Basically a WCF Contract is an agreement between the two parties, in other words a Service and a Client.

WCF has five types of contracts: service contract, operation contract, data contract, message contract and fault contract.

In Windows Communication Foundation, contracts can be categorized as behavioral or structural.

## Behavioral Contracts

* The ServiceContract attribute marks a type as a Service Contract that contains operations.
* OperationContract attribute marks the operations that will be exposed.
* FaultContract defines what errors are raised by the service being exposed.

## Structural Contracts

* The DataContract attribute defines the types that will be moved between the parties.
* The MessageContract attribute defines the structure of the SOAP message.

## S ervice Contract and Operation Contract

Service Contract

A **Service Contract** basically describes the **operations** a **service** exposes to another party (in other words a client). We can map a **WCF Service Contract** to a Web **Service** Description Language (WSDL). It's recommended to apply the **ServiceContract** attribute to an interface, although it can be applied to a class as well



It describes:

* What operations are exposed by the service
* Platform independent description of the interface as well as methods of our service
* A Message Exchange Pattern (MEP) between the parties, in other words Request/Response, One-Way or Duplex. Please [follow here](http://www.topwcftutorials.net/2012/06/wcf-top-10-interview-questions.html) for a detailed description of MEPs.

Data Contract

A data contract defines the data type of the information that will be exchange be-tween the client and the service. A data contract can be used by an operation contract as a parameter or return type, or it can be used by a message contract to define elements.

What is a Fault Contract?

**A Fault Contract is a way to handle an error/exception in WCF. In C# we can handle the error using try and catch blocks at the client-side. The purpose of a Fault Contract is to handle an error by the service class and display in the client-side. ... FaultException<T>: to send a typed fault data to the client.**

In WCF (Windows Communication Foundation), we will not expose exception directly to client if it occurs at service level. There is a valid reason for that, “*WCF exceptions are basically CLR exceptions containing internal details of service code (e.g. stack trace etc), so it doesn’t make sense to expose it outside CLR*“.  
In Windows Communication Foundation, Fault Contract is used to return error details to the other party i.e. client. Fault Contract is a behavioral contract that contains the details of possible exception(s) that might occur in a service code.

In order to fully understand the idea, let’s create a WCF Service that generate exception.

*[ServiceContract]*  
*public interface ISimpleService*  
*{*  
*[OperationContract]*  
*string SimpleOperation();*  
*}*

*public class SimpleService : ISimpleService*  
*{*  
*public string SimpleOperation()*  
*{*  
*//Some code here…*  
*throw new Exception(“Exception occurred at service level : SimpleOperation error”);*  
*}  
}*

While consuming this service on client side, we are expecting above given exception message “Exception occurred at service level : SimpleOperation error” when SimpleOperation is called. But we will get the following error in all cases.

*“The server was unable to process the request due to an internal error. For more information about the error, either turn on IncludeExceptionDetailInFaults(either from ServiceBehaviorAttribute or from the configuration behavior) on the server in order to send the exception information back to client, or turn on tracing as per the Microsoft .NET Framework 3.0 SDK documentation and inspect the server trace logs”*

so, in order to avoid the above generic error and get a customized error message, we will modify our service code as follows:

*public string SimpleOperation()*  
*{*  
*//Some code here…*  
*throw new FaultException(“Exception occurred at service level : SimpleOperation error”);*  
*}*

Now, if we run the client application and call the WCF Service, we will get a proper meaningful error as:

“*Exception occurred at service level : SimpleOperation error*“

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace WCFServiceLibraray

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Maths" in both code and config file together.

public class Maths : IMaths

{

public int add(int x, int y)

{

return x + y;

}

public int subt(int x, int y)

{

return x - y;

}

public int product(int x, int y)

{

try

{

return x \* y;

}

catch(Exception e)

{

throw e;

}

}

public int divide(int x, int y)

{

try

{

return x / y;

}

catch (Exception e)

{

throw e;

}

}

}

}

Now use FaultContract

Create a class

[DataContract]

public class CustomFaultDetails

{

[DataMember]

public string ErrorID { get; set; }

[DataMember]

public string ErrorDetails { get; set; }

}

[OperationContract]

**[FaultContract(typeof(CustomFaultDetails))]**

int divide(int x, int y);

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace WCFServiceLibraray

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Maths" in both code and config file together.

public class Maths : IMaths

{

public int add(int x, int y)

{

return x + y;

}

public int subt(int x, int y)

{

return x - y;

}

public int product(int x, int y)

{

try

{

return x \* y;

}

catch(Exception e)

{

throw new FaultException("Exception Occured");

}

}

public int divide(int x, int y)

{

try

{

return x / y;

}

//catch (Exception e)

//{

// throw new FaultException("Exception Occured");

//}

catch

{

CustomFaultDetails ex = new CustomFaultDetails();

ex.ErrorID = "12345";

ex.ErrorDetails = "Specific error details here.";

throw new FaultException<CustomFaultDetails>(ex);

}

}

}

}

A **message contract** is used to control the structure of a **message** body and serialization process. It is used to send/access the information in the soap header. By use of a **Message Contract** we can customize the parameters sent using a SOAP **message** between the client and the server.

In WCF a data contract enables us to define the structure of the data. This data is sent in the body of an envelope.

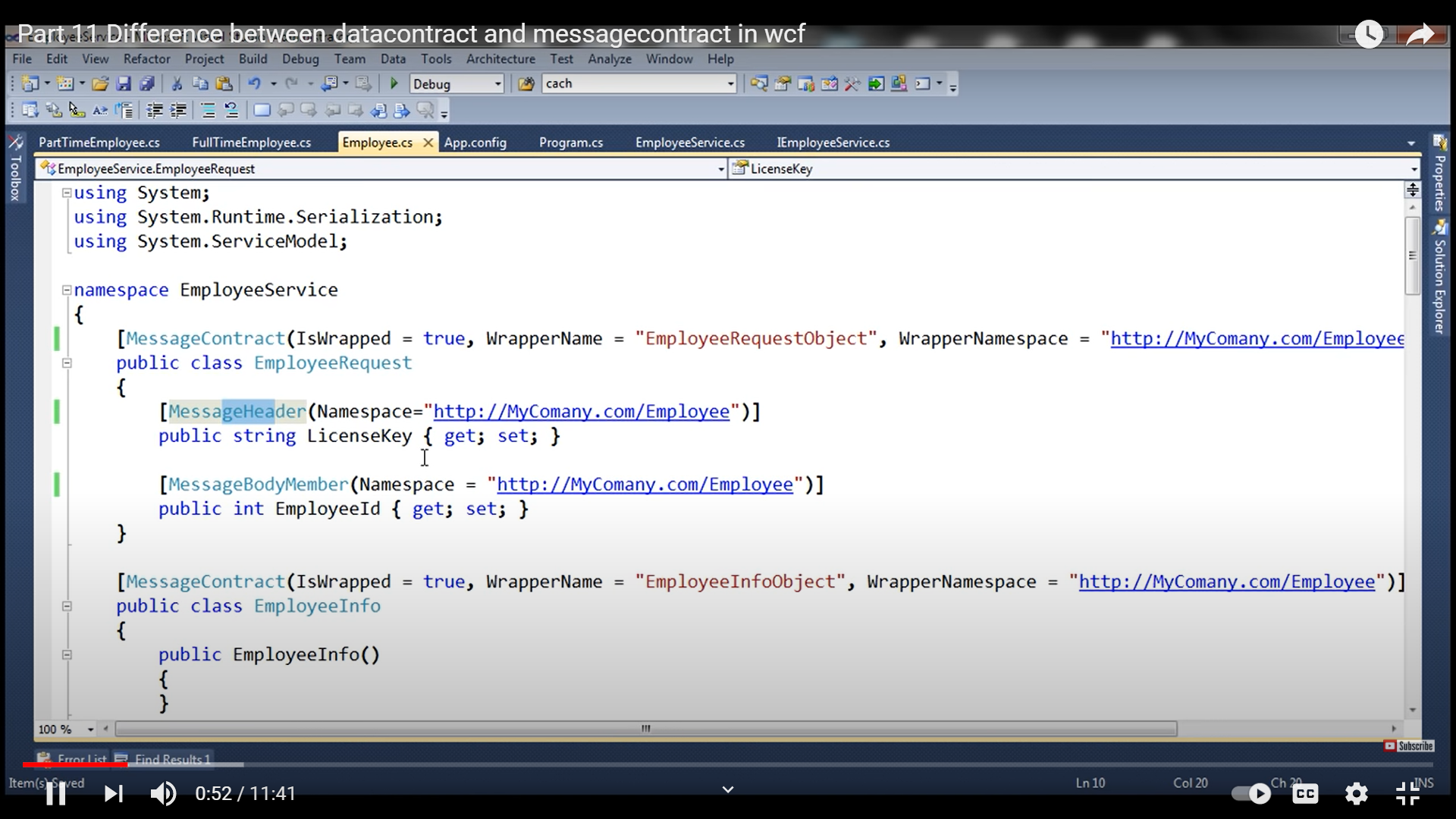


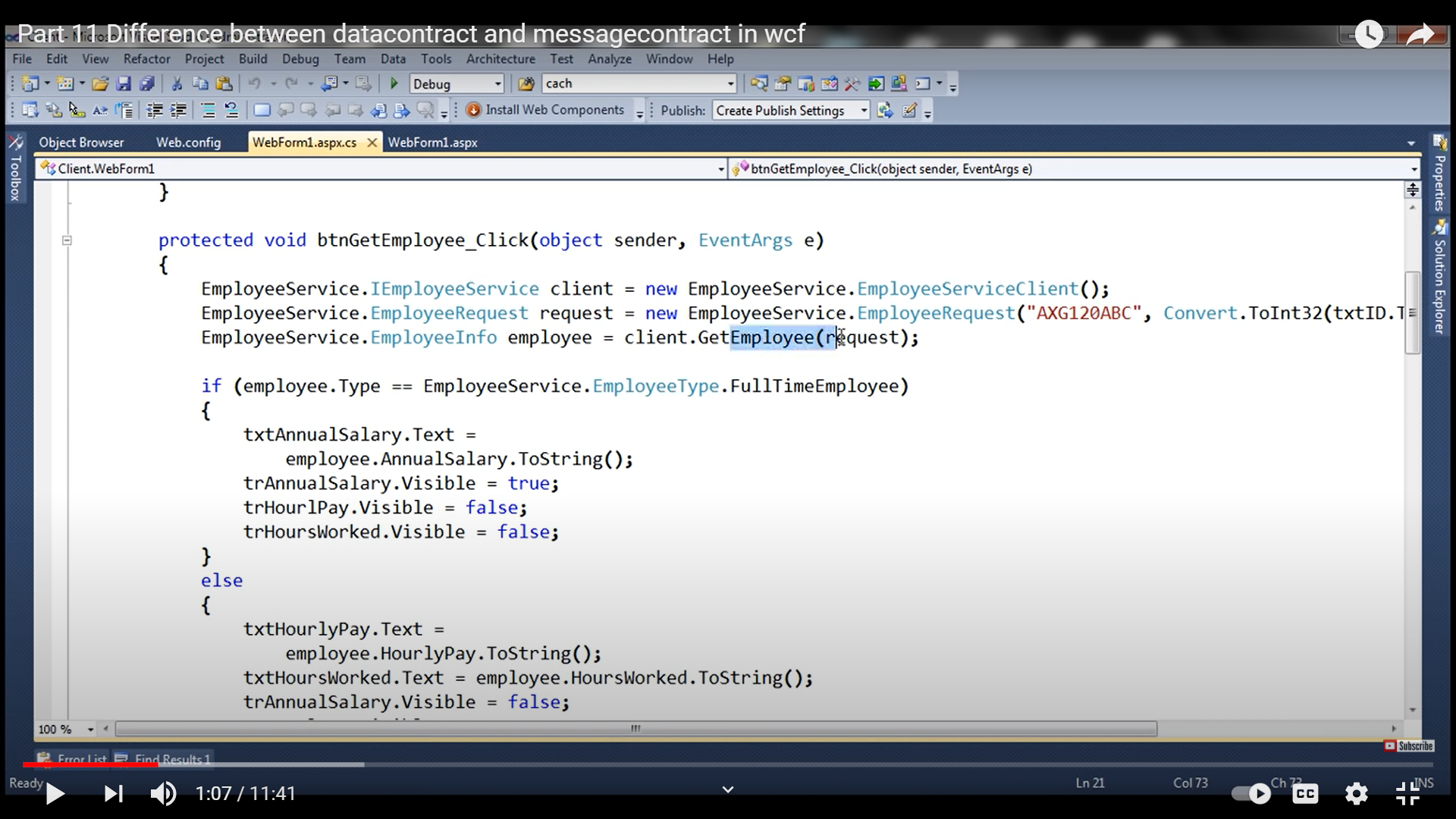
A message is nothing but a packet and WCF uses this packet to transfer the information from source to destination. This message is contained in the header or body.

