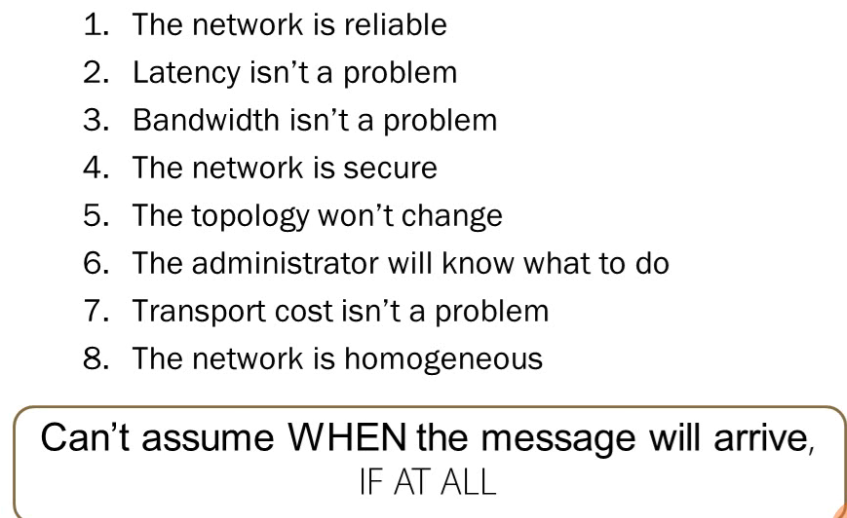
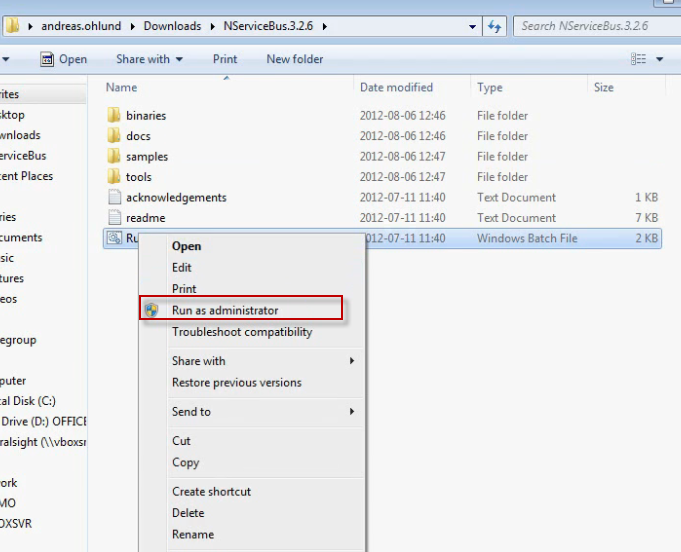
NServiceBus

