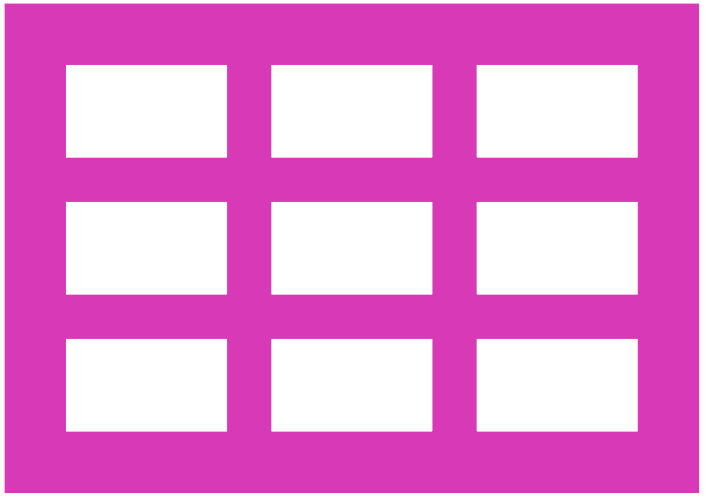
Session 5 - Single Table List Using MudBlazor Grids

Displaying Customer Data

Using Tables in Blazor to Display Data

Tables are a fundamental component for displaying structured data in web applications. Blazor makes it easy to create dynamic tables that can display data from various sources such as databases or APIs.



Overview

- NOTE: We are reusing the code from the OLTP – GetCustomers.linq
- Copy the “Customer Search” view model (CustomerSearchView) to the view model folder.
- Create the Customer Service.
- Add the GetCustomers() method to the service.
- Add transient for the “Customer Service” to “HogWildExtension.”
- Add a navigational page reference for navigating to the Customer List page.
- Create a Blazor Page and code for retrieving a list of customers based on the search parameters.
- Add GetCustomers methods from the “CodeBehind” class of the GetCustomers.linq

