Telerik Academy

Windows Communication Foundation (WCF)

Web Services & Cloud

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

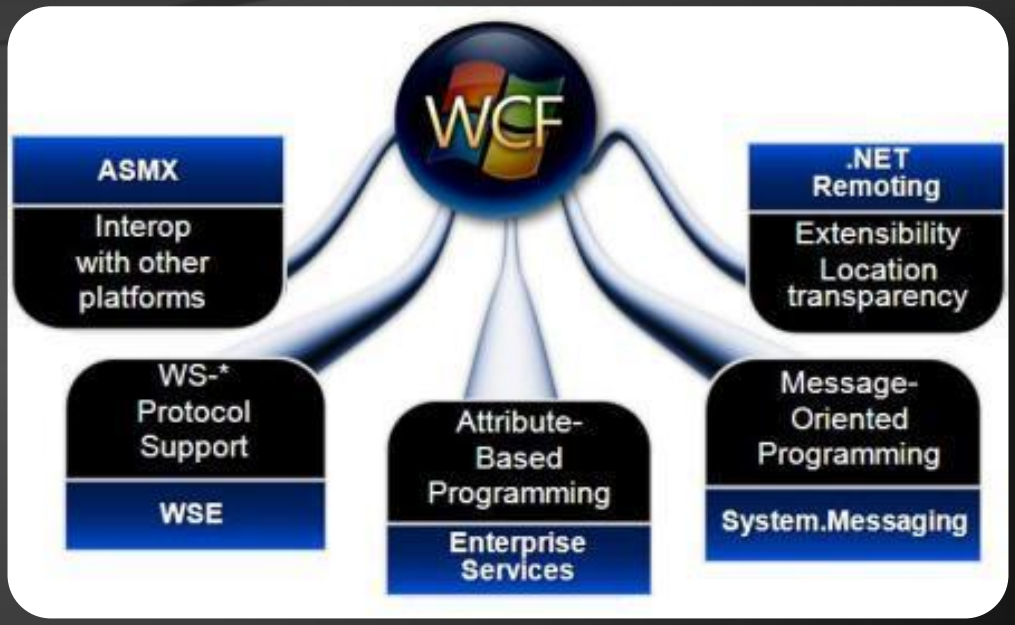
1. WCF – Quick Introduction
2. Creating a WCF Service
   - Creating a Service Contract
   - Implementing the Service Contract
   - Configuring the Service
3. Hosting WCF Services (Self-Hosting and in IIS)
4. Consuming WCF Services Through Proxies
   - Generating Proxies with the `svcutil.exe`
   - Adding Service Reference in Visual Studio

Telerik Academy

Table of Contents (2)

