

ASP.NET Core Blazor

MOHAMED AIROUCHE

AUTOMNE 2020

ASP.NET Core Blazor

- Blazor est un Framework permettant de créer des applications web interactives en utilisant C# au lieu de JavaScript. Il devient alors possible pour un développeur C# d'utiliser un seul langage pour écrire la partie Client et la partie Serveur d'une application web.
- Les applications Blazor sont basées sur des *composants*. Un composant Blazor est un élément de l'interface utilisateur, tel qu'une page, une boîte de dialogue ou un formulaire de saisie de données.
- Les composants sont des classes C# en .NET qui :
 - ✓ Définissent la logique de rendu de l'interface utilisateur flexible.
 - ✓ Gèrent les événements de l'utilisateur.
 - ✓ Peuvent être imbriqués et réutilisés.
 - ✓ Peut être partagé et distribué en tant que classe de bibliothèque Razor ou packages NuGet.

<https://nacoumblesoro.developpez.com/tutoriel/blazor-crud-postgresql/>

<https://nacoumblesoro.developpez.com/tutoriel/blazor-crud-postgresql/>

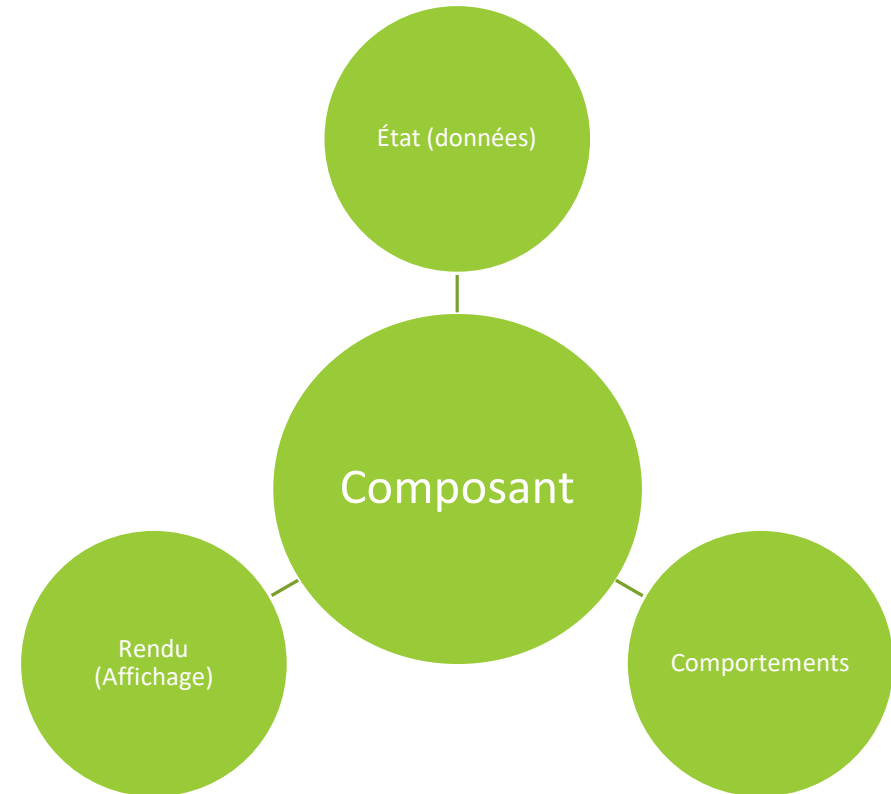
ASP.NET Core Blazor

Une composante Blazor

Blazor prend en charge l'encapsulation de l'interface utilisateur par le biais de *composants*.

Un composant :

- ✓ Est un bloc d'interface utilisateur autonome.
- ✓ Conserve son propre État et sa logique de rendu.
- ✓ Peut définir des gestionnaires d'événements d'interface utilisateur, les lier aux données d'entrée et gérer leur propre cycle de vie.
- ✓ Est généralement défini dans un fichier *.Razor* à l'aide de syntaxe Razor.



ASP.NET Core Blazor

- La classe Component est généralement écrite sous forme d'une page Razor avec une extension de fichier .razor. Razor est une syntaxe pour combiner le balisage HTML avec du code C# conçu pour la productivité des développeurs. Razor vous permet de basculer entre le balisage HTML et C# dans le même fichier avec la prise en charge de l'IntelliSense en programmation avec Visual Studio. Les pages Razor et MVC utilisent également Razor. Contrairement aux Razor pages et MVC, qui sont générés autour d'un modèle de demande/réponse, les composants sont utilisés spécifiquement pour la logique et la composition de l'interface utilisateur côté client.

