Brug af XML i C#

Denne artikel vil vise lidt om hvordan man kan bruge XML i C#.

Den forudsætter kendskab til XML og C# men ikke til brug af XML i C#.

Der er en anden artikel med præcis samme indhold bare i VB.NET !

Skrevet den **15. Feb 2010** af **arne_v** I kategorien **Programmering / C#** | ⭐⭐⭐☆☆

Historie:
V1.0 - 05/11/2004 - original
V1.1 - 25/12/2008 - lidt småjusteringer og link til efterfølgeren
V1.2 - 14/02/2010 - smårettelser

Indledning

.NET har en glimrende support for XML i namespace
System.Xml og under namespaces.

Denne artikel vil vise nogle af mulighederne indenfor basal XML brug.

Jeg vil ikke forklare baggrund bag XML, XPath og XSLT. Det
må man finde et andet sted.

Artiklen http://www.eksperten.dk/artikler/627 "XML hvad, hvorfor og hvornår ?" giver en oversigt over nogle af begreberne.

Eksemplerne vil være console applikationer, men selve
XML klasserne og metoderne kan lige så godt bruges i
GUI applikationer eller web applikationer.

De fleste af kode eksemplerne vil bruge denne simple
XML fil.

test.xml

```
<?xml version='1.0' standalone='yes'?>
<biler>
    <fabrikat>
        <navn>Volvo</navn>
        <land>Sverige</land>
    </fabrikat>
    <fabrikat>
        <navn>Opel</navn>
        <land>Tyskland</land>
    </fabrikat>
    <fabrikat>
        <navn>Toyota</navn>
```

```
        <land>Japan</land>
    </fabrikat>
    <fabrikat>
        <navn>Fiat</navn>
        <land>Italien</land>
    </fabrikat>
</biler>
```

Læse XML fra fil

Det er meget nemt at læse et XML dokument ind fra en fil.

Man laver et nyt tomt XmlDocument og bruger Load metoden.

```
XmlDocument doc = new XmlDocument();
doc.Load(@"C:\test.xml");
```

Løbe gennem XML

Man kan finde alle tags med et bestemt navn med GetElementsByTagName metoden som returnerer en collection af noder.

```
XmlNodeList fabrikater = doc.GetElementsByTagName("fabrikat");
foreach(XmlNode fabrikat in fabrikater)
{
    string navn = fabrikat.ChildNodes[0].FirstChild.Value;
    string land = fabrikat.ChildNodes[1].FirstChild.Value;
    Console.WriteLine(navn + " " + land);
}
```

Manipulere XML

Man kan også udvælge noder med SelectNodes eller SelectSingleNode metoderne og et XPath udtryk.

XPath udtryk udvælger noder ved at angive stien til dem i en speciel syntax:
xxxx - finder elementer med navn xxxx
//xxxx/yyyy - finder elementer med navn yyyy under elementer med navn xxxx
xxxx[yyyy='abc'] - finder elementer med navn xxxx som har et under element med navn yyyy og en tekstværdi 'abc'
xxxx[@yyyy=123] - finder elementer med navn xxxx som har en attribut med navn yyyy og en talværdi 123

```
XmlNodeList volvo = doc.DocumentElement.SelectNodes("fabrikat[navn='Volvo']");
```

```
volvo[0].ChildNodes[1].FirstChild.Value = "USA";
XmlNodeList opel = doc.DocumentElement.SelectNodes("fabrikat[navn='Opel']");
opel[0].ChildNodes[1].FirstChild.Value = "USA";
```

Man kan tilføje noder til træet.

```
XmlNode hyundainavn = doc.CreateElement("navn");
hyundainavn.AppendChild(doc.CreateTextNode("Hyundai"));
XmlNode hyundailand = doc.CreateElement("land");
hyundailand.AppendChild(doc.CreateTextNode("Syd Korea"));
XmlNode hyundai = doc.CreateElement("fabrikat");
hyundai.AppendChild(hyundainavn);
hyundai.AppendChild(hyundailand);
doc.DocumentElement.AppendChild(hyundai);
```

Man kan også fjerne noder fra træet med element.RemoveChild(node) !

Skrive XML til fil

Det er meget nemt at skrive et XML dokument til en fil.

