XML ”Mandag”

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gunstore>
    <class>
      <className>Assault rifle</className>
      <gun type="AR1">
          <name>AK-47</name>
          <year>2005</year>
          <price>749.00</price>
      </gun>
      <gun type="AR2">
          <name>XM4</name>
          <year>2009</year>
          <price>800</price>
      </gun>
  </class>
  <class>
    <className>Small machine gun</className>
    <gun type="SMG1">
        <name>MP5</name>
         <year>2007</year>
        <price>650.00</price>
    </gun>
    <gun type="SMG2">
        <name>MAC10</name>
         <year>1997</year>
        <price>349.00</price>
    </gun>
  </class>
</gunstore>
```

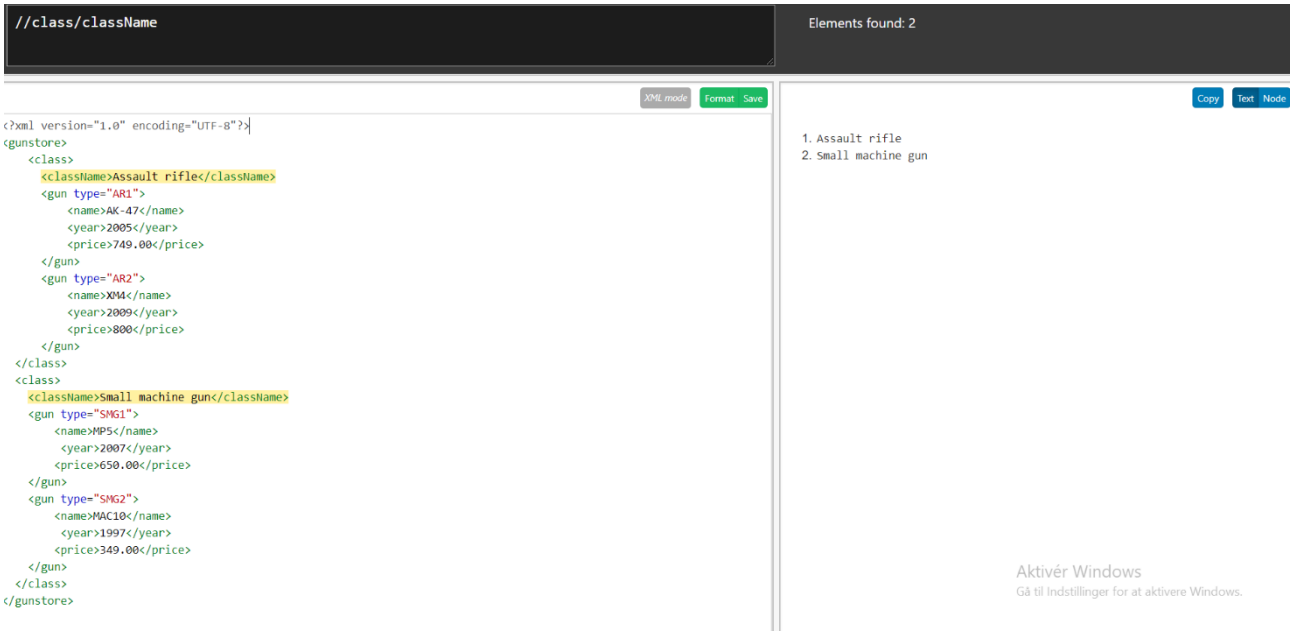
Det her er vores XML som holder på alt den data vi har i vores såkaldte ”gunstore”. Hvor vi har delt de forskellige guns op med nogle attributer med typen af våben. Sådan det er nemmere at finde dem igen med fx xPath. Og dette var det vi havde mest om mandag.

XSD ”Tirsdag”

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:
xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="gunstore">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="class" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="gun" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element type="xs:positiveInteger" name="year"/>
                    <xs:element type="xs:float" name="price"/>
                  </xs:sequence>
                  <xs:attribute type="xs:string" name="type"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Her viser vi vores XSD fil som sætter vores standard for det vi tager ind fra andre virksomheder sådan, vi ikke modtager et våben der kun indeholder halvdelen af dataen, som vi skal bruge i vores gunstore. Og på denne måde er vi også sikrer på, at prisen ikke bliver sat til fx ”12asd23” da den kun kan sættes til værdien ”float”. Dette var det vi fik lavet i starten af tirsdag.

xPath ”Tirsdag”



Her har vi et værktøj kaldet ”xPath” som bliver brugt til at finde bestemte data fra vores XML fil af fx her ser vi, at vi finder alle ”className’s” som er childs af elementet ”class”. Dette blev lavet samtidig med vi lavede vores XSD tirsdag formiddag.

XSLT ”Tirsdag”

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h2>My Gun Store</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Name</th>
      <th style="text-align:left">Year</th>
      <th style="text-align:left">Price</th>
    </tr>
    <xsl:for-each select="gunstore/class/gun">
    <xsl:sort select="year" data-type="number"/>
    <xsl:if test="price &gt; 200">
    <tr>
      <td><xsl:value-of select="name"/></td>
      <td><xsl:value-of select="year"/></td>
      <td><xsl:value-of select="price"/></td>
    </tr>
    </xsl:if>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Her vises vores XSLT fil som i dette tilfælde kan vise alt vores data som bliver kørt igennem et foreach loop ”xsl:for-each” hvor vi selecter, den komplekse type gunstore og finder childet af den som er class og derefter finder vi childet som er vores ”gun’s”. Efterfølgende bliver de sorteret med ”xsl:sort” hvor vi selecter ”year” og de skal sorteres med data typen ”number” så de bliver sorteret i numerisk orden og ikke i alfabetisk orden. Dette var det vi lavede i slutningen af tirsdagen.

Konvertering af XSD til C# ”Onsdag”