5. The WCF Test Client (WcfTestClient.exe)
6. Asynchronous Calls to WCF Services
7. WCF RESTful Services

Telerik Academy

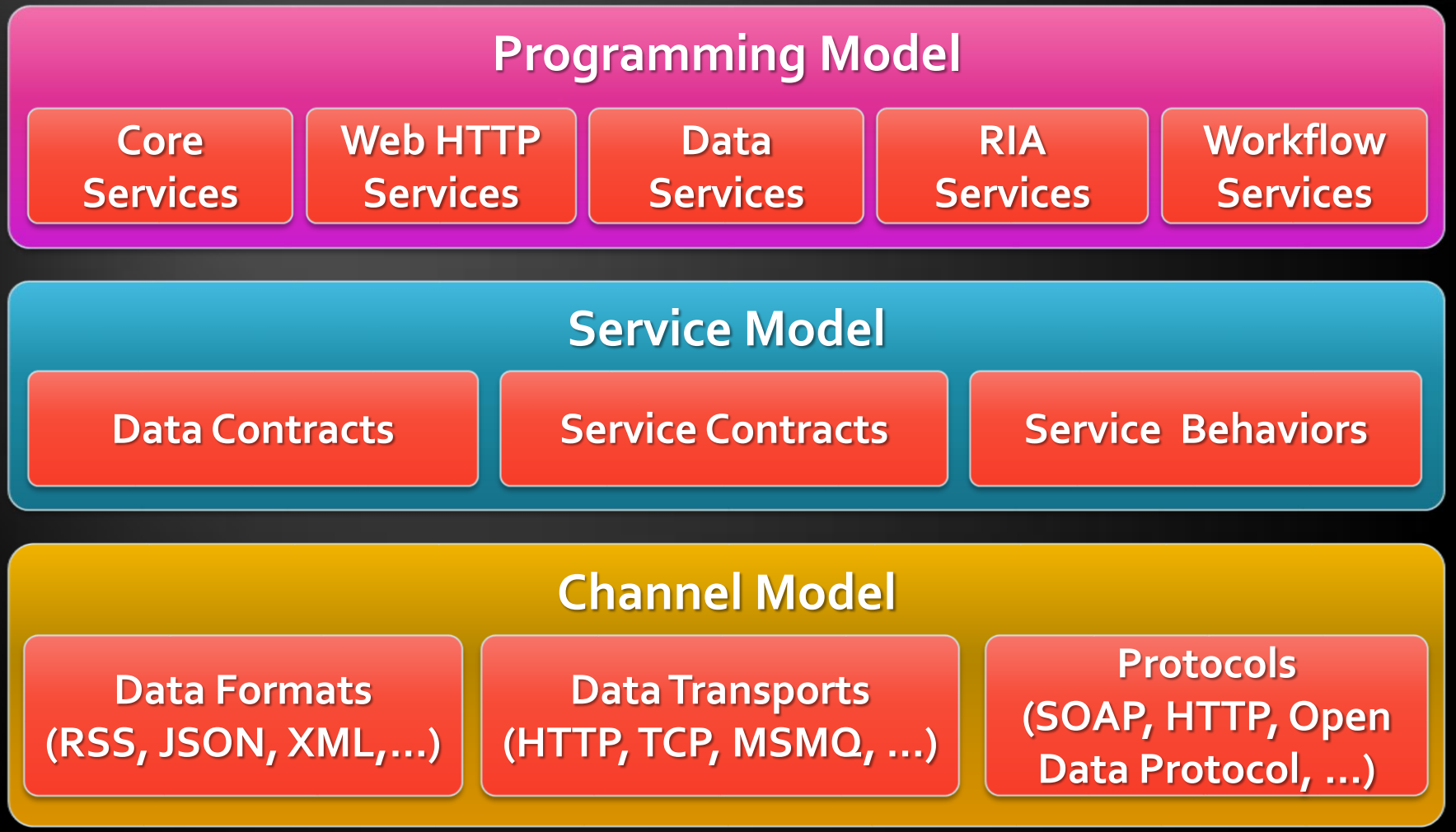


Windows Communication Foundation (WCF)

Quick Introduction

Telerik Academy

Windows Communication Foundation (WCF)



Telerik Academy

WCF Structure: Contracts

- Service contract
  - Describes the service methods, parameters, and returned result

```
[ServiceContract]
public interface ISumator
{
  [OperationContract]
  int Sum(int a, int b);
}
```

- Data contract
  - Describe the data types used in the contract and their members

```
[DataContract]
public class Person
{
  [DataMember]
  string Name {get; set;}
}
```

Telerik Academy

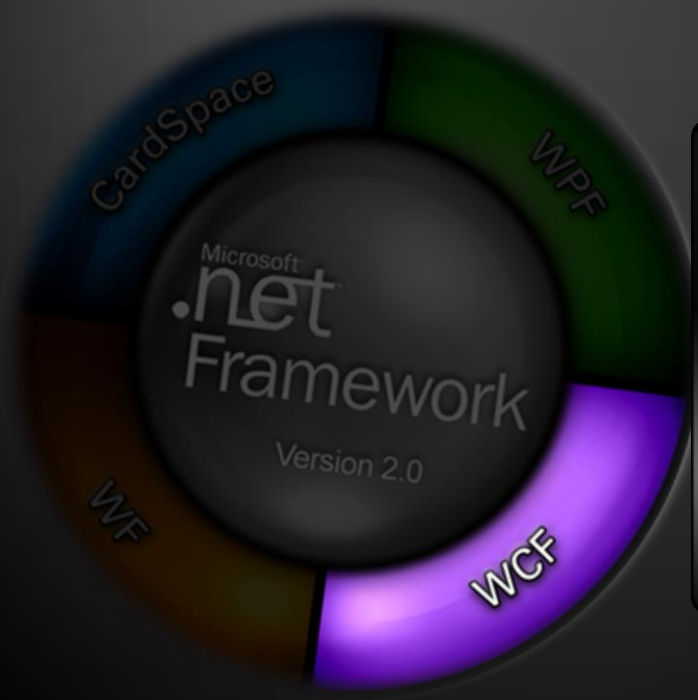
WCF Structure: Bindings and Addressing

- Binding
  - Transport – HTTP, TCP, MSMQ, …
  - Channel type – one-way, duplex, request-reply
  - Encoding – XML, JSON, binary, MTOM
  - WS-* protocols – WS-Security, WS-Addressing, WS-ReliableMessaging, WS-Transaction, …
- Addressing
  - `http://someURI`
  - `net.tcp://someOtherURI`
  - `net.msmq://stillAnotherURI`

Telerik Academy

Creating a WCF Service

Step by Step



Telerik Academy

Using WCF: Creating and Consuming Services

- WCF is a powerful Microsoft technology stack for the SOA world
- Three steps to use WCF:
  - Create a WCF Service
    - Service contract, data contract, implementation
  - Host the WCF Service
    - Several hosting options: IIS, standalone, …
  - Consume (use, invoke) the WCF service from a client application

