

C# Project

CRUD acties met C#/XML/JSON/YAML



<?xml?>

{JSON}
JavaScript Object Notation

YAML

Koos Starreveld

Raymond van Dorresteyn

Pieter Brouwer

Woensdag 16-maart-2022

Inhoudsopgave

Project	2
Opdracht.....	2
Criteria	2
Beoordeling	2
Tips.....	2
Project aanmaken in Visual Studio.....	3
XML bestand aanmaken	5
Muzieknummers weergeven in een pagina	6
Nieuwe pagina toevoegen.....	6
De class SongMethods voorzien van code	7
Het XML bestand inlezen.....	8
Inhoud XML bestand zichtbaar maken in HTML.....	10
Een nieuw muzieknummer toevoegen aan de playlist	13
Aanpassingen op de bestaande webapplicatie	13
Nieuwe pagina toevoegen.....	14
Request en Response	15
Een muzieknummer verwijderen uit de playlist.....	16
Nieuwe pagina toevoegen.....	16
\Pages\XML\Index.cshtml pagina gereed maken voor gebruik	17
De class SongMethods aanpassen.....	18
\Pages\XML\Delete.cshtml pagina gereed maken voor gebruik	18
Een muzieknummer wijzigen in de playlist	19
Een eerste blik op JSON en YAML.....	19

Project

Opdracht

Voor dit project gaan we gegevens uit XML en JSON (en eventueel YAML) bestanden lezen en bewerkingen op uitvoeren. We gaan een website maken met meerdere afspeellijsten met muzieknnummers. Je kan ervoor kiezen om de muzieknnummers te uploaden en te kunnen afspelen.

Dit project dient in tweetallen (of waar nodig drietallen) te worden uitgevoerd. Jullie kiezen zelf met wie je wil samenwerken en geeft vervolgens de groep aan bij de docent.

Er zal een methodiek genaamd Scrum worden gebruikt tijdens dit project. In deze vorm zullen de volgende onderdelen aan bod komen:

- Product Backlog Item (PBI)
 - Per groep bepaal je welke onderdelen er dienen te worden uitgevoerd om het project succesvol te maken. Dit kan zo gedetailleerd als jullie zelf willen, zolang er minimaal maar 10 items zijn. Denk hier bijvoorbeeld aan: web project opstarten, song class maken, CSS stijl zoeken, CSS stijl toevoegen, pagina om de muzieknnummers weer te geven, enzovoort. Deze items zet je in een e-mail, met als titel **Product Backlog Items**, en stuurt deze naar je docent.
- Sprint Backlog Item
 - Aan het begin van de sprint overleg je met elkaar welke Product Backlog Items jullie gaan uitvoeren in de aankomende sprint en welke jullie dus aan het einde van deze sprint denken af te hebben. Deze items zet je in een e-mail, met als titel **Sprint x Backlog Items** (waarbij de x het nummer is van de sprint), en stuurt deze naar je docent.
- Sprint
 - We zullen tijdens dit project sprints hanteren van twee weken. In totaal zullen we 4 sprints uitvoeren voordat het project klaar dient te zijn.
- Sprint Review
 - Aan het einde van de sprint kijk je terug en geef je antwoord op de volgende vragen:
 - Wat is er goed gegaan?
 - Wat kon er beter?
 - Wat hebben we nodig om verder te kunnen?

De antwoorden op deze vragen zet je in een e-mail, met als titel **Sprint x Review** (waarbij de x het nummer is van de sprint), en stuurt deze naar je docent.

Criteria

De volgende criteria gelden voor dit project:

- Je website dient opgezet te zijn zoals beschreven in het stappenplan hieronder, waar dus gebruik wordt gemaakt van een webapplicatie in C#;
- Je dient een mooie webapplicatie op te leveren die naar behoren werkt. Dit betekent dat je een eigen stijl moet hebben voor je webapplicatie;
- Er moet zowel een mogelijkheid voor XML als voor JSON (en voor drietallen ook YAML) inzitten;
- Er dient een eigen toevoeging / feature op te zijn genomen die jullie hebben bedacht / gekozen;

Beoordeling

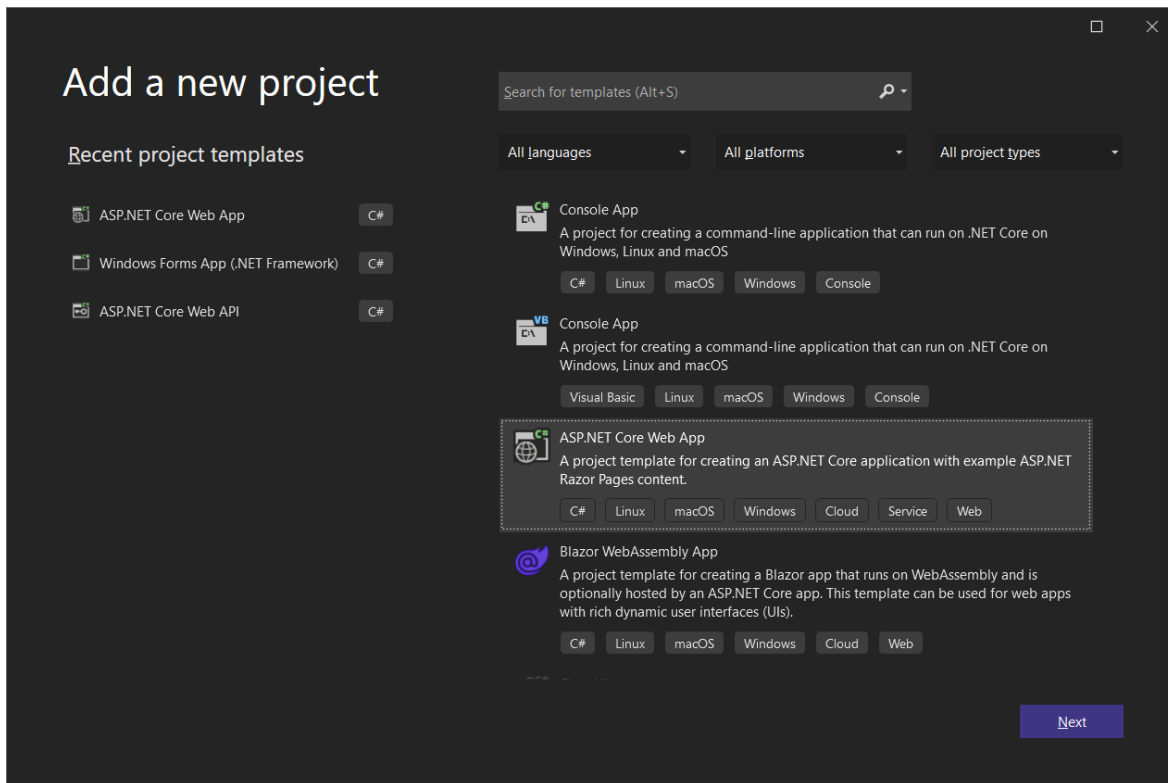
Je wordt uiteindelijk beoordeeld op de webapplicatie, de individuele bijdrage, jullie samenwerking en het toepassen van Scrum en de bij te houden documentatie.

