|  |
| --- |
| Here are the 10 important differences between WCF Services and ASP.NET Web Services:<http://1.bp.blogspot.com/_hGraFK2-w-4/TTNPXRFz7VI/AAAAAAAAAKE/hY9jffzAVxg/s1600/WCFandWebServicesDiff.PNG> |

**Web Service in ASP.NET**

A Web Service is programmable application logic accessible via standard Web protocols. One of these Web protocols is the Simple Object Access Protocol (SOAP). SOAP is a W3C submitted note (as of May 2000) that uses standards based technologies (XML for data description and HTTP for transport) to encode and transmit application data.

Consumers of a Web Service do not need to know anything about the platform, object model, or programming language used to implement the service; they only need to understand how to send and receive SOAP messages (HTTP and XML).

**WCF Service**

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

**In what scenarios must WCF be used**

* A secure service to process business transactions.
* A service that supplies current data to others, such as a traffic report or other monitoring service.
* A chat service that allows two people to communicate or exchange data in real time.
* A dashboard application that polls one or more services for data and presents it in a logical presentation.
* Exposing a workflow implemented using Windows Workflow Foundation as a WCF service.
* A Silverlight application to poll a service for the latest data feeds.

**Features of WCF**

* Service Orientation
* Interoperability
* Multiple Message Patterns
* Service Metadata
* Data Contracts
* Security
* Multiple Transports and Encodings
* Reliable and Queued Messages
* Durable Messages
* Transactions
* AJAX and REST Support
* Extensibility

**Difference between Web Service in ASP.NET & WCF Service**

WCF is a replacement for all earlier web service technologies from Microsoft. It also does a lot more than what is traditionally considered as "web services".

WCF "web services" are part of a much broader spectrum of remote communication enabled through WCF. You will get a much higher degree of flexibility and portability doing things in WCF than through traditional ASMX because WCF is designed, from the ground up, to summarize all of the different distributed programming infrastructures offered by Microsoft. An endpoint in WCF can be communicated with just as easily over SOAP/XML as it can over TCP/binary and to change this medium is simply a configuration file mod. In theory, this reduces the amount of new code needed when porting or changing business needs, targets, etc.

ASMX is older than WCF, and anything ASMX can do so can WCF (and more). Basically you can see WCF as trying to logically group together all the different ways of getting two apps to communicate in the world of Microsoft; ASMX was just one of these many ways and so is now grouped under the WCF umbrella of capabilities.

Web Services can be accessed only over HTTP & it works in stateless environment, where WCF is flexible because its services can be hosted in different types of applications. Common scenarios for hosting WCF services are IIS,WAS, Self-hosting, Managed Windows Service.

The major difference is that Web Services Use XmlSerializer. But WCF Uses DataContractSerializer which is better in Performance as compared to XmlSerializer.

**Key issues with XmlSerializer to serialize .NET types to XML**

* Only Public fields or Properties of .NET types can be translated into XML
* Only the classes which implement IEnumerable interface
* Classes that implement the IDictionary interface, such as Hash table cannot be serialized

**Important difference between DataContractSerializer and XMLSerializer**

* A practical benefit of the design of the DataContractSerializer is better performance overXmlserializer.
* XML Serialization does not indicate which fields or properties of the type are serialized into XML whereasDataCotractSerializer
* Explicitly shows the which fields or properties are serialized into XML
* The DataContractSerializer can translate the HashTable into XML

## The need for standards

One of the most common issues involved in web communications is the mismatch in the format of the data sent. Every proprietary system is free to use whatever rules it wishes when composing a message between its modules. With the advance of the internet, however, it became clear that the cost of the synchronization between the different applications/platforms is too high to afford, and software community began to look for a standardized solutions to address the problem.

One of these solutions is **SOAP**.

## What is SOAP ?

[**The Simple Object Access Protocol (SOAP)**](http://en.wikipedia.org/wiki/SOAP) is an attempt to define a standard for creating web service APIs. It is a pattern, a web service architecture, which specifies the basic rules to be considered while designing web service platforms. It typically uses HTTP as a layer 7 protocol, although this is not mandatory. The SOAP message itself consists of an envelope, inside of which are the SOAP headers and body, the actual information we want to send. It is based on the standard XML format, designed especially to transport and store structured data. SOAP may also refer to the format of the XML that the envelope uses.

SOAP is a mature standard and is heavily used in many systems, but it does not use many of the functionality build in HTTP. While some consider it slow, it provides a heavy set of functionality which is a necessity in many cases. It might now be the best solution for browser-based clients, due to its custom format.

