**1. 직렬화**

|  |
| --- |
| WCF의 서비스는 자신이 제공하는 다양한 기능을 클라이언트에게 알리기 위한 방법으로 계약을 사용한다.  - 서비스 계약 : 서비스의 인터페이스 명시  - 데이터 계약 : 서비스가 사용하는 커스텀 데이터 타입 명시 |

**서비스 계약**

* 작업 계약 : 닷넷의 메소드
* 인터페이스 계약과 클래스 계약 : 인터페이스 외 클래스 또한 계약에 포함될 수 있다.

( ASMX 통신 메커니즘을 WCF 기반 시스템으로 마이그레이션 하는 경우 클래스 계약이 사용될 수 있다.)

**서비스 계약 버전 관리**

* 클라이언트는 서비스가 제공하는 메타데이터를 통해 서비스를 호출할 때 필요한 매개변수, 타입, 리턴 타입 등을 알아낸다. 만약, 서비스의 계약이 변경되었다면 서비스의 메타데이터도 변경될 것이지만, 클라이언트가 아직 새로운 메타데이터를 통해 프록시를 업데이트하지 않았을 경우 문제를 발생할 수도 있다.
* 서비스 계약이 변경되는 사항

기존 메소드에 새로운 매개변수 추가

기존 메소드에서 매개변수 삭제

기존 메소드의 매개변수 타입 변경

기존 메소드의 리턴 타입 변경

기존 메소드의 제거

새로운 메소드의 추가

* 일반적 권고 사항

불특정 다수의 클라이언트에 대해 서비스를 제공하고 있다면 서비스는 클라이언트의 업데이트에 대한 제어권을 가지고 있지 않다.

한정된 클라이언트에 대해 서비스를 제공한다면 서비스는 클라이언트에 대한 업데이트 제어권을 가질 수 있다. 즉 클라이언트의 업데이트를 강요할 수 있다.

* 계약 상속을 통한 버전관리가 가능하다.

**메시지 계약**

* 서비스와 클라이언트가 주고받는 SOAP 메시지에 대해 보다 세밀한 제어가 가능하게 해준다.
* WCF를 사용하지 않는 타 플랫폼의 서비스 혹은 클라이언트를 위해 SOAP 메시지의 표현을 보다 세부적으로 제어할 필요가 생기곤 한다.
* 이 기능을 사용하면 WCF 서비스를 호출할 때 사용되는 SOAP 메시지 Body 요소에 포함되는 요소들의 구성이나 요소 이름을 제어 할 수 있을 뿐만 아니라 SOAP 메시지 Header요소에 추가적인 정보를 포함시킬 수 있다.

**데이터 계약**

* 서비스에서 사용되는 데이터 타입을 서비스 외부에 알리기 위한 방법으로 XML 스키마를 사용하며, XML 스키마를 통해 서비스와 클라이언트는 동일한 데이터 타입을 공유할 수 있다.
* 서비스의 계약 내에 사용된 닷넷 데이터 타입은 XML 스키마에 의해 XML 데이터로 변환되며, 이 XML 데이터는 SOAP 메시지 내에 포함되어 서비스와 클라이언트가 주고받게 된다.
* WCF는 닷넷 데이터 타입을 서비스가 사용하는 XML 스키마로 변환하는 다양한 기능을 제공한다. 단, 사용자 정의 데이터 타입을 XML 스키마로 변환하고 데이터 객체를 SOAP 메시지의 XML로 전환하는 방법을 WCF에게 알려줄 필요가 있는데, 이를 데이터 계약이라고 한다.
* 즉 데이터 계약은 WCF 기반으로 구현된 서비스가 사용하는 닷넷 데이터 타입을 XML 스키마로 변환하고 이 데이터 타입을 SOAP 메시지로 직렬화하는 데 필요한 정보를 제공하는 것을 말한다.