```
namespace XSD
{
    using System.Collections;
    using System.Collections.Generic;
    using System.Collections.ObjectModel;
    using System.Xml.Serialization;
```

```
    [System.CodeDom.Compiler.GeneratedCodeAttribute("XmlSchemaClassGenerator", "2.0
.479.0")]
    [System.SerializableAttribute()]
    [System.Xml.Serialization.XmlTypeAttribute("gunstore", Namespace="", AnonymousT
ype=true)]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("gunstore", Namespace="")]
    public partial class Gunstore
    {

        [System.Xml.Serialization.XmlIgnoreAttribute()]
        private System.Collections.ObjectModel.Collection<GunstoreClassGun> _class;

        [System.Xml.Serialization.XmlArrayAttribute("class")]
        [System.Xml.Serialization.XmlArrayItemAttribute("gun")]
        public System.Collections.ObjectModel.Collection<GunstoreClassGun> Class
        {
            get
            {
                return this._class;
            }
            private set
            {
                this._class = value;
            }
        }

        /// <summary>
        /// <para xml:lang="de">Ruft einen Wert ab, der angibt, ob die Class-
Collection leer ist.</para>
        /// <para xml:lang="en">Gets a value indicating whether the Class collectio
n is empty.</para>
        /// </summary>
        [System.Xml.Serialization.XmlIgnoreAttribute()]
        public bool ClassSpecified
        {
            get
            {
                return (this.Class.Count != 0);
            }
        }

        /// <summary>
        /// <para xml:lang="de">Initialisiert eine neue Instanz der <see cref="Guns
tore" /> Klasse.</para>
```

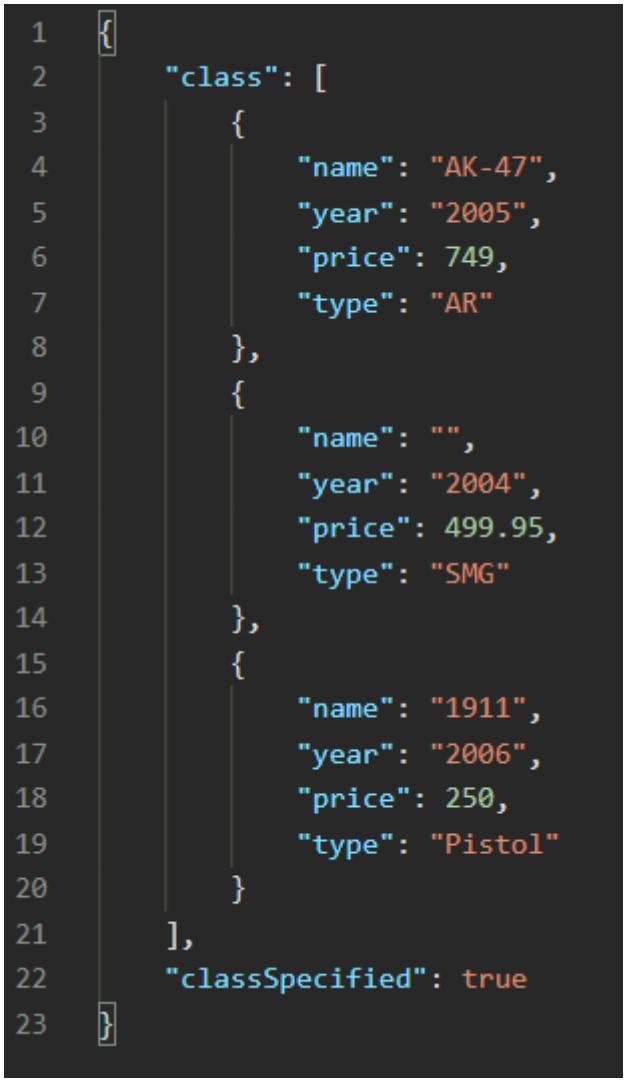
```
        /// <para xml:lang="en">Initializes a new instance of the <see cref="Gunsto
re" /> class.</para>
        /// </summary>
        public Gunstore()
        {
            this._class = new System.Collections.ObjectModel.Collection<GunstoreCla
ssGun>();
        }
    }

