需要编写 XML 编辑器、转换系统或报告编写器这类程序的 LINQ to XML 开发人员通常需要编写在比元素和属性更细的粒度下运行的程序。 开发人员通常需要在节点级别上工作，操作文本节点、处理指令和注释。 本文提供了关于节点级编程的信息。

**示例：XDocument 的子节点的 Parent 属性值设置为 null**

[Parent](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xobject.parent) 属性包含父 [XElement](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xelement)，而不是父节点。 [XDocument](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xdocument) 的子节点没有父 [XElement](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xelement)。 它们的父级是文档，因此这些节点的 [Parent](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xobject.parent) 属性设置为 null。

下面的示例演示这一操作：

C#复制

XDocument doc = XDocument.Parse(@"<!-- a comment --><Root/>");

Console.WriteLine(doc.Nodes().OfType<XComment>().First().Parent == null);

Console.WriteLine(doc.Root.Parent == null);

该示例产生下面的输出：

输出复制

True

True

**示例：添加文本可能会或可能不会创建新的文本节点**

在许多 XML 编程模型中，相邻的文本节点始终会合并到一起。 这有时也称为文本节点的规范化。 LINQ to XML 不会规范化文本节点。 如果向同一个元素添加两个文本节点，则会产生相邻文本节点。 但如果将指定内容添加为字符串而不是 [XText](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xtext) 节点，则 LINQ to XML 可能会将该字符串与相邻的文本节点合并在一起。 下面的示例演示这一操作。

C#复制

XElement xmlTree = new XElement("Root", "Content");

Console.WriteLine(xmlTree.Nodes().OfType<XText>().Count());

// this doesn't add a new text node

xmlTree.Add("new content");

Console.WriteLine(xmlTree.Nodes().OfType<XText>().Count());

// this does add a new, adjacent text node

xmlTree.Add(new XText("more text"));

Console.WriteLine(xmlTree.Nodes().OfType<XText>().Count());

该示例产生下面的输出：

输出复制

1

1

2

**示例：将文本节点值设置为空字符串不会删除节点**

在某些 XML 编程模型中，文本节点保证不包含空字符串。 原因是这种文本节点对 XML 的序列化没有影响。 但由于可以存在相邻文本节点这一相同的原因，如果通过将文本节点的值设置为空字符串来移除文本节点中的文本，则不会删除文本节点本身。

C#复制

XElement xmlTree = new XElement("Root", "Content");

XText textNode = xmlTree.Nodes().OfType<XText>().First();

// the following line doesn't cause the removal of the text node.

textNode.Value = "";

XText textNode2 = xmlTree.Nodes().OfType<XText>().First();

Console.WriteLine(">>{0}<<", textNode2);

该示例产生下面的输出：

输出复制

>><<

**示例：具有一个空文本节点的元素的序列化方式不同于没有文本节点的元素**

如果一个元素仅包含一个空的子文本节点，则会用长标记语法 <Child></Child> 序列化该元素。 如果一个元素不包含任何子节点，则会用短标记语法 <Child /> 序列化该元素。

C#复制

XElement child1 = new XElement("Child1",

new XText("")

);

XElement child2 = new XElement("Child2");

Console.WriteLine(child1);

Console.WriteLine(child2);

该示例产生下面的输出：

XML复制

<Child1></Child1>

<Child2 />

**示例：命名空间是 LINQ to XML 树中的属性**

即使命名空间声明与特性具有完全相同的语法，在某些编程接口（如 XSLT 和 XPath）中，也不会将命名空间声明视为属性。 但在 LINQ to XML 中，命名空间存储为 XML 树中的 [XAttribute](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xattribute) 对象。 如果循环访问包含命名空间声明的元素的属性，则命名空间声明为返回的集合中的一项声明。 [IsNamespaceDeclaration](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xattribute.isnamespacedeclaration) 属性 (property) 指示某一属性 (attribute) 是否为命名空间声明。

C#复制

XElement root = XElement.Parse(

@"<Root

xmlns='http://www.adventure-works.com'

xmlns:fc='www.fourthcoffee.com'

AnAttribute='abc'/>");

foreach (XAttribute att in root.Attributes())

Console.WriteLine("{0} IsNamespaceDeclaration:{1}", att, att.IsNamespaceDeclaration);

该示例产生下面的输出：

输出复制

xmlns="http://www.adventure-works.com" IsNamespaceDeclaration:True

xmlns:fc="www.fourthcoffee.com" IsNamespaceDeclaration:True

AnAttribute="abc" IsNamespaceDeclaration:False

**示例：XPath 轴方法不返回 XDocument 的子文本节点**

LINQ to XML 允许 [XDocument](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xdocument) 具有子文本节点，但这些文本节点只能包含空白。 不过，XPath 对象模型没有将空白符添加为文档子节点。因此，如果使用 [Nodes](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xcontainer.nodes) 轴循环访问 [XDocument](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xdocument) 的子级，将会返回空白符文本节点。 然而，如果使用 XPath 轴方法循环访问 [XDocument](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xdocument) 的子级，就不会返回空白符文本节点。

C#复制

// Create a document with some white space child nodes of the document.

XDocument root = XDocument.Parse(

@"<?xml version='1.0' encoding='utf-8' standalone='yes'?>

<Root/>

<!--a comment-->

", LoadOptions.PreserveWhitespace);

// count the white space child nodes using LINQ to XML

Console.WriteLine(root.Nodes().OfType<XText>().Count());

// count the white space child nodes using XPathEvaluate

Console.WriteLine(((IEnumerable)root.XPathEvaluate("text()")).OfType<XText>().Count());

该示例产生下面的输出：

输出复制

3

0

**XDocument 的 XML 声明节点是属性，而不是子节点**

在循环访问 [XDocument](https://learn.microsoft.com/zh-cn/dotnet/api/system.xml.linq.xdocument) 的子节点时，将看不到 XML 声明对象。 它是文档的属性，不是文档的子节点。

C#复制

XDocument doc = new XDocument(

new XDeclaration("1.0", "utf-8", "yes"),

new XElement("Root")

);

doc.Save("Temp.xml");

Console.WriteLine(File.ReadAllText("Temp.xml"));

// this shows that there is only one child node of the document

Console.WriteLine(doc.Nodes().Count());

该示例产生下面的输出：

输出复制

<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<Root />

1