LINQ to XML. Классы модели X-DOM

Основные классы модели X-DOM

Описания основных классов объектной модели X-DOM расположены в порядке наследования. В этих описаниях слова «предок» и «потомок» относятся не к иерархии наследования классов, а к *иерархии расположения объектов этих классов в дереве объектов, моделирующем XML-документ* (для указания иерархии наследования классов используется слово «наследник»).

В данном разделе основное внимание уделяется тому, как в модели X-DOM реализуются *связи* между компонентами XML-документа. Методы, связанные с «внешними» действиями (разбор, загрузка и сохранение XML-документа или его элемента), а также средства, предназначенные для работы со *значениями* атрибутов и элементов, описываются в разделе «Дополнительные возможности технологии LINQ to XML».

Для любого неабстрактного класса модели X-DOM предусмотрен конструктор с параметром `other`, тип которого совпадает с типом конструируемого объекта. Подобные конструкторы позволяют создавать *копии* уже существующих объектов. В дальнейшем, при описании конкретных классов, мы не будем повторно упоминать о наличии подобных конструкторов.

XObject

`XObject` — базовый тип в иерархии X-DOM; является абстрактным классом и наследуется непосредственно от класса `object`. Его основные свойства (оба read-only):

- `XDocument Document` — ссылка на документ;
- `XElement Parent` — ссылка на элемент-родитель.

Кроме того, в классе `XObject` определены два события:

- `Changed` (наступает при изменении данного объекта или его объектов-потомков);
- `Changing` (наступает непосредственно перед изменением данного объекта или его объектов-потомков).

Прямые наследники класса `XObject`: `XAttribute` и `XNode`.

XAttribute

`XAttribute` — класс для хранения информации об одном *атрибуте* элемента; наследуется от класса `XObject`. Его основные свойства:

- `XNode Name` — имя атрибута, read-only;
- `string Value` — значение атрибута, доступное для изменения.

Атрибуты любого элемента организованы в последовательность типа `IEnumerable<XAttribute>`; причем в самом классе `XAttribute` предусмотрены два свойства типа `XAttribute` (оба read-only), обеспечивающие перемещение по этой последовательности: `NextAttribute` (следующий атрибут) и `PreviousAttribute` (предыдущий атрибут). Если следующий или предыдущий атрибут отсутствует или текущий атрибут не связан с элементом (т. е. его свойство `Parent` равно `null`), то соответствующее свойство равно `null`.

Для *задания или изменения значения* атрибута можно, помимо свойства `Value`, использовать метод `void SetValue(object value)`. Особенности данного метода описаны в разделе «Дополнительные возможности технологии LINQ to XML».

Для *удаления атрибута* из списка атрибутов элемента XML достаточно вызвать для удаляемого атрибута метод `void Remove()` (если свойство `Parent` удаляемого атрибута равно `null`, то данный метод возбуждает исключение).

Для *создания* объекта типа `XAttribute` предусмотрен конструктор с параметрами `(XName name, object value)`. Параметр `value` обрабатывается так же, как в методе `SetValue`.

XNode

`XNode` — абстрактный базовый класс для всех компонентов XML-документа, кроме атрибутов. Подобные компоненты являются *узлами* дерева, порождающего XML-документ. К узлам дерева относятся не только его *элементы*, но и другие компоненты — текстовое содержимое элементов, комментарии и т. д.

Все узлы, имеющие общего родителя (*узлы-братья*, англ. sibling nodes), хранятся в последовательности типа `IEnumerable<XNode>`, и в классе `XNode` предусмотрены методы и свойства для работы с элементами последовательности, содержащей узел, для которого вызывается данное свойство или метод (будем называть такой узел self-узлом). Прежде всего, это два свойства типа `XNode` (read-only), обеспечивающие перемещение по этой последовательности: `NextNode` (следующий узел) и `PreviousNode` (предыдущий узел). Как и в случае аналогичных свойств класса `XAttribute`, если следующий или предыдущий узел для self-узла отсутствует или self-узел не имеет элемента-родителя, то соответствующее свойство равно `null`. Узлы-братья хранятся в виде *односвязного списка*, поэтому свойство `NextNode` вычисляется гораздо быстрее, чем `PreviousNode`.

