Cours ASP.NET CORE

Réalisé par : Malek ZRIBI

Prérequis du cours

- Html
- Css
- C#
- Connaissance de MVC est préférable mais pas obligatoire

CHAPITRE 1 : Concepts de base ASP.NET CORE

Plan du Chapitre

- Le Framework .NET CORE
- ASP.NET CORE
- Installation
- Créer une application ASP.NET CORE
- Structure d’un projet
- Les éléments d’un projet :
  - le fichier projet .csproj
  - Les dépendances
  - Les propriétés
  - les fichiers de configuration lunchsettings.json et appsettings.json
  - le dossier wwwroot
  - le fichier program.cs
- Notion de Middleware
- Configurer le pipeline d’exécution des middleware
- StaticFiles
- DefaultFiles
- FileServer
- Les Variables d'environnement

Le Framework .NET CORE

- .NET Core est la nouvelle version de .NET Framework

- Plate-forme de développement gratuite, open source et polyvalente développée par Microsoft.

- Il s'agit d'un Framework multiplateforme qui s'exécute sur les systèmes d'exploitation Windows, macOS et Linux.

- .NET Core Framework peut être utilisé pour créer différents types d'applications telles que mobile, bureau, web, cloud, IoT, machine learning, microservices, jeux, etc.

Le Framework .NET CORE

- .NET Core est écrit à partir de zéro pour en faire un Framework modulaire, léger, rapide et multiplateforme.
- Inclut les fonctionnalités de base requises pour exécuter une application .NET Core.
- D'autres fonctionnalités sont fournies sous forme de packages NuGet, que vous pouvez ajouter dans votre application si nécessaire.
- .NET Core accélère les performances des applications, réduit l'encombrement mémoire et facile à maintenir.

Le Framework .NET CORE

Caractéristiques .NET Core

- **Framework open-source.**
- **Multiplateforme :** .NET Core fonctionne sur les systèmes d'exploitation Windows, macOS et Linux. Il existe différents runtime pour chaque système d'exploitation qui exécute le code et génère la même sortie.
- **Cohérence entre les architectures:** exécutez le code avec le même comportement dans différentes architectures de jeu d'instructions, y compris x64, x86 et ARM.
- **Large gamme d'applications:** Différents types d'applications peuvent être développés et exécutés sur la plate-forme .NET Core tels que mobile, bureau, web, cloud, IoT, apprentissage automatique, microservices, jeux, etc.

Le Framework .NET CORE

- **Prise en charge de plusieurs langages:** vous pouvez utiliser les langages de programmation C#, F# et Visual Basic pour développer des applications .NET Core. Vous pouvez utiliser votre IDE préféré parmi Visual Studio 2017/2019, Visual Studio Code, Sublime Text, Vim, etc.

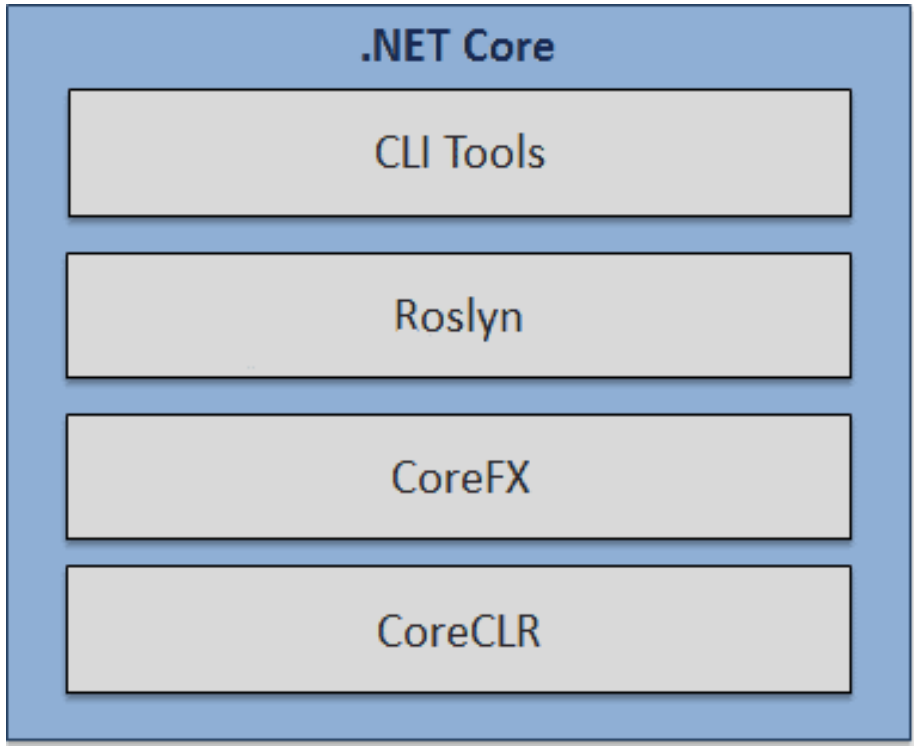
- **Architecture modulaire:** .NET Core prend en charge l'approche de l'architecture modulaire à l'aide des packages NuGet. Il existe différents packages NuGet pour diverses fonctionnalités qui peuvent être ajoutés au projet .NET Core selon les besoins.

- **Outils CLI:** L'interface de ligne de commande (CLI) est un nouvel outil multiplateforme pour installer des packages, et pour la création, l'exécution et la publication d'applications .NET.

Le Framework .NET CORE

Composition du .NET Core :

- **Outils CLI**: Command Line Interface: ensemble d'outils pour le développement et le déploiement.
- **Roslyn**: compilateur de langage pour C# et Visual Basic.
- **CoreFX**: ensemble de bibliothèques du framework.
- **CoreCLR**: un CLR basé sur JIT (Commun Language Runtime).



Le Framework .NET CORE- Historique

Version	Date de sortie	Sorti avec	Dernière mise à jour	Dernière date de mise à jour	Fin du support
.NET Core 1.0	27/06/2016	Mise à jour 3 de Visual Studio 2015	1.0.16	14/05/2019	27 juin 2019
.NET Core 1.1	16/11/2016	Visual Studio 2017 version 15.0	1.1.13	14/05/2019	27 juin 2019
.NET Core 2.0	14/08/2017	Visual Studio 2017 version 15.3	2.0.9	10/07/2018	1 octobre 2018
.NET Core 2.1	30/05/2018	Visual Studio 2017 version 15.7	2.1.19	09/06/2020	21 août 2021
.NET Core 2.2	04/12/2018	Visual Studio 2019 version 16.0	2.2.8	2019-11-19	23 décembre 2019
.NET Core 3.0	23/09/2019	Visual Studio 2019 version 16.3	3.0.3	18/02/2020	3 mars 2020
.NET Core 3.1	03/12/2019	Visual Studio 2019 version 16.4	3.1.7	2020-08-11	**3 décembre 2022**
.NET 5	11/2020	Visual Studio 2019 version 16.9			


ASP.NET
Core

Le framework Web ASP.NET CORE

- ASP.NET Core est la nouvelle version du Framework Web ASP.NET principalement destinée à s'exécuter sur la plate-forme .NET Core.
- ASP.NET Core est un Framework gratuit, open source et multiplateforme pour la création d'applications web, basées sur le cloud, des applications IoT et des backends mobiles.
- Framework modulaire avec un minimum de composants, d'autres fonctionnalités peuvent être ajoutées en tant que packages NuGet selon les exigences de l'application.
- Performances élevées, nécessite moins de mémoire, moins de taille de déploiement et plus facile à tester et entretenir par rapport à ASP.NET 4.
- ASP.NET CORE est une refonte de ASP.NET 4.x conçue pour .NET CORE

Caractéristiques de ASP.NET CORE

Cross Platform

Plus facile à maintenir et tester

1 seul modèle pour MVC et Web Api

Open source

Injection de dépendances

Modulaire

Pourquoi ASP.NET Core?

