которое читает данные из XML-файла.**



```
//Добавляем поля в класс Form1
private XmlDocument doc;
private XmlElement root;
private XmlElement currentPerson;
private const string PATH = @"..\..\sample.xml";
private int current = 0; private int max;

//Добаляем метод, который будет отображать текущую запись из
   файла
private void ShowDetails(XmlElement currentPerson)
{
textBoxName.Text = currentPerson.Attributes["name"].Value;
textBoxAge.Text =
   currentPerson.GetElementsByTagName("Age")[0].InnerText;
textBoxGender.Text =
   currentPerson.GetElementsByTagName("Gender")[0].InnerText;
 }
```

```
//Добавляем обработчик события Load формы
private void Form1_Load(object sender, EventArgs e)
 {
  doc = new XmlDocument();
  doc.Load(PATH);
  //Получаем корневой элемент
  root = doc.DocumentElement;
  //Определяем максимальное количество записей
  max = root.GetElementsByTagName("Person").Count - 1;
  //получаем запись по текущему индексу
  currentPerson = (XmlElement)root.ChildNodes[current];
  //Выводим информацию на форму
  ShowDetails(currentPerson);
 }
```

```
//Добавляем обработчик для кнопки “First”
private void buttonFirst_Click(object sender, EventArgs e)
{current = 0; currentPerson = (XmlElement)root.ChildNodes[current];
 ShowDetails(currentPerson);}

// Добавляем обработчик для кнопки “Previous”
private void buttonPrevious_Click(object sender, EventArgs e)
{ current = (current - 1 < 0) ? 0 : current - 1;
  currentPerson = (XmlElement)root.ChildNodes[current];
   ShowDetails(currentPerson); }
// Добавляем обработчик для кнопки “Next”
private void buttonNext_Click(object sender, EventArgs e)
 { current = (current + 1 > max) ? max : current + 1;
   currentPerson = (XmlElement)root.ChildNodes[current];
   ShowDetails(currentPerson); }
// Добавляем обработчик для кнопки “Last”
private void buttonLast_Click(object sender, EventArgs e)
{ current = max;
  currentPerson = (XmlElement)root.ChildNodes[current];
   ShowDetails(currentPerson); }
```

# Использование XMLPath для выбора узла

XPath это специальный язык запросов, который используется для выбора узлов XML-документа.

Используется два метода для XPath-языка:
XmlNode.SelectNodes() - возвращает XmlNodeList, который содержит все узлы, соответствующие XPath-строке.
XmlNode.SelectSingleNode() – возвращает один узел.

Например, необходимо получить из документа возраст (age) всех персон:

```
XmlDocument document = new XmlDocument();
document.Load("Persons.xml");
XmlNodeList nodes =
document.DocumentElement.SelectNodes("/Persons/Person/Age");
foreach(XmlNode node in nodes) { textBoxResult.Text +=
node.InnerText + "\r\n"; }
```

# Использование XMLPath для выбора узла

**Выбрать текущий узел:**

```
XmlNode current = document.DocumentElement.SelectSingleNode("."); }
```

**Выбрать все элементы <Person>, которые являются детьми элемента <Persons> :**

```
XmlNodeList personNodes =
document.DocumentElement.SelectNodes("/Persons/Person");
```

**Найти всех Peson мужского пола (Gender=Male) :**

```
XmlNodeList personNodes =
document.DocumentElement.SelectNodes("//Person[Gender='Male']");
```

Спасибо за внимание