Методы класса `XNode` связанные с последовательностью узлов-братьев:

- `bool IsAfter(XNode node)` — возвращает `true`, если self-узел расположен в последовательности после узла `node`, в противном случае (в частности, если self-узел не имеет родителя) возвращает `false`;
- `void AddAfterSelf(object content)` и `void AddAfterSelf(params object[] content)` — добавляет непосредственно после self-узла один или несколько новых узлов (если self-узел не имеет родителя, то возбуждается исключение `InvalidOperationException`);
- `IEnumerable<XNode> NodesAfterSelf()` — возвращает последовательность узлов-братьев, расположенных в последовательности после self-узла;
- `IEnumerable<XElement> ElementsAfterSelf([XName name])` — возвращает последовательность *элементов*-братьев, расположенных в последовательности после self-узла; вариант метода с параметром `name` оставляет в возвращаемой последовательности только элементы с именем `name`.

Имеются «парные» методы, в которых слово «`After`» заменено словом «`Before`». Они работают аналогично, но анализируют, добавляют или возвращают узлы, *предшествующие* self-узлу.

Методы для *удаления* или *замены* self-узла:

- `void Remove()` — удаляет self-узел из последовательности его узлов-братьев;
- `void ReplaceWith(object content)` и `void ReplaceWith(params object[] content)` — заменяют self-узел в последовательности его узлов-братьев на один или несколько узлов, указанных в качестве параметра `content`.

Если свойство `Parent` self-узла равно `null`, то вызов методов `Remove` или `ReplaceWith` приводит к возбуждению исключения `InvalidOperationException`.

Отметим также метод `IEnumerable<XElement> Ancestors([XName name])`, возвращающий последовательность всех элементов, являющихся *предками* self-узла, начиная с ближайшего предка, т. е. *родителя* (сам self-узел в эту последовательность не входит, потому что его тип может отличаться от типа `XElement`). Вариант метода с параметром `name` оставляет в возвращаемой последовательности только тех предков self-узла, которые имеют имя `name`.

Таким образом, в классе `XNode` предусмотрена функциональность, связанная не только с данным узлом XML-документа, но и с его узлами-*братьями* и узлами-*предками*.

XText

XText — простейший узел XML-документа, представляющий собой обычный *текст*. Наследуется от класса XNode. Имеет изменяемое свойство string Value.

Для создания объекта типа XText предусмотрен конструктор с параметром string value. Обычно этот конструктор явно не вызывается, так как если в качестве очередного узла XML-дерева указывается строковое выражение (или выражение, которое может быть преобразовано к строковому), то оно автоматически преобразуется к типу XText.

При работе со свойством Value *не следует* в качестве значений специальных символов использовать их предопределенные или нумерованные символьные сущности. Например, если узел XText должен содержать значение «&», то это значение следует задавать обычным образом: "&" (при визуализации XML-дерева с данным узлом его значение будет автоматически преобразовано к предопределенной сущности &):

```
var e = new XElement("demo", "&");
Console.WriteLine(e); // <demo>&amp;</demo>
```

Еще один пример:

```
var e = new XElement("demo", "<\">");
Console.WriteLine(e); // <demo>&lt;"&gt;</demo>
```

XCData

XCData — узел XML-документа с текстовым содержимым, которое сохраняется в документе *буквально*, без привлечения сущностей-заместителей. Наследуется от класса XText. Подобно классу XText имеет изменяемое свойство string Value и конструктор с параметром string value. При записи данного узла в XML-документ используются теги специального вида:

```
var e = new XElement("demo", new XCData("<\">"));
Console.WriteLine(e); //<demo><![CDATA[<">]]></demo>
```

Если не учитывать особый формат представления узлов типа XCData в XML-документе, работа с ними ничем не отличается от работы с обычными текстовыми узлами типа XText.

