.NET客户端API

重构说明

代码逻辑说明

项目地址

rocketmq-client-dotnet

项目目录简介

```
.
├── rocketmq-client-dotnet RocketMQ .NET Client 文件
    └── example
        └── demo
            ├── ConsumerDemo PushConsumer Demo
            ├── ProducerDemo Producer Demo
            └── PullConsumerDemo PullConsumer Demo
        └── nugetTest
                └──  nugetTest Nuget package test.
    └── src
        └── RocketMQ.NETClient  NET Client
            ├── Consumer Consumer API
            ├── Interop  Const Value
            ├── Message Message API
            └── Producer Producer API
```

QuickStart

QuickStart

API参考

Nuget包

1. 包地址

rocketmq-client-dotnet

2. 说明

- 使用说明
  QuickStart
- 使用要求

```
    .NET FrameWork >=4.6.1

or

    .NET Standard >=2.0
```

API 对齐说明

功能	C	.NET
同步消息发送	Y	Y
顺序消息发送	Y	Y
单向消息发送	Y	Y
拉取消息消费	Y	Y
推送消息消费	Y	Y
延时消息	Y	Y
消息压缩	Y	Y
消息过滤	Y	Y
字符串消息体	Y	Y
字节流消息体	N	N
Topic设置	Y	Y

Producer 创建说明

之前的方式

1. 创建`DefaultProducerBuilder`对象 `producerBuilder`
2. 使用 `producerBuilder`设置想要生成的`producerBuilder`参数
3. 调用 `producerBuilder`中的`Builder`函数返回一个`IProducer` 实例 `producer`
4. 使用 `producer`

示例代码：

```
    class MainClass
    {
        private static ProducerWrap.QueueSelectorCallback _queueSelectorCallback =
new ProducerWrap.QueueSelectorCallback(
            (size, message, args) =>
            {
                Console.WriteLine($"size: {size}, message: {message}, ptr:
{args}");
```

```
                return 0;
            });

        public static void Main(string[] args)
        {
            Console.Title = "Producer";

            Console.WriteLine("Start create producer.");
            var producerPtr = ProducerWrap.CreateProducer("GID_test");
            if (producerPtr == IntPtr.Zero)
            {
                Console.WriteLine("zero. Oops.");
            }

            Console.WriteLine(producerPtr.ToString());
            Console.WriteLine("end create producer.");

            var p = new MainClass();
            var producer = new HandleRef(p, producerPtr);
            try
            {
                var setNameServerAddressResult =
ProducerWrap.SetProducerNameServerAddress(producer, "127.0.0.1: 9876");
                Console.WriteLine("set name server address result:" +
setNameServerAddressResult);

                var setProducerLogPathResult =
ProducerWrap.SetProducerLogPath(producer, "C:/rocketmq_log.txt");
                Console.WriteLine("set producer log path result:" +
setProducerLogPathResult);

                var setLogLevelResult = ProducerWrap.SetProducerLogLevel(producer,
CLogLevel.E_LOG_LEVEL_TRACE);
                Console.WriteLine("set producer log level result:" +
setLogLevelResult);


                var startResult = ProducerWrap.StartProducer(producer);
                Console.WriteLine("start result:" + startResult);

                while (true)
                {
                    // message
                    var message = MessageWrap.CreateMessage("Test");
                    Console.WriteLine("message intPtr:" + message);

                    var p1 = new MainClass();
                    var messageIntPtr = new HandleRef(p1, message);

                    var setMessageBodyResult =
MessageWrap.SetMessageBody(messageIntPtr, "hello" + Guid.NewGuid());
                    Console.WriteLine("set message body result:" +
setMessageBodyResult);
```

```
                    var setTagResult = MessageWrap.SetMessageTags(messageIntPtr,
"tag_test11");
                    Console.WriteLine("set message tag result:" + setTagResult);

                    var setPropertyResult =
MessageWrap.SetMessageProperty(messageIntPtr, "key1", "value1");
                    Console.WriteLine("set message property result:" +
setPropertyResult);


                   //SendMessageSync
                   var sendResult = ProducerWrap.SendMessageSync(producer,
messageIntPtr, out CSendResult sendResultStruct);
                    Console.WriteLine("send result:" + sendResult + ", msgId: " +
sendResultStruct.msgId.ToString());

                   {sendResultStruct.sendStatus}, offset:
{sendResultStruct.offset}");

                    Thread.Sleep(500);
                }

                var shutdownResult = ProducerWrap.ShutdownProducer(producer);
                Console.WriteLine("shutdown result:" + shutdownResult);

                var destoryResult = ProducerWrap.DestroyProducer(producer);
                Console.WriteLine("destory result:" + destoryResult);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
            Console.ReadKey(true);
        }
    }
```

