**WCF Behaviors**

<http://sankarsan.wordpress.com/2009/01/04/wcf-behavior/#comments>

Posted: January 4, 2009 in [Uncategorized](http://sankarsan.wordpress.com/category/uncategorized/), [WCF](http://sankarsan.wordpress.com/category/wcf/)  
Tags: [Behaviors](http://sankarsan.wordpress.com/tag/behaviors/), [WCF](http://sankarsan.wordpress.com/tag/wcf/)

The various WCF behaviors and how they can be plugged into the WCF runtime.

There are 4 types of behaviors in WCF: Service, Endpoint, Operation and Contract ,each with a different scope of effect. To create each of this behavior we need to implement a separate interface having the same methods with similar intentions but with different parameters. The purpose of these methods is described below:

1. *AddBindingParameters* – This method is used to pass custom information to the binding.
2. *Validate* – This method is used to validate the service description.
3. *ApplyDispatchBehavior/ApplyClientBehavior* – This method is used to add the extension components to DispatchRuntime/ClientRuntime e.g. MessageInspector, ParameterInspectors etc.

The WCF behaviors can be added to the runtime using the following ways:

* Programmatically – This I will discuss in details with the individual types of behavior
* Attributes – In this case the behavior class needs to implement the corresponding behavior interface and extend the attribute class.Then that attribute needs to be applied to the service class implementing the ServiceContract as shown below:

public class ServicePoolBehavior : Attribute, IServiceBehavior

{

// Code

}

[ServiceContract]

public interface ITest

{

// Code

}

[ServicePoolBehavior(MaxPoolSize=10,MinPoolSize=2,IncrementSize=2)]

public class Test : ITest

{

// Code

}

* Configuration – Here we need to implement a custom behavior extension element by extending the System.ServiceModel.BehaviorExtensionElement class and then add the new behavior in the config as shown below:

public class ServicePoolBehaviorExtensionElement : BehaviorExtensionElement

{

// Code

public override Type BehaviorType

{

get { return typeof(ServicePoolBehavior); }

}

protected override object CreateBehavior()

{

ServicePoolBehavior behavior = new ServicePoolBehavior();

behavior.MaxPoolSize = Convert.ToInt32(this.ElementInformation.Properties["maxPoolSize"].Value);

behavior.MinPoolSize = Convert.ToInt32(this.ElementInformation.Properties["minPoolSize"].Value);

behavior.IncrementSize = Convert.ToInt32(this.ElementInformation.Properties["incrementSize"].Value);

return behavior;

}

}

<extensions>

<behaviorExtensions>

<add name="poolBehavior" type=”SB.ServiceModel.Pool.ServicePoolBehaviorExtensionElement, SB.ServiceModel.Pool, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null” />

</behaviorExtensions>

</extensions>

<behaviors>

<serviceBehaviors>

<behavior name=”testBehavior”>

<serviceDebug includeExceptionDetailInFaults=”true” />

<serviceMetadata httpGetEnabled=”true” httpGetUrl=”http://localhost:9001/Meta” />

<poolBehavior minPoolSize=”2″””maxPoolSize=”10″””incrementSize=”2″/>”

</behavior>

</serviceBehaviors>

</behaviors>

Now we will discuss about the individual types of behaviors:

* ServiceBehavior
  + Need to implement the System.ServiceModel.Description.IServiceBehavior interface.
  + This behavior applies for the entire service runtime including the ServiceHostBase.
  + This behavior can be added by **attribute,configuration or code**.The following snippet shows how we can add a service behavior programmatically:

using (ServiceHost host = new ServiceHost(typeof(SampleService.Test)))

{

SB.ServiceModel.Pool.ServicePoolBehavior behavior = new SB.ServiceModel.Pool.ServicePoolBehavior();

behavior.IncrementSize = 2;

behavior.MaxPoolSize = 10;

behavior.MinPoolSize = 2;

host.Description.Behaviors.Add(behavior);

// Code

}

* EndpointBehavior
  + Need to implement the System.ServiceModel.Description.IEndpointBehavior interface.
  + This behavior customizes the runtime of a particular endpoint and associated EndpointDispatcher object.
  + This behavior can be added by **configuration or code**.The following snippet shows how we can add a endpoint behavior programmatically:

foreach (ServiceEndpoint se in host.Description.Endpoints)

{

se.Behaviors.Add(new CustomBehavior());

}

* ContractBehavior
  + Need to implement the System.ServiceModel.Description.IContractBehavior interface.
  + This behavior customizes the runtime of a particular contract and associated DispatchRuntime/ClientRuntime object.
  + This behavior can be added by **attribute or code**.The following snippet shows how we can add a contract behavior programmatically:

foreach (ServiceEndpoint se in host.Description.Endpoints)

{

se.Contract.Behaviors.Add(new CustomBehavior);

}

* OperationBehavior
  + Need to implement the System.ServiceModel.Description.IOperationBehavior interface.
  + This behavior customizes the runtime of a particular operation and associated DispatchOperation/ClientOperation object.
  + This behavior can be added by **attribute or code**.The following snippet shows how we can add a operation behavior programmatically:

foreach (ServiceEndpoint se in host.Description.Endpoints)

{

foreach (OperationDescription od in se.Contract.Operations)

{

od.Behaviors.Add(new CustomBehavior);

}

}

With this much understanding what all different behaviors are available and how to hook them to the runtime we are ready to explore the individual extension points in details from next post onwards.