您的潜力，我们的动力

Microsoft
微软（中国）有限公司

# 在Web Service中的异步开发模式

付仲恺
微软特邀开发专家

msdn

MSDN Webcasts

# 课程预习

- Level 300
- 了解Microsoft .NET XML WebServices
- 了解Microsoft Visual C#，包括：
  - 基本的delegate和event代码
  - Attributes
  - Web Service代理

# 议题

- 异步服务概览
- 客户端异步方法
  - 基于代理的开发模式
- 异步服务器端行为
  - Soap one-way方法
  - WSE
  - SoapMSMQ

# 异步方法概览
## 定义



- 异步方法
  - 不必等待方法处理完成而直接返回
  - 可以在服务器和客户端上实现
- 客户端异步方法
  - 完全在客户端实现
  - 通常在proxy或者plumbing层中实现
- 异步服务器端行为
  - 用于改善服务器扩展性
  - 作为同步方法的一部分实现

# 异步方法概览
## 动机



- 对于同步方法：
  - UI线程依赖于方法的实现
    - 方法执行时间过长将导致UI无法及时与用户进行交互
  - 服务器线程受到其它服务器的支配
- 线程是稀缺资源：
  - 在Windows客户端中,每个进程有单一的UI线程
  - 在服务器中,可扩展性依赖于线程的使用
- 对于异步方法：
  - 在用户交互方面：
    - 提供良好的用户体验
    - 及时交互响应
  - 在服务器:
    - 改善可扩展性
    - 将服务器与通讯问题隔离

# 客户端异步方法
## 基于Proxy的方法



- 异步行为的最简单类型
  - 缺省方式
  - Visual Studio和WSDL.exe直接支持
  - 客户端代理包含异步方法
- 效果和影响
  - 改进UI响应度
  - 在服务器不需要实现异步操作
    - 对服务器透明
  - 客户端能够在任何时间选择阻塞

# 客户端异步方法
## 基于Proxy的方法



- WSDL.EXE可以生成异步方法
  - 将方法放入调用队列即返回
  - 通用.NET异步编程模式
    - 通过BeginInvoke/EndInvoke实现
  - 充分利用delegate和IASyncResult特性
    - BeginXxxx和EndXxxx
    - BeginXxxx返回IAsyncResult
      - 客户端可以轮询,等待,或者delegate方法

# 客户端异步模型
## 结束状态查询方法



- 轮询IAsyncResult.IsCompleted以判断处理是否结束
  - 允许客户端继续其它工作
- 使用IAsyncResult.AsyncWaitHandle
  - 允许客户端使用wait处理语义
- 调用delegate方法
  - 等待被运行时调用的delegate
- 阻塞调用End方法
  - 最简单模型

# 客户端异步方法
## 线程池



- 客户端线程池
  - 性能较IAsyncResult略高
  - 对于无响应的请求特别有用
  - delegate能够被异步调用
    - WaitCallback
- QueueUserWorkItem
  - 支持delegate方法调用
  - 可以选择提供参数对象

# 演示

## 客户端异步方法的使用

# 服务器端异步方法

- 在服务器端实现异步方法
- 实现方式：
  - BeginInvoke/EndInvoke
  - Soap one-way方法
  - Web Services增强（WSE）
  - MSMQ

# 服务器端异步方法
## One-Way



- 将单向消息发送到端点
  - 例如：不需要响应的请求
- 问题：
  - 没有返回值
  - 无法判断方法结束的时间
    - 对于结果需要显式通知或者轮询

# 服务器端异步方法
## 通知



- 从端点发送的单向消息
- 客户端通知策略
  - 客户端能够轮询完成状态
    - 要求客户端提供相关令牌
    - 注意可扩展性问题
  - 服务器能够通知客户端
    - WS-Eventing或者类似机制
    - 广播通道

# 服务器端异步方法
## One-Way方法实现



- 使用[SoapDocumentMethod]定义one-way方法
  - [SoapDocumentMethod(OneWay=true)]
  - 数据包反序列化后，服务器端方法即返回
  - 客户端方法不会从调用的服务器端方法中收到返回值
- One-Way方法不适用于
  - 方法需要对结果轮询
    - 其它方法会更加合适
  - 方法需要同步

# 演示

## 服务器端异步one-way方法实现

# 服务器异步方法
## Web Service增强（WSE）



- WSE支持面向消息的编程
  - SoapSender和SoapReceiver基类
  - 支持发送和接收SoapEnvelopes
  - 更多的事务模型
    - 通过SoapClient和SoapService提供
- SoapSender和SoapReceiver
  - 客户端和服务器同时实现
  - 客户端使用SoapSender发送消息
    - 可选择使用SoapReceiver接收消息
  - 服务器使用SoapReceiver接收消息
    - 可选择使用SoapSender发送通知和回应