# 分布式计算的条件

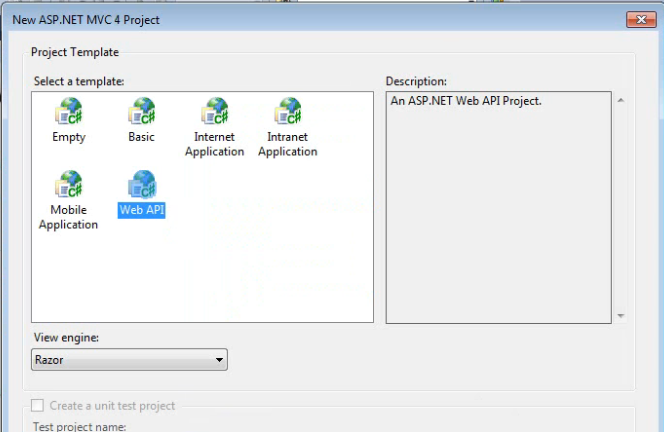


# 安装



# 创建Project

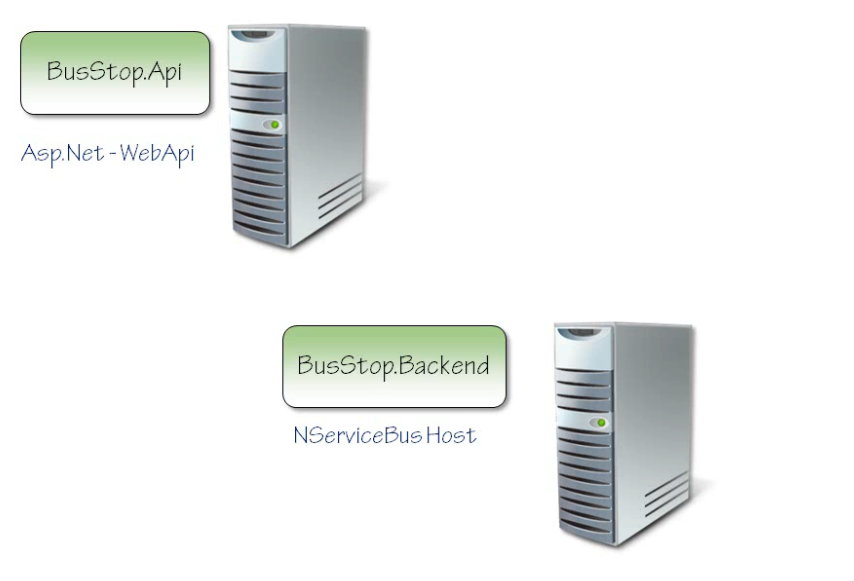
创建一个MVC项目，注意Template要用Web API，因为用NServiceBus了



# 架构

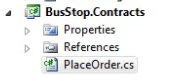
前端BusStop.Api，用来接受用户输入，Host在Web Application里；后端Busstop.Backend，后来处理请求，Host在**NServiceBus Host**中

前端往后端传Message



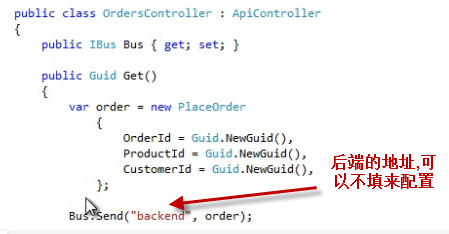
# 创建Message

Message是一种Contract，是前端和后端共享的，因此新建一个Library，在其中创建一个普通的Class作为Message，要实现NServiceBus的IMessage





