# XML mit C#







## 1 Vorbemerkungem

2 Aufgaben für XML-API's:

- Vorhandene Dokumente lesen und auswerten (Parsen)
- Vorhandene Dokumente verändern, bzw. neue Dokumente erzeugen

XML-Parser

Vorhandene Dokumente lesen und verstehen

2 Arten von Parsern

| | Modell 1 | Modell 2 |
|---|---|---|
| Kennzeichnung | Erst komplettes Dokument einlesen und anschließend auswerten, | Dokument beim Einlesen bereits auswerten |
| Programmier-modell | Aufsuchen spezieller Knoten unter Verwendung von Xpath-Pattern und Bearbeitung dieser Knoten | Erstellung von „Event-Routinen", die vom Parser immer dann aufgerufen werden, wenn ein Knoten vollständig eingelesen wurde. |
| Programmierung | Einfach | Schwierig |
| Speicherbedarf | Hoch | Gering |
| Schelligkeit | Niedrig | Hoch |

Hier: Modell 1

## 2 Klassenübersicht

`XmlNode` ist die Elterklasse aller Knoten in Xml-Dokumenten.

Liste aller „Knotenklassen“ nebst Klassenhierarchie (fett markiert: die für uns wichtigsten)

**`XmlNode`**

- **`XmlDocument`**
- **`XmlAttribute`**
- `XmlEntity`
- `XmlDocumentFragment`
- `XmlNotation`
- `XmlLinkedNode`
  - `XmlProcessingInstruction`
  - **`XmlElement`**
  - **`XmlDeclaration`**
  - `XmlCharacterData`
  - `XmlDocumentType`
  - `XmlEntityReference`





## 2.1 XmlNode

Namensraum `System.Xml`

Anmerkung: Viele Properties und Methoden sind als `virtual` deklariert. Das bedeutet, dass sie von abgeleiteten Klassen überschreiben werden können. Dies betrifft hier lediglich spezielle Aspektet der Implementierung. Die grundsätzliche Funktionsweise (Außenwirkung) entspricht aber weiterhin der Beschreibung in `XmlNode`.

Properties

```
public virtual boolean HasChildNodes
```

Liefert `true`, wenn es mindestens einen Kindknoten (Element-Knoten, Text-Knoten, Kommentar-Knoten, Processing-Instruction-Knotens, keine Parameter-Knoten!) gibt, andernfalls `false`.

```
public virtual XmlNode FirstChild
public virtual XmlNode LastChild
```

Liefert den ersten/letzten Kindknoten. Existiert kein Kindknoten so wird `null` zurückgegeben.

```
public virtual XmlNode NextSlibing
public virtual XmlNode PreviousSlibing
```

Liefert den vorhergehenden/nachfolgenden Nachbarknoten auf derselben Hierarchiebene. Wenn dieser nicht existiert wird `null` zurückgegeben.

```
public virtual XmlNode Parent
```

Liefert den Elterknoten

```
public virtual XmlNodeList ChildNodes
```

Liefert alle Kindknoten als `XmlNodeList` zurück, durch die anschließend navigiert werden kann.

```
public virtual String InnerXml
public virtual String OuterXml
```

Gibt das XML-Teildokument unterhalb (`OuterXml`: einschließlich) des aktuellen Knotens aus.

```
public virtual String InnerText
```

Liefert den Knotentyp-abhängigen String-Wert des Knotens („String-Wert“ siehe XSLT-Kapitel).





```
public virtual String Value
```

Wie `InnerText`. Unterschied: Der `Value` von Elementknoten ist der Leerstring

## Methoden

```
public virtual XmlNode AppendChild(XmlNode newChild)
public virtual XmlNode PrependChild(XmlNode newChild)
```

Fügt den Knoten `newChild` als neuen Kindknoten hinzu (am Ende bzw. am Anfang). Rückgabewert: Eingefügter-Knoten im Kontext des umgebenden XML-Dokuments?

```
public Type GetType()
```

Liefert den Knotentyp zurück. Diese können sein:

```
XmlDocument
XmlElement
XmlProcessingInstruction
XmlComment
XmlDeclaration
```

```
public virtual XmlNode InsertAfter(
    XmlNode newChild, XmlNode refChild)
```

Fügt den Knoten `newChild` als neuen Kindknoten hinzu und zwar hinter den Kindknoten `refChild.` Rückgabewert: Eingefügter-Knoten im Kontext des umgebenden XML-Dokuments?

```
public virtual XmlNode InsertBefore(XmlNode newChild,
XmlNode refChild)
```

Fügt den Knoten `newChild` als neuen Kindknoten hinzu und zwar vor den Kindknoten `refChild`. Rückgabewert: Eingefügter-Knoten im Kontext des umgebenden XML-Dokuments?

```
public virtual XmlNode ReplaceChild(XmlNode newChild,
XmlNode oldChild)
```

