**Why WCF?**

One of the goals of WCF was to unify the programming experience for building all types of distributed applications on the Microsoft platform. Developers would be able to write many different types of distributed applications without having to learn many different programming models (ASMX, Remoting, COM+, MSMQ, and WSE).

**WCF Definition:**

**Is** the unified programming model for writing distributed applications on the Microsoft platform.

**Is** a framework for working with XML Web Services and is compatible with most technology stacks.

**Why WCF important?**

Because of services:

- **WCF:** Easiest way to produce and consume services on Microsoft platform.
- **WCF:** Developer can focus on the application rather than on communication protocol.

Because of Modern Application Architecture:

- **Account**
- Services
- **Client App**
- **Devices**

Simple—because services are the core of the global distributed network, and WCF is the easiest way to produce and consume services on the Microsoft platform. By leveraging WCF, developers can focus on their applications rather than on communication protocols. It’s a classic case of technology encapsulation and tooling. Developers are more productive if their tools encapsulate (but not hide) technical chores wherever possible. WCF, combined with Visual Studio 2008, does just this.
Modern application architecture takes devices, client software, and services into account. There is no doubt that the model of the circa 1995 Web site (host an application on a Web server and deliver the UI via HTML to any browser) will endure, but new models that combine local software with Web services will also become common.

Examples are the iPod, XBOX 360, RSS, AJAX, Microsoft Office, and SharePoint and 3D immersive environments, where they each combine locally installed software and Web services.

For consumer applications, the prevalent Web service interface circa 2008 is Representational Entity State Transfer (REST). This combines HTTP and a good URI scheme for addressing XML-based data. Data manipulation using REST typically mirrors the Create Read Update Delete (CRUD) pattern, and simplicity is the hallmark of the REST protocol. For business applications, the prevalent Web service interface circa 2008 is Simple Object Access Protocol (SOAP). This provides a more robust model for exchanging complex data. SOAP messages contain an envelope and body so they can be encrypted and securely routed around the Internet.

If the message is part of a logical session or transaction, semantics are placed in the envelope and propagate along with the message.

If the information must be secured, the body of the message can be encrypted, with security information placed in the envelope.

SOAP messages are strongly typed, which makes them developer friendly. Like REST, SOAP messages circa 2008 are primarily transmitted over HTTP and encoded as text.

Notes: Developers writing the code for business transactions in a service, or compelling user experiences in a client, typically prefer not to work directly with XML. Why not? Because decades of language research and compiler design have produced much better tools.
Working with objects, classes, and components produces more robust code than laborious string manipulation in XML. Developers building .NET applications use Visual Studio. WCF and Visual Studio provide the tooling for implementing services.

WCF has a built-in model for hosting, so services can reside within IIS or in Managed Services on Windows. It provides a rich threading and throttling model where instancing is controlled with minimal effort. Whether defining a singleton or a multithreaded service to handle simultaneous requests, the programming model remains constant, and the developer is insulated (but not obfuscated) from the details.

WCF supports various message exchange patterns, such as request response, one-way, and duplex. Peer networking is also supported by leveraging mesh networks and addressing so that clients can find and communicate with each other without a central control mechanism.

**Summary:**
WCF matters because the modern applications are all about services, and that’s what WCF is all about.