Bài 3
ADDRESSES (ĐỊA CHỈ) VÀ BINDINGS

Mục lục

Những bài trước chúng ta có nói qua khái niệm về những thành phần của điểm cuối, một trong những thành phần được nhắc tới là các địa chỉ, cách chúng được sử dụng trong điểm cuối và vai trò quan trọng của chúng trong liên lạc. Như đã biết, mọi điểm cuối đều cần có một địa chỉ để các chương trình khác có thể liên lạc với nó. Thực chất một địa chỉ có ý nghĩa là khai báo "tôi đây" cho thế giới bên ngoài.

Ngoài ra trong bài này còn giới thiệu với các bạn về bindings trong Windows Communication Foundation. Bindings cũng giống như các địa chỉ là một trong ba thành phần quan trọng định nghĩa ra một điểm cuối. Các bindings quy định cách mà một điểm cuối liên lạc, đặc biệt là nó chỉ ra những yêu cầu một máy khách cần thực hiện khi kết nối tới một điểm cuối.

1 Địa chỉ trong Windows Communication Foundation

Để dễ hình dung xem địa chỉ trong WCF như thế nào, ta lấy một ví dụ thực tế về địa chỉ như sau:

```
http://localhost:8080/DichVuNhanTin
```

Ở ví dụ trên, ta nhận thấy rằng địa chỉ gồm có 4 phần:

- Giao thức vận chuyển, trong ví dụ trên là `http:`
- Tên của máy thực hiện dịch vụ, trong ví dụ này là `//localhost`
- Đường dẫn tới điểm cuối dịch vụ, trong ví dụ này là `/DichVuNhanTin`
- Phần tuỳ chọn là cổng dịch vụ, trong ví dụ này là 8080. Nếu cổng không được chỉ ra thì giá trị mặc định là cổng 80 cho giao thức `http:` hay cổng 443 cho giao thức `https:`

Để hiểu chi tiết về địa chỉ ta cần hiểu về các kiểu địa chỉ và các định dạng của địa chỉ.

1.1 Address types (Các kiểu địa chỉ)

Trong WCF có một số kiểu địa chỉ sau

1.1.1 Endpoint Address (Địa chỉ điểm cuối)

Địa chỉ điểm cuối giống như ở ví dụ trên, một địa chỉ điểm cuối quy định địa chỉ của một điểm cuối dịch vụ cụ thể. Máy khách (chương trình khách) có thể truy nhật dịch vụ qua địa chỉ điểm cuối. Ví dụ qua địa chỉ sau:

```
http://localhost:8080/DichVuNhanTin
```

Khi máy khách truy nhập dịch vụ thông qua địa chỉ điểm cuối, máy khách có thể nói chuyện với dịch vụ và mọi liên lạc từ dịch vụ và đến dịch vụ đều thực hiện thông qua địa chỉ này.

1.1.2 Base Address (Địa chỉ cơ sở)

Địa chỉ cơ sở cung cấp một cách để xác định một địa chỉ đơn nhất cho một dịch vụ và gán các địa chỉ tương đối cho từng điểm cuối riêng lẻ. Ví dụ, giả sử bạn có một dịch vụ với ba điểm cuối, bạn có thể gán cho dịch vụ đó một địa chỉ cơ sở như sau

```
http://localhost:8080/DichVuNhanTin
```

Với địa chỉ cơ sở được gán cho dịch vụ, bạn có thể gán cho ba điểm cuối các địa chỉ tương đối sau

```
http://localhost:8080/DichVuNhanTin/8753
http://localhost:8080/DichVuNhanTin/8812
http://localhost:8080/DichVuNhanTin/8890
```

1.1.3 MEX Address (Địa chỉ MEX)

Địa chỉ MEX cho phép một máy khách thu thập các thông tin về một dịch vụ nào đó. MEX, nghĩa là metadata exchange (trao đổi siêu dữ liệu), là một địa chỉ điểm cuối HTTP được sử dụng để lấy thông tin về dịch vụ. Ví dụ địa chỉ sau là một địa chỉ MEX:

```
http://localhost:8080/DichVuNhanTin/mex
```

Thông tin về dịch vụ cung cấp cho máy khách thông qua địa chỉ MEX được lấy từ siêu dữ liệu của dịch vụ. Siêu dữ liệu này chính là những mô tả về dịch vụ.