## What is REST ?

[**The Representational State Transfer (REST)**](http://en.wikipedia.org/wiki/Representational_state_transfer) is another architectural pattern (resource-oriented), analternative to SOAP. Unlike SOAP, RESTful applications use the HTTP build-in headers (with a variety of media-types) to carry meta information and use the [GET, POST, PUT and DELETE verbs](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods) to perform CRUD operations. REST is resource-oriented and uses clean URLs (or RESTful URLs).

For example :

http://www.developingthefuture.com/index.php?page=foo

becomes

http://www.developingthefuture.com/foo

This kind of URLs syntax greatly improves both visibility and usability. It doesn’t use any query strings, and it also provides certain SEO benefits (the crawlers just love plain text). The body of the REST message can be XML, JSON or any other format, although JSON is usually the preferred choice. On the other hand, you can’t use JSON with SOAP. Or at least not entirely, because SOAP uses XML by definition for it’s envelope. It should also mentioned that REST is, by design, protocol-agnostic, although it is usually used with HTTP.

**Read more**: [**What's the difference between JSON and XML ?**](http://www.developingthefuture.net/difference-between-json-and-xml/)

## Atom and AtomPub

The name Atom relates to two similar terms :[*Atom Syndication Format*](http://en.wikipedia.org/wiki/Atom_%28standard%29) and [Atom Publishing Format](http://en.wikipedia.org/wiki/Atom_%28standard%29)  
The Atom Syndication Format is a XML-based format used for web feeds (or news feeds). It’s basically an alternative to RSS.

The Atom Publishing Protocol (AtomPub) is a simple HTTP-based protocol used for the management (creation, deletion, update) of web resources – the feeds at first exactly.

## And what is OData ?

Remember that part about the need for standardization ?

[**The Open Data Protocol**](http://www.odata.org/) (or simply OData) is a protocol based on AtomPub and REST. It has most of the REST specifics and more than AtomPub’s specification have defined. It’s designed to define the standard way of communication to a web service. Like SQL for relational data, you don’t care what RDBMS is down there – you simply use SQL to query data. And that’s the idea.

Don’t forget we are talking about specifications here, protocols, not implementations. Whether or not a given framework or library will implement a given functionality is up to the development team. It is also worth mentioning that OData was released by Microsoft, although it is truly open source (just like version 4 of Asp.Net MVC, by the way). It will probably be the protocol of choice for any future service-oriented technologies developed by Microsoft.

## What improvements does OData provide over AtomPub ?

AtomPub provides a way of getting, updating and deleting records from a model. It does not provide, however, a data model that defines typed or untyped values on an entity. It also doesn’t provide any kind of a build-in query language to filter the entities we need. More on that later.

## Summary

#### SOAP

- A service architecture  
- XML based  
- Runs on HTTP but envelopes the message  
- Slower than REST  
- Very mature, a lot of functionality  
- Not suitable for browser-based clients

#### REST

- A service architecture (resource-oriented)  
- Uses the HTTP headers to hold meta information (although it is protocol-agnostic)  
- Can be used with XML, JSON or whatever necessary  
- Usually used with JSON due to the easily parsable content  
- Faster than SOAP  
- It uses semantic media types

#### OData

- A protocol  
- Based on HTTP, AtomPub and JSON  
- RESTful by design  
- Microsoft implementation, yet truly open source  
- Used in the latest Microsoft service-oriented technologies (more on that later)

My idea of writing this article was to introduce you to the new Asp.Net Web API, compared to the various services in WCF (SOAP services, WebHttpServices, Data services, etc.) and to the Asp.Net MVC alternative. I decided, however, to first give you this quick introduction so you can know what’s behind the new Microsoft service-oriented technologies (although these terms are platform-agnostic) and why it was used.

In my next article on the subject, I’ll focus on the practical part and the actual technologies/frameworks that use these protocols and patterns.

If you want to read more on the topic, you can check the following [document](http://vivin.net/pub/REST.pdf) written by Vivin Suresh Paliath, which forms a nice introduction into the differences between SOA, ROA, SOAP and REST in general.

**HttpContext**

HttpContext is an object that wraps all http related information into one place.HttpContext.Current is a context that has been created during the active request. Here is the list of some data that you can obtain from it.

1. Request type (Post, Get)
2. Request parameters (querystring, posted data)
3. User's IP address
4. Cookies

Further you can control your output through this object. In Items property, which is a dictionary, you can store instances of objects to ensure that they are created once for the request. You can control the output stream applying your custom filters.