Web Services

CS5200

March 17, 2014

Web Server/Web Service

- Web services cannot be accessed directly by browsers.
- Web services are an extension of the Web and can provided by web servers.
- Web services need to be on the Web server.
- Web server provides a basic HTTP service, whereas a Web service provides services based on the operations defined in its interface.

Windows Communication Foundation WCF

- Windows Communication Foundation (WCF) is Microsoft's framework for building Service-Oriented Applications.
- It enables developers to build secure and reliable solutions that integrate across platforms and interoperate with existing investments.
- Windows Communication Foundation (WCF) is a communication infrastructure that is used to create distributed applications.

Setup

1. Download ASP.NET and WebTools for Your version of Visual Studio http://www.asp.net/downloads
2. Ensure that .NET Framework version 4 is installed

Check the ProjectTypeGuids in .csproj

ProjectTypeGuids	Support
{fae04ec0-301f-11d3-bf4b-00c04f79efbc}	C# project
{349c5851-65df-11da-9384-00065b846f21}	Web Application
{E3E379DF-F4C6-4180-9B81-6769533ABE47}	Asp.Net MVC 4

The basic tasks to perform are, in order:

1. Define the service contract (signature and data).
2. Implement the contract (create the class that implements the contract).
3. Configure the service by specifying endpoint information and other behavior information.
4. Host the service in an application.
5. Build a client application.

if you want to build a client for a pre-existing service, you start at step 5. Or if you are building a service that others will use, you may skip step 5.

Example (Step 1 & 2): Define and Implement

```
// Define the IMath contract.
[ServiceContract]
public interface IMath
{
    [OperationContract]
    double Add(double A, double B);

    [OperationContract]
    double Multiply (double A, double B);
}

// Implement the IMath contract in the MathService class.
public class MathService : IMath
{
    public double Add (double A, double B) { return A + B; }
    public double Multiply (double A, double B) { return A * B; }
}
```

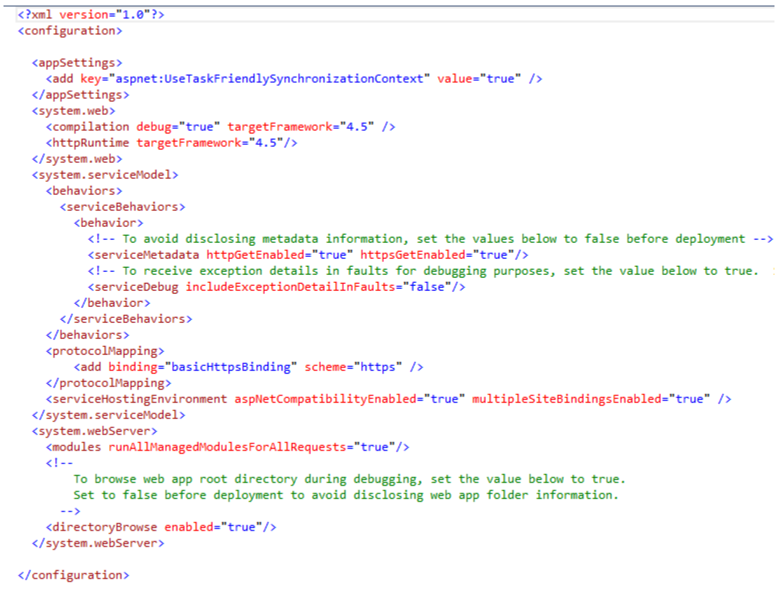
Define the Service Contract directly on the Service Class.

```
// Define the MathService contract directly on the service class.
[ServiceContract]class MathService
{
    [OperationContract]    public double Add(double A, double B) { return A + B; }
    [OperationContract]
    private double Multiply (double A, double B) { return A * B; }
}
```

Step 3: Configuring WCF

- Once you have designed and implemented your service contract, you are ready to configure your service.
- This is where you define and customize how your service is exposed to clients, including:
  - Specifying the address where it can be found,
  - The transport and message encoding it uses to send and receive messages,
  - The type of security it requires.

Step 4: Host a WCF Service

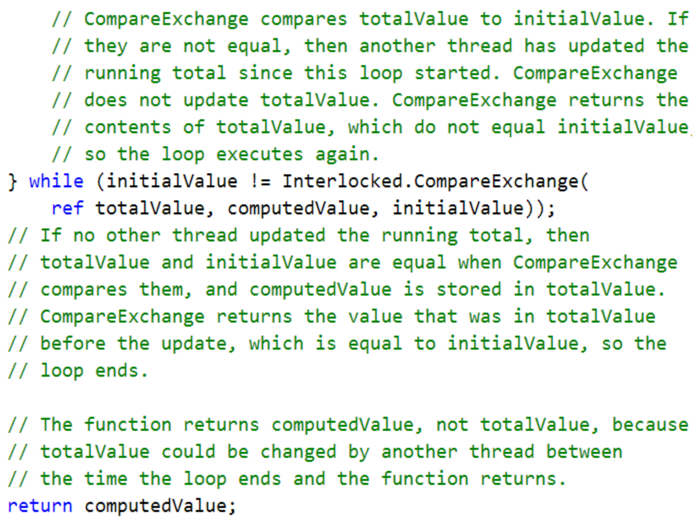


```xml
<?xml version="1.0"?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5"/>
  </system.web>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- To avoid disclosing metadata information, set the values below to false before deployment -->
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the value below to true.
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <protocolMapping>
        <add binding="basicHttpsBinding" scheme="https" />
    </protocolMapping>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true"/>
    <!--
        To browse web app root directory during debugging, set the value below to true.
        Set to false before deployment to avoid disclosing web app folder information.
      -->
    <directoryBrowse enabled="true"/>
  </system.webServer>

</configuration>
```