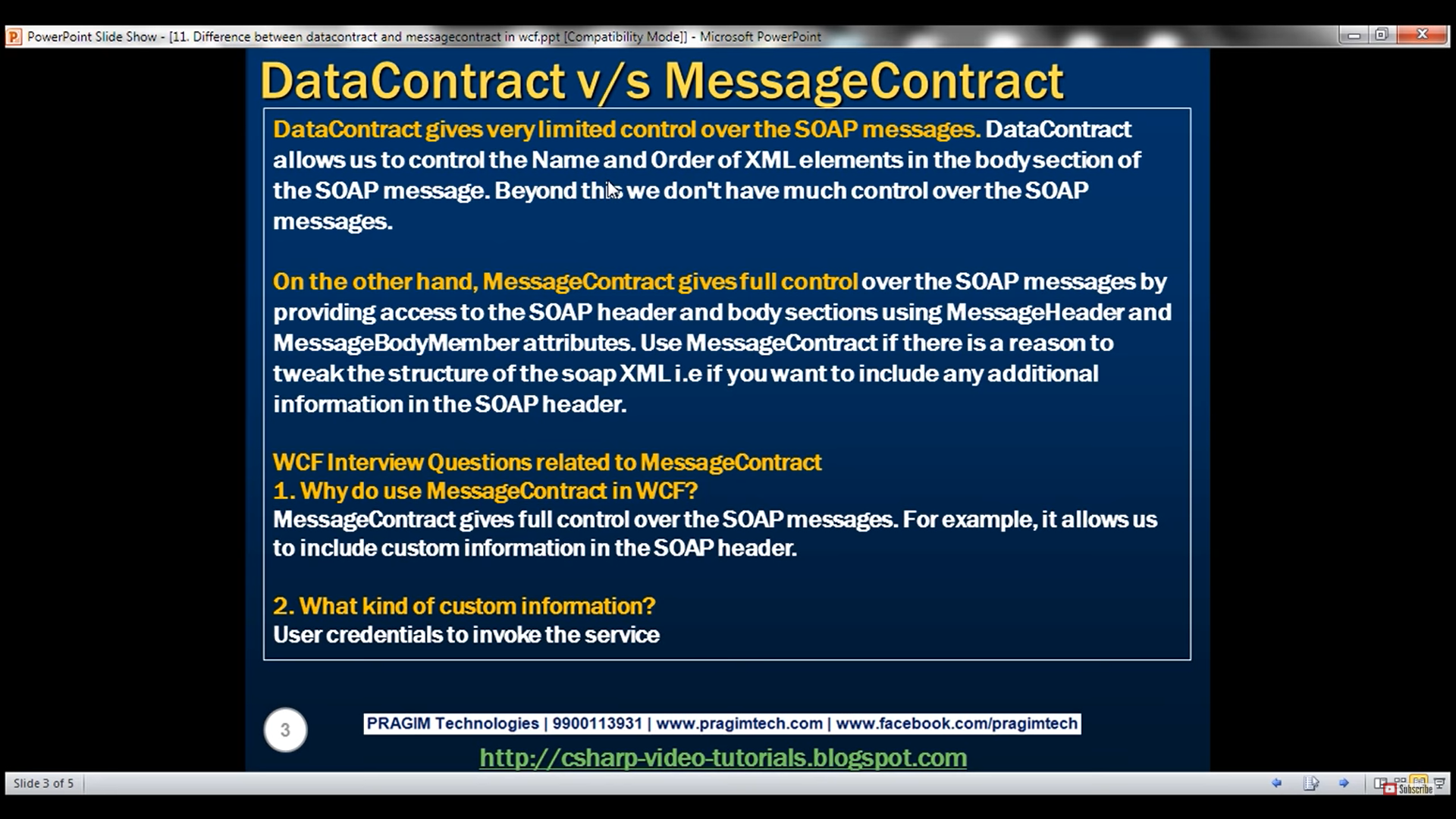
A message contract is used to control the structure of a message body and serialization process. It is used to send/access the information in the soap header. By use of a Message Contract we can customize the parameters sent using a SOAP message between the client and the server. The SOAP header is implemented in the namespace system.web.services.protocol.

