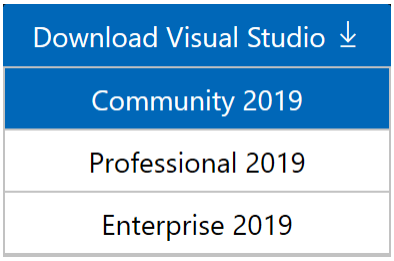
Tutorialr.com

Spotify for Developers Getting Started

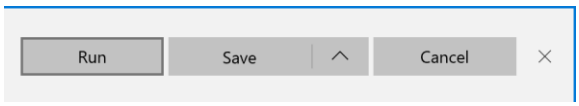
Setup & Start

Step 1

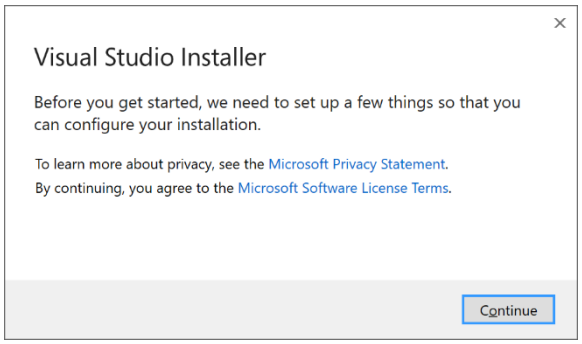
Firstly, you will need to install **Visual Studio 2019 Community**, if not done already by doing the following:



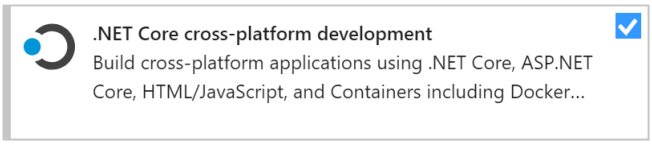
Visit **VisualStudio.com** and then from the **Visual Studio IDE** section choose **Download Visual Studio** then **Community 2019**



Next on the **Thank you for downloading Visual Studio** page when the download prompt appears, select **Run**



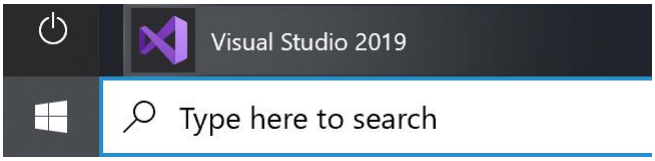
Once downloaded, this should start the **Visual Studio Installer** and select **Continue** to begin the installation



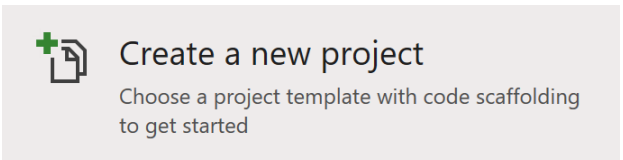
Next once ready select **.NET Core cross-platform development** from the **Workloads** section

Step 2

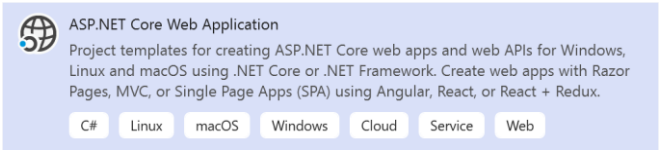
Next if **Visual Studio 2019 Community** is installed, you can then start **Visual Studio 2019 Community** and **Create a new project**, by doing the following:



In **Windows 10** choose **Start**, and then from the **Start Menu** find and select **Visual Studio 2019**



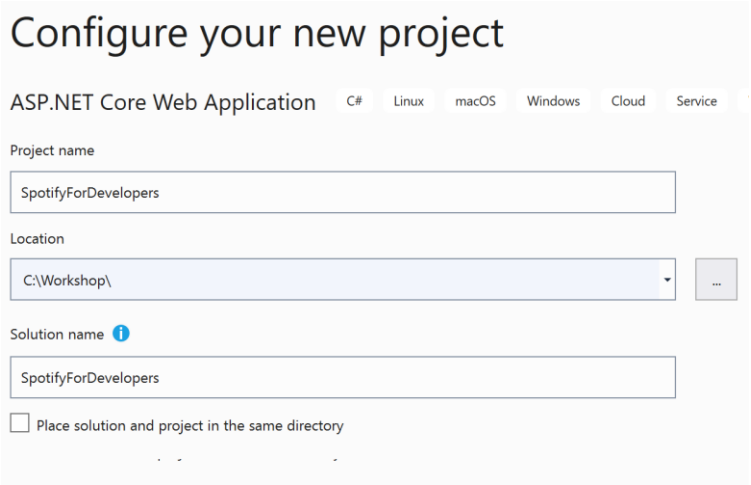
Once done, from the **Get started** screen for **Visual Studio 2019** select **Create a new project**



Then choose **ASP.NET Core Web Application** and select **Next**

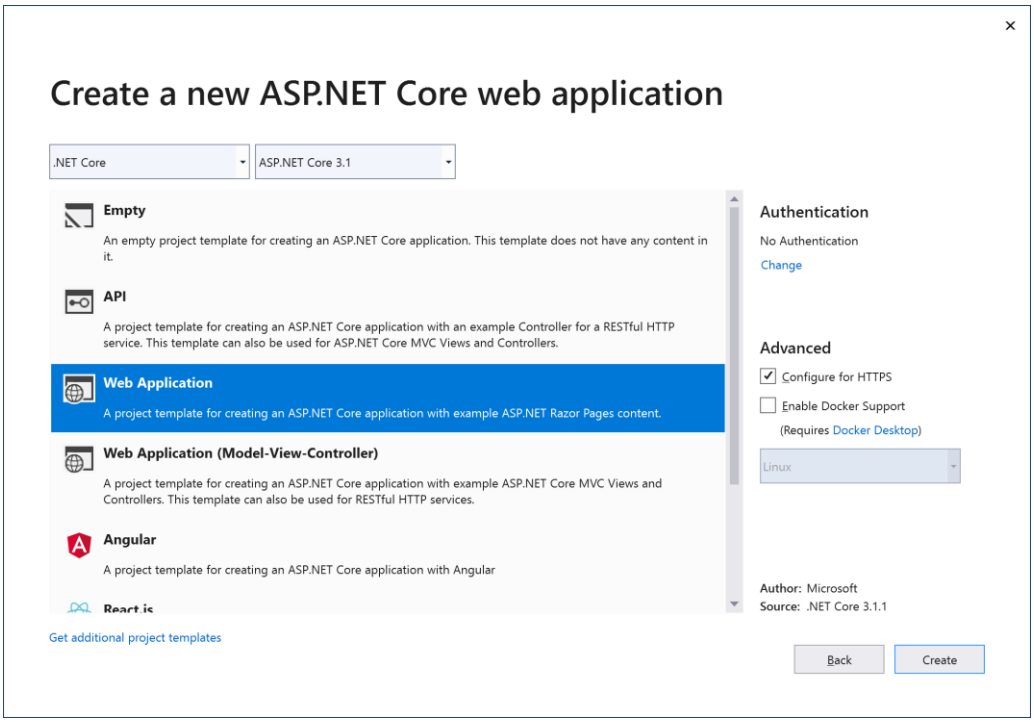
Step 3

Next, in **Configure your new project** enter a **Project name** as **SpotifyForDevelopers** and then choose a **Location** and then select **Create**