XComment

XComment — узел XML-документа, представляющий собой *комментарий*. Наследуется от класса XNode. Имеет изменяемое свойство string Value и конструктор с параметром string value. Текст комментария игнорируется при разборе XML-документа, поэтому он может содержать практически любые символы и их комбинации, которые никогда не будут заменяться на предопределенные или нумерованные сущности:

```
var e = new XElement("demo", new XComment("<\">"));
Console.WriteLine(e); // <demo><!--<">--></demo>
```

Единственное исключение делается при попытке указать в тексте комментария завершающий тег комментария -->; в этом случае между соседними дефисами добавляется пробел.

XProcessingInstruction

XProcessingInstruction — узел XML-документа, содержащий так называемые *инструкции обработки* (processing instructions) — вспомогательные данные, не относящиеся к содержимому XML-документа, но доступные для использования каким-либо *процессором XML*, т. е. программой, предназначенной для анализа или преобразования XML-документов.

Инструкции обработки в XML-документе заключаются в специальные теги <? и ?> и состоят из двух элементов:

- target — имя получателя (обязательный элемент);
- data — данные для обработки (необязательный элемент).

Класс XProcessingInstruction наследуется от класса XNode и имеет изменяемые свойства Target и Data типа string. Его конструктор также принимает два параметра target и data, которые не могут быть равны null (параметр data может быть произвольной строкой, в том числе пустой, а параметр target должен удовлетворять правилам именования элементов XML).

Примеры:

```
var e = new XElement("demo", new XProcessingInstruction
  ("SetMark", "Value='&' Count=1"));
Console.WriteLine(e);
  // <demo><?SetMark Value='&' Count=1?></demo>
e = new XElement("demo", new XProcessingInstruction
  ("SetMark", ""));
Console.WriteLine(e); // <demo><?SetMark?></demo>
```

XContainer

XContainer — абстрактный класс для реализации *XML-контейнеров* — узлов дерева XML, которые могу содержать другие узлы-потомки. Наследуется от класса XNode. В дополнение к унаследованным от XNode свойствам имеет свойства FirstNode и LastNode типа XNode (read-only), возвращающие первый и последний *дочерний узел* self-узла соответственно.

Для *добавления дочерних узлов* класс XContainer имеет методы Add и AddFirst, которые позволяют добавлять один или несколько дочерних узлов в конец или начало списка дочерних узлов соответственно; оба метода имеют параметр content типа object или params object[].

Для *удаления или замены всех дочерних узлов* класс XContainer имеет методы RemoveNodes (без параметров; удаляет всех дочерние узлы) и ReplaceNodes (с параметром content типа object или params object[]; заменяет все дочерние узлы на один или несколько узлов, указанных в параметре content).

Имеется несколько методов, позволяющих получить различные варианты *последовательностей*, связанных с узлами-потомками данного XML-контейнера:

- Nodes() и DescendantNodes() — возвращают последовательность типа IEnumerable<XNode>; в случае метода Nodes последовательность содержит *дочерние узлы* (т. е. потомки первого уровня вложенности), в случае DescendantNodes — *узлы-потомки* (любого уровня вложенности); порядок возвращаемых узлов соответствует порядку, в котором они расположены в XML-документе;
- Elements([XName name]) и Descendants([XName name]) — возвращают последовательность типа IEnumerable<XElement>, в случае метода Elements последовательность содержит *дочерние элементы*, в случае Descendants — *элементы-потомки* (любого уровня вложенности); при наличии параметра name в возвращаемую последовательность включаются только те элементы-потомки self-узла, которые имеют имя name.

Метод Element(XName name) возвращает первый дочерний элемент XML-контейнера с указанным именем name (или null, если ни одного требуемого элемента не найдено).

XElement

XElement — основной класс модели X-DOM, предназначенный для хранения *именованного элемента* XML-документа. Наследуется от XContainer.

