Sự kiện truyền - lấy dữ liệu không dùng SignalR

by Luu Gia Thanh - 2023

Thuộc tính `OnCommentAdded` được khai báo là một `EventCallback`. Cho phép thành phần nhận một sự kiện từ bên ngoài khi có một comment được thêm vào.

Khi 1 comment được thêm vào và muốn kích hoạt sự kiện `OnCommentAdded`, sử dụng `await OnCommentAdded.InvokeAsync();` ở phương thức sẽ thực hiện. Điều này gọi phương thức `InvokeAsync` trên `OnCommentAdded` và đợi cho phương thức đó được thực thi.

Ví dụ: Có component Comment và Activity. Khi nhấn button Comment của component Comment là sẽ load lại được danh sách Comment hiện trên Activity

Trang component Comment

```
        //Truyen su kien
        [Parameter]
        public EventCallback OnCommentAdded
{ get; set; }

        public async Task CreateCommentAsyn
c()
        {
            try
            {
                savedContent = await richTe
xtEditRef.GetHTML();
                if (MentionedUsers.Count()
> 0)
                {
                    foreach (var mentionId
in MentionedUsers)
                    {
                        var mentionedUser =
```

Trang component Activity

```
public async Task GetCommentListAsync()
        {
            FilterCommentList.MaxResultCoun
t = PageSize;
            FilterCommentList.SkipCount =
(CurrentPage - 1) * PageSize;

            var result = await CommentsAppS
ervice.GetListAsync(FilterCommentList);
            commentsList = result.Items.Ord
erByDescending(m => m.CreationTime).ToList
();
            TotalCount = (int)result.TotalC
ount;
            StateHasChanged();
        }
```

Trang Test:

```
UserList.FirstOrDefault(x => x.Id == (Guid.
Parse(mentionId)));
                        if (mentionedUser !
= null)
                        {
                            // Access the m
entioned user's properties here
                            mentionedUserId
= mentionedUser.Id;
                            mentionedUserNa
me = mentionedUser.Name;
                            mentionedUserEm
ail = mentionedUser.Email;
                        }
                        _message.FromUserId
= (Guid)CurrentUser.Id;
                        _message.ToUserId =
mentionedUserId;
                        _message.DocId = Do
cId;
                        _message.Url = Url;

                        if (await CheckNul
l.ValidateAll() == false)
                        {
                            return;
                        }

                        else if (_message.T
oUserId != Guid.Empty)
                        {
                            _message.NotiTi
tle = CurrentUser.Name + " " + "mentioned y
ou";
                            _message.NotiBo
dy = "Comment Body";
                            _message.Type =
"M";
                            await Notificat
ionsAppService.CreateAsync(_message);
                            await SendTestE
mailAsync();
                        }
                    }
                    _comment.FromUserId =
(Guid)CurrentUser.Id;
                    _comment.Content = save
dContent;
                    _comment.DocId = DocId;
                    _comment.Url = Url;
                    await CommentsAppServic
e.CreateAsync(_comment);
                    await Notify.Success(L
["SuccessfullySent"]);
                    await richTextEditRef.C
learAsync();
                    await OnCommentAdded.In
vokeAsync();
```

```
@page "/Common/TestRichEdit"

<Div>
    <HQSOFTComment OnCommentAdded="HandleCo
mmentAdded" DocId='@EditingTestCommonId' Ur
l='/Common/TestRichEdit' />