    [System.CodeDom.Compiler.GeneratedCodeAttribute("XmlSchemaClassGenerator", "2.0
.479.0")]
    [System.SerializableAttribute()]
    [System.Xml.Serialization.XmlTypeAttribute("GunstoreClass", Namespace="", Anony
mousType=true)]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    public partial class GunstoreClass
    {

        [System.Xml.Serialization.XmlIgnoreAttribute()]
        private System.Collections.ObjectModel.Collection<GunstoreClassGun> _gun;

        [System.Xml.Serialization.XmlElementAttribute("gun")]
        public System.Collections.ObjectModel.Collection<GunstoreClassGun> Gun
        {
            get
            {
                return this._gun;
            }
            private set
            {
                this._gun = value;
            }
        }

        /// <summary>
        /// <para xml:lang="de">Ruft einen Wert ab, der angibt, ob die Gun-
Collection leer ist.</para>
        /// <para xml:lang="en">Gets a value indicating whether the Gun collection
is empty.</para>
        /// </summary>
        [System.Xml.Serialization.XmlIgnoreAttribute()]
        public bool GunSpecified
        {
            get
            {
```

```
                return (this.Gun.Count != 0);
            }
        }

        /// <summary>
        /// <para xml:lang="de">Initialisiert eine neue Instanz der <see cref="Guns
toreClass" /> Klasse.</para>
        /// <para xml:lang="en">Initializes a new instance of the <see cref="Gunsto
reClass" /> class.</para>
        /// </summary>
        public GunstoreClass()
        {
            this._gun = new System.Collections.ObjectModel.Collection<GunstoreClass
Gun>();
        }
    }

    [System.CodeDom.Compiler.GeneratedCodeAttribute("XmlSchemaClassGenerator", "2.0
.479.0")]
    [System.SerializableAttribute()]
    [System.Xml.Serialization.XmlTypeAttribute("GunstoreClassGun", Namespace="", An
onymousType=true)]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    public partial class GunstoreClassGun
    {

        [System.ComponentModel.DataAnnotations.RequiredAttribute()]
        [System.Xml.Serialization.XmlElementAttribute("name")]
        public string Name { get; set; }

        [System.ComponentModel.DataAnnotations.RequiredAttribute()]
        [System.Xml.Serialization.XmlElementAttribute("year")]
        public string Year { get; set; }

        [System.ComponentModel.DataAnnotations.RequiredAttribute()]
        [System.Xml.Serialization.XmlElementAttribute("price")]
        public float Price { get; set; }

        [System.Xml.Serialization.XmlAttributeAttribute("type")]
        public string Type { get; set; }
    }
}
```

Her har vi vores lange C# fil som bare er vores XSD fil konverteret til C# ved brug af et værktøj der hedder ”xscgen” som opretter en class model af vores XSD bare i en C# fil i stedet sådan vi efterfølgende kunne lave en ASP.Net med en api der kunne vise vores data i en browser.

```
using (var reader = XmlReader.Create("my.xml"))
            {
                var serializer = new XmlSerializer(typeof(XSD.Gunstore));

                instans = (XSD.Gunstore)serializer.Deserialize(reader);
            }
```

Her gør vi brug af en XML serializer til vores type af Gunstore elementet og vi efterfølgende laver en instans, hvor vi deserializere vores XML fil så den kan blive vist i ”Postman”.



```
[HttpGet]
    public XSD.Gunstore Get()
    {
        var test = new DataStore();
        return test.instans;
    }
```

Vi bruger denne Get metode til at hente alt dataen fra instansen af DataStore'n som indeholder alle våben fra Gunstoren i XML filen

Razor Pages "Torsdag"

```
@page
@model webApplicationAsp.Pages.GameModel
@{
    <h1>We are in the game page.</h1>
    <table>
        <tr>
            <th style="color:blue;">Name |</th>
            <th style="color:green;">Damage |</th>
            <th style="color:orange;">Price |</th>
            <th style="color: mediumvioletred;">Age</th>
        </tr>

    </table>
    foreach (var data in Model.gameLibrary.Data.Weapon)
```

Her har vi vores kode til vores razor pages hvor den viser vores data fra XML filen inde på en hjemmeside ved hjælp af denne HTMLC# fil.

```
namespace webApplicationAsp.Pages
{
    5 references
    public class GameModel : PageModel
    {
        2 references
        public Game gameLibrary { get;  set; }
        0 references
        public void OnGet()
        {
            gameLibrary = new Game("D:\\Valfag\\XSDTestFile\\GameInXML.xml");
```

Her er vi inde under vores C# fil hvor vi opretter en instans af vores gameLibrary hvor vi kommer til at hente alt dataen ud fra når vi kører IIS'en.