然后在前端创建Controller去创建Contract并发送，用**IBus**

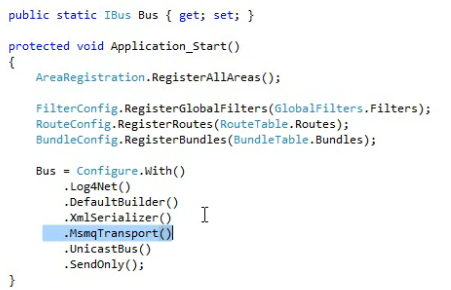


地址可以配置，如果不填@后的主机名，就默认服务端是本地。



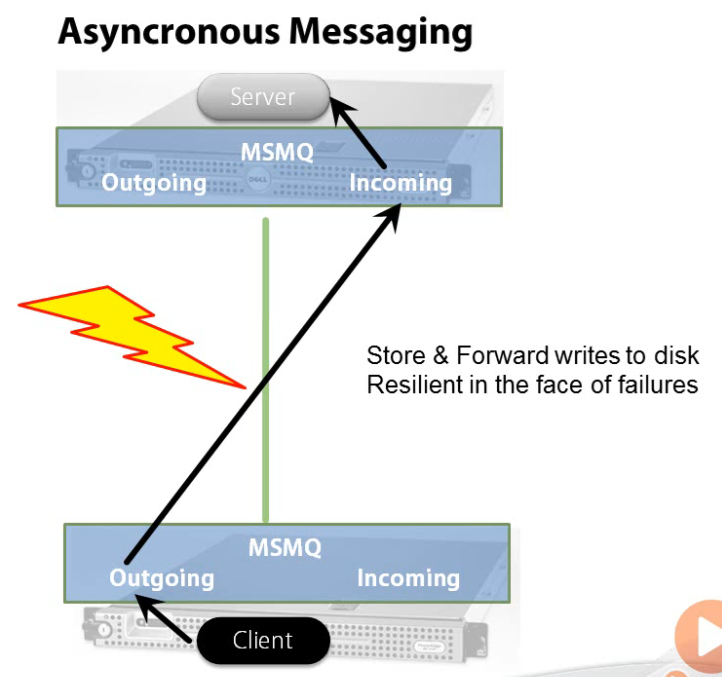
# 创建IBus – 前端的Endpoint

在Global.asax中去创建，IBus是一个Endpoint



# 创建Message Queuing

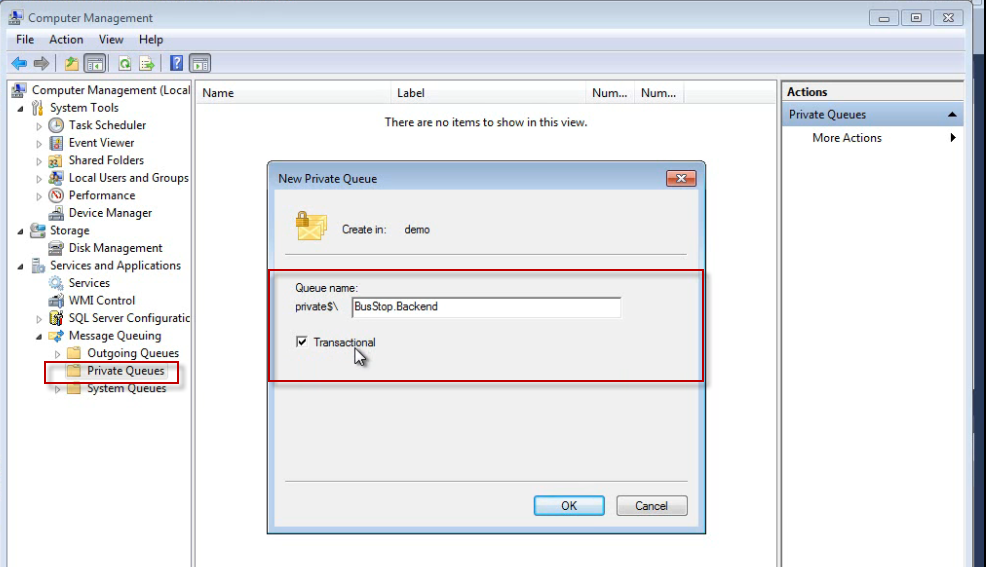
Queuing System负责Message的传递与接收，客户端只要把Message往Queuing System一放就可以做自己的事，Queuing System会处理剩下的事情，包括正常的收发和错误处理。Client和Server接收发送Message的端口就是Endpoint.



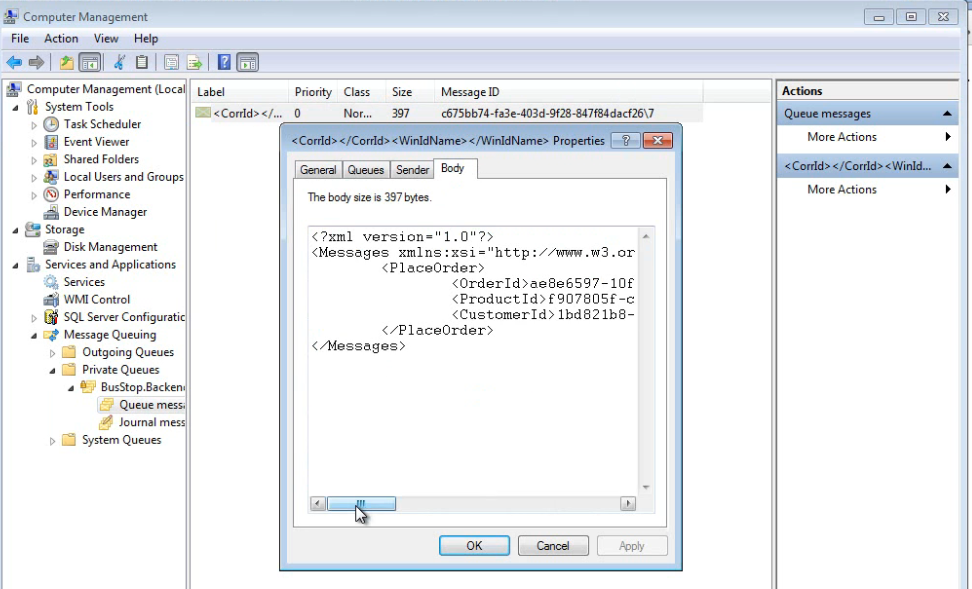
现在Endpoint能把Message发出去了，它是发到Message Queuing, 由它负责去找后端的Endpoint并把Message传给它。