</Div>

<Div>
    <HQSOFTFormActivity DocId='@EditingTest
CommonId' Url='/Common/TestRichEdit' @ref
="@formActivity" />
</Div>

@code {

    private Guid EditingTestCommonId { get;
set; } = Guid.Empty;

    private HQSOFTFormActivity formActivit
y;

    private async Task HandleCommentAdded()
    {
        await formActivity.GetCommentListAs
ync();
    }
}
```

```
                }
                else
                {
                    _comment.FromUserId =
(Guid)CurrentUser.Id;
                    _comment.Content = save
dContent;
                    _comment.DocId = DocId;
                    _comment.Url = Url;
                    await CommentsAppServic
e.CreateAsync(_comment);
                    await Notify.Success(L
["SuccessfullySent"]);
                    await richTextEditRef.C
learAsync();
          await OnCommentAdded.InvokeAsync
();
          }
            catch (Exception ex)
            {
                await HandleErrorAsync(ex);
            }
        }
```

Code History:

```
Razor:

<DxTabPage Text="@L["History"]">
            <MudTimeline TimelinePosition
="TimelinePosition.Start">
                @foreach (var entityChange
in entityChangesList)
                {
                    var urlEntity = urlEnti
tyList[entityChangesList.IndexOf(entityChan
ge)];
                    var auditLogs = auditLo
gsList.FirstOrDefault(a => a.Id == entityCh
ange.AuditLogId);

                    if (Guid.Parse(entityCh
ange.EntityId) == DocId || (Guid.Parse(enti
tyChange.EntityId) == DocId && urlEntity ==
Url))
                    {
                        if (entityChange.Ch
angeType == Volo.Abp.Auditing.EntityChangeT
ype.Created && auditLogsList.Count() > 0)
                        {
                            var fromUserPOS
T = UserList.FirstOrDefault(u => u.Id == au
ditLogs.UserId);
                            <MudTimelineIte
m Color="MudBlazor.Color.Info">
```

```
private string desiredUrl = "";
        private List<string> urlEntityList
= new List<string>();
        private List<string> MentionedUserI
ds { get; set; } = new List<string>();
        private List<AuditLogDto> auditLogs
List { get; set; } = new List<AuditLogDto>
();
        private List<EntityChangeDto> entit
yChangesList { get; set; } = new List<Entit
yChangeDto>();
        private IReadOnlyList<EntityChangeD
to> entityChangeList { get; set; } = new Li
st<EntityChangeDto>();


protected override async Task OnInitialized
Async()
        {
            await GetEntityChangeAsync();
            await GetHistoryListAsync();
            StateHasChanged();
            await base.OnInitializedAsync
();
        }

public async Task GetEntityChangeAsync()
        {
            var entityChanges = await Audit
```

```
                                    <ItemDot>
                                        <MudAva
tar Size="MudBlazor.Size.Medium" Image="@(a
uthServerUrl + "/api/account/profile-pictur
e-file/" + auditLogs.UserId.ToString())" />
                                    </ItemDot>
                                    <ItemConten
t>
                                        <MudTex
t Typo="Typo.body2">
                                            @if
(CurrentUser.Id == auditLogs.UserId)
                                            {

<b style="font-weight: 500;">You</b>
                                            }
                                            els
e if (fromUserPOST != null)
                                            {

<b style="font-weight: 500;">@fromUserPOST.
