

Original source - Sergey Zvezdin: personal blog (<http://blog.zvezdin.com/en/38>)

Displaying of progress of downloading data in WCF services

Platform Windows Communication Foundation can be used not only for a simple operations call, but also for transfer of big parts of data (for example, files in some mbyte). Sometimes such scenarios are applicable and on slow channels. In this case an indicator of executing of operation (progress bar) is necessary.

For simplicity we will use BasicHttpBinding binding. The key moment at implementation of the given scenario is setting of TransferMode parameter to Streamed value. Thus, data will be transferred as a stream.

After that we will define the contract. It is required to transfer a file name, its size and contents to the client side.

```
01 [ServiceContract]
02 public interface IFileTransferService
03 {
04     [OperationContract]
05     RemoteFileInfo DownloadFile(DownloadRequest request);
06 }
07
08 [MessageContract]
09 public class DownloadRequest
10 {
11     [MessageBodyMember]
12     public string FileName;
13 }
14
15 [MessageContract]
16 public class RemoteFileInfo : IDisposable
17 {
18     [MessageHeader(MustUnderstand = true)]
19     public string FileName;
20
21     [MessageHeader(MustUnderstand = true)]
22     public long Length;
23
24     [MessageBodyMember(Order = 1)]
25     public Stream FileByteStream;
26
27     public void Dispose()
```

```

28     {
29         if (FileByteStream != null)
30         {
31             FileByteStream.Close();
32             FileByteStream = null;
33         }
34     }
35 }

```

Apparently, DownloadFile operation returns RemoteFileInfo object which contains the necessary information.

```

01 public class FileTransferService : IFileTransferService
02 {
03     public RemoteFileInfo DownloadFile(DownloadRequest request)
04     {
05         var filePath = request.FileName;
06         var fileInfo = new FileInfo(filePath);
07
08         if (fileInfo.Exists==false)
09         {
10             throw new FileNotFoundException("File not found", request.FileName);
11         }
12
13         var stream = new FileStream(filePath, FileMode.Open, FileAccess.Read);
14
15         var result = new RemoteFileInfo
16             {
17                 FileName = request.FileName,
18                 Length = fileInfo.Length,
19                 FileByteStream = stream
20             };
21
22         return result;
23     }
24 }

```

Now it is necessary to host service

```

01 class Program
02 {
03     static void Main()
04     {
05         var myServiceHost = new ServiceHost(typeof(FileService.FileTransferService));
06         myServiceHost.Open();
07
08         Console.ReadKey();
09

```

```

10     myServiceHost.Close();
11     }
12 }

```

Thus the configuration file will have the following data.

```

01 <?xml version="1.0" encoding="utf-8" ?>
02 <configuration>
03     <system.serviceModel>
04         <bindings>
05             <basicHttpBinding>
06                 <binding name="FileTransferServicesBinding" transferMode="Streamed"
messageEncoding="Mtom" maxReceivedMessageSize="20134217728" />
07             </basicHttpBinding>
08         </bindings>
09         <services>
10             <service behaviorConfiguration="MyServiceTypeBehaviors"
name="FileService.FileTransferService">
11                 <endpoint address="getfile"
12                     binding="basicHttpBinding"
13                     bindingConfiguration="FileTransferServicesBinding"
14                     contract="FileService.IFileTransferService" />
15                 <host>
16                     <baseAddresses>
17                         <add baseAddress="http://localhost/" />
18                     </baseAddresses>
19                 </host>
20             </service>
21         </services>
22         <behaviors>
23             <serviceBehaviors>
24                 <behavior name="MyServiceTypeBehaviors">
25                     <serviceMetadata httpGetEnabled="true" />
26                     <serviceDebug includeExceptionDetailInFaults="true" />
27                 </behavior>
28             </serviceBehaviors>
29         </behaviors>
30     </system.serviceModel>
31 </configuration>

```

Now it is necessary to create a client side.

```

01 <?xml version="1.0" encoding="utf-8" ?>
02 <configuration>
03     <system.serviceModel>
04         <bindings>
05             <basicHttpBinding>

```

```

06         <binding name="fileBinding" maxReceivedMessageSize="20134217728"
messageEncoding="Mtom" transferMode="Streamed" />
07     </basicHttpBinding>
08 </bindings>
09 <client>
10     <endpoint address="http://localhost/getfile"
11         binding="basicHttpBinding"
12         bindingConfiguration="fileBinding"
13         contract="Client.FileTransferClient.IFileTransferService" />
14 </client>
15 </system.serviceModel>
16 </configuration>

```

After generation of a client proxy it looks as follows.

```

01 [System.Diagnostics.DebuggerStepThroughAttribute()]
02 [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "3.0.0.0")]
03 public partial class FileTransferServiceClient :
04     System.ServiceModel.ClientBase<Client.FileTransferClient.IFileTransferService>,
05     Client.FileTransferClient.IFileTransferService
06 {
07     public FileTransferServiceClient ()
08     {
09     }
10
11     public FileTransferServiceClient (string endpointConfigurationName) :
12         base (endpointConfigurationName)
13     {
14     }
15
16     public FileTransferServiceClient (string endpointConfigurationName, string remoteAddress) :
17         base (endpointConfigurationName, remoteAddress)
18     {
19     }
20
21     public FileTransferServiceClient (string endpointConfigurationName,
System.ServiceModel.EndpointAddress remoteAddress) :
22         base (endpointConfigurationName, remoteAddress)
23     {
24     }
25
26     public FileTransferServiceClient (System.ServiceModel.Channels.Binding binding,
System.ServiceModel.EndpointAddress remoteAddress) :
27         base (binding, remoteAddress)
28     {
29     }
30
31     Client.FileTransferClient.RemoteFileInfo

```

```

Client.FileTransferClient.IFileTransferService.DownloadFile(Client.FileTransferClient.DownloadRequest
request)
32 {
33     return base.Channel.DownloadFile(request);
34 }
35 }

```

Let's create more convenient DownloadFile method.

```

1 public long DownloadFile(ref string fileName, out System.IO.Stream fileByteStream)
2 {
3     Client.FileTransferClient.DownloadRequest inValue = new
Client.FileTransferClient.DownloadRequest();
4     inValue.FileName = fileName;
5     Client.FileTransferClient.RemoteFileInfo retVal =
((Client.FileTransferClient.IFileTransferService) (this)).DownloadFile(inValue);
6     fileByteStream = retVal.FileByteStream;
7     return retVal.Length;
8 }
9 }

```

Now we should receive this stream and read out from it the information in the portions.

```

01 var client = new FileTransferClient.FileTransferServiceClient();
02
03 Stream inputStream;
04 long length = client.DownloadFile(ref fileName, out inputStream);
05
06 using (var writeStream = new FileStream(fileName, FileMode.CreateNew, FileAccess.Write))
07 {
08     const int bufferSize = 2048;
09     var buffer = new byte[bufferSize];
10
11     do
12     {
13         int bytesRead = inputStream.Read(buffer, 0, bufferSize);
14         if (bytesRead == 0)
15         {
16             break;
17         }
18
19         writeStream.Write(buffer, 0, bytesRead);
20
21         progressBar1.Value = (int)(writeStream.Position * 100 / length);
22     }
23     while (true);
24 }

```

```
25     writeStream.Close();
26 }
27
28 inputStream.Dispose();
29 client.Close();
```

Great! Now we have received the application which will display progress of downloading data on the client. Similarly it is possible to make uploading data to a server. If this information is interesting to you I recommend to read more detailed [article](#).

Source code:

[wcf-progress.zip](#)