Componente Modal – Usando Blazor

Guías de Curso – 14/10/2022

Autor:
Gustavo Eduardo Gil Prado
Docente

Resumen:

Los componentes modales permiten adicionar interactividad al proyecto, por medio de esta guía, adicionaremos una ventana modal sencilla a un proyecto Blazor Server, usando el .Net 6 y Visual Studio Code

Palabras clave:
Programación, web, Blazor, C#, Spa

Descripción:

Este documento presenta los pasos para adicionar un componente modal a una aplicación Blazor empleando Visual Studio Code como editor de código

FACULTAD DE INGENIERÍA
Ingeniería Informática
Institución Universitaria Colegio Mayor del Cauca

Referencie este documento así: Gil, G. E. (2022). Componente Modal – Usando Blazor. Institución Universitaria Colegio Mayor del Cauca.

Creación de un componente Modal en Blazor (Usando .Net 6.0)

Para este ejercicio crearemos un componente reutilizable que nos ayudara en aquellos escenarios que es necesaria la confirmación del usuario o cuando debamos confirmar la culminación de una tarea previamente iniciada.

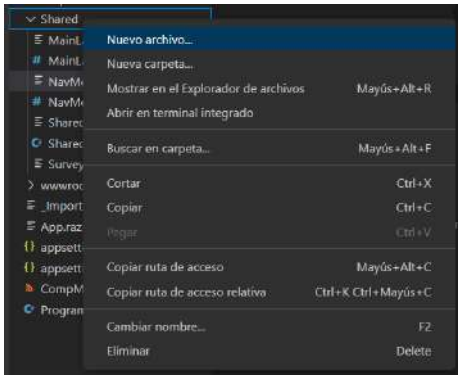
A continuación, encontraremos los diferentes pasos para completar esta tarea.

Creación del componente reutilizable

Iniciamos VS Code y creamos un nuevo proyecto de Blazor Server que emplearemos como base para esta prueba, para esto no olvidemos que deberemos crear un directorio (carpeta) en el ligar de nuestra elección, para en este lugar crear los archivos mediante el comando `dotnet new`

```
C:\CompModal\dotnet new blazorserver
```

A continuación, encontraremos un nuevo proyecto y sus archivos base en el directorio indicado, ahora, crearemos un nuevo archivo en donde codificaremos el componente reutilizable, este estará ubicado en el directorio *Shared* del proyecto, en mi caso lo llamare: `ModalComponent.razor`



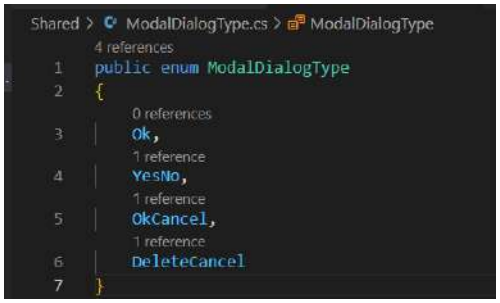
Img. 1 - Nuevo Archivo en VS Code

En este componente deberemos codificar la estructura del HTML que dará soporte a los elementos visuales y la lógica en C# que le permita adaptarse y responder a los diferentes escenarios, es importante notar que este componente no contara con la directiva `@page` que permite especificar la ruta del componente, por lo que no podrá ser invocado desde la url como es el caso de los componentes *`Counter.razor`* y *`FetchData.razor`* creados por la plantilla genérica del proyecto Blazor Server.

La siguiente imagen presenta la estructura genérica de un modal-dialog en Bootstrap, que incluye un encabezado, el cuerpo de la ventana modal y un pie de pagina (opcional) en donde ubicaremos los diferentes botones de acuerdo con la opción necesaria

Img. 2 – Primera versión del componente modal (sección HTML)

De la misma manera crearnos el archivo `ModalDialogType.cs` en el directorio *`shared`*, que mantendrá los tipos de datos enumerados (`enum`) que describen el grupo de tipos de ventanas modales que estarán disponibles



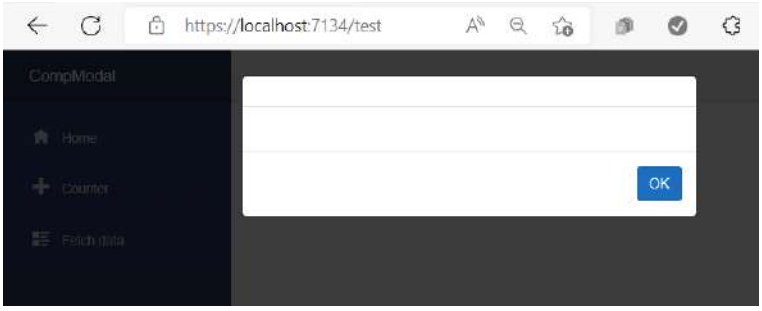
Img. 3 – enum ModalDialogTypes

Creación del componente direccionable

Una vez terminadas estas tareas, es momento de crear el componente direccionable que emplearemos para lanzar el componente modal, en este caso estará en el archivo `Test.razor` que se ubicará en el directorio *`Pages`*.