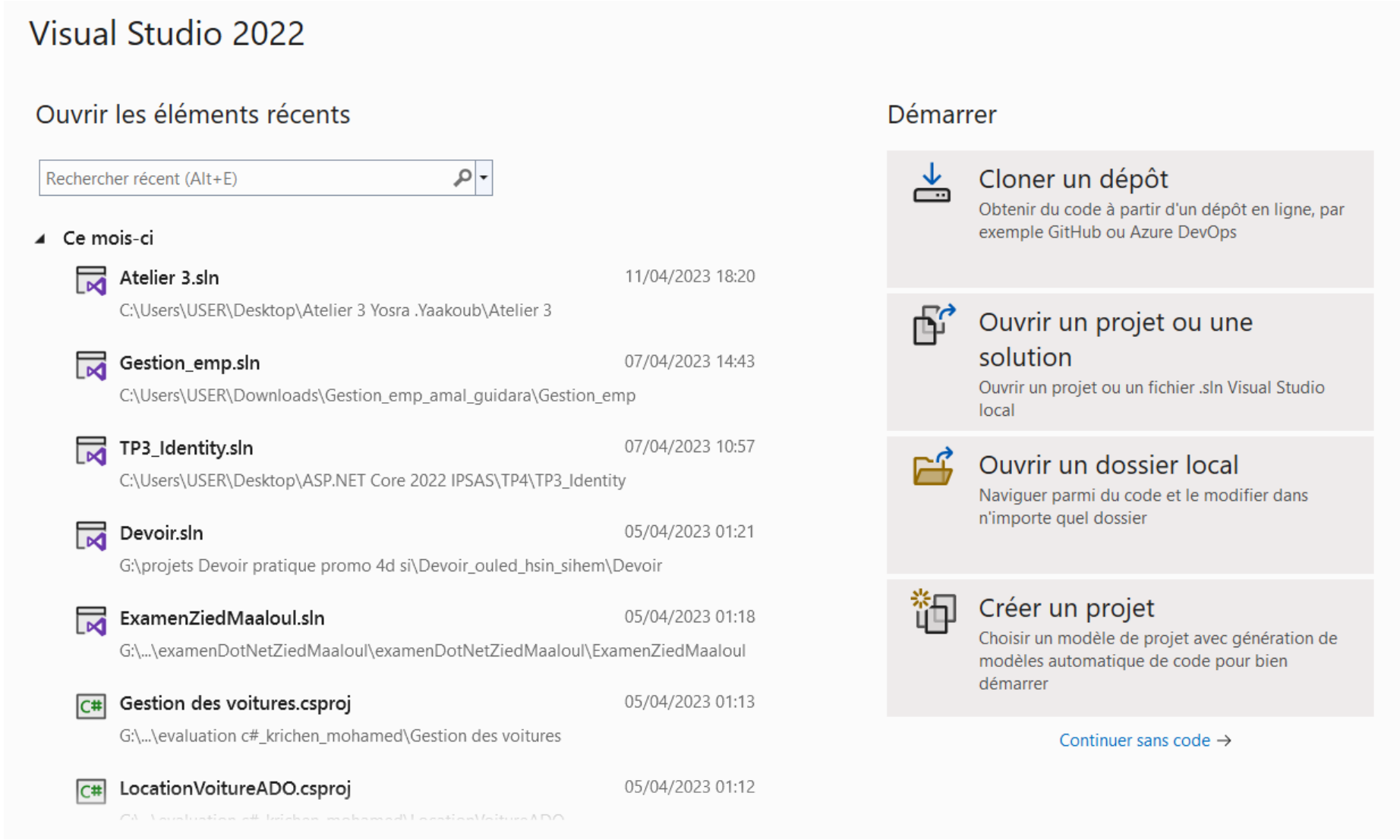
- **Cross plateform :** les applications ASP.NET Core peuvent s'exécuter sur Windows, Linux et Mac.
- **Rapide :** un projet inclut uniquement les packages dont nous avons besoin pour notre application. Cela réduit le pipeline de requêtes et améliore les performances et l'évolutivité.
- **IoC Container :** il comprend le conteneur IoC intégré (Inversion of control container) pour l'injection automatique de dépendances, ce qui le rend plus maintenable et testable.
- **Modulaire** : grâce aux packages Nuget et l'utilisation des composants Middleware

Pourquoi ASP.NET Core?

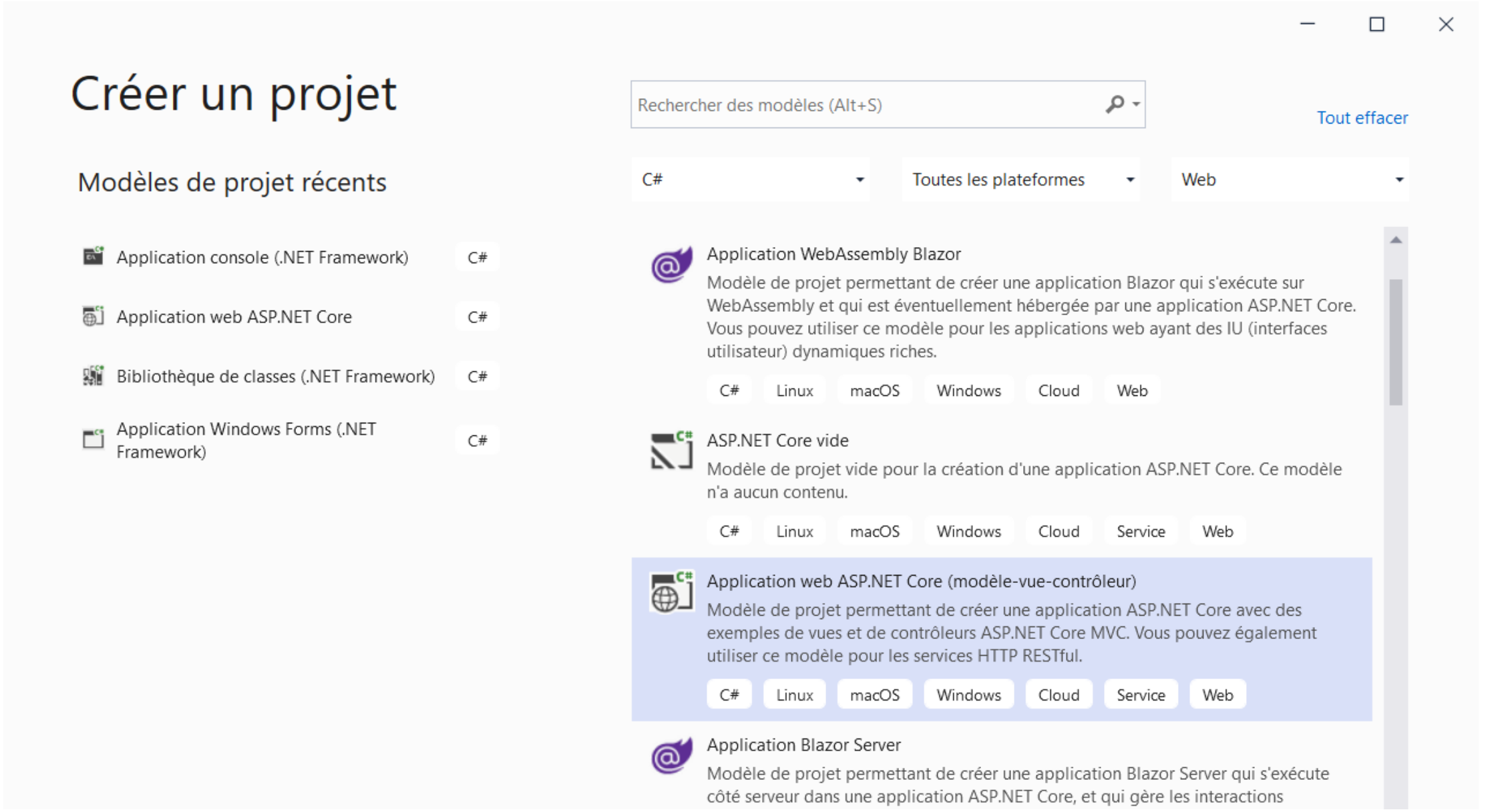
- **Intégration avec les framework front end modernes:** il vous permet d'utiliser et de gérer des Framework modernes tels que Angular, React, Bootstrap, etc.
- **Hébergement:** l'application Web ASP.NET Core peut être hébergée sur plusieurs plates-formes avec plusieurs serveurs Web tel que IIS, Apache, Docker, Nginx.
- **Partage de code:** il vous permet de créer une bibliothèque de classes qui peut être utilisée avec d'autres versions du framework .NET (.Net standard)
- **Un modèle de programmation unifié:** pour les applications MVC et WEB API, dans ASP.NET, c'était deux Framework séparés, dans .NET Core ils sont unifiés.

Créer un Projet ASP.NET Core

- Ouvrez Visual Studio et cliquez sur **Créer un nouveau projet** , comme indiqué ci-dessous.