NServiceBus只用Private Queues，名字默认和需要传到的Endpoint一样。

Private Queues里面显示的是收到的Message，Outgoing Queues里面显示的是发出去的Queue

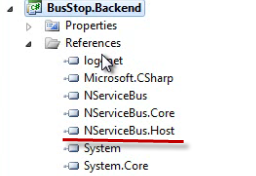


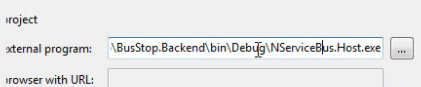
创建完队列后再发Message，就可以看到有Message了，它的Body如下，是XML序列化后的PlaceOrder



# 创建后端

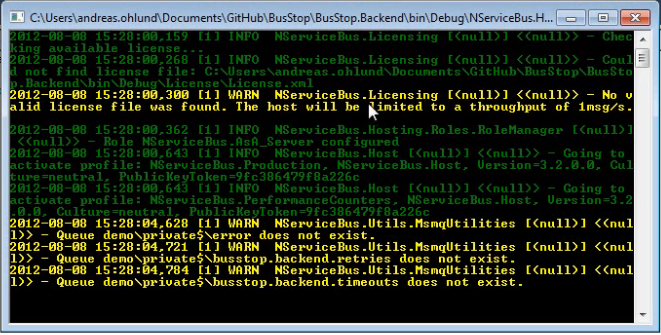
创建一个Class Library Project作为BusStop.Backend， 需要使用NServiceBus.host. NServiceBus.host是作为一个exe被加入到reference中的，启动后就会运行。



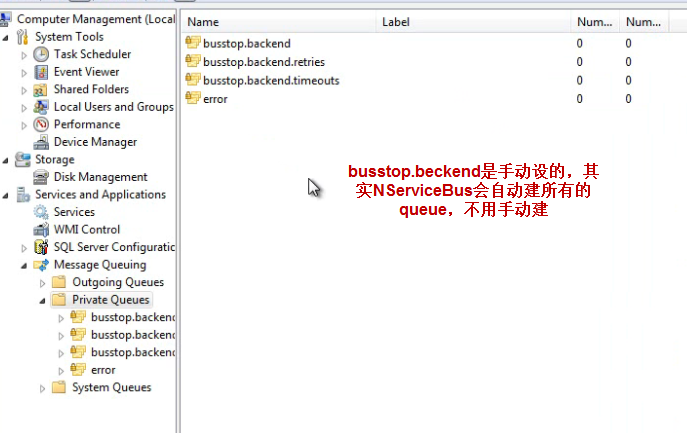


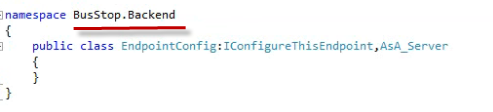
然后为其配置Endpoint，只要在Project中创建一个class即可





运行之后。Message Queue里加了新的Queue，Queue的名字默认和Endpoint一样，而Endpoint的名字默认和Config Class的Namespace一样。