Ersetzt den Kindknoten `oldChild` durch den neuen Kindknoten `newChild`. Rückgabewert: Eingefügter-Knoten im Kontext des umgebenden XML-Dokuments?

```
public virtual XmlNode RemoveChild(XmlNode oldChild)
```





Entfernt den Kindknoten `oldChild`. Rückgabewert: Rausgelöster Knoten, der nicht sich nicht mehr im Kontext des umgebenden XML-Dokuments befindet.

```
public virtual XmlNodeList SelectNodes(String xpath)
```

Liefert eine Liste aller Knoten, die dem Xpath Select-Pattern `xpath` entsprechen.

```
public virtual XmlNode SelectSingleNode(String xpath)
```

Liefert den ersten Knoten, der dem Xpath Select-Pattern `xpath` entspricht

## 2.2 XmlNodeList

Abfragen, die mehr als einen Knoten liefern können, liefern eine Liste von Knoten: `XmlNodeList`.

Properties:

```
public abstract Int32 Count
```

Liefert die Anzahl der in der Liste befindlichen Knoten.

Methoden:

```
public abstract XmlNode Item(Int32 index)
```

Liefert den Knoten an Position `index` (0 .. Count-1)

## 2.3 XmlElement

Properties

```
public virtual boolean HasAttributes
```

```
public override XmlAttributeCollection Attributes
```

Methoden

```
public virtual XmlNodeList GetElementsByTagName(String
name)
```

Liefert als Knotenliste alle Nachfahrelemente mit Elementname `name` (Xpath-Ausdruck).

```
public virtual boolean HasAttribute(String name)
```





```
public virtual void RemoveAttribute(String name)
```

```
public virtual void SetAttribute(String name, String value)
```

Ändert den Wert des Attributs `name` auf `value`, bzw. legt ein neues Attribut an.

```
public virtual string GetAttribute(String name)
```

Liefert den Attribut-Wert des Attributs mit Attribut-Name = `name`

```
public virtual XmlAttribute GetAttributeNode(String name)
```

Liefert den Attribut-Knoten des Attributs mit Attribut-Name = `name`

## 2.4 XmlDocument

```
public virtual Load(String fileName)
```

Lädt das XML-Dokument aus der angegebenen Datei

```
public virtual Save(String fileName)
```

Speichert das XML-Dokument in die angegebene Datei

```
public XmlElement DocumentElement
```

Liefert das Wurzelelement des Dokuments

Methoden zur Generierung neuer Knoten

Die einzelnen Knoten-Klassen besitzen keine öffentlichen Konstruktoren. Statt dessen stellt die Klasse XmlDokument Methoden zur Generierung neuer Knoten zur Verfügung (Vollständige Liste, die für uns wichtigsten fett markiert)

```
CreateComment
CreateCDataSection
CreateDocumentFragment
CreateDocumentType
CreateElement
CreateProcessingInstruction
CreateTextNode
CreateXmlDeclaration
CreateWhitespace
CreateSignificantWhitespace
```

Die Rückgabewerte sind jeweils isolierte Knoten, die dann noch in das Dokument einzumontieren sind.

Beispiel für die vollständige Generierung eines Dokuments:





```
   XmlElement buch,titel,preis;

A  XmlDocument doc=new XmlDocument();
B  doc.AppendChild(
     doc.CreateXmlDeclaration("1.0","iso-8859-1","yes"));
C  doc.AppendChild(doc.CreateElement("Bucherliste"));

D  buch= doc.CreateElement("buch");
E  buch.SetAttribute("auflager","25");
F  doc.DocumentElement.AppendChild(buch);

   titel=doc.CreateElement("titel");
G  titel.InnerText="Herr der Ringe";
   buch.AppendChild(titel);

   preis=doc.CreateElement("preis");
H  preis.AppendChild(doc.CreateTextNode("26Euro"));
   buch.AppendChild(preis);

   doc.Save(@"c:\irgendwo\buecherliste.xml");
```

**A** erzeugt ein „leeres“ XMLDokument. `doc` zeigt auf den Wurzelknoten.

**B** .. erzeugt eine Xml-Deklaration und fügt sie als ersten Kindknoten des Wurzelknotens hinzu

**C** erzeugt einen Elementknoten und fügt diesen als zweiten Kindknoten des Wurzelknotens hinzu. Dieser Knoten wird damit zum Wurzelelement.

**D** erzeugt ein Element `buch`,

**E** .. für das Attribute `auflager="25" hinzu`

**F** .. und fügt das Element `buch` als erstes Kindelement dem Element `buecherliste` (Wurzelelement) hinzu

**G** zeigt, wie man in den Textinhalt eines Elements festlegt, das lediglich Textinhalt, d.h. lediglich einen einzigen Textknoten besitzen soll.

**H** zeigt als Alternative zu **G** wie man einem Element allgemein einen Textknoten hinzufügt.