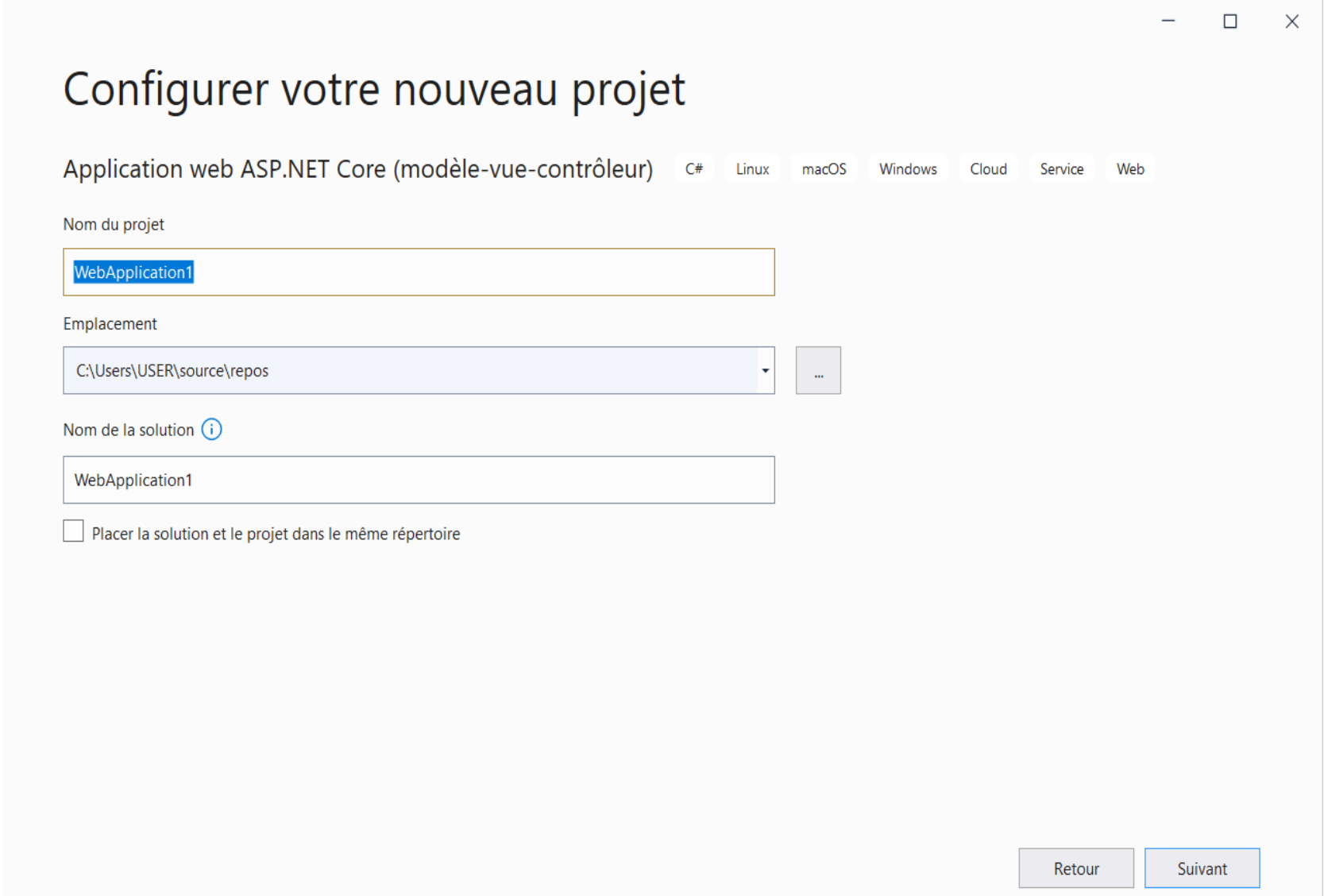


Créer une application ASP.NET Core

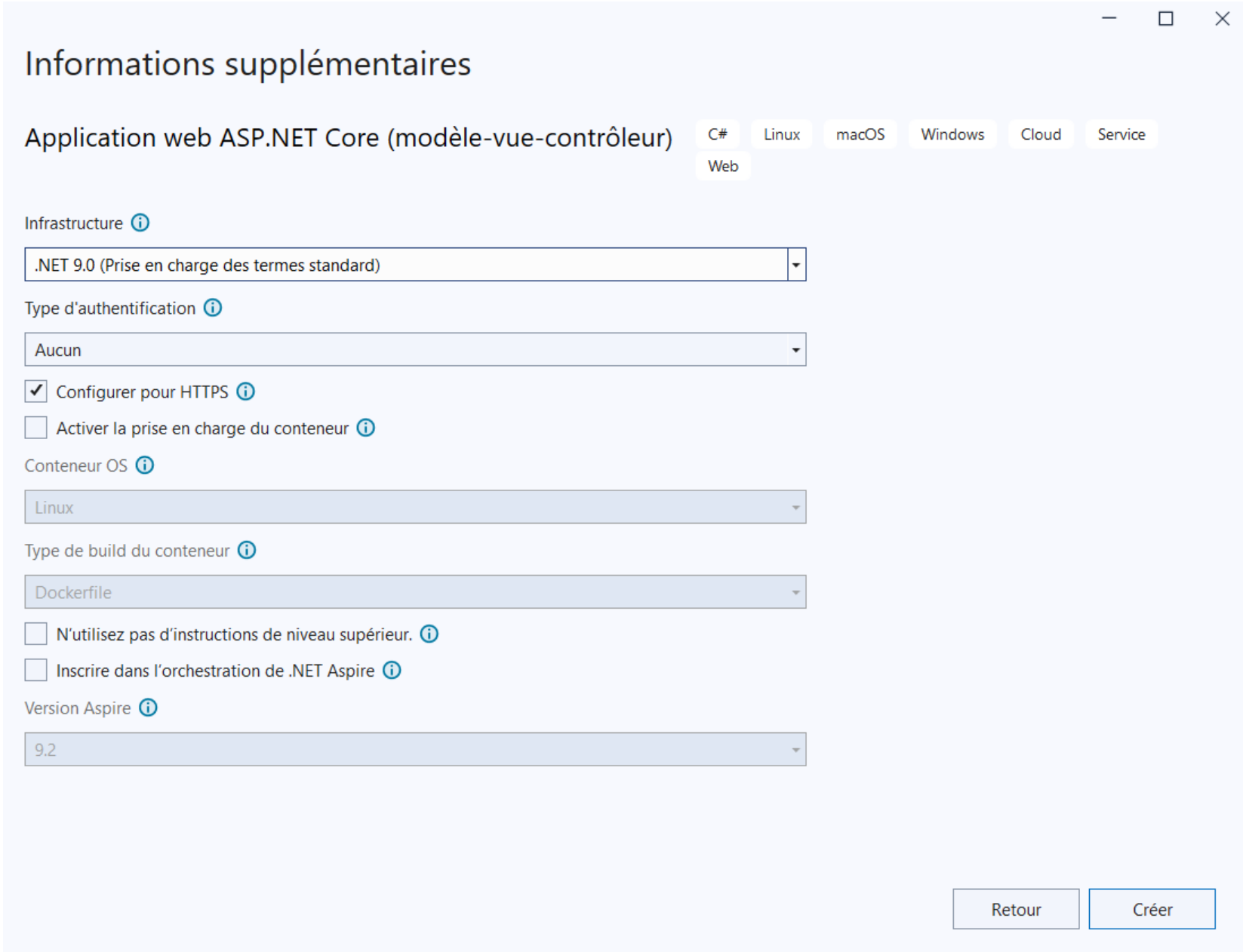


Créer une application ASP.NET Core

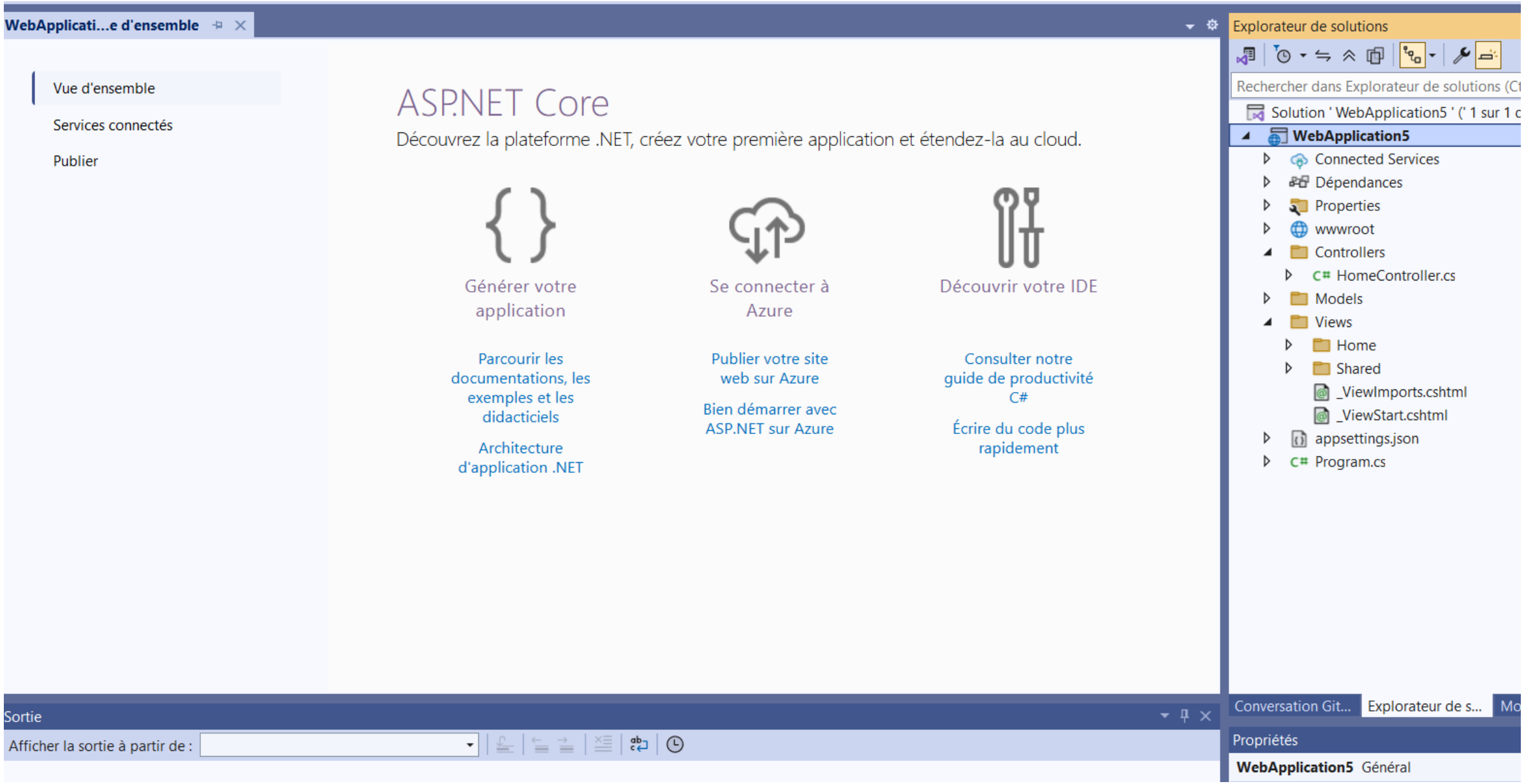
- Indiquez le nom, l'emplacement et le nom de solution appropriés pour l'application ASP.NET Core, puis cliquez sur suivant:



Créer une application ASP.NET Core

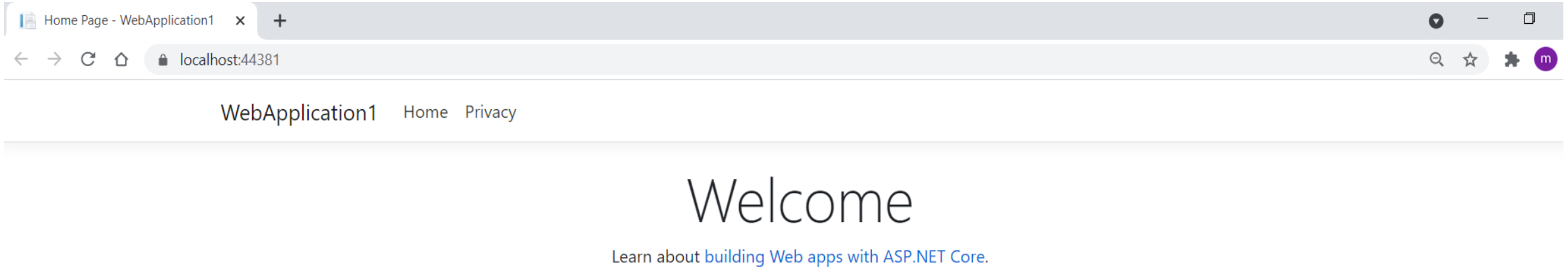


Créer une application ASP.NET Core



Créer une application ASP.NET Core

- Pour exécuter cette application Web, cliquez sur **IIS Express** ou appuyez sur Ctrl + F5. Cela ouvrira le navigateur et affichera le résultat suivant :



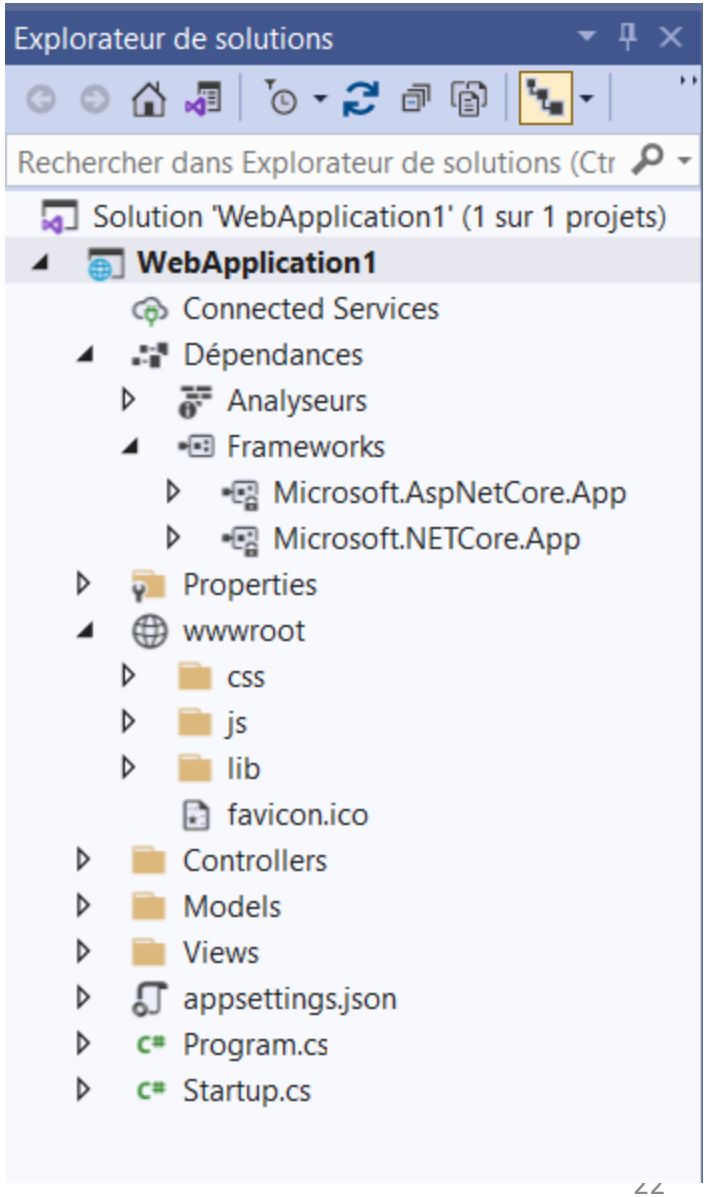
ASP.NET Core - Structure du projet

- L’exemple suivant est une structure de projet par défaut lorsque vous créez une application ASP.NET Core MVC dans Visual Studio.

- L'explorateur de solutions ci-dessus affiche les éléments de la solution du projet.

- Nous pouvons le changer en vue dossier en cliquant sur l' icône **Solution et dossiers** et en sélectionnant l'option Vue dossier.

- Affiche l'explorateur de solutions avec tous les dossiers et fichiers du projet, comme indiqué ci-dessous.