Tips

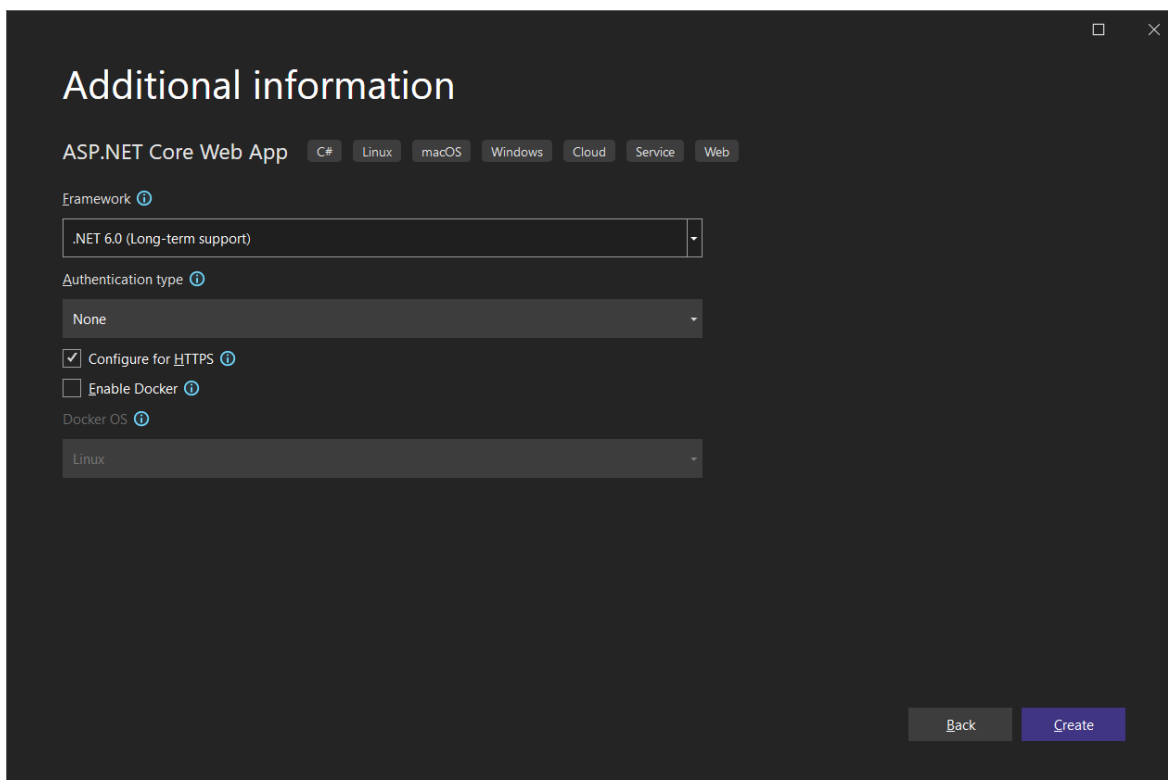
- Gebruik een website zoals Azure DevOps of Trello voor het bijhouden van Backlog Items;
- Gebruik voor het gezamenlijk ontwikkelen aan een project zoiets als GitHub of Subversion;

Project aanmaken in Visual Studio

- Open Visual Studio 2022
- Kies in het hoofdmenu de optie **File > New > Project ...**
- Kies in het dialoogvenster Add a new project de optie **ASP.NET Core Web App**.



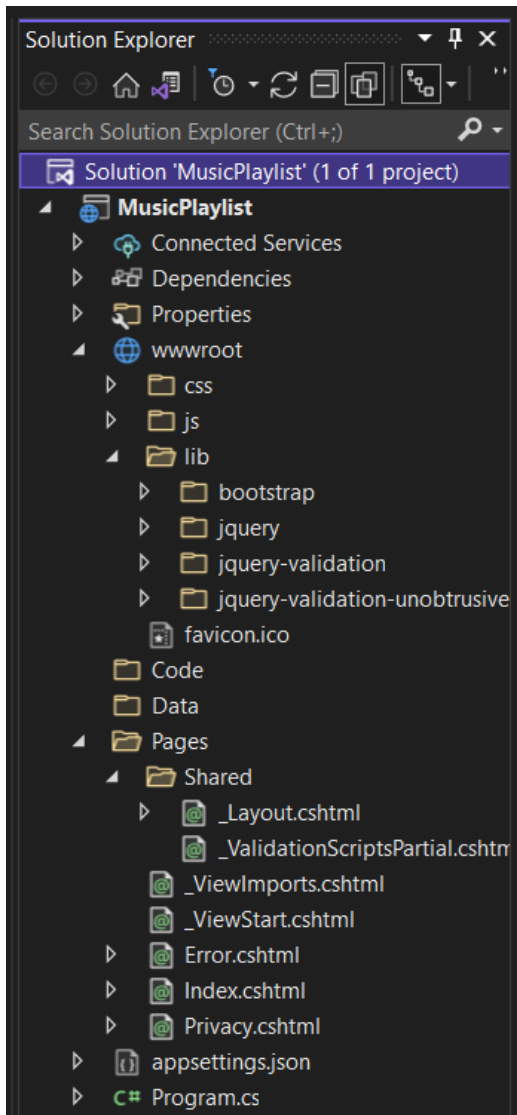
- Sla dit project op onder de naam **MusicPlaylist** in een map naar keuze. Klik op de knop **Next**.
- In het scherm Additional information hoeft er niets veranderd te worden. Klik op de knop **Create**.



- Een basis project is nu aangemaakt. Open en bekijk de Solution Explorer. In dit project zijn verschillende functionaliteiten standaard al opgenomen, denk o.a. aan (Twitter) Bootstrap, jQuery en jQuery Validation.

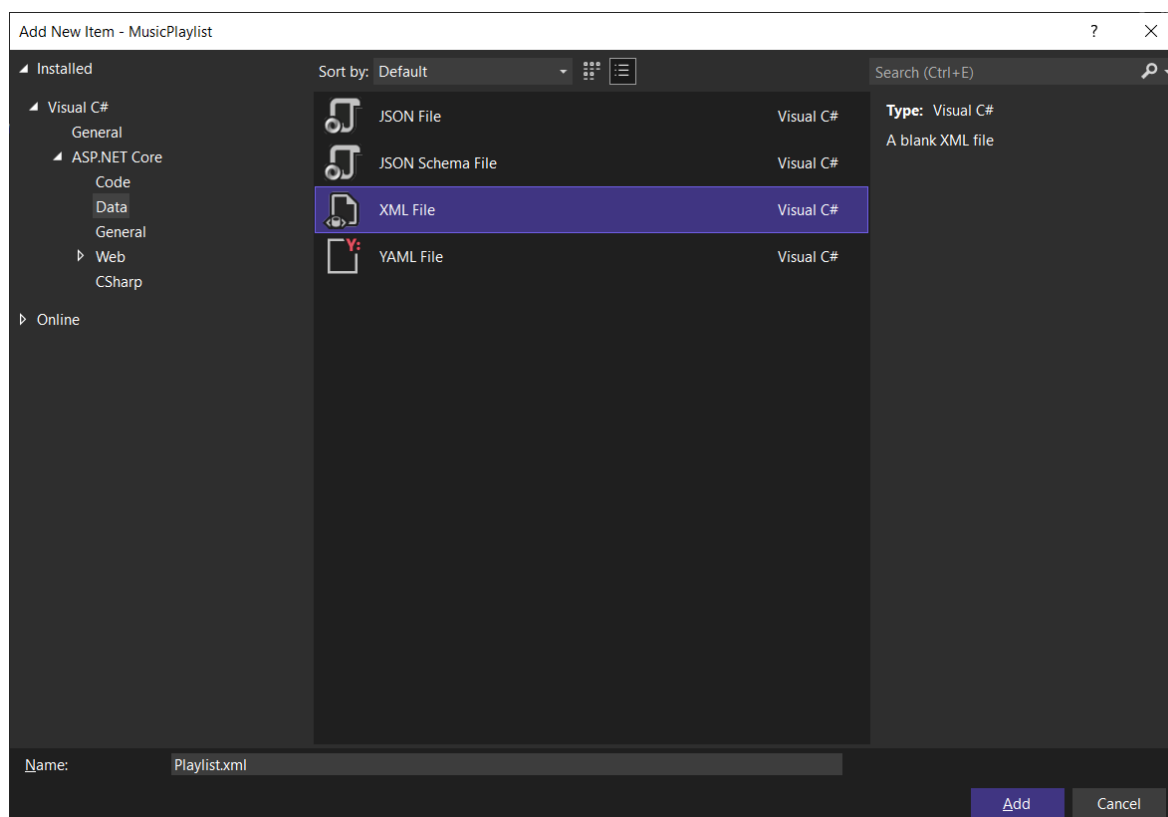
(Twitter) Bootstrap is een collectie van tools voor het creëren van websites en (web)applicaties. Bootstrap bevat html- en css-based design templates voor forms, buttons, navigation en andere interfacecomponenten.