**WCF Serializer**

* 직렬화는 어떤 객체를 순차적으로 바이트의 나열로 바꾸는 것을 말한다.
* 역직렬화는 바이트들의 나열로 직렬화된 객체가 다시 원래의 객체로 복원되는 것이다.
* 닷넷 프레임워크에서 제공되는 직렬화는 크게 두 종류이다.

객체를 XML 표현으로 바꾸어 주는 XmlSerializer : XML 스키마와 호환성을 위해 설계되었다. 따라서 XML 스키마로 표현이 불가능한 닷넷 객체는 직렬화 될 수 없다.

닷넷 리모팅에서 사용하는 포매터 :모든 닷넷 타입을 직렬화하여 바이트의 나열로 표시할 수 있도록 하는 목적으로 설계되었다. 따라서 모든 닷넷의 데이터 타입을 바이너리 형식 혹은 SOAP XML 형식으로 바꿀 수 있다.

* 즉, XmlSerializer 는 XML 스키마라는 인터넷 표준이 필요한 웹 서비스에서 닷넷 객체들을 직렬화하는 데 사용된 반면 닷넷 포매터는 닷넷 애플리케이션끼리의 원격 호출인 닷넷 리모팅에 사용된다.
* WCF의 DataContractSerializer

명시적으로 직렬화할 데이터 타입과 데이터 타입 내의 필드 속성 지정이 가능하다.

데이터 맴버순서 : 알파벳

직렬화 대상 : DataMember 특성이 표시된 필드/속성

상호 운영성을 높이기 위해 XML 스키마를 일부 제한한다.

**2. 바인딩과 메시지 전송**

**메시지 인코딩**

3종류의 메시지 인코더가 제공된다.

1) 텍스트

2) MTOM(Message Transmission Optimization Mechanism)

3) 바이너리 인코더

- 텍스트 인코더 : 메모리 상에 XML DOM(Document Object Model) 형태로 존재하는 SOAP 메시지를 문자열 텍스트 그대로 내보내는 것을 말한다. 상호 운영성은 대단히 높지만 텍스트를 파싱해야 하는 부담과 커다란 파일과 같은 바이너리 데이터를 전송할 때는 효율이 크게 떨어진다.

- MTOM : SOAP 메시지를 MIME(Multi-part Message Encoding)을 이용하여 SOAP 메시지 텍스트와 바이너리로 분리한 트랜스포트 메시지를 구성함으로써 바이너리 데이터를 웹 서비스를 통해 전송하고 수신할 때 효율을 높인 인코딩이다. 비록 SOAP 메시지 자체는 텍스트 형태를 띄지만 SOAP 메시지 내에 존재할 수 있는 대량의 바이너리를 별도의 MIME 파트로 분리할 수 있기 때문에 서비스를 통해 파일 전송하는 경우에 유용하다. W3C 표준이기 때문에 상호 운영성도 매우 뛰어나다는 장점을 가지고 있지만 성능면에서는 텍스트 인코딩보다 크게 뛰어난 편은 아니다.

- 바이너리 인코딩 : 메모리 상에 XML DOM 형태로 존재하는 SOAP 메시지를 곧바로 WCF 고유의 바이너리의 형태로 인코딩한다. 오로지 WCF 만이 바이너리 메시지를 이해하기 때문에 상호 운용성이 크게 떨어지지만 어떤 인코딩보다 작은 크기의 메시지를 생성하므로 네트워크 대역폭을 아낄 수 있을 뿐만 아니라 텍스트 XML을 파싱하는 것보다 빠르게 SOAP 메시지를 재구성할 수 있다.

|  |  |  |
| --- | --- | --- |
| 바인딩 | 지원하는 인코딩 | 디폴트 인코딩 |
| BasicHttpBinding | Text, MTOM | Text |
| WSHttpBinding | Text, MTOM | Text |
| WS2007HttpBinding | Text, MTOM | Text |
| WSDualHttpBinding | Text, MTOM | Text |
| WSFederationHttpBinding | Text, MTOM | Text |
| WS2007FederationHttpBinding | Text, MTOM | Text |
| NetTcpBinding | Binanry | Binanry |
| NetnamedPipeBinding | Binanry | Binanry |
| NetpeerTcpBinding | Binanry | Binanry |
| NetMsmqBinding | Binanry | Binanry |
| MsmqIntegrationBinding | N/A | N/A |