Exemple :

```
<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-2.1&tabs=visual-studio>

ASP.NET Core Blazor

Les différents modèles de projet Blazor

- Blazor vient avec deux modèles différents : Blazor Server et Blazor Web Assembly. Les deux versions sont aujourd'hui disponibles pour un environnement de production.

Blazor Serveur

L'application Blazor est exécutée sur le serveur à partir d'une application ASP.NET Core. Les échanges entre l'interface utilisateur et le serveur sont effectués à l'aide de Signal R. Signal R est une technologie facilitant les échanges temps réel entre le client et le serveur. Le client peut réceptionner des informations du serveur sans avoir préalablement émis une demande.

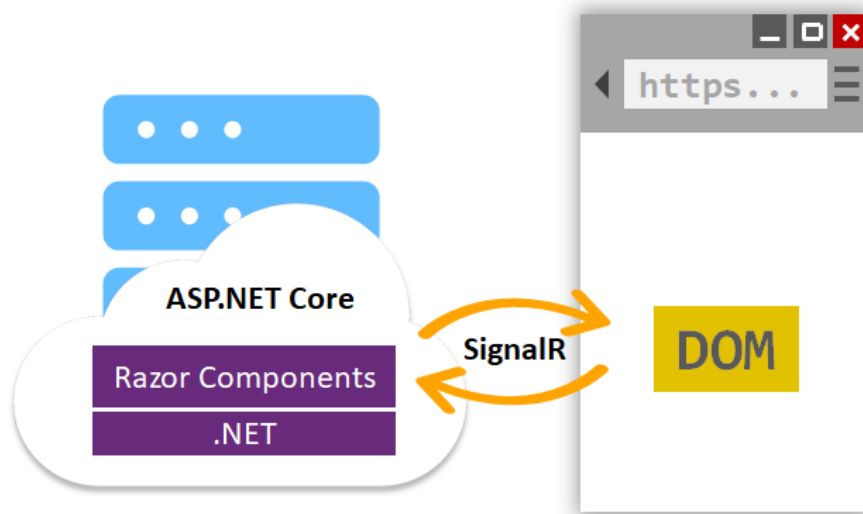
Blazor Web Assembly

L'application Blazor, ses dépendances et le Runtime .NET sont téléchargés dans le navigateur. Ainsi l'application s'exécute directement à partir du thread d'interface utilisateur du navigateur web.

ASP.NET Core Blazor

Blazor Serveur

Blazor Server prend en charge l'hébergement des composants Razor sur le serveur dans une application ASP.NET Core. Les mises à jour de l'interface utilisateur sont gérées via une connexion Signal R.



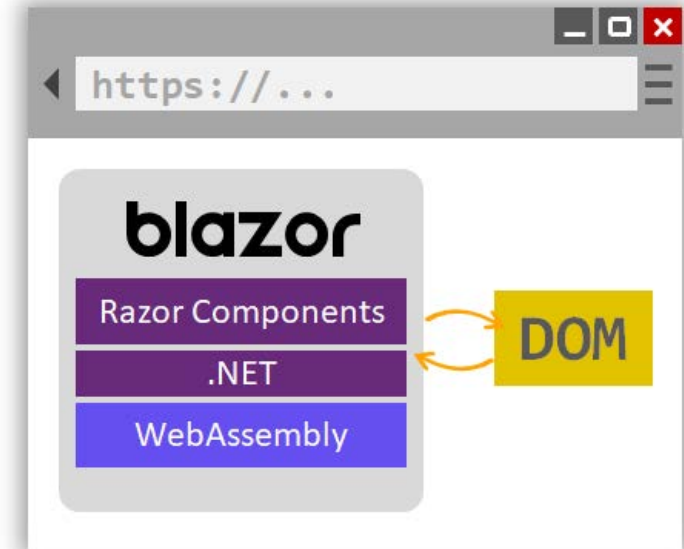
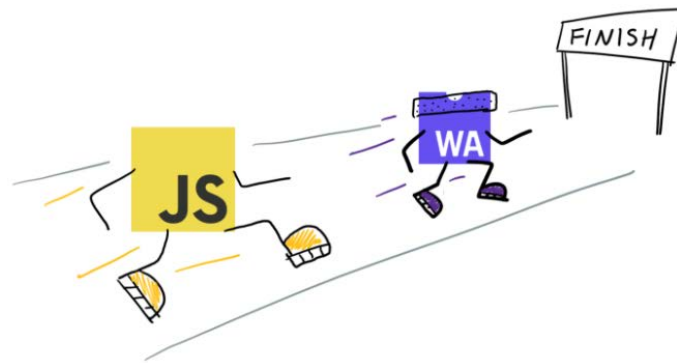
