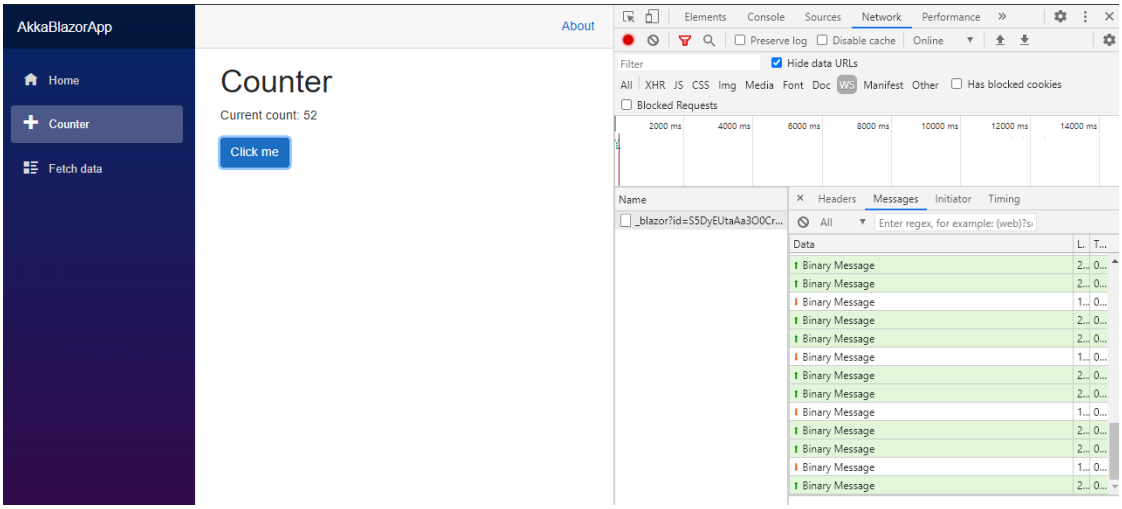
Blazor With AKKA

Blazor AKKA .

Blazor




```
@page "/counter"

<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

Blazor Counter

Server C# , .

XHR(Ajax) , .

, Html + C#

UI ~

node.js

node.js/socket.io

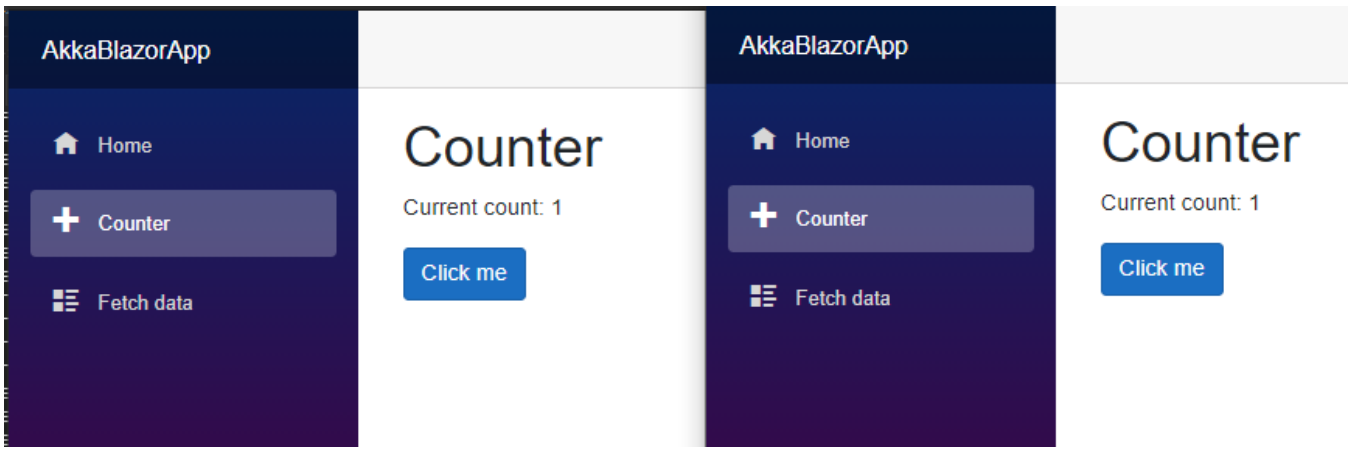
,

..... Blazor .

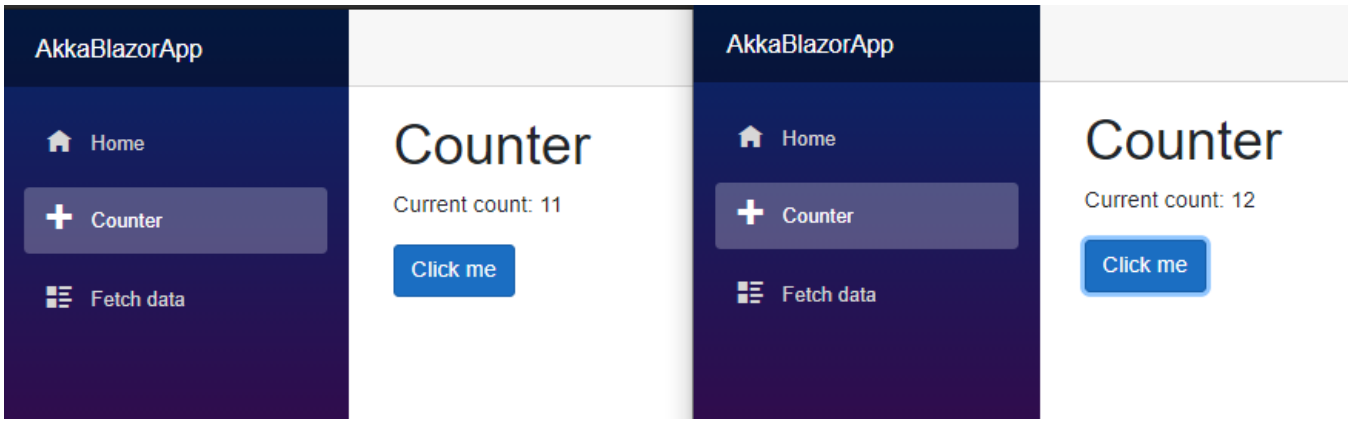
, .

- UI

Blazor , 0 , Clickme 1,1 .



-



.

