



Reading/Writing XSD defined XML Files



Shaun C Curtis

2 Dec 2020 CPOL

How to quickly generate code to handle XML files with an XSD definition

This article shows the reader how to quickly read/write XML Files with an XSL Definition File into DotNetCore Objects.

Introduction

There are plenty of articles on the web showing you how to Serialize/Deserialize XML - some good, but most regurgitation of the same basic content. I'm not going to do that here. Almost nobody covers how to do it quickly in scenarios where you have an XSD definition file.

There's a [Github Repository here](#) to accompany this article.

To demonstrate the process we're going to import GPX files. These are XML formatted files with a *gpx* extension. [The detailed XSD definition is here.](#)

One look at the definition will tell you how complex and detailed a GPX file can be, and how much effort would be required to code and test the handling classes manually. Luckily Microsoft provide a little known tool to automate most of the process, buried away in the Windows SDK - *XSD.exe*.

Building the Classes

1. Set up a bare bones DotNetCore console project (if you use Visual Studio the IDE will do some clever XSD to class associations for you).
2. Get the XSD file and add it to a subfolder - *GPX.xsd* - in this case *Gpx*.
3. Find *XSD.exe*. It's in the Windows SDK - at the moment on my machine its in *C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools*.
4. Open a console/powershell window, and change to the *XSD.exe* directory.
5. Run *XSD.exe* against the xsd file. Note I've set the output to the input directory.

```
.\xsd.exe "D:\Documents\GitHub\GPXReader\Gpx\gpx.xsd"
/c /outputdir:"D:\Documents\GitHub\GPXReader\Gpx"
```

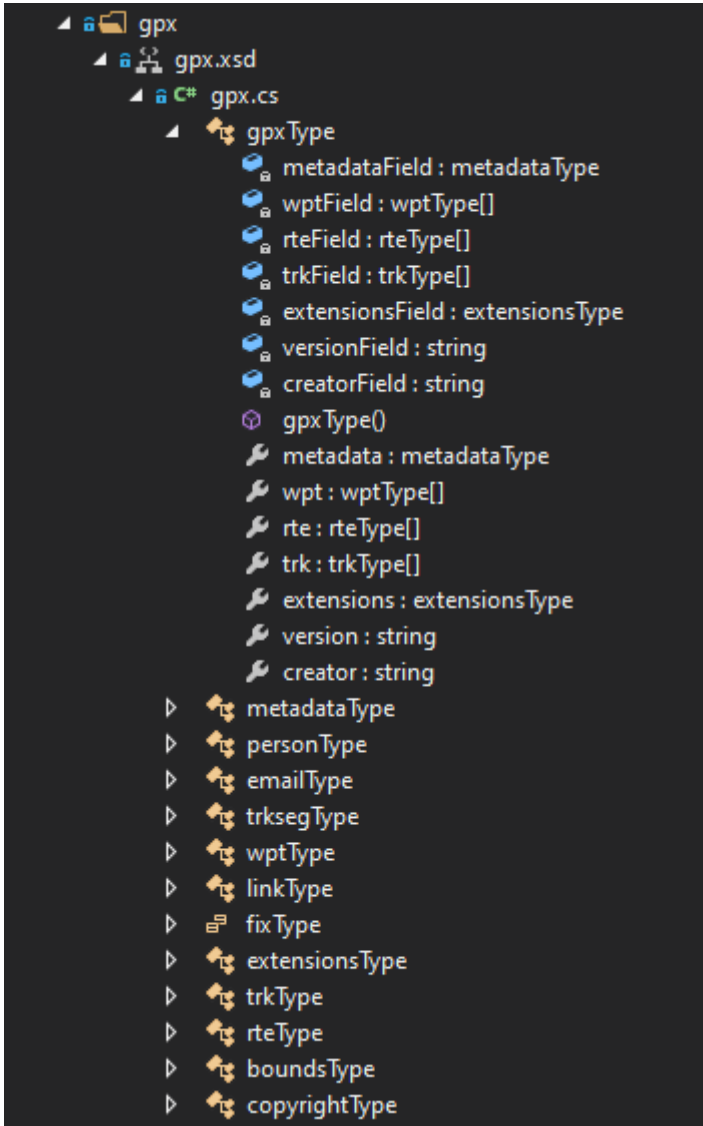
The result should be (I've added the separation ===== to make it clearer):

```
PS C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools>
.\xsd.exe "D:\Documents\GitHub\GPXReader\Gpx\gpx.xsd"
/c /outputdir:"D:\Documents\GitHub\GPXReader\Gpx"

=====
Microsoft (R) Xml Schemas/DataTypes support utility
[Microsoft (R) .NET Framework, Version 4.8.3928.0]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'D:\Documents\GitHub\GPXReader\Gpx\gpx.cs'.
```

```
PS C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools>
```

