阅读本篇文章前，你应该对CQRS Saga模式有了解

**定义事件和命令**

// 下单命令，用于开启 Saga

public class PlaceOrderCommand : CorrelatedBy<Guid>

{

    // SagaID，此字段的值将被用来初始化 Saga 的 SagaID

    public Guid CorrelationId { get; }

    public PlaceOrderCommand () {

        CorrelationId = Guid.NewGuid ();

    }

}

// 检查库存命令

public class CheckInStoreCommand : CorrelatedBy<Guid>

{

public Guid CorrelationId { get; }

    public CheckInStoreCommand (Guid correlationId) {

        CorrelationId = correlationId;

    }

}

// 支付完成事件

public class PayCompletedEnvet : CorrelatedBy<Guid>

{

public Guid CorrelationId { get; }

    public PayCompletedEnvet (Guid correlationId) {

        CorrelationId = correlationId;

    }

}

**定义Saga**

// 下单 Saga

public class PlaceOrderSaga:

    // Saga 接口，必须继承

    ISaga,

    // 用于创建Saga实例的命令

    InitiatedBy<PlaceOrderCommand>,

    // 后续命令

    Orchestrates<CheckInStoreCommand>,

    Orchestrates<PayCompletedEnvet>

{

    // SagaId

    public Guid CorrelationId { get; set; }

    // 处理"下单开始命令"

    public Task Consume (ConsumeContext<PlaceOrderCommand> context) {

        Console.WriteLine ("下单开始");

        // 发送"检查库存命令"

        context.Publish (new CheckInStoreCommand (context.CorrelationId.Value));

        return Task.CompletedTask;

}

    // 处理"检查库存命令"

    public Task Consume (ConsumeContext<CheckInStoreCommand> context) {

        Console.WriteLine ("检查库存");

        return Task.CompletedTask;

    }

    // 处理"支付完成事件"

    public Task Consume (ConsumeContext<PayCompletedEnvet> context) {

        Console.WriteLine ("支付完成");

        return Task.CompletedTask;

    }

}

**将Saga添加到队列**

// 创建基于内存的总线

var bus = Bus.Factory.CreateUsingInMemory (config => {

    // 设置接收队列，队列名 test\_queue

    config.ReceiveEndpoint ("test\_queue", ep => {

        // 添加 Saga PlaceOrderSaga

        // 使用内存仓库持久化Saga

        ep.Saga<PlaceOrderSaga> (new InMemorySagaRepository<PlaceOrderSaga> ());

    });

});

bus.Start ();

// 发布下单开始命令

bus.Publish (new PlaceOrderCommand ()).Wait ();

bus.Stop ();