

Windows Communication Foundation (WCF) Service Using VisualAPL

Windows Communication Foundation (WCF) is the methodology by which .Net assemblies containing classes with methods and properties can be exposed as web services. For an overview of the WCF technology see: <http://msdn.microsoft.com/en-us/library/ms734712.aspx>.

WCF projects are composed of server-side and web-client components.

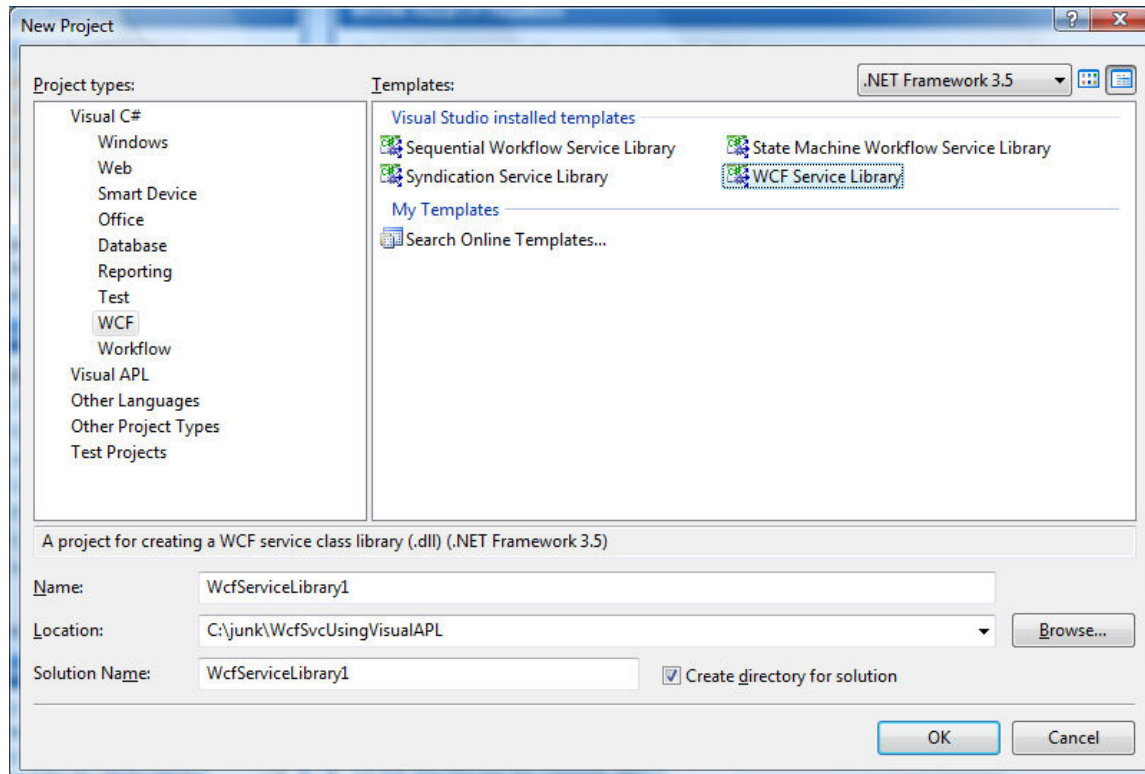
The web-client components provide the graphical user interface (GUI) of the solution. The web-client component may be supported by browser-based or locally (permanently or transiently) installed elements. VisualAPL .Net assemblies (class libraries) are useful as components of the web-client components when these are housed in locally installed elements.

The server-side components provide the business rules, calculations and data store of the solution. VisualAPL .Net assemblies (class libraries) are useful as components of the business rules and calculation elements of the server-side portion of the solution.