1.2 Address formattings (Các định dạng địa chỉ)

Các địa chỉ điểm cuối được định dạng trên cơ sở của việc lựa chọn cách vận chuyển được sử dụng trong truyền thông. Như đã chỉ ra trong ví dụ đầu tiên của bài này, giao thức vận chuyển là một phần của địa chỉ nhằm xác định và quyết định vận chuyển nào được sử dụng. Điểm nhấn ở đây là chính các nhà lập trình có thể chọn địa chỉ và cách nó được phân phối. Tuy nhiên, điểm quan trọng cần nhớ là với bất kể định dạng nào, máy khách cần phải có khả năng lấy được địa chỉ tới dịch vụ và sử dụng dịch vụ.

Khi phát triển một dịch vụ bạn cần lưu ý các điểm sau:

- Môi trường chứa dịch vụ. Môi trường có thể bắt buộc hoặc đòi hỏi địa chỉ phải được định dạng theo cách này hay cách khác.
- Nơi địa chỉ được xác định. Bạn có các tuỳ chọn để lưu nó ở tệp cấu hình hoặc bạn có thể lưu luôn trong mã nguồn của chương trình của bạn.

Phần này sẽ giới thiệu với bạn các định dạng địa chỉ khác nhau để bạn có thể hình dung về tính linh hoạt và các tuỳ chọn có thể có khi phát triển và phân phối dịch vụ. Bạn sẽ thấy rằng các định dạng địa chỉ gần như giống nhau. Phần lớn các định dạng địa chỉ chứa các phần sau:

- **Phương thức (scheme)**: xác định giao thức liên lạc
- **Tên miền**: xác định tên máy chứa dịch vụ
- **Cổng**: xác định cổng mà dịch vụ thực hiện, nếu không được quy định thì giá trị mặc định là 80
- **Đường dẫn**: đường dẫn tới dịch vụ

Thực chất, để phân biệt các định dạng địa chỉ khác nhau, ta căn cứ vào scheme của địa chỉ.

1.2.1 HTTP Address (Địa chỉ HTTP)

Có lẽ định dạng địa chỉ hay dùng nhất là địa chỉ HTTP. Định dạng của địa chỉ HTTP như sau:

```
http://domainname|hostname [:8080]/path
```

Định dạng địa chỉ HTTP chứa bốn phần. Sau đây là ví dụ của một địa chỉ HTTP sử dụng cả bốn phần:

```
http://localhost:8080/DichVuNhanTin
```

Ví dụ địa chỉ HTTP chứa nhiều đường dẫn

```
http://www.asp.net:8080/Hello/Web
```

1.2.2 HTTPS Address (Địa chỉ HTTPS)

Địa chỉ HTTPS giống như địa chỉ HTTP chỉ khác là nó quy định việc vận chuyển được bảo mật bằng cách sử dụng SSL (Secure Socket Layer), và được quy định bằng `https`. Ngoài scheme là `https`, thì địa chỉ HTTPS không khác gì với địa chỉ HTTP. Ví dụ một địa chỉ HTTPS như sau:

```
https://localhost:8080/DichVuNhanTin
```

Khi sử dụng HTTPS, bạn cần phải lấy một chứng nhận từ một nhà chứng nhận hợp lệ như Verisign hay Thawte.

1.2.3 TCP Address (Địa chỉ TCP)

Định dạng của một địa chỉ TCP như sau:

```
net.tcp://localhost:8080/DichVuNhanTin
```

1.2.4 MSMQ Address (Địa chỉ MSMQ)

Định dạng của địa chỉ MSMQ hơi khác so với các địa chỉ khác. Định dạng như sau:

```
net.msmq://hostname / [private] /queue-name
```

Ví dụ một địa chỉ MSMQ như sau:

```
net.msmq://localhost/msmqshare/DichVuNhanTin
```

Khi sử dụng địa chỉ MSMQ, cần chỉ ra các phần sau địa chỉ MSMQ:

- **Scheme**: Quy định giao thức MSMQ
- **HostName:** là tên đầy đủ của máy thực hiện MSMQ hoặc localhost nếu MSMQ được cài ở máy cục bộ.
- **[private]:** Là phần tuỳ chọn, khi sử dụng nó chứa địa chỉ tới một hàng đợi đích là một hàng đợi riêng. Giá trị này không cần xác định khi truy nhập hàng đợi công cộng.
- **Queue-name**: Tên của hàng đợi MSMQ.

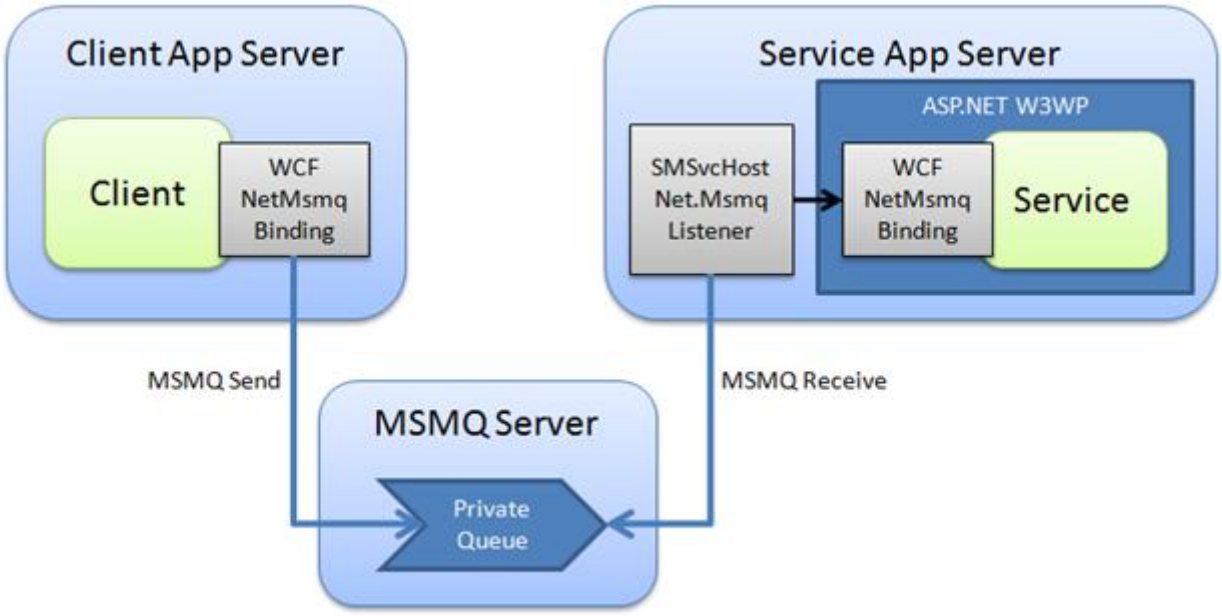

Figure 1 Cách làm việc với MSMQ

1.2.5 Named Pipe Address (Địa chỉ Ống đặt tên)

Định dạng của địa chỉ Named Pipe như sau:

```
net.pipe://localhost/dichvu
```

Trong định dạng địa chỉ này cần lưu ý hai điểm sau:

- Trong định dạng này không có cổng, do địa chỉ Named Pipe không sử dụng cổng
- Truyền thông sử dụng các ống đặt tên không thể "cross-machine", nghĩa là không liên lạc với máy khác.

