LA TECHNOLOGIE BLAZOR

Framework de Dotnet

M'Ba KUMENA
&
Chérifa OGBONE INOUSSA

Table de matières

Qu'est ce que Blazor

Blazor est un nouveau framework Web de Microsoft conçu pour concurrencer les plates-formes leaders du secteur telles que React. Sauf qu'au lieu d'utiliser JavaScript, il s'exécute sur le runtime .NET et permet aux développeurs de créer des applications Web interactives avec C# et HTML.

les différents modèles de Blazor

Server

fonctionne comme React Server Side Rendering et effectue la majeure partie du traitement sur le serveur. Les composants Razor s'exécutant côté serveur renvoyant du code HTML, CSS et Javascript aux clients via SignalR.

Web Assembly

le code Blazor compilé est envoyé et exécuté dans le navigateur Web et communique avec une application distante via des services (Web API / gRPC).

Hybrid

les composants sont capables de s'exécuter nativement (runtime .NET natif) dans des applications Web, Windows et mobiles (en ligne et hors ligne).

Blazor server

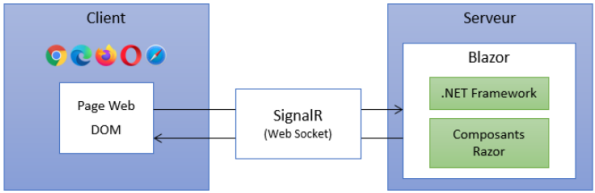

Figure: Server

Blazor Web Assembly

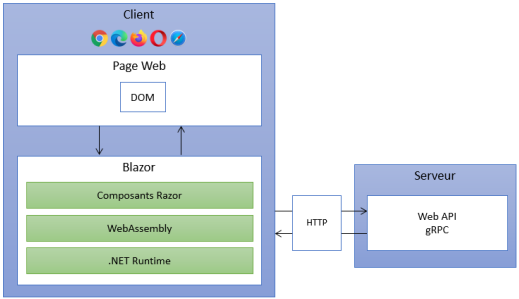

Figure: Web Assembly

C'est ce que Dotnet(.NET)

.NET est une plateforme de développement gratuite, multiplateforme et open source permettant de créer de nombreux types d'applications. Elle peut exécuter des programmes écrits dans plusieurs langages, C étant le plus populaire. Elle repose sur un runtime hautes performances utilisé en production par de nombreuses applications à grande échelle.
Il inclut les composants comme:

- Runtime : pour exécuter le code d'application.
- Bibliothèques
- Compilateur : compile le code source C (et d'autres langages) dans le code exécutable (runtime).
- Kit de développement logiciel (SDK) et autres outils : activez la création et la surveillance d'applications avec des flux de travail modernes.
- Piles d'applications

Installation du SDK .Net

Pour cette formation nous allons utiliser la version 7 SDK de Dotnet (.Net).

Installation des dépendances

- libc6
- libgcc-s1
- libgssapi-krb5-2
- libicu70
- liblttng-ust1
- libssl3
- libstdc++6
- libunwind8
- zlib1g

Installation du SDK .Net

- installer les dépendances:
  sudo apt install zlib1g libc6 libgcc-s1 libgssapi-krb5-2 libicu70 liblttng-ust1 libssl3 libstdc++6 libunwind8
- installer le Sdk .Net :
  sudo apt-get update
  sudo apt-get install -y dotnet-sdk-7.0

Pour plus d'infos :

https://learn.microsoft.com/en-us/dotnet/core/install

Création d'un projet

la commande: **dotnet new** va permettre de créer des modèles d'application en blazor

Blazor server

dotnet new blazorserver -o appName

Blazor web assembly

dotnet new blazorwasam -o appName

Structure d'un projet Blazor

- **Dossier shared** : contient les composants de disposition et les feuilles de style suivants :
  - composant MainLayout : le composant de disposition de l'application.
  - MainLayout.razor.css : feuille de style pour la disposition principale de l'application.
  - Composant NavMenu (NavMenu.razor) : implémente la navigation dans la barre latérale. Inclut le NavLinkcomposant (NavLink), qui affiche les liens de navigation vers d'autres Razorcomposants. Le composant NavLink indique à l'utilisateur quel composant est actuellement affiché.
  - NavMenu.razor.css : Feuille de style pour le menu de navigation de l'application.