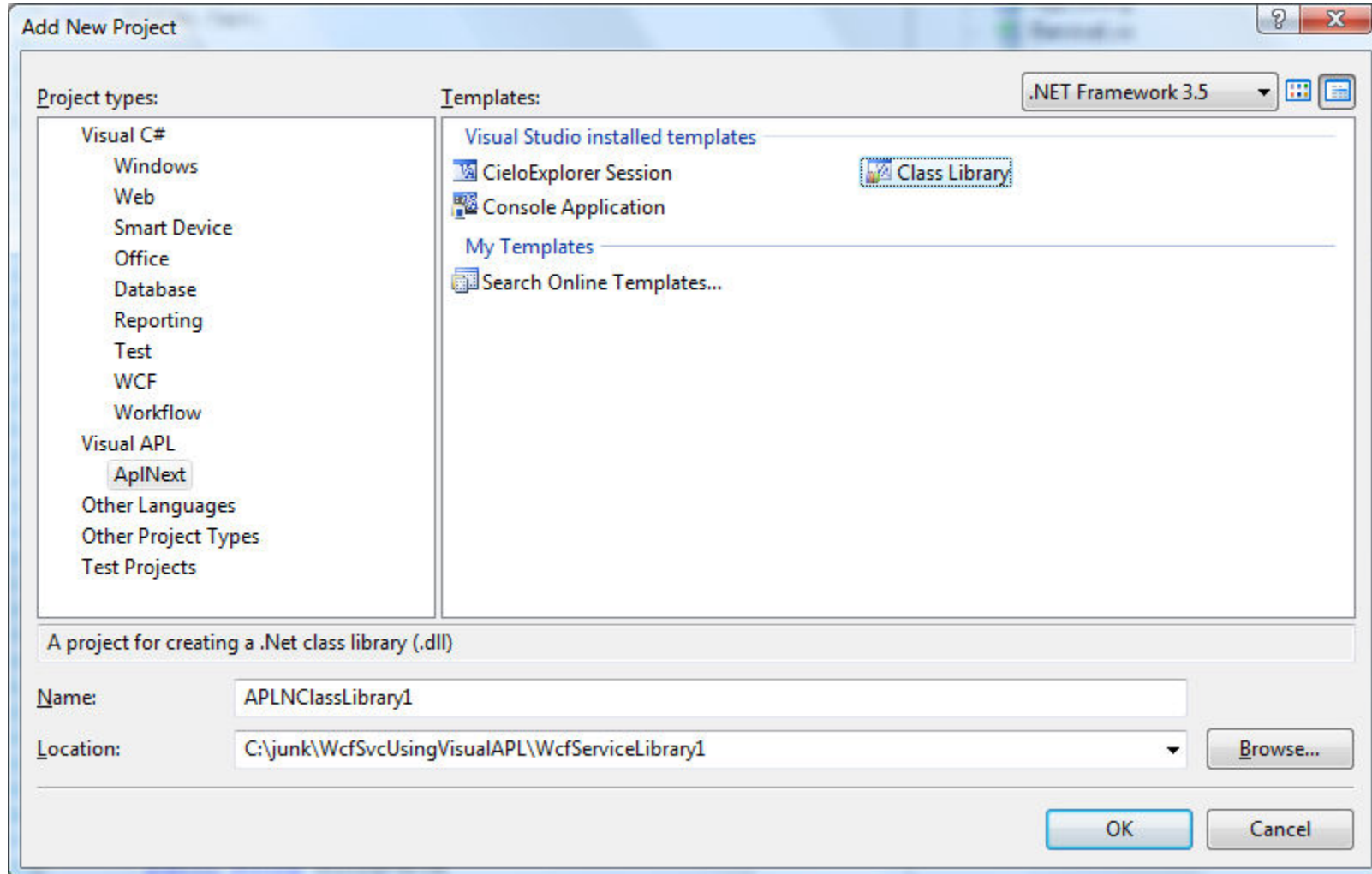
This 'n-tier' application system design methodology creates separate components for GUI (graphical user interface), business rules/calculations and data store. Designing an application system this way means that maintenance and enhancement of one of the components can be done more independently of the other components.

The following example illustrates the creation of a very simple WCF web service server-side component of an application system which incorporates a VisualAPL .Net assembly (class library).