Это единственный вид узла XML-дерева, который может иметь атрибуты, поэтому класс XElement содержит набор свойств и методов, предназначенных для работы с последовательностью атрибутов (все свойства read-only):

- bool HasAttributes — равно true, если self-элемент содержит хотя бы один атрибут;
- XAttribute FirstAttribute и XAttribute LastAttribute — возвращают первый и, соответственно, последний атрибут из последовательности атрибутов self-элемента; если элемент не содержит атрибутов, возвращают null;

- XAttribute Attribute(XName name) — возвращает атрибут self-элемента с указанным именем name (или null, если атрибут с указанным именем не существует);
- IEnumerable<XAttribute> Attributes([XName name]) — возвращает последовательность атрибутов self-элемента; если указан параметр name, то последовательность содержит только атрибут с именем name (или является пустой);
- void RemoveAttributes() — удаляет все атрибуты self-элемента;
- void ReplaceAttributes(object content) и void ReplaceAttributes(params object[] content) — заменяют все атрибуты self-элемента на один или несколько атрибутов, указанных в параметре content (в данном параметре можно указывать не только атрибуты, но и *узлы*, которые будут добавлены в последовательность дочерних узлов self-элемента);
- void SetAttributeValue(XName name, object value) — метод, позволяющий создавать, изменять и удалять атрибуты self-элемента. Если параметр value равен null, то атрибут с именем name удаляется; в противном случае, если атрибут с именем name отсутствует, то он создается, иначе его значение заменяется на value (в качестве value можно указывать любой объект, кроме объектов-наследников класса XObject; по поводу преобразования этих объектов к строковому типу см. пункт «Работа со значениями атрибутов и элементов»).

Для добавления атрибута к элементу, кроме метода SetAttributeValue, можно использовать метод Add, поскольку в классе XElement этот метод может использоваться не только для добавления дочерних узлов, но и для добавления атрибутов (в то же время метод AddFirst не может использоваться для добавления к элементу атрибутов).

Класс XElement имеет изменяемое свойство string Value, которое возвращает объединенное текстовое содержимое самого self-элемента и его потомков (всех уровней). При этом предопределенные и нумерованные символьные сущности, используемые в XML для обозначения символов, заменяются на сами эти символы. Если же изменить это свойство, то все дочерние узлы будут заменены на единственный дочерний узел типа XText, содержащий указанную строку. Свойству Value нельзя присвоить значение null. Если у элемента и его потомков отсутствует текстовое содержимое (в частности, если элемент представлен единственным комбинированным тегом вида <name />), то свойство Value возвращает *пустую строку*.

Метод void SetValue(object value) также позволяет изменить свойство Value, при этом параметр value может иметь любой тип (кроме типов-наследников класса XObject, при использовании которых возбуждается исключение ArgumentException). Перед присваиванием свойству Value параметр value преобразуется к строковому типу (см. пункт «Работа со значениями атрибутов и элементов»).

Свойство bool IsEmpty (read-only) позволяет распознать элемент, представленный комбинированным тегом; в этом случае оно будет равно true (наличие атрибутов в расчет не принимается). Например, для элемента свойство IsEmpty будет равно true, а для элемента <a> — false.

Для работы с дочерними элементами в классе XElement предусмотрены дополнительное свойство и методы:

- bool HasElements (read-only) — равно true, если self-элемент содержит хотя бы один дочерний элемент;
- void SetElementValue(XName name, object value) — позволяет создавать, изменять и удалять дочерние элементы self-элемента. Если параметр value равен null, то дочерний элемент с именем name удаляется; в противном случае, если дочерний элемент с именем name отсутствует, то он создается и его свойству Value присваивается значение value, иначе его свойство Value заменяется на value (в качестве параметра value можно указывать любой объект — см. пункт «Работа со значениями атрибутов и элементов»).