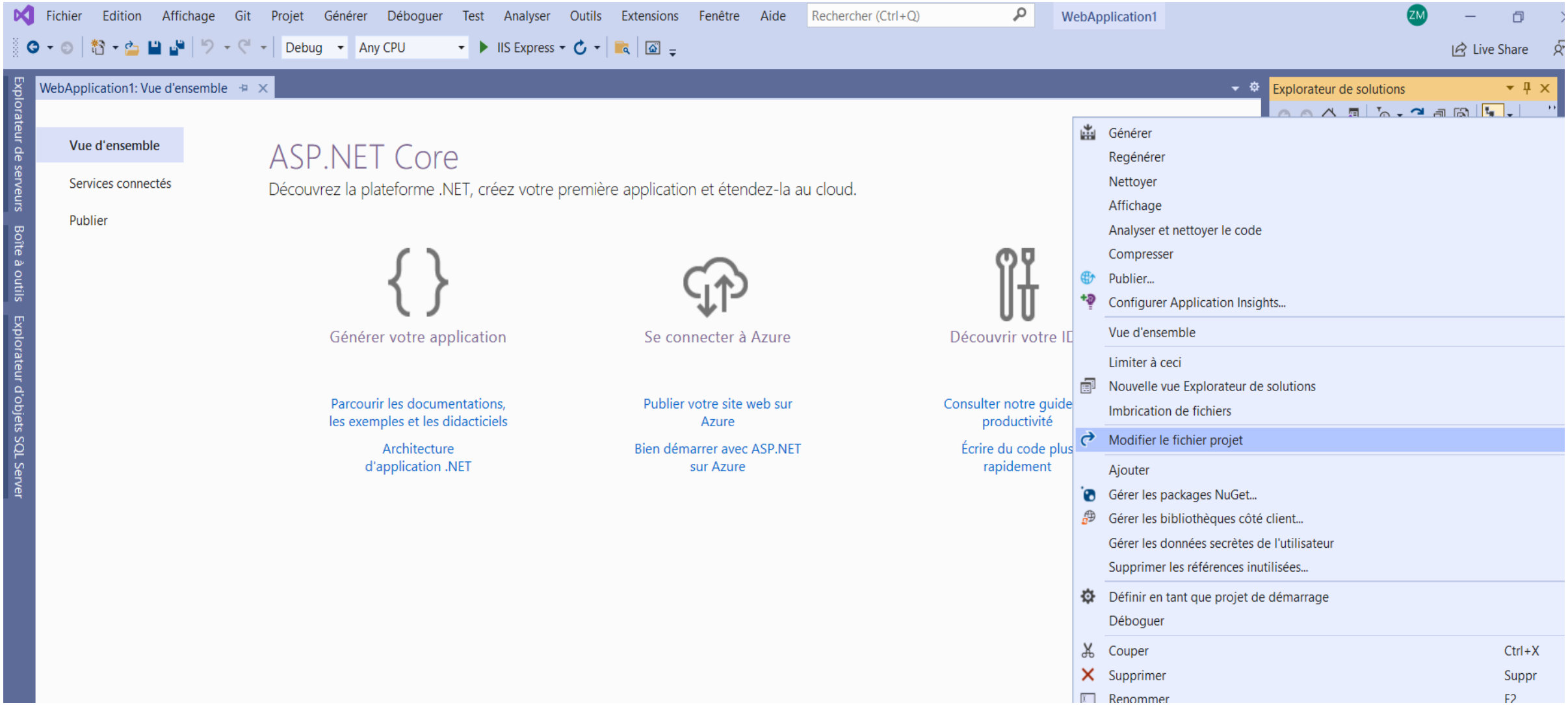
ASP.NET Core - Structure du projet

Le Fichier projet (.csproj)

- Le fichier projet peut être .csproj ou .vbproj selon le language utilisé.

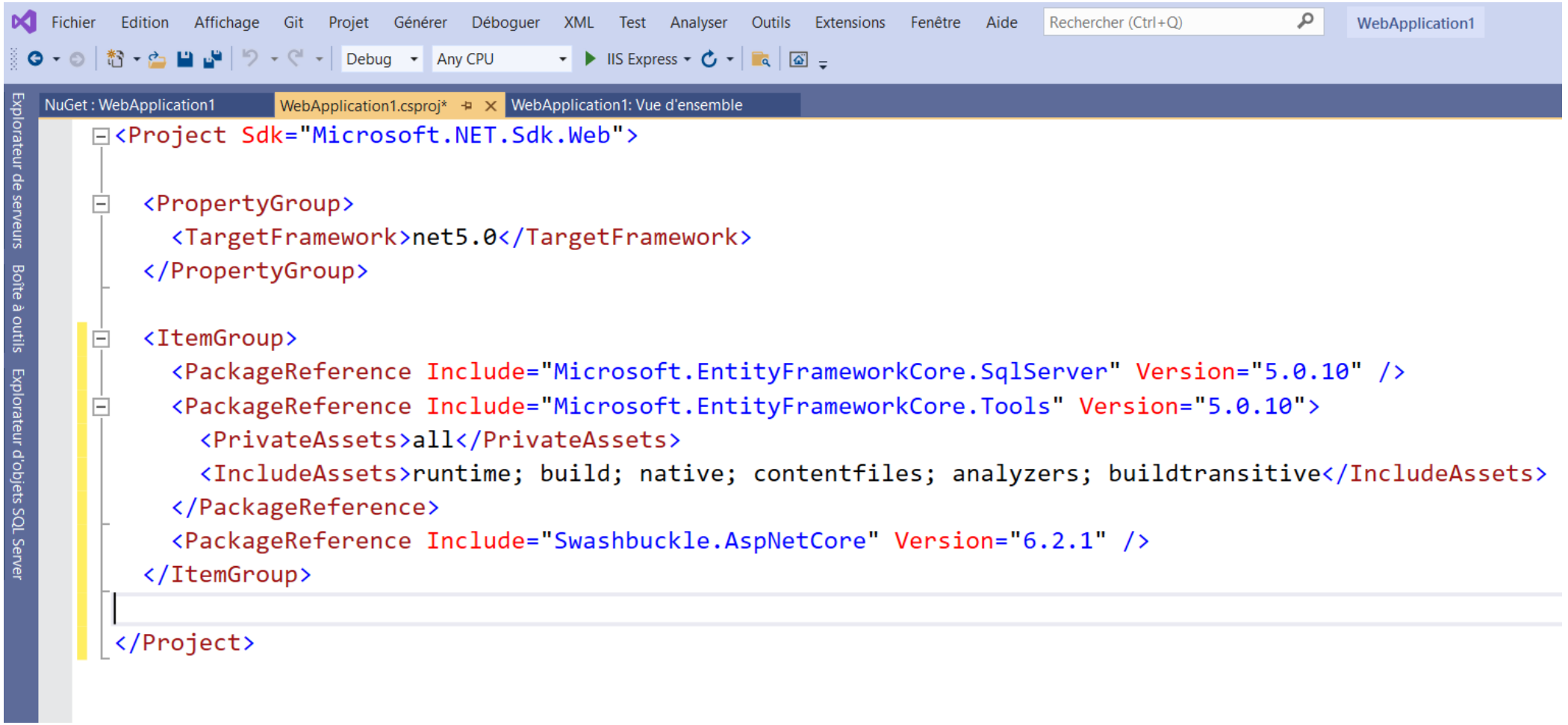
- Nous pouvons modifier les paramètres .csproj en cliquant avec le bouton droit sur le projet et en sélectionnant **Modifier le fichier projet**.

ASP.NET Core - Structure du projet



ASP.NET Core - Structure du projet

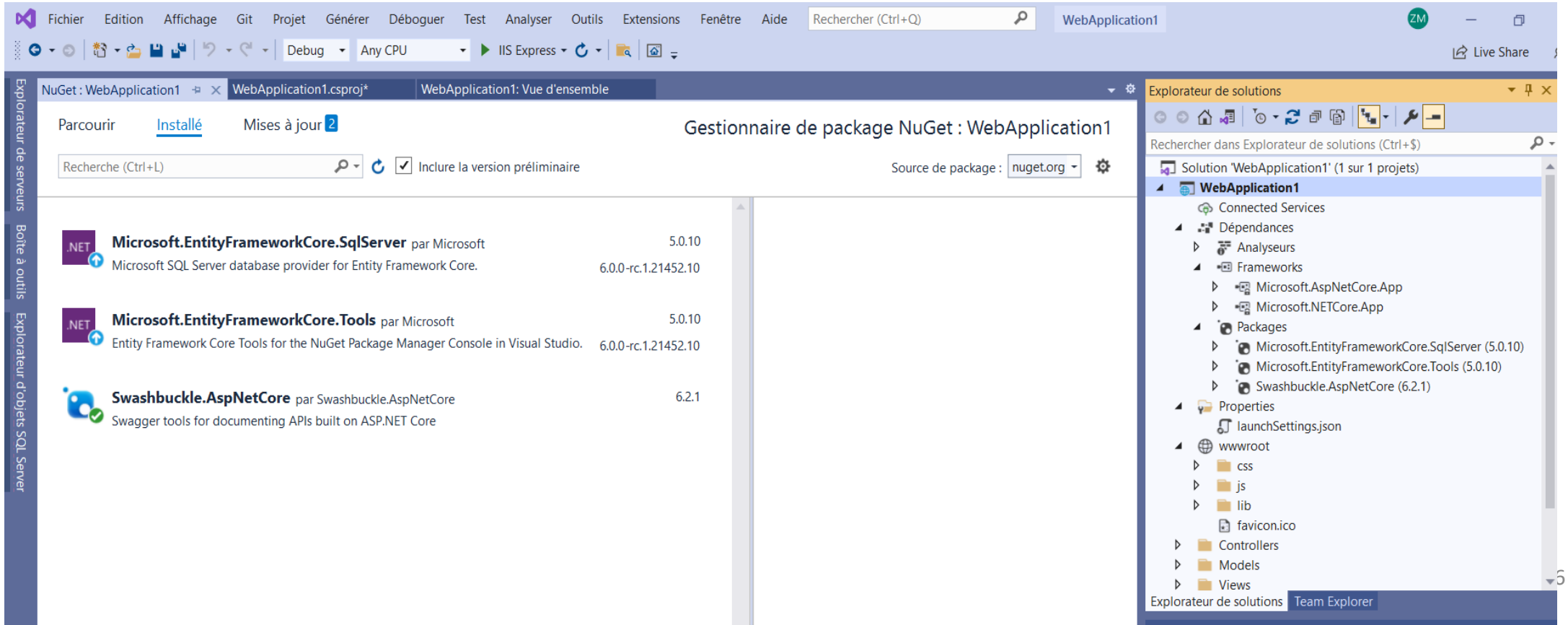
Le fichier csproj comprend des paramètres liés aux framework .NET ciblé, aux dossiers de projet, aux références de package NuGet, etc.