- Klik rechts op het project en kies **Add > New Folder**. Noem de nieuwe folder **Code**.
- Klik rechts op het project en kies **Add > New Folder**. Noem de nieuwe folder **Data**.



XML bestand aanmaken

Voeg een nieuw XML bestand toe door rechts te klikken op de **Data** folder en te kiezen voor **Add → New Item**. Vervolgens kies je voor **Visual C# → ASP.NET Core → Data** en dan **XML File**. Vergeet je XML bestand niet een goede naam te geven, bijvoorbeeld **Playlist.xml**. Klik vervolgens op **Add**.



In het XML bestand gaan we een structuur aanmaken waar we voor onze playlist onze muzieknnummers in op kunnen nemen. Een aantal zaken zijn noodzakelijk voor een XML bestand, namelijk:

- De eerste regel geeft de XML tag. In deze tag wordt XML informatie opgenomen;
- De tweede regel geeft de root tag aan. Deze is benodigd om onze muzieknnummers in op te slaan;

Typ de onderstaande code over in het aangemaakte XML bestand.

```
Playlist.xml  x MusicPlaylist: Overview
1  <?xml version="1.0" encoding="utf-8" ?>
2  <playlist>
3    <song>
4      <id>1</id>
5      <artist>Twenty One Pilots</artist>
6      <title>Stressed Out</title>
7      <year>2016</year>
8      <genre>Rap-rock</genre>
9      <time>3:22</time>
10   </song>
11   <song>
12     <id>2</id>
13     <artist>Twenty One Pilots</artist>
14     <title>Heathens</title>
15     <year>2016</year>
16     <genre>Rap-rock</genre>
17     <time>3:15</time>
18   </song>
19 </playlist>
```

Muzieknummers weergeven in een pagina

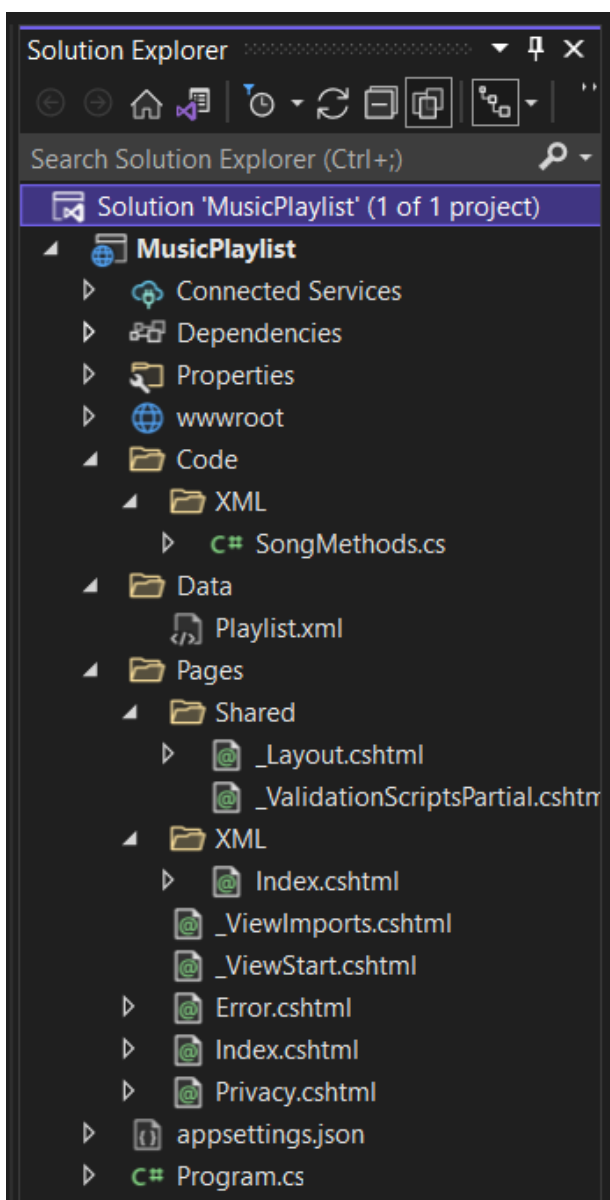
Nieuwe pagina toevoegen

We gaan nu een webpagina aanmaken waarin we de muzieknummers zichtbaar maken. Hiervoor moeten we de inhoud van het XML bestand inlezen en in HTML zichtbaar maken. Je gaat hier gebruik maken van een Razor Page en een Class.

- Klik rechts op de **Code** folder en kies **Add > New Folder**. Noem de nieuwe folder **XML**.
- Voeg een nieuwe Class toe door rechts te klikken op de nieuw gemaakte **XML** folder en te kiezen voor **Add → Class....** Geef de class de naam **SongMethods**.
- Klik rechts op de **Pages** folder en kies **Add > New Folder**. Noem de nieuwe folder **XML**.
- Voeg een nieuwe Razor Page (cshtml) toe door rechts te klikken op de nieuw gemaakte **XML** folder en te kiezen voor **Add → Razor Page....** Kies voor **Razor Page – Empty** en klik op **Add**. Geef de nieuwe Razor Page de naam **Index.cshtml**. Klik vervolgens op **Add**.