Overview

Customers

Customer Search

Last Name: foster

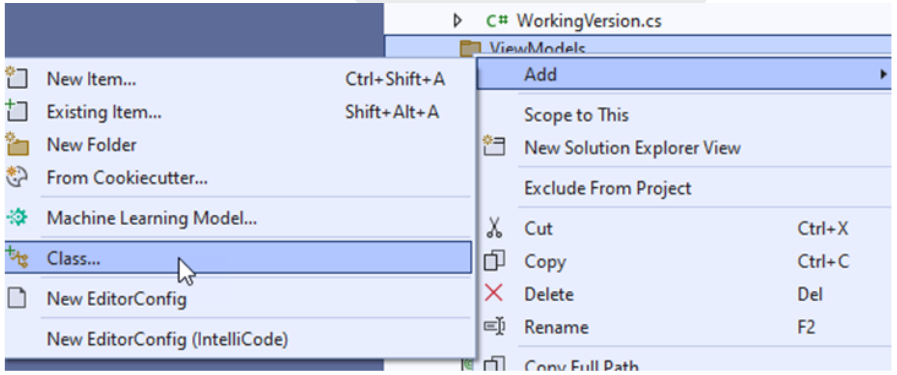
Phone Number

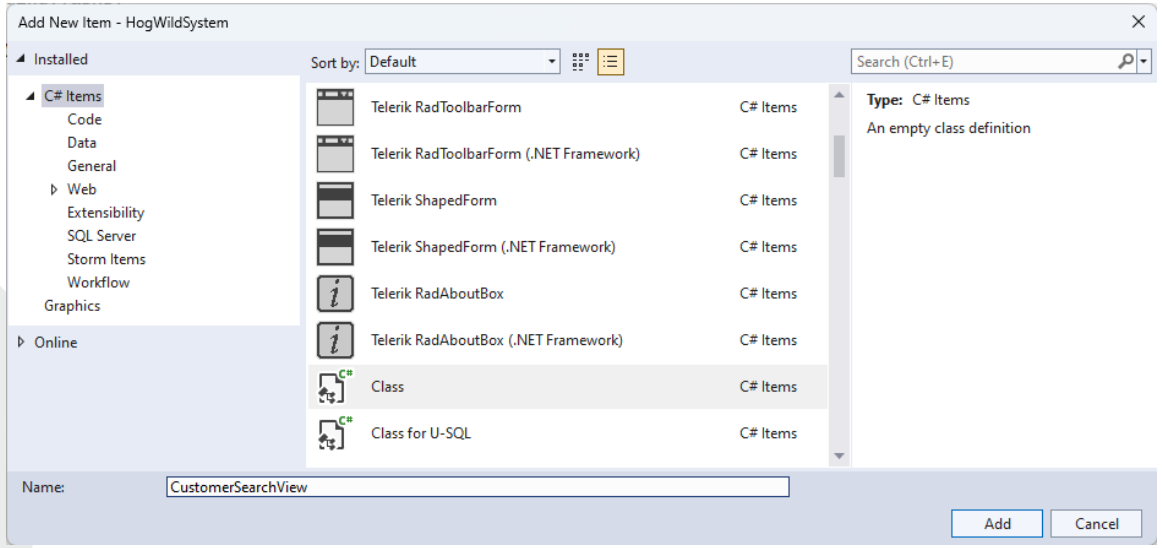
SEARCH NEW

Actions	First Name	Last Name	City	Phone	Email	Total Sales
EDIT NEW INVOICE	Fred	Foster	Edmonton	7804326565	ffoster@hotmail.com	$28,629.10
EDIT NEW INVOICE	Lucy	Foster	Edmonton	7802021177	l.foster@outlook.com	$0.00
EDIT NEW INVOICE	Kellan	Foster	Edmonton	7808986311	k.foster@yahoo.com	$0.00
EDIT NEW INVOICE	Kevin	Foster	Edmonton	7809408551	k.foster@tutanota.com	$0.00
EDIT NEW INVOICE	Paul	Foster	Edmonton	7802095117	p.foster@mail.ru	$0.00
EDIT NEW INVOICE	Madeline	Foster	Edmonton	7804541144	m.foster@icloud.com	$0.00

Creating Customer Search View Model

We must create a view model for the “Customer” table by adding a new class called CustomerSearchView to the ViewModels folder.





Creating Customer Search View Model (Continue)

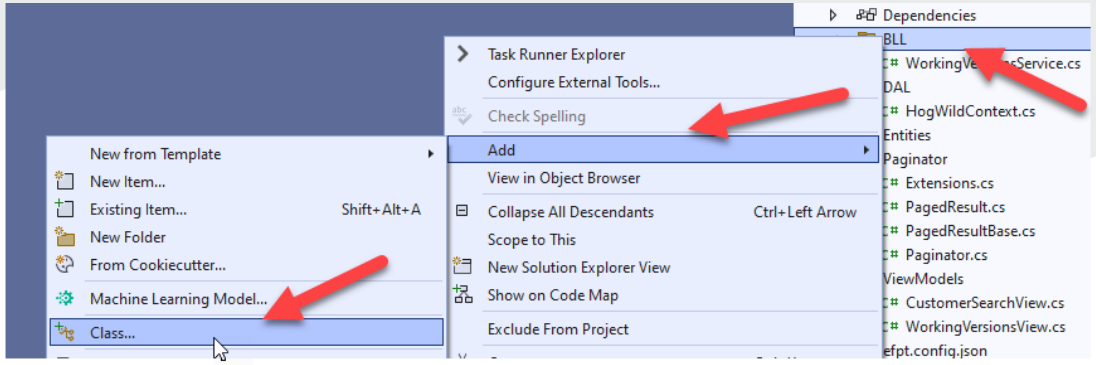
- Copy the view model from the “GetCustomers.linq”

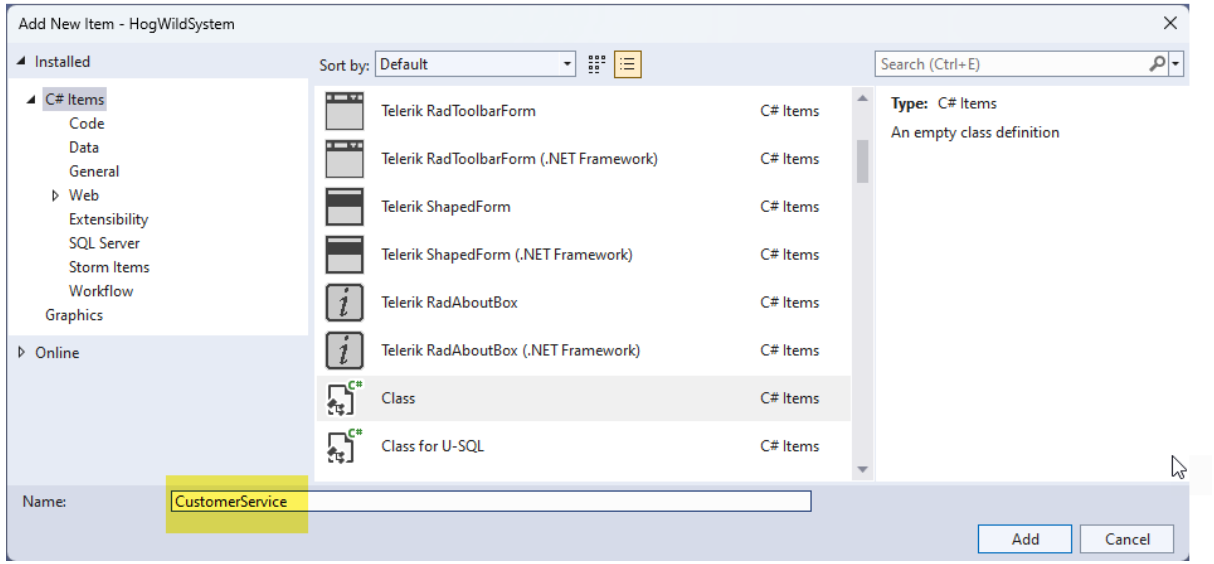
CustomerSearchView.cs

```
namespace HogWildSystem.ViewModels
{
    0 references
    public class CustomerSearchView
    {
        0 references
        public int CustomerID { get; set; }
        0 references
        public string FirstName { get; set; }
        0 references
        public string LastName { get; set; }
        0 references
        public string City { get; set; }
        0 references
        public string Phone { get; set; }
        0 references
        public string Email { get; set; }
        0 references
        public int StatusID { get; set; }
        0 references
        public decimal TotalSales { get; set; }
    }
}
```