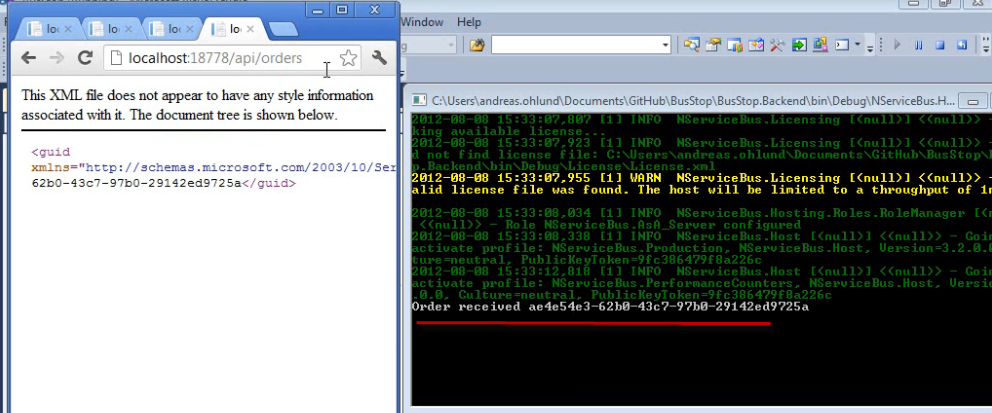




创建PlaceOrderHandler来处理前端的Message

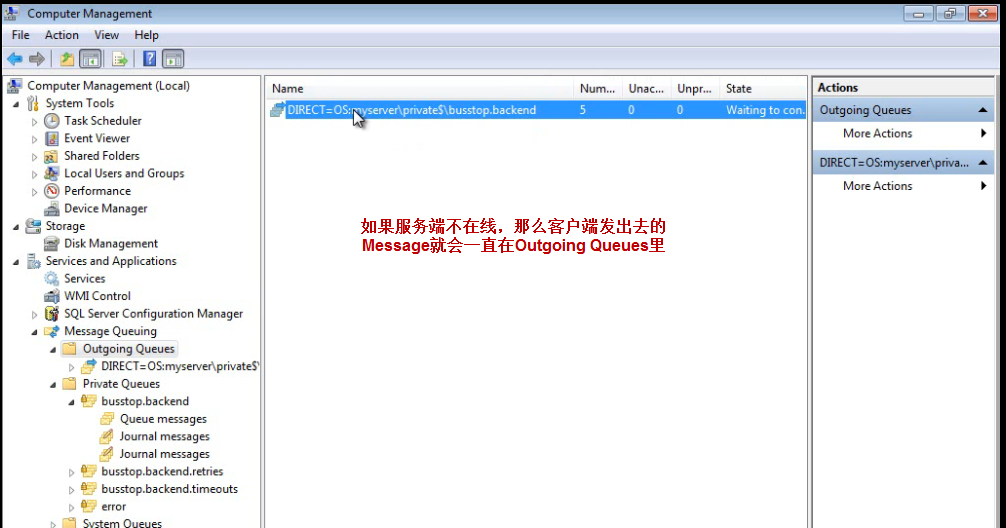


当前端被调用以后，前端发送Message到后端

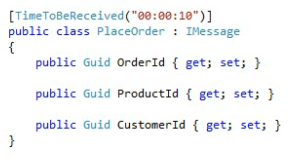


# 处理Message Queue的传输问题

Message的发送是异步的，如果Server端下线了，那么客户端的Message将会在客户端的Outgoing Message Queue里不断积压，最后充满整个硬盘，要等到服务端再打开，才会去处理这些积压的Message。

