.net core 3.0使用BackgroudService运行多个后台服务

前言

由于业务系统需要向京东推送支付状态，在接口层接收到京东的支付json之后推送给阿里的MNS队列，然后后台程序监听MNS队列进行支付状态同步等。由于之前的是基于控制台程序，开启了两个异步Task，而这个Task应该会随着应用程序池回收而消失，也有可能是由于队列消费出错而被回收。导致后续消息都不消费了。所以基于dotnet core 3.0的BackgroudService来运行后台程序

创建HostBuilder

> 在微软官方文档里有关于HostBuilder还有BackgroudService的使用，这里就不深入了

```
public static IHostBuilder CreateHostBuilder (string[] args) =>
    Host.CreateDefaultBuilder (args)
    .ConfigureServices ((hostContext, services) => {
        var configBuilder = new ConfigurationBuilder ()
            .SetBasePath (Directory.GetCurrentDirectory ())
            .AddJsonFile ("appsettings.json", true, true);

        services.AddConfig (options => {
            options.ConfigurationBuilder = configBuilder;
            options.Namespace = "Dev001.ThirdService";
        });

        IServiceProvider serviceProvider = services.BuildServiceProvider ();
        IConfig _config = serviceProvider.GetService<IConfig> ();

        MqttConfig mqttConfig = new MqttConfig {
            Server = _config.Get ("EMQServer"),
            Port = int.Parse (_config.Get ("EMQPort")),
            ClientIdPre = "dyIotJDExpressListenerTaskCloud" //
        };

        services.AddCache (options => {
            options.RedisEndPoints = _config.Get ("RedisEndPoints");
            options.RedisServer = _config.Get ("RedisServer");
            options.RedisPwd = _config.Get ("RedisPwd");
            options.RedisDBId = _config.Get ("RedisDBId");
        }, cacheDbOptions => {
            cacheDbOptions.AssemblyName = "ExpressOrder.Dao";
        });
        serviceProvider = services.BuildServiceProvider ();
        ICache _cache = serviceProvider.GetService<ICache> ();
        services.AddSingleton (new AliMNSHelper (_config.Get ("MNSSH_AccessKey"),
_config.Get ("MNSSH_SecretKey"), _config.Get ("MNSSH_EndPoint")));
        RedisCache cache = new RedisCache (_config.Get ("RedisServer"), int.Parse
(_config.Get ("RedisDBId")));
        services.AddSingleton (new MqttNetClient (mqttConfig, null, cache));
```

```
        services.AddSingleton (
            new Dictionary<DbConnectionNameEnum, string> { {
DbConnectionNameEnum.MainConnection, _config.Get ("MySqlServer") },
                { DbConnectionNameEnum.AccountConnection, _config.Get
("MySqlServerAcc") }
            });
        services.AddSingleton<IElasticClient> (es => new ElasticClient (new
ConnectionSettings (new Uri (_config.Get ("ElasticSearchUrl"))).RequestTimeout
(TimeSpan.FromSeconds (5)).DefaultFieldNameInferrer ((name) => name)));

        //以上是注入IOC的实例
        services.AddHostedService<SyncMessageReplyListenerServiceImpl> ();
        services.AddHostedService<JDPayNotifyListenerServiceImpl> ();
        //注入两个后台运行服务
    });
```