Create Customer Service

Add a new “CustomerService” class to the BLL folder





Create Customer Service (Continue)

- Add the “Hog Wild“ context.
- Add the constructor for the “Customer” service.

CustomerService.cs

```
CustomerService.cs
HogWildSystem | HogWildSystem.BLL.CustomerService | CustomerSer

using HogWildSystem.DAL;

namespace HogWildSystem.BLL
{
    1 reference
    public class CustomerService
    {
        #region Fields

        /// <summary>
        /// The hog wild context
        /// </summary>
        private readonly HogWildContext _hogWildContext;

        #endregion

        // Constructor for the CustomerService class.
        0 references
        internal CustomerService(HogWildContext hogWildContext)
        {
            // Initialize the _hogWildContext field with the provided HogWildContext instance.
            _hogWildContext = hogWildContext;
        }
    }
}
```

Add GetCustomers() Method

Add the GetCustomers method from LINQPad to the CustomerService class.

CustomerService.cs

CustomerService.cs

HogWildSystem | HogWildSystem.BLL.CustomerService

```
15      //  Constructor for the CustomerService class.
        1 reference
16      internal CustomerService(HogWildContext hogWildContext)
17      {
18          //  Initialize the _hogWildContext fiekd with the provieded HogWildContext instance.
19          _hogWildContext = hogWildContext;
20      }
21
        1 reference
22      public Result<List<CustomerSearchView>> GetCustomers(string lastName, string phone)
23      {
24          // Create a Result container that will hold either a
25          //  ArtistEditView objects on success or any accumulated errors on failure
26
27          var result = new Result<List<CustomerSearchView>>();
28          #region Business Rules
29          //    These are processing rules that need to be satisfied
30          //        for valid data
31
32          //  rule:   Both last name and phone number cannot be empty
33          //  rule:   RemoveFromViewFlag must be false (soft delete)
34          if (string.IsNullOrEmpty(lastName) && string.IsNullOrWhiteSpace(phone))
35          {
36              result.AddError(new Error("Missing Information",
37                  "Please provide either a last name and/or phone number"));
```

Handling Issues with Current Context

This error typically occurs when trying to access a variable or class named CustomerSearchView,' but the compiler can't find it in the current context.

We need to add a reference to _hogWildContext, which we define at the beginning of the file.

Add “#nullable disable” to the top of the file to disable nullable warning message.

NOTE: If you have followed the BYSResults pattern that we have created by building the functionality in LINQPad, then you do not have to do this step as we are already using the “_hogWildContext”

CustomerService.cs

```
// Return customers that meet our criteria
return Customers
    .Where(x => (x.LastName.Contains(lastName)
                 || x.Phone.Contains(phone))
                && !x.RemoveFromViewFlag)
    .Select(x => new CustomerSearchView
```

```
// Return customers that meet our criteria
return _hogWildContext.Customers // DbSet<Customer>
    .Where(x :Customer => (x.LastName.Contains(lastName)
                 || x.Phone.Contains(phone))
                && !x.RemoveFromViewFlag) // IQueryable<Customer>
    .Select(x :Customer => new CustomerSearchView
```

Add Customer Service to Transient Service

Create a new service.AddTransient<t> to the HogWildExtension.

If you copy the previous AddTransient and update the previous service name to your current service name.

HogWildExtension.cs

```
services.AddTransient<WorkingVersionsService>((ServiceProvider) =>
{
    //  Retrieve an instance of HogWildContext from the service provider.
    var context = ServiceProvider.GetService<HogWildContext>();

    // Create a new instance of WorkingVersionsService,
    //   passing the HogWildContext instance as a parameter.
    return new WorkingVersionsService(context);
});

services.AddTransient<CustomerService>((ServiceProvider) =>
{
    //  Retrieve an instance of HogWildContext from the service provider.
    var context = ServiceProvider.GetService<HogWildContext>();

    // Create a new instance of CustomerService,
    //   passing the HogWildContext instance as a parameter.
    return new CustomerService(context);
});
```

