一、为什么Demo里没有创建Topic，却可以使用

首先来看看 producer 在send的时候执行了哪些步骤

1. 消息发送之前，首先需要获取主题的路由信息，只有获取了这些信息我们才知道消息要发送哪个具体 Broker

```
private SendResult sendDefaultImpl(
    Message msg,
    final CommunicationMode communicationMode,
    final SendCallback sendCallback,
    final long timeout
) throws MQClientException, RemotingException, MQBrokerException, InterruptedException {
    this.makeSureStateOK();
    Validators.checkMessage(msg, this.defaultMQProducer);
    final long invokeID = random.nextLong();
    long beginTimestampFirst = System.currentTimeMillis();
    long beginTimestampPrev = beginTimestampFirst;
    long endTimestamp = beginTimestampFirst;
    TopicPublishInfo topicPublishInfo = this.tryToFindTopicPublishInfo(msg.getTopic());
```


2. tryToFindTopicPublishInfo 是查找主题的路由信息的方法。如果生产者缓存了 topic 的路由信息，如果该路由信息中包含了消息队列，则直接返回该路由信息，如果没有缓存或没有包含消息队列， 则向 nameServer 查询该 topic 路由信息 如果最终未找到路由信息，则抛出异常无法找到主题相关路由信息异常

```
private TopicPublishInfo tryToFindTopicPublishInfo(final String topic) {
    TopicPublishInfo topicPublishInfo = this.topicPublishInfoTable.get(topic);
    if (null == topicPublishInfo || !topicPublishInfo.ok()) {
        this.topicPublishInfoTable.putIfAbsent(topic, new TopicPublishInfo());
        this.mQClientFactory.updateTopicRouteInfoFromNameServer(topic);
        topicPublishInfo = this.topicPublishInfoTable.get(topic);
    }

    if (topicPublishInfo.isHaveTopicRouterInfo() || topicPublishInfo.ok()) {
        return topicPublishInfo;
    } else {
        this.mQClientFactory.updateTopicRouteInfoFromNameServer(topic, isDefault: true, this.defaultMQProducer);
        topicPublishInfo = this.topicPublishInfoTable.get(topic);
        return topicPublishInfo;
    }
}
```


这里我们因为 topic 是空的，所以会向 nameServer 查询这个 topic 的路由信息

```java
public boolean updateTopicRouteInfoFromNameServer(final String topic, boolean isDefault,
    DefaultMQProducer defaultMQProducer) {
    try {
        if (this.lockNamesrv.tryLock(LOCK_TIMEOUT_MILLIS, TimeUnit.MILLISECONDS)) {
            try {
                TopicRouteData topicRouteData;
                if (isDefault && defaultMQProducer != null) {
                    topicRouteData = this.mQClientAPIImpl.getDefaultTopicRouteInfoFromNameServer(defaultMQProducer.getCreateTopicKey(),
                        clientConfig.getMqClientApiTimeout());
                    if (topicRouteData != null) {
                        for (QueueData data : topicRouteData.getQueueDatas()) {
                            int queueNums = Math.min(defaultMQProducer.getDefaultTopicQueueNums(), data.getReadQueueNums());
                            data.setReadQueueNums(queueNums);
                            data.setWriteQueueNums(queueNums);
                        }
                    }
                } else {
                    topicRouteData = this.mQClientAPIImpl.getTopicRouteInfoFromNameServer(topic, clientConfig.getMqClientApiTimeout());
                }
```



```java
public TopicRouteData getTopicRouteInfoFromNameServer(final String topic, final long timeoutMillis)
    throws RemotingException, MQClientException, InterruptedException {

    return getTopicRouteInfoFromNameServer(topic, timeoutMillis,  allowTopicNotExist: true);
}
```



```java
public TopicRouteData getTopicRouteInfoFromNameServer(final String topic, final long timeoutMillis,
    boolean allowTopicNotExist) throws MQClientException, InterruptedException, RemotingTimeoutException, RemotingSendRequestException
    GetRouteInfoRequestHeader requestHeader = new GetRouteInfoRequestHeader();
    requestHeader.setTopic(topic);

    RemotingCommand request = RemotingCommand.createRequestCommand(RequestCode.GET_ROUTEINFO_BY_TOPIC, requestHeader);

    RemotingCommand response = this.remotingClient.invokeSync( addr: null, request, timeoutMillis);
    assert response != null;
    switch (response.getCode()) {
        case ResponseCode.TOPIC_NOT_EXIST: {
            if (allowTopicNotExist) {
                log.warn("get Topic [{}] RouteInfoFromNameServer is not exist value", topic);
            }

            break;
        }
```



```java
    case ResponseCode.SUCCESS: {
        byte[] body = response.getBody();
        if (body != null) {
            return TopicRouteData.decode(body, TopicRouteData.class);
        }
    }
    default:
        break;
}

throw new MQClientException(response.getCode(), response.getRemark());
```