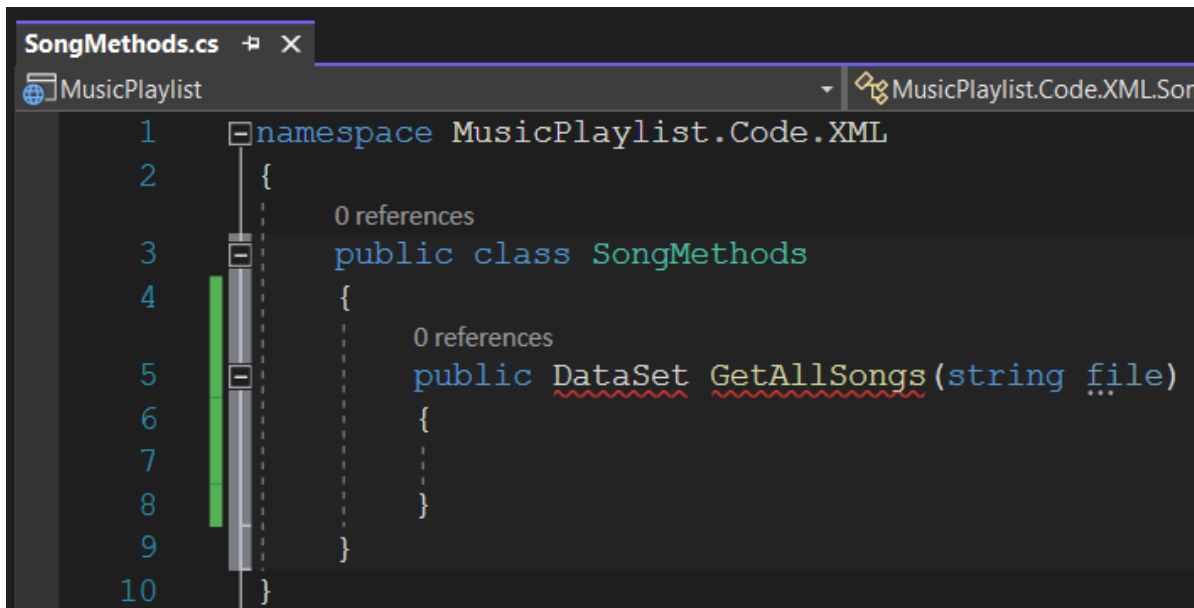
Bestanden eindigend op .cshtml staan voor de combinatie C-Sharp (C#) en HTML. Je kunt deze bestanden vergelijken met de extensie .phtml bij PHP. In cshtml bestanden kun je eenvoudig C# combineren met HTML, waarbij C# altijd begint met een @ symbol of als het een geheel code blok betreft met @{ }.

De structuur in de Solution Explorer ziet er nu als volgt uit:



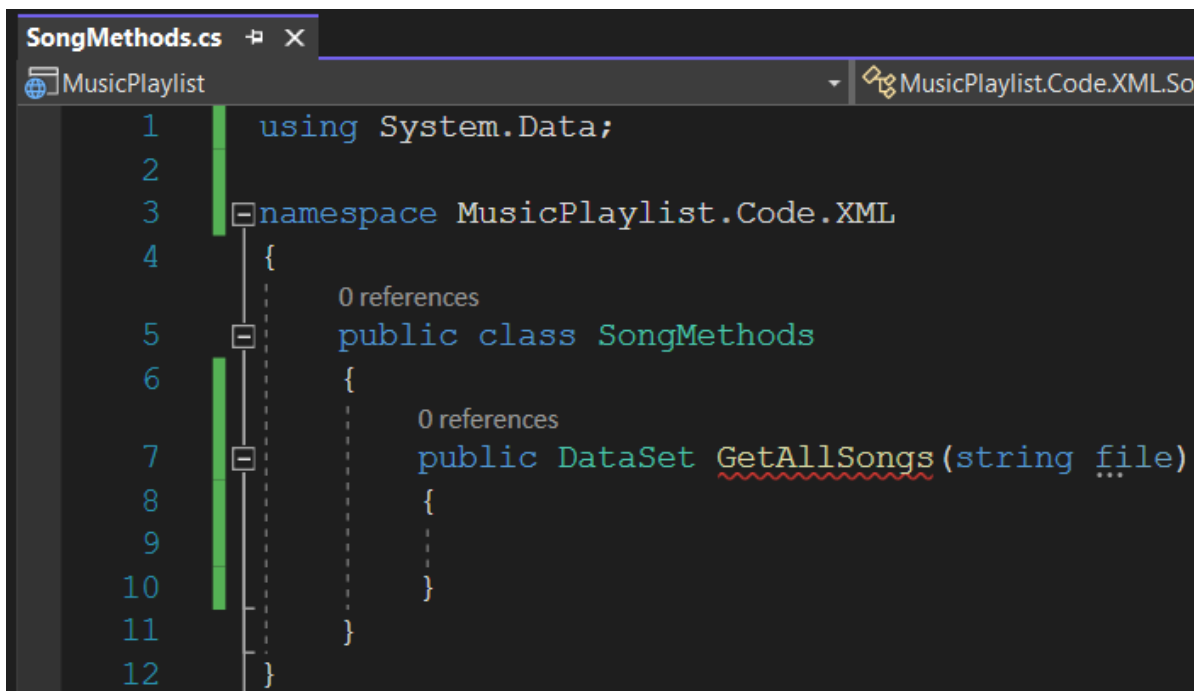
De class SongMethods voorzien van code

- Maak in de class SongMethods een methode GetAllSongs(string file) aan met de volgende code.



```
1 namespace MusicPlaylist.Code.XML
2 {
3     public class SongMethods
4     {
5         public DataSet GetAllSongs(string file)
6         {
7             ...
8         }
9     }
10 }
```

Je ziet dat Visual Studio twee fouten geeft door code met rood te onderstrepen. Eén van die fouten wordt veroorzaakt doordat het resultaat van de methode een DataSet is. Het DataSet type wordt nog niet herkend. De code voor het uitlezen van een XML bestand zit binnen een bepaalde library in C#. Om deze code te kunnen gebruiken moeten we ons bestand laten weten dat we die library willen gaan gebruiken. In dit geval dienen we de library System.Data in te laden, daarom voegen we bovenaan de pagina de volgende code toe:



```
1 using System.Data;
2
3 namespace MusicPlaylist.Code.XML
4 {
5     public class SongMethods
6     {
7         public DataSet GetAllSongs(string file)
8         {
9             ...
10        }
11    }
12 }
```

De class DataSet zit dus in de library System.Data. Eén van de foutmeldingen verdwijnt nu, de andere foutmelding zullen we hierna gaan oplossen.

Het XML bestand inlezen

De eerste class waar we een instantie van nodig hebben is een DataSet. Dit is te vergelijken met een database. Binnen de DataSet maken we een DataTable (een tabel) en daarbinnen zitten DataColumnns (kolommen). Na het inlezen van het XML bestand zitten er ook DataRowns (regels) in deze DataTable. In iedere constructor van deze objecten geven we de naam mee van de overeenkomende XML tag zodat we straks bij het inlezen daar profijt van hebben.

- Voeg nu onderstaande code toe aan de methode **GetAllSongs(string file)**. Met deze code wordt een instantie van een DataSet, een DataTable en meerdere DataColumnns gemaakt. Let hierbij op de parameter van de constructor van deze classes en vergelijk deze met de tags in het XML bestand.

```
0 references
7 public DataSet GetAllSongs(string file)
8 {
9     DataSet ds = new DataSet("playlist");
10
11     DataTable dtSongs = new DataTable("song");
12
13     DataColumn dcId = new DataColumn("id");
14     DataColumn dcArtist = new DataColumn("artist");
15     DataColumn dcTitle = new DataColumn("title");
16     DataColumn dcYear = new DataColumn("year");
17     DataColumn dcGenre = new DataColumn("genre");
18     DataColumn dcTime = new DataColumn("time");
19 }
```