Threading Safe: Lock & Interlocked

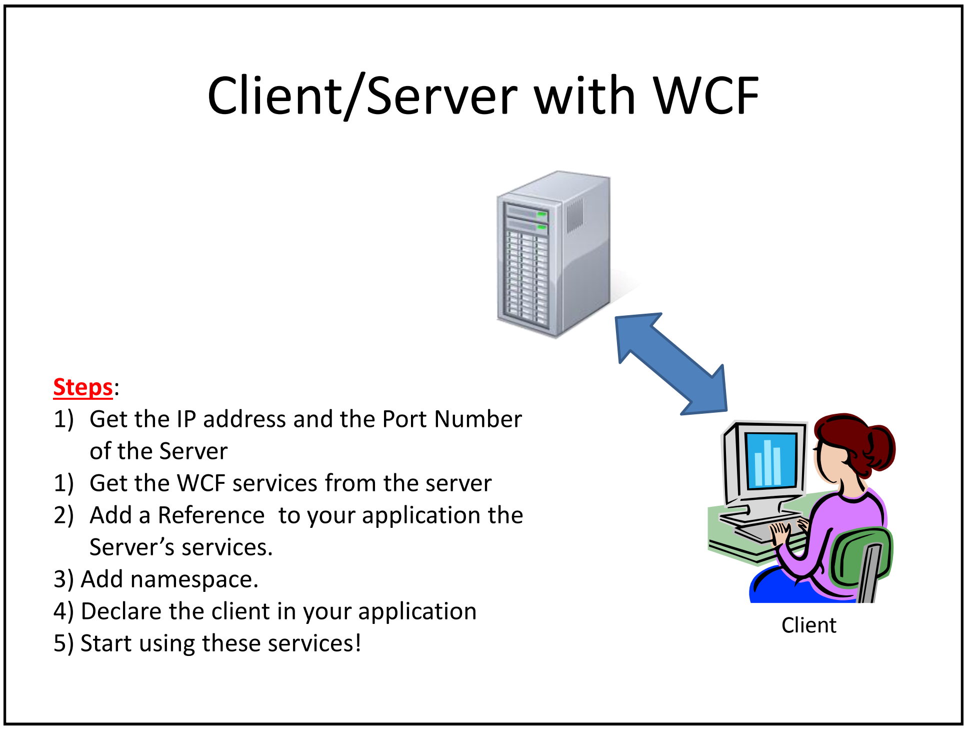
- **Interlocked:**
  - Helps with threaded programs.
  - It safely changes the value of a shared variable from multiple threads.
  - This is possible with the lock statement. But you can instead use the Interlocked type for simpler and faster code.

This changes the operations to be atomic. This means no operations can be performed on the value during the call.

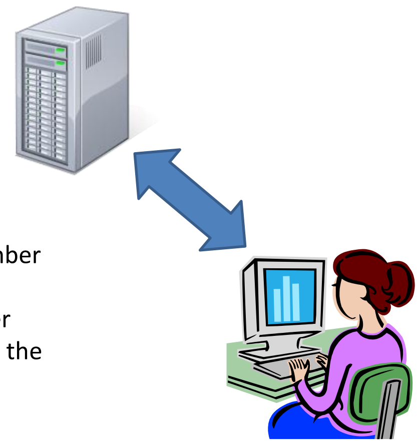


```
    // CompareExchange compares totalValue to initialValue. If
    // they are not equal, then another thread has updated the
    // running total since this loop started. CompareExchange
    // does not update totalValue. CompareExchange returns the
    // contents of totalValue, which do not equal initialValue,
    // so the loop executes again.
} while (initialValue != Interlocked.CompareExchange(
    ref totalValue, computedValue, initialValue));
// If no other thread updated the running total, then
// totalValue and initialValue are equal when CompareExchange
// compares them, and computedValue is stored in totalValue.
// CompareExchange returns the value that was in totalValue
// before the update, which is equal to initialValue, so the
// loop ends.

// The function returns computedValue, not totalValue, because
// totalValue could be changed by another thread between
// the time the loop ends and the function returns.
return computedValue;
```

© "http://msdn.microsoft.com/en-us/library/801kt583(v=vs.110).aspx"



Client/Server with WCF



Steps:

1) Get the IP address and the Port Number of the Server
1) Get the WCF services from the server
2) Add a Reference to your application the Server's services.

3) Add namespace.
4) Declare the client in your application
5) Start using these services!