**WCF**

WCF es un marco de trabajo para la creación de aplicaciones orientadas a servicios. Con WCF es posible enviar datos como mensajes asíncronos de un extremo a otro. Los mensajes pueden ser simples o complejos. La base de WCF es lo que se conoce como ABC:

A (Address) que especifica la ubicación del servicio.

B (Binding) que indica de que forma debe establecerse la conexión (http, https, etc…).

C (Contract) que determina la funcionalidad del servicio.

**Contrato**

Un contrato en WCF define las operaciones soportadas por el servicio, los parámetros y los tipos de datos que devuelven dichas operaciones. Los contratos proporcionan la metadata necesaria para comunicarse con el servicio, como los tipos de datos, operaciones, patrones de intercambio de mensajes y el protocolo de transporte que se utiliza. En WCF existen tres tipos de contrato:

1. **Service Contract** .- Define las operaciones que el servicio tendrá disponible para que los clientes puedan invocar mediante peticiones *request*, se identifican sus operaciones con el atributo *[OperationContract]*.
2. **Data Contract** .- Define la estructura de los datos que serán incluidos en le intercambio de los mensajes que van y vienen del servicio al cliente y viceversa. Se coloca el atributo *DataMember* a cada uno de los campos que necesiten ser serializados (Tipo de dato complejo).
3. **Message Contract** .- Habilita el control de los headers que utilizan los mensajes y como son utilizados por el servicio.

**Configurar un servicio**

**Endpoint** .- Un servicio WCF es un programa que expone una colección de Endpoints (puntos de entrada). Un cliente es un programa que intercambia mensajes con uno o más endpoints. Cada endpoint está compuesto por una dirección (A - Address) que indica donde está desplegado el servicio, es una dirección de red, enlace (B - Binding) que indica como establecer el contacto con el servicio, protocolo, codificación y seguridad y un contrato (C – Contract) que indica que funcionalidades se exponen en el servicio.

La configuración del servicio se hace en el webconfig dentro de la etiqueta <system.ServiceModel>. En nuestro ejemplo añadiremos las siguientes etiquetas:

<bindings>

<basicHttpBinding>

<binding name="EnlaceHttp" closeTimeout="00:10:00" maxReceivedMessageSize="65536" maxBufferSize="65536" messageEncoding="Text" textEncoding="utf-8"></binding>

</basicHttpBinding>

</bindings>

<services>

<service name="WcfService.WcfClientService">

<endpoint address="" binding="basicHttpBinding"

bindingConfiguration="EnlaceHttp"

contract="Credit.Contract.IClientService"></endpoint>

</service>

</services>

Dejamos sin completar el address para que tome la dirección de nuestro directorio de publicación del servicio.

**Administrar excepciones**

Si lanzamos una excepción normal de un servicio de WCF, el cliente que consume ese servicio recibirá un error genérico que no proporciona ningún detalle sobre el problema. En su lugar, debemos utilizar *FaultException*, que dará una excepción formateada al cliente.

**Excepciones personalizadas**

Personaliza la información de la excepción de cara al cliente:

**FaultContract**: Counicar información de error del servicio al cliente.

**FaultException<T>**: T puede ser cualquier tipo que se pueda serializar, se debe enviar una nueva excepción de tipo *FaultException<T>* en la captura de la excepción.

Para poder capturas las excepciones debemos modificar el webconfig, dentro de la etiqueta <behavior> poner a true el valor de includeExceptionDetailInFaults.

<serviceDebug includeExceptionDetailInFaults="true"/>

De esta forma ya obtenemos más detalles del porqué se produce una excepción. Pero la mejor opción es crear excepciones personalizadas en una clase, para nuestro ejemplo será la clase *Error*, implementada dentro del Dominio y añadir el atributo [FaultContract(typeof(Error))] en la firma de los métodos que queremos que capturen las excepciones, en nuestro caso en la interfaz IClientService.

**Diferencias en los contratos SOAP y REST**

Para REST debemos adicionar (además del anterior *[OperatonContract]*) alguno de los siguientes atributos:

1. *WebGet* .- operaciones de consulta

*[WebGet(ResponseFormat=WebMessageFormat.Json,*

*UriTemplate=”Produc/{productID}”)]*

1. *WebInvoke* .- transacciones de tipo CUD (POST, PUT, DELETE)

*[WebInvoke(Method=”PUT”,*

*ResponseFormat=WebMessageFormat.Json, UriTemplate=”producto/[productId}”)]*

Para configurar el servicio REST y poder intercambiar mensajes en formato JSON debemos añadir un nuevo Endpoint. Para llevarlo a cabo configuraremos la dirección o Address el Enlace o Binding (webHttpBinding), el bidningConfiguration (crossDomainScriptAccessEnabled), además de añadir el atributo webHttp (behaviorConfiguration). El contrato no varía respecto a SOAP.

Todo esto se configura en el webconfig añadiendo las siguientes etiquetas para nuestro proyecto concreto:

Dentro de behaviors

<endpointBehaviors>

<behavior name="webRest">

<webHttp helpEnabled="true"/>

</behavior>

</endpointBehaviors>

Dentro de bindings

<webHttpBinding>

<binding name="EnlaceHTTPRest" closeTimeout="00:10:00"

crossDomainScriptAccessEnabled="true">

</binding>

</webHttpBinding>

Y finalmente el endpoint dentro de services

<endpoint address="rest" binding="webHttpBinding"

contract="Application.Credit.Contract.IClientService"

bindingConfiguration="EnlaceHTTPRest"

behaviorConfiguration="webRest">

</endpoint>

Pendiente de añadir como configurar y desplegar WCF en Windows Service.