Question
Thembelani Mlalazi · Jan 4, 2018

C# to Cache objects

I am trying to achieve this in cache objects I am using 2014.1 here is the original code in C# and would like to convert this to cache

here is my code first c# and cache follows

```
 class Program
{
    /// <summary>
    /// This function loads a XML document from the specified string.
    /// </summary>
    /// <param name="xml">Input XML string</param>
    /// <returns>XML to Json converted string</returns>
    public static string XmlToJSON(string xml)
    {
        XmlDocument xmlDoc = new XmlDocument();
        xmlDoc.LoadXml(xml);

        StringBuilder sbJSON = new StringBuilder();
        sbJSON.Append("{ ");
        XmlToJSONnode(sbJSON, xmlDoc.DocumentElement, true);
        sbJSON.Append("}");
        return sbJSON.ToString();
    }

    /// <summary>
    /// Output a XmlElement, possibly as part of a higher array
    /// </summary>
    /// <param name="sbJSON">Json string to be created</param>
    /// <param name="node">XML node name</param>
    /// <param name="showNodeName">ArrayList of string or XmlElement</param>
    private static void XmlToJSONnode(StringBuilder sbJSON, XmlElement node,
                                      bool showNodeName)
    {
        if (showNodeName)
        {
            sbJSON.Append("\"" + SafeJSON(node.Name) + "\": ");
        }
        sbJSON.Append("{");

        // Building a sorted list of key-value pairs where key is case-sensitive
        // nodeName value is an ArrayList of string or XmlElement so that we know
        // whether the nodeName is an array or not.
        SortedList<string, object> childNodeNames = new SortedList<string, object
>();

        // Add in all node attributes.
        if (node.Attributes != null)
        {
```

```
            foreach (XmlAttribute attr in node.Attributes)
                StoreChildNode(childNodeNames, attr.Name, attr.InnerText);
        }

        // Add in all nodes.
        foreach (XmlNode cnode in node.ChildNodes)
        {
            if (cnode is XmlText)
            {
                StoreChildNode(childNodeNames, "value", cnode.InnerText);
            }
            else if (cnode is XmlElement)
            {
                StoreChildNode(childNodeNames, cnode.Name, cnode);
            }
        }

        // Now output all stored info.
        foreach (string childname in childNodeNames.Keys)
        {
            List<object> alChild = (List<object>)childNodeNames[childname];
            if (alChild.Count == 1)
                OutputNode(childname, alChild[0], sbJSON, true);
            else
            {
                sbJSON.Append(" \"" + SafeJSON(childname) + "\": [ ");
                foreach (object Child in alChild)
                    OutputNode(childname, Child, sbJSON, false);
                sbJSON.Remove(sbJSON.Length - 2, 2);
                sbJSON.Append(" ], ");
            }
        }
        sbJSON.Remove(sbJSON.Length - 2, 2);
        sbJSON.Append(" }");
    }

    /// <summary>
    /// Store data associated with each nodeName so that we know whether
    /// the nodeName is an array or not.
    /// </summary>
    /// <param name="childNodeNames">Collection of child nodes</param>
    /// <param name="nodeName">XML node name</param>
    /// <param name="nodeValue">XML node value</param>
    private static void StoreChildNode(SortedList<string, object> childNodeNames,

        string nodeName, object nodeValue)
    {
        // Pre-process contraction of XmlElements.
        if (nodeValue is XmlElement)
        {
            // Convert <aa></aa> into "aa":null
            // <aa>xx</aa> into "aa":"xx".
            XmlNode cnode = (XmlNode)nodeValue;
            if (cnode.Attributes.Count == 0)
            {
                XmlNodeList children = cnode.ChildNodes;
                if (children.Count == 0)
                {
                    nodeValue = null;
```

```
                }
                else if (children.Count == 1 && (children[0] is XmlText))
                {
                    nodeValue = ((XmlText)(children[0])).InnerText;
                }
            }
        }
        // Add nodeValue to ArrayList associated with each nodeName.
        // If nodeName doesn't exist then add it.