京东支付后台运行服务

```
public class JDPayNotifyListenerServiceImpl : BackgroundService {
    private readonly AliMNSHelper _aliMNSHelper;
    private readonly IConfig _config;
    private readonly IKernelorderChargeDao _kernelorderChargeDao;
    private readonly IKernelPostOrderDao _kernelPostOrderDao;
    private readonly IUtilService _utilService;
    private readonly IMQTTDeviceService _deviceService;
    private readonly ICache _cache;
    private readonly IKernelDeviceGroupDao _kernelDeviceGroupDao;
    private readonly IKernelPostorderPayInfoDao _kernelPostorderPayInfoDao;
    private readonly IKernelSmartboxBoxactInfoDao _kernelSmartboxBoxactInfoDao;
    public JDPayNotifyListenerServiceImpl (AliMNSHelper aliMNSHelper, IConfig
config, IKernelorderChargeDao kernelorderChargeDao,
        IUtilService utilService, IMQTTDeviceService deviceService,
IKernelPostOrderDao kernelPostOrderDao, ICache cache, IKernelDeviceGroupDao
kernelDeviceGroupDao, IKernelPostorderPayInfoDao kernelPostorderPayInfoDao,
IKernelSmartboxBoxactInfoDao kernelSmartboxBoxactInfoDao) {
        _aliMNSHelper = aliMNSHelper;
        _config = config;
        _kernelorderChargeDao = kernelorderChargeDao;
        _utilService = utilService;
        _deviceService = deviceService;
        _kernelPostOrderDao = kernelPostOrderDao;
        _cache = cache;
        _kernelPostorderPayInfoDao = kernelPostorderPayInfoDao;
        _kernelDeviceGroupDao = kernelDeviceGroupDao;
        _kernelSmartboxBoxactInfoDao = kernelSmartboxBoxactInfoDao;
    }

    protected override Task ExecuteAsync(CancellationToken stoppingToken)
    {
        Task.Run(() => Process(stoppingToken));
        return Task.CompletedTask;
```

```
    }

    protected void Process (CancellationToken stoppingToken) {
        Console.WriteLine ("开启监听京东支付回调");
        string queueName = _config.Get ("JDPayNotifyQueueName");
        while (!stoppingToken.IsCancellationRequested) {
            Message mqMsg = _aliMNSHelper.ReceiveMsg (queueName);
            if (mqMsg != null) {
                string mqResInfo = string.Empty;
                try {
                    //只要进了这里就干掉了
                    if (mqMsg != null && !string.IsNullOrEmpty (mqMsg.Body)) {
                        _aliMNSHelper.DeleteMsg (queueName, mqMsg.ReceiptHandle,
out mqResInfo);
                        NLogger.Default.Info ("JDPayNotifyQueue支付回调接收信息: "
+ mqMsg.Body);
                        Console.WriteLine ($"JDPayNotifyQueue接收信息:
{mqMsg.Body}:当前时间:{DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss")}");
                        JDPayReturnModel payNotifyInModel =
JsonConvert.DeserializeObject<JDPayReturnModel> (mqMsg.Body);
                        string signData = JDEncryptionUtil.SignReturnData
(payNotifyInModel, _config.Get ("JDPayMd5Key"));

                        if (!signData.EqualIgnoreCase (payNotifyInModel.sign)) {
                            NLogger.Default.Info ("京东支付异步回调非法请求: " +
mqMsg.Body);
                            continue;
                        }
                        string payNotifyInModelStr = EncryptUtil.Base64Decode
(payNotifyInModel.data);
                        JDPayNotifyModel jDPayNotifyModel =
JsonConvert.DeserializeObject<JDPayNotifyModel> (payNotifyInModelStr);
                        bool success = false;
                        //订单号查询是否支付
                        var rechargeOrderInfo =
_kernelorderChargeDao.GetRechargeOrder (jDPayNotifyModel.orderNo).Result;
                        if (rechargeOrderInfo != null) {
                            //充值订单还没处理过
                            if (!rechargeOrderInfo.IsPaid) {
                                //处理京东支付类型
                                int payType = jDPayNotifyModel.payType.Equals
("WXPAY") ? 2 : 10;
                                success =
_kernelorderChargeDao.SetRechargeOrderPaid (rechargeOrderInfo.OrderId, payType,
Convert.ToDecimal (jDPayNotifyModel.payAmount), Convert.ToDateTime
(jDPayNotifyModel.payTime), jDPayNotifyModel.orderType,
jDPayNotifyModel.orderSource, jDPayNotifyModel.externalId,
jDPayNotifyModel.businessType, jDPayNotifyModel.outBizNo,
jDPayNotifyModel.termNo).Result;
                                NLogger.Default.Info ("订单号{0},更新支付结果{1}",