Img. 4 – Primera versión del componente Test, llamado al componente modal

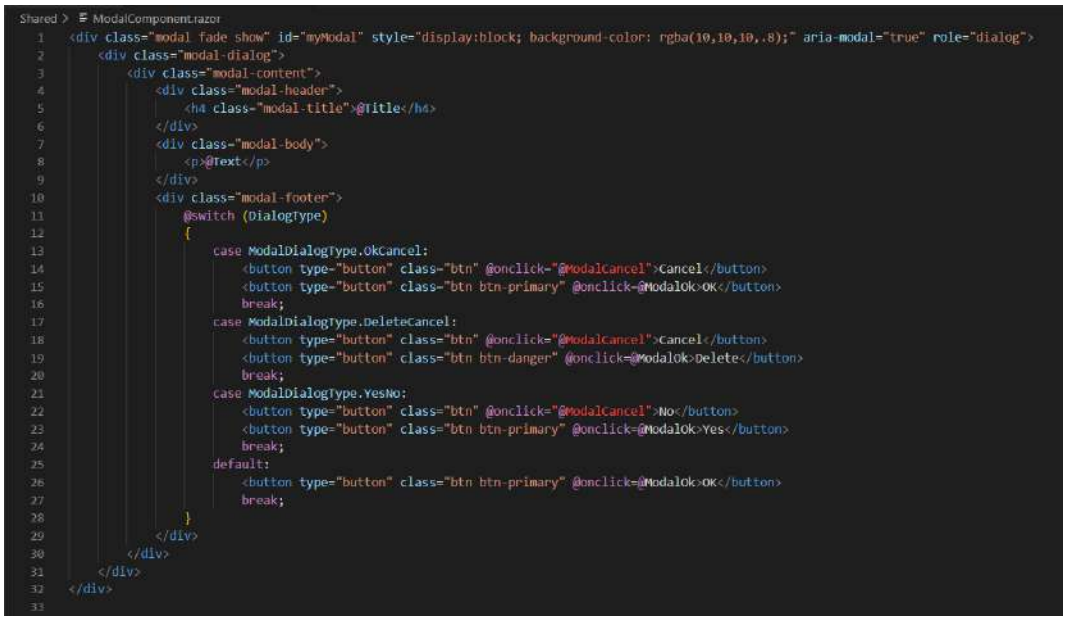
De esta sencilla manera podremos incluir el componente modal en `Test.razor`, al ejecutarlo obtendremos una primera vista del elemento que esta en construcción



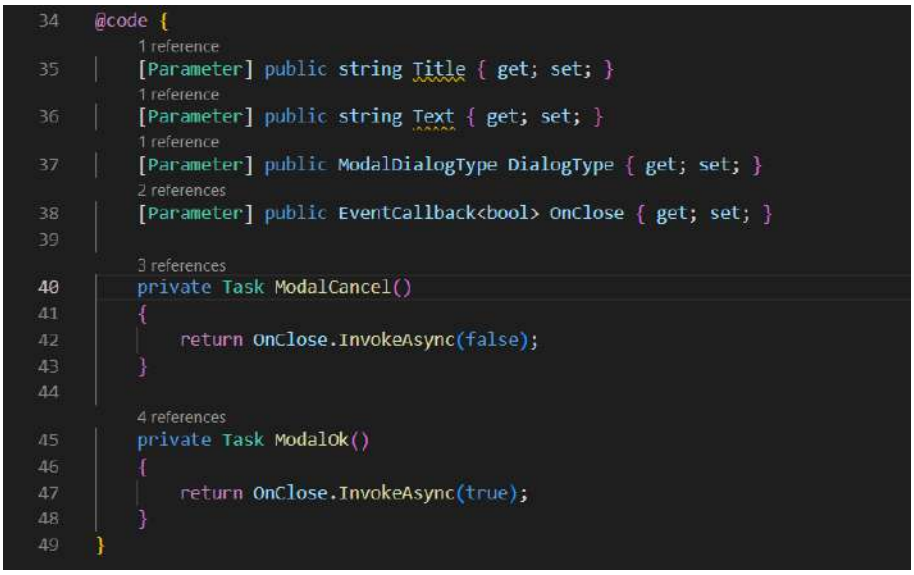
Img. 5 – El componente modal por primera vez

Completar el código y realizar la prueba final

Es momento de adicionar los parámetros para el título, texto y demás elementos necesarios para completar la funcionalidad del componente modal, en este caso adicionaremos el atributo `[Parameter]` a aquellas propiedades que recibirán desde quien invoca al componente los valores necesarios. Ahora agregaremos nuevos elementos a `ModalComponent.razor`, las siguientes capturas de pantalla ilustran estas dos secciones de código de forma separada, pero constituyen un único archivo