* 메시지를 수신할 때 수신된 메시지는 트랜스포트 버퍼 내에 쌓이게 된다. 커다란 크기의 메시지를 전송하여 서비스를 마비시키는 보안 공격을 막기 위해 버퍼 크기는 64KB로 제약되어 있다. 변경 가능하다.

(MaxReceivedMessageSize / MaxArrayLength )

* 파일 업로드/다운로드와 같이 큰 바이너리 데이터를 매개변수나 리턴값으로 사용한다면 MTOM 인코딩이나 바이너리 인코딩을 사용하는 것이 좋다.
* MIME 타입은 자체 헤더들의 크기가 1~2KB에 달하는 경우도 있기 때문에 작은 크기의 SOAP 메시지에 대해서는 MTOM 인코딩이 오히려 메시지 크기를 키우게 된다.
* 따라서 필요에 따라 서비스의 계약을 두 개로 분리하여 하나는 텍스트 인코딩을, 다른 하나는 MTOM 인코딩을 사용하도록 설정해야 하는 경우도 발생한다.

**메시지 스트리밍**

* 클라이언트가 서비스를 호출하거나 서비스가 결과값을 반환할 때 생성되는 SOAP 메시지는 메모리 상에 버퍼링된다. 최종적으로 SOAP 메시지가 생성되고 인코딩된 후에야 메시지가 트랜스포트를 통해 서비스 혹은 클라이언트에게 전송된다.
* 이렇나 버퍼링은 효율적이며 성능이 우수하지만 커다란 SOAP 메시지를 전송하는 상황에서는 많은 양의 메모리를 소모하는 비효율적인 상황으로 치닫기도 한다.
* WCF는 버퍼링 모드 외 스트림 모드를 지원한다.
* 스트림 모드는 메모리 상의 버퍼를 사용하지 않고 스트림을 사용하여 점진적으로 SOAP 메시지를 작성하고 인코딩한다. 스트림 모드를 사용함으로써 커다란 메시지를 전송하더라도 적은 양의 메모리를 사용하도록 서비스를 구성할 수 있다.
* 클라이언트가 스트림 객체에 대해 close메소드를 호출하면 서비스 측의 스트림은 close되며 할당된 자원을 해제한다.
* 스트림 모드는 상대적으로 적은 메모리만을 소요하여 대용량 파일 혹은 데이터 셋을 전송할 수 있다 뿐이지 실제 전송되는 바이트 수를 줄여주지는 않는다.

**3. 메타 데이터 및 호스팅 환경**

WCF에서 지원하는 메타데이터 관련 표준은 2가지 이다.

* WSDL(Web Service Description Language) : HTTP Get방식을 사용하여 WSDL을 알아내고 서비스의 계약, 바인딩, 주소 정보를 알아낼 수 있다.
* WS-MetadataExchange : HTTP 프로토코뿐만 아니라 TCP, 명명된 파이프를 통해 보다 안전하게 WSDL을 클라이언트에게 제공할 수 있다.

WCF가 지원하는 서비스 호스팅 환경은 3가지 이다.