<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-2.1&tabs=visual-studio>

ASP.NET Core Blazor

Blazor Web Assembly

L'exécution de code .NET dans des navigateurs Web est rendue possible par WebAssembly (abrégé wasm).

WebAssembly est un nouveau standard Web qui vous permet d'exécuter du code natif dans le navigateur. WebAssembly est un format bytecode compact optimisé pour un téléchargement rapide et une vitesse d'exécution maximale. WebAssembly est un standard web ouvert pris en charge dans les navigateurs web sans plug-in.



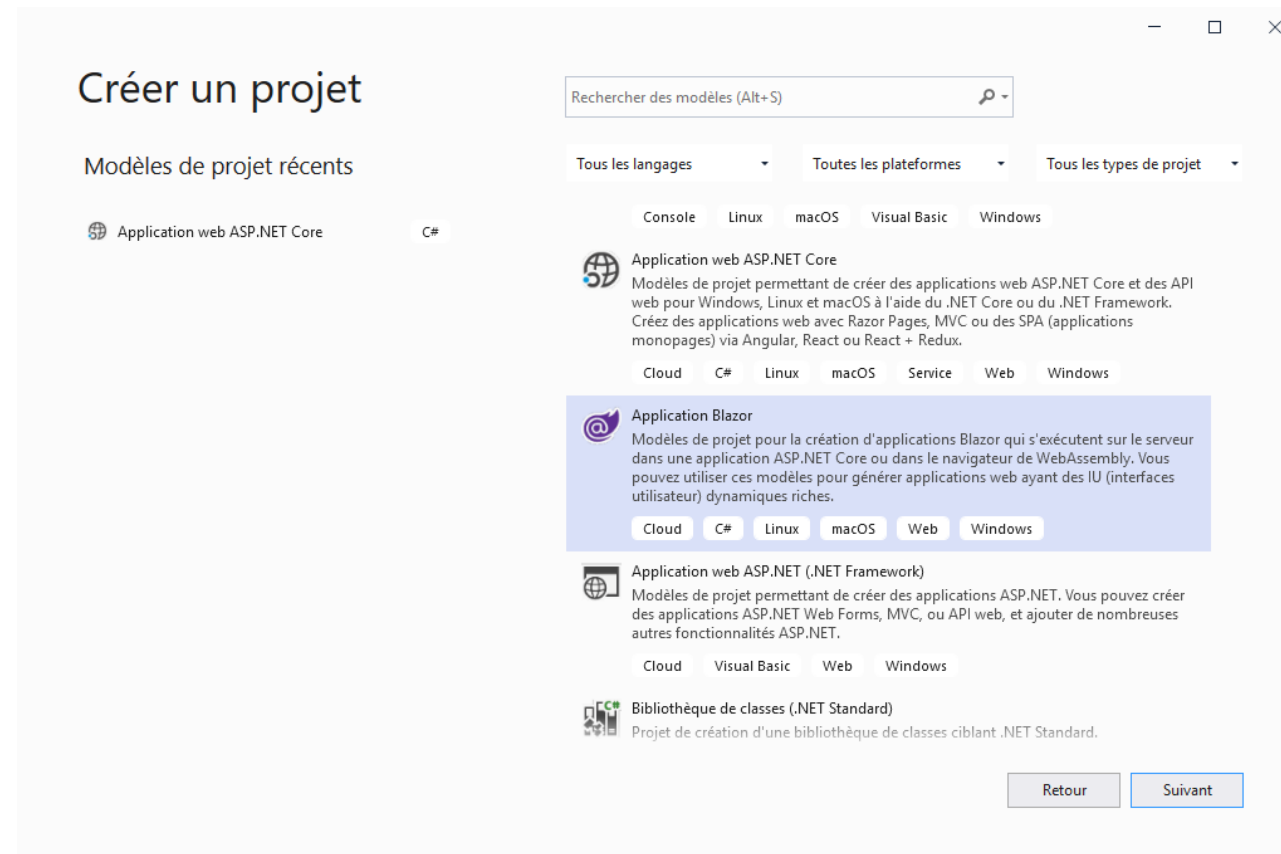
<https://medium.com/samsung-internet-dev/performance-testing-web-assembly-vs-javascript-e07506fd5875>

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-2.1&tabs=visual-studio>

Exemple 1

Création d'un projet Blazor WebAssembly :

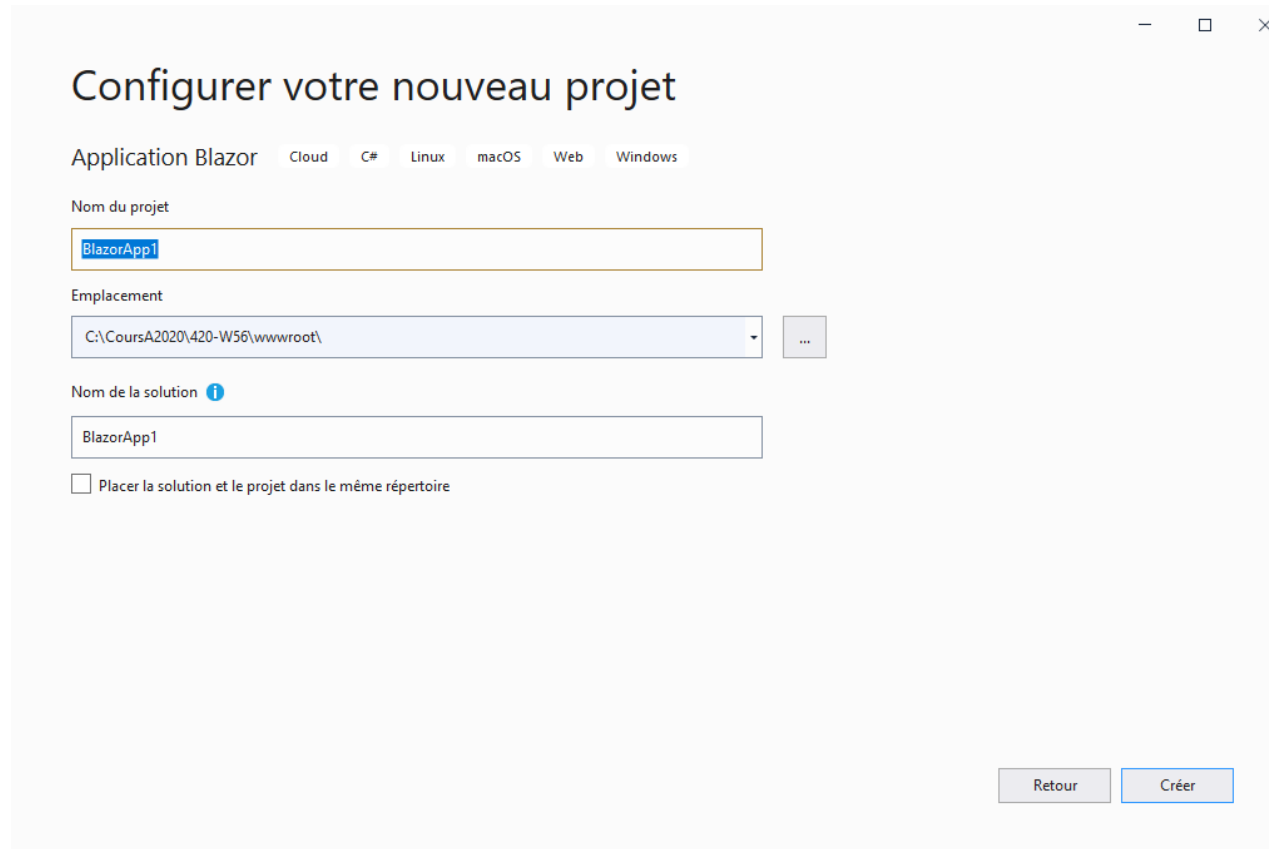
Ouvrez Visual Studio 2019. Cliquez sur Créer un nouveau projet. Sélectionnez Application Blazor. Cliquez sur Suivant.



Exemple 1

Création d'un projet Blazor WebAssembly :

Donnez le nom **BlazorApp1** pour le projet et cliquez sur créer.



Configurer votre nouveau projet

Application Blazor Cloud C# Linux macOS Web Windows

Nom du projet

BlazorApp1

Emplacement

C:\CoursA2020\420-W56\wwwroot\

Nom de la solution ⓘ

BlazorApp1

☐ Placer la solution et le projet dans le même répertoire