Na het maken van alle objecten (instanties van de classes DataSet, DataTable, DataColumnns) moeten al deze objecten goed aan elkaar gekoppeld worden. De DataColumnns moeten allemaal aan de collectie DataTable.Columns worden toegevoegd. De DataTable zelf moet aan de collectie DataSet.Tables worden toegevoegd. Al deze koppelacties gaan met de standaard Add methode. Zorg dat de nieuwe code wordt toegevoegd wordt aan de methode GetAllSongs(string file).

```
0 references
7 public DataSet GetAllSongs(string file)
8 {
9     DataSet ds = new DataSet("playlist");
10
11     DataTable dtSongs = new DataTable("song");
12
13     DataColumn dcId = new DataColumn("id");
14     DataColumn dcArtist = new DataColumn("artist");
15     DataColumn dcTitle = new DataColumn("title");
16     DataColumn dcYear = new DataColumn("year");
17     DataColumn dcGenre = new DataColumn("genre");
18     DataColumn dcTime = new DataColumn("time");
19
20     dtSongs.Columns.Add(dcId);
21     dtSongs.Columns.Add(dcArtist);
22     dtSongs.Columns.Add(dcTitle);
23     dtSongs.Columns.Add(dcYear);
24     dtSongs.Columns.Add(dcGenre);
25     dtSongs.Columns.Add(dcTime);
26
27     ds.Tables.Add(dtSongs);
28 }
```

De laatste stap is het inlezen van de data (lees: muzieknnummers) uit het XML bestand en in deze in de DataSet zetten. Dit gebeurt middels de ReadXml methode. Voeg de volgende code toe aan de GetAllSongs(string file) methode.

```
29 ds.ReadXml(file);
```

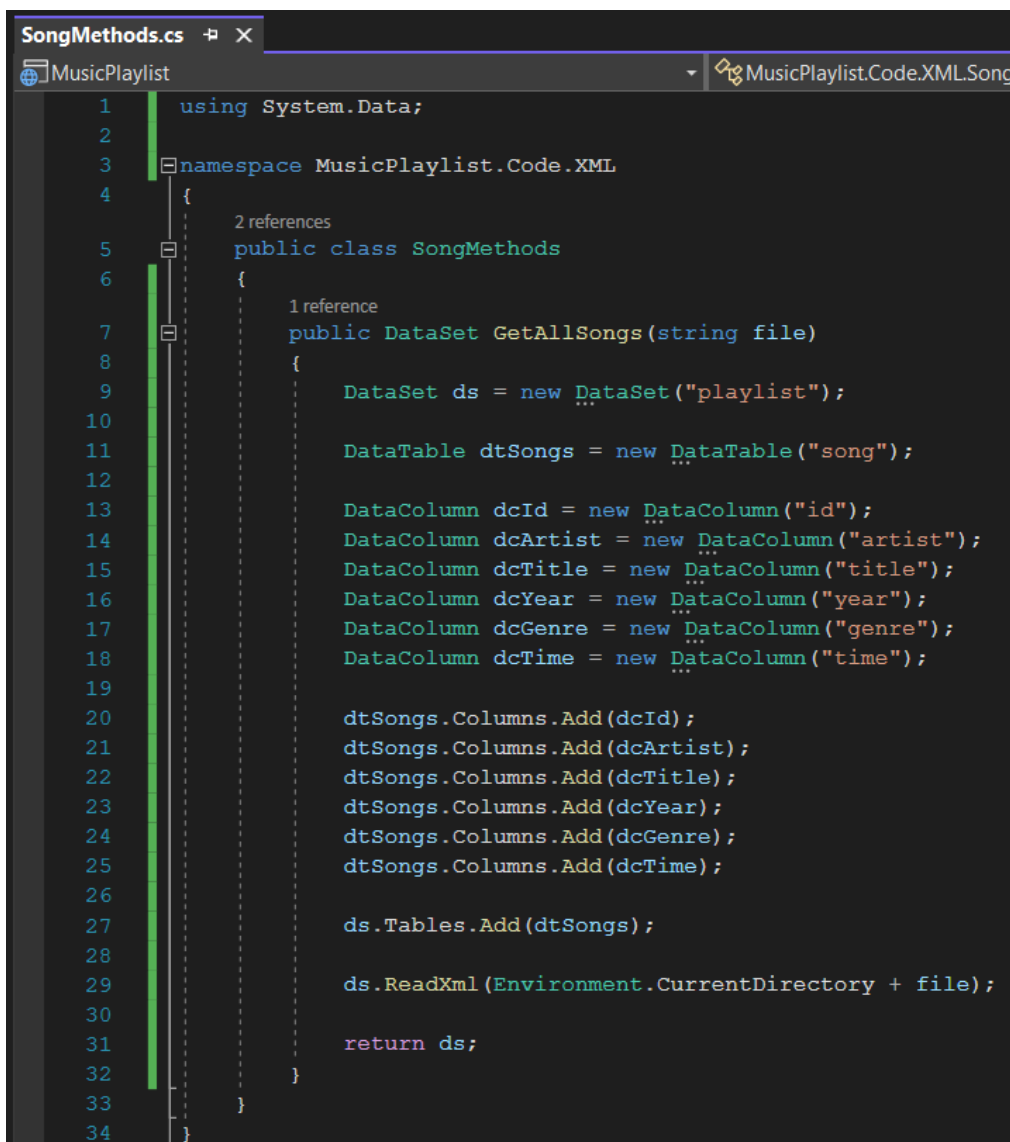
Let er wel op dat de webapplicatie zelf niet weet waar het XML bestand staat. De webserver van Visual Studio is standaard een IIS-express variant en start ergens bij de Program Files map. Dit is ook de plek waar de webapplicatie zal gaan zoeken naar het XML bestand, maar deze zal niet gevonden worden. Om de juiste plek van het aangemaakte XML bestand aan te geven dienen we gebruik te maken van de volgende code: **Environment.CurrentDirectory**. Deze variabele geeft het pad naar de huidige directory aan. De zojuist toegevoegde ReadXml code dient dus veranderd te worden inclusief deze Environment.CurrentDirectory:

```
29 ds.ReadXml(Environment.CurrentDirectory + file);
```

Als laatste stap voor dit onderdeel dienen we een variabele terug te geven (returnen), zoals we hebben aangegeven in de structuur van de GetAllSongs(string file) methode. Dit zullen we doen door als laatste code regel in GetAllSongs(string file) de volgende code op te nemen:

```
31 return ds;
```

Hiermee verdwijnt ook direct de foutmelding die we nog hadden bij de GetAllSongs(string file) methode.

