Blazor ListBox Overview

The ListBox for Blazor is an enhanced version of the HTML `<select multiple>` element. The ListBox provides many additional features such as item reordering, item removal, and moving items from one ListBox to another through toolbar buttons or drag-and-drop. The ListBox also allows single or multiple item selection and will show a vertical scrollbar automatically if the items don't fit. The component features templates to customize its rendering.

Creating Blazor ListBox

To use a Telerik ListBox for Blazor:

1. Add the `TelerikListBox` tag.
2. Set the `Data` parameter to an `IEnumerable<T>`.
3. Set `TextField` to the property name that holds the string values to display in the ListBox. Skip this step when binding the component to a collection of primitive values.
4. Set `SelectedItems` to an `IEnumerable<T>` to store or change the component selection. Optionally, enable multiple selection.
5. Configure the ListBox toolbar in `<ListBoxToolBarSettings>` and specify which buttons

will be visible. By default, the toolbar shows all buttons. Each button requires an event handler to work.
6. (optional) Set the `Width` and `Height` parameters, based on the number of toolbar buttons and desired number of visible items. The component will automatically show a vertical scrollbar if needed. Long items will wrap.
7. Set the `@ref` attribute and obtain reference to the ListBox instance. This is necessary to `Rebind()` the component after programmatic data changes.

Basic Blazor ListBox

```
@* ListBox with item selection and reordering *@

<TelerikListBox @ref="@ListBoxRef"
                Data="@ListBoxData"
                TextField="@nameof(ListBoxModel.Name)"
                SelectionMode="@ListBoxSelectionMode.Multiple"
                @bind-SelectedItems="@ListBoxSelectedItems"
                OnReorder="@( (ListBoxReorderEventArgs<ListBoxModel> args) =>
OnListBoxReorder(args) )"
                Width="180px"
                Height="auto">
    <ListBoxToolBarSettings>
        <ListBoxToolBar>
            <ListBoxToolBarMoveUpTool />
            <ListBoxToolBarMoveDownTool />
        </ListBoxToolBar>
    </ListBoxToolBarSettings>
</TelerikListBox>

@code {
    private TelerikListBox<ListBoxModel> ListBoxRef { get; set; } = null!;

    private List<ListBoxModel> ListBoxData { get; set; } = new
List<ListBoxModel>();

    private IEnumerable<ListBoxModel> ListBoxSelectedItems { get; set; } = new
List<ListBoxModel>();

    private void OnListBoxReorder(ListBoxReorderEventArgs<ListBoxModel> args)
    {
        ListBoxData.RemoveAll(x => args.Items.Contains(x));
        ListBoxData.InsertRange(args.ToIndex, args.Items);

        ListBoxRef.Rebind();
    }

    protected override void OnInitialized()
    {
        for (int i = 1; i <= 7; i++)
        {
            ListBoxData.Add(new ListBoxModel()
```

```
            {
                Id = i,
                Name = $"ListBox Item {i}",
            });
        }
    }

    public class ListBoxModel
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
    }
}
```

Data Binding

The ListBox supports binding to a model class, which requires setting the `TextField` parameter, unless the property name is `"Text"`. Another option is to bind the component to a collection of primitive strings.

Bind ListBox to List<string>

```
<TelerikListBox Data="@ListBoxStrings"
                @bind-SelectedItems="@ListBoxSelectedStrings"
                SelectionMode="@ListBoxSelectionMode.Multiple"
                Height="auto">
    <ListBoxToolBarSettings>
        <ListBoxToolBar Visible="false" />
    </ListBoxToolBarSettings>
</TelerikListBox>

@code {
    private List<string> ListBoxStrings { get; set; } = new List<string>();

    private IEnumerable<string> ListBoxSelectedStrings { get; set; } = new
List<string>();

    protected override void OnInitialized()
    {
        for (int i = 1; i <= 7; i++)
        {
            ListBoxStrings.Add($"String {i}");
        }
    }
}
```

Toolbar

The ListBox includes a toolbar with some built-in buttons. These tools fire events, which are related to item removal, reordering, or transfer to another ListBox. The component supports removing some of the default buttons or adding custom ones. You can also control the toolbar position with regard to the ListBox item list, or hide the toolbar completely.

Reordering

The ListBox allows users to move one or multiple items up and down with built-in toolbar buttons. The component will fire its `OnReoder` event, so that the app can apply the new item order in the ListBox `Data`. See an example above or on the ListBox Events page.

Selection

Users can select just one ListBox item or multiple items with the mouse or keyboard. The behavior depends on the `SelectedItems` parameter value.

Move Items Between ListBoxes

You can connect several ListBox components and enable users to move items from one ListBox to another.

Drag and Drop Items

Users can also reorder items or move items to another ListBox with drag and drop.

Templates

The ListBox component provides templates to enable developers to customize the rendering and appearance of the component.

Events

The various ListBox events allow you to implement custom functionality and handle user interactions with the component's toolbar.

Next Steps

- Configure the ListBox toolbar
- Choose the ListBox selection mode
- Connect Multiple ListBoxes
- Enable ListBox drag-and-drop
- Implement ListBox templates
- Handle ListBox events