In the project, you should now see:



Don't be tempted into a renaming exercise (I don't like those class names either!). Think about your code maintenance process if the definition gets updated.

Import Code

We now need a simple Importer Class to show we're importing data correctly. The methods are all **static** and use streams because **XmlSerializer** uses streams, not strings.

ReadFile - there are two versions:

1. Creates a **StreamReader** to get the import file
2. Creates an **XmlSerializer** object with the correct object type and XSD definition
3. Runs **Deserialize**, casting the result to the correct object type
4. Object disposal is handled by **using**

WriteFile:

1. Creates a **StreamWriter** to accept the **XmlSerializer** output - points to the output file
2. Creates an **XmlSerializer** object with the correct object type and XSD definition
3. Runs **Serialize**, outputting to the **StreamWriter**
4. Flushes the **StreamWriter** to write data to the file

5. Object disposal is handled by **using**

```

using System;
using System.IO;
using System.Xml.Serialization;

namespace GPXReader
{
    public class GPXReader
    {
        public static gpxType ReadFile(Uri url)
        {
            gpxType gpxdata = null;
            try
            {
                using (StreamReader reader = new StreamReader(url.AbsolutePath))
                {
                    XmlSerializer serializer = new XmlSerializer(typeof(gpxType),
                        "http://www.topografix.com/GPX/1/1");
                    gpxdata = serializer.Deserialize(reader) as gpxType;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine($"An Error has occurred accessing file
                    {url.AbsolutePath}.{Environment.NewLine} Details:{Environment.NewLine}
                    {e.StackTrace}.");
            }
            return gpxdata;
        }

        // Bool return version
        public static bool ReadFile(Uri url, out gpxType gpxdata)
        {
            gpxdata = null;
            try
            {
                using (StreamReader reader = new StreamReader(url.AbsolutePath))
                {
                    XmlSerializer serializer = new XmlSerializer(typeof(gpxType),
                        "http://www.topografix.com/GPX/1/1");
                    gpxdata = serializer.Deserialize(reader) as gpxType;
                    return true;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine($"An Error has occurred accessing file
                    {url.AbsolutePath}.{Environment.NewLine} Details:{Environment.NewLine}
                    {e.StackTrace}.");
            }
            return false;
        }

        public static bool WriteFile(gpxType data, Uri url)
        {
            try
            {
                using (StreamWriter writer = new StreamWriter(url.AbsolutePath, false))
                {
                    XmlSerializer serializer = new XmlSerializer(typeof(gpxType),
                        "http://www.topografix.com/GPX/1/1");
                    serializer.Serialize(writer, data);
                    writer.Flush();
                    return true;
                }
            }
            catch (Exception e)
            {

```

```

        Console.WriteLine($"An Error has occurred accessing file
                           {url.AbsolutePath}.{Environment.NewLine}
                           Details:{Environment.NewLine} {e.StackTrace}.");
    }
    return false;
}
}
}