Есть также методы класса XElement, имеющие отношение ко всему содержимому элемента: и атрибутам, и дочерним узлам:

- void RemoveAll() — удаляет у self-элемента все его атрибуты и дочерние узлы;
- void ReplaceAll(object content) и void ReplaceAll(params object[] content) — заменяют все атрибуты и дочерние узлы self-элемента на один или несколько атрибутов и дочерних узлов, указанных в параметре content.

Имеются модификации унаследованных методов, возвращающих *последовательности* узлов или элементов. Имена этих модификаций оканчиваются на «AndSelf»; все они отличаются от своих исходных вариантов тем, что включают в возвращаемую коллекцию сам self-элемент:

- AncestorsAndSelf([XName name]) — модификация метода Ancestors из класса XNode;
- DescendantNodesAndSelf() — модификация метода DescendantNodes из класса XContainer;
- DescendantsAndSelf([XName name]) — модификация метода Descendants из класса XContainer.

Для создания объекта типа XElement предусмотрено несколько конструкторов со следующими наборами параметров (параметр name определяет имя создаваемого элемента):

- XElement(XName name) — создает элемент без содержимого;
- XElement(XName name, object content) и XElement(XName name, params object[] content) — создает элемент с содержимым content (одним или несколькими дочерними узлами и/или атрибутами).

Если требуется представить элемент в виде одного комбинированного тега, необходимо, чтобы при его создании содержимое content не включало данных, отличных от null (если задать в содержимом content хотя бы одну пустую строку, то элемент будет отображаться в виде открывающего и закрывающего тега). Элемент отображается в виде комбинированного тега также в случае, если для него вызван метод RemoveNodes или RemoveAll.

XDocument

XDocument — класс, предназначенный для хранения «оболочки» дерева XML. Наследуется от XContainer. В любом XML-документе имеется единственный объект типа XDocument, причем любой объект-наследник XObject имеет свойство Document, ссылающееся на документ, с которым связан данный объект. По сравнению с классом XElement класс XDocument содержит немного «собственных» свойств и методов. Перечислим основные из них:

- XDeclaration Declaration — изменяемое свойство; предназначено для хранения объявления XML;
- XElement Root (read-only) — содержит корневой элемент XML-документа.

Объект XDocument не является родителем своего корневого элемента, поскольку родителем любого узла дерева XML может быть только *элемент*, т. е. объект типа XElement. Поэтому свойство Parent корневого элемента равно null.

Для создания объекта типа XDocument предусмотрено несколько конструкторов со следующими наборами параметров:

- XDocument() — создает «пустой» XML-документ;

- XDocument(params object[] content) — создает XML-документ с содержимым content (в наборе параметров должно быть не более одного объекта типа XElement, определяющего корневой элемент, однако может присутствовать, например, любое число XML-комментариев);
- XDocument(XDeclaration declaration, params object[] content) — действует подобно ранее рассмотренному конструктору, но дополнительно позволяет определить *объявление XML* (параметр declaration).

Из примера использования модели W3C DOM нетрудно заметить, что объект типа XmlDocument является основным, «цементирующим» компонентом документа XML. В частности, в модели W3C DOM любой компонент XML-документа создается с помощью одного из методов объекта типа XmlDocument. Это означает, что в программе нельзя создать фрагмент XML-документа, не связанный с объектом типа XmlDocument, что в ряде случаев является неудобным ограничением.

Модель X-DOM свободна от этого ограничения. Хотя в ней и предусмотрен класс XDocument, играющий ту же роль, что и класс XmlDocument в модели W3C DOM, создавать фрагменты XML-документа можно без его участия.

XDocumentType

XDocumentType — специализированный узел XML-документа, содержащий *объявление типа документа* (type document declaration; не следует путать объявление типа документа с объявлением XML, для которого предусмотрен особый класс XDeclaration). Наследуется от XNode. Этот узел может содержаться только в прологе XML-документа, поэтому его можно добавлять (причем в единственном числа) только непосредственно в объект типа XDocument. Данный класс связан со специальными возможностями XML-документов, которые мы не будем рассматривать.