1.2.6 IIS Address (Địa chỉ IIS)

Địa chỉ IIS cũng giống như địa chỉ MSMQ, hơi khác về định dạng một chút so với các định dạng khác. Do địa chỉ IIS đòi hỏi phải có tên thư mục ảo cũng như tên tệp dịch vụ (.svc). Định dạng của địa chỉ IIS như sau:

```
http://domainname|hostname
[:port]/VirtualDirectory/TênTệpDịchVụ.svc
```

Ví dụ một địa chỉ IIS như sau:

```
http://localhost/dichvu/DichVuNhanTin.svc
```

2 Lập trình với địa chỉ trong Windows Communication Foundation

Khi lập trình với các địa chỉ trong Windows Communication Foundation bạn có thêm sự linh hoạt trong làm việc với các điểm cuối và dịch vụ bởi bạn hoàn toàn có thể lập trình để định nghĩa và xử lý các địa chỉ cơ sở.

Như đã nói, trong thực tế, không mấy khi các nhà lập trình tạo ra các điểm cuối và các địa chỉ điểm cuối thông qua mã nguồn do các địa chỉ và binding bạn sử dụng trong quá trình phát triển dịch vụ thường không giống so với các địa chỉ và binding sẽ sử dụng khi phân phối dịch vụ. Thông thường là ta định nghĩa các điểm cuối và các địa chỉ trong tệp tin cấu hình.

Vậy tại sao ta lại cần lập trình với các địa chỉ? Một trong những lợi điểm của Windows Communication Foundation là tính linh hoạt, và có thể trong một lúc nào đó việc lập trình các địa chỉ là cần thiết. Phần này sẽ giới thiệu với các bạn các mục sau:

- Lớp EndpointAddress, cung cấp cho nhà phát triển cách để tạo ra một địa chỉ mạng duy nhất truy nhập được bởi các máy khách, cho phép chúng liên lạc với dịch vụ của bạn.
- Lập trình các địa chỉ điểm cuối.

2.1 Lớp EndpointAddress

Mục đích của lớp này nhằm cung cấp một địa chỉ mạng duy nhất mà một máy khách sử dụng để liên lạc với một điểm cuối dịch vụ. **EndpointAddress** chứa một URI và các thuộc tính địa chỉ bao gồm:

- Identity (Định danh)
- Các thành phần WSDL
- Optional header collection (Một tập hợp các tuỳ chọn đầu đề). Các tuỳ chọn đầu đề được sử dụng để cung cấp them thông tin chi tiết hơn về cách đánh địa chỉ để định ra hoặc tương tác với điểm cuối. Ví dụ chúng có thể được sử dụng để chỉ ra thực thể nào của một dịch vụ được sử dụng để xử lý bản tin đến từ một người dùng nào đó khi có nhiều thực thể của dịch vụ.

Địa chỉ điểm cuối cho một dịch vụ có thể được xác định hoặc là sử dụng mã nguồn hoặc là được khai báo thông qua tệp tin cấu hình.

Lớp `EndpointAddress` không cài đặt giao diện `ISerializable`, do đó nó không lưu lại được. Để điểm cuối được đưa ra là một phần của contract dịch vụ, nó cần phải lưu lại được, và cần phải tương thích với giao thức địa chỉ Web Service Addressing.

Ví dụ tạo một thực thể của EndpointAddress như sau:

```
EndpointAddress endpointAddress = new

      EndpointAddress("http://localhost:8003/servicemodelsamples/service");
```

2.1.1 Lớp EndpointIdentity

Một trong các thuộc tính của lớp `EndpointAddress` là phần định danh (identity). Phần định danh này được quy định bởi lớp `EndpointIdentity`. Đây là một lớp ảo để cài đặt một định danh cho phép xác thực một điểm cuối bởi các máy khách khi trao đổi bản tin với điểm cuối đó. Ta có các kiểu định danh có thể dùng trong xác thực như sau:

Tên	Mô tả
DnsEndpointIdentity	Quy định định danh DNS của máy chủ. Lớp này quy định định danh cần thiết của máy chủ. Định danh này hợp lệ cho cách xác thực chứng nhận X509 nếu chứng nhận của phía máy chủ chứa một DNS với cùng giá trị. Trong trường hợp này, một máy khách quy định DnsEndpointIdentity là "server1.microsoft.com" cho chế độ xác thực Windows tương đương với xác định SpnEndpointIdentity là "host/server1.microsoft.com"
RsaEndpointIdentity	Quy định một định danh RSA cho điểm cuối dịch vụ. RSA là thuật toán mã hoá sử dụng khoá công cộng được sử dụng cho việc ký và mã hoá bản tin.
SpnEndpointIdentity	Biểu diễn một tên cơ bản của dịch vụ (Service principal name – SPN) cho một định danh khi binding sử dụng Kerberos. Một SPN là một tên theo đó một máy khách có thể xác định ra duy nhất một thực thể của một dịch vụ. Nếu bạn cài đặt nhiều thực thể khác nhau của một dịch vụ trên các máy tính, mỗi thực thể phải có SPN riêng của chúng. Một thực thể của dịch vụ có thể có nhiều SPN nếu có nhiều tên mà máy khách có thể sử dụng để xác thực.
UpnEndpointIdentity	Biểu diễn một tên cơ bản người dùng (user principal name – UPN) cho một định danh được sử dụng khi binding sử dụng chế độ xác thực là SSPINegotiate. Đây là cách tiêu chuẩn để đăng nhập vào một domain Windows. Định dạng là someone@example.com (cho địa chỉ email)
X509CertificateEndpointIdentity	Biểu diễn định danh chứng nhận cho điểm cuối dịch vụ. Các máy khách cố gắng để liên lạc với điểm cuối dịch vụ này nên xác thực với dịch vụ dựa trên chứng nhận được cung cấp bởi định danh của điểm cuối.

Thông thường không cần phải xác định định danh điểm cuối do kiểu xác thực ở máy khách xác định kiểu định danh sử dụng. Tuy nhiên, bạn có thể thay đổi định danh dịch vụ mặc định bằng cách sử dụng thành phần `<identity>` trong tệp cấu hình hoặc bằng cách thiết lập bằng lập trình qua mã nguồn.

Ví dụ sau tạo ra một định danh cho một điểm cuối bằng mã nguồn:

```
ServiceEndpoint sep = MyServiceHost.AddServiceEndPoint(typeof(IMyWCFService),
new WSHttpBinding(), String.Empy);
EndpointAddress ea = new EndpointAddress(new
Uri("http://localhost:8080/testservice/service"),

EndpointIdentity.CreateDnsIdentity("BookOrder.com"));;
sep.Address = ea;
```

Cấu hình cho định danh sử dụng tệp cấu hình như sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service behaviorConfiguration="CalculationService.CalcServiceBehavior"
        name="CalculationService.CalcService">
   <endpoint address="" binding="wsHttpBinding"
contract="CalculationService.ICalcService">
          <identity>
            <dns value="localhost" />
          </identity>
    </endpoint>
    <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8731 /Calculation/CalcService/"
/>
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

2.1.2 Header collection (Tập hợp các đầu đề)

Tập hợp các headers (đầu đề) là một cách theo đó các thông tin thêm vào được cung cấp cho các địa chỉ điểm cuối. Bước đầu tiên là tạo các đầu đề địa chỉ và thêm chúng vào một mảng:

```
AddressHeader ah1 = AddressHeader.CreateAddressHeader("service1",
"http://localhost:8080/testservice /service", 1);
AddressHeader ah2 = AddressHeader.CreateAddressHeader("service2",
"http://localhost:8080/testservice /service", 2);
```

Sau khi các đầu đề địa chỉ được tạo ra, ta thêm các đầu đề này vào trong một mảng:

```
AddressHeader[] addressHeaders = new AddressHeader[2] { ah1, ah2 };
```

