module 27

wcf basics

INTRODUCTION

This module defines what WCF is and why you might use it in your own solutions. It shows you how to define, host and consume a basic WCF service.

OUTCOMES

By the end of this module, you should be able to do the following:

* Construct a simple end to end system of a WCF client consuming a WCF service.
* Define an interface that can be exposed by a WCF host

# module 27: wcf basics

## 27.1 – Service Oriented Architecture (SOA) Comparison

In the OOP world code would be written to be reusable. Source code could be written for one project, and then be copied and pasted into another application when it requires the same functionality. The problems with this is there is now code duplication going on, and changing code (bug fixes, new business logic) in one place will not update the code in other areas (DLLs can be used, but these still have similar problems with deployment and versioning). Further problems arise when trying to share functionality with other departments/businesses.

SOA unifies business processes by structuring large applications as an ad-hoc collection of smaller components called ‘services’. These services can be used by people both inside and outside the company, and new applications can be built by combining these services into new ways.

Instead of having every application having to handle the creation or retrieval of customer records, this can be created as a standard service which all applications can consume.

In summary:

* **OOP**
  + “Code Reuse”
  + Deployed code would have to be recompiled and redeployed
* **SOA**
  + Progression from ‘code reuse’ to ‘Service Reuse’
  + Standard functionality can be written once, applications can consume this service
  + Logic only needs to be updated in a single place

Building all applications from the same common set of services reduces development time and when exposed to external consumers, can make the goal of transactions bridging companies easier.

A good example of a service on the web is Amazon.com. They have services exposed which allow you to query and retrieve book details in a standardised fashion.

### 27.1.1 – The 4 Tenets of SOA

1. Explicitly expose functionality
   * Location is irrelevant
   * Technology is irrelevant
2. Services are autonomous
   * they are Operated independently
3. Services share Schema and Contract, not Class
   * Only deal with what is allowed to pass in and come out.
4. Compatibility is based upon Policy
   * Every service advertises its capabilities and requirements in the form of a machine-readable policy expression

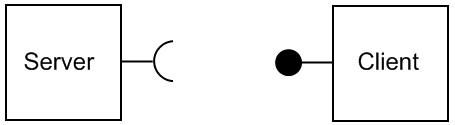
## 27.2 – Windows Communication Foundation

WCF is an SDK for building SOA on Windows and was introduced in .NET 3.0. WCF combines many previous Microsoft messaging technologies together into a standard coding framework. A service is written as a logical piece of work and is then exposed using whichever method is most appropriate. This keeps the service logic and the technical transportation pieces of the service separate and reduces the complexity.

* Services send and receive messages
* All messages are SOAP messages
* WCF services may interoperate with non-WCF services
* Brings together all of the disparate Microsoft messaging technologies
  + ASMX web services
  + .NET remoting
  + Microsoft Message Queuing

### 27.2.1 – Overview

* Server exposes a service
* Client creates a ‘proxy class’ of the service using metadata
* Client connects and consumes the service by making method calls as if the class was local
* .NET handles the plumbing to get from A to B



## 27.3 – WCF Endpoints - the ABC of WCF

An easy way to remember how a WCF service is defined is to remember its ABCs:

* **A**ddress - where the service is
* **B**inding - how to talk to the service
* **C**ontract - what the service can do

Consider the following real world analogy. The address of the service is where it is located; this is unique to it just like the address of a house. Just as there can be many houses with unique addresses the same goes with WCF, each address can only be used for one service. Addresses must be unique.

Now that you know where you are going, the next question is how are you going to get there? You could go by truck or train or walking, each has its own advantage/disadvantage. The same question needs to be asked of the messages sent with WCF, how will they travel to where they are going? This could be via HTTP or TCP, or could be guarded with extra security as it travels.

Contracts are the menu of available functions the service provides.

## 27.4 – Exposing endpoints

* Every service must expose at least one endpoint
* Each endpoint has exactly one contract
* All endpoints should have unique addresses
* Single service can expose multiple endpoints
  + Can use same or different bindings
  + Can expose same or different contracts
* Are set up in a config file separated from the code

### 27.4.1 – Address

The address is a combination of the **server name**, **port number** and **path**. The transport at the beginning will be determined by the binding e.g. HTTP.

### 27.4.2 – Bindings

There are numerous ways that a message can be formatted/sent/secured, this allows you to tailor your service for the compatibility/performance you require for your solution.

* Transport
  + HTTP
  + TCP
  + MSMQ
* Message formats and encoding
  + Plain text
  + Binary
  + MTOM
* Communication security
  + No security
  + Transport security
  + Message security
  + Authenticating and authorizing callers

Because there is such a large number of choices that can be made these have been packaged into ‘standard binding’ sets for easy use.

A service can support multiple sets of bindings, however each must be on a separate address. The Client must use exactly the same binding as service otherwise they will not be able to communicate properly.

### 27.4.3 – Standard Bindings

These standard binding sets mostly relate back to each of the previous Microsoft message technologies that were combined under the WCF banner.

* **BasicHTTP** – for integration with legacy WebService clients
* **TCP** – messages are sent over TCP
* **Peer TCP** – Peer to Peer (P2P) networks
* **Named Pipe** – memory streams for inter-process communication
* **WS** – Web services with support for security, reliability, etc.
* **MSMQ** – MicroSoft Message Queuing

The standard bindings can be customised and new custom bindings can be created.

