eNJoy CQRS + ES Documentation

Release 1.0.0

Nelson C. Viana Junior

Jul 09, 2016

Contents

Contents:

CHAPTER 1

Getting Started

1.1 Creating your first aggregate

To create an aggregate with event sourcing support you need inherit from `Aggregate`.

```
public class Game : Aggregate
{
    public Game() { }

    protected override void RegisterEvents()
    {
    }
}
```

This is the most basic implementation, but the aggregate not have any behavior. Lets go to put something then.

1.2 Creating your first domain event

You have two choices when you will create your domain event. First option is the must simple base type is `IDomainEvent`. The second one is the `DomainEvent` (it implements `IDomainEvent`).

The difference between them is that `DomainEvent` requires the unique identifier of the respective aggregate on the constructor.

```
public class GameStarted : DomainEvent
{
    public string PlayerOne { get; }
    public string PlayerTwo { get; }

    public GameStarted(Guid aggregateId, string playerOne, string playerTwo) : base(aggregateId)
    {
        PlayerOne = playerOne;
        PlayerTwo = playerTwo;
    }
}
```

After to implement domain event, lets go back to the aggregate.

```
public class Game : Aggregate
{
    public string PlayerOne { get; private set; }
    public string PlayerTwo { get; private set; }

    public Game()
    {
    }

    public Game(string playerOne, string playerTwo) : this()
    {
        Emit(new GameStarted(Guid.NewGuid(), playerOne, playerTwo));
    }

    protected override void RegisterEvents()
    {
    }
}
```

Now, when create an instance of Game with two strings arguments, it will emit the domain event. In this case will be `GameStarted`.

Although we have done it, nothing happens to the instance. Why?

Because the instance not have subscription for `GameStarted` event.

```
// omitted code

protected override void RegisterEvents()
{
    SubscribeTo<GameStarted>(@event =>
    {
        Id = @event.AggregateId;
        PlayerOne = @event.PlayerOne;
        PlayerTwo = @event.PlayerTwo;
    });
}

// omitted code
```

If you inspect the aggregate you will see the `GameStarted` instance in the UncommitedEvents property, it indicate that event not persisted into event store yet. Dont worry, we will see how to persist events soon.

1.3 Next steps

- *Aggregate Anatomy*

CHAPTER 2

Aggregate Anatomy

CHAPTER 3

Indices and tables

- genindex
- modindex
- search