Retour Créer

Exemple 1


Création d'un projet Blazor WebAssembly :

Sélectionnez le modèle Blazor WebAssembly App et cliquez sur créer.

Créer une application Blazor

**Application serveur Blazor**

Modèle de projet permettant de créer une application serveur Blazor qui s'exécute côté serveur dans une application ASP.NET Core, et qui gère les interactions utilisateur via une connexion SignalR. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.

**Blazor WebAssembly App**

Modèle de projet permettant de créer une application Blazor qui s'exécute sur WebAssembly. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.

[Obtenir des modèles de projet supplémentaires](#)

Authentification

Aucune authentification

[Changer](#)

Avancé

☒ Configurer pour HTTPS

☐ Activer le support de Docker
(Nécessite Docker Desktop)

Linux

☐ ASP.NET Core hosted

☐ Progressive Web Application

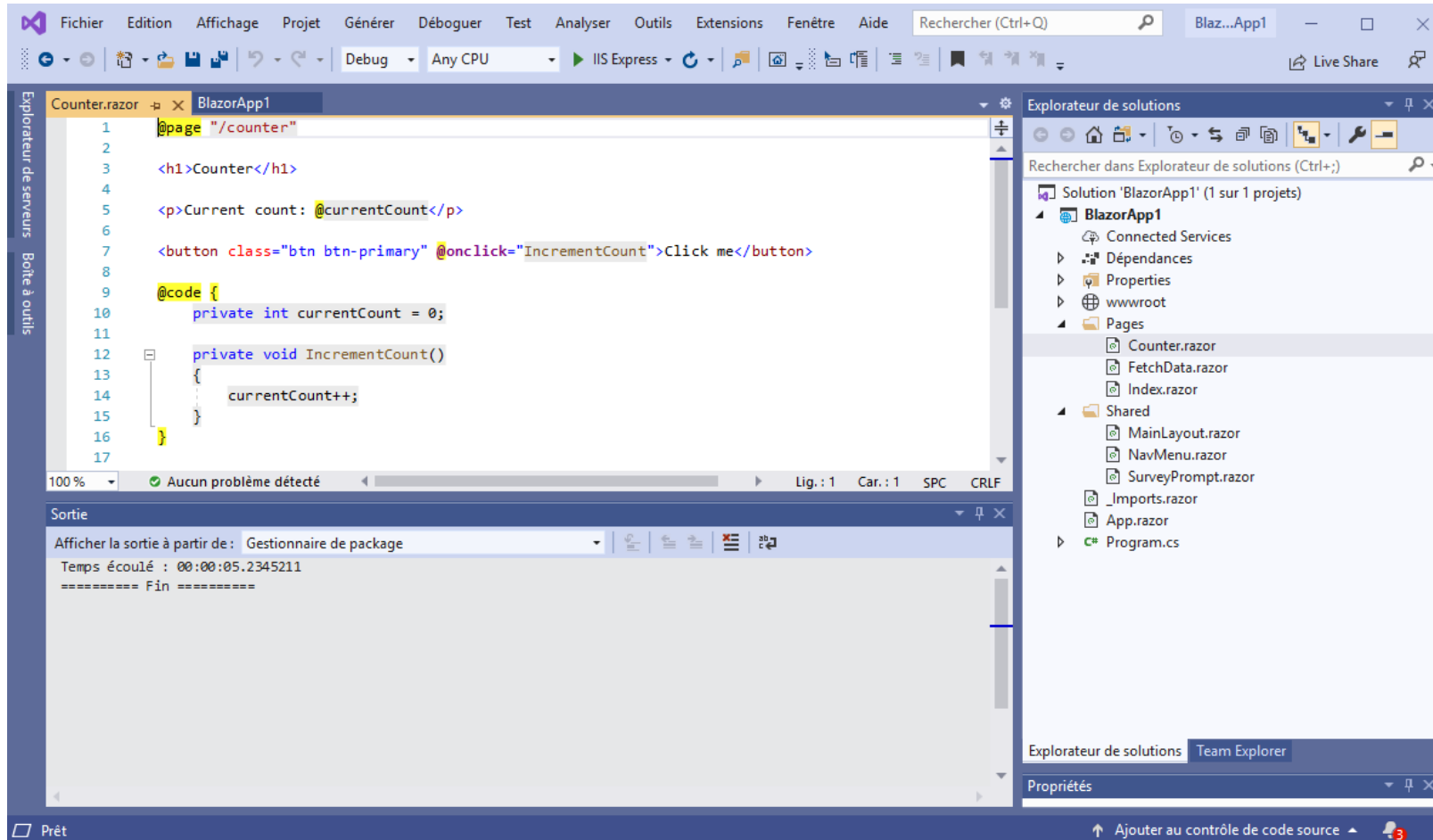
Auteur : Microsoft
Source : Modèles 3.1.6

Retour

Créer

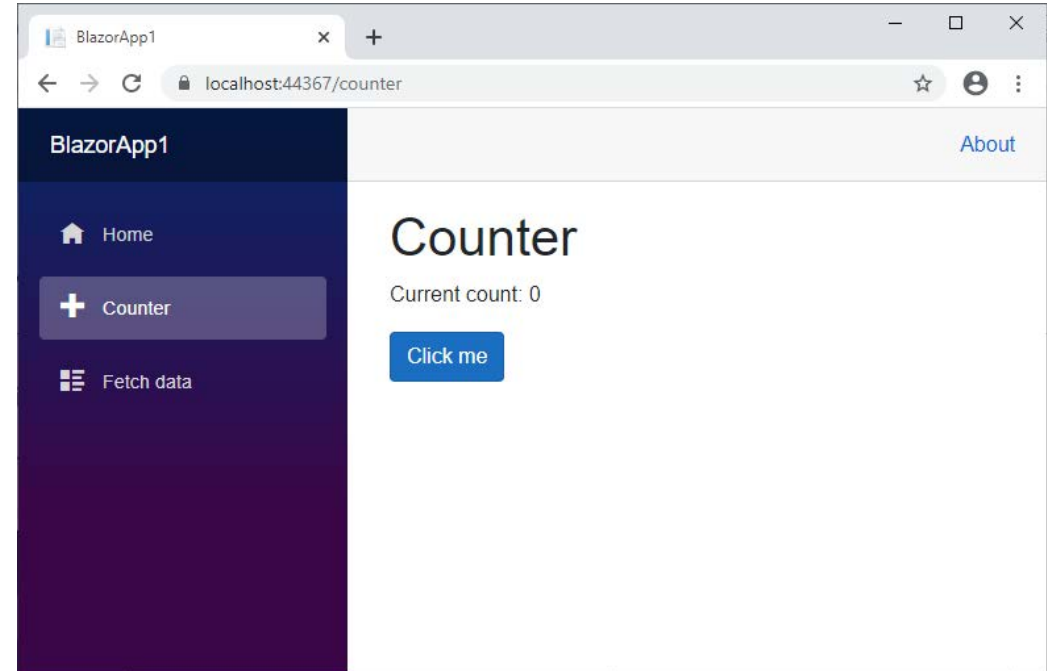
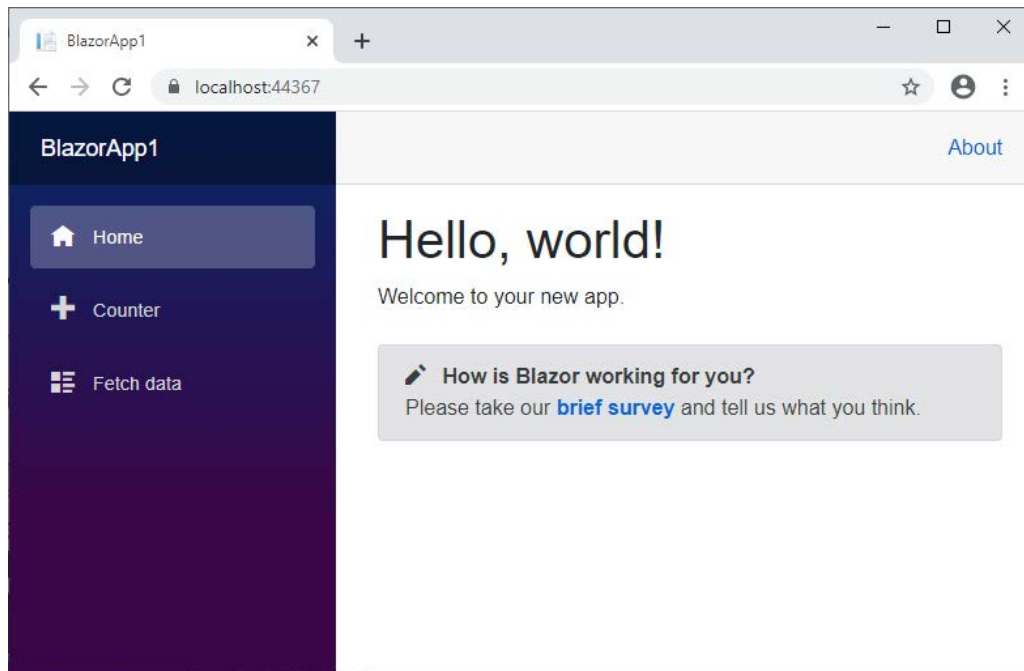
Exemple 1

Voilà! Le projet Blazor WebAssembly est maintenant créé.



Exemple 1

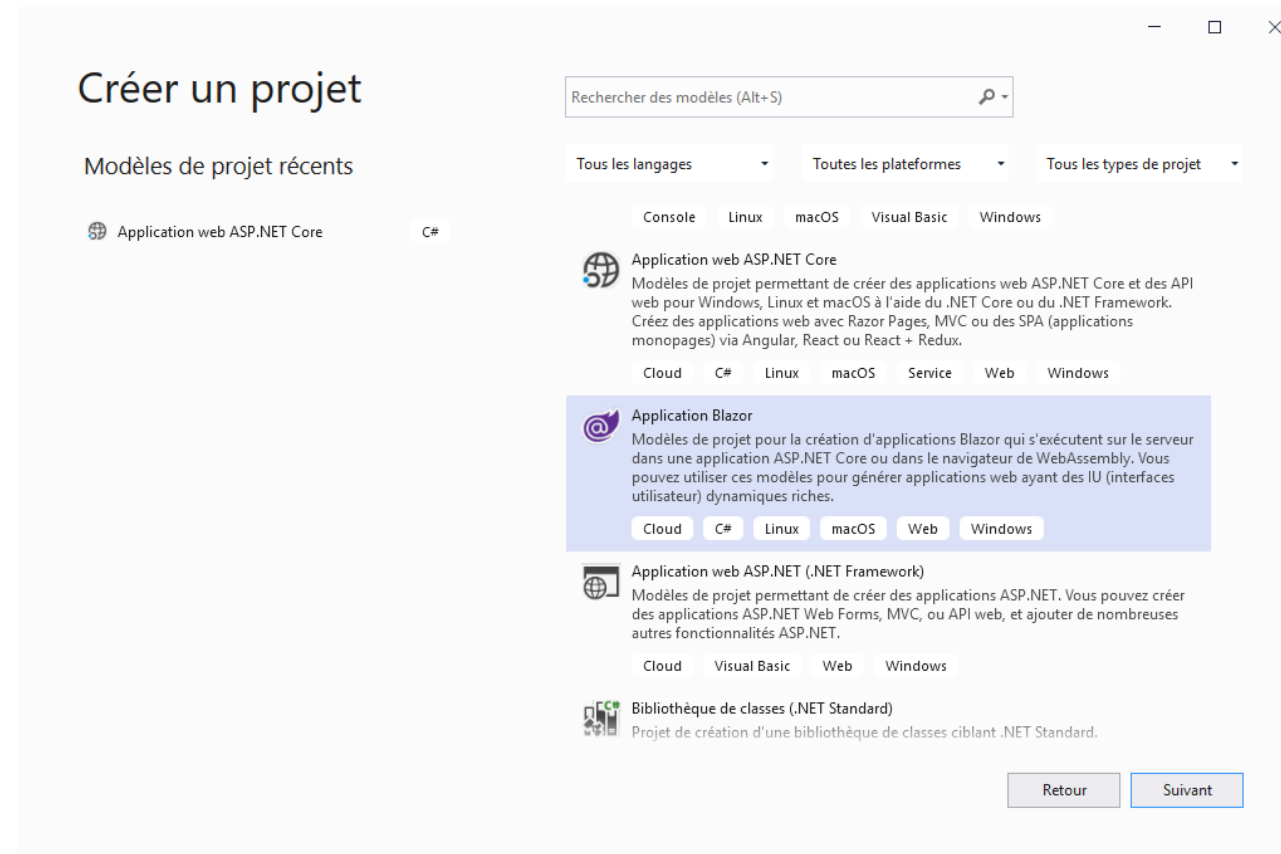
L'exécution de projet Blazor WebAssembly donne les pages suivantes.



Exemple 2

Création d'un projet Blazor Serveur :

