

Alphabetical sorting of the XmlNodeNodes

```
#region License and Copyright
```

```
/*
```

```
* Dotnet Commons Xml
```

```
*
```

```
*
```

```
* This library is free software; you can redistribute it  
and/or modify it
```

```
* under the terms of the GNU Lesser General Public License as  
published by
```

```
* the Free Software Foundation; either version 2.1 of the  
License, or
```

```
* (at your option) any later version.
```

```
*
```

```
* This library is distributed in the hope that it will be  
useful, but
```

```
* WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY
```

```
* or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser  
General Public License
```

```
* for more details.
```

```
*
```

```
* You should have received a copy of the GNU Lesser General  
Public License
```

```
* along with this library; if not, write to the
```

```
* Free Software Foundation, Inc.,
```

```
* 59 Temple Place,
```

```
* Suite 330,
```

```
* Boston,
```

```
* MA 02111-1307
```

```
* USA
```

```
*
```

```

*/
#endregion

using System;
using System.Collections;
using System.Collections.Specialized;
using System.Diagnostics;
using System.IO;
using System.Reflection;
using System.Text;
using System.Text.RegularExpressions;
using System.Xml;
using System.Xml.Xsl;
using System.Xml.Serialization;

//using Dotnet.Commons.Reflection;

namespace Dotnet.Commons.Xml
{
    ///
    ///
    /// This utility class contains wrapper functions that help to
    ease the handling and
    /// manipulation of Xml documents, such as adding an element,
    adding an attribute
    /// to an element, copying and cloning of nodes, etc.
    ///
    ///
    ///
    public abstract class XmlUtils
    {
        //
        #####
        ##### //
        // These code is derived from Mainsoft.com //
        //

```

<http://www.koders.com/csharp/fid439BB5BEF93D1AEFAF0B9206236AB0ECE49BC229.aspx>

```
//
```

```
#####
```

```
##### //
```

```
/// _____
```

```
///
```

```
/// Alphabetical sorting of the XmlNodeNodes
```

```
/// and their attributes in the .
```

```
///
```

```
/// to be sorted /// _____
```

```
public static void SortXml(XmlDocument document)
```

```
{
```

```
SortXml(document.DocumentElement);
```

```
}
```

```
/// _____
```

```
///
```

```
/// Inplace pre-order recursive alphabetical sorting of the  
XmlNodeNodes child
```

```
/// elements and .
```

```
///
```

```
/// The root to be sorted. /// _____
```

```
public static void SortXml(XmlNode rootNode)
```

```
{
```

```
SortAttributes(rootNode.Attributes);
```

```
SortElements(rootNode);
```

```
foreach (XmlNode childNode in rootNode.ChildNodes)
```

```
{
```

```
SortXml(childNode);
```

```
}
```

```
}
```

```
/// _____
```

```
///
```

```
/// Sorts an attributes collection alphabetically.
```

```

/// It uses the bubble sort algorithm.
///
/// The attribute collection to be sorted. ///
-----
public static void SortAttributes(XmlAttributeCollection
attribCol)
{
if (attribCol == null)
return;

bool hasChanged = true;
while (hasChanged)
{
hasChanged = false;
for (int i = 1; i < attribCol.Count; i++) { if
(String.Compare(attribCol[i].Name, attribCol[i-1].Name, true)
< 0) { //Replace attribCol.InsertBefore(attribCol[i],
attribCol[i - 1]); hasChanged = true; } } } } /// -----
----- ///
/// Sorts a alphabetically, by the names of the elements.
/// It uses the bubble sort algorithm.
///

/// The node in which its childNodes are to be sorted. ///
-----
public static void SortElements(XmlNode node)
{
bool changed = true;
while (changed)
{
changed = false;
for (int i = 1; i < node.ChildNodes.Count; i++) { if
(String.Compare(node.ChildNodes[i].Name,
node.ChildNodes[i-1].Name, true) < 0) { //Replace:
node.InsertBefore(node.ChildNodes[i], node.ChildNodes[i-1]);
changed = true; } } } } } } [/csharp]

```