ASP.NET Core - Structure du projet

Les Dépendances

- Les dépendances dans le projet ASP.NET Core contiennent tous les packages NuGet côté serveur installés, comme illustré ci-dessous.
- Cliquez avec le bouton droit sur "Dépendances", puis cliquez sur "Gérer les packages NuGet .." pour afficher les packages NuGet installés.



Les Dépendances

Dans ce projet 2 packages sont ajoutés par défaut avec le projet ,

- le package **Microsoft.AspNetCore.App** est nécessaire pour les applications Web ASP.NET.
- le package **Microsoft.NETCore.App** est un ensemble d'API inclus par défaut dans le modèle d'applications .NET Core.
- Vous pouvez installer d'autres dépendances côté serveur requises en tant que packages NuGet à partir de la fenêtre Gérer les packages NuGet ou à l'aide de la console du gestionnaire de package.

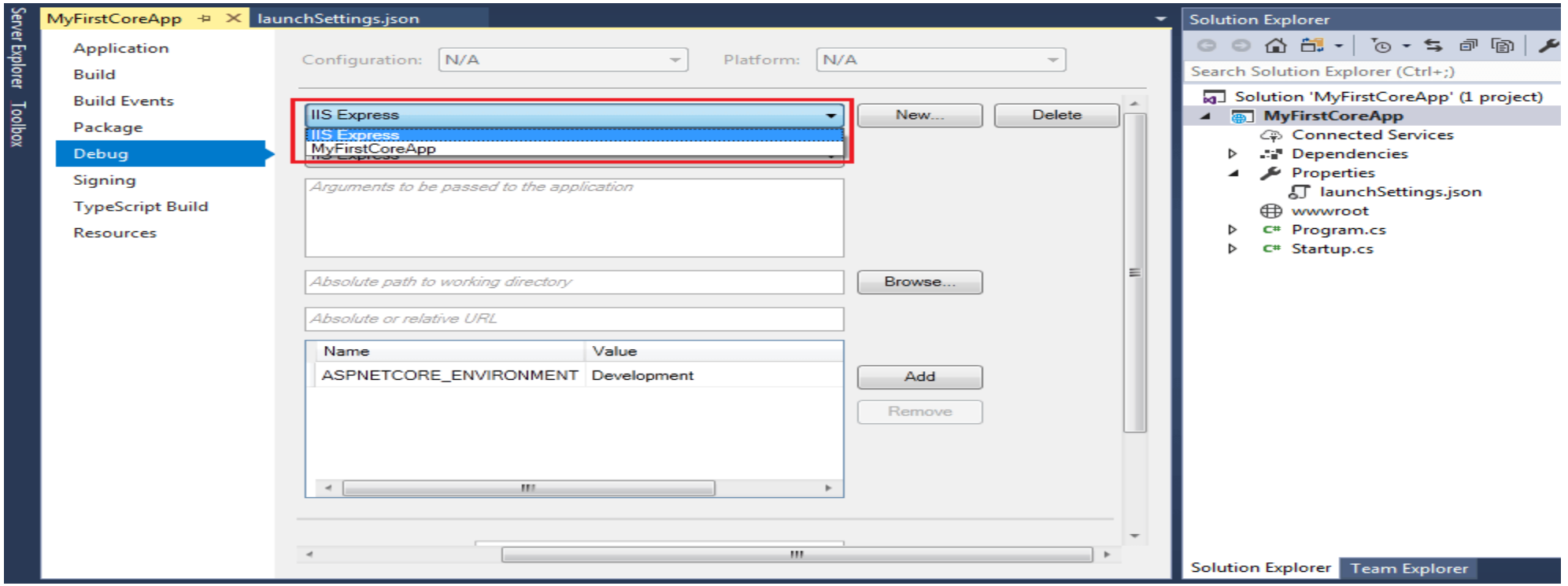
launchSettings.json

- Le nœud Propriétés comprend le fichier launchSettings.json qui inclut les profils pris en considération lors du débogage du projet.

- Deux profils existent par défaut, IIS EXPRESS et un deuxième profil portant le nom de l'application. Ce qui suit est un fichier launchSettings.json par défaut.

Les propriétés d'un projet

- Nous pouvons également modifier les paramètres à partir de l'onglet de débogage des propriétés du projet. Faites un clic droit sur le projet -> sélectionnez Propriétés -> cliquez sur l'onglet Déboguer.
- Dans l'onglet de débogage, sélectionnez un profil que vous souhaitez modifier comme indiqué ci-dessus. Vous pouvez modifier les variables d'environnement, l'URL, etc.



Appsettings.json

- Dans les versions précédentes d'ASP.NET, nous stockons les paramètres de configuration de l'application, comme les chaînes de connexion à la base de données par exemple, dans le fichier **web.config**.

- Dans ASP.NET Core, les paramètres de configuration de l'application peuvent provenir des **différentes sources de configuration** :
  - Fichiers (appsettings.json, appsettings. {Environment} .json, où {Environment} est l'environnement d'hébergement actuel de l'application)
  - Variables d'environnement
  - Arguments de ligne de commande

Appsettings.json

School.cs | SchoolController.cs | ListRoles.cshtml | _Layout.cshtml | Program.cs | AccountController.cs | appsettings.json | AdminController.cs | Student.cs

Schéma : https://json.schemastore.org/appsettings.json

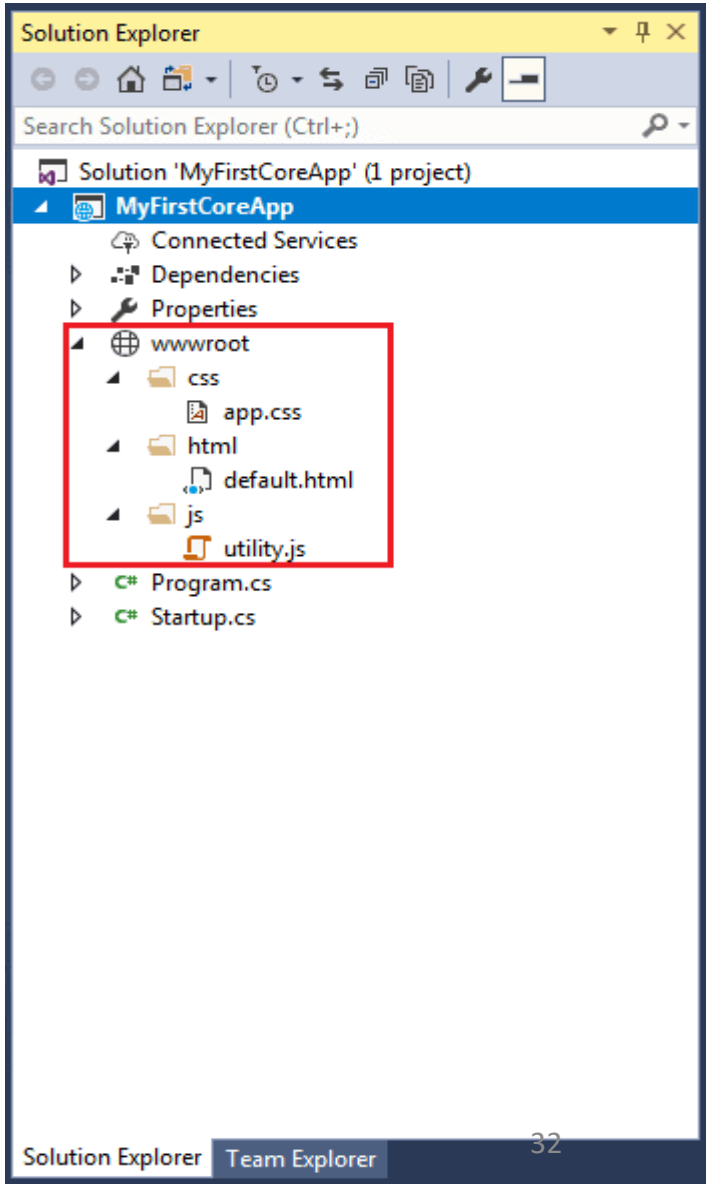
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "StudentDBConnection": "Server=(localdb)\\MSSQLLocalDB;Database=mySchoolDB_identity;Trusted_Connection=
  }
}
```