* IIS 호스트 : IIS 프로세스 재활용, 웹 가든, Health 모니터링 등의 안정적인 호스팅 환경 제공, HTTP 프로토콜만 허용된다.
* WAS 호스트 : Windows Vista와 Windows 2008 Server 가 제공한다. HTTP, TCP, MSMQ, 명명된 파이프 프로토콜을 모두 사용할 수 있다. IIS가 제공하는 안정성 기능을 모두 사용할 수 있다.
* 독립 호스트

**4. WCF 서비스 제공 방식**

|  |
| --- |
| 1) 요청응답(Request-Reply) : 기본적인 클라이언트 호출 방식  2) 단방향 호출  3) 비동기 호출  4) 서비스 콜백(웹 서비스 표준이 아닌 WCF만의 기능)  \*) 상호 운영성이 중요한 서비스라면 양방향 호출 방식의 바인딩을 사용해서는 안된다.  콜백을 포함하는 서비스 계약을 정의해서도 안 된다.  자바 기반의 클라이언트나 WCF를 사용하지 않는 닷넷 클라이언트도 양방향 호출  방식이나 콜백을 이해하지 못할 가능성이 높다.    따라서 전통적인 방식을 사용하여 클라이언트가 주기적으로 사건의 발생을 모니터링 하는  폴링 방식을 사용해야 한다.  즉, 주기적으로 클라이언트가 서비스를 호출하여 새로운 공지 사항이 발생했는지 등을  알아봐야 한다는 것이다. |

**2) 단방향 호출**

클라이언트가 서비스로부터 응답을 전혀 받지 않는 호출

단순히 네트워크 상에 SOAP 요청 메시지를 보내는 것 이상을 포함

- 메시지가 트랜스포트 상에서 정상적으로 전송되지 않으면 CommunicationException이

발생할 수 있다.

- 이러한 상황은 어떠한 바인딩을 사용하고 있는가와 세션 사용 여부 등에 따라 약간의 차이가 날 수 있다.

- 단방향 호출을 수행했음에도 불구하고 클라이언트가 대기 상태에 빠지는 상황도 발생할 수 있다.

WCF는 요청-응답 모델을 기본적으로 사용하기 때문에 속성값을 변경하여 단방향 호출메서드 임을 표현해야 함(단방향 메소드의 리턴값은 void : 규약)

- 서비스 메타데이터를 통해 클라이언트에게 알려지게 됨

- 단방향 호출을 위해 클라이언트가 해주어야 할 일은 없슴

고려사항

- 단방향 호출과 트랜스포트 채널

1) 클라이언트가 단방향 호출을 수행하면 호출을 위한 SOAP 요청 메시지가 작성되고 바인딩이 사용하는 트랜스포트(HTTP, TCP, 명명된 파이프 등)를 통해 서비스에 이 메시지를 전송하려고 시도한다.

\*) 이 때 클라이언트가 트랜스포트를 통해 서비스에 접근할 수 없다면 단방향 호출일지라도 예외가 발생할 수 있다.

ex) 아직 구동되지 않은 서비스에 대해 단방향 호출을 수행하면 클라이언트는 ServerTooBusyException 예외를 받게 된다.

ex) 클라이언트에게 알려지지 않은 계약 변경(메소드 이름, 매개변수 변경)등이 있을 때 발생할 수 있다.

2) SOAP 요청 메시지가 서비스에 성공적으로 전달되었고 그 요청이 서비스에 의해 처리가 시작된 경우에 단방향 호출은 성공적으로 수행된다고 할 수 있다.

ex) BasicHttpBinding을 사용하여 단방향 호출 수행

SOAP 요청 메시지를 담은 HTTP 요청이 서버에 전송

SOAP 요청 메시지가 WCF 런타임에 의해 성공적으로 처리를 시작하면 HTTP

202Accepted를 반환하게 됨

* 즉, HTTP 트랜스포트에 의해 요청-응답이 발생한 것이지 WCF 서비스가 서비스 호출 결과를 담은 SOAP 응답 메시지를 작성한 것이 아님.

\*) 주의점

단방향 호출시에 발생하는 트랜스포트상의 오류는 어떤 트랜스포트를 사용하는가에 따라

클라이언트에 영향을 줄 수도 있고 그렇지 않을 수도 있다.

클라이언트 입장에서는 일관된 결과를 얻을 수 있어야 하겠지만 실제 서비스와

클라이언트가 사용하는 바인딩과 트랜스포트에 따라서 서로 다른 결과를 얻을 수 있기

때문에 주의를 해야 한다.