Name</b>
                                            }
                                            els
e
                                            {

<span style="color: red;">Unknown User</spa
n>
                                            }
                                            cre
ated this. - @GetTimeAgo(entityChange.Chang
eTime)
                                        </MudTe
xt>
                                    </ItemConte
nt>
                                </MudTimelineIt
em>
                            }
                            else if (entityChan
ge.ChangeType == Volo.Abp.Auditing.EntityCh
angeType.Updated && auditLogsList.Count() >
0)
                            {
                                var fromUserPUT
= UserList.FirstOrDefault(u => u.Id == audi
tLogs.UserId);
                                foreach (var hi
story in entityChange.PropertyChanges)
                                {
                                    <MudTimelin
eItem Color="MudBlazor.Color.Info">
                                        <ItemDo
t>
                                            <Mu
dAvatar Size="MudBlazor.Size.Medium" Image
="@(authServerUrl + "/api/account/profile-p
```

```
LogsAppService.GetEntityChangesAsync(new Ge
tEntityChangesDto
            {
                MaxResultCount = LimitedRes
ultRequestDto.MaxMaxResultCount,
                EntityId = DocId.ToString()
            });
            entityChangeList = entityChange
s.Items;
            StateHasChanged();
        }

public async Task GetHistoryListAsync()
        {
            var tempEntityChangesList = new
List<EntityChangeDto>();
            var tempUrlList = new List<stri
ng>();
            auditLogsList.Clear();
            entityChangesList.Clear();
            urlEntityList.Clear();

            foreach (var entityChange in en
tityChangeList.Where(x =>
                (x.EntityId == DocId.ToStri
ng() && x.ChangeType == EntityChangeType.Cr
eated) ||
                (x.EntityId == DocId.ToStri
ng() && x.ChangeType == EntityChangeType.Up
dated)))
            {
                var auditLog = await AuditL
ogsAppService.GetAsync(entityChange.AuditLo
gId);

                var extraProperties = audit
Log.ExtraProperties;
                var docUrl = extraPropertie
s[UrlName];
                string desiredUrl = GetDesi
redUrl(docUrl.ToString());

                auditLogsList.Add(auditLo
g);
                entityChangesList.Add(entit
yChange);
                urlEntityList.Add(desiredUr
l);
            }
            StateHasChanged();
        }
```

```
icture-file/" + auditLogs.UserId.ToString
())" />
                                        </ItemD
ot>
                                        <ItemCo
ntent>
                                            <Mu
dText Typo="Typo.body2">

@if (CurrentUser.Id == auditLogs.UserId)

{

<b style="font-weight: 500;">You</b>

}

else if (fromUserPUT != null)

{

<b style="font-weight: 500;">@fromUserPUT.N
ame</b>

}

else

{

<span style="color: red;">Unknown User</spa
n>

}

changed value of @history.PropertyName

from <b style="font-weight: 500;">@(histor
y.OriginalValue == "null" ? "\"\"" : Trunca
teText(history.OriginalValue, 30))</b>

to <b style="font-weight: 500;">@(history.N
ewValue == null ? "\"\"" : TruncateText(his
tory.NewValue, 30))</b> in @GetLastSegmentA
fterDot(entityChange.EntityTypeFullName) -
@GetTimeAgo(entityChange.ChangeTime)
                                            </M
udText>
                                    </ItemC
ontent>
                                </MudTimeli
neItem>
                            }
                        }
                    }
                }
            </MudTimeline>
```

Dùng component History ở trang khác:

```
<HQSOFTFormActivity DocId="@EditingCustomer
Id" Url="/AccountReceivable/Customers"
@ref="@formActivity" />


[Parameter]
public string Id { get; set; }
private Guid EditingCustomerId { get; set;
}

private HQSOFTFormActivity formActivity;

protected override async Task OnInitialized
Async()
        {
            PanelVisible = true;

            EditingCustomerId = Guid.Parse
(Id);
            .....
            await LoadDataAsync(EditingCust
omerId);
            await HandleHistoryAdded();
            await InvokeAsync(StateHasChang
ed);

            PanelVisible = false;
        }

protected virtual ValueTask SetToolbarItems
Async()
{
    Toolbar.AddButton(L["Save"], async () =
>
    {
        await SaveClassesAsync(false);
    },
      IconName.Save,
      Color.Primary,
       requiredPolicyName: CoreBackendPermi
ssions.CustomerClasses.Edit);

    return ValueTask.CompletedTask;
}


private async Task HandleHistoryAdded()
        {
            await InvokeAsync(StateHasChang
ed);
            await formActivity.GetEntityCha
ngeAsync();
            await formActivity.GetHistoryLi
```

```
stAsync();
        }


private async Task SaveClassesAsync(bool Is
NewNext)
        {
            try
            {
                if (!EditFormMain.EditConte
xt.Validate())
                {
                    return;
                }

                if (EditingCustomerClassId
== Guid.Empty)
                {
                    if (!codeExists)
                    {
                        ....