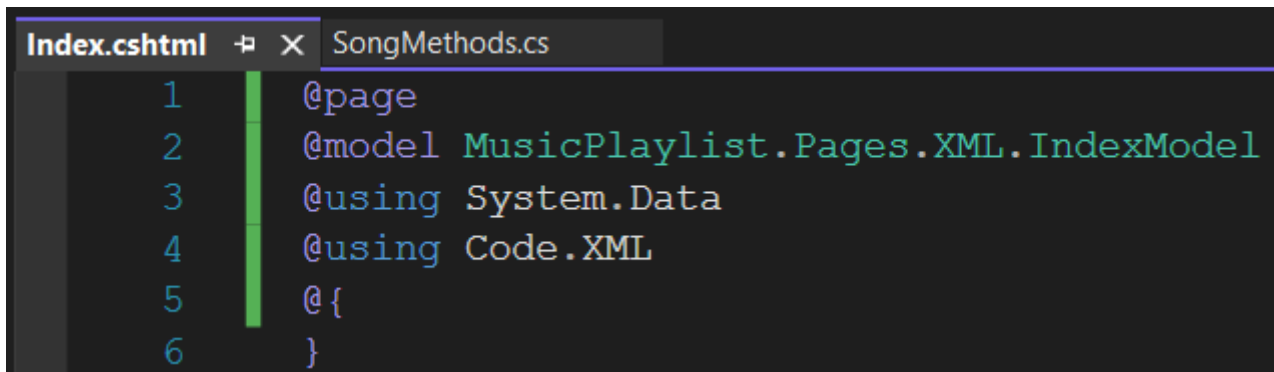


```
SongMethods.cs  X
MusicPlaylist  MusicPlaylist.Code.XML.Song
1  using System.Data;
2
3  namespace MusicPlaylist.Code.XML
4  {
5      2 references
6      public class SongMethods
7      {
8          1 reference
9          public DataSet GetAllSongs(string file)
10         {
11             DataSet ds = new DataSet("playlist");
12
13             DataTable dtSongs = new DataTable("song");
14
15             DataColumn dcId = new DataColumn("id");
16             DataColumn dcArtist = new DataColumn("artist");
17             DataColumn dcTitle = new DataColumn("title");
18             DataColumn dcYear = new DataColumn("year");
19             DataColumn dcGenre = new DataColumn("genre");
20             DataColumn dcTime = new DataColumn("time");
21
22             dtSongs.Columns.Add(dcId);
23             dtSongs.Columns.Add(dcArtist);
24             dtSongs.Columns.Add(dcTitle);
25             dtSongs.Columns.Add(dcYear);
26             dtSongs.Columns.Add(dcGenre);
27             dtSongs.Columns.Add(dcTime);
28
29             ds.Tables.Add(dtSongs);
30
31             ds.ReadXml(Environment.CurrentDirectory + file);
32
33             return ds;
34         }
35     }
36 }
```

Inhoud XML bestand zichtbaar maken in HTML

Na het inlezen van het XML bestand moet de data zichtbaar worden in een cshtml pagina. Dit doe je door gebruik te maken van de class `SongMethods` in het bestand `\Pages\XML\Index.cshtml`.

- Open het bestand `\Pages\XML\Index.cshtml`.
- Voeg bovenin het bestand de `System.Data` en `Code.XML` libraries toe:

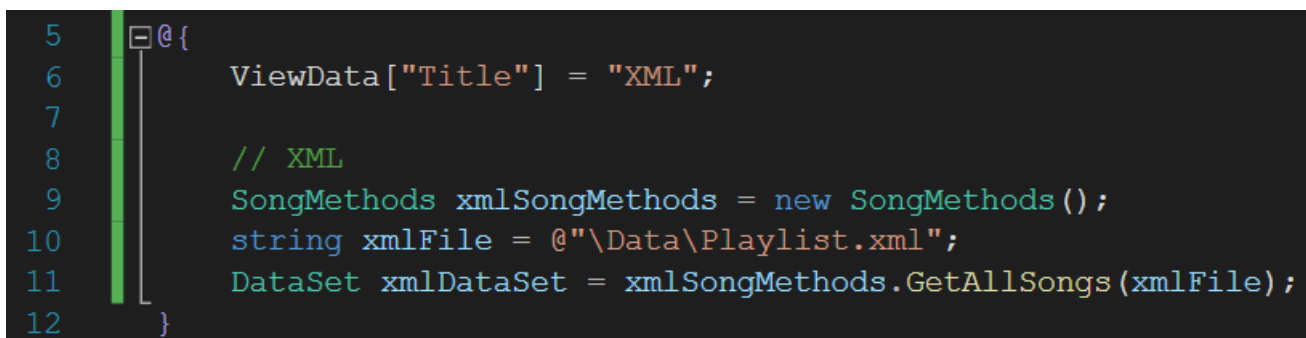


```
Index.cshtml  SongMethods.cs
1  @page
2  @model MusicPlaylist.Pages.XML.IndexModel
3  @using System.Data
4  @using Code.XML
5  @{
6  }
```

Let hierbij op dat deze code moet worden voorafgegaan door `@` omdat het C#-code is. `@` wordt in een cshtml bestand een Razor tag genoemd.

De `Code.XML` library hebben we nodig omdat onze `SongMethods` class hierin staat.

- Voeg nu in het `@{ }` code blok de volgende code toe:



```
5  @@{
6      ViewData["Title"] = "XML";
7
8      // XML
9      SongMethods xmlSongMethods = new SongMethods();
10     string xmlFile = @"Data\Playlist.xml";
11     DataSet xmlDataSet = xmlSongMethods.GetAllSongs(xmlFile);
12 }
```

Ook hier wordt gebruik gemaakt van Razor. In dit geval is het een Razor blok, omdat er meerdere C# statements achter elkaar worden uitgevoerd.

Er wordt op regel 10 gebruik gemaakt van een `@` voor het XML bestand om ervoor te zorgen dat de `\` goed wordt geïnterpreteerd. Neem je de `@` niet op dan kan je het pad alleen correct schrijven met dubbele `\\`, dus dan zou het pad worden `\\Data\\Playlist.xml`.

In regel 9 wordt een instantie van de class `SongMethods` gemaakt. Dit object heet **xmlSongMethods**.

In regel 10 wordt een deel van het pad van het XML bestand in de variabele **xmlFile** gezet.

In regel 11 wordt de methode `GetAllSongs(string file)` in het object `xmlSongMethods` uitgevoerd. In deze methode wordt een `DataSet` opgebouwd en gevuld met de data uit het XML bestand. Daarna geeft deze methode de gevulde `DataSet` terug. In regel 11 wordt die gevulde `DataSet` in de variabele **xmlDataSet** gezet.

Het uitlezen van de DataSet kan door middel van een foreach loop die door alle DataRows (lees: muzieknnummers) loopt die in de DataTable zitten.

```
14 <div class="text-center">
15     <h1 class="display-4">XML</h1>
16     @foreach (DataRow drSong in xmlDataSet.Tables["song"].Rows)
17     {
18
19     }
20 </div>
```

Deze foreach loop gaat rij voor rij door de tabel met de naam **song** heen en zet de gevonden rij telkens in een variabele genaamd **drSong**, deze is van het type **DataRow**.

De DataRow drSong kan uitgelezen worden als een soort Array. Gebruik hiervoor de blokhaken.

HTML en C# code mag door elkaar gebruikt worden (met Razor tags). Hieronder zie je een voorbeeld:

```
14 <div class="text-center">
15     <h1 class="display-4">XML</h1>
16     <table class="table">
17         <thead>
18             <tr>
19                 <th scope="col">Id</th>
20                 <th scope="col">Artiest</th>
21                 <th scope="col">Titel</th>
22                 <th scope="col">Jaar</th>
23                 <th scope="col">Genre</th>
24                 <th scope="col">Tijd</th>
25             </tr>
26         </thead>
27         <tbody>
28             @foreach (DataRow drSong in xmlDataSet.Tables["song"].Rows)
29             {
30                 <tr>
31                     <th scope="row">@drSong["id"]</th>
32                     <td>@drSong["artist"]</td>
33                     <td>@drSong["title"]</td>
34                     <td>@drSong["year"]</td>
35                     <td>@drSong["genre"]</td>
36                     <td>@drSong["time"]</td>
37                 </tr>
38             }
39         </tbody>
40     </table>
41     <br />
42     <br />
43     <a href="/XML/Create">Een nieuwe song toevoegen.</a>
44 </div>
```

Voordat we de webapplicatie kunnen starten dienen we de \Pages\XML\Index.cshtml pagina beschikbaar te maken in de menu navigatie. In de \Pages\Shared\ folder staat een bestand genaamd **_Layout.cshtml**, hier wordt de layout van de webapplicatie opgebouwd.

