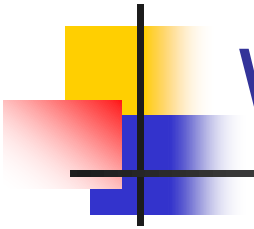
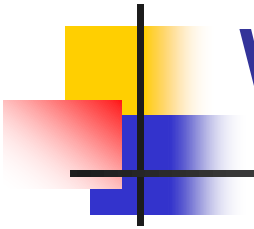
Windows Communication Foundation (WCF)

Windows Communication Foundation (WCF)



WCF

- WCF is a framework for building service-oriented applications.
- Using WCF, you can send data as asynchronous messages from one service endpoint to another.
- A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application.
- An endpoint can be a client of a service that requests data from a service endpoint.
- The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.



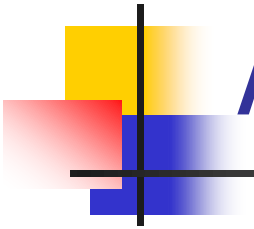
WCF

- WCF enables you to create *service oriented* applications.
- Service-oriented architecture (SOA) is the reliance on Web services to send and receive data.
- The services have the general advantage of being loosely-coupled instead of hard-coded from one application to another.
- A loosely-coupled relationship implies that any client created on any platform can connect to any service as long as the essential contracts are met.

Endpoints: Addresses, Bindings, and Contracts

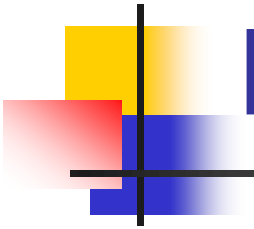
- All communication with a Windows Communication Foundation (WCF) service occurs through the *endpoints* of the service. Endpoints provide clients access to the functionality offered by a WCF service.

- Each endpoint consists of three properties:
  - An address that indicates where the endpoint can be found.
  - A binding that specifies how a client can communicate with the endpoint.
  - A contract that identifies the operations available.



Address

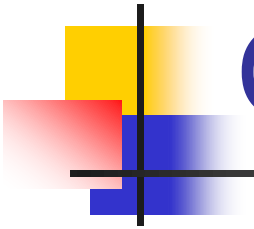
- The address uniquely identifies the endpoint and tells potential consumers of the service where it is located. It is represented in the WCF object model by the EndpointAddress class.
- An EndpointAddress class contains:
  - A Uri property, which represents the address of the service.
  - An Identity property, which represents the security identity of the service and a collection of optional message headers. The optional message headers are used to provide additional and more detailed addressing information to identify or interact with the endpoint.



Binding

The binding specifies how to communicate with the endpoint. This includes:

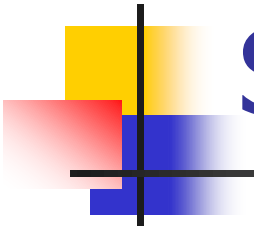
- The transport protocol to use (for example, TCP or HTTP).
- The encoding to use for the messages (for example, text or binary).
- The necessary security requirements (for example, SSL or SOAP message security).



Contract

The contract outlines what functionality the endpoint exposes to the client. A contract specifies:

- What operations can be called by a client.
- The form of the message.
- The type of input parameters or data required to call the operation.
- What type of processing or response message the client can expect.



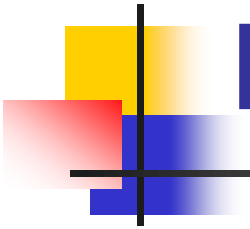
Service Contract

- A service contract specifies what an endpoint communicates to the outside world.
- At a more concrete level, it is a statement about a set of specific messages organized into basic message exchange patterns (MEPs), such as request/reply, one-way, and duplex.
- If a service contract is a logically related set of message exchanges, a service operation is a single message exchange.
- For example, a **Hello** operation must obviously accept one message (so the caller can announce the greeting) and may or may not return a message (depending upon the courtesy of the operation)

Service Contract

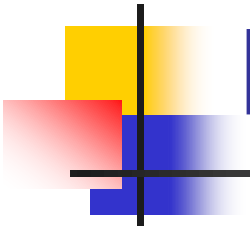
- To create a service contract you define an interface with related methods representative of a collection of service operations, and then decorate the interface with the *ServiceContract* Attribute to indicate it is a service contract.
- Methods in the interface that should be included in the service contract are decorated with the OperationContract Attribute.

```
[ServiceContract()]
public interface ISimpleCalculator {
  [OperationContract()]
  int Add(int num1, int num2);
}
```



Data Contract

- A *data contract* is a formal agreement between a service and a client that abstractly describes the data to be exchanged.
- That is, to communicate, the client and the service do not have to share the same types, only the same data contracts.
- A data contract precisely defines, for each parameter or return type, what data is serialized (turned into XML) to be exchanged.



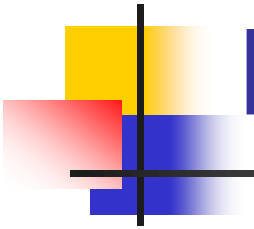
Data Contract

```
[DataContract(Namespace="http://www.something.com/students/")]
public class StudentInfo
{
    [DataMember(Name="StudentId", Order=1)]
    public int ID;

    [DataMember(Name="FirstName", Order=2)]
    public string FName;

    [DataMember(Name="LastName", Order=3)]
    public string LName;
}
```

Message Contract

Data Contract in WCF is an agreement between parties (i.e. a service and a client) that describes what type of data will be exchanged between them? On the other hand, ***Message Contract*** describes the structure of SOAP message that is passed between parties (i.e. a service and a client). Using Data Contract, we actually control the contents (i.e. Payload data or message body) of a SOAP message while Message Contract provides complete control over structure of SOAP message.

```
[MessageContract]
public class BankingTransaction
{
        [MessageHeader]
        public Operation operation;
         [MessageBodyMember]
        private Account sourceAccount;
         [MessageBodyMember]
        public int amount;
}
```



Message Contract

- You can use the Message class as an input parameter of an operation, the return value of an operation, or both. If Message is used anywhere in an operation, the following restrictions apply:
  - The operation cannot have any out or ref parameters.
  - There cannot be more than one input parameter. **If the parameter is present, it must be either Message or a message contract type**.
  - The return type must be either void, Message, or a message contract type.