- 단방향 호출과 클라이언트 블록

단방향 호출을 수행할 때 클라이언트는 일반적으로 블록되지 않는다. 하지만 서비스와 클라이언트가 사용하는 트랜스포트 채널, 바인딩 및 바인딩 설정에 의해 클라이언트가 블록되는 상황이 발생할 수 있다.

WCF에서는 단방향 호출이라도 HTTP 트랜스포트를 사용한다면 HTTP 응답이 발생하게 된다. 이 때 응답이 늦게 된다면 클라이언트는 대기상태에 빠지게 된다.

- 단방향 호출과 서비스 예외

서비스에서 처리되지 않은 예외가 서비스 측의 채널을 폴트 상태로 빠지게 할 수 있다.

서비스 측의 채널이 폴트 상태로 바뀌면 그 채널과 연결된 클라이언트 프록시를 통해 추가적인 서비스 호출은 실패하게 된다.

|  |
| --- |
| [ 정리 ]  - 단방향 호출을 수행하면 트랜스포트 수준에서 서비스 호출 요청이 서비스에 도달했는지 확인한다.  - 이러한 이유에서 클라이언트가 서비스를 호출하는 동안 블록되거나 예상하지 못한 예외를 받을 수도 있지만 정상적인 대다수의 상황에서는 클라이언트가 단방향 호출을 하는 동안 블록되지 않는다.  - 따라서 서비스의 수행 결과가 중요하지 않거나 어떤 상황에 대해 이벤트를 통지하는 용도로 단방향 호출을 사용하는 것이 좋다. ( 로깅이나 통지 그리고 콜백에서 단방향 호출을 많이 사용한다.)  - 클라이언트가 서비스를 호출할 때 블록이 되는 것을 방지하는 목적 하나만을 위해 단방향 호출을 사용하는 것은 옳지 못하다. 단방향 호출도 클라이언트가 블록되는 상황이 발생할 수 있기 때문이다.  - 긴 수행 시간을 갖는 서비스를 호출하는 동안 클라이언트가 다른 작업을 수행해야 한다면 비동기 호출을 선택해야 할 것이다. |

**3) 비동기 호출**

WCF는 클라이언트가 서비스에 대해 비동기 호출을 손쉽게 수행할 수 있도록 서비스 프록시 코드를 생성해 주기 때문에 보다 적은 노력으로 비동기 호출을 수행할 수 있다.

WCF 비동기 호출은 순수하게 클라이언트 측에만 적용되는 기능이다.

서비스는 클라이언트가 동기적으로 혹은 비동기적으로 호출했는지 알아낼 방법이 없다.

서비스는 클라이언트의 동기 혹은 비동기 호출 방식에 상관없이 동일하게 작동할 뿐이다.

[ 흐름 ]

1) 클라이언트에서 비동기 메서드 호출

2) 작업 스레드에게 서비스 호출을 맡기고 곧바로 제어를 클라이언트에게 반환한다.

[주의점]

- 비동기 호출을 동시에 여러 개 수행할 때 이들은 모두 하나의 프록시를 통해 호출된다.

만약 서비스와 클라이언트가 세션을 사용한다면 이들 호출은 동일 세션 내에서 동시에

서비스를 호출하게 된다.

만약 서비스가 PerCall 서비스라면 병행성에 관련하여 고려해야 할 사항이 적겠지만

PerSession 서비스나 Singleton 서비스라면 병행성 모드에 따라서 서비스에 동기화 관련

코드를 작성해야 할 수도 있다.

- 서비스 호출 결과를 알아낼 때이다.

\*) 비동기 호출 폴링

WCF 클라이언트는 IAsyncResult 객체를 통해 비동기 호출 결과를 분리 확인할 수 있다.

비동기 호출의 완료를 지속적으로 감시하는 것을 완료를 감시하는 폴링 작업이라 한다.