                        await HandleHistory
Added();
                    }
                    else
                    {
                        await _uiMessageSer
vice.Error(L["Notification:CodeAlreadyExist
s"]);
                    }
                }
                else if (EditingCustomerCla
ssId != Guid.Empty && IsDataEntryChanged ==
true)
                {
                        ....
                        await HandleHistory
Added();
                    }
                    else
                    {
                        await _uiMessageSer
vice.Error(L["Notification:CodeAlreadyExist
s"]);
                    }
                }

                if (IsNewNext)
                {
                    await NewClass();
                    ResetToolbar();
                }
                else
                {
                    IsDataEntryChanged = fa
lse;
                    NavigationManager.Navig
ateTo($"/AccountReceivable/CustomerClasses/
```

```
{EditingCustomerClassId}");
                }
            }
            catch (Exception ex)
            {
                await HandleErrorAsync(ex);
            }
        }
```

Code History nhiều EntityId:

```
Razor:

<DxTabPage Text="@L["History"]">
    <MudTimeline TimelinePosition="Timeline
Position.Start">
        @foreach (var entityChange in entit
yChangesList.OrderByDescending(e => e.Chang
eTime))
        {
            var urlEntity = urlEntityList[e
ntityChangesList.IndexOf(entityChange)];
            var auditLogs = auditLogsList.F
irstOrDefault(a => a.Id == entityChange.Aud
itLogId);

            // Kiểm tra giá trị của entityC
hange.EntityId
            if (entityChange.EntityId == Do
cIds[0].ToString())
            {
                // Thực hiện code cho ID đầ
u tiên
                if (entityChange.ChangeType
== Volo.Abp.Auditing.EntityChangeType.Creat
ed && auditLogsList.Count() > 0)
                {
                    var fromUserPOST = User
List.FirstOrDefault(u => u.Id == auditLogs.
UserId);
                    <MudTimelineItem Color
="MudBlazor.Color.Info">
                        <ItemDot>
                            <MudAvatar Size
="MudBlazor.Size.Medium" Image="@(authServe
rUrl + "/api/account/profile-picture-file/"
+ auditLogs.UserId.ToString())" />
                        </ItemDot>
                        <ItemContent>
                            <MudText Typo
="Typo.body2">
                                @if (Curren
tUser.Id == auditLogs.UserId)
                                {
                                    <b styl
e="font-weight: 500;">You</b>
```

```
private string desiredUrl = "";
private List<string> urlEntityList = new Li
st<string>();
private List<string> MentionedUserIds { ge
t; set; } = new List<string>();
private List<AuditLogDto> auditLogsList { g
et; set; } = new List<AuditLogDto>();
private List<EntityChangeDto> entityChanges
List { get; set; } = new List<EntityChangeD
to>();
private IReadOnlyList<EntityChangeDto> enti
tyChangeList { get; set; } = new List<Entit
yChangeDto>();

[Parameter]
public List<Guid> DocIds { get; set; } = ne
w List<Guid>();


protected override async Task OnInitialized
Async()
        {
            await GetEntityChangeAsync();
            await GetHistoryListAsync();
            StateHasChanged();
            await base.OnInitializedAsync
();
        }

        public async Task GetEntityChangeAs
ync()
        {
            var tempEntityChangesList = new
List<EntityChangeDto>();

            foreach (var docId in DocIds)
            {
                var entityChanges = await A
uditLogsAppService.GetEntityChangesAsync(ne
w GetEntityChangesDto
                {
                    MaxResultCount = Limite
dResultRequestDto.MaxMaxResultCount,
                    EntityId = docId.ToStri
ng()
```

```
                                }
                                else if (fr
omUserPOST != null)
                                {
                                    <b styl
e="font-weight: 500;">@fromUserPOST.Name</b
>
                                }
                                else
                                {
                                    <span s
