Componente de selección múltiple en Blazor.

Contenido

[1. Tecnología 1](#_Toc90564083)

[2. Uso básico 1](#_Toc90564084)

[3. Lista de ítems 1](#_Toc90564085)

[4. Propiedades 2](#_Toc90564086)

[5. Visibilidad y ocultamiento 2](#_Toc90564087)

[6. Evento OnCheckChange 3](#_Toc90564088)

[7. Archivos 3](#_Toc90564089)

[8. Relación entre los archivos y el código 4](#_Toc90564090)

[9. Order By 4](#_Toc90564091)

[10. Cantidad de ítems 5](#_Toc90564092)

[11. Accediendo al listado de ítems 5](#_Toc90564093)

1. Tecnología: El componente se ha diseñado con tecnología Blazor, para ser utilizado en aplicaciones del tipo “blazor server side” en cuanto al lenguaje es C#. El desarrollo se llevó a cabo utilizando “Visual Studio 2019” como IDE (Entorno de Desarrollo Integrado).
2. Uso básico: En caso de no establecerse las propiedades, el componente “**TMultiSelectTore**” las inicializara con valores por defecto (default values).

|  |
| --- |
|  |
|  |

1. Lista de ítems: La propiedad “Items” nos permite establecer la lista que el componente debe manejar. Podemos ver que el “**Volvo**” está marcado ya que la propiedad “**Checked**” así lo indica.

|  |
| --- |
|  |
|  |

1. Propiedades: La siguiente imagen presenta algunas de las propiedades y su correspondiente impacto en el componente.

|  |
| --- |
|  |
|  |

1. Visibilidad y ocultamiento: Las propiedades que inician con el prefijo “**Visible**”, nos permiten ocultar funcionalidad no deseada. El ejemplo siguiente muestra la propiedad “**VisibleSearch**” establecida como verdadero, razón por la cual el comportamiento de búsqueda de ítems por parte de su texto, es otorgado al componente. En contraposición a las propiedades “**VisibleViewOnlyChecked**="false"” y “**VisibleCheckAll**="false"”, que desactivan la funcionalidad de filtro por ítems marcados y el marcado automático de todos los ítems, respectivamente.

|  |
| --- |
|  |
|  |

1. Evento OnCheckChange: El evento se dispara cada vez que un ítem es marcado o desmarcado. El siguiente código muestra un método de atención al evento “**OnCheckChange**”, que recibe como parámetro el ítem que ha cambiado de estado.

|  |
| --- |
|  |
|  |

1. Archivos: El componente consta de tres archivos de código, “**TItem.cs**”, “**TMultiSelectTore.razor**” y “**TOrderByTypes.cs**” que se encuentran en el directorio “**ComponentMultiSelectTore**”. En al archivo “Pages\**Index.razor**”, encontrara un ejemplo de uso del componente embebido dentro de una página con sus propiedades y eventos. Puede el desarrollador estudiar el código, modificarlo y mejorarlo ya que el mismo se encuentra abierto (Open Source).

|  |
| --- |
|  |
|  |

1. Relación entre los archivos y el código: Podemos ver como el nombre de los archivos coinciden con los de las clases y del propio componente.

|  |
| --- |
|  |
|  |

1. Order By: Mediante una enumeración, la propiedad “**OrderBy**” nos permite darle un orden a los ítems de la lista. Siendo esta propiedad opcional y su valor por defecto “TOrderByTypes.**Nothing**”, sin orden. Pudiendo tomar alguno de los siguientes valores:
   1. **Label**: Ordena por la etiqueta o texto que ve el usuario.
   2. **Value**: Ordena por el valor, no es visto por el usuario.
   3. **Checked**: Muestra los ítems marcados primero y los no marcados después.
   4. **Nothing**: No ordena, respetando de esta manera el orden natural o predefinido en la lista de ítems.

|  |
| --- |
|  |
|  |

1. Cantidad de ítems: Se han realizado pruebas con listas de 1000 ítems y la performance es, ¿Cómo decirlo?, hunamente usable, en cuanto a los tiempos de respuesta de la carga inicial y de las búsquedas, aunque estos tiempos dependerán de la potencia de la maquina sobre la que se esté ejecutando el código.
2. Accediendo al listado de ítems: Puede acceder al listado de ítems mediante una referencia al componente y luego accediendo a su propiedad “Items” para determinar cuál está marcado y cual no. Otra manera es suscribirse utilizar el evento “**OnCheckChange**” ya visto en este documento.

|  |
| --- |
|  |
|  |