Ouvrez Visual Studio 2019. Cliquez sur Créer un nouveau projet. Sélectionnez Application Blazor. Cliquez sur Suivant.



Exemple 2

Création d'un projet Blazor Serveur :

Donnez le nom **BlazorAppServer** au projet et cliquez sur créer.

Configurer votre nouveau projet

Application Blazor Cloud C# Linux macOS Web Windows

Nom du projet

BlazorAppServer

Emplacement

C:\CoursA2020\420-W56\wwwroot

Nom de la solution ⓘ

BlazorAppServer

☐ Placer la solution et le projet dans le même répertoire

Retour Créer


Exemple 2

Création d'un projet Blazor Serveur :


Sélectionnez le modèle Application Serveur Blazor et cliquez sur créer.

×

Créer une application Blazor

**Application serveur Blazor**

Modèle de projet permettant de créer une application serveur Blazor qui s'exécute côté serveur dans une application ASP.NET Core, et qui gère les interactions utilisateur via une connexion SignalR. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.

**Blazor WebAssembly App**

Modèle de projet permettant de créer une application Blazor qui s'exécute sur WebAssembly. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.

Authentification

Aucune authentification
[Changer](#)

Avancé

☒ Configurer pour HTTPS
☐ Activer le support de Docker
(Nécessite [Docker Desktop](#))

Linux

Auteur : Microsoft
