着眼未来设计分布式应用程序

任旻
MinRen@msdncare.com

日程

- 服务的观念
  - 何谓服务
  - 面向服务，面向对象
  - 服务的ABC
- 如何创建分布式的程序
- 面向服务设计分布式应用程序

龙

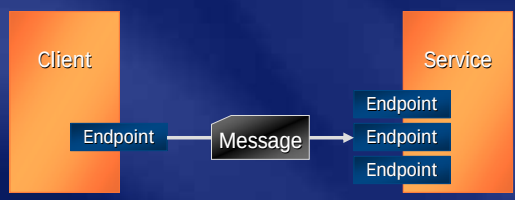
龙能大能小，能升能隐。
大则兴云吞雾，小则隐介藏形，
升则飞腾宇宙之间，隐则潜伏于波涛之间……

OO与SO

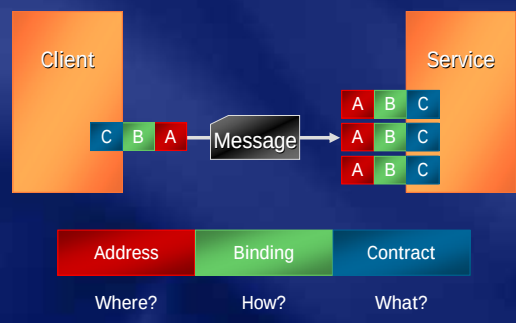
SO Entity	OO Entity
Service contract	interface
Service operation	method
Implementation class	class
Implementation method	method
Data contract	class
Message contract	interface

Services and Clients

Endpoints



Address, Binding, Contract



契约（Contract）

- *"What usually comes first is the contract."*
  —Benjamin Disraeli

Indigo 契约（Contract）

```
[DataContract] public class Problem
{
  // Constructors omitted…
  [DataMember] int i;
  [DataMember] int j;
  [DataMember] FuncType function;
  [DataMember] int problemId;
}

[ServiceContract] interface IDoProblems
{
  [OperationContract(IsOneWay=true)]
  void DoProblem (Problem p);
}
```

Indigo Service

```
public class Math : IDoProblems
{
  public void DoProblem (Problem p)
  {
    if (p.function == FuncType.Add)
      Store (p, p.i + p.j);
    else if (p.function == FuncType.Subtract)
      Store (p, p.i - p.j);
    // …
  }

  private void Store (Problem p, int solution)
  {
    // …
  }
}
```

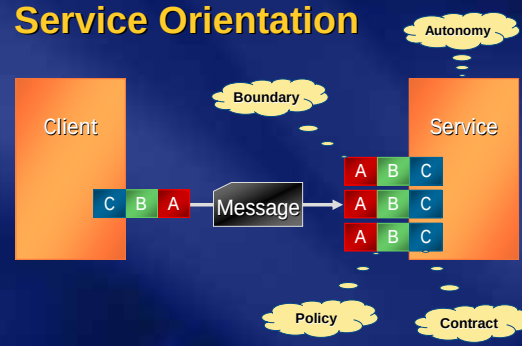
Indigo Service Config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service serviceType="Math">
        <endpoints>
          <endpoint
            address="http://www.mit.edu/MathService.svc"
            bindingtype="BasicProfileBinding"
            contractType="IDoProblems"
          />
        </endpoints>
      </service>
    </services>
  </system.serviceModel>
</configuration
```

Indigo Client

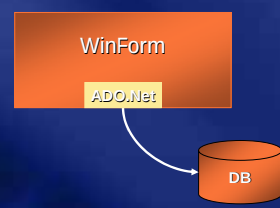
```
// …

  MathProxy math = new MathProxy();
  Problem problem =
            new Problem (3, 4, FuncType.Add, 1001);
  math.DoProblem(problem);
  math.Close();

// …
```

Service Orientation

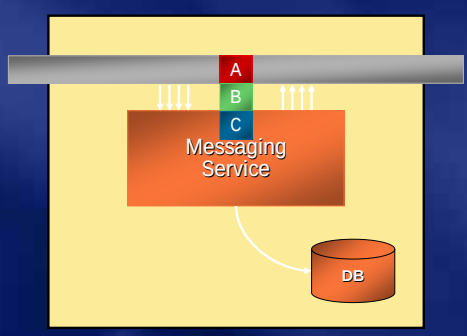


Service Oriented Systems

没有服务的情况下



基于消息的服务



基于调用（Call）的服务

日程

- 服务的观念
- 如何创建分布式的程序
  - 使用现有技术
  - 使用Indigo
  - Indigo与现有技术的对比
- 面向服务设计分布式应用程序

MS现有的技术

- ASMX
- .NET Remoting
- Enterprise Services
- WSE
- MSMQ

- Queued Messaging
- Support for WS-* Specifications
- Distributed Transactions
- .NET – .NET Communication
- Interoperable Web Services

Indigo

- Microsoft Communication Foundation
- Indigo的设计目标：
  在Windows平台上为搭建安全、可靠的面向服务的应用程序提供统一的编程模式和运行环境。
  - 统一的编程模式
  - 互操作性和集成性
  - 面向服务的语言支持

Indigo

```
[ServiceContract] interface IAmAMessageContract
{
  [OperationContract(IsOneWay=true)]
  void DoProblem (Problem p);
}

[ServiceContract] interface IAmACallContract
{
  [OperationContract]
  void DoProblem (Problem p);
}
```

Indigo的兼容性

- 实现“消息服务”可以用：
  - System.Messaging, 队列组件
    - Indigo可以和MSMQ互操作
- 实现“调用服务”可以用：
  - ASP.NET Web 服务(“ASMX”)
    - Indigo’s可以调用现有的Web服务
  - Enterprise Services
    - Indigo’s ComSvcUtil

日程

- 服务的观念
- 如何创建分布式的程序
- 面向服务设计分布式应用程序
  - 分布式应用程序的开发要求
  - 分层
  - 设计服务要如何考虑

分布式应用程序的开发要求

- 性能
- 安全
- 互操作性
- Internet与防火墙
- 配置
- 位置无关性
- 对象生存周期的管理

分层

- 将系统按照逻辑划分为：
  - “表示层”, “业务层”, “数据层”
- 原因
  - 分工明确
  - 结构灵活
  - 部署方便
  - 易于管理
  - 扩展性强
- 注意：“层”并不是指“进程”或“主机”

设计服务要如何考虑

- 在分布式系统中，你的服务扮演什么角色？
- 服务的潜在使用者是谁？
- 是否依赖于其他服务？
- 服务应该实现什么契约（Contract and Schema）？
- 应该选择何种消息的交互模式？
- 需要什么安全级别？
- 是否需要互操作？

我们关心的是业务

- 技术为应用服务，解决实际问题。
- Indigo——Microsoft Communication Foundation 为通讯提供了基础的框架。让程序员将更多的精力集中到业务问题。

Resources

HTTP:
http://msdn.microsoft.com/webservices/
NNTP:
nntp://microsoft.public.windows.developer.winfx.indigo
Email:
stevesw@microsoft.com
Blog:
http://pluralsight.com/blogs/stevesw/

- 目前创建面向服务的应用程序的首选技术是什么？
- Indigo和现有技术的兼容性如何？
- 设计服务要考虑哪些问题？