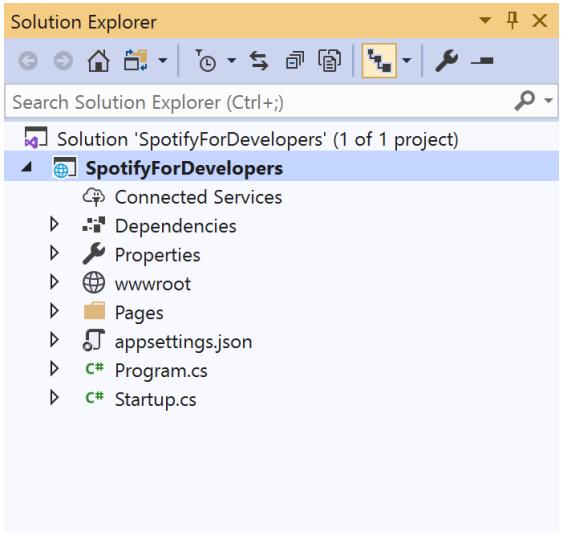


Step 4

Then, in **Create a new ASP.NET Core web application** make sure from the two dropdowns that **.NET Core** and **ASP.NET Core 3.1** are selected, and **Web Application** is selected from the list and in **Advanced** configure for HTTPS has been selected then select **Create**

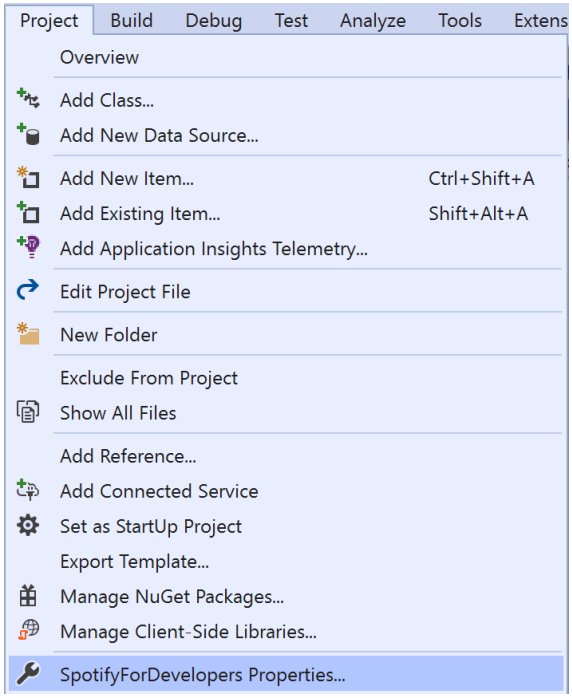


Step 5



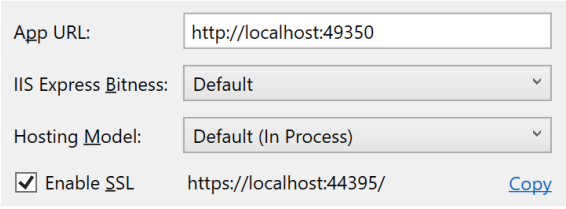
In the **Solution Explorer** of **Visual Studio 2019** select the **Project** for **SpotifyForDevelopers**

Step 6



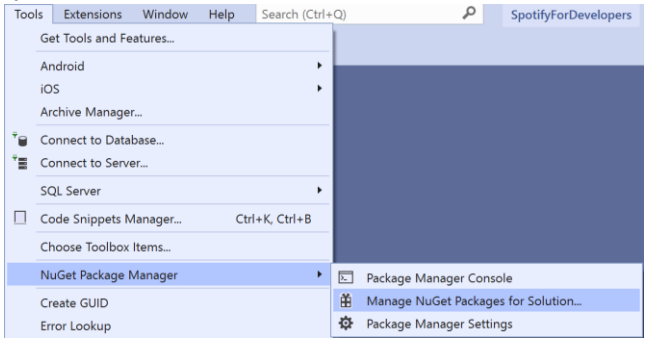
From the **Menu** of **Visual Studio 2019** select **Project** then **SpotifyForDevelopers Properties...**

Step 7



Then in **Properties** select the **Debug** section and then in the **Web Server Settings** select the **Copy** option to **Copy** the URL to the **Clipboard** e.g. https://localhost:44395/ - please note that your URL may be different. Then **Paste** the contents into your text editor such as **Notepad** this will be used as a **Redirect URI** for use with the **Spotify API**

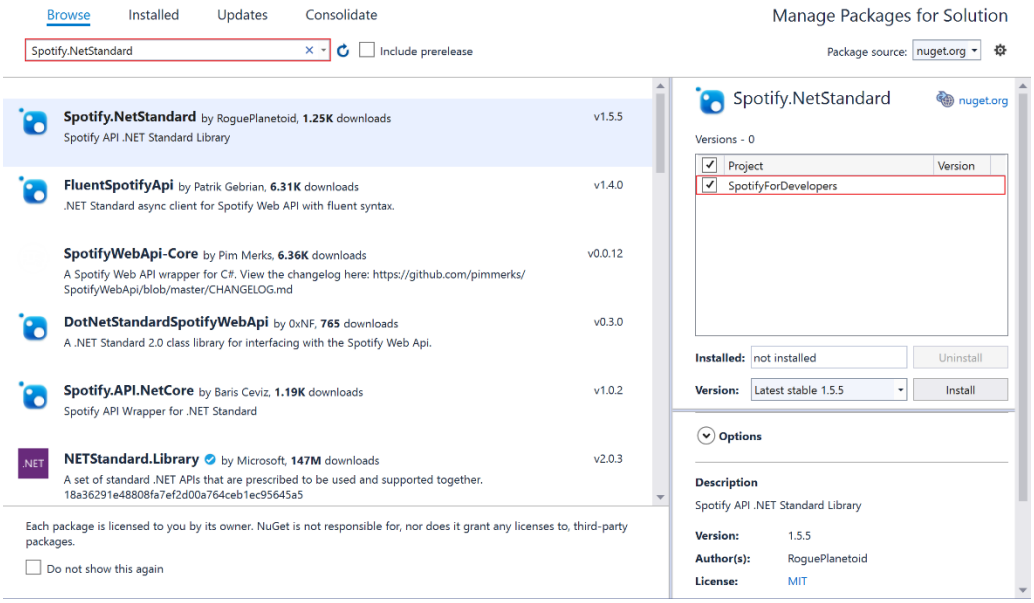
Step 8



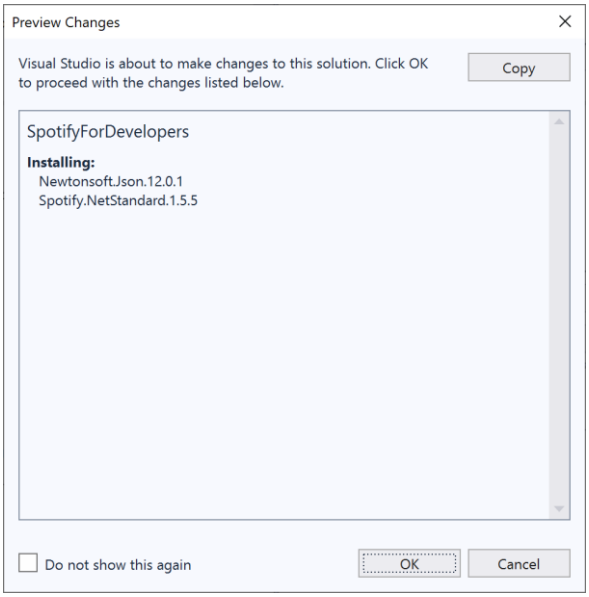
From the **Menu** of **Visual Studio 2019** select **Tools** then select **NuGet Package Manager** and **Manage NuGet Packages for Solution...**

Step 9

Then in **NuGet** select **Browse** and search for **Spotify.NetStandard by RoguePlanetoid** as indicated and select **Spotify.NetStandard** then check the box under **Project** as indicated and select **Install**.