- Open het bestand **\Pages\Shared_Layout.cshtml**.
- Kopieer één van de menu items en pas deze code aan voor de XML Index pagina:

```
20 <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
21 <ul class="navbar-nav flex-grow-1">
22 <li class="nav-item">
23 <a class="nav-link text-dark" asp-area="" asp-page="/Index">Home</a>
24 </li>
25 <li class="nav-item">
26 <a class="nav-link text-dark" asp-area="" asp-page="/XML/Index">XML</a>
27 </li>
28 <li class="nav-item">
29 <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>
30 </li>
31 </ul>
32 </div>
```

- Start de webapplicatie en klik op het XML menu item. De data (lees: muzieknnummers) uit het XML bestand wordt nu zichtbaar gemaakt in de webpagina.

MusicPlaylist Home XML Privacy

XML

Id	Artiest	Titel	Jaar	Genre	Tijd	Verwijderen
1	Twenty One Pilots	Stressed Out	2016	Rap-rock	3:22	Verwijderen
2	Twenty One Pilots	Heathens	2016	Rap-rock	3:15	Verwijderen

[Een nieuwe song toevoegen](#)

© 2022 - MusicPlaylist - [Privacy](#)

Een nieuw muziknummer toevoegen aan de playlist

Aanpassingen op de bestaande webapplicatie

Er dient een aanpassing te worden gemaakt in de SongMethods class.

- Verplaats de regel `DataSet ds = new DataSet("playlist");` in de methode `GetAllSongs(string file)` naar de bovenkant van de class (buiten de methode `GetAllSongs(string file)`).

```
5 | 2 references  
6 | public class SongMethods  
7 | {  
8 |     private DataSet ds = new DataSet("playlist");  
9 |  
10 |     1 reference  
11 |     public DataSet GetAllSongs(string file)  
    |     {  
    |         DataTable dtSongs = new DataTable("song");
```

Deze wijziging moet je toepassen omdat het object `ds` ook in andere methodes binnen deze class moet kunnen worden gebruikt. Door deze variabele uit de methode `GetAllSongs(string file)` te halen maak je de variabele globaal. In de oude situatie kon je het object alleen binnen de methode `GetAllSongs(string file)` gebruiken.

- Maak een nieuwe methode in de class `SongMethods` aan met de naam `GetEmptyDataRow()`.

```
34 | 0 references  
35 | public DataRow GetEmptyDataRow()  
36 | {  
37 |     DataRow dr = ds.Tables["song"].NewRow();  
38 |     return dr;  
    | }
```

In de regel `DataRow dr = ds.Tables["song"].NewRow();` wordt een nieuwe `DataRow` gecreëerd. Deze `DataRow` bevat nog geen gegevens, maar heeft wel dezelfde structuur als een `DataRow` in de `DataTable` met de naam `song`. Deze methode zullen we straks nodig zijn.

- Maak een nieuwe methode in de class `SongMethods` aan met de naam `CreateSong(...)`. Deze methode heeft twee parameters.

```
40 | 0 references  
41 | public void CreateSong(DataRow dr, string file)  
42 | {  
43 |     ds.Tables["song"].Rows.Add(dr);  
44 |     ds.Xml(Environment.CurrentDirectory + file);  
    | }
```

De eerste parameter bij de `CreateSong` methode is van het type `DataRow`. In deze `DataRow` zullen de gegevens van het nieuwe muziknummer komen te staan. De tweede parameter is van het type `string` en hier zal het pad naar het XML bestand komen te staan.

In de regel `ds.Tables["song"].Rows.Add(dr);` wordt de `DataRow dr` in de collectie `Rows` van de `DataTable` met de naam `song` toegevoegd. Tot slot worden de gegevens uit de `DataSet` in het XML bestand weggeschreven (lees: overschreven).

Nieuwe pagina toevoegen

- Maak een nieuwe cshtml pagina in de folder `\Pages\XML\` en noem deze pagina **Create.cshtml**. Zorg ervoor dat je via deze pagina een nieuw muzieknummer kunt toevoegen. Hieronder zie je hoe zo'n webpagina er uit kan zien. Je dient dit HTML formulier dus zelf te coderen.

MusicPlaylist Home XML Privacy

Voeg een nieuw muzieknummer toe

<input type="text" value="Id"/>	<input type="text" value="Id"/>
<input type="text" value="Artist"/>	<input type="text" value="Artist"/>
<input type="text" value="Title"/>	<input type="text" value="Title"/>
<input type="text" value="Year"/>	<input type="text" value="Year"/>
<input type="text" value="Genre"/>	<input type="text" value="Genre"/>
<input type="text" value="Time"/>	<input type="text" value="Time"/>

© 2022 - MusicPlaylist - [Privacy](#)

Visual Studio Code

- Voeg vervolgens bovenin de `\Pages\XML\Create.cshtml` pagina de volgende code toe:

```
1 @page
2 @model MusicPlaylist.Pages.XML.CreateModel
3 @using System.Data
4 @using Code.XML
5 @{
6     ViewData["Title"] = "Voeg een nieuw muzieknummer toe";
7
8     // XML
9     SongMethods xmlSongMethods = new SongMethods();
10    string xmlFile = @"\Data\Playlist.xml";
11    DataSet xmlDataSet = xmlSongMethods.GetAllSongs(xmlFile);
12 }
```

- Voeg nu het volgende stuk code toe aan het C# code blok op deze pagina.

```

1  @page
2  @model MusicPlaylist.Pages.XML.CreateModel
3  @using System.Data
4  @using Code.XML
5  @{
6      ViewData["Title"] = "Voeg een nieuw muzieknummer toe";
7
8      // XML
9      SongMethods xmlSongMethods = new SongMethods();
10     string xmlFile = @"Data\Playlist.xml";
11     DataSet xmlDataSet = xmlSongMethods.GetAllSongs(xmlFile);
12
13     if (HttpMethods.IsPost(Request.Method))
14     {
15         DataRow drSong = xmlSongMethods.GetEmptyDataRow();
16
17         drSong[0] = Request.Form["id"];
18         drSong[1] = Request.Form["artist"];
19         drSong[2] = Request.Form["title"];
20         drSong[3] = Request.Form["year"];
21         drSong[4] = Request.Form["genre"];
22         drSong[5] = Request.Form["time"];
23
24         xmlSongMethods.CreateSong(drSong, xmlFile);
25         Response.Redirect("/XML/Index");
26     }
27 }

```

De regel **if (HttpMethods.IsPost(Request.Method))** controleert of er een POST verstuurd is, oftewel wanneer een formulier met de method POST verstuurd wordt.

In de regel **drSong[0] = Request.Form["id"];** wordt aan de eerste cel van de nieuwe DataRow een waarde toegevoegd. Deze waarde is een POST uit het formulier (in dit geval uit een textbox met name="id"). **Request.Form["id"]** is te vergelijken met **\$_POST["id"]** in PHP. In de regels 17 t/m 22 wordt de DataRow dus gevuld met de waardes die zijn ingevuld in het formulier.

Op regel 24 wordt de methode **CreateSong(...)** in het object **xmlSongMethods** uitgevoerd met de parameters **drSong** en **xmlFile**. De nieuwe DataRow wordt in de DataSet gezet en daarna opgeslagen in het XML bestand.

In regel 25 geef je aan de server de opdracht de pagina **/XML/Index** naar de client (jouw computer) te sturen.