tyle="color: red;">Unknown User</span>
                                }
                                created thi
s. - @GetTimeAgo(entityChange.ChangeTime)
                            </MudText>
                        </ItemContent>
                    </MudTimelineItem>
                }
                else if (entityChange.Chang
eType == Volo.Abp.Auditing.EntityChangeTyp
e.Updated && auditLogsList.Count() > 0)
                {
                    var fromUserPUT = UserL
ist.FirstOrDefault(u => u.Id == auditLogs.U
serId);
                    foreach (var history in
entityChange.PropertyChanges)
                    {
                        <MudTimelineItem Co
lor="MudBlazor.Color.Info">
                            <ItemDot>
                                <MudAvatar
Size="MudBlazor.Size.Medium" Image="@(authS
erverUrl + "/api/account/profile-picture-fi
le/" + auditLogs.UserId.ToString())" />
                            </ItemDot>
                            <ItemContent>
                                <MudText Ty
po="Typo.body2">
                                @if (Cu
rrentUser.Id == auditLogs.UserId)
                                {
                                    <b
style="font-weight: 500;">You</b>
                                }
                                else if
(fromUserPUT != null)
                                {
                                    <b
style="font-weight: 500;">@fromUserPUT.Name
</b>
                                }
                                else
                                {
                                    <sp
an style="color: red;">Unknown User</span>
                                }
```

```
                });

                tempEntityChangesList.AddRa
nge(entityChanges.Items); // Thêm tất cả cá
c phần tử vào danh sách
            }
            entityChangeList = tempEntityCh
angesList;

            Console.WriteLine("entityChange
List.Count(): " + entityChangeList.Count
());
            StateHasChanged();
        }


        public async Task GetHistoryListAsy
nc()
        {
            var tempEntityChangesList = new
List<EntityChangeDto>();
            var tempUrlList = new List<stri
ng>();
            auditLogsList.Clear();
            entityChangesList.Clear();
            urlEntityList.Clear();

            foreach (var docId in DocIds)
            {
                foreach (var entityChange i
n entityChangeList.Where(x =>
                (x.EntityId == docId.ToStri
ng() && x.ChangeType == EntityChangeType.Cr
eated) ||
                (x.EntityId == docId.ToStri
ng() && x.ChangeType == EntityChangeType.Up
dated)))
                {
                    var auditLog = await Au
ditLogsAppService.GetAsync(entityChange.Aud
itLogId);

                    var extraProperties = a
uditLog.ExtraProperties;
                    var docUrl = extraPrope
rties[UrlName];
                    string desiredUrl = Get
DesiredUrl(docUrl.ToString());

                    auditLogsList.Add(audit
Log);
                    entityChangesList.Add(e
ntityChange);
                    urlEntityList.Add(desir
edUrl);
                }
            }
            StateHasChanged();
```

```
                                    changed
value of @history.PropertyName
                                    from <b
style="font-weight: 500;">@(history.Origina
lValue == "null" ? "\"\"" : TruncateText(hi
story.OriginalValue, 30))</b>
                                    to <b s
tyle="font-weight: 500;">@(history.NewValue
== null ? "\"\"" : TruncateText(history.New
Value, 30))</b> in @GetLastSegmentAfterDot
(entityChange.EntityTypeFullName) - @GetTim
eAgo(entityChange.ChangeTime)
                                </MudText>
                            </ItemContent>
                        </MudTimelineItem>
                    }
                }
            }
            else
            {
                // Thực hiện code cho ID đầ
u tiên
                if (entityChange.ChangeType
== Volo.Abp.Auditing.EntityChangeType.Creat
ed && auditLogsList.Count() > 0)
                {
                    var fromUserPOST = User
List.FirstOrDefault(u => u.Id == auditLogs.
