阅读本篇文章前，请了解上篇文章“Saga 模式”

使用EF持久化需要使用依赖注入容器（不使用依赖注入我不知道怎么做）

**安装包**

依赖注入包：

MassTransit.Extensions.DependencyInjection

EF Core包：

MassTransit.EntityFrameworkCore

SqlServer EF包：

由于我使用SqlServer，所有安装如下包

Microsoft.EntityFrameworkCore.SqlServer

**Saga指定主键**

// 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>

{

    // 指定主键

    [Key]

    public Guid CorrelationId { get; set; }

    public int MySagaState { 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;

}

    ...

}

**新建数据库上下文**

public class MyDbContext : DbContext {

    public MyDbContext (DbContextOptions<MyDbContext> options) : base (options) { }

    // 存储 PlaceOrderSaga

    public DbSet<PlaceOrderSaga> PlaceOrderSagas { get; set; }

}

**添加Saga到总线**

static void Main (string[] args) {

    // 新建服务集合（DI容器）

    IServiceCollection services = new ServiceCollection ();

    // 添加 MassTransit

    services.AddMassTransit (config => {

        // 注入 Saga 到 DI

        // 使用基于 EFCore 的仓库

        config.AddSaga<PlaceOrderSaga> ().EntityFrameworkRepository (r => {

            // Pessimistic 模式

            r.ConcurrencyMode = ConcurrencyMode.Pessimistic;

            // 添加注入数据库

            r.AddDbContext<DbContext, MyDbContext> ((provider, builder) => {

                builder.UseSqlServer ("Data Source=(localdb)\\ProjectsV13;Initial Catalog=MassTransitTest;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

            });

        });

        // 添加总线

        config.AddBus (service => {

            // 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

                    ep.ConfigureSaga<PlaceOrderSaga> (service);

                });

            });

            return bus;

        });

});

var serviceProvicer = services.BuildServiceProvider ();

    // 创建数据库

    // 这里为了方便直接使用代码创建数据库

    // 在实际项目中你应该使用EF迁移创建数据库

    var myDbContext = serviceProvicer.GetService<DbContext> ();

myDbContext.Database.EnsureCreated ();

var bus = serviceProvicer.GetService<IBusControl> ();

bus.Start ();

    // 发布消息，Publish 方法向所有队列发送消息

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

    Console.WriteLine ("Press any key to exit");

Console.ReadKey ();

    bus.Stop ();

}