消费者与处理器执行相似的功能，一个消息消费者可以消费一个或多个消息类型的类

**定义消费者**

// 消费者

public class UpdateCustomerConsumer :

    // 处理的消息类型，可以多继承

    IConsumer<YourMessage>

{

    // 处理消息

    public async Task Consume (ConsumeContext<YourMessage> context) {

        await Console.Out.WriteLineAsync ($"Updating customer: {context.Message.Text}");

    }

}

**将消费者添加到队列**

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

    // 设置接收队列，队列名 test\_queue

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

        // 添加消息消费者

        ep.Consumer<UpdateCustomerConsumer> ();

    });

});

Consume方法是异步的，并返回一个task。MassTransit 等待该任务, 在此期间消息对其他接收端点不可用，如果consume 方法成功完成，则消息将被确认并从队列中删除。

如果消费者错误（例如抛出异常，导致Faulted的任务状态），或者以某种方式被取消cancelled（被取消的Canceled任务状态），则异常被传播回管道，在那里消息最终被重试或移动到错误队列

如果端点的配置改变，则可能接收到没有对应消息类型的消费者。如果发生这种情况，则将消息移动到\_skipped队列（由原始队列名称前缀）。保留原始消息内容，并添加附加标题来指示移动消息的主机