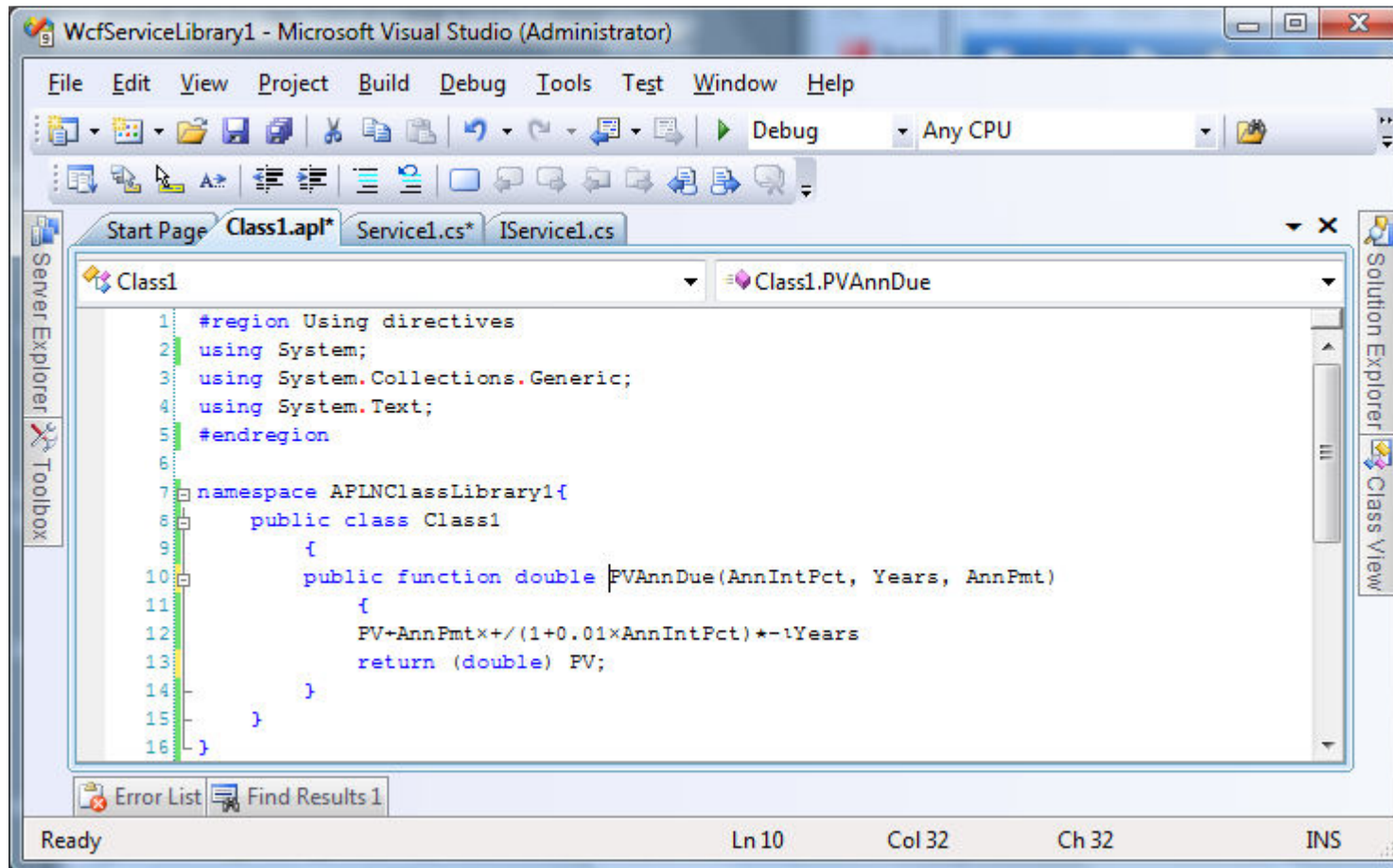
In VS2008 create a new WCF Service Library project. This project will be installed on the server side to support web clients who have access to the server via the Internet. Note that the VS2008 WCF Service Library template will create two classes “Service1.cs” which will contain the C# functions which will ‘cover’ the VisualAPL functions which will perform the web service’s calculations/business rules and the IService1.cs class which will be the WCF interface to expose the cover functions to web clients.