```
// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContextPool<StudentContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("StudentDBConnection")));
```

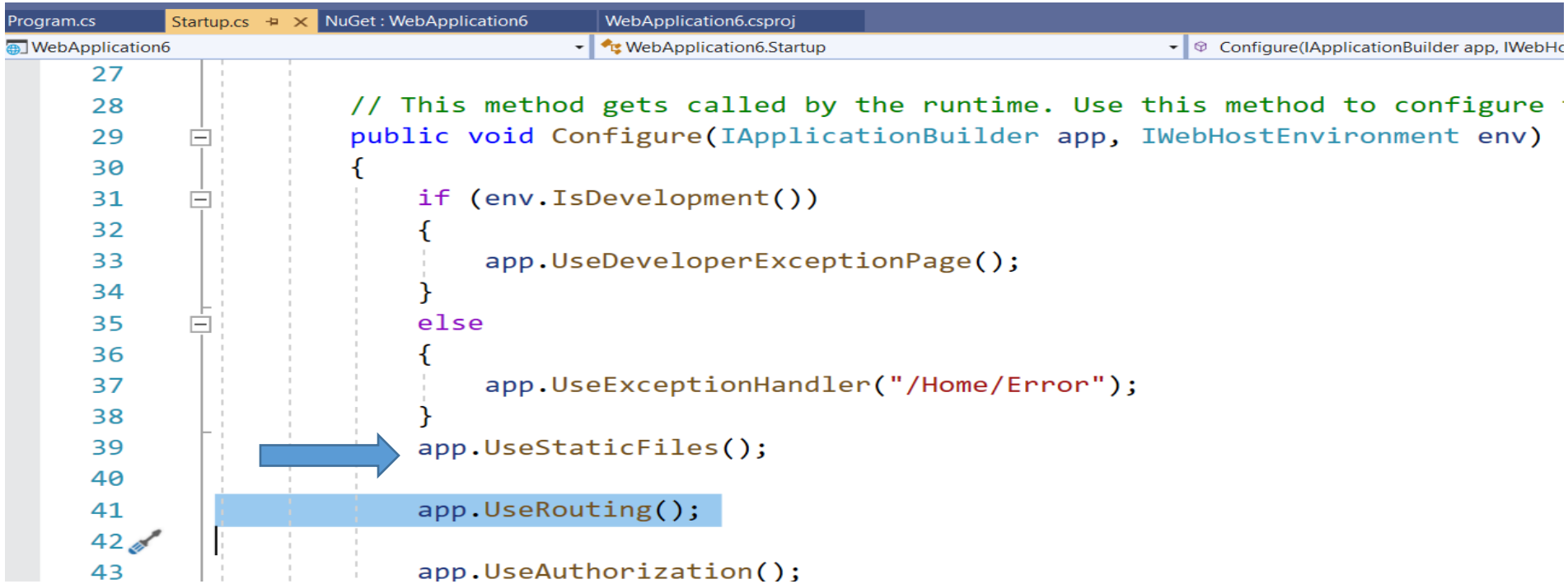
Le Dossier wwwroot

- Par défaut, le dossier **wwwroot** dans le projet ASP.NET Core est traité comme un dossier racine Web (web root folder).
- Les fichiers statiques peuvent être stockés dans n'importe quel dossier sous la racine Web et accessibles avec un chemin relatif vers cette racine.
- En générale, il devrait y avoir des dossiers séparés pour les différents types de fichiers statiques tels que JavaScript, CSS, Images, scripts de bibliothèque, etc. dans le dossier wwwroot comme indiqué dans cette image écran.



Le Dossier wwwroot

- Vous pouvez accéder aux fichiers statiques avec l'URL de base et le nom de fichier. Par exemple, nous pouvons accéder au fichier site.css ci-dessus dans le dossier css par *http://localhost:<port>/css/app.css* .
- N'oubliez pas que vous devez inclure un middleware pour servir les fichiers statiques dans la méthode Configure de Startup.cs. (app.UseStaticFiles())



Le fichier Program.cs

Inscription des services

- Dans ce fichier, vous pouvez enregistrer vos classes dépendantes avec le conteneur IoC intégré (injection des dépendances).
- Après avoir enregistrer la classe dépendante, elle peut être utilisée n'importe où dans l'application.
- ASP.NET Core fait référence à une classe dépendante en tant que service. Ainsi, chaque fois que vous lisez "Service", comprenez-le comme une classe qui sera utilisée dans une autre classe.

Program.cs

WebApplication1

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

```

Le fichier Program.cs

Configuration des MiddleWare

- Endroit où vous pouvez configurer le pipeline de requêtes http pour votre application à l'aide de l'instance IApplicationBuilder fournie par le conteneur IoC intégré.
- ASP.NET Core a introduit les composants middleware pour définir un pipeline de requêtes, qui sera exécuté à chaque appel.
- Vous incluez uniquement les composants middleware requis par votre application et augmentez ainsi les performances de votre application.

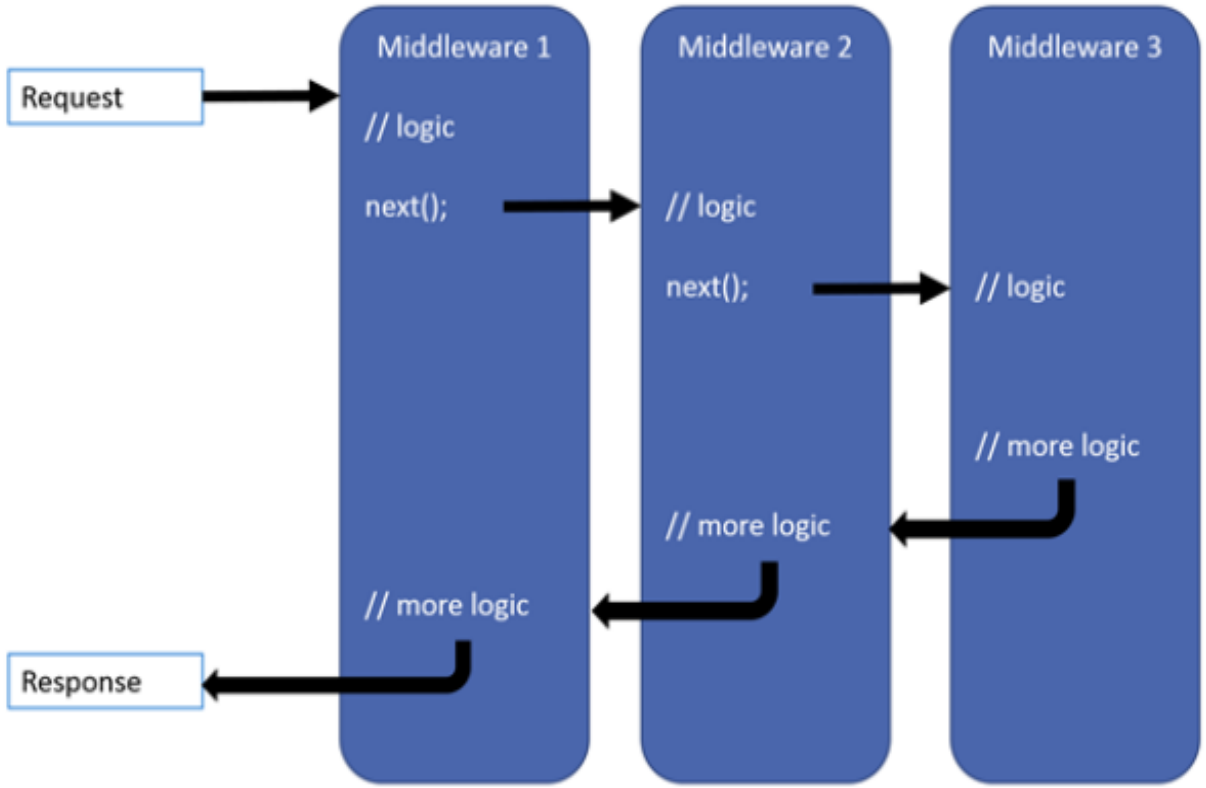
Notion de Middleware dans ASP.NET Core

- Dans ASP.NET Core, le middleware est un composant logiciel capable de gérer une requête ou une réponse HTTP.
- Un composant middleware a un objectif très spécifique. Par exemple authentifier un utilisateur, gérer les erreurs, servir des fichiers statiques tels que des fichiers JavaScript, des fichiers CSS, des images, etc.
- Ces composants middleware sont utilisés pour configurer un pipeline de traitement des demandes dans ASP.NET Core.
- Le pipeline détermine comment une requête est traitée.
- Le pipeline de demandes est configurée dans le cadre du démarrage de l'application dans la méthode Configure() de la classe Startup.

Notion de Middleware dans ASP.NET Core

- Dans ASP.NET Core, **un composant Middleware a accès aux deux: la requête entrante et la réponse sortante.**
- Un composant middleware peut traiter une requête entrante et la transmettre au middleware suivant dans le pipeline pour un traitement ultérieur.
- Un composant middleware peut gérer la demande et décider de ne pas appeler le middleware suivant dans le pipeline.
- Un composant middleware peut également traiter la réponse sortante.
- Les composants middleware sont exécutés dans l'ordre dans lequel ils sont ajoutés au pipeline.
- Les composants middleware sont disponibles sous forme de packages NuGet.

