Netty 是异步执行的，也就是说，请求发送到 Broker 被处理后，返回结果时，在客户端的处理线程已经不再是发送请求的线程，那么客户端如何确定返回结果对应哪个请求呢？很简单，我们可以通过返回标志来判断。

其次，做一系列存储前发送请求的数据检查，比如死信消息处理、Broker 是否拒绝事务消息处理、消息基本检查等。消息基本检查方法为 AbstractSendMessageProcessor.msgCheck()，该方法的主要功能如下：

- 校验 Broker 是否配置可写。
- 校验 Topic 名字是否为默认值。
- 校验 Topic 配置是否存在。
- 校验 queueId 与读写队列数是否匹配。
- 校验 Broker 是否支持事务消息（msgCheck 之后进行的校验）。

（3）执行 DefaultMessageStore.putMessage()方法进行消息校验和存储模块检查。

在真正保存消息前，会对消息数据做基本检查、对存储服务做可用性检查、对 Broker 做是否 Slave 的检查等，总结如下：

- 校验存储模块是否已经关闭。
- 校验 Broker 是否是 Slave。
- 校验存储模块运行标记。
- 校验 Topic 长度。
- 校验扩展信息的长度。
- 校验操作系统 Page Cache 是否繁忙。具体实现代码如下：

```
    public boolean isOSPageCacheBusy() {
    long begin = this.getCommitLog().getBeginTimeInLock();
    long diff = this.systemClock.now() - begin;
    return diff < 10000000 && diff > this.messageStoreConfig.
getOsPageCacheBusyTimeOutMills();
    }
```