```
using Akka.Actor;

namespace AkkaBlazorApp.Actors
{
    public enum CmdCount
    {
        CUR_NUM = 0,
        ADD_NUM = 1
    }

    public class CountActor : ReceiveActor
    {
        protected int currentCount = 0;

        public CountActor()
        {
            ReceiveAsync<CmdCount>(async cmdCount =>
            {
                if(cmdCount == CmdCount.CUR_NUM)
                {
                    Sender.Tell(currentCount);
                }
                else if(cmdCount == CmdCount.ADD_NUM)
                {
                    currentCount += 1;
                    Sender.Tell(currentCount);
                }
            });
        }
    }
}
```

- CUR_NUM : .
- ADD_NUM : +1 .

1 : Counter.razor

```
@page "/counter"
@using AkkaDotModule.Config
@using Akka.Actor
@using AkkaBlazorApp.Actors

<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;

    IActorRef countActor = AkkaLoad.ActorSelect("countActor");

    protected override void OnInitialized()
    {
        base.OnInitialized();

        currentCount = (int)countActor.Ask(CmdCount.CUR_NUM).Result;
    }

    private void IncrementCount()
    {
        currentCount = (int)countActor.Ask(CmdCount.ADD_NUM).Result;
    }
}
```

, Lock .

, Blazor razot

- Actor

Lock , .

Lock

Actor Razor .

2. (Push)

```
    private Timer timer;

    protected override void OnAfterRender(bool firstRender)
    {
        if (firstRender)
        {
            timer = new Timer();
            timer.Interval = 1000;
            timer.Elapsed += OnTimerInterval;
            timer.AutoReset = true;
            // Start the timer
            timer.Enabled = true;
        }
        base.OnAfterRender(firstRender);
    }

    private void OnTimerInterval(object sender, ElapsedEventArgs e)
    {
        currentCount = (int)countActor.Ask(CmdCount.CUR_NUM).Result;

        InvokeAsync(() => StateHasChanged());
    }

    public void Dispose()
    {
        // During prerender, this component is rendered without calling OnAfterRender and then immediately
disposed
        // this mean timer will be null so we have to check for null or use the Null-conditional operator ?
        timer?.Dispose();
    }
```

~ (.)

, Blazor

.

3.

.

- 10 . 8 , 8 1 .
- .
- 1, 8 .

.

```
using Akka.Actor;
using System.Threading.Tasks;

namespace AkkaBlazorApp.Actors
{
    public enum CmdCount
    {
        CUR_NUM = 0,
        ADD_NUM = 1,
        ADD_NUM2 = 2,
    }

    public class CountActor : ReceiveActor
    {
        protected int currentCount = 0;

        public CountActor()
        {
            ReceiveAsync<CmdCount>(async cmdCount =>
            {
                if(cmdCount == CmdCount.CUR_NUM)
                {
                    Sender.Tell(currentCount);
                }
                else if(cmdCount == CmdCount.ADD_NUM)
                {
                    currentCount += 1;
                    Sender.Tell(currentCount);

                    if(currentCount % 10 == 0)
                    {
                        //10   .
                        DelayIncrease();
                    }
                }
                else if (cmdCount == CmdCount.ADD_NUM2)
                {
                    //   .
                    currentCount += 8;
                }
            });
        }

        protected void DelayIncrease()
        {
            Task.Run(async () =>
            {
                //      .
                await Task.Delay(1000);
                CmdCount reply = CmdCount.ADD_NUM2;
                return reply;
            }).PipeTo(Self);
        }
    }
}
```

Task Pipe ,

.

.

, .

- : https://getakka.net/articles/actors/finite-state-machine.html
- : https://getakka.net/articles/persistence/architecture.html
- : https://github.com/psmon/AkkaDotModule/tree/master/AkkaBlazorApp