UserId);
                    <MudTimelineItem Color
="MudBlazor.Color.Info">
                        <ItemDot>
                            <MudAvatar Size
="MudBlazor.Size.Medium" Image="@(authServe
rUrl + "/api/account/profile-picture-file/"
+ auditLogs.UserId.ToString())" />
                        </ItemDot>
                        <ItemContent>
                            <MudText Typo
="Typo.body2">
                                @if (Curren
tUser.Id == auditLogs.UserId)
                                {
                                    <b styl
e="font-weight: 500;">You</b>
                                }
                                else if (fr
omUserPOST != null)
                                {
                                    <b styl
e="font-weight: 500;">@fromUserPOST.Name</b
>
                                }
                                else
                                {
                                    <span s
tyle="color: red;">Unknown User</span>
                                }
```

```
        }

public string AddSpaceToCamelCase(string in
put)
        {
            string output = Regex.Replace(i
nput, "(?<!^)([A-Z])", " $1");
            return output;
        }

        public string GetLastSegmentAfterDo
t(string data)
        {
            string[] segments = data.Split
('.');
            string lastSegment = segments[^
1];
            string formattedSegment = AddSp
aceToCamelCase(lastSegment);
            return formattedSegment;
        }

#region Truncate Text
        public static string TruncateText(s
tring text, int maxLength) // Cắt chuỗi
        {
            if (text.Length <= maxLength)
                return text;

            return text.Substring(0, maxLen
gth) + "...";
        }
        #endregion

        #region Get Uri
        public string GetDesiredUrl(string
inputUrl)
        {
            string trimmedData = inputUrl.T
rim('[', ']');
            string cleanedData = trimmedDat
a.Replace("\\", "").Replace("\"", ""); // L
oại bỏ ký tự "\", "\" và ký tự nháy kép
"\""
            string url = cleanedData;// Lấy
URL từ chuỗi đã được làm sạch
            string[] parts = url.Split
('/'); // Tách chuỗi URL thành các phần tử
            if (parts.Length >= 4) // Kiểm
tra đủ phần tử để tạo URL
            {
                // Lấy phần "/AccountReceiv
able/CustomerClasses/" từ phần tử thứ 3 và
thứ 4
                desiredUrl = "/" + parts[3]
+ "/" + parts[4] + "/";
                // Cắt bỏ ký tự "/" cuối cù
ng
```

```
                                    added rows
for  @GetLastSegmentAfterDot(entityChange.E
ntityTypeFullName). - @GetTimeAgo(entityCha
nge.ChangeTime)
                            </MudText>
                        </ItemContent>
                    </MudTimelineItem>
                }
            }

        }
    </MudTimeline>
</DxTabPage>
```

Dùng component History ở trang khác:

```
<HQSOFTFormActivity DocIds="@DocIds" Url="/
AccountReceivable/Customers" @ref="@formAct
ivity" />

[Parameter]
public string Id { get; set; }
private Guid EditingCustomerId { get; set;
}
public List<Guid> DocIds { get; set; } = ne
w List<Guid>();

private HQSOFTFormActivity formActivity;

private List<CustomerAttributeUpdateDto> Cu
stomerAttributesList { get; set; } = new Li
st<CustomerAttributeUpdateDto>();

protected override async Task OnInitialized
Async()
        {
            PanelVisible = true;

            EditingCustomerId = Guid.Parse
(Id);
            .....
            await LoadDataAsync(EditingCust
omerId);
            await HandleHistoryAdded();
            await InvokeAsync(StateHasChang
ed);

            PanelVisible = false;
        }

protected virtual ValueTask SetToolbarItems
Async()
{
    Toolbar.AddButton(L["Save"], async () =
>
```

```
                desiredUrl = desiredUrl.Tri
mEnd('/');
            }

            return desiredUrl;
        }

        public List<string> GetDesiredUrls
(string inputUrl)
        {
            List<string> desiredUrls = new
List<string>();

            string[] urls = inputUrl.Split
(',');

            foreach (string url in urls)
            {
                string trimmedData = url.Tr
im('[', ']');
                string cleanedData = trimme
dData.Replace("\\", "").Replace("\"", "");
                string singleUrl = cleanedD
ata;

                string[] parts = singleUrl.