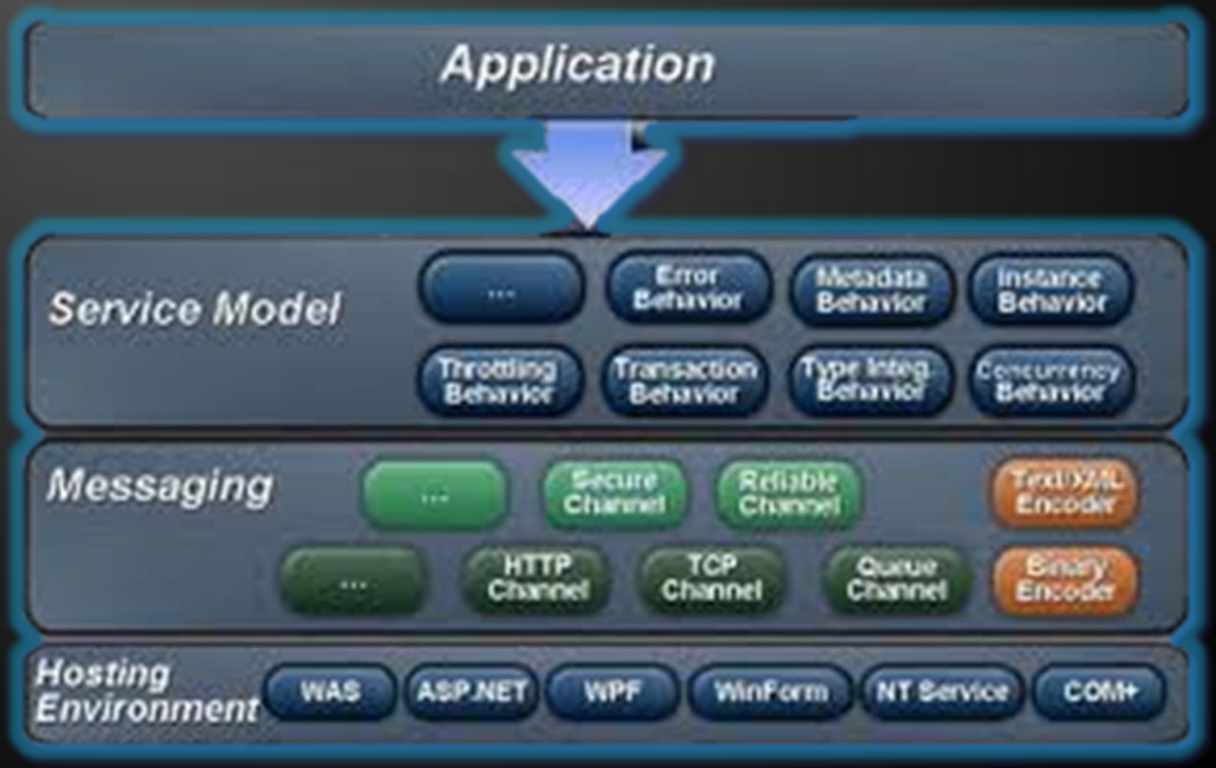
Creating a WCF Service

- To create a WCF Web Service use one of the following VS templates:
  - WCF Service Application
  - WCF Service Library
- Three steps to create WCF services:
  - Define one or more interfaces describing what our service can do (operations)
  - Implement those interfaces in one or more classes (implementation)
  - Configure the service (XML configuration file)

Telerik Academy

Creating a WCF Service

Step 1: Defining a Service Contract



Telerik Academy

Defining a Service Contract

- In order for a WCF service to be used you have to define a service contract
  - Defines the functionality offered by the service to the outside world
  - Describes to potential users how they communicate with the service
- The WSDL description of the service depends on the service contract
- To define a contract create an interface marked with the `[ServiceContract]` attribute

Telerik Academy

Defining a Service Contract (2)

- Extensive use of attributes:
  - `ServiceContractAttribute` – for the exposed service interface
    - `OperationContractAttribute` – for the contract methods that a client can see and use
  - `DataContractAttribute` – for any classes used as parameters in a method or returned as a result from a method
    - `DataMemberAttribute` – the properties of those classes visible to the clients

Defining a Service Contract – Example

```
[ServiceContract]
public interface IServiceTest
{
  [OperationContract]
  string GetData(int value);

  [OperationContract]
  CompositeType GetDataUsingDataContract(
    CompositeType composite);
}

[DataContract]
public class CompositeType
{
  bool boolValue = true;
  string stringValue = "Hello";
  …
}
```

Telerik Academy

Creating a WCF Service

Step 2: Implementing the Service Contract

Implementing a WCF Service

- Once a contract is created, the functionality offered by the service has to be implemented
- We need a class that implements the interface that defines the contract
- For example:

```
public class ServiceTest : IServiceTest
{
  public string GetData(int value)
  {
    return string.Format("You entered: {0}", value);
  }
  …
}
```

Telerik Academy

Creating a WCF Service

Step 3: Configuring the Service

Configuring Endpoints

- WCF services are visible to the outside world through the so called endpoints
- Each endpoint consists of three parts:
  - `Address` – the `URL` of the endpoint
    - Can be relative to the base address
  - `Binding` – the protocol over which the communication with the service is made
  - `Contract` – which service contract is visible through this endpoint

Configuring Endpoints (2)

- Configure endpoints in two ways
  - In the application settings
    - By a special section `<system.serviceModel>` in the application configuration file
      - `Web.config` / `App.config`
    - The most frequently used approach
  - By C# code:

```
selfHost.AddServiceEndpoint(typeof(ICalculator),
  new WSHttpBinding(), "CalculatorService");
```