보통 짧은 시간 동안만 블록시키면서 지속적으로 메시지 펌프를 수행함

\*) 비동기 호출 콜백

완료 콜백 메소드는 비동기 호출을 시작한 스레드와 다른 스레드에서 수행된다.

비동기 호출 도중 에러가 발생하면 예외에 대해 콜백 메소드가 호출된다. 심지어 서비스가

시작되지 않아서 서비스와 접속이 안 된 상황에서 발생한 에외들에 대해서도 완료 콜백

메소드는 호출된다.

|  |
| --- |
| [정리 ] - 비동기 호출은 여러 가지 프로그램의 응답성과 동시에 많은 긍정적인 요소를 가지고 있다.  - 복잡한 스레드 동기화 문제를 유발할 수 있고, 오류가 발생했을 때 이를 해결하는 것이 복잡해 질 수 있다.  - 또한 제어 흐름이 직관적이지 않다. |

**4) 서비스 콜백**

WCF가 ASP.NET 웹 서비스와 다른 점 중 하나는 서비스가 클라이언트를 호출할 수 있다는 점이다.

HTTP, TCP, 명명된 파이프를 통해 서비스가 클라이언트를 호출하는 것을 모두 지원한다.

더 이상 클라이언트는 서비스의 상태를 모니터링하기 위해 주기적으로 서비스를 호출하는 폴링 기법을 사용하지 않아도 된다.

- 양방향 호출

WCF가 콜백 기능을 지원하기 위해 사용하는 클라이언트/서비스 호출 방식이 양방향 호출(Duplex Call) 이다.

양방향 호출방식은 클라이언트가 SOAP 요청 메시지를 전송하면 서비스가 처리 결과를 담은 SOAP 응답 메시지를 클라이언트에게 ‘전송’하는 것이다.

요청-응답 호출과의 차이점은 양방향 호출시 SOAP 응답 메시지를 클라이언트에게 전송하기 때문에 클라이언트가 SOAP 메시지를 수신하기 위해 트랜스포트 채널을 리스닝해야만 한다.

클라이언트가 트랜스포트 채널을 리스닝하지 않으면 양방향 호출에서 서비스의 결과를 수신할 수 없다.

|  |
| --- |
| [ 요청 – 응답(Request-Reply Call) ]  Client --------------(SOAP Request)-----------🡪 Server  🡨------------(SOAP Response)-----------  [ 단방향(One-Way Call) ]  Client ---------------(SOAP Request)-------------🡪 Server  🡨--------(HTTP 202 Accepted or TCP ACK -----  [ 양방향( Duplex Call) ]  Client ---------------(SOAP Request)---------------🡪 Server  🡨-----------(HTTP 202 Acccepted or TCP ACK) ----  🡨--------------(SOAP Request)---------------------  -----------(HTTP 202 Accepted or TCT ACK)----------- |

\*) 기본적인 HTTP 트랜스포트는 양방향 호출 방식을 사용할 수 없다.

\*) WCF는 HTTP 트랜스포트에서도 양방향 통신이 가능하도록 해 주는 WSDualHttpBinding

을 제공한다.

해당 바인딩은 서비스와클라이언트가 모두 HTTP 서버 역할을 하게 한다.

서비스가 클라이언트를 호출하는 콜백을 수행하기 위해서는 콜백 계약이 선언되어야 한다.

클라이언트는 서비스가 사용하는 콜백 계약을 구현하는 콜백 핸들러 객체를 구성해야 하며 콜백을 지원하는 양방향 채널을 구성하여 프록시 객체를 작성해야 한다

콜백 호출

서비스가 콜백을 통해 클라이언트를 호출할 필요가 있다면 서비스는 클라이언트 콜백에 대한 프록시 객체를 알아내야만 한다.

서비스가 클라이언트 콜백을 호출하기 위해서는 콜백에 대한 프록시 객체를 필요로 한다.