NavMenu.razor

Customer List Navigation

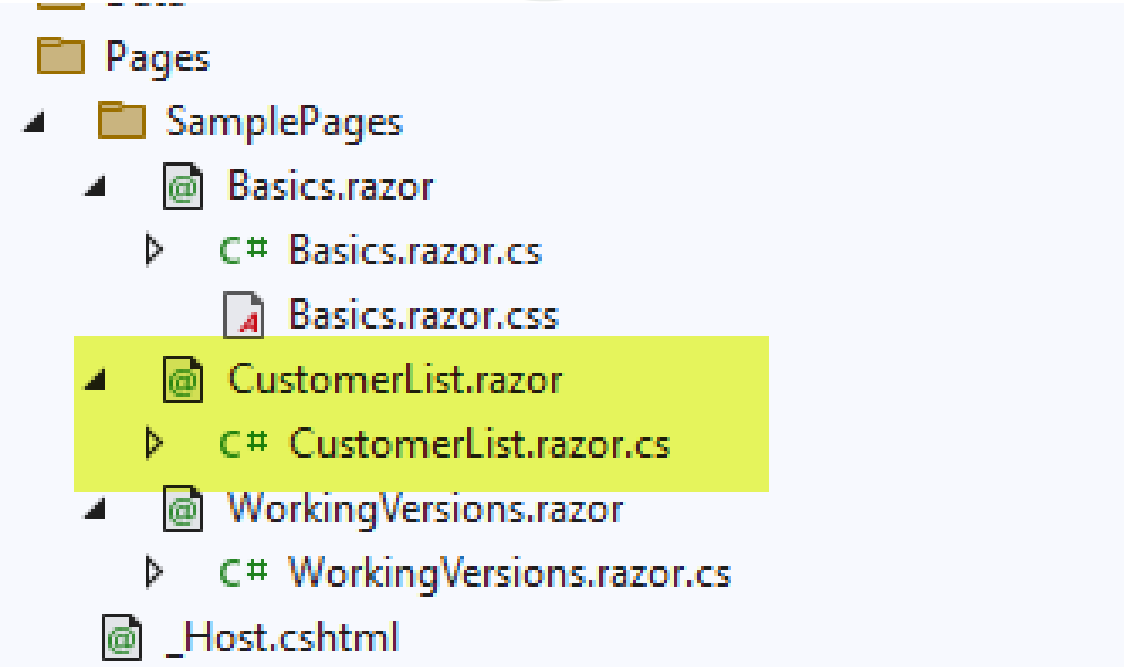
Add a “Customer List” MudNavLink to the NavMenu class.

```
<MudNavLink Href="SamplePages/WorkingVersions" Match="NavLinkMatch.Prefix"
            Icon="@Icons.Material.Filled.Album">Working Versions</MudNavLink>

<MudNavLink Href="SamplePages/CustomerList" Match="NavLinkMatch.Prefix"
            Icon="@Icons.Material.Filled.People">Customers</MudNavLink>
```

Customer List Page

- In the “Sample Page” folder, add two new files
  - CustomerList.razor (Razor Component)
  - CustomerList.razor.cs (Class)



Initial Refactoring of Customer List

- Remove all code from the CustomerList.razor and updated with the following:
  - @Page – Razor component for page accessible
  - <PageTitle> - Page title component to set the page title
  - <h3> - Main heading of the page
- Updated the class with “partial”

CustomerList.razor

```
CustomerList.razor
1  @page "/SamplePages/CustomerList"
2  <PageTitle>Customer List</PageTitle>
3  <h3>Customer List</h3>
```

```
CustomerList .razor | .razor.cs
HogWildWebApp | HogWildWebApp.Pages.SamplePages.Customerl
1  namespace HogWildWebApp.Pages.SamplePages
2  {
       1 reference
3      public partial class CustomerList
4      {
5      }
6  }
7
```

CustomerList.razor.cs

Customer List Overview

- The customer list is made up of 4 functional areas.

1. Search (Partial search using minimum one of the following parameters)
   - Last Name
   - Phone #
2. New Customer
   - Unique First Name, Last Name, Phone #
3. Customer List
   - List of customer properties
   - Edit a selected customer
   - Create a new invoice for a selected customer
   - Pagination
4. Feedback area for any messages

Customer List Overview

Customers

Search for customer(s) was successful

Customer Search

Last Name: fo

Phone Number

SEARCH NEW

Actions	First Name	Last Name	City	Phone	Email	Email
EDIT NEW INVOICE	Ted	Crawford	Edmonton	7809753079	t.crawford@yahoo.com	$0.00
EDIT NEW INVOICE	John	Crawford	Edmonton	7805001646	j.crawford@gmx.com	$0.00
EDIT NEW INVOICE	Dominik	Crawford	Edmonton	7805954376	d.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Annabella	Crawford	Edmonton	7803822921	a.crawford@mail.com	$0.00
EDIT NEW INVOICE	Maria	Crawford	Edmonton	7807237348	m.crawford@aol.com	$0.00
EDIT NEW INVOICE	Dainton	Crawford	Edmonton	7809247015	d.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Jared	Crawford	Edmonton	7802284629	j.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Abigail	Crawford	Edmonton	7802089784	a.crawford@mail.com	$0.00