Configuring Endpoints in Config Files

- The `<system.serviceModel>` contains many configuration options
- Two very important elements:
  - `<services>` – defines services and endpoints
  - `<service>` elements are configured by their `behaviorConfiguration` attribute
    - Points to an existing behavior
  - They have endpoint elements
    - The endpoints are configured through attributes

Telerik Academy

Configuring Endpoints in Config Files – Example

```
<system.serviceModel>
  <services>
    <service name="WcfServiceTest.ServiceTest"
     behaviorConfiguration=
      "WcfServiceTest.ServiceTestBehavior">
      <endpoint address="" binding="wsHttpBinding"
       contract="WcfServiceTest.IServiceTest">
        <identity>
          <dns value="localhost"/>
        </identity>
      </endpoint>
      <endpoint address="mex"
       binding="mexHttpBinding"
       contract="IMetadataExchange"/>
  </service>
</services>
```

Telerik Academy

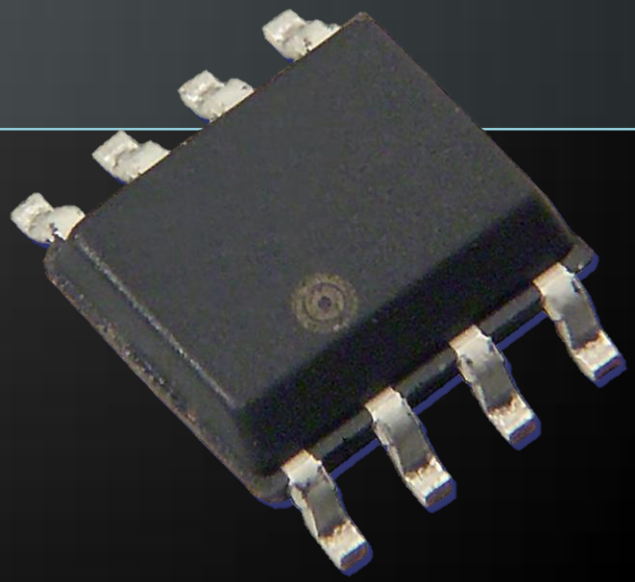
Configuring Services in Config Files

- The `serviceBehaviors` section defines behaviors (configurations) for services
- Services are configured by specifying which behavior they use
  - Many sub-elements
    - `serviceMetadata` – metadata options
    - `serviceDebug` – debug options
    - `dataContractSerializer` – controls the way data contracts are serialized
    - …

Telerik Academy

Configuring Services in Config Files – Example

```
<serviceBehaviors>
  <behavior
    name="WcfServiceTest.ServiceTestBehavior">
      <serviceMetadata httpGetEnabled="true"/>
      <serviceDebug
        includeExceptionDetailInFaults="false"/>
  </behavior>
</serviceBehaviors>
```



Telerik Academy

Microsoft .NET | Windows Communication Foundation

Creating and Configuring a WCF Service

Live Demo

Telerik Academy

Hosting a WCF Service

Hosting WCF Service

- The WCF model has many hosting options
  - Self hosting
    - In a console application, Windows Forms or WPF application
    - Mainly for development and testing
  - Windows Service
  - In IIS 5.1 (or later), 6.0 or 7.0
    - Used in production environment
  - Windows Process Activation Service (WAS)

Telerik Academy

Hosting a WCF Service in IIS

- Hosting WCF services in IIS is a typical scenario
  - IIS itself handles creation and initialization of the `ServiceHost` class
- We need a `*.svc` file to host in IIS
- Base content of the `.svc` file:

```
<%@ServiceHost Language="C#" Debug="true"
  Service="WcfServiceTest.ServiceTest" %>
```

- It has the `ServiceHost` directive
  - Tells IIS in what language is the service written and the main class of the service

Telerik Academy

Hosting a WCF Service in IIS (2)

- The code for the service (the contract and the class implementing it) can be stationed in three places
  - Inline – following the `ServiceHost` directive
  - In an assembly in the `Bin` directory
  - As `.cs` files in the `App_Code` directory
- You must set the necessary permissions in order for IIS to have access to your files

Telerik Academy

Hosting WCF Services in IIS

Live Demo

The ServiceHost Class

- Hosting of the service is managed by the `ServiceHost` class
  - Located in `System.ServiceModel` namespace
  - You must have reference the `System.ServiceModel.dll` assembly
- In self-hosting scenarios we initialize and start the host ourselves

```
ServiceHost selfHost = new ServiceHost(
  typeof(CalculatorService),
  new Uri("http://localhost:1234/service"));
selfHost.Open();
```