        List<object> ValuesAL;

        if (childNodeNames.ContainsKey(nodeName))
        {
            ValuesAL = (List<object>)childNodeNames[nodeName];
        }
        else
        {
            ValuesAL = new List<object>();
            childNodeNames[nodeName] = ValuesAL;
        }
        ValuesAL.Add(nodeValue);
    }

    /// <summary>
    /// This functions outputs all the stored information inside a child node.
    /// </summary>
    /// <param name="childname">Chile node name</param>
    /// <param name="alChild">Child node</param>
    /// <param name="sbJSON">Json string</param>
    /// <param name="showNodeName">Node visibility</param>
    private static void OutputNode(string childname, object alChild,
        StringBuilder sbJSON, bool showNodeName)
    {
        if (alChild == null)
        {
            if (showNodeName)
            {
                sbJSON.Append("\"" + SafeJSON(childname) + "\": ");
            }
            sbJSON.Append("null");
        }
        else if (alChild is string)
        {
            if (showNodeName)
            {
                sbJSON.Append("\"" + SafeJSON(childname) + "\": ");
            }
            string sChild = (string)alChild;
            sChild = sChild.Trim();
            sbJSON.Append("\"" + SafeJSON(sChild) + "\"");
        }
        else
        {
            XmlToJSONnode(sbJSON, (XmlElement)alChild, showNodeName);
        }
        sbJSON.Append(", ");
    }

    /// <summary>
```

```
        /// This function makes a string safe for JSON.
        /// </summary>
        /// <param name="sIn">Input child node</param>
        /// <returns>String safe Json</returns>
        private static string SafeJSON(string sIn)
        {
            StringBuilder sbOut = new StringBuilder(sIn.Length);
            foreach (char ch in sIn)
            {
                if (Char.IsControl(ch) || ch == '\'')
                {
                    int ich = (int)ch;
                    sbOut.Append(@"\u" + ich.ToString("x4"));
                    continue;
                }
                else if (ch == '\"' || ch == '\\' || ch == '/')
                {
                    sbOut.Append('\\');
                }
                sbOut.Append(ch);
            }
            return sbOut.ToString();
        }

        /// <summary>
        /// This function converts Json string to XML
        /// </summary>
        /// <param name="json">Inout Json string</param>
        /// <returns>Converted XML string</returns>
        public static XmlDocument JsonToXml(string json)
        {
            XmlNode newNode = null;
            XmlNode appendToNode = null;
            string[] arrElementData;

            XmlDocument returnXmlDoc = new XmlDocument();
            returnXmlDoc.LoadXml("<menu />");
            XmlNode rootNode = returnXmlDoc.SelectSingleNode("menu");
            appendToNode = rootNode;

            string[] arrElements = json.Split('\r');
            foreach (string element in arrElements)
            {
                string processElement = element.Replace("\r", "").Replace("\n", "").R
eplace("\t", "").Trim();
                if ((processElement.IndexOf("}") > -1 || processElement.IndexOf("]")
> -1) &&
                    appendToNode != rootNode)
                {
                    appendToNode = appendToNode.ParentNode;
                }
                else if (processElement.IndexOf("[") > -1)
                {
                    processElement = processElement.Replace(":", "").Replace("[", ""
).Replace("\"", "").Trim();
                    newNode = returnXmlDoc.CreateElement(processElement);
                    appendToNode.AppendChild(newNode);
                    appendToNode = newNode;
                }
```

```
                else if (processElement.IndexOf("{") > -1 && processElement.IndexOf("
:") > -1)
                {
                    processElement = processElement.Replace(":", "").Replace("{", "")
.Replace("\"", "").Trim();
                    newNode = returnXmlDoc.CreateElement(processElement);
                    appendToNode.AppendChild(newNode);
                    appendToNode = newNode;
                }
                else
                {
                    if (processElement.IndexOf(":") > -1)
                    {
                        arrElementData = processElement.Replace(": \"", ":").Replace(