Rows per page: 10 1-10 of 43

Initial Code Behind

- Methods
  - Search (partial last name & or partial phone #)
  - New customer
  - Edit customer (selected customer)
  - New invoice (selected customer)

Initial Code Behind

- Fields for holding the search parameters
  - Last Name
  - Phone Number
  - Feedback message and flag that feedback exists
  - Error messages and flag that errors exist
  - List of error messages
- Properties
  - Customer service
  - Navigation Manager (navigate to customer edit page)
  - List of customer searches
  - Pagination

Fields

CustomerList.razor.cs

```
public partial class CustomerList
{
    #region Fields

    // The last name
    private string lastName = string.Empty;

    // The phone number
    private string phoneNumber = string.Empty;

    // Tells us if the search has been performed
    private bool noRecords;

    // The feedback message
    private string feedbackMessage = string.Empty;

    // The error message
    private string errorMessage = string.Empty;
```

CustomerList.razor.cs

Fields

```
// has feedback
1 reference | 0 changes | 0 authors, 0 changes
private bool hasFeedback => !string.IsNullOrWhiteSpace(feedbackMessage);

// has error
1 reference | 0 changes | 0 authors, 0 changes
private bool hasError => !string.IsNullOrWhiteSpace(errorMessage);

// error details
private List<string> errorDetails = new();
#endregion
```

Properties

CustomerList.razor.cs

```
#endregion

#region Properties
// Injects the CustomerService dependency.
[Inject]
1 reference | 0 changes | 0 authors, 0 changes
protected CustomerService CustomerService { get; set; } = default!;

// Injects the NavigationManager dependency.
[Inject]
2 references | 0 changes | 0 authors, 0 changes
protected NavigationManager NavigationManager { get; set; } = default!;

// Gets or sets the customers search view.
5 references | 0 changes | 0 authors, 0 changes
protected List<CustomerSearchView> Customers { get; set; } = new();
#endregion
```

CustomerList.razor.cs

Methods

NOTE: All methods liste

are a placeholder

```
#region Methods
//  search for an existing customer
1 reference | 0 changes | 0 authors, 0 changes
private void Search()
{

}

//  new customer
1 reference | 0 changes | 0 authors, 0 changes
private void New()...

//  edit selected customer
1 reference | 0 changes | 0 authors, 0 changes
private void EditCustomer(int customerID)...

//  new invoice for selected customer
1 reference | 0 changes | 0 authors, 0 changes
private void NewInvoice(int customerID)...

#endregion
```

Customer List - Search

Customers

Search for customer(s) was successful

Customer Search

Last Name: fo

Phone Number

SEARCH NEW

Actions	First Name	Last Name	City	Phone	Email	Email
EDIT NEW INVOICE	Ted	Crawford	Edmonton	7809753079	t.crawford@yahoo.com	$0.00
EDIT NEW INVOICE	John	Crawford	Edmonton	7805001646	j.crawford@gmx.com	$0.00
EDIT NEW INVOICE	Dominik	Crawford	Edmonton	7805954376	d.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Annabella	Crawford	Edmonton	7803822921	a.crawford@mail.com	$0.00
EDIT NEW INVOICE	Maria	Crawford	Edmonton	7807237348	m.crawford@aol.com	$0.00
EDIT NEW INVOICE	Dainton	Crawford	Edmonton	7809247015	d.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Jared	Crawford	Edmonton	7802284629	j.crawford@icloud.com	$0.00
EDIT NEW INVOICE	Abigail	Crawford	Edmonton	7802089784	a.crawford@mail.com	$0.00

Rows per page: 10 1-10 of 43

CustomerList.razor

Customer List - Errors

```
@page "/SamplePages/CustomerList"
<PageTitle>Customer List</PageTitle>
<MudText Typo="Typo.h3">Customers</MudText>

@if (hasError)
{
    <MudAlert Elevation="2"
              Severity="Severity.Error"
              Dense="true">
        <MudText Typo="Typo.h3">@errorMessage</MudText>
        @foreach (var error in errorDetails)
        {
            <MudText Typo="Typo.body2">@error</MudText>
        }
    </MudAlert>
}
```

CustomerList.razor

Customer List - Feedback

```
@if(hasFeedback)
{
    <MudAlert Elevation="2"
              Severity="Severity.Info"
              Dense="true">
        <MudText Typo="Typo.body2">@feedbackMessage</MudText>
    </MudAlert>
}
```