Khi tập hợp địa chỉ được tạo ra, tập hợp này có thể được thêm vào trong một thể hiện của lớp `EndpointAddress`. Sử dụng ví dụ về EndpointAddress ở trên, mã nguồn sẽ được sửa lại như sau:

```
EndpointAddress ea = new
      EndpointAddress("http://localhost:8080/testservice/service"),
addressHeaders);
```

2.2 Lập trình với các địa chỉ (address)

2.2.1 Lập trình các địa chỉ cơ sở

Thông qua lớp `ServiceHost`, một địa chỉ cơ sở được xác định như ví dụ sau:

```
Uri ba = new Uri("http://localhost:8080/testservice");
ServiceHost sh = new ServiceHost(typeof(Service), ba));
```

Ví dụ trên là trong trường hợp dịch vụ được cung cấp thông qua một địa chỉ cơ sở đơn nhất. Tuy nhiên trong thực tế, một dịch vụ thông thường có hơn một điểm cuối, và các điểm cuối này có thể sử dụng các giao thức và vận chuyển khác nhau. May mắn là cấu tử của lớp `ServiceHost` chấp nhận nhiều hơn một địa chỉ cơ sở như ví dụ sau:

```
Uri[] bas = new Uri[]
            {
```

```
                new Uri("http://localhost:8080/testservice"),
                new Uri("net.tcp://localhost:8090/testservice")
            };
ServiceHost sh = new ServiceHost(typeof(Service), bas));
```

Sau khi tạo ra một thể hiện của lớp `ServiceHost`, ta cần gọi hàm `Open` để mở ra service và sử dụng.

2.2.2 Lập trình địa chỉ điểm cuối (endpoint address)

Sau khi định nghĩa địa chỉ cơ sở, bạn có thể định nghĩa các địa chỉ điểm cuối có thể là địa chỉ tuyệt đối hoặc địa chỉ tương đối với địa chỉ cơ sở. Việc định nghĩa địa chỉ điểm cuối có thể thực hiện thông qua tệp cấu hình hoặc bằng lập trình.

Bằng tệp tin cấu hình ta có thể định nghĩa một địa chỉ điểm cuối như sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service
        behaviorConfiguration="CalculationService.CalcServiceBehavior"
        name="CalculationService.CalcService">
        <endpoint address="http://localhost:8731/CalcService/"
                  binding="wsHttpBinding"
                  contract="CalculationService.ICalcService">
        </endpoint>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Trong trường hợp định nghĩa hai hay nhiều địa chỉ điểm cuối, ta có tệp tin cấu hình như sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service
        behaviorConfiguration="CalculationService.CalcServiceBehavior"
        name="CalculationService.CalcService">
        <endpoint address="http://localhost:8731/CalcService/"
                  binding="wsHttpBinding"
                  contract="CalculationService.ICalcService">
        </endpoint>
        <endpoint address="http://localhost:8731/MultService/"
                  binding="wsHttpBinding"
                  contract="CalculationService.ICalcService">
        </endpoint>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Khi định nghĩa địa chỉ bởi lập trình, ta có thể định nghĩa như sau:

```
ServiceHost sh = new ServiceHost(typeof(), new WsHttpBinding(),String.Empty);
```

3 Giới thiệu về Bindings trong Windows Communication Foundation

Các bindings là phương pháp theo đó các chi tiết trong truyền thông được xác định để tạo kết nối tới điểm cuối dịch vụ WCF. Các bindings trong WCF có thể thay đổi mức độ phức tạp. Các mức độ này có

thể từ đơn giản nhất cho tới cực kỳ phức tạp. Khi định nghĩa một binding, bạn cần chỉ ra các thông tin ở trong các lĩnh vực sau:

- **Protocol (Giao thức)**: Định nghĩa các thông tin cần sử dụng trong binding ví như tính bảo mật, khả năng thực hiện giao dịch, hoặc khả năng truyền bản tin một cách tin cậy.
- **Transport (Vận chuyển)**: Định nghĩa giao thức cơ bản được sử dụng trong truyền thông
- **Encoding (mã hoá)**: Định nghĩa việc mã hoá được sử dụng cho các bản tin trong quá trình liên lạc.