Split('/');

                if (parts.Length >= 4)
                {
                    string desiredUrl = "/"
+ parts[3] + "/" + parts[4] + "/";
                    desiredUrl = desiredUr
l.TrimEnd('/');
                    desiredUrls.Add(desired
Url);
                }
                else
                {
                    Console.WriteLine("URL
không hợp lệ: " + singleUrl);
                }
            }

            return desiredUrls;
        }
        #endregion
```

```
    {
        await SaveClassesAsync(false);
    },
      IconName.Save,
      Color.Primary,
       requiredPolicyName: CoreBackendPermi
ssions.CustomerClasses.Edit);

    return ValueTask.CompletedTask;
}


private async Task HandleHistoryAdded()
{
           DocIds.Clear();

           List<Guid> customerAttributeIds
= CustomerAttributesList.Select(c => c.Id).
ToList();
           List<Guid> customerCompanyIds =
CustomerCompaniesList.Select(c => c.Id).ToL
ist();
           List<Guid> customerSaleAndDeliv
eryIds = SaleAndDeliveriesList.Select(c =>
c.Id).ToList();
           List<Guid> customerPaymentMetho
dIds = CustomerPaymentMethodsList.Select(c
=> c.Id).ToList();
           List<Guid> customerPaymentMetho
dDetailIds = CustomerPaymentMethodDetailsLi
st.Select(c => c.Id).ToList();

           DocIds = new List<Guid> { Editi
ngCustomerId };
           DocIds.AddRange(customerAttribu
teIds);
           DocIds.AddRange(customerCompany
Ids);
           DocIds.AddRange(customerSaleAnd
DeliveryIds);
           DocIds.AddRange(customerPayment
MethodIds);
           DocIds.AddRange(customerPayment
MethodDetailIds);

           await InvokeAsync(StateHasChang
ed);
           await formActivity.GetEntityCha
ngeAsync();
           await formActivity.GetHistoryLi
stAsync();
}


private async Task SaveClassesAsync(bool Is
NewNext)
       {
           try
```

```
            {
                if (!EditFormMain.EditConte
xt.Validate())
                {
                    return;
                }

                if (EditingCustomerClassId
== Guid.Empty)
                {
                    if (!codeExists)
                    {
                        ....
                        await HandleHistory
Added();
                    }
                    else
                    {
                        await _uiMessageSer
vice.Error(L["Notification:CodeAlreadyExist
s"]);
                    }
                }
                else if (EditingCustomerCla
ssId != Guid.Empty && IsDataEntryChanged ==
true)
                {
                        ....
                        await HandleHistory
Added();
                    }
                    else
                    {
                        await _uiMessageSer
vice.Error(L["Notification:CodeAlreadyExist
s"]);
                    }
                }

                if (IsNewNext)
                {
                    await NewClass();
                    ResetToolbar();
                }
                else
                {
                    IsDataEntryChanged = fa
lse;
                    NavigationManager.Navig
ateTo($"/AccountReceivable/CustomerClasses/
{EditingCustomerClassId}");
                }
            }
            catch (Exception ex)
            {
                await HandleErrorAsync(ex);
            }
        }
```

```
private async Task GetCustomerAttributesAsy
nc()
{
        var result = await CustomerAttribut
esAppService.GetListNoPagedAsync(new GetCus
tomerAttributesInput
        {
                CustomerId = EditingCustome
rId,
                MaxResultCount = MaxCount,
        });
        CustomerAttributesList = ObjectMapp
er.Map<List<CustomerAttributeDto>, List<Cus
tomerAttributeUpdateDto>>((List<CustomerAtt
ributeDto>)result);
}
```