Telerik Academy

Microsoft .NET | Windows Communication Foundation

Self-Hosting a WCF Service

Live Demo

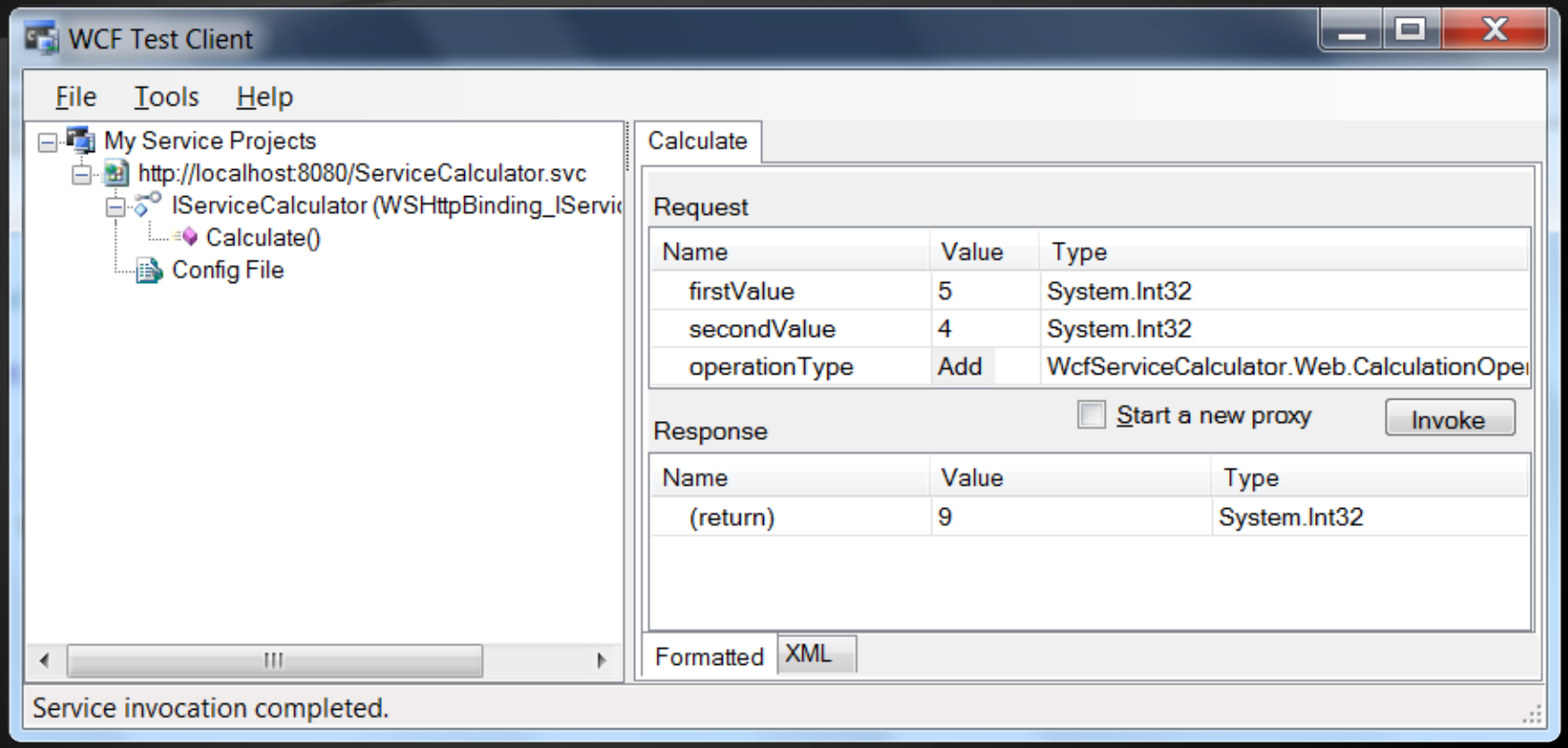
Using the WCF Test Client

- The WCF Test Client is a GUI tool for playing with WCF services
  - Enables users to input test parameters, submit that input to the service, and view the response
  - WCF Test Client (`WcfTestClient.exe`) is part of Visual Studio
    - Can be started from the VS Command Prompt
  - Typical location:

```
C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\
```