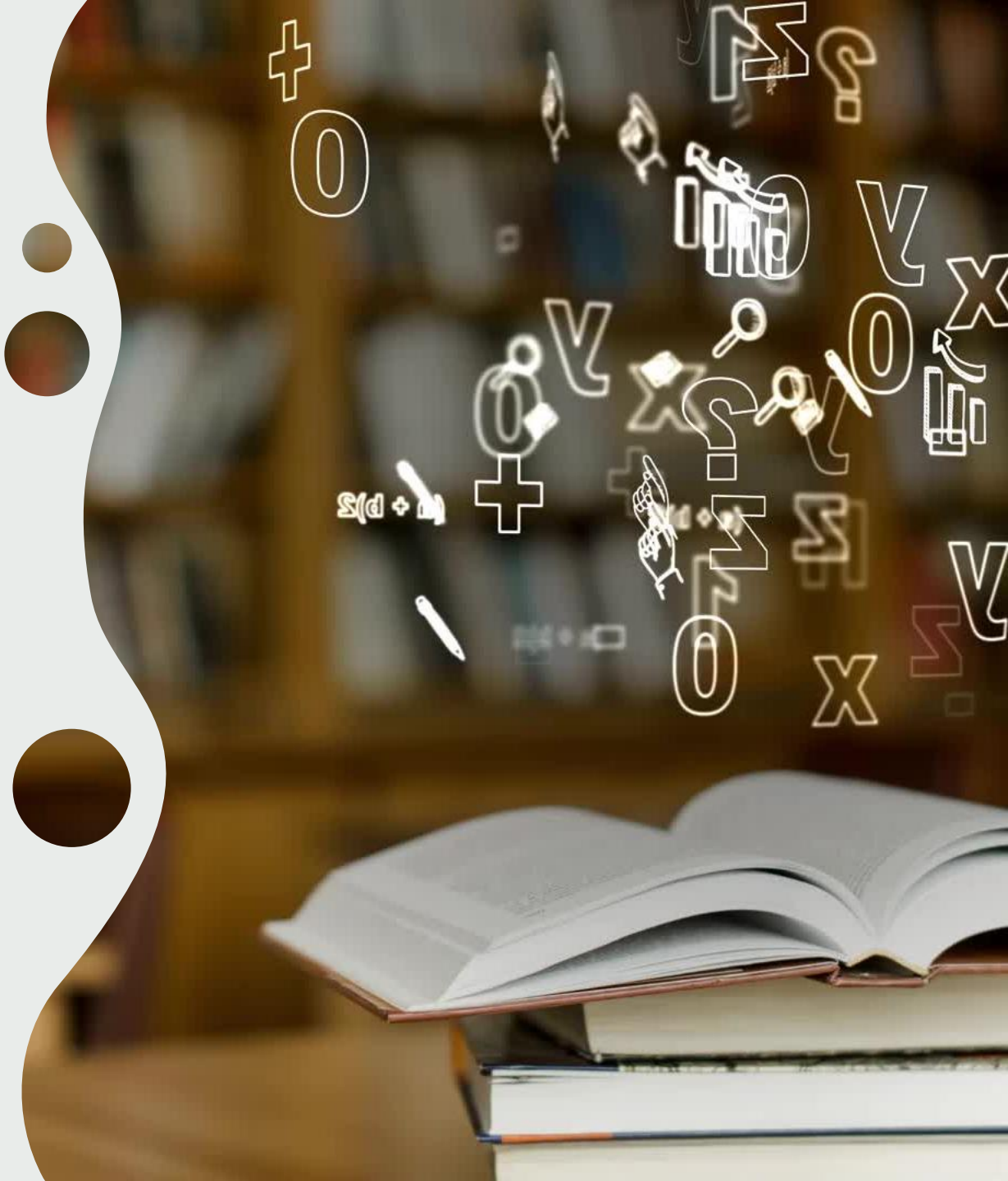
CustomerList.razor

Customer List - Search

```
<MudStack Row="true" Spacing="4" AlignItems="AlignItems.Center">
    <MudText Typo="Typo.h5">Customer Search</MudText>
    <MudTextField @bind-Value="lastName"
                  Label="Last Name"
                  Variant="Variant.Outlined" />
    <MudTextField @bind-Value="phoneNumber"
                  Label="Phone Number"
                  Variant="Variant.Outlined" />
    <MudButton Variant="Variant.Filled"
               Color="Color.Primary"
               OnClick="Search">
        Search
    </MudButton>
    <MudButton Variant="Variant.Filled"
               Color="Color.Success"
               OnClick="New">
        New
    </MudButton>
</MudStack>
```

Customer List - Search

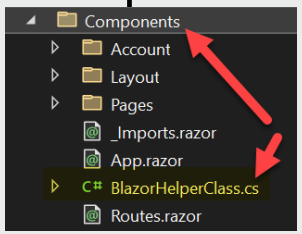
- For any method that calls the System Library (HogWildSystem), we must add a Try/Catch area to catch any errors that might be thrown from the library.
  - Clear the feedback and error messages.
  - Validate our rules.
- We need to create a new class called “BlazorHelperClass”. By creating a global “GetErrorMessages”, we will not have to add it to each of our pages repeatedly. Copy the code from the GetCustomers.linq -> “GetErrorMessages”