Step 10

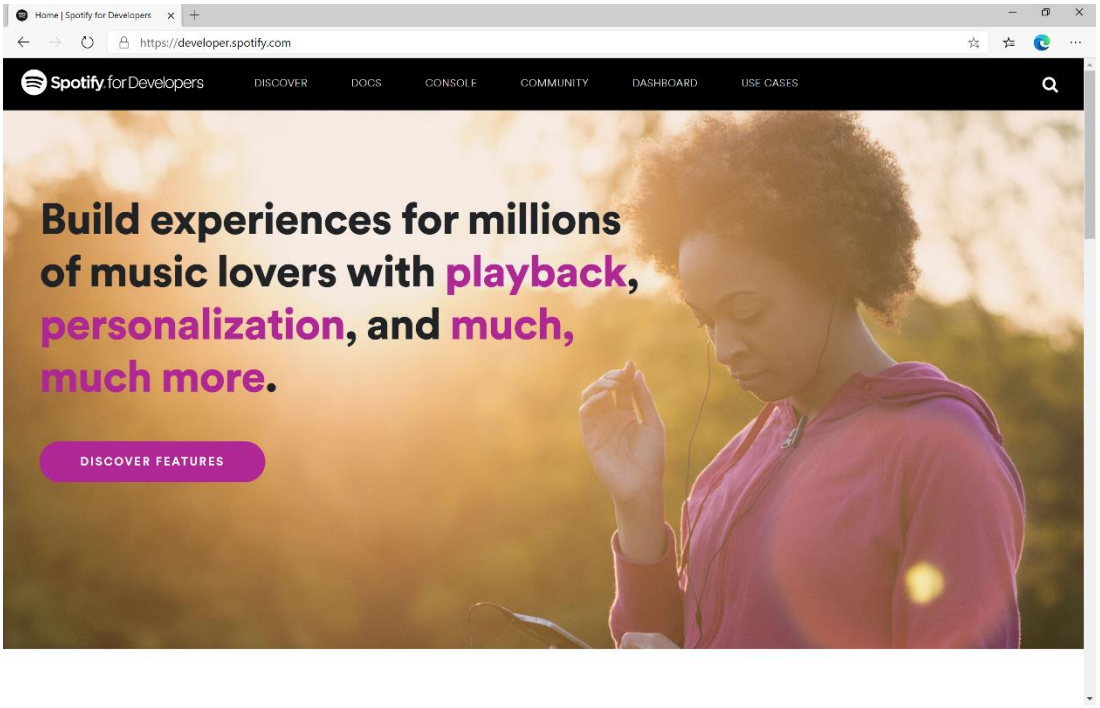


Then, if **Preview Changes** is displayed, select **OK** then or otherwise the **NuGet** package **Spotify.NetStandard** will be installed and then make sure to keep **Visual Studio 2019** open as you'll come back to it later.

Dashboard & Settings

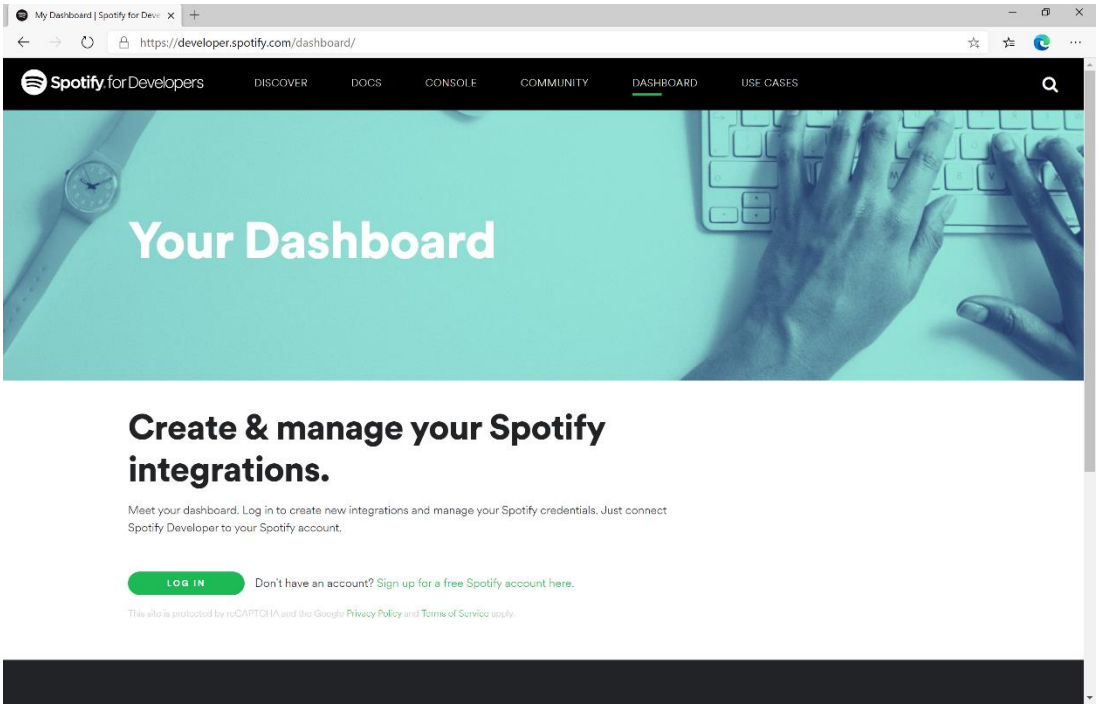
Step 1

Start your favourite **Web Browser** such as **Microsoft Edge** and navigate to **developer.spotify.com** for the **Spotify for Developers** website.



Step 2

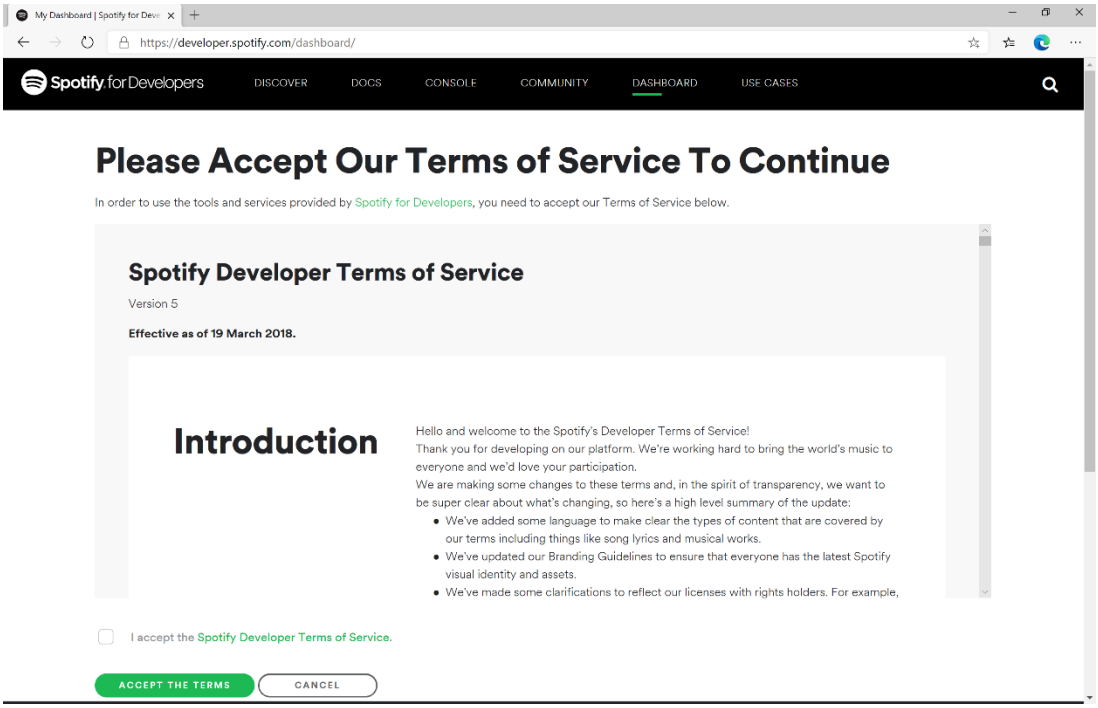
Then on the **Spotify for Developers** website select **Dashboard** then on the **Your Dashboard Page** choose **Log In** to sign in with a **Spotify Account**



If you don't have a **Spotify** account already you can use the **Sign up for a free Spotify account here** option to get a **Spotify** account – you will already have one if you have a **Spotify** subscription or already listen to their music service.