重构使用说明

1. 创建producer对象(多种构造函数)
2. 使用producer发送消息

```
    //创建一个 producer
    MQProducer producer = new MQProducer("GroupA", "127.0.0.1:9876");
    producer.StartProducer();

    // 创建一个消息 message
    MQMessage message = new MQMessage("test");

    // 使用producer发送消息
    // SendMessageSync
```

```
    var sendResult = producer.SendMessageSync(message);
    Console.WriteLine("send result:" + sendResult + ", msgId: " +
sendResult.MessageId);
```

PushConsumer使用说明(推荐使用)

使用说明

1. 创建一个Push消费者
2. 订阅一个topic
3. 注册回调函数
4. 启动消费者

```
   // 创建一个Push消费者
    var consumer = new MQPushConsumer("xx", "127.0.0.1:9876");
    Console.WriteLine($"consumer: {consumer}");

    // 设置日志目录和级别
    consumer.SetPushConsumerLogPath(".\\consumer_log.txt");
    consumer.SetPushConsumerLogLevel(LogLevel.Trace);

    // 获取消费者组号
    var groupId = consumer.GetPushConsumerGroupID();
    Console.WriteLine($"groupId: {groupId}");

    //  订阅一个`topic`
    consumer.Subscribe("test", "*");

    //注册回调函数
    consumer.RegisterMessageCallback(_callback);

    //启动消费者
    var result=consumer.StartPushConsumer();
    Console.WriteLine($"start push consumer ptr: {result}");
```

PullConsumer 使用说明

使用说明

1. 创建一个PullConsumer
2. 开启消费者
3. 填充消息队列
4. 主动拉取消费

```
    //创建一个PullConsumer
    MQPullConsumer consumer = new MQPullConsumer("xxx", "127.0.0.1:9876",
".\\log.txt", LogLevel.Trace);
```

```
    //开启消费者
    var result = consumer.StartPullConsumer();
    Console.WriteLine($"start Pull consumer ? : {result}");

    //填充消息队列
    CMessageQueue[] msgs = consumer.FetchSubscriptionMessageQueues("test");


    for (int j = 0; j < msgs.Length; j++)
    {
        int flag = 0;


        Console.WriteLine("msg topic : " + new string(msgs[j].topic));

        MessageQueue mq = new MessageQueue { topic = new string(msgs[j].topic),
brokeName = new string(msgs[j].brokerName), queueId = msgs[j].queueId };

        while (true)
        {
            try
            {

                //主动拉取消费
                CPullResult cPullResult = consumer.Pull(mq,msgs[j], "",
MQPullConsumer.getMessageQueueOffset(mq), 32);
                Console.WriteLine(new string(msgs[j].topic) + " status : " +
cPullResult.pullStatus +"Max offset "+ cPullResult.maxOffset + " offset: " +
cPullResult.nextBeginOffset + " Quene Id" + msgs[j].queueId);
                //Console.WriteLine(" " + msg.topic);
                long a = cPullResult.nextBeginOffset;
                MQPullConsumer.putMessageQueueOffset(mq, a);
                switch (cPullResult.pullStatus)
                {
                    case CPullStatus.E_FOUND:
                        break;
                    case CPullStatus.E_NO_MATCHED_MSG:
                                break;
                    case CPullStatus.E_NO_NEW_MSG:
                        flag = 1;
                        break;
                    case CPullStatus.E_OFFSET_ILLEGAL:
                        flag = 2;
                        break;
                    default:
                        break;
                }

                if(flag == 1|| cPullResult.nextBeginOffset ==
cPullResult.maxOffset)
                {
                    break;
                }
```

```
                if (flag == 2)
                {
                    break;
                }

            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }


    }
```