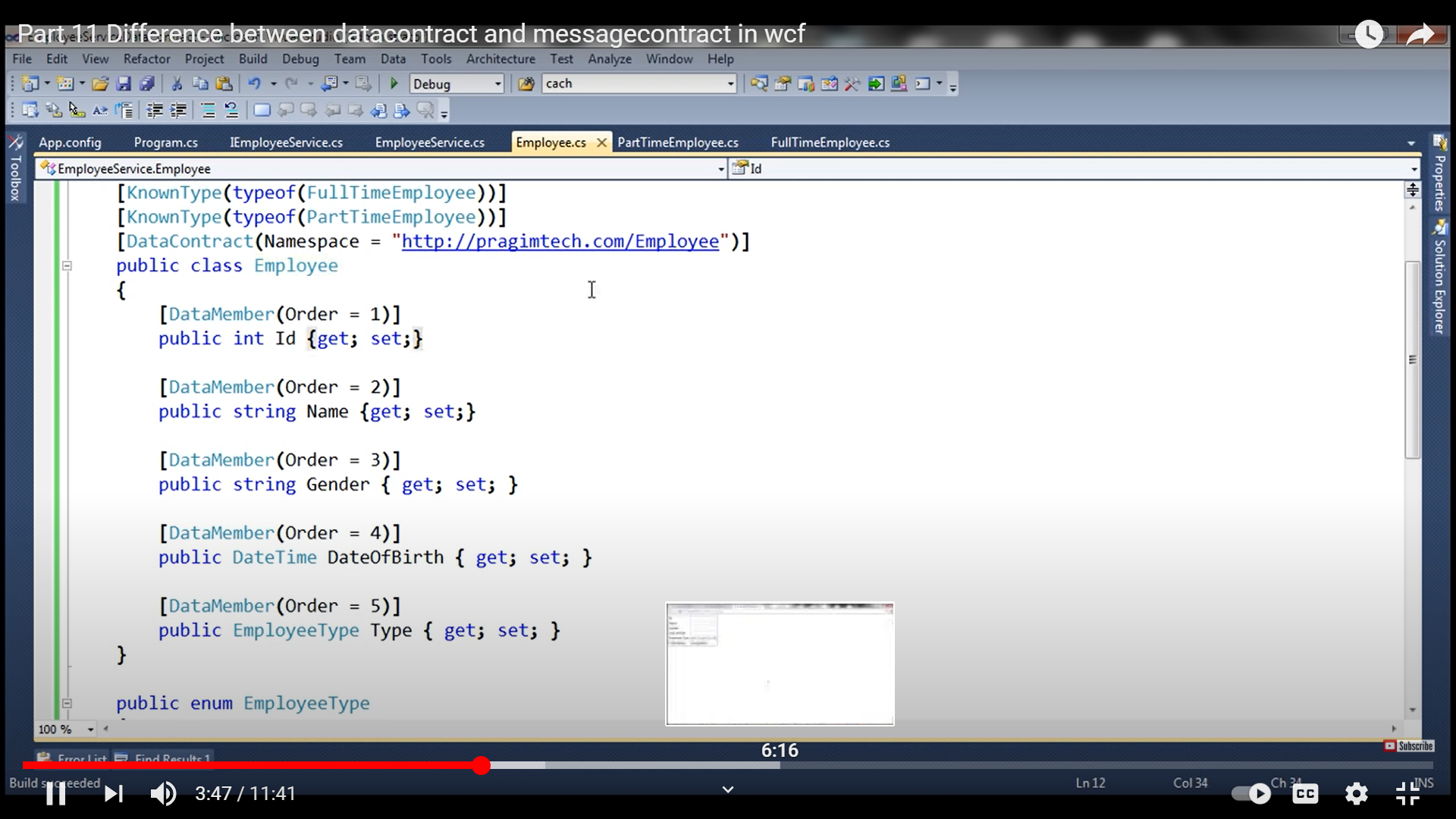
When an operation contract required to pass a message as a parameter or return value as a message, the type of this message will be defined as message contract. A message contract defines the elements of the message (like as Message Header, Message Body), as well as the message-related settings, such as the level of message security.

