WCF Session-3 : EndPoints,Contracts and Serialization

Gursharan Singh Virdi

WCF: EndPoints and Contracts

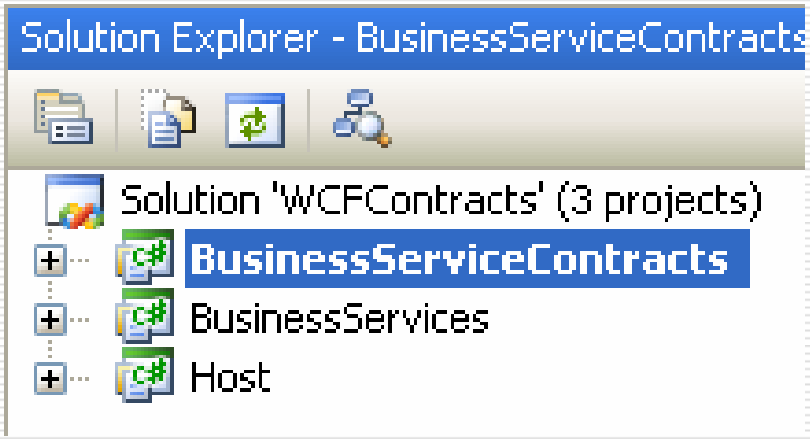
- This session we will take a look at designing endpoints with respect to service contracts.
- We will create a expose a service which will implement multiple contracts.
- Overview of how Serialization works(Second part after these set of slide…).

WCF: EndPoints and Contracts

- **Rules to remember**
  - A service can have multiple contracts.
  - To expose the operations in the contracts , the service needs to provide as many endpoints.
  - 1 Endpoint is associated with one contract only.
  - In this case we will have 2 service contracts :
    - 1. CustomerService contract
    - 2.OrderService contract
  - The idea here is to expose the CustomerService contract over tcp binding and OrderService contract on both tcp and http bindings.
  - The service contracts are part of the same service or exposed by the same service.

WCF: EndPoints and Contracts

- **DEMO App breakup(refer to solution explorer snapshot)**
  - Keep all Service contracts in a separate project ie.BusinessServiceContracts
  - Reference the service contract in the project where the actual service is implemented ie. BusinessServices
  - Host the service in a separate service host.



WCF: Configuration Files

- **The Application**
  - The Service and the Service contracts are simply one function implementations.
  - The real magic happens in the config file of the host app.

```
<service name="BusinessServices.DataService" behaviorConfiguration="serviceBehaviour">
  <host>
    <baseAddresses>
      <add baseAddress="http://localhost:8002"/>
      <add baseAddress="net.tcp://localhost:9000"/>
    </baseAddresses>
  </host>

  <!-- Service Endpoints -->
  <!-- HTTP and TCP for ICustomerService -->
  <endpoint address="DataService"
            binding="basicHttpBinding"
            contract="BusinessServiceContracts.ICustomerService" />

  <endpoint address="DataService"
          binding="netTcpBinding"
          contract="BusinessServiceContracts.ICustomerService" />

  <!-- TCP for ICustomerService -->
  <endpoint address="Admin"
          binding="netTcpBinding"
          contract="BusinessServiceContracts.IOrderService" />
```

