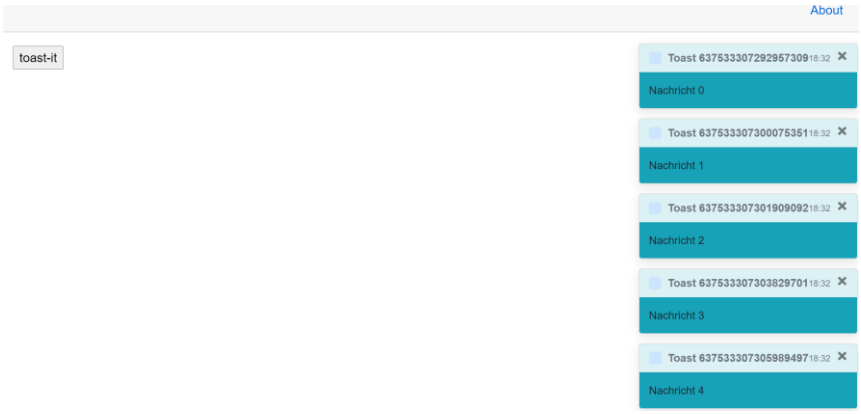
Blazor Lab 6 Toaster

Dauer ca 30 Minuten

In diesem Lab werden Toast Popup Nachrichten angezeigt, wie man das von einer Windows Notification kennt. Eine Liste von ToastItems wird mit ToastPopup befüllt. Nach einer gewissen Zeit wird per Timer das Objekt von der Liste eigenständig entfernt.



Starten Sie Visual Studio und legen ein neues **Blazor Web App** Projekt mit dem Namen BlazorLab6 an.

Im Verzeichnis Data fügen Sie zwei Klassen hinzu. Die erste als Objekt Container und die zweite als ViewModel, mit einer Liste und Funktionen zum hinzufügen und entfernen. Die Besonderheit ist, das das VM die Eigenschaft hat über Veränderungen (neu oder gelöscht) referenzierende Pages zu benachrichtigen.

ToastItem.cs

```
public class ToastItem
    {
        public string Titel { get; set; }
        public string Text { get; set; }
    }
```

ToasterVM.cs

```
public class ToasterVM
    {
    public event Action OnToastsUpdated;
    public ToasterVM()
    {
        Toasts = new List<ToastItem>();
    }
    public void Add(ToastItem _item)
    {

        Toasts.Add(_item);

        OnToastsUpdated?.Invoke();
    }
    public void Remove(ToastItem _item)
    {
        Toasts.Remove(_item);

        OnToastsUpdated?.Invoke();
```

```
    }

    public List<ToastItem> Toasts { get; set; }
```

Dieses ViewModel muss natürlich in dem DI Container bekannt gemacht werden. Dazu öffnen Sie die Datei programm.cs aus dem Projekt Stammverzeichnis und ergänzen sie folgende Zeile in der Methode ConfigureServices

```
builder.Services.AddScoped<ToasterVM>();
```

Für die Darstellung der Popup Nachrichten wird ein HTML Container benötigt der als Blazor Komponente in eine eigen Datei gelegt wird. Dieser ist mit dem VM verküpft und zeigt per foreach alle Einträge an.

Fügen Sie im Pages Verzeichnis dem Projekt ein Blazor Komponente mit dem Namen ToastContainer.razor hinzu und ergänzen Sie den Code.

```
@using Data
<div id="toastcontainer" style="position: absolute; top: 50px; right: 0;">

    @foreach (var toast in _vm.Toasts)
    {
    <div>@toast.Titel</div>
    }
</div>
@code {
    [Inject]
    public ToasterVM _vm { get; set; }

    protected override void OnInitialized()
    {
        _vm.OnToastsUpdated += () => InvokeAsync(StateHasChanged);
    }
}
```

Da die Änderungen im HTML DOM im Hintergrund passieren, muss der Aufwand über die event Notification OnToastsUpdated betrieben werden. Um im UI Thread zu landen wird InvokeAsync genutzt. Erst so kann der Renderer den DOM Ausschnitt neu rendern.

Der HTML Container von eben wird außerhalb der normalen HTML Struktur angelegt, damit von jeder Blazor Seite oder Event Code erreichbar. Dazu öffnen Sie die Datei App.razor aus dem Projekt Stammverzeichnis und fügen ganz am Ende den HTML Code für die Komponente ein.

```
<BlazorLab6.Pages.ToastContainer></BlazorLab6.Pages.ToastContainer>
```

Nun wird eine Page angelegt um neue Toast Popup Nachrichten anzulegen. Öffnen Sie dazu die Blazor Page Index.razor. Ändern Sie den Code so das diese aussieht wie folgt.

```
@page "/"

<div>
    <input  type="button" value="toast-it"  @onclick="Show"  />

</div>
@code {
    [Inject]
```

```
    public BlazorLab6.Data.ToasterVM _vm { get; set; }
    public void Show()
    {

        _vm.Add(new Data.ToastItem()
        {
            Titel = "Toast " + DateTime.Now.Ticks,
            Text = "Nachricht " + _vm.Toasts.Count().ToString()
        });
    }
}
```

Starten Sie das Projekt und drücken im Browser mehrmals den Button Toast-IT. Noch ist das Lab nicht abgeschlossen.

Blazor Toast Popup Komponente

Fügen Sie im Pages Verzeichnis eine neue Blazor Komponente mit dem Namen ToastPopup.Razor. Der HTML Teil ist dem Bootstrap Framework entnommen. Über den Timer können Sie die Zeit in ms einstellen (30 Sekunden), nachdem das Objekt sich selbst aus der Liste entfernt.

```
@using System.Timers


<div class="toast fade show bg-info">
    <div class="toast-header">
        <span class="rounded mr-2 alert-primary p-2"  ></span>
        <strong class="mr-auto">@Toast.Titel</strong>
        <small>@DateTime.Now.ToShortTimeString()</small>
        <button type="button" class="ml-2 mb-1 close"
                @onclick="Remove1">
            <span>&times;</span>
        </button>
    </div>
    <div class="toast-body">
       @Toast.Text

    </div>

</div>

@code {
    [Inject]
    public Data.ToasterVM _vm { get; set; }
    [Parameter]
    public Data.ToastItem Toast { get; set; }
     void Remove1()
    {
        _vm.Remove(Toast);
    }
    protected override void OnInitialized()
    {

        var toastTimer = new Timer(30000);
        toastTimer.Elapsed += (sender, args) => { _vm.Remove(Toast); };
        toastTimer.AutoReset = false;
        toastTimer.Start();
    }
}
```

Es fehlt nun nur noch die Änderung im ToastContainer.razor, um die das neue Toast Design anzuwenden. Ersetzen Sie `<div>@toast.Titel</div>` durch den grau markierten Bereich

```
@foreach (var toast in _vm.Toasts)
  {
  <ToastPopup Toast="@toast"></ToastPopup>
  }
```