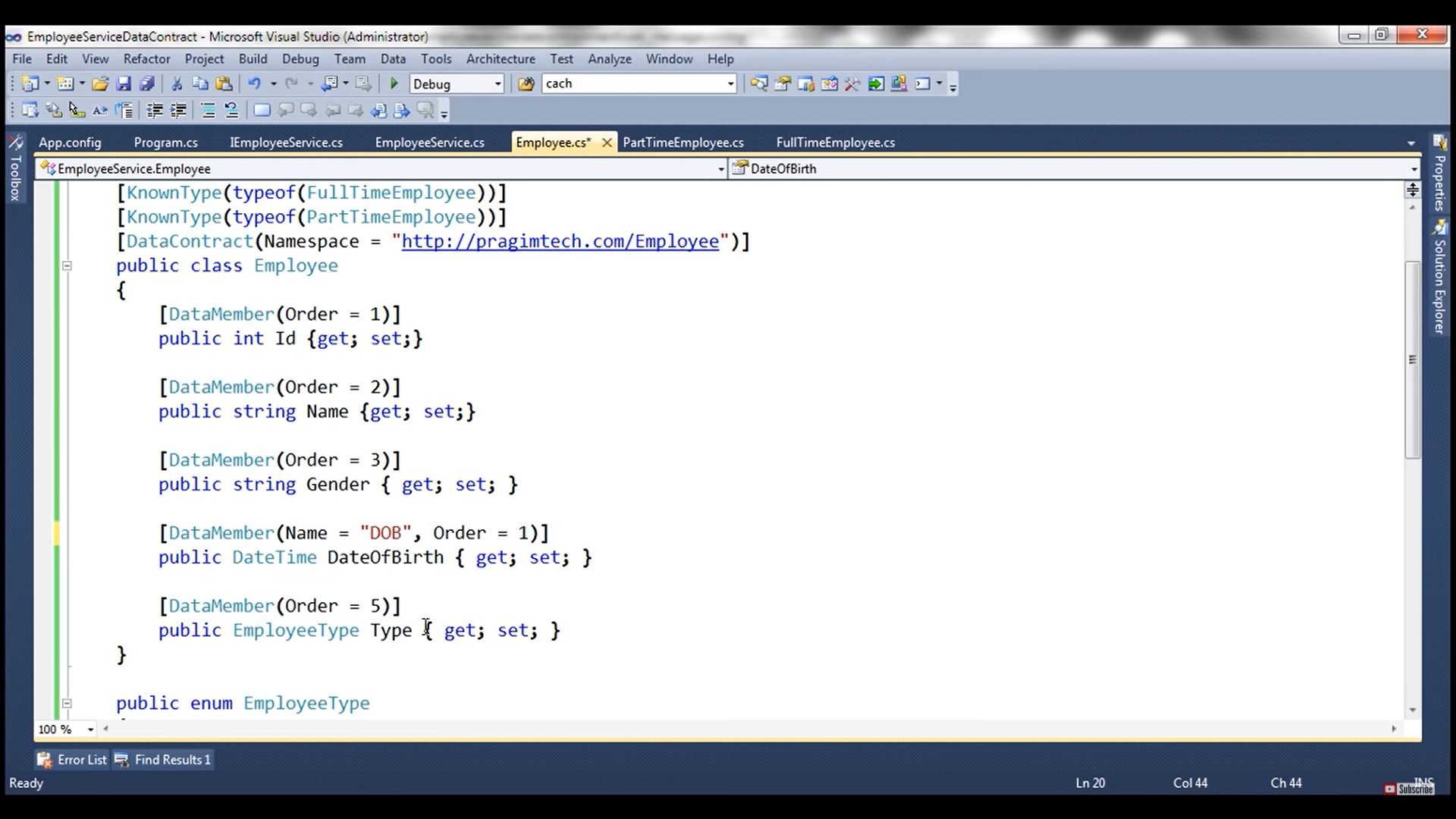
Message contracts give you complete control over the content of the SOAP header, as well as the structure of the SOAP body.