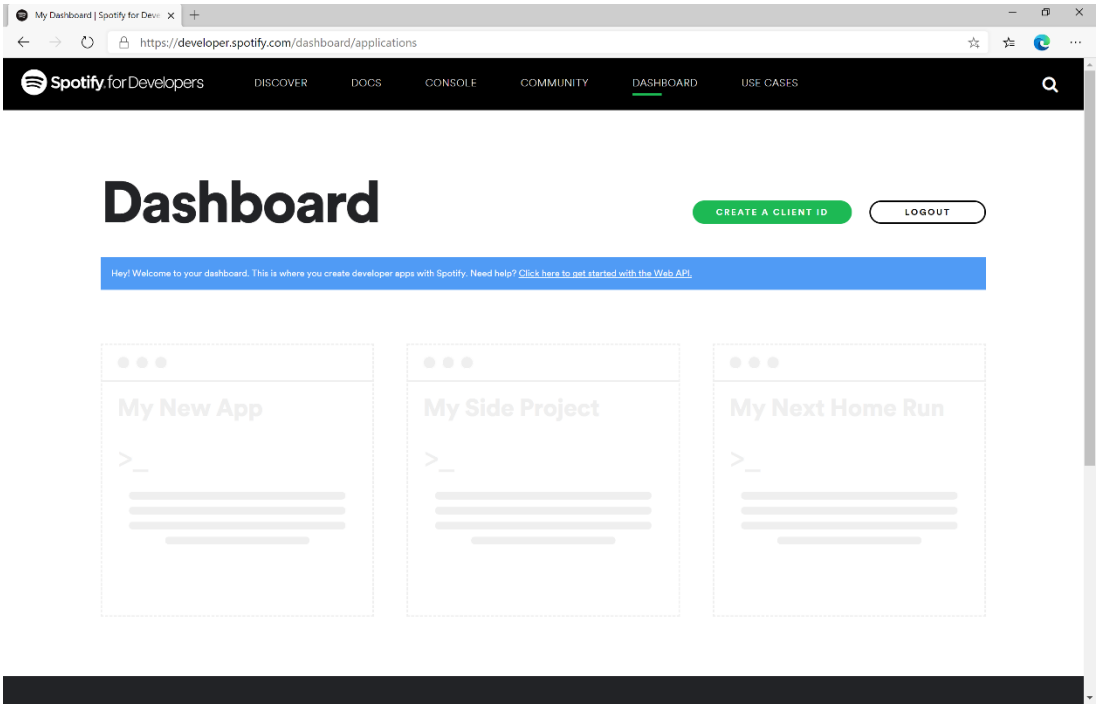
Step 3

Once you've signed in with a **Spotify** account for the first time you will need to read through the **Spotify Developer Terms of Service** then once done select the **I accept the Spotify Developer Terms of Service** and then select **Accept the Terms** to continue.



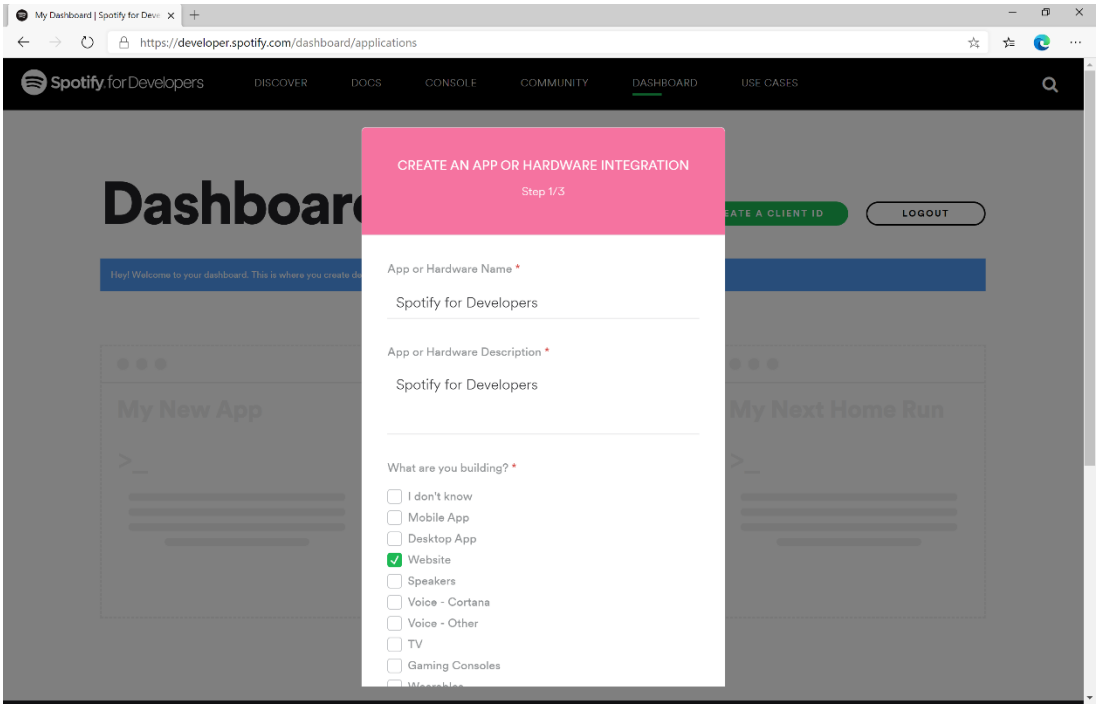
Step 4

Once signed in with a **Spotify** Account and agreed to the **Spotify Developer Terms of Service** you'll be taken to the **Dashboard**



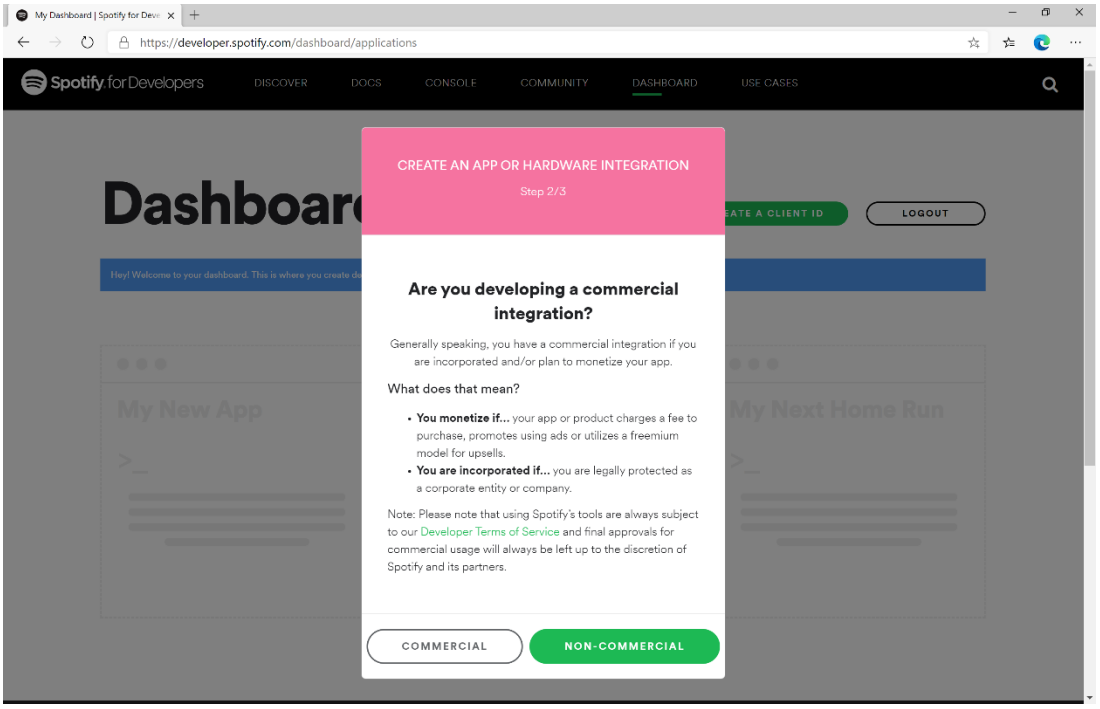
Step 5

Then in the **Dashboard** select **Create a Client Id** then in **Step 1/3** of **Create an App or Hardware Integration** you need to enter the **App or Hardware Name** as **Spotify for Developers** and the **App or Hardware Description** as **Spotify for Developers** then under **What are you building** tick the **Website** option and select **Next**.