"\",", "").Replace("\"", "").Split(':');
                        newNode = returnXmlDoc.CreateElement(arrElementData[0]);
                        for (int i = 1; i < arrElementData.Length; i++)
                        { newNode.InnerText += arrElementData[i]; }
                        appendToNode.AppendChild(newNode);
                    }
                }
            }
            return returnXmlDoc;
        }

        static void Main(string[] args)
        {
            string xml = "<menu id=\"file\" value=\"File\"> " +
                  "<popup>" +
                    "<menuitem value=\"New\" onclick=\"CreateNewDoc()\" />" +
                    "<menuitem value=\"Open\" onclick=\"OpenDoc()\" />" +
                    "<menuitem value=\"Close\" onclick=\"CloseDoc()\" />" +
                  "</popup>" +
                "</menu>";
            Console.WriteLine("Input XML string:\n");
            Console.WriteLine(xml);
            Console.WriteLine("\nOutput JSON string:\n");
            Console.WriteLine(XmlToJSON(xml));
            Console.WriteLine("\n=================================================
============================\n");

            const string json = @"{
                                ""foo"":""bar"",
                                ""complexFoo"": {
                                    ""subFoo"":""subBar""
                                    }
                                }";
            Console.WriteLine("Input JSON string:\n");
            Console.WriteLine(json);
            Console.WriteLine("\nOutput XML string:\n");
            Console.WriteLine(JsonToXml(json).InnerXml);
            Console.WriteLine();
        }
    }



Class SharpJSONToXML Extends %RegisteredObject
{
```

```
///<summary>
/// This functions loads a xml stream from a specified file
/// <param name="myfile">input xml file
Method LoadXMLFile(myfile As %String = "C:\test2.xml") As %Stream
{
  // open a text file using a %Library.File stream
  Set file = ##class(%File).%New(myfile)
  Do file.Open("RU") // same flags as the OPEN command
nbsp; // and copy the file into Memo
  Set incomeStream =##class(%Stream.FileCharacter).%New()
  Do incomeStream.CopyFrom(file)
nbsp; Set file = ""     // close the file
  Do ..XmlToJSON(incomeStream)
 return incomeStream
}
<summary>
/// This function loads a XML stream.
/// </summary>
/// <param name="xml">Input XML stream</param>
/// <returns>XML to Json converted stream</returns>
Method XmlToJSON(xml As %Stream) As %Stream
{

            set xmlDoc =##class(%XML.TextReader).ParseStream(xml,.textreader)

            set sbJSON =##class(%Stream.FileCharacter).%New()
            do sbJSON.WriteLine($C(123))
            do ..XmlToJSONnode(sbJSON,textreader.Name,1)
            do sbJSON.writeLine($C(125))
            return sbJSON
}
<summary>
/// Output a XmlElement, possibly as part of a higher array
/// </summary>
/// <param name="sbJSON">Json string to be created</param>
/// <param name="node">XML node name</param>
/// <param name="showNodeName">ArrayList of string or XmlElement</param>
Method XmlToJSONnode(sbJSON As %AbstractStream, node As %XML.Document, showNodeName A
s %Boolean)
{

            if (showNodeName)
            {
               do sbJSON.WriteLine($C(92)_..SafeJSON(node.GetDocumentNode())_$C(92)_$
C(58))
            }
            do sbJSON.WriteLine($C(123))

            // Building a sorted list of key-value pairs where key is case-sensitive
            // nodeName value is an ArrayList of string or XmlElement so that we know
            // whether the nodeName is an array or not.
            Set childNodeNames =##class(%ListOfObjects).%New()

            // Add in all node attributes.
            For tI=1:1:node.Count()
         {
        Set tResult=node.GetAt(tI)

        if (tResult.HasAttributes)
```

```
               {
                    For tJ=1:1:tResult.AttributeCount
      {
       Do tResult.MoveToAttributeIndex(tJ)

       do ..StoreChildNode(childNodeNames, tResult.Name, tResult.Value)
      }
                }
                  // Add in all nodes.
                if tResult.HasChildNodes()
                {
                 if (tResult.Name){do ..StoreChildNode(childNodeNames,tResult.Name, t
I) }elseif (tResult.Value){ do ..StoreChildNode(childNodeNames, "value",tResult.Value
)}
             }
      }

            // Now output all stored info.