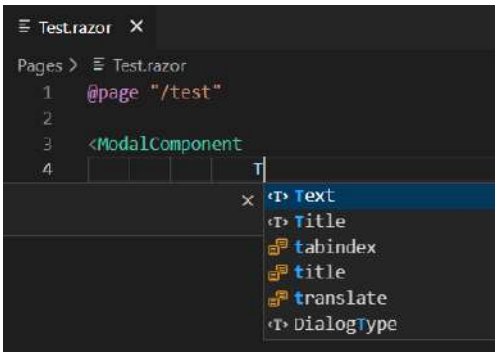


Img. 6 – Sección HTML completa del componente modal



Img. 7 – Sección @code del componente modal

Ahora, adicionaremos al llamado del componente modal, los detalles que se han de mostrar, para esto en la etiqueta inicial adicionamos los que consideremos necesarios



Img. 8 – Llamado a los Parámetros del componente modal

Obtendremos un resultado similar al de la siguiente imagen, aquí tenemos una variable booleana que permite alternar entre mostrar u ocultar el componente modal de acuerdo a si se ha hecho clic en el botón o si ya se obtuvo respuesta desde el método de cierre del componente modal (mediante el control de eventos a través del uso de EventCallback, que en estos componentes anidados permite el paso de mensajes) lo que permite ejecutar diferentes acciones de acuerdo a la respuesta dada por el usuario.

```
Pages > Test.razor
1  @page "/test"
2
3  <button type="button" class="btn btn-danger btn-sm"
4          @onclick="() => OpenDialog()">Abrir</button>
5
6  @if (dialogOpen)
7  {
8      <ModalComponent
9          Title="Desea continuar ...?"
10         Text="Al confirmar se realizaran los cambios realizados"
11         DialogType=ModalDialogType.YesNo
12         OnClose="@OnDialogClose">
13     </ModalComponent>
14 }
15 @if(!string.IsNullOrEmpty(resp))
16 {
17     <p>Respuesta de la ventana modal @resp</p>
18 }
19
```

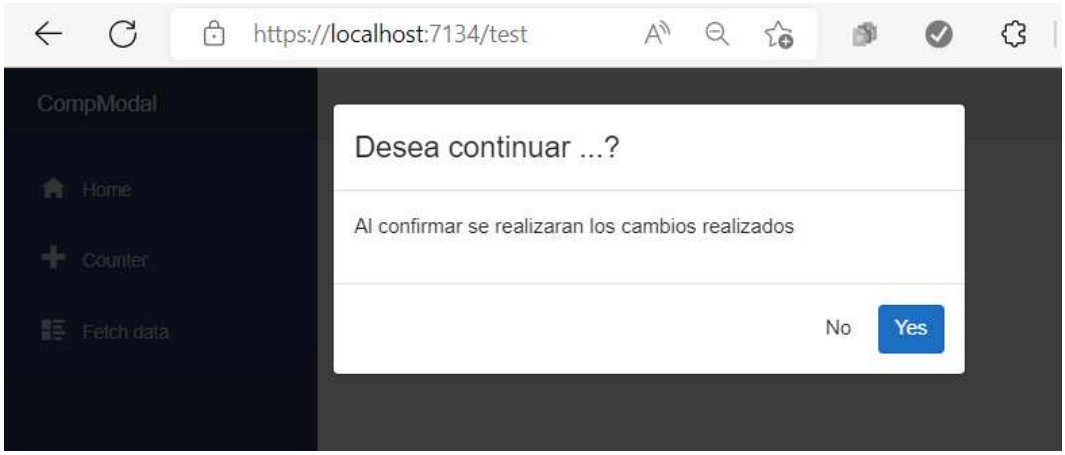
Img. 9 – Sección HTML del componente Test

```
20
21  @code{
        4 references
22      private string resp = string.Empty;
        3 references
23      private bool dialogOpen;
24
        1 reference
25      private async Task OnDialogClose(bool accepted)
26      {
27          if (accepted)
28              resp = "Si";
29          else
30              resp = "No";
31
32          dialogOpen = false;
33          StateHasChanged();
34      }
35
        1 reference
36      private async Task OpenDialog()
37      {
38          dialogOpen = true;
39          StateHasChanged();
40      }
41  }
```

Img. 10 - Seccion @code del componente Test

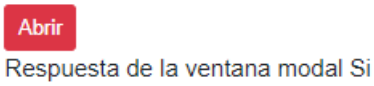
Resultado final ¿eso existe?

Ya con estos cambios listos, podremos nuevamente realizar una prueba, que nos permite evidenciar la flexibilidad del componente, así como responder adecuadamente según la opción seleccionada por el cliente de acuerdo con los botones expuestos.

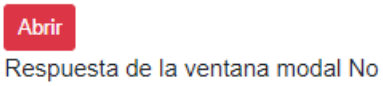


Img. 11 – Resultado final al invocar al componente modal

Podemos tratar la respuesta del usuario y obtener diferentes acciones a partir de su elección, en este caso, se mostrará un sencillo párrafo en cada caso particular



Img. 12 – Respuesta en Test al presionar [Yes] en el componente modal



Img. 13 - Respuesta en Test al presionar [No] en el componente modal

Contenido adaptado de: https://www.puresourcecode.com/dotnet/blazor y de otros sitios web públicos

Ultima Actualización: 28/09/2022