### 27.4.4 – Contracts

Each service has three main sets of contracts

* **Service contracts** 
  + List of operations the service is exposing
* **Data contracts**
  + Allows you to create ‘data objects’ to be sent between client/server. Specifies the schema.
* **Fault contracts** 
  + Abstraction away from “Exceptions”
  + Which errors can raised by the service. To allow handling of them

To expose a class and methods as a service you need to decorate them with metadata, this allows .NET to map the CLR to WCF. A single Class can implement multiple **ServiceContracts**; it just needs to implement multiple Interfaces. The methods exposed with [OperationContract] can only use primitive or data contracts as parameters, i.e.:

* + **[ServiceContract]** – Defines a ‘set’ of operations
  + **[OperationContract]** – Defines a single method

[ServiceContract]

public interface IService1

{

[OperationContract]

string GetData(int value);

}

public class Service1 : IService1

{

public string GetData(int value)

{

return string.Format("You entered: {0}", value);

}

public string OtherMethod()

{

return "This method can't be called by WCF";

}

}

## 27.5 – MetaData

A service can expose multiple endpoints with different bindings and addresses. It can also expose a MetaData endpoint which allows clients to discover how to interact with the service. The bindings used, operation contracts and data contracts are all defined allowing the client to consume the service without having to contact the developers. MetaData endpoints allow WCF to be self describing.

## 27.6 – Hosting

The available hosting options are

* **IIS**
  + HTTP only
* **WAS** (Windows Activation Service)
  + Can use any transport
  + Vista and Server2008 only
* **Self hosting**
  + Can use any transport
  + Useful for communicating between applications
  + Self hosting can be done in any .NET application: WPF, Winforms, WF, console, windows service, etc.

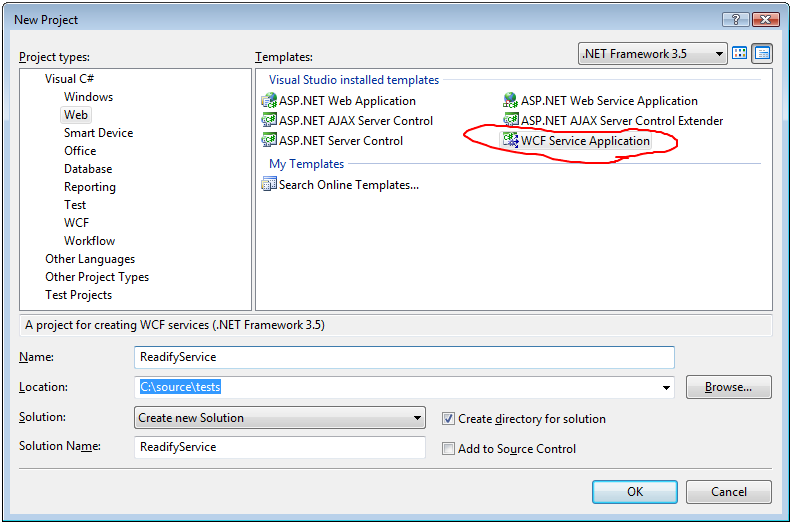
An advantage of hosting on IIS or WAS over self hosting is that the process needs to be running constantly with a self hosted solution, whereas one hosted on IIS or WAS isn’t launched until there is a client request.

# module 27: demo notes

The point of this demo is to just show the students that WCF is really easy and a client/server architecture can be setup in 5 minutes.

## Server

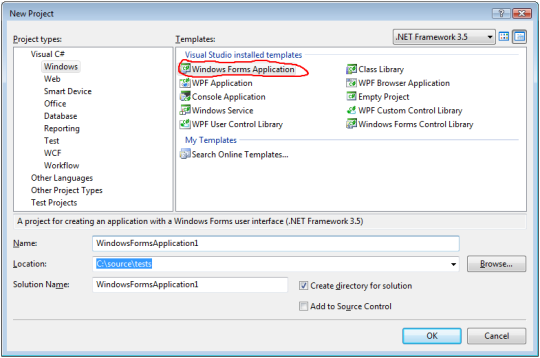
1. Open a new VS2008 session.
2. Create a new WCF service application



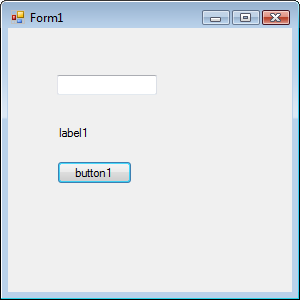
1. Press F5 to run it
2. Get the address (e.g. <http://localhost:56553/Service1.svc>)
3. Keep it open

## Client

1. Open a new VS2008 session
2. Create a winforms app



1. Rightclick the project and add a new service reference
2. Paste in the URI for the service
3. Drag a button, textbox and label onto the form



1. Double click on the button
2. Paste in this code (or type it yourself, it is easy enough)

int number = int.Parse(textBox1.Text);

using (ServiceReference1.Service1Client serviceproxy =   
 new ServiceReference1.Service1Client() )

{

label1.Text = serviceproxy.GetData(number);

}

1. Run the program, throw a number into the textbox and click run
2. High five everyone for being able to create a network app in <5minutes