WCF의 메시지 교환 패턴

1.요청/응답

- wcf 메시지 교환의 가장 기본적인 패턴

using System.ServiceModel;

[ServiceContract]

public interface ICalculator

{

[OperationContract]

// It would be equivalent to write explicitly:

// [OperationContract(IsOneWay=false)]

int Add(int a, int b);

[OperationContract]

int Subtract(int a, int b);

int Multiply(int a, int b)

}

2.단방향

- 출력이 없는 작업(반환 값이 없고 out 또는 ref 매개 변수가 없음)은 IsOneWay 속성을 true로 설정하여 단방향으로 지정. 메서드에 대한 단방향 계약을 원하는 경우 특성 속성 값을 명시적으로 true로 지정

[ServiceContract]

public interface ICalculatorSession

{

[OperationContract(IsOneWay=true)]

void Clear();

[OperationContract(IsOneWay = true)]

void AddTo(double n);

[OperationContract(IsOneWay = true)]

void SubtractFrom(double n);

[OperationContract(IsOneWay = true)]

void MultiplyBy(double n);

[OperationContract(IsOneWay = true)]

void DivideBy(double n);

[OperationContract]

double Equals();

}

-

3. 이중 계약(동시 송수신?)ㄴ

3-1.인터페이스

- 기본 인터페이스의 CallbackContract 속성을 콜백 인터페이스의 형식으로 설정하여 이중 계약에 두 개의 인터페이스를 연결.

[ServiceContract(SessionMode=SessionMode.Required,

CallbackContract=typeof(ICalculatorDuplexCallback))]

public interface ICalculatorDuplex

{

[OperationContract(IsOneWay=true)]

void Clear();

[OperationContract(IsOneWay = true)]

void AddTo(double n);

[OperationContract(IsOneWay = true)]

void SubtractFrom(double n);

[OperationContract(IsOneWay = true)]

void MultiplyBy(double n);

[OperationContract(IsOneWay = true)]

void DivideBy(double n);

}

public interface ICalculatorDuplexCallback

{

[OperationContract(IsOneWay = true)]

void Result(double result);

[OperationContract(IsOneWay = true)]

void Equation(string eqn);

}

3-2. 클래스

기본 계약의 서비스 구현에서 콜백 인터페이스에 대한 변수를 선언.

변수를 OperationContext 클래스의 GetCallbackChannel 메서드에서 반환한 개체 참조로 설정

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]

public class CalculatorService : ICalculatorDuplex

{

double result;

string equation;

ICalculatorDuplexCallback callback = null;

public CalculatorService()

{

result = 0.0D;

equation = result.ToString();

callback = OperationContext.Current.GetCallbackChannel<ICalculatorDuplexCallback>();

}

public void Clear()

{

callback.Equation(equation + " = " + result.ToString());

result = 0.0D;

equation = result.ToString();

}

public void AddTo(double n)

{

result += n;

equation += " + " + n.ToString();

callback.Equals(result);

}

public void SubtractFrom(double n)

{

result -= n;

equation += " - " + n.ToString();

callback.Equals(result);

}

public void MultiplyBy(double n)

{

result \*= n;

equation += " \* " + n.ToString();

callback.Equals(result);

}

public void DivideBy(double n)

{

result /= n;

equation += " / " + n.ToString();

callback.Equals(result);

}

}