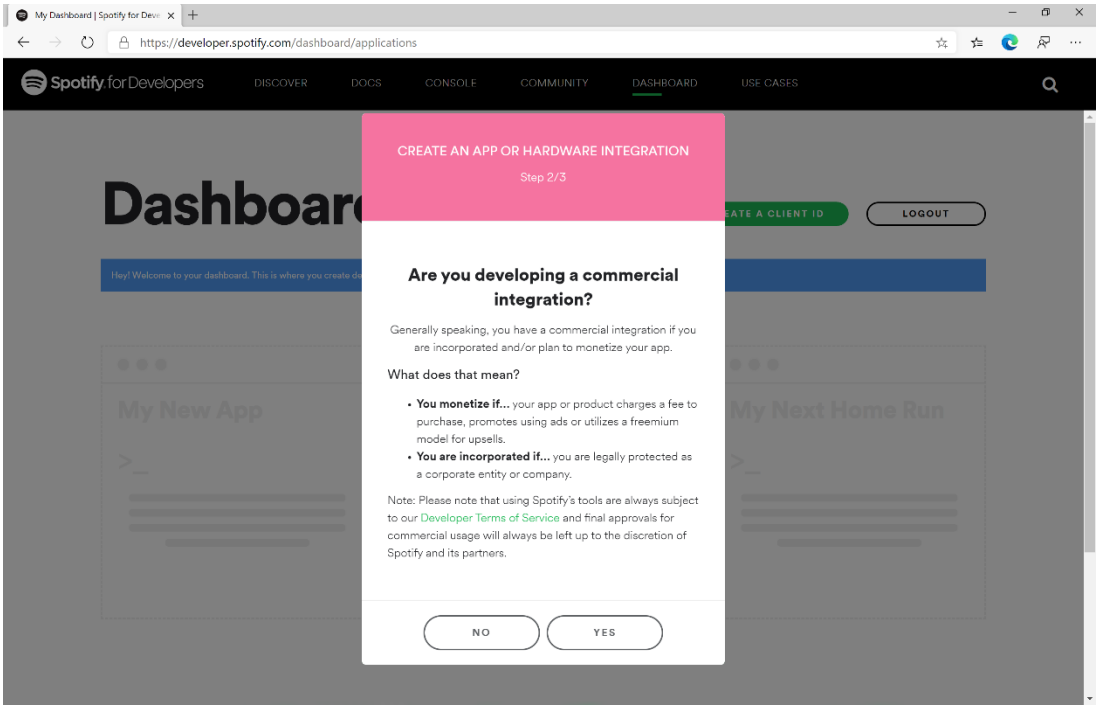
Step 6

Then in **Step 2/3** of **Create an App or Hardware Integration** for **Are you developing a commercial integration** select **Non-Commercial**.



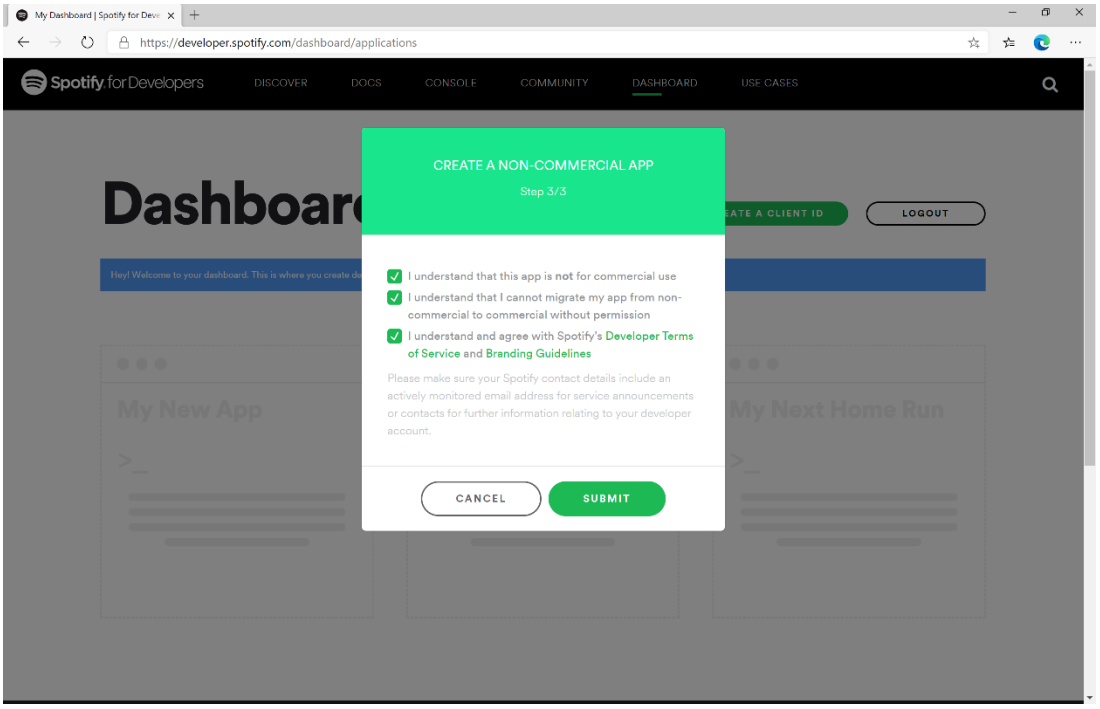
Step 7

Then in **Step 2/3** of **Create an App or Hardware Integration** for the question **Are you developing a commercial integration** select **No**.