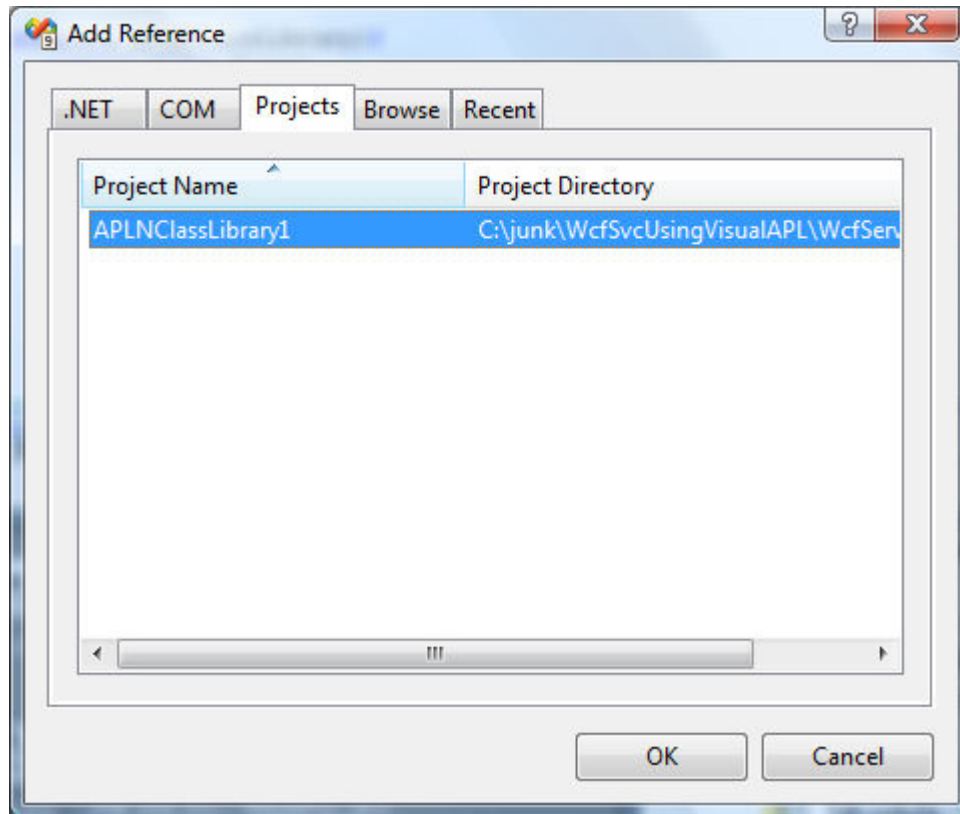
From the Solution Explorer, right click the Solution node and select Add > Project > VisualAPL > APLNext > Class Library:



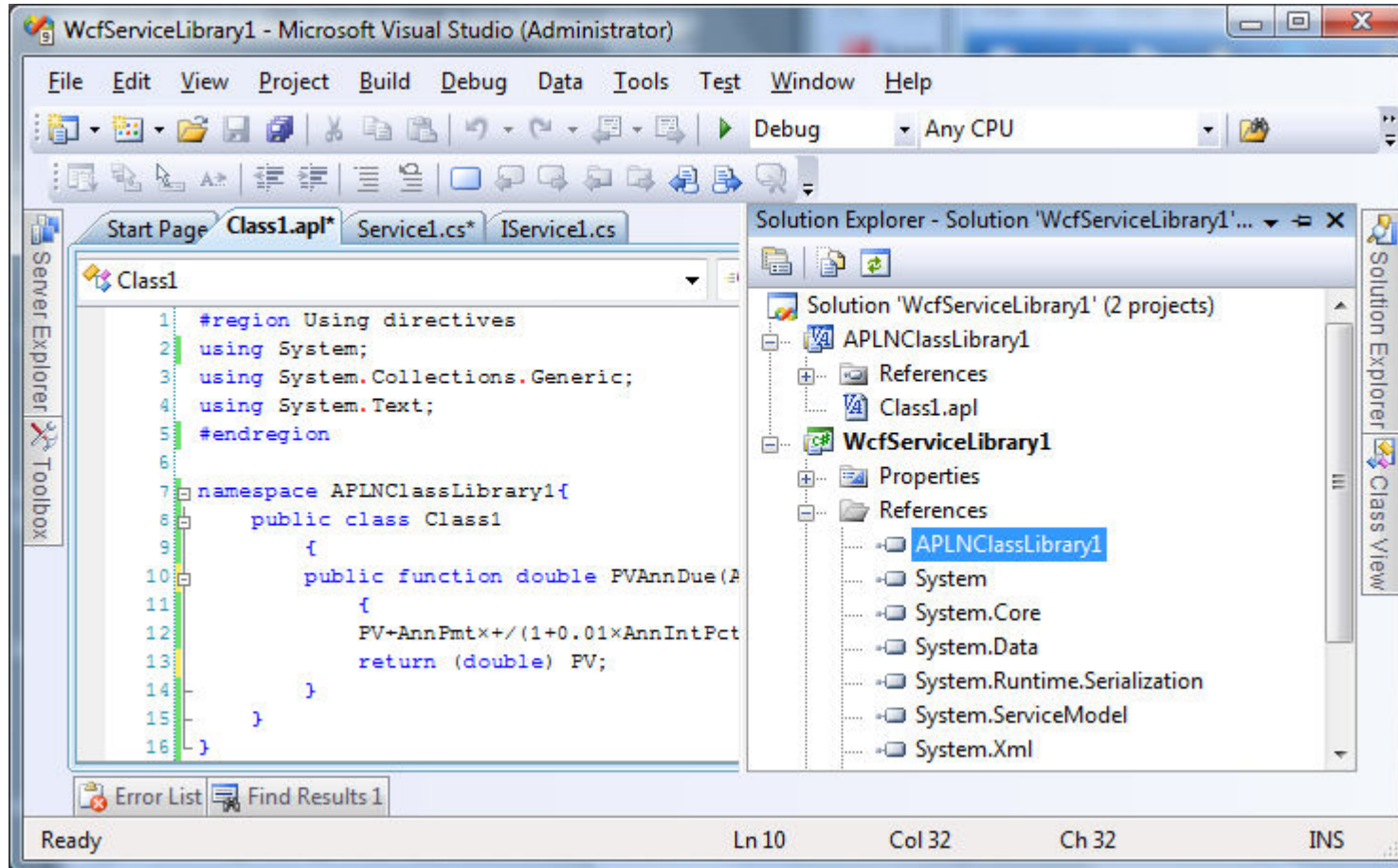
For example create a function (method) in the VisualAPL class to perform a specific calculation, in this case the present value of an immediate annuity due. Dynamic data typing can be used in VisualAPL up to the point where the information is passed back to C# which requires strong data types:



Add a reference to the APLNClassLibrary1 project by right clicking the References tab of the WcfServiceLibrary1 project in the Solution Explorer, selecting the Projects tab and double clicking the APLNClassLibrary1 project:

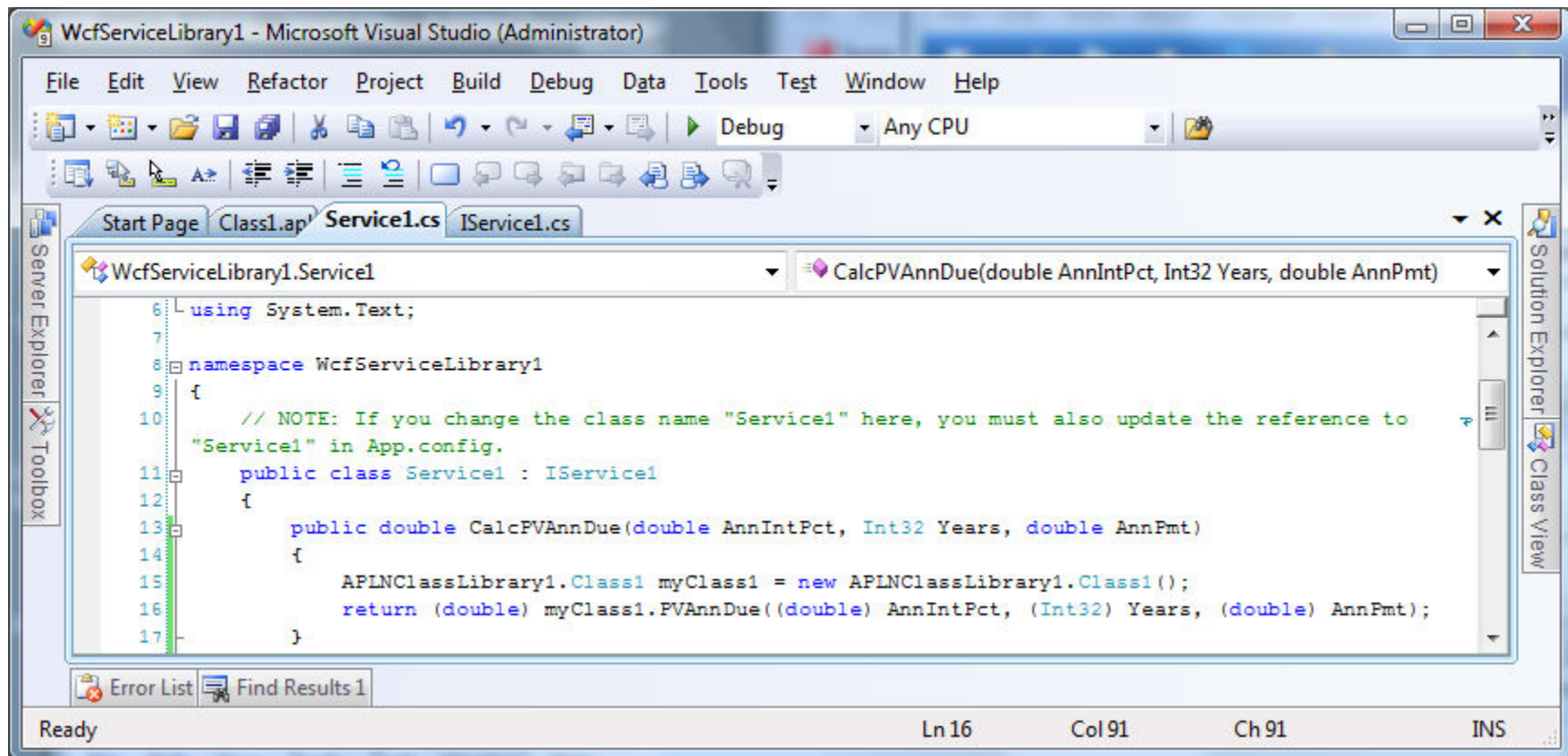


The reference to the VisualAPL class library has been added to the C# WcfServiceLibrary1 project:

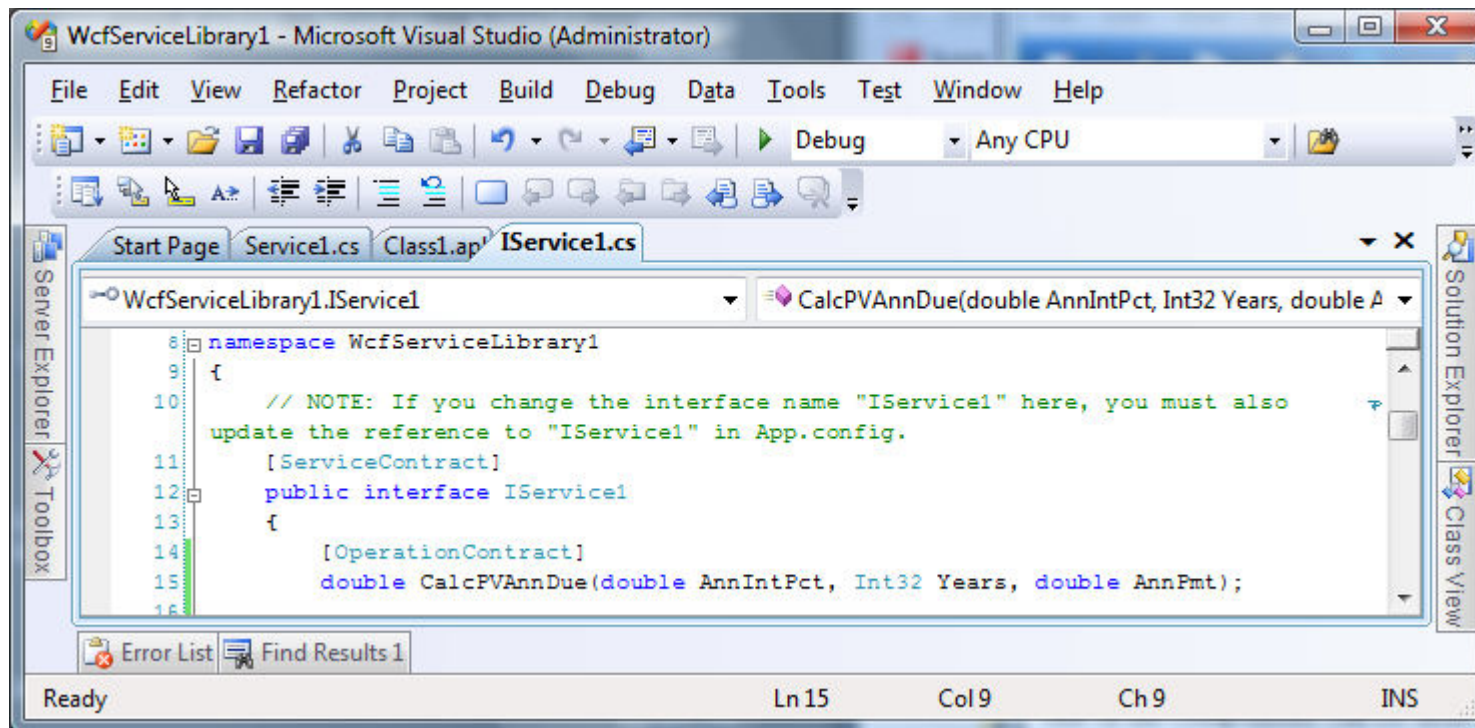


In the Service1.cs class, create a 'cover function' method which uses the APL function operation just defined. Note that the VS2008 WCF Service Library project template creates several sample functions in the Service1.cs class including the GetData() and