            for childname=1:1:childNodeNames
            {
               #dim alChild as %ListOfObjects
                if (alChild.Count = 1)
                {
                  do ..OutputNode(childname, alChild, sbJSON,1)
                }
                else
                {
                do sbJSON.WriteLine($C(92)_..SafeJSON(childname)_$C(92)_$C(58)_$C(91)
)

                    for tm=1:1: alChild
                    {
                        do ..OutputNode(childname, Child, sbJSON,0)
                   do sbJSON.Remove(sbJSON.Length - 2, 2)
                   do sbJSON.WriteLine($C(93)_$C(44))
                    }
                }
            }
           do sbJSON.Remove(sbJSON.Length - 2, 2)
           do sbJSON.WriteLine($C(125))
}
/// <summary>
/// Store data associated with each nodeName so that we know whether
/// the nodeName is an array or not.
/// </summary>
/// <param name="childNodeNames">Collection of child nodes</param>
/// <param name="nodeName">XML node name</param>
/// <param name="nodeValue">XML node value</param>
Method StoreChildNode(childNodeNames As %ListOfObjects(CLASSNAME="%XML.XPATH.Result")
, nodeName As %String, nodeValue As %XML.Node)
{

            // Pre-process contraction of XmlElements.
            if (nodeValue.NodeType="Element")
            {
                // Convert <aa></aa> into "aa":null
                // <aa>xx</aa> into "aa":"xx".
                #dim cnode As %XML.Node
                #dim children As %XML.Node
```

```
                if (cnode.Attributes.Count = 0)
                {
                    set children = cnode.HasChildNodes()
                    if (children.Count = 0)
                    {

                    }
                    elseif (children.Count = 1)
                    {

                    }
                }
            }
            // Add nodeValue to ArrayList associated with each nodeName.
            // If nodeName doesn't exist then add it.
            #dim ValuesAL as %ListOfObjects

            if (childNodeNames.ContainsKey(nodeName))
            {
               set ValuesAL = childNodeNames
            }
            else
            {

               set childNodeNames = ValuesAL
            }
            set ValuesAL=$LISTBUILD(nodeValue)
}

/// <summary>
/// This functions outputs all the stored information inside a child node.
/// </summary>
/// <param name="childname">Chile node name</param>
/// <param name="alChild">Child node</param>
/// <param name="sbJSON">Json string</param>
/// <param name="showNodeName">Node visibility</param>
Method OutputNode(childname As %String, alChild As %RegisteredObject, sbJSON As %Stre
am, showNodeName As %Boolean)
{

            if (alChild = null)
            {
                if (showNodeName)
                {
                   do sbJSON.WriteLine($C(92)_..SafeJSON(childname)_$C(92)_$C(58))
                }
               do sbJSON.WriteLine("null")
            }
            elseif (alChild)
            {
                if (showNodeName)
                {
                    do sbJSON.WriteLine($C(92)_..SafeJSON(childname)_$C(92)_$C(58))
                }
                set sChild = alChild
               set sChild = TRIM("")
                do sbJSON.writeLine($C(92)_..SafeJSON(sChild)_$C(92))
            }
```

```
            else
            {
               do ..XmlToJSONnode(sbJSON,alChild, showNodeName)
            }
             do sbJSON.WriteLine($C(4))
/// <summary>
/// This function makes a string safe for JSON.
/// </summary>
/// <param name="sIn">Input child node</param>
/// <returns>String safe Json</returns>
Method SafeJSON(sIn As %XML.Node) As %Stream
{

            Set sbOut = ##class(%Stream.FileCharacter).%New()
            for i=1:1:sIn
            {
                if ($C(60) || $C(92)_$C(62))
                {

                    continue
                }
                elseif ($C(92)_$C(34) || $C(92)_$C(92) || $C(47))
                {
                    do sbOut.write($C(92)_$C(92))
                }
               do sbOut.write()
            }
            return sbOut
}
}
```

#.NET #Key Value #Object Data Model #XML #Caché

Source URL:https://community.intersystems.com/post/c-cache-objects