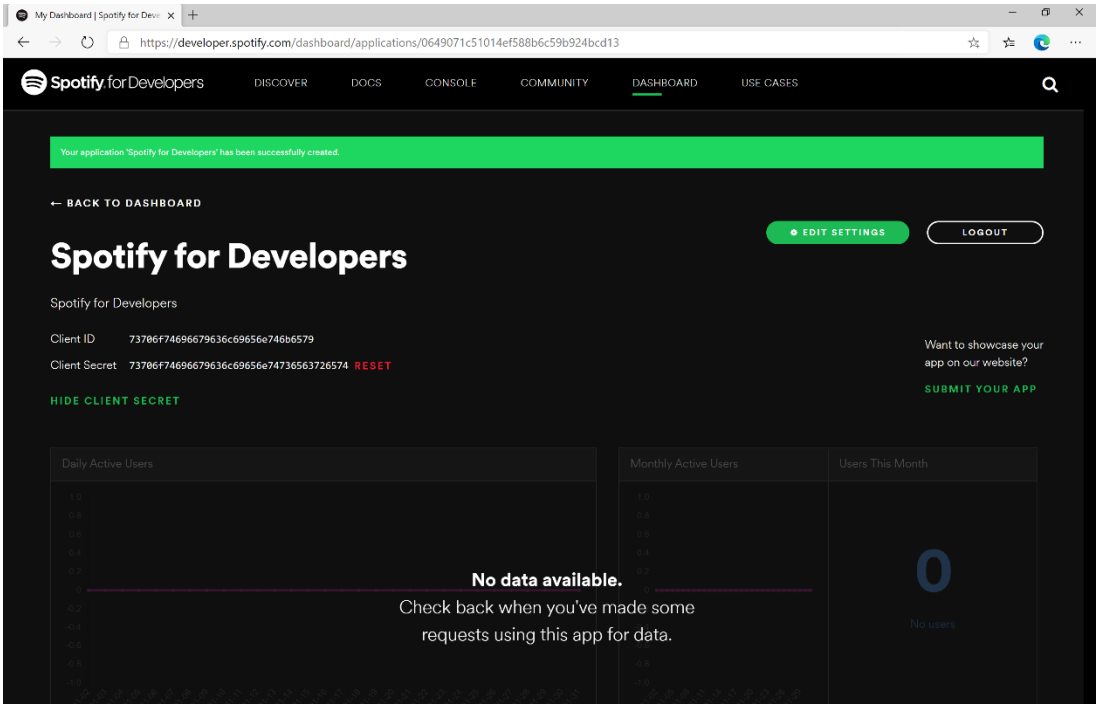
Step 8

Then in **Step 3/3** of **Create an App or Hardware Integration** then tick the option for **I understand that this app is not for commercial use**, tick the option for **I understand that I cannot migrate my app from non-commercial to commercial without permission** and follow and read the linked pages then tick the option for **I understand and agree with Spotify's Developer Terms of Service and Branding Guidelines,** then choose **Submit**



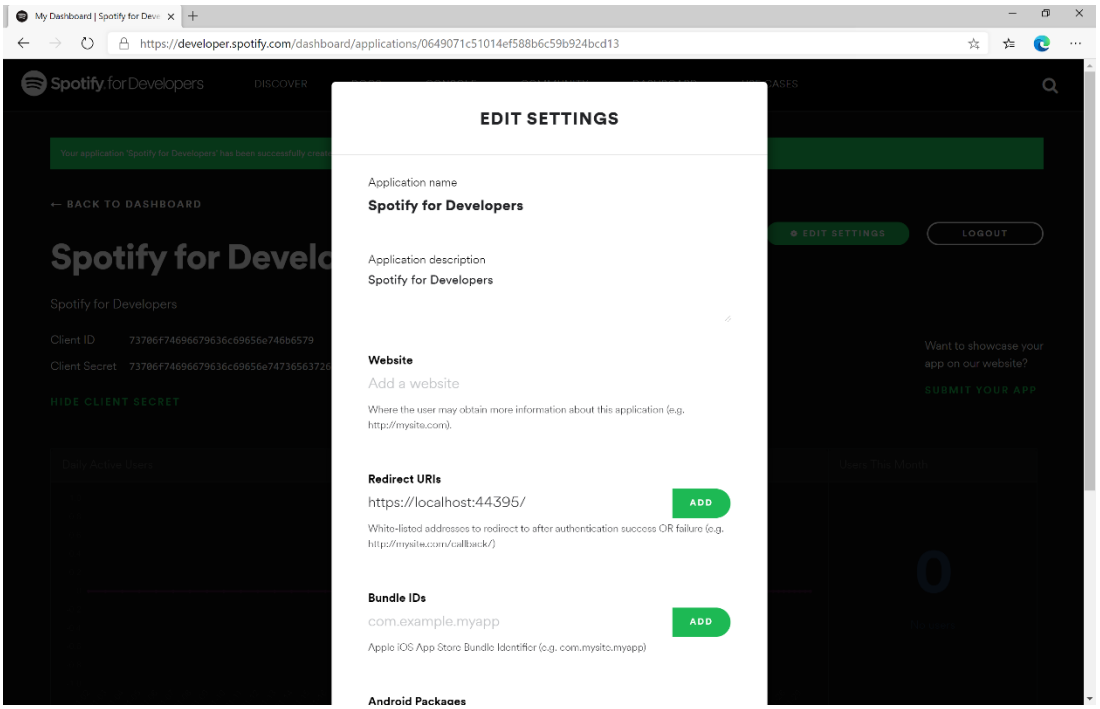
Step 9

Once created, you will need to **Copy** the **Client ID** to the **Clipboard** e.g. 73706f74696679636c69656e746b6579 – note that your id will be different, then **Paste** the contents into a text editor such as **Notepad** where you copied the **Redirect URI** previously. Then select **Show Client Secret** and then **Copy** the **Client Secret** to the **Clipboard** e.g. 73706f74696679636c69656e74736563726574 – your secret will be different, then **Paste** this into your text editor.



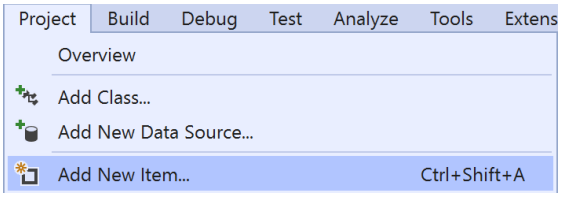
Step 10

Finally, in your text editor such as **Notepad** you need to **Copy** the **Redirect URI** e.g. https://localhost:44395/ - please note that your **Redirect URI** may be different. Then in the **Dashboard** choose **Edit Settings** then **Paste** the contents of your **Clipboard** into the **Redirect URIs** and select **Add** then to complete the process select **Save**.



Token & Result

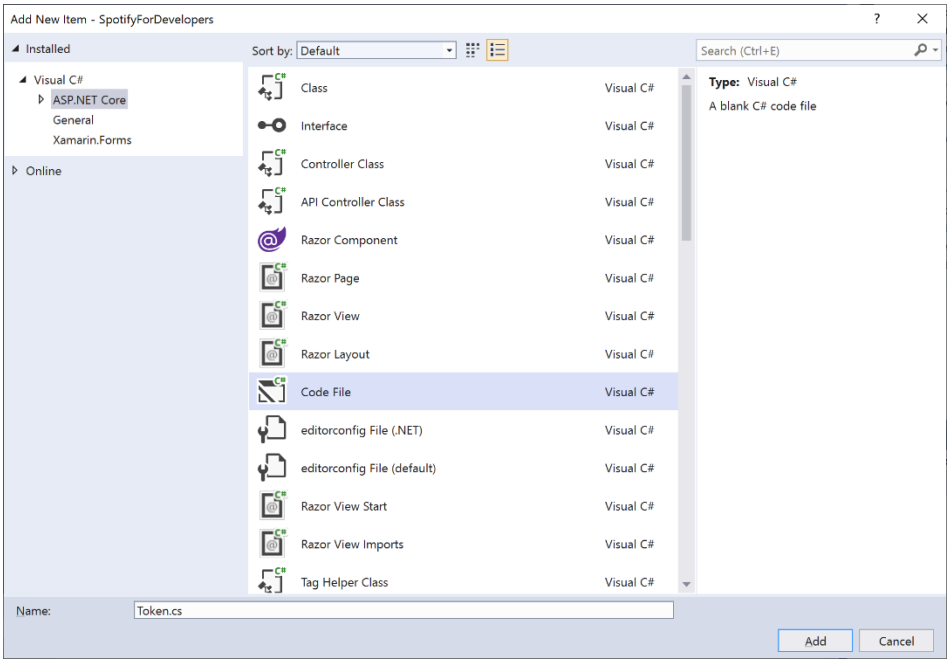
Step 1



First return to **Visual Studio 2019** and then from the **Menu** choose **Project** then **Add New Item...**

Step 2

Then, from the **Add New Item** window from **Installed** select **Visual C#** then **ASP .NET Core** and select **Code File** from the list, then type in the **Name** as **Token.cs** before selecting **Add** to add the file to the **Project**