  - Starts on [F5] in VS for WCF Service Libraries

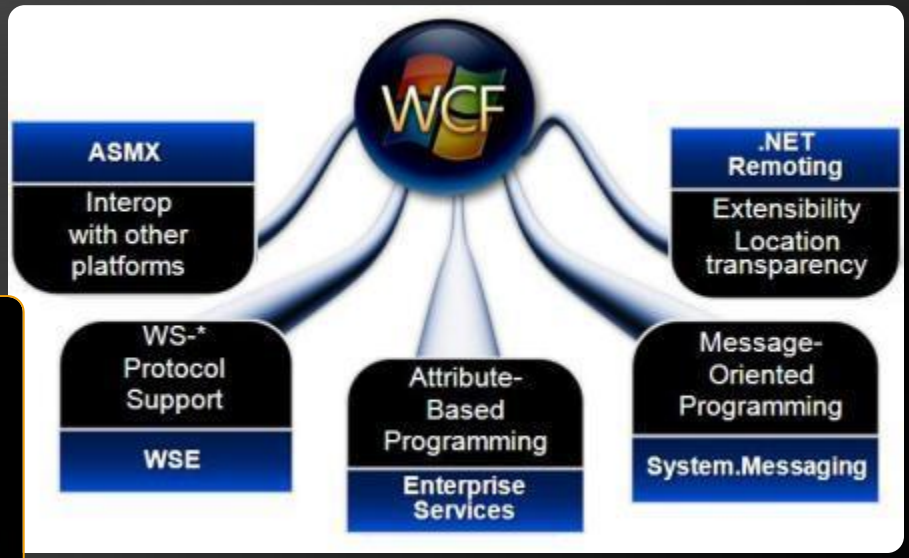
Telerik Academy



WCF Test Client

Live Demo

Telerik Academy



Consuming a WCF Service

Consuming WCF Services

- To consume a WCF service we make use of the so called proxy classes
- Those look like normal classes in the code
  - When you call a method of the proxy it makes a SOAP request to the service
  - When the service returns the SOAP result the proxy transforms it to a .NET type

Generating a Proxy Class

- Proxy classes can be generated using the `svcutil.exe` tool
  - The simplest form (from the command line):

```
svcutil http://localhost:8080/ServiceCalculator.svc
```

  - The service must be configured to allow getting its metadata
    - Via `WS-Metadata-Exchange` or `Disco`
  - The result is the `AddService.cs` class and the `output.config` file

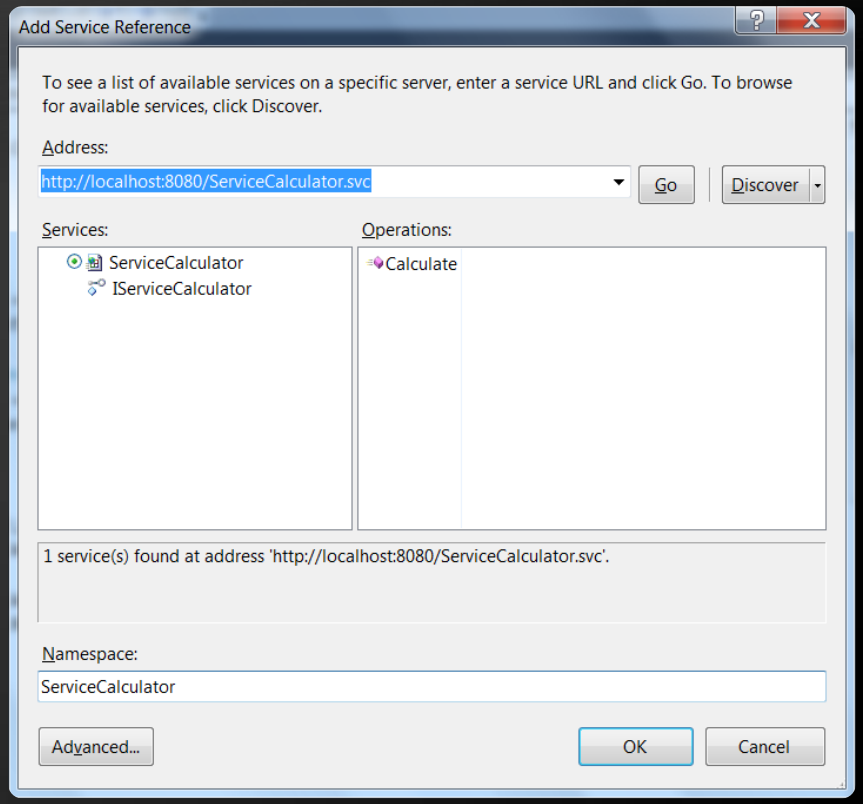
Generating a Proxy Class (2)

- There are many options to this command line
  - You can set the names of the generated files
  - You can set the namespace
  - Asynchronous methods generation
- The `AddService.cs` must be visible in our project
- The contents of the `output.config` must be merged with the `app.config` / `web.config`
- No automerging is supported

Telerik Academy

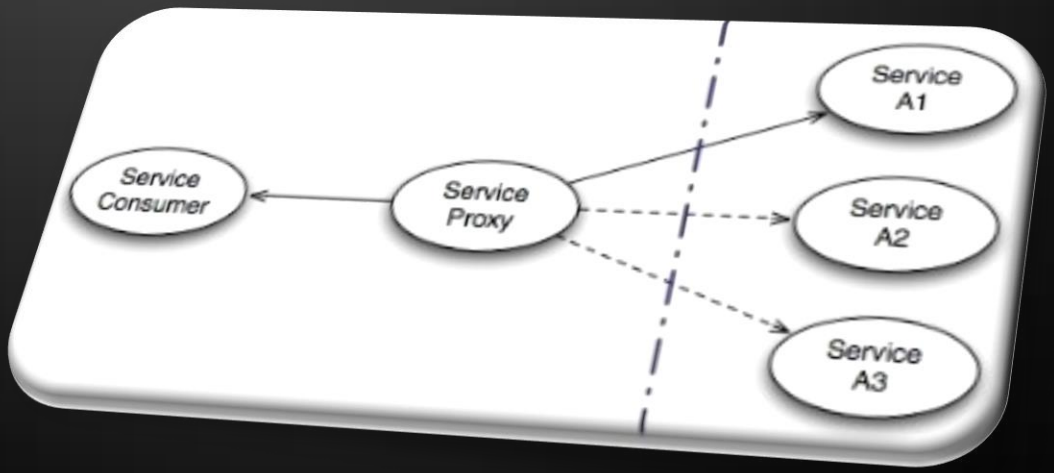
Generating a Proxy Class in VS

- In Visual Studio you can also use the "Add Service Reference" option in a project
  - It's a wrapper around `svcutil.exe`
  - A set of XML config files is generated