Request en Response

In de code die je zojuist hebt toegevoegd heb je het **Request-** en het **Responseobject** gebruikt. Dit zijn objecten binnen het .NET framework, maar wanneer gebruik je deze objecten nu precies?

Een **request** is een aanvraag die een client (b.v. jouw computer) naar een server stuurt. In ons geval wordt een post request met data uit een form naar de server gestuurd. Binnen de server kan iets met die data worden gedaan (met server-side C#- of PHP-code). Een post is dus een voorbeeld van een request.

Een **response** is een reactie van een server (bijvoorbeeld op een request van een client). De regel **Response.Redirect("/XML/Index");** is een request, maar de client vraagt daarin de server een response, namelijk: stuur de pagina **/XML/Index** naar de client. Op de client wordt deze pagina dan getoond. Een omleiding naar een andere webpagina (redirecten) naar de client is dus een voorbeeld van een response.

Een muzieknummer verwijderen uit de playlist

Nieuwe pagina toevoegen

- Maak een nieuwe cshtml pagina in de folder `\Pages\XML\` en noem deze pagina **Delete.cshtml**. Zorg ervoor dat je via deze pagina een muzieknummer kunt verwijderen. Hieronder zie je hoe zo'n webpagina er uit kan zien. Je dient dit HTML formulier dus zelf te coderen.

MusicPlaylist Home XML Privacy

Verwijder een muzieknummer

Weet u zeker dat u dit muzieknummer wil verwijderen?

Ja
 Nee

Verwijderen

© 2022 - MusicPlaylist - [Privacy](#)

- Voeg bovenin de `\Pages\XML\Delete.cshtml` pagina de volgende code toe:

```
1 @page
2 @model MusicPlaylist.Pages.XML.DeleteModel
3 @using System.Data
4 @using Code.XML
5 @{}
6     ViewData["Title"] = "Verwijder een muzieknummer";
7
8     // XML
9     SongMethods xmlSongMethods = new SongMethods();
10    string xmlFile = @"\Data\Playlist.xml";
11    DataSet xmlDataSet = xmlSongMethods.GetAllSongs(xmlFile);
12 }
```

- Pas het bestand \Pages\XML\Index.cshtml aan door op regel 25 een kolom toe te voegen voor de verwijder links. Daarna voeg je regel 38 toe om per muziknummer een verwijder link te maken. Bij de verwijder link bouwen we de URL op met een `id` parameter. Deze parameter nemen we op zodat we de verwijder actie kunnen uitvoeren specifiek voor een gekozen muziknummer met dat id.

```
14 <div class="text-center">
15   <h1 class="display-4">XML</h1>
16   <table class="table">
17     <thead>
18       <tr>
19         <th scope="col">Id</th>
20         <th scope="col">Artiest</th>
21         <th scope="col">Titel</th>
22         <th scope="col">Jaar</th>
23         <th scope="col">Genre</th>
24         <th scope="col">Tijd</th>
25         <th scope="col">Verwijderen</th>
26       </tr>
27     </thead>
28     <tbody>
29       @foreach (DataRow drSong in xmlDataSet.Tables["song"].Rows)
30       {
31         <tr>
32           <th scope="row">@drSong["id"]</th>
33           <td>@drSong["artist"]</td>
34           <td>@drSong["title"]</td>
35           <td>@drSong["year"]</td>
36           <td>@drSong["genre"]</td>
37           <td>@drSong["time"]</td>
38           <td><a href="/XML/Delete?id=@drSong["id"]">Verwijderen</a></td>
39         </tr>
40       }
41     </tbody>
42   </table>
43   <br />
44   <br />
45   <a href="/XML/Create">Een nieuwe song toevoegen.</a>
46 </div>
```

De class SongMethods aanpassen

- Maak een nieuwe methode in de class SongMethods aan met de naam DeleteSong(...). Deze methode heeft twee parameters. De methode heeft de volgende twee parameters: file (die zijn we al tegengekomen bij de methode CreateSong) en id. Deze laatste is het id van het muzieknnummer die verwijderd moet worden.

```
0 references
46 public void DeleteSong(string id, string file)
47 {
48     DataRow[] drSongs = ds.Tables["song"].Select("id = '" + id + "'");
49     if (drSongs != null && drSongs.Length > 0)
50     {
51         drSongs[0].Delete();
52         ds.WriteXml(Environment.CurrentDirectory + file);
53     }
54 }
```

In de code wordt gebruik gemaakt van de methode Select van de collection Tables. Deze methode gebruiken we om op zoek te gaan naar een muzieknnummer met een specifieke waarde voor de id kolom. Als de DataRow met het juiste id gevonden wordt, dan wordt deze opgeslagen in een array **drSongs**. Mochten er muzieknnummers zijn gevonden dan wordt het eerste element van die drSongs array verwijderd en daarna worden de gegevens in de DataSet opgeslagen (lees: overschreven) in het XML bestand.

\Pages\XML\Delete.cshtml pagina gereed maken voor gebruik

Tot slot moeten we de pagina \Pages\XML\Delete.cshtml weer aanpassen.

- Voeg nu het volgende stuk code toe aan het C# code blok op deze pagina.

```
1 @page
2 @model MusicPlaylist.Pages.XML.DeleteModel
3 @using System.Data
4 @using Code.XML
5 @{
6     ViewData["Title"] = "Verwijder een muzieknnummer";
7
8     // XML
9     SongMethods xmlSongMethods = new SongMethods();
10    string xmlFile = @"Data\Playlist.xml";
11    DataSet xmlDataSet = xmlSongMethods.GetAllSongs(xmlFile);
12
13    if (HttpMethods.IsPost(Request.Method))
14    {
15        if (Request.Form["inputOptie"] == "Ja")
16        {
17            string id = Request.Query["id"];
18            xmlSongMethods.DeleteSong(id, xmlFile);
19        }
20
21        Response.Redirect("/XML/Index");
22    }
23 }
```

Als het formulier gepost is, dan wordt eerst nagegaan of de radiobutton met value="ja" is geselecteerd. Let er wel op dat in jouw gemaakte HTML formulier dan radiobuttons moeten zijn opgenomen met de naam **inputOptie** en één van deze twee radiobuttons moet als value **Ja** hebben.

In de regel **string id = Request.Query["id"];** wordt de waarde van de parameter **id** in de url gelezen en in de variabele **id** geplaatst. De code **Request.Query["id"]** is te vergelijken met **\$_GET["id"]** in PHP.

Een muzieknummer wijzigen in de playlist

De volgende stap is het wijzigen van een muzieknummer, maar hier ontvang je geen stappenplan voor. Er is een combinatie benodigd van het zoeken naar een correct muzieknummer (zoals bij verwijderen gedaan is) en het aanbrengen van wijzigingen in een muzieknummer middels een formulier op een cshtml pagina.

Een eerste blik op JSON en YAML

Wanneer je klaar bent met de XML variant van de music playlist dien je de webapplicatie uit te breiden met een JSON (en wellicht zelfs een YAML) playlist. Via de volgende link is een eerste blik mogelijk hoe JSON en YAML er ongeveer uit zien en wat de verschillen zijn t.o.v. XML:

<http://sangsoonam.github.io/2017/03/13/yaml-vs-json.html>.

Om gebruik te maken van JSON is het aan te raden om te werken met Newtonsoft. Wanneer je begint met het toevoegen van een JSON music playlist zoek dan op Google op termen als **C# JSON Newtonsoft**.

Om YAML toe te passen is het gebruik van de library genaamd **YamlDotNet** een aanrader.