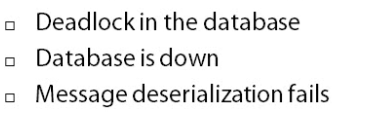


要解决这个问题，就要定期清除Message。方法是在Contract里给Message设置存在时间。



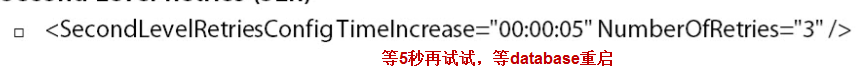
# Exceptions

### 造成Exception的原因

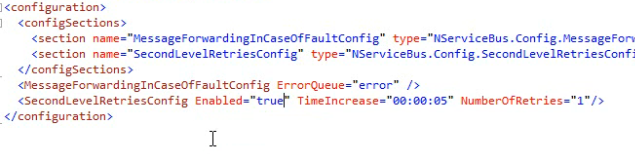


### 解决方法



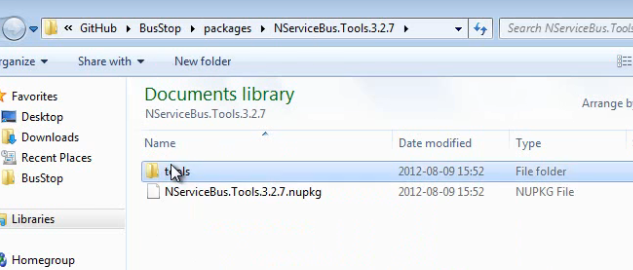
 默认是10秒

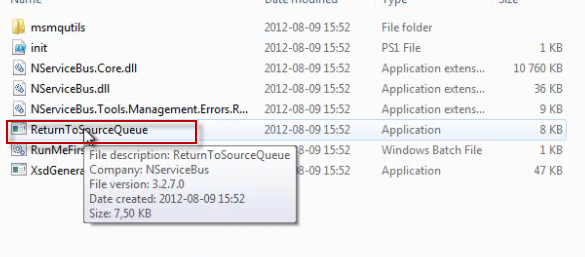




### 从Error Queue中把Error Message拿出来

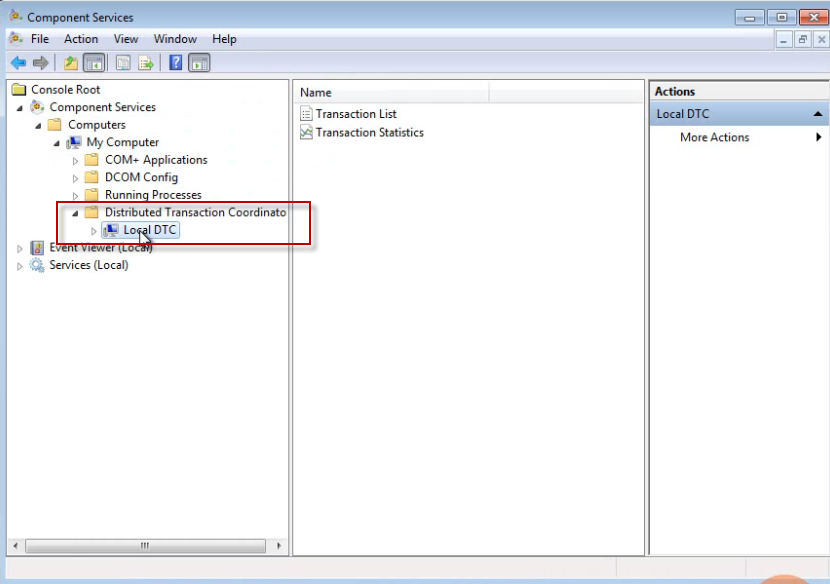


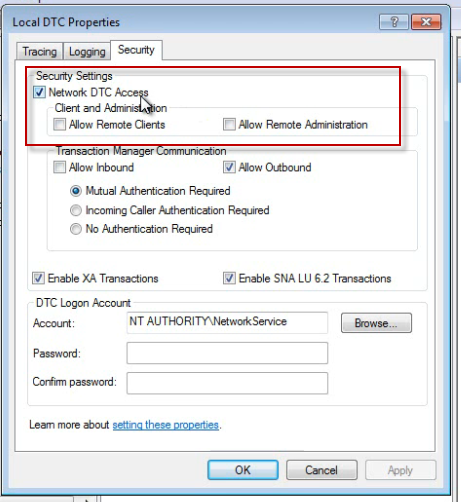




# Transaction

使用的是Windows的Distributed Transaction Coordinator，直接开始 Component Services打开





# RavenDB的使用与Dependency Injection

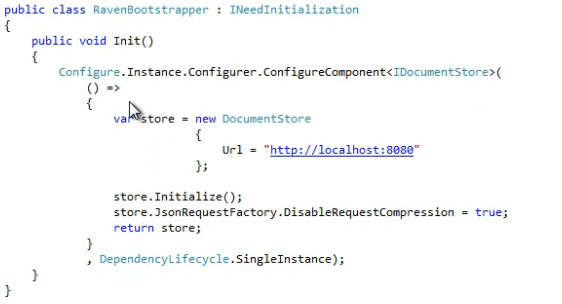
NServiceBus使用RavenDB来存储，它管理页面默认是localhost:8080. RavenDB是一个文件数据库，每一个都是一个Document



使用RavenDb只要创建一个DocumentStore即可



把创建IDocumentStore的方法加进Container，然后在另一个Class里声明IDocumentStore，不用显式new就已经付完值了。

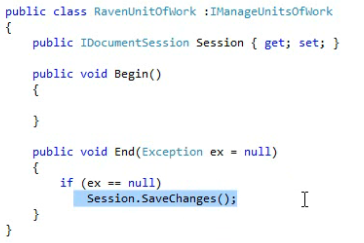


它实现了INeedInitialization，估计NServiceBus会自动调用所有INeedInitialuzation的Init（）



# UnitOfWork

做一个类，实现IManageUnitOfWork，并把它加到Container中





# Authentication