Generating a Proxy Class in VS (2)

- When adding a service reference
  - Entries are added to the `web.config` or `app.config` respectively
  - Don't manipulate those entries manually unless you know what you're doing



Using a Proxy Class

- Using proxy classes is identical to using normal classes but produces remote calls
  - You instantiate the proxy class
  - Then call its methods

```
AddServiceClient addSevice =
  new AddServiceClient();
int sum = addService.Add(5, 6);
```

  - Call the `Close()` method of the proxy
- If the interface of the service changes you have to regenerate the proxy class

Telerik Academy

Consuming a WCF Service in Visual Studio

Live Demo

Telerik Academy

Asynchronous Calls to a WCF Service

Calling a WCF Service Asynchronously

- Sometimes a service call takes too long and we don't need the result immediately
- We can make asynchronous calls in two ways
  - By using a delegate
    - The delegate has this functionality built-in
  - By generating the proxy class with the "`svcutil /async`" option
    - Creates asynchronous methods as well
    - When adding a service reference in Visual Studio there is an option under Advanced Settings

Calling a WCF Service Asynchronously (2)

- When the proxy class is generated to support asynchronous calls we get few new methods
  - For every `OperationContract` method we get the `BeginXXX()` and `EndXXX()` methods
  - You can make use of the callback method

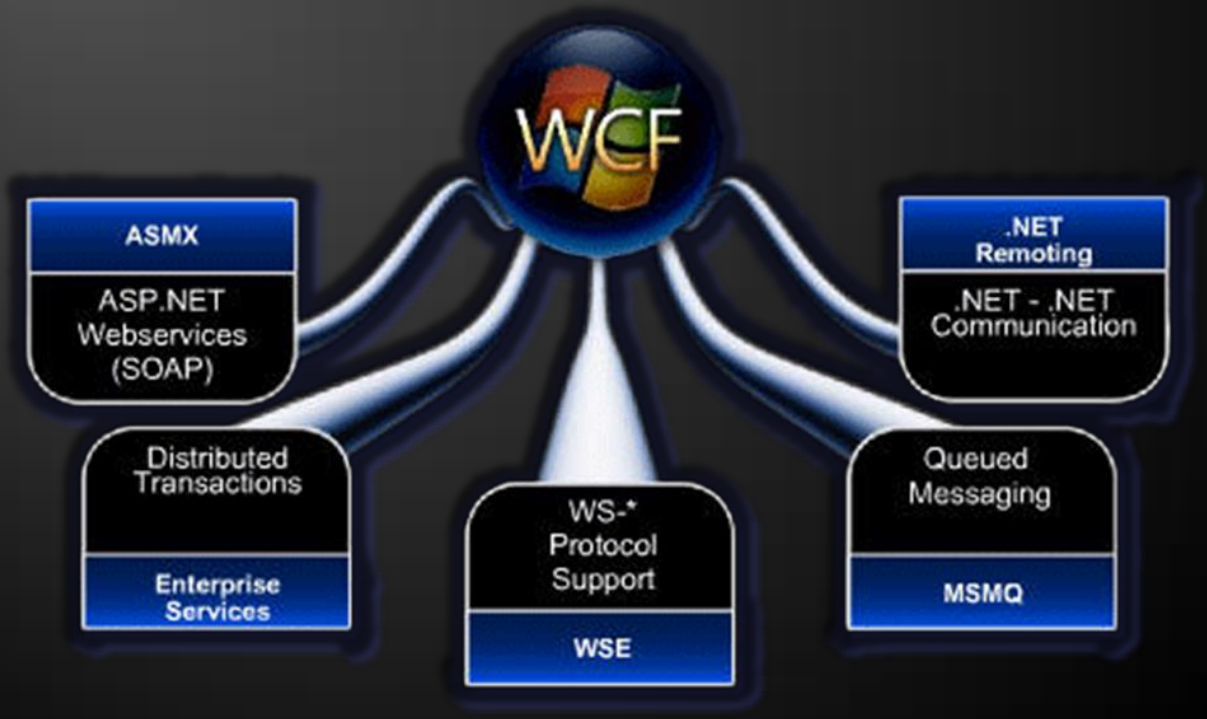
```
public void static Main() {
  AsyncCallback cb = new AsyncCallback(CallFinished);
  service.BeginLongProcess(params, cb, service);
}

private void CallFinished(IAsyncResult asyncResult) {
  Console.WriteLine("Async call completed.");
}
```