Step 3

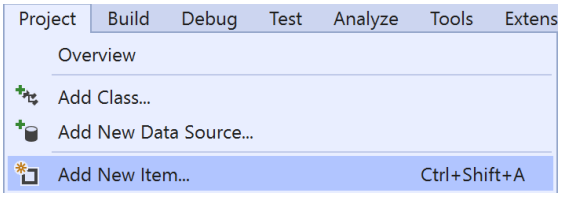
Once in the **Code View** for **Token.cs** the following should be entered:

```
using Spotify.NetStandard.Client.Authentication;
using Spotify.NetStandard.Client.Authentication.Enums;
using System.Text.Json;

public class Token
{
    public AccessToken AccessToken { get; set; }
    public bool HasUserToken => AccessToken?.TokenType == TokenType.User;
    public bool HasToken => AccessToken?.TokenType == TokenType.Access || HasUserToken;
    public Token(AccessToken token) => AccessToken = token;
    public Token(string content) =>
        AccessToken = JsonSerializer.Deserialize<AccessToken>(content);
    public override string ToString() => JsonSerializer.Serialize(AccessToken);
}
```

This will represent an `AccessToken` which is part of the **NuGet** package for **Spotify.NetStandard** and will allow the **AccessToken** to be stored and retrieved in **JSON** format using `System.Text.Json`.

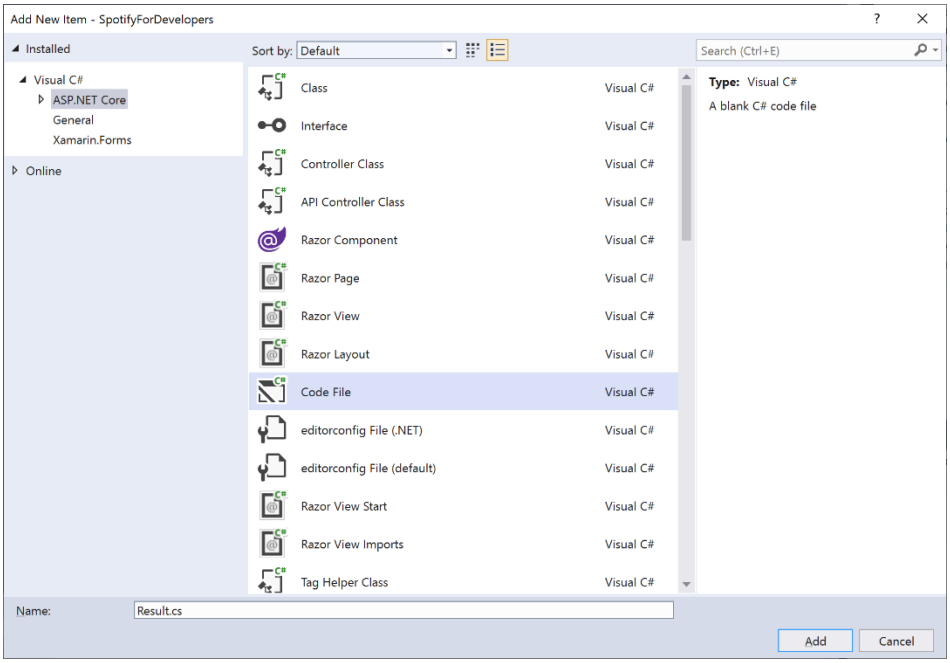
Step 4



Then again from the **Menu** choose **Project** then **Add New Item...**

Step 5

Then, from the **Add New Item** window from **Installed** select **Visual C#** then **ASP .NET Core** and select **Code File** from the list, then type in the **Name** as **Result.cs** before selecting **Add** to add the file to the **Project**



Step 6

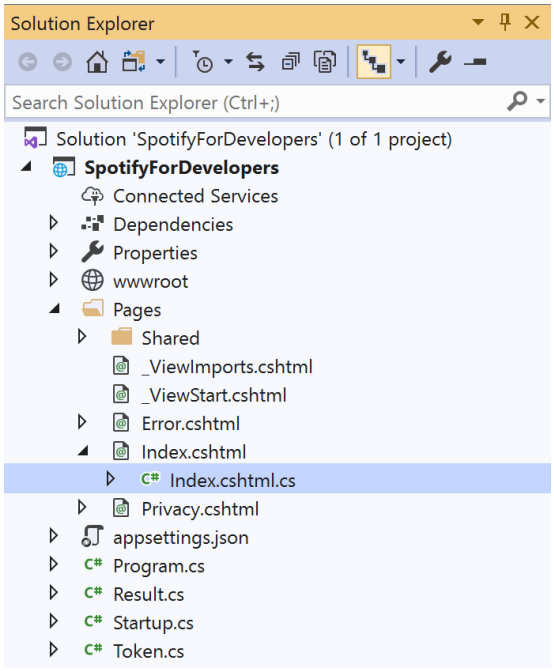
Once in the **Code View** for **Result.cs** the following should be entered:

```
public class Result
{
    public string Id { get; set; }
    public string Name { get; set; }
    public string Image { get; set; }
    public Result Inner { get; set; }

    public Result(
        string id = null,
        string name = null,
        string image = null,
        Result inner = null)
    {
        Id = id;
        Name = name;
        Image = image;
        Inner = inner;
    }
}
```

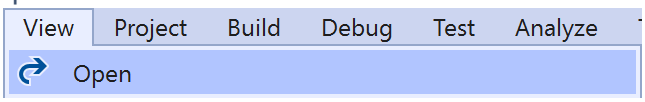
This will represent any response to be displayed from **NuGet** package for **Spotify.NetStandard**.

Step 7



In the **Solution Explorer** open the **Pages** section, then open the **Index.cshtml** section and select **Index.cshtml.cs**

Step 8



Then from the **Menu** choose **View** and then **Open**

Step 9

Once in the **Code View** for **Index.cshtml.cs** above the `namespace SpotifyForDevelopers.Pages` add the following `using` statements:

```
using Microsoft.AspNetCore.Http;
using Spotify.NetStandard.Client;
using Spotify.NetStandard.Client.Interfaces;
using Spotify.NetStandard.Requests;
using Spotify.NetStandard.Enums;
```

Then below `private readonly ILogger<IndexModel> _logger;` add the following `const` and `readonly` values:

```
private const string client = "clientid";
private const string secret = "clientsecret";
private const string state = "spotify.workshop";
private const string token = "token";
private const string country = "GB";

public static readonly ISpotifyApi Api = SpotifyClientFactory.CreateSpotifyClient(
    client, secret).Api;
```

You will need to **Copy** the **Client ID** to the **Clipboard** that you saved in your text editor e.g. **Notepad** and then **Paste** to replace `clientid` e.g. 73706f74696679636c69656e746b6579 - note that your id will be different. You will then need to **Copy** the **Client Secret** to the **Clipboard** from your text editor then **Paste** to replace `clientsecret` e.g. 73706f74696679636c69656e74736563726574 - note that your secret will be different. If done correctly those values should appear like the following:

```
private const string client = "73706f74696679636c69656e746b6579";
private const string secret = "73706f74696679636c69656e74736563726574";
```

You can also set the `country` to your e.g. `US` for United States or any supported Country such as `GB` for Great Britain.

Step 10

While still in the **Code View** for **Index.cshtml.cs** above `public IndexModel(ILogger<IndexModel> logger)` add the following **properties**:

```
public Token Token { get; set; }
public string Value { get; set; }
public string Option { get; set; }
public bool Flag { get; set; }
public IFormFile Upload { get; set; }
public IEnumerable<Result> Results { get; set; }
public Uri RedirectUri => new
Uri($"{HttpContext.Request.Scheme}://{HttpContext.Request.Host}");
public Uri CurrentUri => new
Uri($"{HttpContext.Request.Scheme}://{HttpContext.Request.Host}{HttpContext.Request.Path}{Htt
pContext.Request.QueryString}");
```

These **properties** will represent the **Token** and **Result** plus other values that will come in useful later.