```

Finally, we build a simple **Program**. I've added a fairly complex gpx file (imported from Google Maps) to the project.

```

using System;

namespace GPXReader
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("GPX Reader");

            var data = GPXReader.ReadFile
                (new Uri($"D:/Documents/GitHub/GPXReader/gpx/Test.gpx"));

            Console.WriteLine($"Read file");

            // Set Break point here to view the imported file

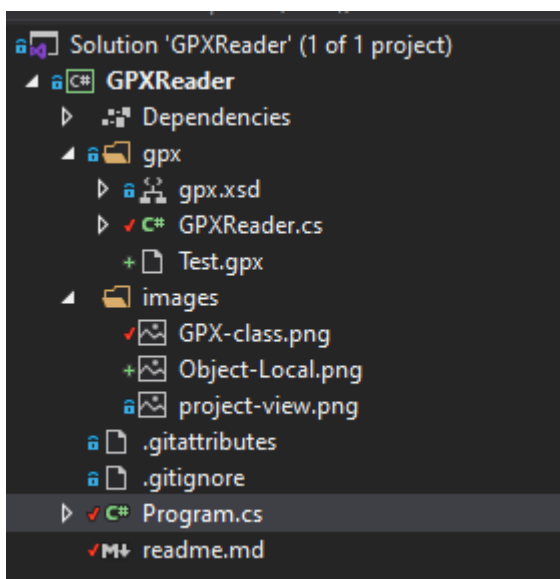
            Console.WriteLine($"Writing output file");

            GPXReader.WriteFile
                (data, new Uri($"D:/Documents/GitHub/GPXReader/gpx/output.gpx"));

            Console.WriteLine("Complete");
        }
    }
}

```

The project looks like this:



Testing Output

Run the project with a breakpoint set and explore the created **data** object.

| Name | Value | Type |
|-----------------|--|-----------------------|
| args | {string[0]} | string[] |
| url | {file:///D:/Documents/GitHub/GPXReader/gpx/Test.gpx} | System.Uri |
| data | {GPXReader.gpxType} | GPXReader.gpxType |
| creator | "KML2GPX.COM" | string |
| creatorField | "KML2GPX.COM" | string |
| extensions | null | GPXReader.extensi... |
| extensionsField | null | GPXReader.extensi... |
| metadata | {GPXReader.metadataType} | GPXReader.metada... |
| metadataField | {GPXReader.metadataType} | GPXReader.metada... |
| rte | null | GPXReader.rteType[] |
| rteField | null | GPXReader.rteType[] |
| trk | {GPXReader.trkType[1]} | GPXReader.trkType[] |
| [0] | {GPXReader.trkType} | GPXReader.trkType |
| cmt | null | string |
| cmtField | null | string |
| desc | "This is track no: 1" | string |
| descField | "This is track no: 1" | string |
| extensions | null | GPXReader.extensi... |
| extensionsField | null | GPXReader.extensi... |
| link | null | GPXReader.linkType[] |
| linkField | null | GPXReader.linkType[] |
| name | "Santander to Bilbao Journey" | string |
| nameField | "Santander to Bilbao Journey" | string |
| number | "1" | string |
| numberField | "1" | string |
| src | null | string |
| srcField | null | string |
| trkseg | {GPXReader.trksegType[1]} | GPXReader.trksegTy... |
| trksegField | {GPXReader.trksegType[1]} | GPXReader.trksegTy... |
| @type | null | string |
| typeField | null | string |
| trkField | {GPXReader.trkType[1]} | GPXReader.trkType[] |
| version | "1.1" | string |
| versionField | "1.1" | string |
| wpt | {GPXReader.wptType[36]} | GPXReader.wptType[] |
| wptField | {GPXReader.wptType[36]} | GPXReader.wptType[] |

That's it!

History

- 1st December, 2020: Initial version

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author




Shaun C Curtis

Retired Cold Elm

United Kingdom 

Ex Geologist, Project Manager, Web Host, Business Owner and IT Consultant. Now, a traveller to places less travelled. And part time developer trying to keep up!

Comments and Discussions

 **0 messages** have been posted for this article Visit <https://www.codeproject.com/Tips/5287636/Reading-Writing-XSD-defined-XML-Files> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#)
[Advertise](#)
[Privacy](#)
[Cookies](#)
[Terms of Use](#)

Article Copyright 2020 by Shaun C Curtis
Everything else Copyright © [CodeProject](#),
1999-2020

Web03 2.8.20201130.1