Để sử dụng binding, cần thực hiện hai bước sau:

1. Chọn một binding từ danh sách định nghĩa trước của các bindings trong WCF mà bạn sẽ sử dụng cho điểm cuối. Khi chọn một binding có sẵn, bạn có lựa chọn là giữ lại hoặc thay đổi các thuộc tính của binding để phù hợp với yêu cầu của bạn.
2. Tạo một điểm cuối sử dụng binding mà bạn vừa chọn hoặc tạo ra.

Lợi điểm của WCF là nó định nghĩa một số các binding phù hợp với hầu hết các yêu cầu phát sinh trong quá trình phát triển dịch vụ. Bạn có thể theo dõi danh sách các binding đó ở bảng sau. Giá trị ở trong ngoặc đơn là các giá trị mặc định cho từng tính năng của binding cụ thể.

Binding	Tính làm việc liên môi trường	Bảo mật	Phiên	Giao dịch
BasicHttpBinding	Basic Profile 1.1	(none), Transport, Message	(None)	(None)
WSHttpBinding	WS	Transport, (Message), Mixed	(None),Transport, Reliable	(None), Yes
WSDualHttpBinding	WS	(Message)	Reliable	(none), Yes
WSFederationHttpBinding	WS	(Message)	(None), Reliable	(none), Yes
NetTcpBinding	.NET	(Transport), Message	Reliable, (Transport)	(none), Yes
NetNamedPipeBinding	.NET	(Transport)	None, (Transport)	(none), Yes
NetMsmqBinding	.NET	Message, (Transport), Both	(None)	(None), Yes
NetPeerTcpBinding	Peer	(Transport)	(None)	None
MsmqIntegrationBinding	MSMQ	(Transport)	(None)	(none), Yes

Trong bảng trên, các tính năng được định nghĩa như sau:

- **Tính làm việc liên môi trường**: Định nghĩa giao thức hay công nghệ mà binding sẽ sử dụng để đảm bảo việc trao đổi và sử dụng thông tin.
- **Bảo mật**: Xác định cách bảo mật kênh trao đổi thông tin
- **Phiên**: Xác định việc hỗ trợ contract phiên
- **Giao dịch**: Xác định việc cho phép hay không các giao dịch (transaction)

Để xác định xem khi nào nên sử dụng binding nào, bạn nên theo sơ đồ dưới đây:

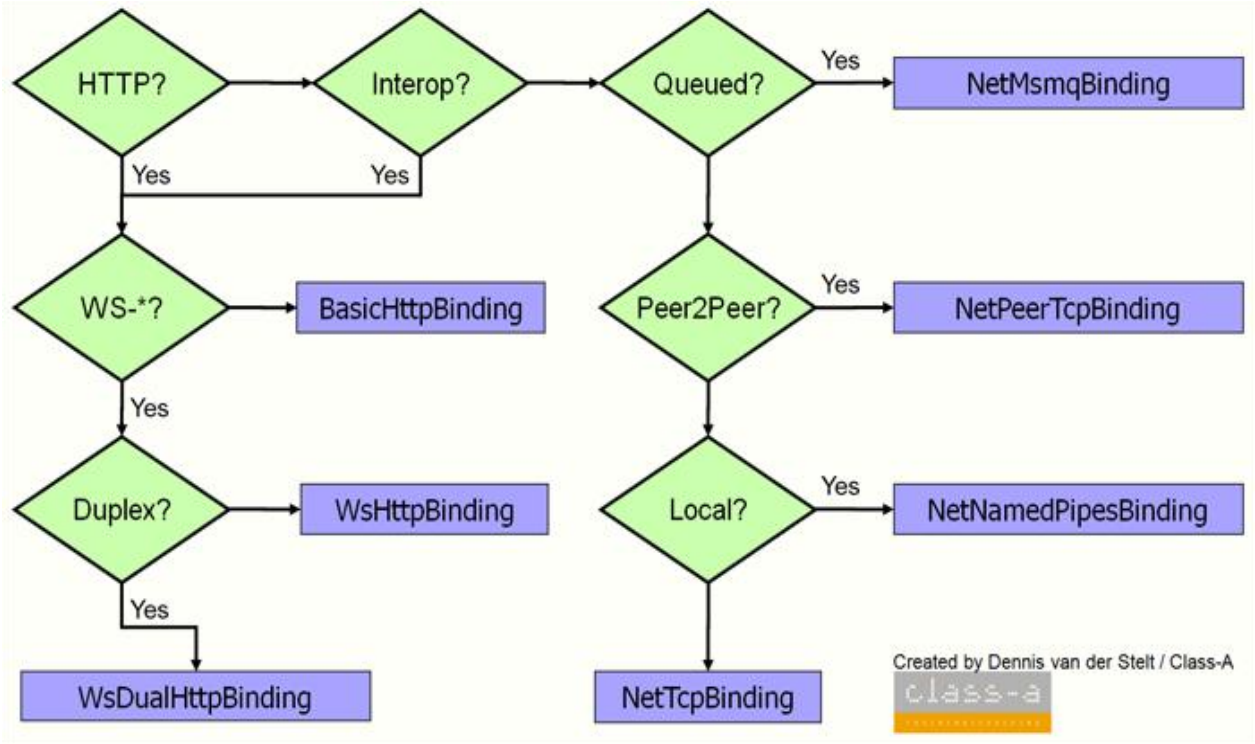

Figure 2 Nên sử dụng binding nào (ảnh từ blog của Dennis van der Stelt)

Bạn nên tham khảo thêm bài viết ở trang web này http://weblogs.asp.net/spano/archive/2007/10/02/choosing-the-right-wcf-binding.aspx để biết thêm các thông tin khi lựa chọng binding phù hợp.

4 Lập trình các Bindings

Các binding cũng giống như các địa chỉ có thể được tạo ra bằng cách sử dụng tập tin cấu hình, hoặc sử dụng lập trình. Ví dụ sau đây định nghĩa một BasicHttpBinding bằng cách sử dụng tệp tin cấu hình:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name = "basichttpbind"
        closeTimeout = "00:00:30"
        openTimeout = "00:00:30"
        sendTimeout = "00:00:30"
        receiveTimeout = "00:00:30">
        </binding>
      </basicHttpBinding>
    </bindings>
  </system.serviceModel>
</configuration>
```

Trong trường hợp lập trình, ta có thể thực hiện như sau:

```
Uri ba = new Uri("http://localhost:8080/WroxService");
BasicHttpBinding binding = new BasicHttpBinding();
binding.CloseTimeout = 30000;
binding.OpenTimeout = 30000;
binding.SendTimeout = 30000;
binding.ReceiveTimeout = 30000;
```

```
ServiceHost host = new ServiceHost(serviceType, baseAddresses);
sh.AddServiceEndpoint("", binding, ba);
```

5 Câu hỏi ôn tập

1. Liệt kê các dạng địa chỉ được sử dụng trong WCF

Có các dạng địa chỉ sau:

- Địa chỉ HTTP
- Địa chỉ HTTPS
- Địa chỉ TCP
- Địa chỉ MSMQ
- Địa chỉ Ống đặt tên
- Địa chỉ IIS

6 Tài liệu tham khảo

1. Basic WCF programming (URL: http://msdn.microsoft.com/en-us/library/ms731067.aspx)
2. Lớp EndpointAddress (URL: http://msdn.microsoft.com/en-us/library/system.servicemodel.endpointaddress.aspx)
3. WCF Binding decision (URL: http://bloggingabout.net/blogs/dennis/archive/2006/12/01/WCF-Binding-decision-chart.aspx)