BlazorHelperClass.cs

Blazor Helper Class

- Add a new class under “Components” call “BlazorHelperClass”




```
using BYSResults;
namespace HogWildWeb.Components
{
    14 references
    public static class BlazorHelperClass
    {
        // Converts a list of error objects into their string representations.
        0 references
        public static List<string> GetErrorMessages(List<Error> errorMessage)
        {
            // Initialize a new list to hold the extracted error messages
            List<string> errorList = new();

            // Iterate over each Error object in the incoming list
            foreach (var error in errorMessage)
            {
                // Convert the current Error to its string form and add it to errorList
                errorList.Add(error.ToString());
            }

            // Return the populated list of error message strings
            return errorList;
        }
    }
}
```

Search Method

CustomerList.razor.cs

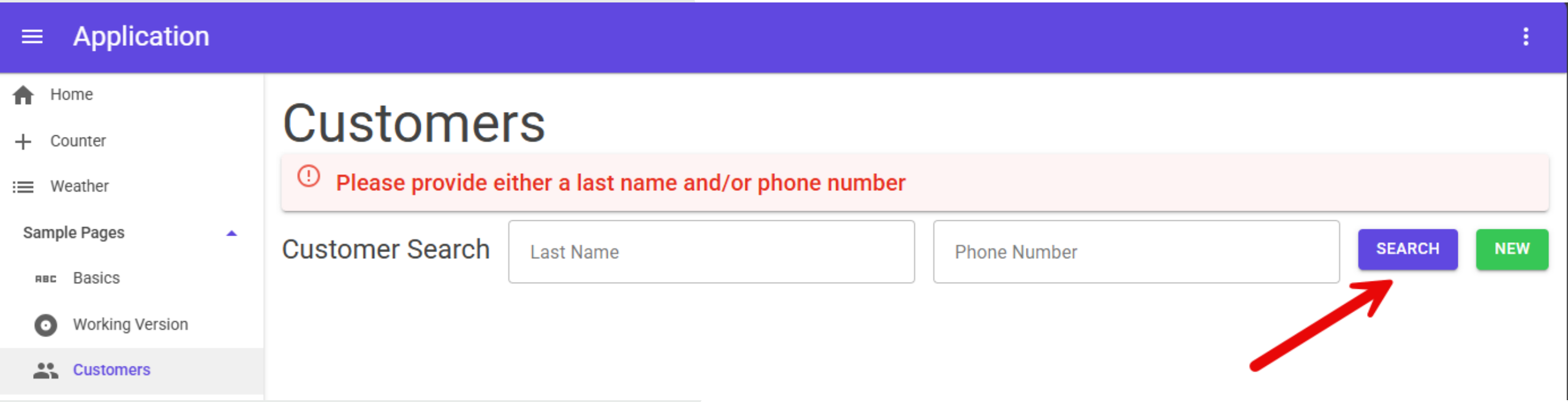
Copy the code from GetCustomers.linq within your CodeBehind -> GetCustomers

1. Refactor the method name from “GetCustomers” to “Search”
2. Add “noRecords”
3. Updated the “service class” name to “CustomerService”
4. Refactor parameter “phone” to “phoneNumber”
5. Set noRecords if no customers are found
6. Add BlazorHelperClass to the GetErrorMessages

```
#region Methods
//  search for an existing customer
1 reference
private void Search()
{
    // clear previous error details and messages
    noRecords = false;
    errorDetails.Clear();
    errorMessage = string.Empty;
    feedbackMessage = String.Empty;

    // wrap the service call in a try/catch to handle unexpected exceptions
    try
    {
        var result = CustomerService.GetCustomers(lastName, phoneNumber);
        if (result.IsSuccess)
        {
            Customers = result.Value;
        }
        else
        {
            // set noRecords
            if(result.Errors.Any(e => e.Code == "No Customers"))
            {
                noRecords=true;
            }
            errorDetails = BlazorHelperClass.GetErrorMessages(result.Errors.ToList());
        }
    }
    catch (Exception ex)
    {
        // capture any exception message for display
        errorMessage = ex.Message;
    }
}
#endregion
```

Blazor Page Output



Customer List - Table (MudBlazor Component)

Customers

Search for customer(s) was successful

Customer Search | Last Name: smi | Phone Number | SEARCH | NEW

Actions	First Name	Last Name	City	Phone	Email	Total Sales
EDIT NEW INVOICE	Bob	Smith	Edmonton	7804444444	ssmith@hotmail.com	$85,593.29
EDIT NEW INVOICE	Arnold	Smith	Edmonton	7806057965	a.smith@protonmail.com	
EDIT NEW INVOICE	Daisy	Smith	Edmonton	7808890984	d.smith@protonmail.com	
EDIT NEW INVOICE	Vanessa	Smith	Edmonton	7809391120	v.smith@gmail.com	
EDIT NEW INVOICE	Haris	Smith	Edmonton	7802186615	h.smith@runbox.com	
EDIT NEW INVOICE	Maximilian	Smith	Edmonton	7804751940	m.smith@hushmail.com	
EDIT NEW INVOICE	Fenton	Smith	Edmonton	7806756066	f.smith@hushmail.com	
EDIT NEW INVOICE	Lucy	Smith	Edmonton	7805202140	l.smith@mail.com	
EDIT NEW INVOICE	Andrew	Smith	Edmonton	7804156910	a.smith@tutanota.com	