Configuration de la pipeline de traitement des demandes à l'aide des composants Middleware



Information Logged

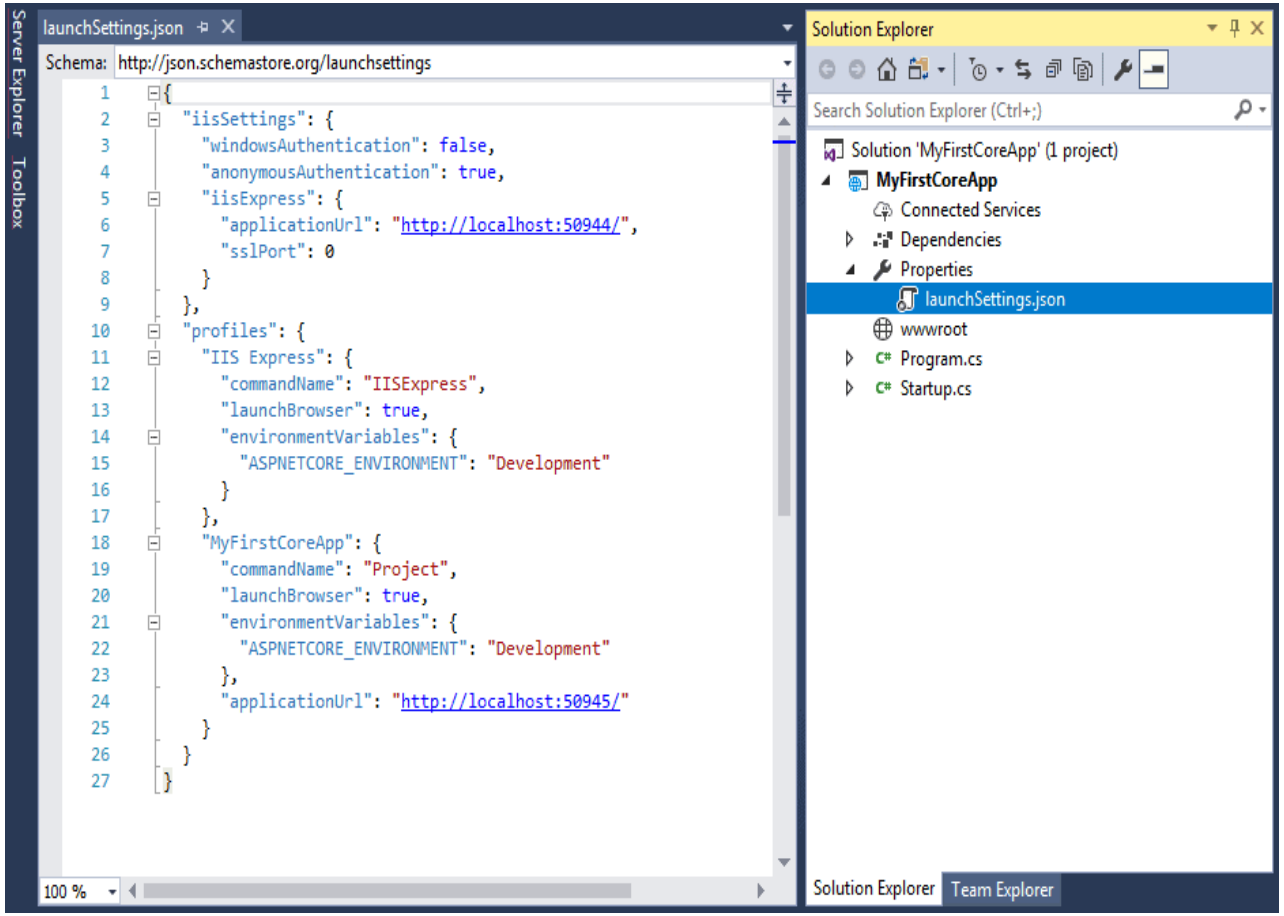
- **MW1: Incoming Request**
- **MW2: Incoming Request**
- **MW3: Request handled and response produced**
- **MW2: Outgoing Response**
- **MW1: Outgoing Response**

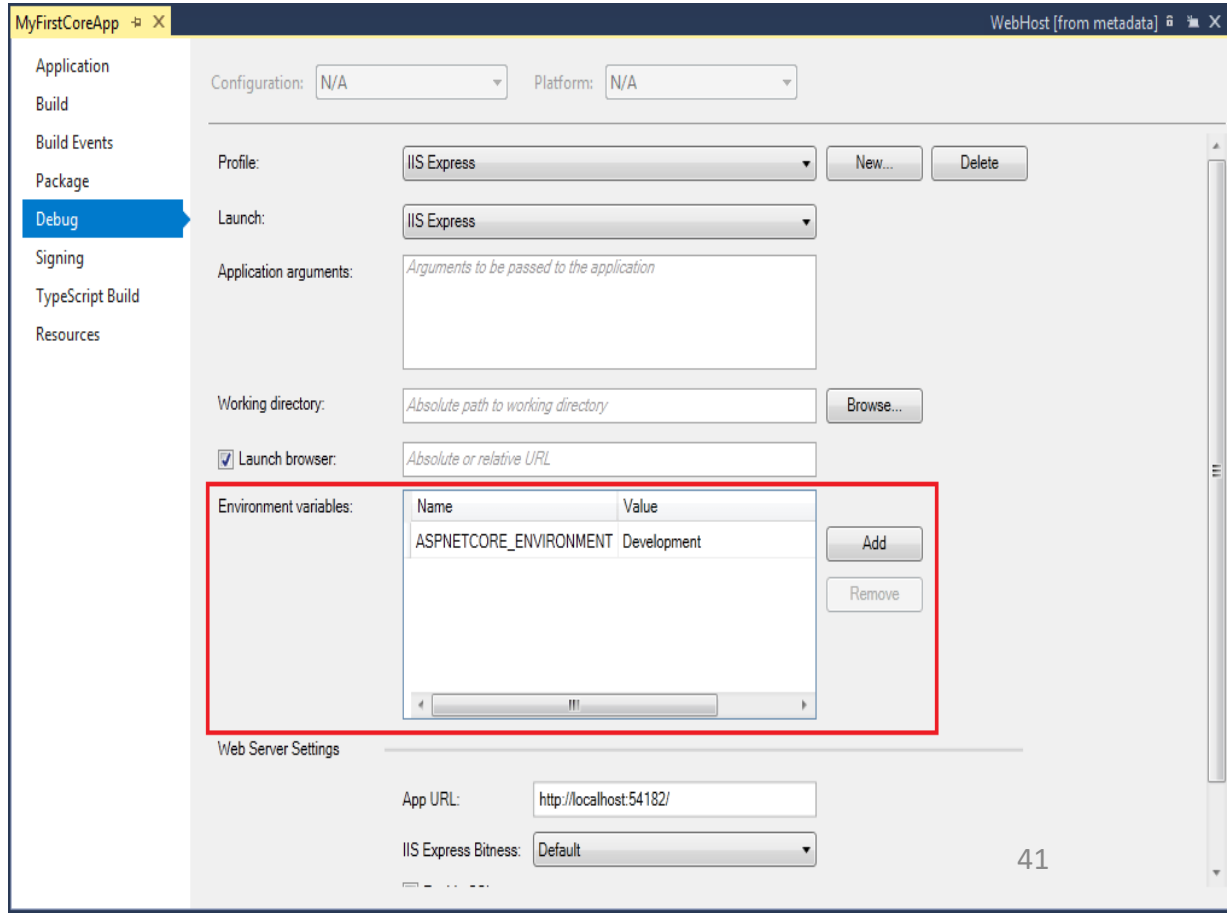
Variable d'environnement

- Dans la plupart des organismes de développement logiciel, on distingue trois phases suivantes : Développement, préparation et production.
- ASP.NET Core utilise une variable d'environnement appelée ASPNETCORE_ENVIRONMENT pour indiquer l'environnement d'exécution.
- On peut définir la valeur de cette variable via la fenêtre propriété du projet ou via le fichier launchSettings.json

Variable d'environnement

- La valeur de cette variable peut être défini par le système d'exploitation.
- Si la variable d'environnement est définie aux deux endroits, c'est-à-dire dans le fichier launchsettings.json et dans le système d'exploitation, la valeur du fichier launchsettings.json remplace la valeur spécifiée au niveau du système d'exploitation.





Variable d'environnement

Méthodes utiles du service IWebHostEnvironment

Utilisez les méthodes suivantes du service **IWebHostEnvironment** pour identifier l'environnement dans lequel notre application s'exécute.

- IsDevelopment ()
- IsStaging ()
- IsProduction ()

Exemple :

```
// Si l'environnement est Development utiliser Developer Exception Page
if (env.IsDevelopment ())
{
    app.UseDeveloperExceptionPage ();
}
/
else if (env.IsStaging () || env.IsProduction ()
{
    app.UseExceptionHandler ( "/ Error" ) ;
}
```