WCF: Configuration Files

- **Extracting metadata for the client**
  - Leave the service host running.
  - Access the metadata endpoint at http://localhost:8002/?wsdl
  - The wsdl will have all the binding for your clients(refer to code)

```
- <wsdl:binding name="NetTcpBinding_IOrderService" type="tns:IOrderService">
    <wsp:PolicyReference URI="#NetTcpBinding_IOrderService_policy" />
    <soap12:binding transport="http://schemas.microsoft.com/soap/tcp" />
  - <wsdl:operation name="GetOrder">
      <soap12:operation soapAction="http://tempuri.org/IOrderService/GetOrder" style="document" />
    - <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
    - <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="DataService">
  - <wsdl:port name="BasicHttpBinding_ICustomerService" binding="tns:BasicHttpBinding_ICustomerService">
      <soap:address location="http://localhost:8002/DataService" />
    </wsdl:port>
  + <wsdl:port name="NetTcpBinding_ICustomerService" binding="tns:NetTcpBinding_ICustomerService">
  - <wsdl:port name="NetTcpBinding_IOrderService" binding="tns:NetTcpBinding_IOrderService">
      <soap12:address location="net.tcp://localhost:9000/Admin" />
    - <wsa10:EndpointReference>
        <wsa10:Address>net.tcp://localhost:9000/Admin</wsa10:Address>
      - <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
          <Upn>singhgu01@doma.corpdir.biz</Upn>
        </Identity>
      </wsa10:EndpointReference>
    </wsdl:port>
  </wsdl:service>
```

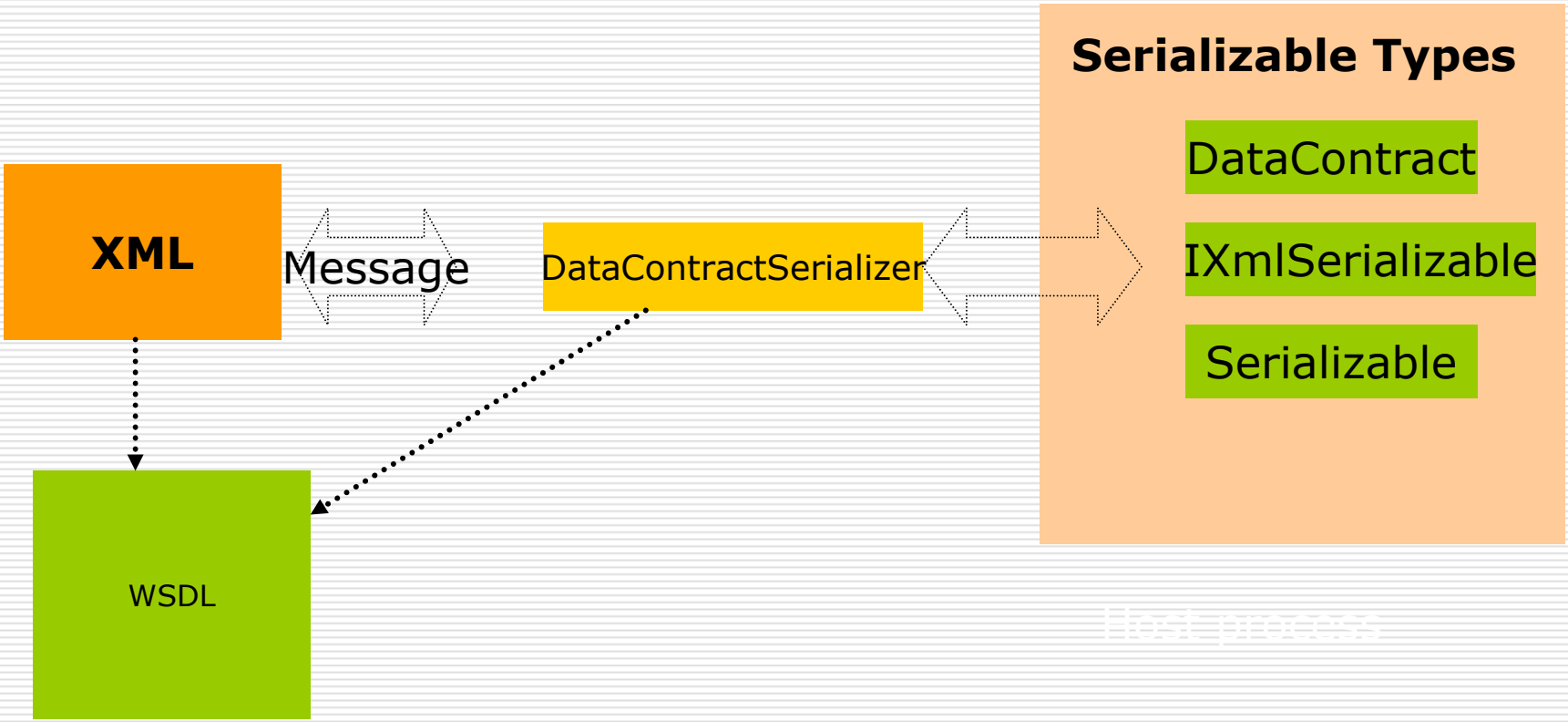
WCF: Complex Types and Serialization

- When we host a service,the service contract exposes the number of operations available within each service contract.
- The messaging parameters and return types are serialized.
- WCF provides the following types of complex serialization:
  - Serializable types :Used with .Net Remoting.
  - Data Contracts :WCF's preferred way of exposing complex type.
  - Known Types :when we have to serialize a polymorphic type based on several interpretations that are derived in the application.
  - IXmlSerializable : based on schema granted to us, we have datatypes over which we have no control of serialization but would like to implement our side of serialization.

WCF: How do we serialize?

- Create a ServiceContract
- Define the operations.
- The operations will have incoming parameters and return types.
- These are regenerated into WSDL and describe what our messages should look like.
- Messages coming in carry one or more types described in our wsdl and map it to the CLR type by looking at the DataContract which is also defined in the WSDL.
- The DataContractSerializer which is the default WCF serializer will look at the incoming XML and map it with the CLR type by looking at ServiceContract.
- DataContractSerialzer support 3 types of serializations
- a. DataContracts b.Serializable Type and c.IXmlSerializable

WCF: DataContractSerializer



WCF: Serializable Attribute

- It is applied using the [Serializable] attribute to a CRL type.

```
[Serializable]
public class customer
{
    private long custid;
    private string name;
    private string address;
    private string postalcode;
    private string telephone;
    private string mobile;
    private DateTime birthdate;
}
```

- This way all fields irrespective of accessibility are serialized,we have no control.
- The order of serialization is also not under the developers control and hence not suitable for cross platform integrations.

WCF: Serializable Attribute

- It is applied using the [Serializable] attribute to a CRL type.

```
[Serializable]
public class customer
{
    private long custid;
    private string name;
    private string address;
    private string postalcode;
    private string telephone;
    private string mobile;
    private DateTime birthdate;
}
```

- This way all fields irrespective of accessibility are serialized,we have no control.
- The order of serialization is also not under the developers control and hence not suitable for cross platform integrations.
- No option to opt in/out the items for serialization.

WCF: DataContracts

- DataContract attribute
  - It provides translation between CLR type and schema
- DataMember attribute
  - Used to opt in/opt out members.
  - Can be used on fields and properties.
  - Usually added to properties.
- Preferred way of DataContractSerializer to work with complex types.

WCF: DataContracts

- For our example the ICustomerService and IOrderService Contracts are defined using complex type

```
[ServiceContract()]
   public interface ICustomerService
   {
      [OperationContract]
      CustomerItem GetCustomer(CustomerItem oCustomer);
   }

 [ServiceContract()]
    public interface IOrderService
    {
        [OperationContract]
        OrderItem GetOrder(OrderItem oOrder);
    }
```

The CustomerItem/OrderItem complex types to be serialized are defined under the **project DataContract** in the **source code** attached.

WCF: DataContracts

- Compile the Host and keep it running you will find that now the service defination WSDL can be browsed at

  http://localhost:8002/?wsdl
- However the CustomerItem/OrderItem complex types defination are not part of the WSDL file.Incase you would like to have a look at it then serach for
  - http://localhost:8002/?xsd=xsd0
  - http://localhost:8002/?xsd=xsd1
  - http://localhost:8002/?xsd=xsd2
  - The complex type defination is at http://localhost:8002/?xsd=xsd2
- Have a look at the WSDL definations for this session and get confortable with the idea of Datacontracts and DataMembers.Do a bit of reading on the internet so that when I follow it up with a full server/client application then you find catching up easier.
- Please refer to the code as too much of technical jargon will make things complex unnecessary.