Rows per page: 10 1-10 of 14

Customer List - Table

- The table component will display a list of customer search results if found.
  - If no results were found, a message will be presented to the user of zero customer(s) found.
- For each result found, two buttons will be shown.
  - “Edit” – Edit the selected customer,
  - “New Invoice” - Create a new invoice for the selected customer.

Customer List - Table

CustomerList.razor

```
@if(Customers.Count > 0)
{
    <MudDataGrid Items="Customers"
                 Striped="true"
                 FixedFooter="true"
                 FixedHeader="true"
                 Height="65vh">
        <Columns>
            <TemplateColumn >
                <HeaderTemplate>
                    Actions
                </HeaderTemplate>
                <CellTemplate>
                    <MudButton Variant="Variant.Filled"
                               Color="Color.Primary"
                               OnClick="() => EditCustomer(context.Item.CustomerID)">
                        Edit
                    </MudButton>
```

Customer List - Table

CustomerList.razor

```
        <MudButton Variant="Variant.Filled"
                   Color="Color.Secondary"
                   OnClick="() => NewInvoice(context.Item.CustomerID)"
                   Class="ml-2">
            New Invoice
        </MudButton>
    </CellTemplate>
</TemplateColumn>
```

Customer List - Table

CustomerList.razor

```
    <PropertyColumn Property="x => x.FirstName" Title="First Name" />
    <PropertyColumn Property="x => x.LastName" Title="Last Name" />
    <PropertyColumn Property="x => x.City" Title="City" />
    <PropertyColumn Property="x => x.Phone" Title="Phone" />
    <PropertyColumn Property="x => x.Email" Title="Email" />
    <PropertyColumn Property="@(x => x.TotalSales.HasValue
                                    ? x.TotalSales.Value.ToString("C2")
                                    : string.Empty)"
                    Title="Total Sales" />
</Columns>
```

Customer List - Table

CustomerList.razor

```
    <NoRecordsContent>
        <MudText Typo="Typo.h6">
            @((noRecords ? "No customers found."
            : "Please search for customers."))
        </MudText>
    </NoRecordsContent>
    <PagerContent>
        <MudDataGridPager />
    </PagerContent>
</MudDataGrid>
```

Search Method – CustomerService.GetCustomer

- Call the “Customer Service.GetCustomers()” with the last name” and or phone #.
  - Update the Customers list with the results.
  - Update feedback message whether customer were found or not.

```
var result = CustomerService.GetCustomers(lastName, phoneNumber);
if (result.IsSuccess)
{
    Customers = result.Value;
}
else
{
    // set noRecords
    if(result.Errors.Any(e => e.Code == "No Customers"))
    {
        noRecords=true;
    }
    errorDetails = BlazorHelperClass.GetErrorMessages(result.Errors.ToList());
}
```

CustomerList.razor.cs

Blazor Page Output

Customers

Search for customer(s) was successful

Customer Search

Last Name: smi

Phone Number

SEARCH NEW

Actions	First Name	Last Name	City	Phone	Email	Total Sales
EDIT NEW INVOICE	Bob	Smith	Edmonton	7804444444	ssmith@hotmail.com	$85,593.29
EDIT NEW INVOICE	Arnold	Smith	Edmonton	7806057965	a.smith@protonmail.com	
EDIT NEW INVOICE	Daisy	Smith	Edmonton	7808890984	d.smith@protonmail.com	
EDIT NEW INVOICE	Vanessa	Smith	Edmonton	7809391120	v.smith@gmail.com	
EDIT NEW INVOICE	Haris	Smith	Edmonton	7802186615	h.smith@runbox.com	
EDIT NEW INVOICE	Maximilian	Smith	Edmonton	7804751940	m.smith@hushmail.com	
EDIT NEW INVOICE	Fenton	Smith	Edmonton	7806756066	f.smith@hushmail.com	
EDIT NEW INVOICE	Lucy	Smith	Edmonton	7805202140	l.smith@mail.com	
EDIT NEW INVOICE	Andrew	Smith	Edmonton	7804156910	a.smith@tutanota.com	

Rows per page: 10 1-10 of 14

Statement Regarding Slide Accuracy and Potential Revisions

Please note that the content of these PowerPoint slides is accurate to the best of my knowledge at the time of presentation. However, as new research and developments emerge, or to enhance the learning experience, these slides may be subject to updates or revisions in the future. I encourage you to stay engaged with the course materials and any announcements for the most current information