Вспомогательные классы модели X-DOM

Вспомогательные классы модели X-DOM не являются наследниками базового класса XObject. Они используются в качестве параметров во многих методах основных классов.

XDeclaration

XDeclaration — вспомогательный класс модели X-DOM, предназначенный для хранения *объявления XML* (XML declaration). Наследуется от класса object, не содержит дополнительных методов. Для настройки трех компонентов объявления класс XDeclaration имеет три изменяемых свойства:

- string Version — свойство, определяющее версию стандарта XML; при сохранении XML-документа в качестве версии стандарта XML *всегда* используется значение "1.0";
- string Encoding — свойство, определяющее кодировку XML-документа, которая учитывается при записи документа на диск; примерами кодировок являются "utf-8" и "windows-1251" (вариант "utf-8" используется по умолчанию);
- string Standalone — свойство, которое определяет, является ли данный XML-документ *автономным* (или он связан с какими-либо внешними документами); данное свойство может принимать только два непустых допустимых значения: "yes" (автономный документ) и "no"; в случае любого другого значения (в частности, пустой строки или null) компонент Standalone в объявлении сохраненного XML-документа не указывается.

Конструктор объекта XDeclaration принимает три строковых параметра — значения компонентов объявления version, encoding и standalone соответственно. Если какой-либо параметр является пустой строкой или равен null, то при сохранении XML-документа в качестве соответствующего компонента объявления будет использовано значение по умолчанию (хотя соответствующие свойства объекта XDeclaration будут содержать значения, указанные в конструкторе, и эти же значения будут использованы при преобразовании объекта XDeclaration к строковому представлению с помощью метода ToString).

XName

XName — вспомогательный запечатанный класс, предназначенный для хранения *имен элементов* XML-документа. Наследуется от класса object, реализует интерфейс IEquitable<XName> (таким образом, имена могут сравниваться на равенство). К типу XName может быть неявно преобразована строка типа string; обратное преобразование должно выполняться явно, с помощью метода ToString.

В классе XName предусмотрены специальные механизмы для поддержки имен, включающих *пространства имен XML* (XML namespaces). В частности, имеются три свойства (read-only), позволяющие получить различные компоненты имени:

- string LocalName — локальное имя,
- XNamespace Namespace — необязательное пространство имен, присоединяемое к локальному имени для получения полного имени,
- string NamespaceName — имя пространства имен.

Полное имя, связанное с объектом XName (qualified name), может быть получено с помощью метода ToString.

XNamespace

XNamespace — вспомогательный запечатанный класс, предназначенный для работы с *пространствами имен XML*. В качестве имени пространства имен XML обычно используется *универсальный идентификатор ресурса* (Uniform Resource Identifier, URI), связанный с той организацией, которая разработала спецификацию соответствующего XML-документа (хотя допустимым является любое строковое выражение).

Класс XNamespace наследуется от класса object. К типу XNamespace может быть неявно преобразована строка типа string; обратное преобразование должно выполняться явно, с помощью метода ToString.

Для класса XNamespace определена операция «+», применяемая к операндам типа XNamespace (первый операнд — пространство имен) и string (второй операнд — локальное имя). Результатом этой операции будет полное имя XML-элемента (типа XName).

Класс XNamespace имеет единственное экземплярное свойство string NamespaceName (read-only), возвращающее имя пространства имен, и три статических свойства (также read-only), возвращающих специальные виды пространств имен:

- XNamespace None — пустое пространство имен; его имя — пустая строка;
- XNamespace Xml — пространство имен, связанное с XML URI; его имя — http://www.w3.org/XML/1998/namespace;
- XNamespace Xmlns — пространство имен, связанное с xmlns URI; его имя — http://www.w3.org/2000/xmlns/.

Вопросы, связанные с использованием пространств имен, обсуждаются в разделе «Дополнительные возможности технологии LINQ to XML».