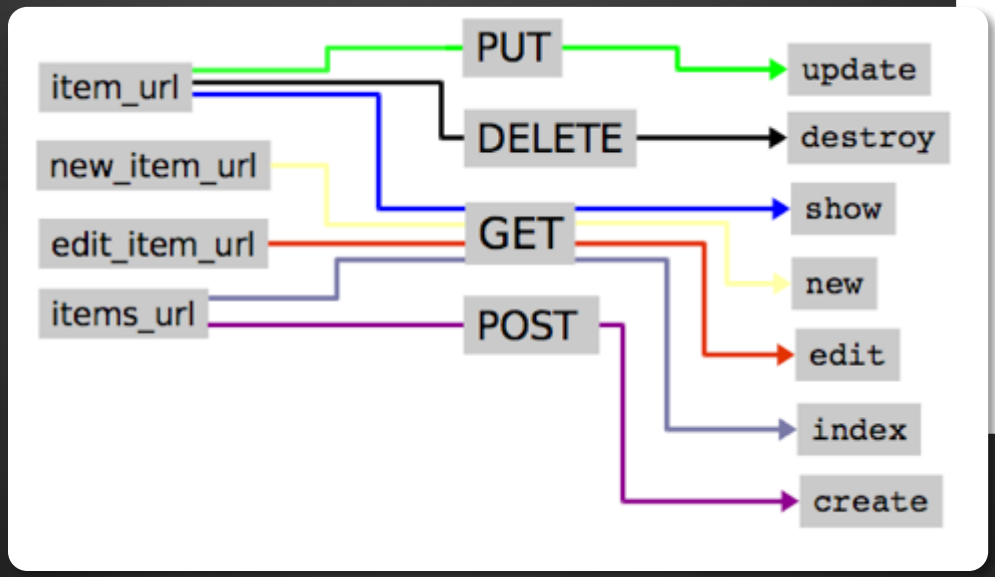
Telerik Academy

Asynchronous WCF Calls

Live Demo



Telerik Academy



RESTful WCF Services

Creating and Consuming RESTful WCF Services

RESTful WCF Services

- In the `.svc` file add the following:

```
<%@ ServiceHost Language="C#" Service="…" CodeBehind="…"
Factory="System.ServiceModel.Activation.WebServiceHostFactory" %>
```

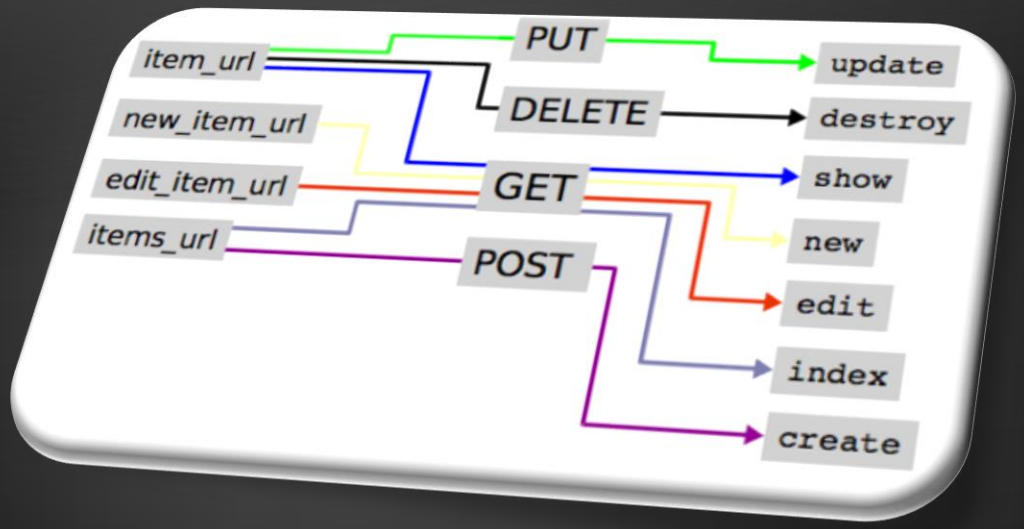
- Use special binding in `Web.config`:

```
<system.serviceModel> <standardEndpoints> <webHttpEndpoint>
<standardEndpoint defaultOutgoingResponseFormat="Json"
  helpEnabled="true" />
</webHttpEndpoint> </standardEndpoints> </system.serviceModel>
```

- Use URL mapping in the service contract

```
[OperationContract]
[WebInvoke(Method = "GET",
  UriTemplate = "Category/{categoryID}")]
Category FindCategoryByID(string categoryID);
```

Telerik Academy



RESTful WCF Services

Live Demo

Telerik Academy

Windows Communication Foundation (WCF)

Exercises

1. Create a simple WCF service. It should have a method that accepts a DateTime parameter and returns the day of week (in Bulgarian) as string. Test it with the integrated WCF client.
2. Create a console-based client for the WCF service above. Use the "Add Service Reference" in Visual Studio.
3. Create a Web service library which accepts two string as parameters. It should return the number of times the second string contains the first string. Test it with the integrated WCF client.
4. Host the latter service in IIS.
5. Create a console client for the WCF service above. Use the `svcutil.exe` tool to generate the proxy classes.