How to get current state from bbv.Common.StateMachine (now Appccelerate.StateMachine) class?

Asked 12 years, 1 month ago Modified 4 years ago Viewed 3k times

13

`bbv.Common.StateMachine` class is the best state machine code I have ever seen. But it lacks just one thing: getting current state.

This is an order tracking system:

```
fsm = new ActiveStateMachine<States, Events>();

        fsm.In(States.OrderCreated)
            .On(Events.Submitted)
            .Goto(States.WaitingForApproval);
        fsm.In(States.WaitingForApproval)
            .On(Events.Reject)
            .Goto(States.Rejected);
        fsm.In(States.WaitingForApproval)
            .On(Events.Approve)
            .Goto(States.BeingProcessed);
        fsm.In(States.BeingProcessed)
            .On(Events.ProcessFinished)
            .Goto(States.SentByMail);
        fsm.In(States.SentByMail)
            .On(Events.Deliver)
            .Goto(States.Delivered);

        fsm.Initialize(States.OrderCreated);
        fsm.Start();
        fsm.Fire(Events.Submitted);
        // Save this state to database
```

You can see how it works easily.

But I want to save the order state in the database. So I will be able to show in which state is the order.

I need a

```
fsm.GetCurrentState()
//show this state in the a table
```

method. Actually there is a way: I can use `ExecuteOnEntry` and change a local value on every state's entry. But it will be cumbersome to write `ExecuteOnEntry` for every state because I will be repeating myself!

There must be a delicate way to do it.

c# state-machine

Share
Improve this question
Follow

edited Dec 22, 2017 at 1:15

WickedW
2,581 ● 5 ● 26 ● 57

asked Nov 12, 2012 at 20:30

ramazan polat
7,860 ● 4 ● 57 ● 85

2 Answers

Sorted by: Highest score (default)

18

As Daniel explained, this is by design. Let me explain why:

The state machine allows queuing of events. Therefore, asking the state machine about its current state can be misleading. It is currently in state A, but there is already an event queued that will get it to state B.

Furthermore, I consider it to be bad design, to couple the state machine internal states (the ones you use in your state machine definition) directly with state machine external states (the ones you want to persist in the database). If you couple these two directly, you lose the ability to refactor the state machine internally without effecting the outside (in your case the database). I frequently encounter the scenario in which I have to split a state A, into A1 and A2 because I have to attach different actions to them, but nonetheless they still represented the same state to the environment. Therefore, I strongly advise you to separate the internal and external states, either as you wrote with ExecuteOnEntry() or by providing a mapping and using an extension. This is an extension that will get you the current state:

```
public class CurrentStateExtension : ExtensionBase<State, Event>
{
    public State CurrentState { get; private set; }

    public override void SwitchedState(
        IStateMachineInformation<State, Event> stateMachine,
        IState<State, Event> oldState,
        IState<State, Event> newState)
    {
        this.CurrentState = newState.Id;
    }
}
```

You can add the extension to the state machine in this way:

```
currentStateExtension = new CurrentStateExtension();
```

```
machine.AddExtension(currentStateExtension);
```

Of course you can use this extension directly to get access to the current state, too. To make it even simpler, let the class that defines the state machine implement the extension and pass itself as an extension. Let you get rid of the extra class.

A last note: when you ask questions about bbv.Common (or Appccelerate as it is called now) in the google group at https://groups.google.com/forum/?fromgroups#!forum/appccelerate, it's easier for me to find the question and answer it ;-)

Share
Improve this answer
Follow

edited Dec 4, 2020 at 18:18
Andrew
20k • 13 • 108 • 121

answered Nov 13, 2012 at 12:45

Urs
228 • 1 • 6

It seems SwitchedState get called after ExecuteOnEntry. So when I use this extension inside ExecuteOnEntry, I get old state – Mohammad Reza Sadreddini Mar 25, 2019 at 14:31

7

This is by design. We consider querying the state of the state machine as design smell. But of course there are exception cases. You have the following two options:

1. Use the `ExecuteOnEntry` methods to save the state of the order. This reflects the way to go because you don't wan't to leak the states of the statemachine into your business logic.
2. Write your own state machine decorator which uses internally `StateMachine<TState, TEvent>`. This exposes the state.

Daniel

Share Improve this answer Follow

answered Nov 12, 2012 at 20:58

Daniel Marbach
2,314 • 13 • 15