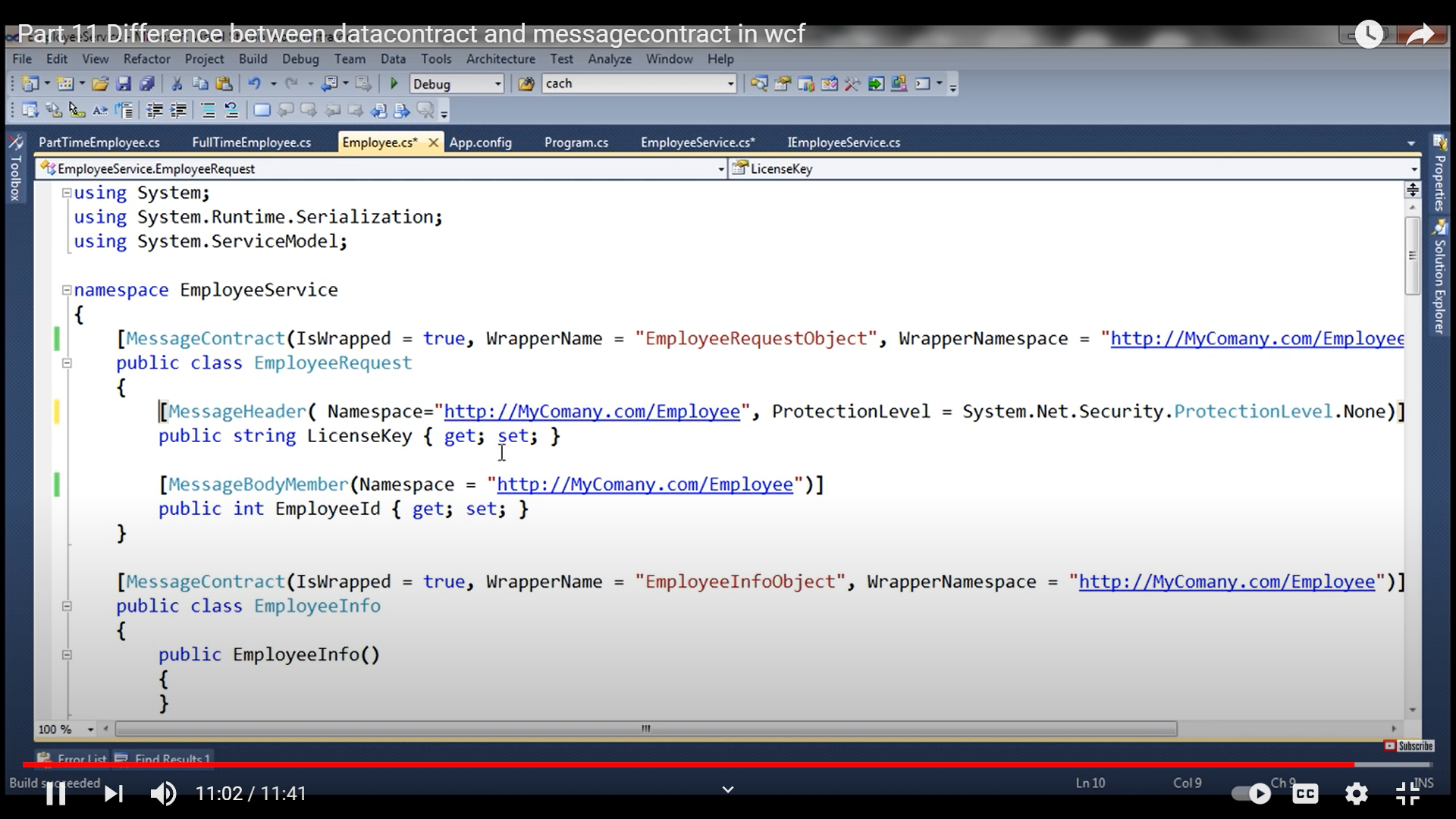






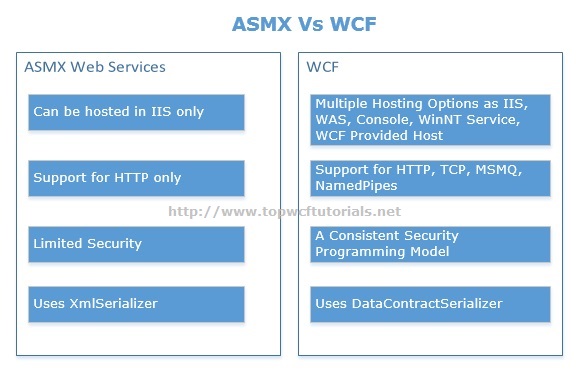




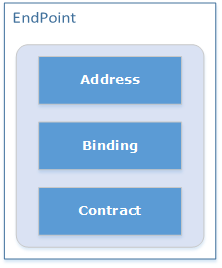


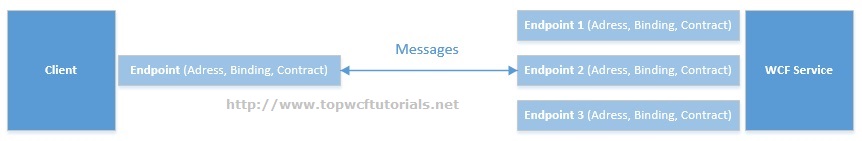
### What is the difference between WCF and ASMX Web Services?

Simple and basic difference is that ASMX or ASP.NET web service is designed to send and receive messages using SOAP over HTTP only. While WCF can exchange messages using any format (SOAP is default) over any transport protocol (HTTP, TCP/IP, MSMQ, NamedPipes etc).



### What are WCF Service Endpoints?

For **Windows Communication Foundation**services to be consumed, it’s necessary that it must be exposed; Clients need information about service to communicate with it. This is where service endpoints play their role. Client uses endpoint to communicate with WCF Service. A **WCF service** endpoint has three basic elements i.e. Address, Binding and Contract.[](http://www.topwcftutorials.net/wp-content/uploads/2012/08/WCFEndPoint.png)

* **Address:** It defines “WHERE”. Address is the URL that identifies the location of the service.
* **Binding:** It defines “HOW”. Binding defines how the service can be accessed.
* **Contract:** It defines “WHAT”. Contract identifies what is exposed by the service.
* 

A WCF Service can have multiple endpoints configured to accommodate different clients, for example, one client may be using HTTP while other configured to communicate over TCP.

**What are the possible ways of hosting a WCF service? Explain.**

For a **Windows Communication Foundation**service to host, we need at least ***a managed process***, ***a ServiceHost instance***and ***an Endpoint configured***. Possible approaches for hosting a service are:

* Hosting in a Managed Application/ Self Hosting
  + Console Application
  + Windows Application
  + Windows Service
* Hosting on Web Server
  + IIS 6.0 (ASP.NET Application supports only HTTP)
  + Windows Process Activation Service (WAS) i.e. IIS 7.0 supports HTTP, TCP,  
    NamedPipes, MSMQ.