# 演示

## 利用WSE开发服务器异步方法

# 服务器端异步模式
## 使用MSMQ



- MSMQ用于消息传输
  - 适用于Web Services
  - 具有可靠的异步行为
    - 要求能够在服务中提供可靠的队列
    - 允许多监听者使用单一队列
  - 支持事务
    - 同时支持本地事务和企业服务
    - SWC安装在Windows Server 2003和Windows XP SP2中

# 服务器端异步模式
## Claim-Check模式



- 提供主要的异步解决方案
  - 使用同步方法来接收请求
  - 返回用于获得服务状态的令牌
  - 绝大部分工作表现为异步
- 同步方法:
  - 检查输入参数和业务逻辑规则
  - 事务记录请求
  - 带有令牌的相关请求
- 异步部分
  - 处理请求
  - 发送请求通知

# 服务器端异步模式
## Claim-Check模式实现



- Web Services和MSMQ能够进行整合
  - 使用企业服务来协调事务
    - 数据库事务记录请求
    - MSMQ事务持久地向队列添加请求
  - 异步方法使得事务变得复杂
    - 没有可行的方法确定一个事务的边界
    - 当请求删除队列时,需要保证事务性
- 下面是我们所不希望的:
  - 服务丢失请求
  - 请求处于不一致状态

# 服务器端异步模式
## 使用WSE进行MSMQ操作



- SoapMSMQ
  - 简化使用WSE进行MSMQ操作
  - 开源软件
    - http://www.codeproject.com/useritems/SoapMSMQ.asp
  - 完全支持事务
    - 要求下列claim-check模式
    - 在事务中,请求要被同步初始化
    - 同步阶段排队请求,并且返回令牌
    - 异步阶段处理各个事务
    - 所有持有令牌的请求都保证会被处理
      - 但可能会不成功
    - 支持向客户端发送通知

# 演示

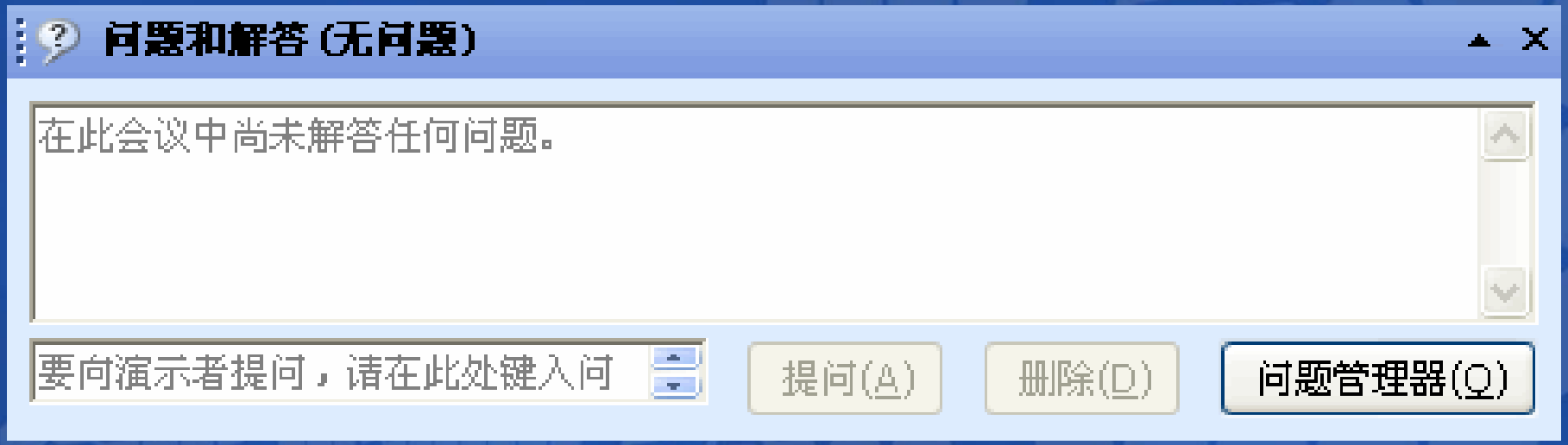
## SoapMSMQ介绍

# 总结

- 异步方法能够：
  - 改善客户端响应
  - 增加服务器端的扩展性
- 对于异步开发的广泛选择：
  - 客户端异步代理方法
  - 使用Soap one-way方法
  - WSE SoapSender和SoapReceiver
  - WSE自定义SoapMSMQ传输

# Q&A

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts