**Introduction on Data Contract**

**Rameshkartik.RS**

**Table of Contents**

[Introduction 2](#_Toc386117845)

[Technologies behind SOA 3](#_Toc386117846)

[UDDI 4](#_Toc386117847)

[Summary 6](#_Toc386117848)

**WHAT’s IN THIS ARTicle?**

* Introduction on DataContracts
* DataContract Serializer
* Source Code Explanation
* Attachment
* Summary

# 

# Introduction

Contract = Agreement,

Data Contract = Data Agreement,

Data Agreement between whom? = The agreement between client and the server.

What exactly the agreement contains? = The structured data to be exchanged between the client and server.

Hope you guys understood the above lines, Yes Data Contract is an agreement between the client and server about the structure of data being exchanged. Data Contract is to describe the external format ,structure and type of data being exchanged. It also defines how the data types are serialized and deserialized, Serialization happens on the client side to convert the objects into the messages that can be stream over the network.On the other hand the dispatcher in the server side performs deserialization to convert the objects into data types and pass it to the service methods for execution. To know the various stages on data transfer from client to the server and how the parameters are converted into objects and the objects converted into messages, kindly refer my previous article “WCF Run Time Architecture”.

# DataContractSerializer

WCF uses the DataContractSerializer class from the namespace System.RunTime.Serialization to serialize and deserialize the objects. This serializer is specially developed for WCF to perform fast,effective and powerful.

DataContractSerializer supports the following data types

* Primitive data types
* Data types with the [DataContract] attribute
* Class which marked as Serializable
* Class which implement the IXmlSerializable
* Enumerations ,collections and generic collections

Primitive data types are already serializable , so no serialization measures required for the primitive data types. Apart from the primitive types whatever the complex types are used as parameters,return values in the service, we need to treat the class with [DataContract] Attribute and the properties,member variables with [DataMember] attribute. These attributes are actually used by the DataContractSerializer class to control the serialization/deserialization. In other words ,these two attributes tells the Datacontractserializer how to serialize the complex types. [DataContract] is a opt in model which means only the member variables or properties marked as [DataMember] will be taken for the Serialization/Deserialization. The definition of data is incorporated into the WSDL document in a XSD format.

# Source Code Explanation

Interface in Service

[ServiceContract(Namespace = "http://Rameshkartik/WCFSamples/OnlineShoppingService",

Name = "OnlineShoppingService")]

public interface IOnlineShoppingService

{

[OperationContract(Name = "StockAvailability")]

bool IsStockAvailable(string sModelNumber);

[OperationContract(Name = "CalculatePrice",

ProtectionLevel = System.Net.Security.ProtectionLevel.None)]

double CalculatePrice(string sModelNumber, int iQuantity, string sDeliveryLocation);

[OperationContract(Name = "GetWholeSaleDealerInfo")]

WholeSaleDealersInfo GetwholeSaleDealerInfo(string sFilterByCity);

}

# Look the above code snippet, In the service interface we have exposed three methods *called IsStockAvailable,CalculatePrice,GetWholeDealerInfo*. If you see the return type of *the GetWholeSaleDealerInfo* method *is WholeSaleDealersInfo* type, which is a complex type. We need to tell the DataContractSerializer about the complex type *WholeSaleDealersInfo* to serialize/deserialize*.*

DataContract WholeSaleDealersInfo

[DataContract]

public class WholeSaleDealersInfo

{

private string sDealerName;

private string sDealerLocation;

private DateTime sDealerSince;

private int iDealerAge;

[DataMember]

public string DealerName

{

get { return sDealerName; }

set { sDealerName = value; }

}

[DataMember(Name="LocationofDealer",Order=3,IsRequired = true, EmitDefaultValue=true)]

public string DealerLocation

{

get { return sDealerLocation; }

set { sDealerLocation = value; }

}

}

Look at the above code snippet, the class has been decorated with the attribute[DataContract] and the properties as [DataMember]. This decoration is to tell the DataContractSerializer how and in which form its objects are to be converted.

Parameter Order

There is an attribute called Order in the Datamember where you can define the order of serialization and deserialization . Sometime datacontract should be serialized and deserialized in specific order of data members.

Be default the Data members are serialized and deserialized in the below order

1. If the DataContract has derived from the base DataContract so the data members from the base class will process first in alphabetical order
2. DataMembers with no order property set will process in alphabetical order
3. Remaining DataMembers will be processed in ascending order of Order property value

Parameter Name

Using the DataMember Attribute,we can define the name of a DataMember on each property . It will be helpful to solve the DataContract versioning issues. The below source code will give you a better understanding of this attribute.

// Version 1 of DataContract Order

[DataContract]

public class Order

{

[DataMember]

private string OrderID;

[DataMember]

private string Name;

}

// Version 2 of DataContract Order

[DataContract]

public class Order

{

[DataMember]

private string OrderID;

[DataMember(Name = "Name")]

private string EmployeeName;

}

Parameter IsRequired,EmitDefaultValue

By Default the IsRequired field will be false, If it is true that particular property must be available during the serialization or deserialization process. If the IsRequired field is true and the EmitDefaultValue is true and if the value is not provided the default value will be taken during the serialization. If the IsRequired field is true and the EmitDefaultValue is false and if the value is not provided the default value will not be taken during the serialization and the serialization exception will be thrown.

# Attachment

Refer the code attachment for further details

# Summary

Whereever the [DataContract] and [DataMember] attributes declared in the class/properties ,these types will be taken by DataContractSerialization class for serialization and deserialization.