WCF Session-2 : Configuration files(host settings)

Gursharan Singh Virdi

WCF:Why Configuration Files?

- Issues with our previous solution
  - In the previous example we hard coded the Service Contract, binding and host address parameters into the service host. This makes any the application inflexible and hence we don't want this setup for our live applications.
  - We copied the service contract meta data into the client along with the host address and bindings.This again is not going to happen in real world situations.
  - Let us see how we can decouple these parameters from code and put them into configuration files so that any future changes are just localized to the config files making our application scalable.

WCF: Configuration Files

- **Add app.config file to the host**
  - Create <system.serviceModel></system.serviceModel> section.
  - Inside <system.serviceModel> add <services></services> section. The name of the service is applied here.
  - The name of the service is the CRL type implementing the [ServiceContract] ie. In our case Customer.CustomerService.

    <services name="namespace.servicename"> or

    <services name="Customer.CustomerService">
  - Next we define the endpoint along with the bindingType,ServiceContract name and address.

    <endpoint binding="basicHttpBinding"
    contract="Customer.ICustomerService"
    address="http://localhost:8000/Customer>

WCF: Configuration Files

- **Extracting metadata declaratively**
  - What we want is to extract the service metadata without having to copy across the whole signature to the client.For this we need to declare another metadata endpoint called “mexHttpBinding”.
  - Define the metadata endpoint along with the bindingType,ServiceContract name and address.

    <endpoint binding=“mexHttpBinding” contract=“IMetadataExchange” address=“mex”/>

    For this example I will use a relative address with respect to the base address which I will show in the next slide.

WCF: Configuration Files

- **Extracting metadata declaratively(contd..)**
  - To use relative addressing we need to declare <host> section inside the <services> section.

    <host>

    <baseAddress>

    <add baseAddress="http://localhost:8000">

    </baseAddress>

    </host>
  - Now the http address can be changed from

    <endpoint binding="basicHttpBinding" contract="Customer.ICustomerService" address="http://localhost:8000/Customer> to

    address="Customer"

WCF: Configuration Files

- **Extracting metadata declaratively(contd..)**
  - We declare the metadata settings for the service in our service host but most importantly we need to turn on the service behaviour to enable metadata exchange.
  - For this we need to modify our configuration file and add the <behaviours><serviceBehaviour> tags as shown below

```
<behaviours>
   <serviceBehaviour>
     <behaviour name="serviceBehaviour">
       <serviceMetaData httpGetEnabled="true"/>
     </behaviour>
   </ serviceBehaviour >
</ behaviours >
```

  - Now our metadata is exposed over http.We still need to make it accessible.

WCF: Configuration Files

- **Extracting metadata declaratively(contd..)**
  - We declared the service behaviour next we associate the behaviour with our service by modify the <Service> tag a little
  - ***<services name="namespace.servicename" behaviourConfiguration= "name of behaviour defined earlier in config file" > or***

    <services name="Customer.CustomerService" behaviourConfiguration="serviceBehaviour">
  - Now our metadata is accessible over http.

WCF: Configuration Files

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="Customer.CustomerService" behaviorConfiguration="serviceBehaviour">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8000"/>
          </baseAddresses>
        </host>

        <!-- Service Endpoints -->
        <endpoint address="Customer"
                  binding="basicHttpBinding"
                  contract="Customer.ICustomerService" >
          <!--
              Upon deployment, the following identity element should be removed or replaced to reflect the
              identity under which the deployed service runs.  If removed, WCF will infer an appropriate identity
              automatically.
          -->
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
      </service>
    </services>

    <behaviors>
      <serviceBehaviors>
        <behavior name="serviceBehaviour">
          <!-- To avoid disclosing metadata information, set the value below to false and remove the metadata endpoint above before deploym
          <serviceMetadata httpGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the value below to true.  Set to false before deployment
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

WCF: Configuration Files

- **Extracting metadata declaratively(contd..)**
  - To access the metadata leave the server running
  - Open the base address http://localhost:8000 in internet explorer.
  - Click the http://localhost:8000/?wsdl link to access the metadata.

WCF: Adding Metadata to client

- **Using metadata on the client**
  - Open the client application
  - RightClick -> add Service reference
  - Add http://localhost:8000/?wsdl link to access the metadata.
  - **The host has to be running for the Endpoint to be accessable.
  - This will generate a proxy for us.
  - From the old code remove the Metadata service contract that we had manually copied.
  - The new proxy already has the been generated in the file localhost.cs automatically for use.
  - The proxy will generate a class **XXX**Client file which we need to instantiate ie. CustomerServiceClient in this case.This is a wrapper over ICustomerService datacontract.
  - Now there is no need to a reference to ChannelFactory.Just create and object of the proxy and call the required operation.

WCF: Conclusion

- Just remember the following
  - The Service Host needs to be running to access the http metadata.
  - Create a service reference
  - …And it is as simple as that ☺☺☺.
  - Go through the source code…