GetDataUsingDataContract() methods. For now these can be left in the project as examples, but in a production environment they would be removed.

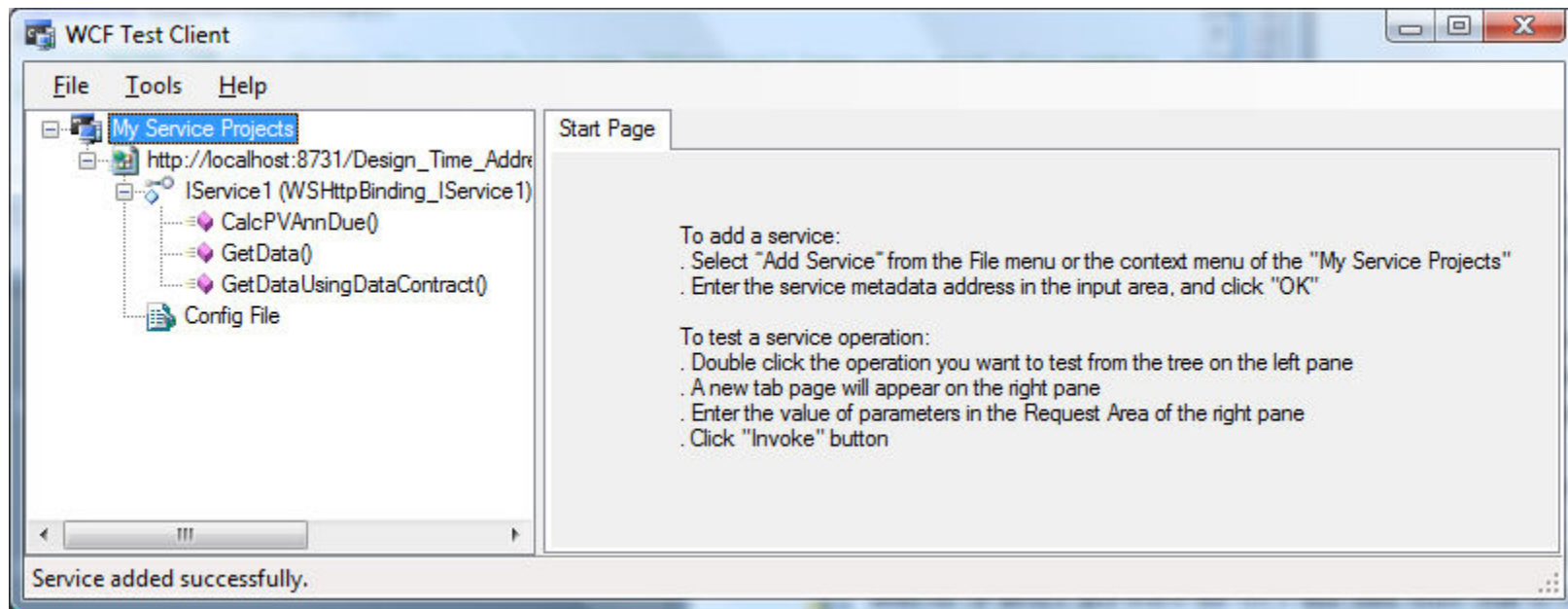
Notice how strong data types are used in the C# cover function because that is what is expected in the .Net programming world.