Structure d'un projet Blazor

- **Dossier Pages** : contient les composants Razor côté serveur routables de l'application (.razor). L'itinéraire de chaque page est spécifié à l'aide de la directive @page. Le modèle comprend les éléments suivants :
  - Composant Counter (Counter.razor) : implémente la page Compteur.
  - Composant Error (Error.razor) : implémente la page d'erreur.
  - Composant Host (_Host.cshtml) :implémente la page Home.
  - Composant Weather (Weather.razor) : implémente la page de prévision météorologique
- **Dossier wwwroot** : dossier racine web du projet serveur contenant les ressources statiques publiques de l'application.

Structure d'un projet Blazor

- **DossierProperties** : contient configuration de l'environnement de développement dans le fichier launchSettings.json.
- Composant App (App.razor) : composant racine de l'application avec balisage HTML head , Routescomposant, et balise Blazor script . Le composant racine est le premier composant que l'application charge.
- _Imports.razor : inclut des directives Razor courantes à inclure dans les composants d'application rendus côté serveur (.razor), tels que les directives @using pour les espaces de noms.

Structure d'un projet Blazor

- **Fichiers de paramètres d'application (appsettings.Development.json, appsettings.json)** : fournissent des paramètres de configuration pour le projet serveur.
- **Program.cs** : point d'entrée du projet serveur qui configure l'application web ASP.NET Core hôte et contient la logique de démarrage de l'application, y compris les inscriptions de service, la configuration, la journalisation et le pipeline de traitement des requêtes.
- **Dossier Data** :Nos migrations, nos seeders

Les composants Razor

Dans Blazor, un composant est l'unité de base du code utilisateur. Il représente un morceau d'interface utilisateur (UI) réutilisable, avec sa logique associée encapsulée en C. Les composants sont à la fois des blocs de construction et les fondements d'une application Blazor, permettant de créer des interfaces complexes à partir de petites parties réutilisables et maintenables.
Un composant peut être aussi simple qu'un bouton avec interaction, ou aussi complexe qu'un formulaire complet avec validation des données. Les composants Blazor se composent de markup HTML, directives Razor, et code C pour manipuler les comportements dynamiques. La souplesse des composants Blazor facilite la création d'interfaces interactives en utilisant des concepts de programmation familiers aux développeurs .NET.

caracteristiques d'un composant

- Encapsulation: Un composant encapsule son markup,sa logique, et son style pour favoriser la réutilisabilité.
- Interactivité Les composants peuvent répondre aux interactions des utilisateurs et déclencher des mises à jour d'UI dynamiques.
- Réutilisabilité : Les composants sont conçus pour être utilisés à de multiples endroits dans une application sans répétition du code.

composant counter

```
<PageTitle>Counter</PageTitle>
<h1>Counter</h1>
<p role="status">Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click
me</button>

@code
{
private int currentCount = 0;
private void IncrementCount()
{
currentCount++;
} }
```

Creation des composants

La convention de nommage d'un composant en blazor est: La première lettre du nom doit être une majuscule et a pour extension **.razor**.
Un composant a deux parties :

- la partie HTML
- la partie Code (la logique)

Nous allons utiliser les directives et des attributs :

- **@bind et @bind-value** : pour lier les valeurs dans le composant

Nous pouvons également utiliser du code c# dans notre composant:

- **@**: permet d'utiliser du code C# dans le HTML (variable)
- **@code** c'est là que toute la logique du composant y réside.

La directive **[Parameter]**: permet déclarer les paramètre d'un composant.

Compteur.razor

```
<h1>Compteur : @current</h1>
<button class="btn btn-primary" @onclick="IncrementCount">Click
me</button>
@code {
[Parameter]
public int Initial { get; set;} =0
[Parameter]
public int step { get; set;} =1; private int current = 0;
protected override void OnInitialized() { current=Initial; }
private void IncrementCount() { current+=Initial; }
}
```