一步步调用，最后会抛出一个客户端异常

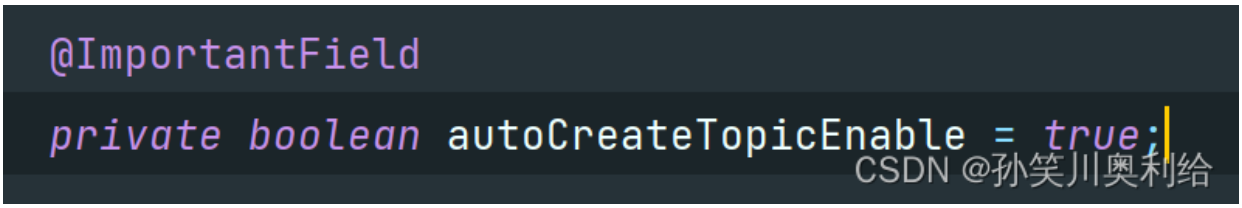
```
} catch (MQClientException e) {
    if (!topic.startsWith(MixAll.RETRY_GROUP_TOPIC_PREFIX) && !topic.equals(TopicValidator.AUTO_CREATE_TOPIC_KEY_TOPIC)) {
        log.warn("updateTopicRouteInfoFromNameServer Exception", e);
    }
```



```
public static final String AUTO_CREATE_TOPIC_KEY_TOPIC = "TBW102"; // Will be created at broker when isAutoCreateTopicEnable
```


发现 isAutoCreateTopicEnable 为 true 时，会自动创建 TBW102 这个 topic ，我们只需要找到在哪里将 isAutoCreateTopicEnable 设置为 true ，并且这个字段在哪里使用即可。

3.



可以看到，这个字段在BrokerConfig中默认被设置为 true

```
{
    if (this.brokerController.getBrokerConfig().isAutoCreateTopicEnable()) {
        String topic = TopicValidator.AUTO_CREATE_TOPIC_KEY_TOPIC;
        TopicConfig topicConfig = new TopicConfig(topic);
        TopicValidator.addSystemTopic(topic);
        topicConfig.setReadQueueNums(this.brokerController.getBrokerConfig()
            .getDefaultTopicQueueNums());
        topicConfig.setWriteQueueNums(this.brokerController.getBrokerConfig()
            .getDefaultTopicQueueNums());
        int perm = PermName.PERM_INHERIT | PermName.PERM_READ | PermName.PERM_WRITE;
        topicConfig.setPerm(perm);
        this.topicConfigTable.put(topicConfig.getTopicName(), topicConfig);
    }
}
```


之后在 broker 初始化时，会一步步执行到上面的代码，并将这个默认 topic 加入到 topicConfigTable 中

4. 我们再回到 producer 的发送逻辑中来，在有了这个 topic 后，我们的发送逻辑就会走另一个分支

```
private TopicPublishInfo tryToFindTopicPublishInfo(final String topic) {
    TopicPublishInfo topicPublishInfo = this.topicPublishInfoTable.get(topic);
    if (null == topicPublishInfo || !topicPublishInfo.ok()) {
        this.topicPublishInfoTable.putIfAbsent(topic, new TopicPublishInfo());
        this.mQClientFactory.updateTopicRouteInfoFromNameServer(topic);
        topicPublishInfo = this.topicPublishInfoTable.get(topic);
    }

    if (topicPublishInfo.isHaveTopicRouterInfo() || topicPublishInfo.ok()) {
        return topicPublishInfo;
    } else {
        this.mQClientFactory.updateTopicRouteInfoFromNameServer(topic,  isDefault: true, this.defaultMQProducer);
        topicPublishInfo = this.topicPublishInfoTable.get(topic);
        return topicPublishInfo;
    }
}
```

```java
try {
    if (this.lockNamesrv.tryLock(LOCK_TIMEOUT_MILLIS, TimeUnit.MILLISECONDS)) {
        try {
            TopicRouteData topicRouteData;
            if (isDefault && defaultMQProducer != null) {
                topicRouteData = this.mQClientAPIImpl.getDefaultTopicRouteInfoFromNameServer(defaultMQProducer.getCreateTopicKey(),
                    clientConfig.getMqClientApiTimeout());
                if (topicRouteData != null) {
                    for (QueueData data : topicRouteData.getQueueDatas()) {
                        int queueNums = Math.min(defaultMQProducer.getDefaultTopicQueueNums(), data.getReadQueueNums());
                        data.setReadQueueNums(queueNums);
                        data.setWriteQueueNums(queueNums);
                    }
                }
```



```java
/**
 * Just for testing or demo program
 */
private String createTopicKey = TopicValidator.AUTO_CREATE_TOPIC_KEY_TOPIC;
```


可以看到，在这里就会把 topic 设置为 AUTO_CREATE_TOPIC_KEY_TOPIC，从而就可以进行正常的发送了

二、元数据的生命周期图