Source : Modèles 3.1.6

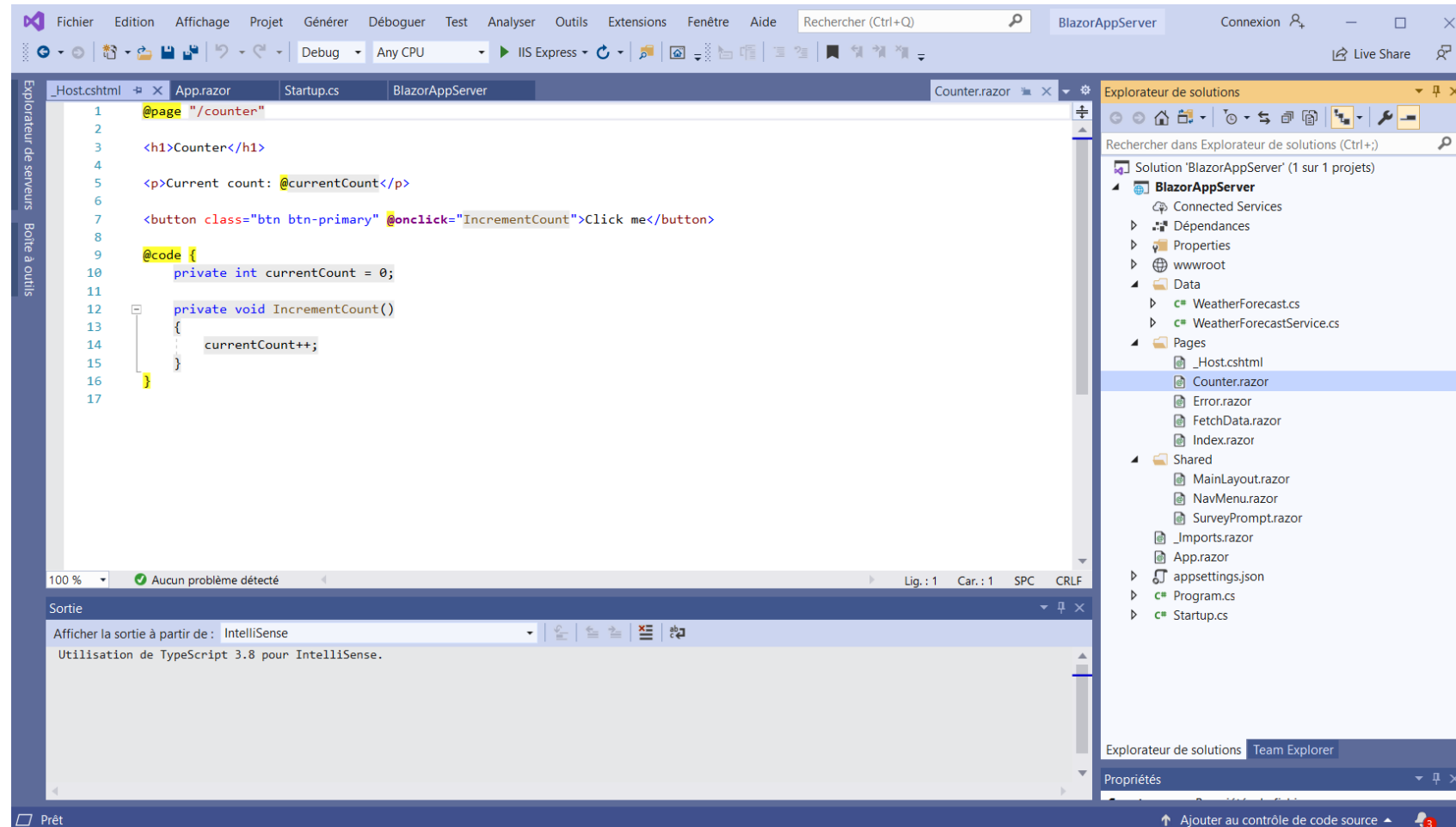
Retour

Créer

[Obtenir des modèles de projet supplémentaires](#)

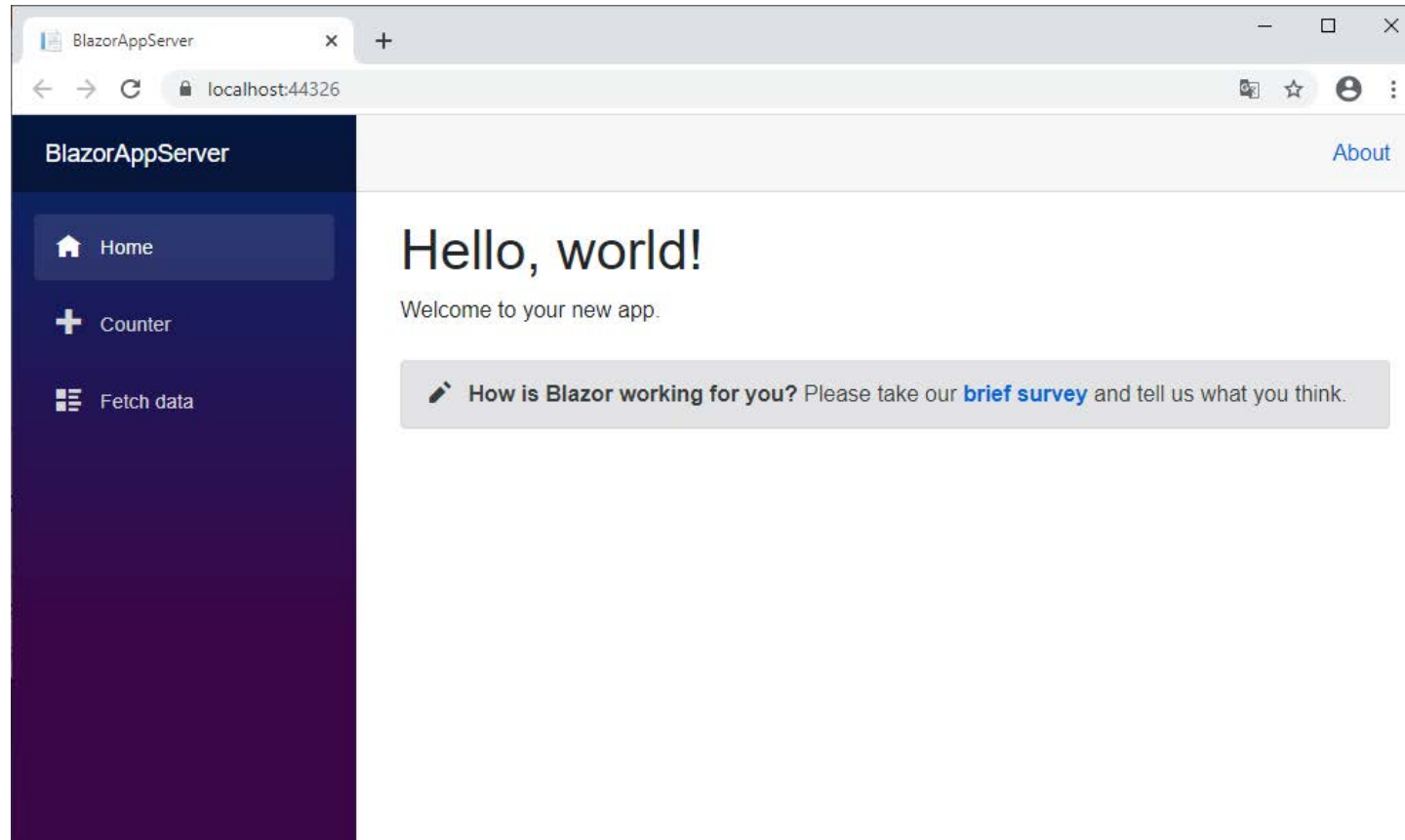
Exemple 2

Voilà! Le projet Blazor Serveur est maintenant créé.



Exemple 2

L'exécution de projet projet Blazor Server donne la page suivante.



ASP.NET Core Blazor

Structure de projet pour les applications Blazor serveur

Quels fichiers ont été créés?

Plusieurs fichiers ont été créés dans le répertoire **BlazorAppServer**, pour vous donner une application Blazor simple prête à fonctionner.

- **Program.cs** est le point d'entrée de l'application qui démarre le serveur.
- **Startup.cs** est l'endroit où vous configurez les services d'application et le middleware.
- **App.razor** est le composant racine de l'application.
- Le **BlazorApp/Pages** répertoire contient des exemples de pages Web pour l'application.
- **BlazorApp.csproj** définit le projet d'application et ses dépendances.

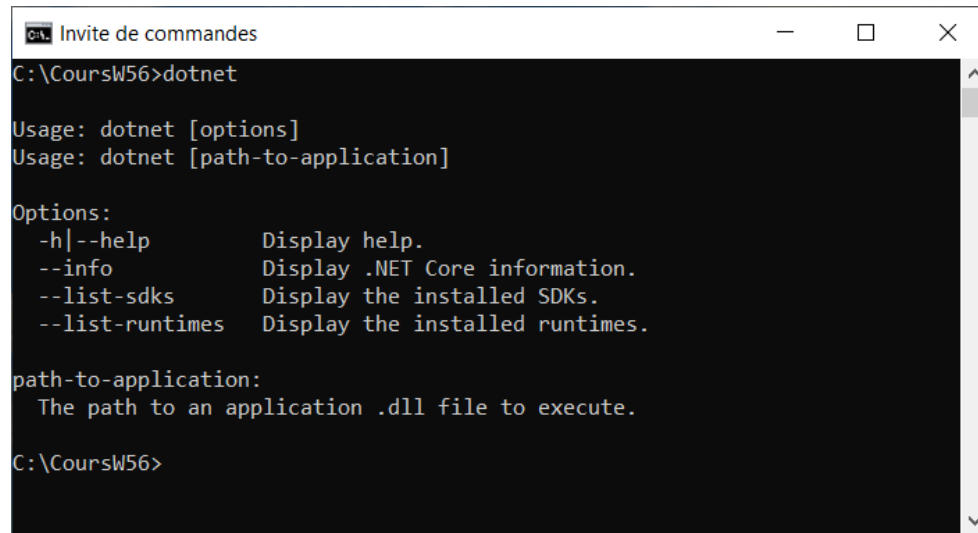
Exemple 3

Création d'un projet Blazor avec ligne de commande : <https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/create>

Ouvrez une nouvelle invite de commande et exécutez la commande suivante:

> dotnet

Si SDK .NET est installé sur votre machine, vous devriez voir une sortie similaire à la suivante:



```
C:\CoursW56>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help          Display help.
  --info             Display .NET Core information.
  --list-sdks        Display the installed SDKs.
  --list-runtimes    Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.

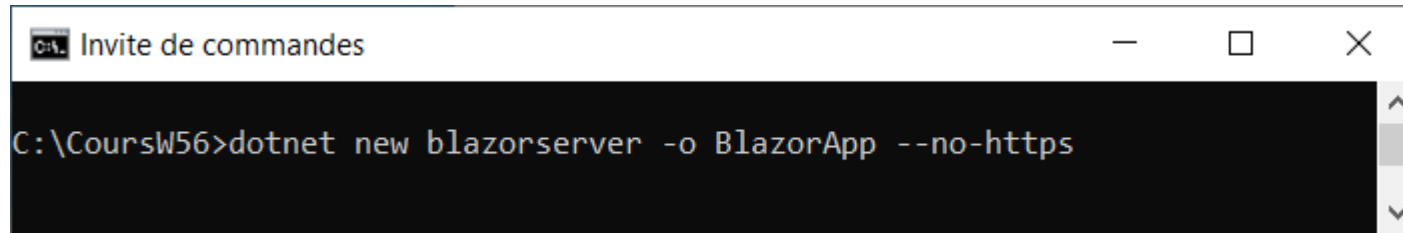
C:\CoursW56>
```

Exemple 3

Création d'un projet Blazor avec ligne de commande : <https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/create>

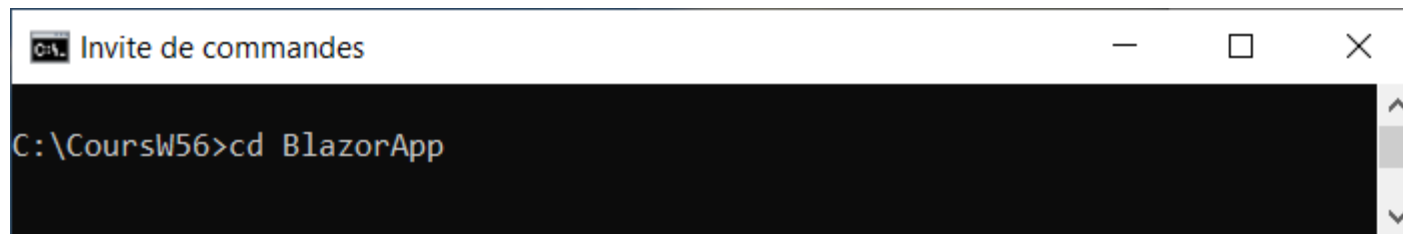
Créez votre appli

Dans votre invite de commande, exécutez la commande suivante pour créer votre application:



```
C:\CoursW56>dotnet new blazorserver -o BlazorApp --no-https
```

Ensuite, accédez au nouveau répertoire créé par la commande précédente:



```
C:\CoursW56>cd BlazorApp
```

Exemple 3

Création d'un projet Blazor avec ligne de commande : <https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/create>

Que signifient ces commandes?

- La commande `dotnet new blazorserver` crée une nouvelle application Blazor Server pour vous.
- Le paramètre `-o` crée un répertoire nommé dans `BlazorApp` lequel votre application est stockée et le remplit avec les fichiers requis.
- L'indicateur `--no-https` spécifie de ne pas activer HTTPS.
- La commande `cd BlazorApp` vous permet d'accéder au nouveau répertoire que vous venez de créer pour la nouvelle application.

Quels fichiers ont été créés?

Plusieurs fichiers ont été créés dans le répertoire `BlazorApp`, pour vous donner une application Blazor simple prête à fonctionner.

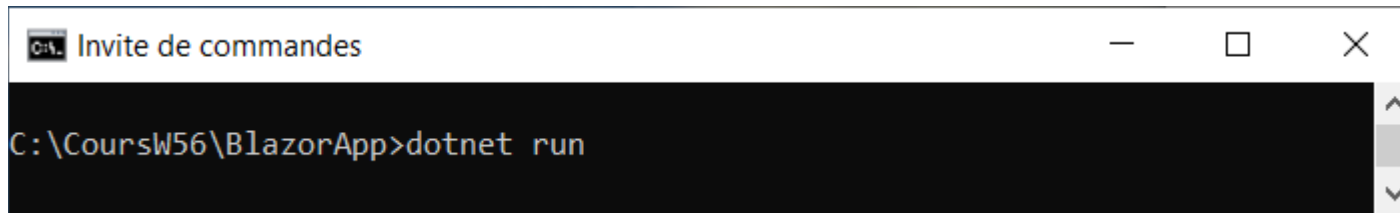
- `Program.cs` est le point d'entrée de l'application qui démarre le serveur.
- `Startup.cs` est l'endroit où vous configurez les services d'application et le middleware.
- `App.razor` est le composant racine de l'application.
- Le répertoire `BlazorApp/Pages` contient des exemples de pages Web pour l'application.
- `BlazorApp.csproj` définit le projet d'application et ses dépendances.

Exemple 3

Exécutez votre application :

<https://dotnet.microsoft.com/learn/aspnet/blazor-tutorial/create>

Dans votre invite de commande, exécutez la commande suivante:



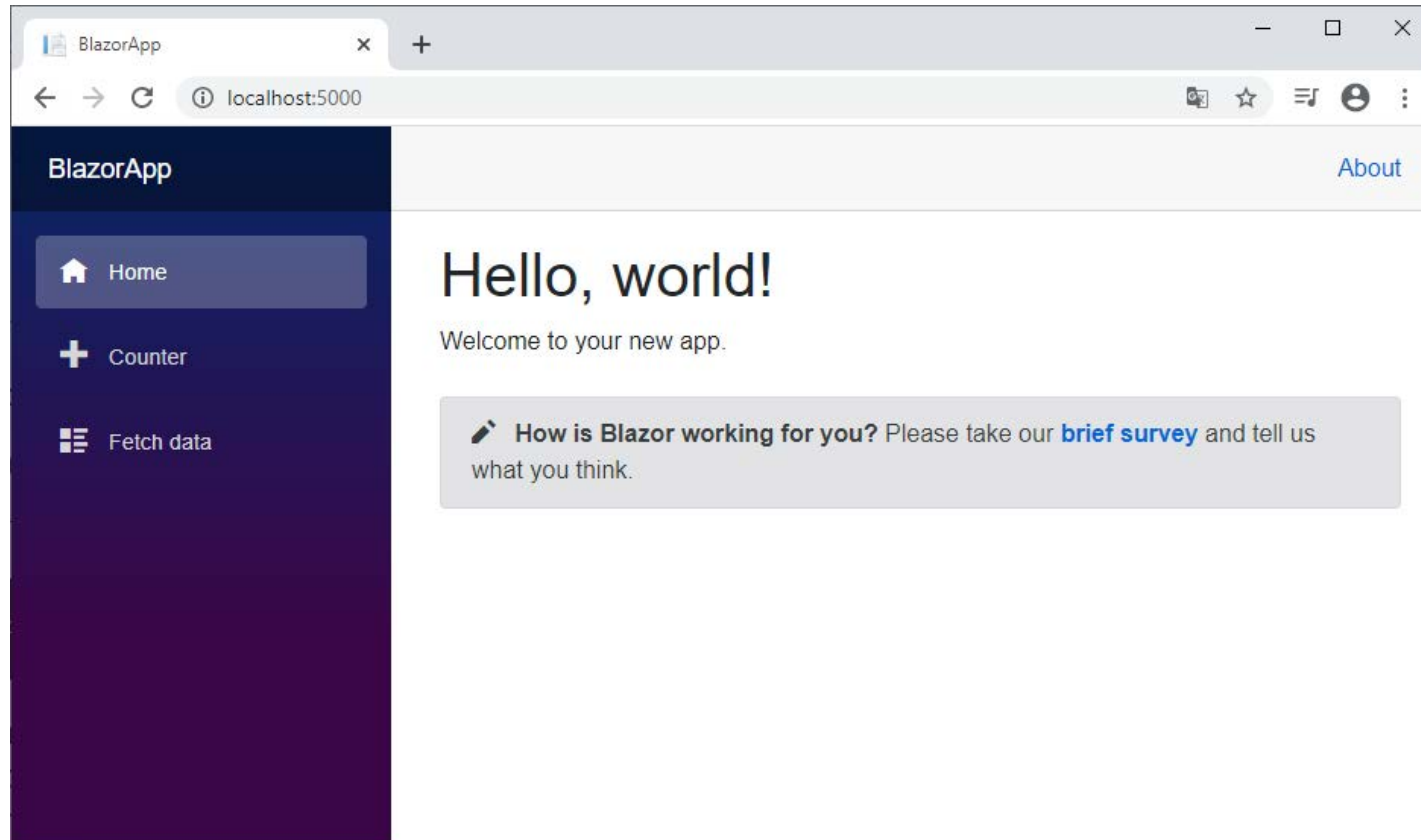
```
C:\CoursW56\BlazorApp>dotnet run
```

Attendez que la commande affiche qu'elle écoute sur <http://localhost:5000>, puis ouvrez un navigateur et accédez à cette adresse.

Une fois que vous arrivez à la page suivante, vous avez exécuté avec succès votre première application Blazor!

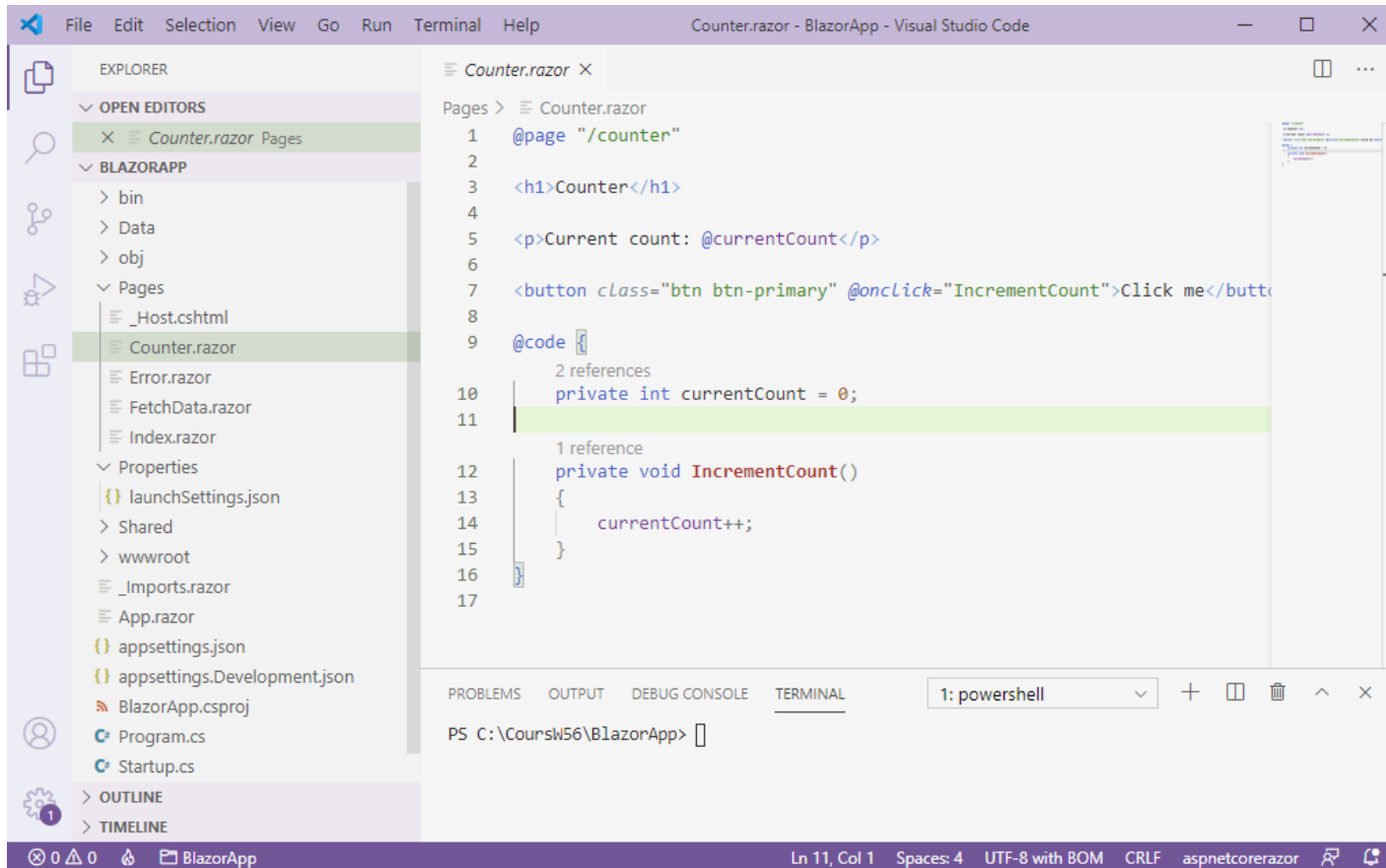
Exemple 3

L'exécution de projet Blazor donne la page suivante.



Exemple 3

Ouvrez le projet dans un éditeur de votre choix, comme exemple nous utilisons Visual studio code



Exemple 3

Vous pouvez ajouter une composante de liste de tâches à votre application Blazor en consultant les détails sur l'adresse suivante :

<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/build-a-blazor-app?view=aspnetcore-5.0>