rechargeOrderInfo.OrderId, success ? "成功" : "失败");

                                if (success) {
                                    var postOrder =
```

```
_kernelPostOrderDao.GetPostOrder (rechargeOrderInfo.OrderId.ToString ()).Result;
                                    var kernelDeviceGroup =
_kernelDeviceGroupDao.GetDeviceGroup (postOrder.DeviceSn).Result;
                                    //查询设备格口信息
                                    var kernelSmartboxBoxactInfo =
_kernelSmartboxBoxactInfoDao.GetCabinetSeptum (kernelDeviceGroup.MainDeviceId, 0,
postOrder.PostOrderIdStr).Result.FirstOrDefault ();
                                    var cellSN = string.Empty;
                                    if (kernelSmartboxBoxactInfo != null) {
                                        cellSN = kernelSmartboxBoxactInfo.CellSN;
                                    }
                                    //支付成功完成后推送给柜机
                                    _deviceService.NotifyPayComplete (new
NotifyPayCompleteModel { PostOrderId = rechargeOrderInfo.OrderId, DeviceGroupSN =
postOrder.DeviceSn, DeviceType = kernelDeviceGroup.DeviceType, CellSN = cellSN });
                                    NLogger.Default.Info ($"
{LogEnum.JDApiJD.GetEnumDescription()}#JDPayNotifySmartBox#{postOrder.DeviceSn}#开
始推送支付完成通知");
                                    //查询设备来区分是哪种设备 云柜/京东寄件柜
                                    var producer = _config.Get ("JDProducerCode");
                                    //不是6 就是云柜
                                    if (kernelDeviceGroup.DeviceType != 6) {
                                        producer = _config.Get
("JDCloundProducerCode");
                                    }
                                    var terminationOfHandover = new
UpdateStatusInModel () {
                                        producer = producer,
                                        thirdCode = postOrder.DeviceSn,
                                        waybillCode = postOrder.ExpressNumber,
                                        status =
PostOrderStatus.WaitPutIn.IntToEnum (), //对于京东 1这个状态是客户支付成功
                                    };

                                    //同步运单状态接口
                                    var courierTakesOutPackage =
_utilService.JDHttpPostSync<UpdateStatusInModel, JDHttpBaseReturnModel<bool>>
(terminationOfHandover, JDInterfaceEnum.updateWaybillStatus, new JDBaseModel { },
postOrder.DeviceSn, null);
                                    if (!courierTakesOutPackage.success) {
                                        //记录日志
                                        NLogger.Default.Info ($"
{LogEnum.JDApiJD.GetEnumDescription()}#{courierTakesOutPackage.ToString()}");

                                    }
                                    //兼容云柜erp订单数据
                                    var orderPayInfo = new PostOrderPayInfo () {
                                        PaymentStatus =
PaymentStatusEnum.Paid.IntToEnum (),
                                        PostOrderId =
rechargeOrderInfo.OrderId.ToString (),
                                        PayTime = DateTime.Now,
                                        PayMessage = "寄件支付",
                                        PaymentType = payType
```

```
                                    };
                                    //添加支付信息
                                    success =
_kernelPostorderPayInfoDao.UpdatePayInfoSync (orderPayInfo).Result;
                                    NLogger.Default.Info ("订单号{0},寄件更新支付结
果{1}", rechargeOrderInfo.OrderId, success ? "成功" : "失败");
                                }
                            }
                        }
                    } catch (Exception ex) {
                        //写入异常日志
                        NLogger.Default.Error ("京东支付回调异常" + ex.ToString ());
                    }
                }
            }
        }
    }
```

如何支持多个后台运行服务

> 主要是实现

```
//该方法随着HostBuilder启动而启动，即生命周期一致
protected override Task ExecuteAsync(CancellationToken stoppingToken)
{
   //必须要开启异步处理才能支持多个后台运行程序
   Task.Run(() => Process(stoppingToken));
   return Task.CompletedTask;
}
```