EventSourcing

.

.

- - .
- .
- (message-driven) .
- .
- CRUD CQRS .

-
- ,

```
namespace AkkaNetCoreTest.Actors
{
    public class PersistentActorTest : TestKitXunit
    {
        protected TestProbe probe;

        protected IActorRef persistentActor;

        public PersistentActorTest(ITestOutputHelper output) : base(output)
        {
            Setup();
        }

        public void Setup()
        {
            //      .
            probe = this.CreateTestProbe();

            persistentActor = Sys.ActorOf(Props.Create(() => new MyPersistentActor(probe)),
"persistentActor");
        }

        //
        [Theory(DisplayName = "-   ")]
        [InlineData(5)]
        public void Test1(int cutoffSec)
        {
            // usage
            int expectedCount = 2;

            //
            Cmd cmd1 = new Cmd("  +1");
            Cmd cmd2 = new Cmd("  -0");
            Cmd cmd3 = new Cmd("  +1");
            Cmd cmd4 = new Cmd("  +2");

            Within(TimeSpan.FromSeconds(cutoffSec), () =>
            {
                persistentActor.Tell(cmd1);
                persistentActor.Tell(cmd2);
                persistentActor.Tell(cmd3);
                persistentActor.Tell(cmd4);
                persistentActor.Tell("print"); //    .
                Assert.Equal(expectedCount, probe.ExpectMsg<int>());

                //  .
                persistentActor.Tell(Kill.Instance);
                Task.Delay(500).Wait();

                // ,
                // ,     .
                persistentActor = Sys.ActorOf(Props.Create(() => new MyPersistentActor(probe)),
"persistentActor");
                persistentActor.Tell("print");
                Assert.Equal(expectedCount, probe.ExpectMsg<int>());

            });
        }
    }
}
```

MyPersistentActor

```
using System;
using System.Collections.Immutable;
using Akka.Actor;
using Akka.Persistence;

namespace AkkaNetCore.Actors.Study
{
    #region MessageData
    public class Shutdown { }

    //  .
    //1  n    .
    public class Cmd
    {
        public Cmd(string data)
        {
            Data = data;
        }

        public string Data { get; }
    }

    public class Evt
    {
        public Evt(string data)
        {
            Data = data;
        }

        public string Data { get; }
    }

    public class ExampleState
    {
        private readonly ImmutableList<string> _events;

        public ExampleState(ImmutableList<string> events)
        {
            _events = events;
        }

        public ExampleState() : this(ImmutableList.Create<string>())
        {
        }

        public ExampleState Updated(Evt evt)
        {
            return new ExampleState(_events.Add(evt.Data));
        }

        public ExampleState RemoveLastItem()
        {
            return new ExampleState(_events.RemoveAt(_events.Count-1));
        }

        public int Size => _events.Count;

        public override string ToString()
        {
            return string.Join(", ", _events.Reverse());
        }

        public int CountInBasket()
        {
            int Count = 0;
            //      .
            foreach(var str in _events)
            {
```

```
            if (str.Contains(""))
            {
                Count++;
            }
            else if (str.Contains(""))
            {
                Count--;
            }
        }
        return Count;
    }
}

#endregion

#region Actor
public class MyPersistentActor : UntypedPersistentActor
{
    private ExampleState _state = new ExampleState();

    protected IActorRef probe;

    public MyPersistentActor(IActorRef probe)
    {
        this.probe = probe;
    }

    private void UpdateState(Evt evt)
    {
        _state = _state.Updated(evt);
    }

    private int NumEvents => _state.Size;

    public override Recovery Recovery => new Recovery(fromSnapshot: SnapshotSelectionCriteria.None);

    protected override void OnRecover(object message)
    {
        switch (message)
        {
            case Evt evt:
                UpdateState(evt);
                break;
            case SnapshotOffer snapshot when snapshot.Snapshot is ExampleState:
                _state = (ExampleState)snapshot.Snapshot;
                break;
        }
    }

    protected override void OnCommand(object message)
    {
        switch (message)
        {
            case Cmd cmd:
                // Command    .
                //Persist(new Evt($"{cmd.Data}-{NumEvents}"), UpdateState);

                Persist(new Evt($"{cmd.Data}-{NumEvents + 1}"), evt =>
                {
                    UpdateState(evt);
                    Context.System.EventStream.Publish(evt);
                });
                break;
            case "snap":
                SaveSnapshot(_state);
                break;
            case "print":
                //  .
                Console.WriteLine("Try print");
                Console.WriteLine(_state);
```

```
                    //   .(  )
                    probe.Tell(_state.CountInBasket());
                    break;
                case Shutdown down:
                    probe.Tell("die");
                    Context.Stop(Self);
                    break;
            }
        }

        public override string PersistenceId { get; } = "sample-id-1";  //    .
    }

    #endregion
}
```