- 바인딩 : 클라이언트가 서비스를 호출했을 때 사용한 바인딩이 그대로 사용됨

- 계약 : ServiceContract 특성의 CallbackContract 속성을 통해 이미 서비스가 알고 있다.

- 주소 : 동적으로 생성되는 콜백 주소를 알아내는 것이 필요하다.

WCF에서 양방향 호출이 이루어지면 클라이언트와 서비스는 항상 SOAP 메시지

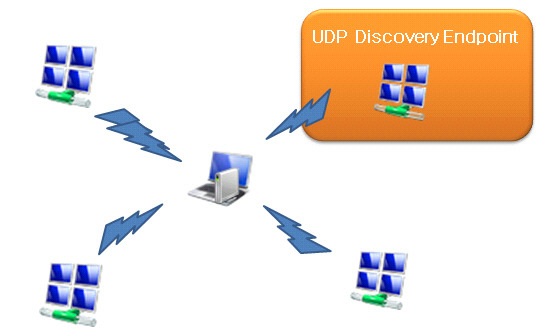
헤더에 상대방이 응답 SOAP 메시지를 전송할 때 사용될 주소를 명시하도록 되어

있다. 응답을 위한 URL은 WS-Addressing 표준에 의해 SOAP <Header> 요소의

<ReplyTo> 요소에 기록된다.

|  |
| --- |
| [ 정리 ]  - 만약 서비스가 서비스 메소드 내에서 콜백을 호출할 때는 재진입 상황이 발생할 수 있기 때문에 주의해야 한다.  - WCF 기본 병행성 모드인 Single 병행성 모드를 사용하는 서비스가 서비스 메소드 내에서 콜백을 수행할 때는 100% 데드락 상황이 발생한다.  - 일반적으로 콜백에 단방향 호출을 적용하여 데드락을 피할 수 있다. 콜백 호출에서 콜백 호출 결과는 중요하지 않은 경우가 매우 많기 때문이다.  - Reentrant 병행성 모드를 사용함 으로서 데드락을 방지할 수 있다.  - 만약, 서비스가 콜백을 호출하고자 할 때 클라이언트가 종료되었다면, 혹은 서비스가 콜백을 수행할 때 클라이언트가 콜백을 수신하지 못하는 상황이 발생된다면?  🡺 콜백 호출 가능 여부를 검사하여 세션이 닫힌 클라이언트에 대해 콜백 수행을 수행하지 않도록 구조화 한다.  🡺 클라이언트가 서비스에게 콜백 호출을 하지 말도록 의사를 밝힐 수도 있다.  ( 이벤트 디자인 패턴 구조) |

**5. Discovery(ad-hoc)**



[ Service Discovery 도식도]

* 서비스 디스커버리 기능은 WCF 에 추가된 흥미로운 기능중 하나로써 자신이 원하는 서비스와 동일한 시그네처를 가진 서비스를 인접한 UDP 노드(UDP multicasting)를 탐색해서 호출하는 기능이다.
* 서비스를 동적으로 게시하거나 검색할 수 있다. 클라이언트는 runtime시 조건에 맞는 서비스를 검색할 수 있고 검색된 Endpoints를 동적으로 이용할 수 있게 된다.
* 해당 udp 노드를 탐색하는 패킷에 응답하기 위해 WCF 서비스는 udpdiscoveryendpoint 를 지원하고 있다.
* Service Discovery 또한 config 파일에 정의하며 , 요소는 다음과 같다.

|  |  |
| --- | --- |
| 요소명 | 설명 |
| serviceDiscovery | 해당서버를 찾을수 있는 서버로 설정한다. |
| udpDiscoveryEndpoint | UDP Endpoint 를 노출시켜 UDP 스캔시 해당 서비스를 노출되게 한다 |
| FindCriteria | 서비스를 스캔할 조건을 설정하고 UDP 를 스캔 |
| FindResponse | 검색된 EndPoint 를 가져온다 |

http://cafe.naver.com/silverlight3040.cafe?iframe\_url=/ArticleRead.nhn%3Farticleid=35&