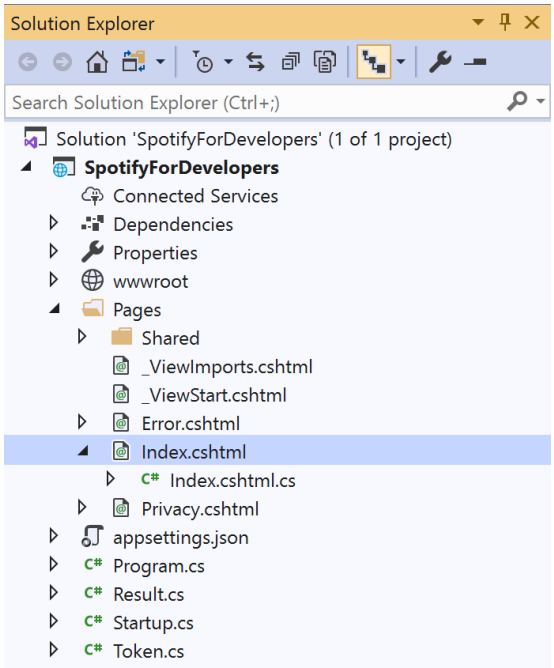
Then below the `public Uri CurrentUri` **property** add the following **methods**:

```
public void LoadToken()
{
    if (Request.Cookies[token] != null)
        Token = new Token(Request.Cookies[token]);
}

public void SaveToken()
{
    if (Token != null)
        Response.Cookies.Append(token, Token.ToString());
}
```

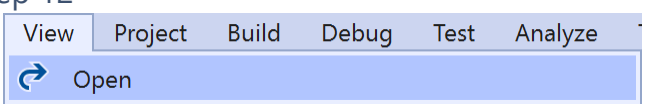
The `LoadToken` **method** will get a `Token` from a **Cookie** and the `SaveToken` **method** will add a `Token` to a given **Cookie**, a **Cookie** is a way to save information for a while in a Web Browser.

Step 11



In the **Solution Explorer** in the **Pages** section select **Index.cshtml**

Step 12



Then from the **Menu** choose **View** and then **Open**

Step 13

Once in the **Code View** for **Index.cshtml** remove all the existing content, which should be like the following:

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with
ASP.NET Core</a>.</p>
</div>
```

Then, once removed you need to replace it with the following:

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Spotify for Developers";
}
<h1>Spotify for Developers</h1>
<div class="container">
    <div class="row mb-2">
        <div class="mx-auto">

            <!—Authorisation Guide -->
        </div>
    </div>
    <div class="row mb-2">
        <div class="col-sm">
            @if (Model?.Token?.HasToken == true)
            {
                <h2 class="text-center">Spotify Web API App Authorisation</h2>

                <!-- Spotify Web API App Authorisation -->
            }
        </div>
        <div class="col-sm">
            @if (Model?.Token?.HasUserToken == true)
            {
                <h2 class="text-center">Spotify Web API User Authorisation</h2>

                <!-- Spotify Web API User Authorisation -->
            }
        </div>
    </div>

    <!-- Results -->
</div>
```

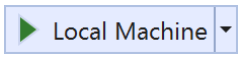
This represents the basic layout where you'll place the various elements in subsequent parts, and with the **<!-- -->** statements which will help you place things correctly as you'll always be adding anything new above those statements.

Step 14

Then, while still in the **Code View** for **Index.cshtml** above `<!-- Results -->` enter the following:

```
<div class="row mb-2">
    @if (Model?.Results?.Count() > 0)
    {
        var first = Model?.Results.First();
        <h2>Results</h2>
        <table class="table">
            <thead>
                <tr>
                    @if (first.Image != null)
                    {
                        <th>Image</th>
                    }
                    @if (first.Id != null)
                    {
                        <th>Id</th>
                    }
                    @if (first.Name != null)
                    {
                        <th>Name</th>
                    }
                    @if (first?.Inner?.Id != null)
                    {
                        <th>Id</th>
                    }
                    @if (first?.Inner?.Name != null)
                    {
                        <th>Name</th>
                    }
                </tr>
            </thead>
            <tbody>
                @foreach (var result in Model.Results)
                {
                    <tr>
                        @if (result.Image != null)
                        {
                            <td><img height="64" width="64" src="@result.Image" /></td>
                        }
                        @if (result.Id != null)
                        {
                            <td>@result.Id</td>
                        }
                        @if (result.Name != null)
                        {
                            <td>@result.Name</td>
                        }
                        @if (result?.Inner?.Id != null)
                        {
                            <td>@result.Inner.Id</td>
                        }
                        @if (result?.Inner?.Name != null)
                        {
                            <td>@result.Inner.Name</td>
                        }
                    </tr>
                }
            </tbody>
        </table>
    }
</div>
```

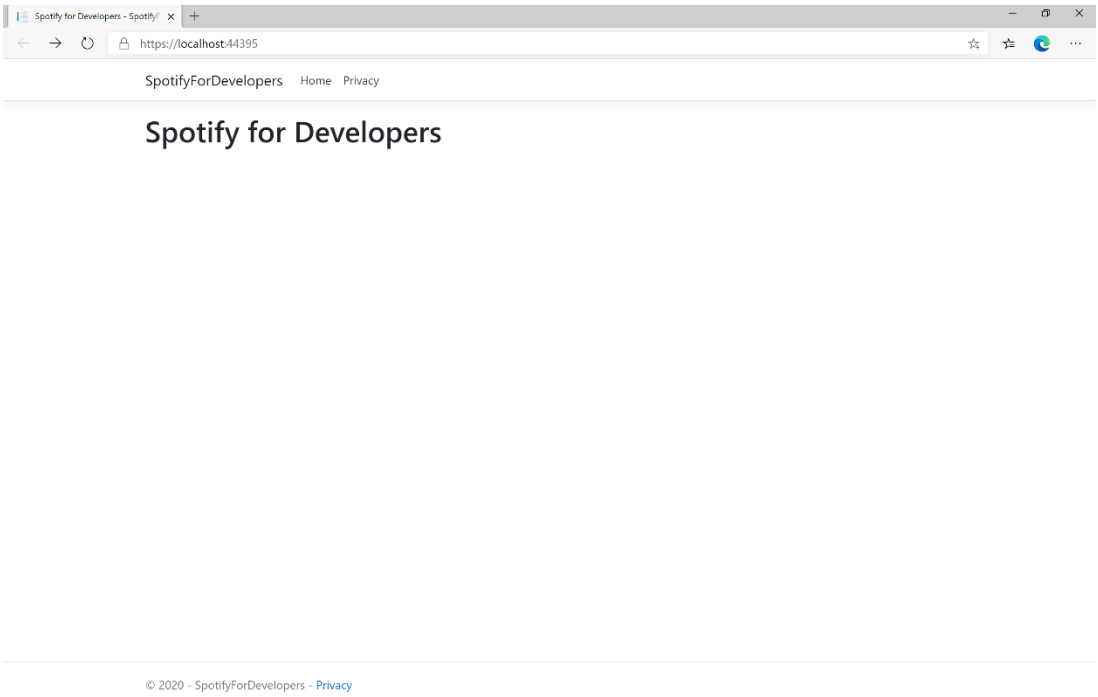
Step 15



Finally, in **Visual Studio 2019** select **IIS Application** to run the **Web Application**

Step 16

Once the **Web Application** is running it should appear something like the following:



Step 17

You can stop the **web application** in **Visual Studio 2019** by selecting the **Stop debugging** button

Step 18

You can exit **Visual Studio 2019** by selecting the **Close** button in the top right of the **application** as that completes this part of the workshop