Man kalder bare Save metoden.

```
doc.Save(@"C:\test1.xml");
```

Opsamling

Vi kombinerer nu alle kode fragmenterne ovenfor til et
helt program.

```
using System;
using System.Xml;

class MainClass
{
    public static void Main(string[] args)
    {
        // Læse XML fra fil
        XmlDocument doc = new XmlDocument();
        doc.Load(@"C:\test.xml");
        // Løbe gennem XML
        XmlNodeList fabrikater = doc.GetElementsByTagName("fabrikat");
        foreach(XmlNode fabrikat in fabrikater)
        {
```

```
            string navn = fabrikat.ChildNodes[0].FirstChild.Value;
            string land = fabrikat.ChildNodes[1].FirstChild.Value;
            Console.WriteLine(navn + " " + land);
        }
        // Manipulere XML
        // 1. sæt Volvo og Opel til USA
        XmlNodeList volvo =
doc.DocumentElement.SelectNodes("fabrikat[navn='Volvo']");
        volvo[0].ChildNodes[1].FirstChild.Value = "USA";
        XmlNodeList opel =
doc.DocumentElement.SelectNodes("fabrikat[navn='Opel']");
        opel[0].ChildNodes[1].FirstChild.Value = "USA";
        // 2. tilføj Hyundai
        XmlNode hyundainavn = doc.CreateElement("navn");
        hyundainavn.AppendChild(doc.CreateTextNode("Hyundai"));
        XmlNode hyundailand = doc.CreateElement("land");
        hyundailand.AppendChild(doc.CreateTextNode("Syd Korea"));
        XmlNode hyundai = doc.CreateElement("fabrikat");
        hyundai.AppendChild(hyundainavn);
        hyundai.AppendChild(hyundailand);
        doc.DocumentElement.AppendChild(hyundai);
        // Skrive XML til fil
        doc.Save(@"C:\test1.xml");
    }
}
```

test1.xml kommer til at se ud som:

```
<?xml version="1.0" standalone="yes"?>
<biler>
  <fabrikat>
    <navn>Volvo</navn>
    <land>USA</land>
  </fabrikat>
  <fabrikat>
    <navn>Opel</navn>
    <land>USA</land>
  </fabrikat>
  <fabrikat>
    <navn>Toyota</navn>
    <land>Japan</land>
  </fabrikat>
  <fabrikat>
    <navn>Fiat</navn>
    <land>Italien</land>
  </fabrikat>
  <fabrikat>
    <navn>Hyundai</navn>
    <land>Syd Korea</land>
  </fabrikat>
</biler>
```

Skrive XML til fil på en anden måde

Man kan naturligvis lave et helt nyt XML dokument på samme måde som der ovenfor tilføjes elementer.

```
using System;
using System.Xml;

class MainClass
{
    public static void Main(string[] args)
    {
        XmlDocument doc = new XmlDocument();
        XmlNode root = doc.CreateElement("all");
        doc.AppendChild(root);
        for(int i = 0; i < 5; i++)
        {
            XmlNode elm = doc.CreateElement("one");
            elm.AppendChild(doc.CreateTextNode("Element #" + (i + 1)));
            root.AppendChild(elm);

        }
        doc.InsertBefore(doc.CreateXmlDeclaration("1.0", "UTF-8", "yes"),
doc.DocumentElement);
        doc.Save(@"C:\test2.xml");
    }
}
```

test2.xml kommer til at se ud som:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<all>
  <one>Element #1</one>
  <one>Element #2</one>
  <one>Element #3</one>
  <one>Element #4</one>
  <one>Element #5</one>
</all>
```

Men der findes en alternativ måde som nogen gange er mere hensigtsmæssig.

```
using System;
using System.IO;
```

```
using System.Text;
using System.Xml;

class MainClass
{
    public static void Main(string[] args)
    {
        XmlTextWriter xtw = new XmlTextWriter(@"C:\test3.xml", Encoding.UTF8);
        xtw.Formatting = Formatting.Indented;
        xtw.WriteStartDocument();
        xtw.WriteStartElement("all");
        for(int i = 0; i < 5; i++)
        {
            xtw.WriteStartElement("one");
            xtw.WriteString("Element #" + ( i + 1));
            xtw.WriteEndElement();
        }
        xtw.WriteEndElement();
        xtw.WriteEndDocument();
        xtw.Close();
    }
}
```

test3.xml kommer til at se ud som:

```
<?xml version="1.0" encoding="utf-8"?>
<all>
  <one>Element #1</one>
  <one>Element #2</one>
  <one>Element #3</one>
  <one>Element #4</one>
  <one>Element #5</one>
</all>
```

Bruge XSLT

Hvis man gerne vil manipulere sit XML dokument er det også en mulighed at bruge XSLT.

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Xsl;

class MainClass
{
    public static void Main(string[] args)
    {
```

```
        XmlDocument doc = new XmlDocument();
        doc.Load(@"C:\test.xml");
        XslTransform xslt = new XslTransform();
        xslt.Load(@"C:\test.xsl");
        XmlTextWriter wrt = new XmlTextWriter(new
StreamWriter(@"C:\test4.xml"));
        wrt.Formatting = Formatting.Indented;
        xslt.Transform(doc, null, wrt, null);
    }
}
```

test.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

<xsl:template match="biler">
<biler>
<xsl:apply-templates/>
</biler>
</xsl:template>

<xsl:template match="fabrikat">
<fabrikat>
<xsl:attribute name="land"><xsl:value-of select="land"/></xsl:attribute>
<xsl:value-of select="navn"/>
</fabrikat>
</xsl:template>

</xsl:stylesheet>
```

test4.xml kommer til at se ud som:

```
<biler>
  <fabrikat land="Sverige">Volvo</fabrikat>
  <fabrikat land="Tyskland">Opel</fabrikat>
  <fabrikat land="Japan">Toyota</fabrikat>
  <fabrikat land="Italien">Fiat</fabrikat>
</biler>
```

Nyt i .NET 2.0 og 3.5

http://www.eksperten.dk/artikler/1259 beskriver:
- hurtige readonly XPathDocument

- XsltTransform -> XsltCompiledTransform i .NET 2.0
- LINQ for XML i .NET 3.5

Kommentar af driis d. 04. Dec 2004 | 1

Absolut god artikel for begynderen mht. XML i C#. Gode eksempler.

Kommentar af simonvalter d. 07. Nov 2004 | 2

Gode eksempler til at komme igang med.

Kommentar af jesperhaun d. 11. Nov 2004 | 3

Nydeligt... Lige hvad man skal bruge for at komme i gang.

Kommentar af repsak d. 12. Dec 2004 | 4

Super - gennemgår dog kun 'in-memory' modellen

Kommentar af mungojerrie d. 10. Nov 2004 | 5

fin artikel

Kommentar af jesperwerge d. 18. Nov 2006 | 6

Perfekt - dejligt simple og ligetil, tak for hjælpen

Kommentar af 2c d. 11. Oct 2006 | 7

Meget præcis. Jeg kigger dog altid her, hver gang jeg vil vælge en node ud fra en atribut, og det kan man ikke finde her.
Eksempel:
doc.SelectSingleNode("//biler/fabrikat[@land='Tyskland']");

Kommentar af daxiez d. 27. Feb 2006 | 8

Helt perfekt artikel. havde alt hvad jeg skulle bruge !

Kommentar af _basil d. 22. Jul 2006 | 9

Kommentar af the_ghost d. 08. Nov 2004 | 10

God og Simpel artikel, nemt at finde ud af!

Kommentar af plugin- d. 31. May 2005 | 11

Som ny i sproget kunne jeg også godt have tænkt mig at hører hvorfor det er en god idé at bruge xml i sine programmer... ellers en god artikel med gode eksempler

Kommentar af visualdeveloper d. 20. Oct 2005 | 12

jeg kunne også godt tænke mig at høre om hvorfor man bruger xml i C#, og hvilke fordele der er ved det ! Ellers en rigtig god artikel !

Kommentar af jimgordon d. 04. Dec 2004 | 13

Perfekt XML miniartikel. Hvorfor gøre det sværer end det er.

Kommentar af zepton d. 07. Apr 2007 | 14

Kan helt klart anbefales.

Kommentar af bergses d. 28. Oct 2008 | 15

Perfekt artikel. Jeg har meget nytte af den. Som potientiel udvidelse kunne jeg godt tænke mig mere om manipuleringsmuligheder, f.eks. workarounds ved sletning eller opdatering af elementer.

Kommentar af zteff d. 15. Oct 2008 | 16

Rigtig fin artikel, der løber over de mest basale XML-operationer.