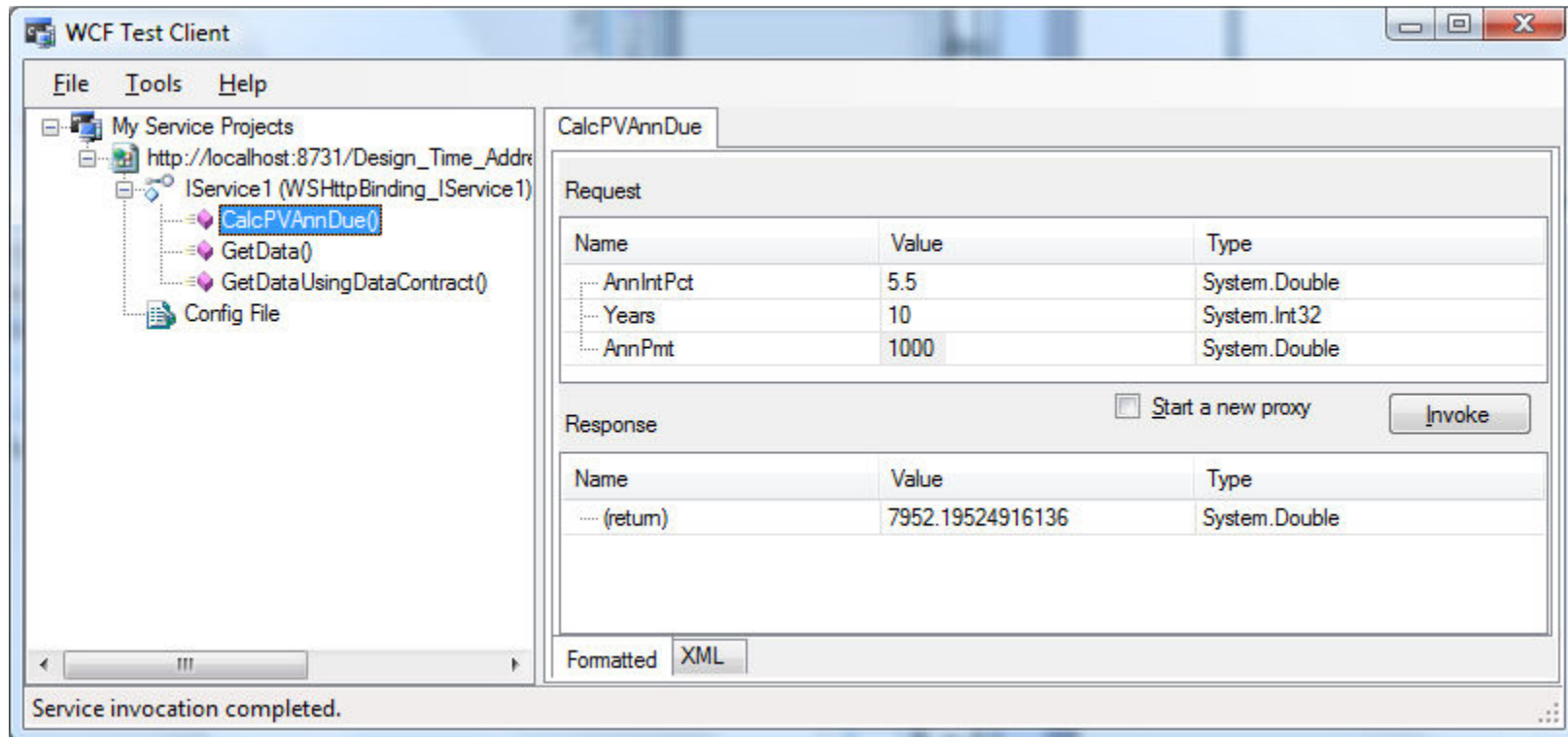
In the IService1.cs class add the C# method signature, without the public attribute, to the interface class prefixing it with the [OperationContract] attribute. Note that the VS2008 WCF Service Library project template creates several service contracts for the GetData() and GetDataUsingDataContract() methods as well as a data contract for the CompositeType class. For now these can be left in the project as examples, but in a production environment they would be removed.



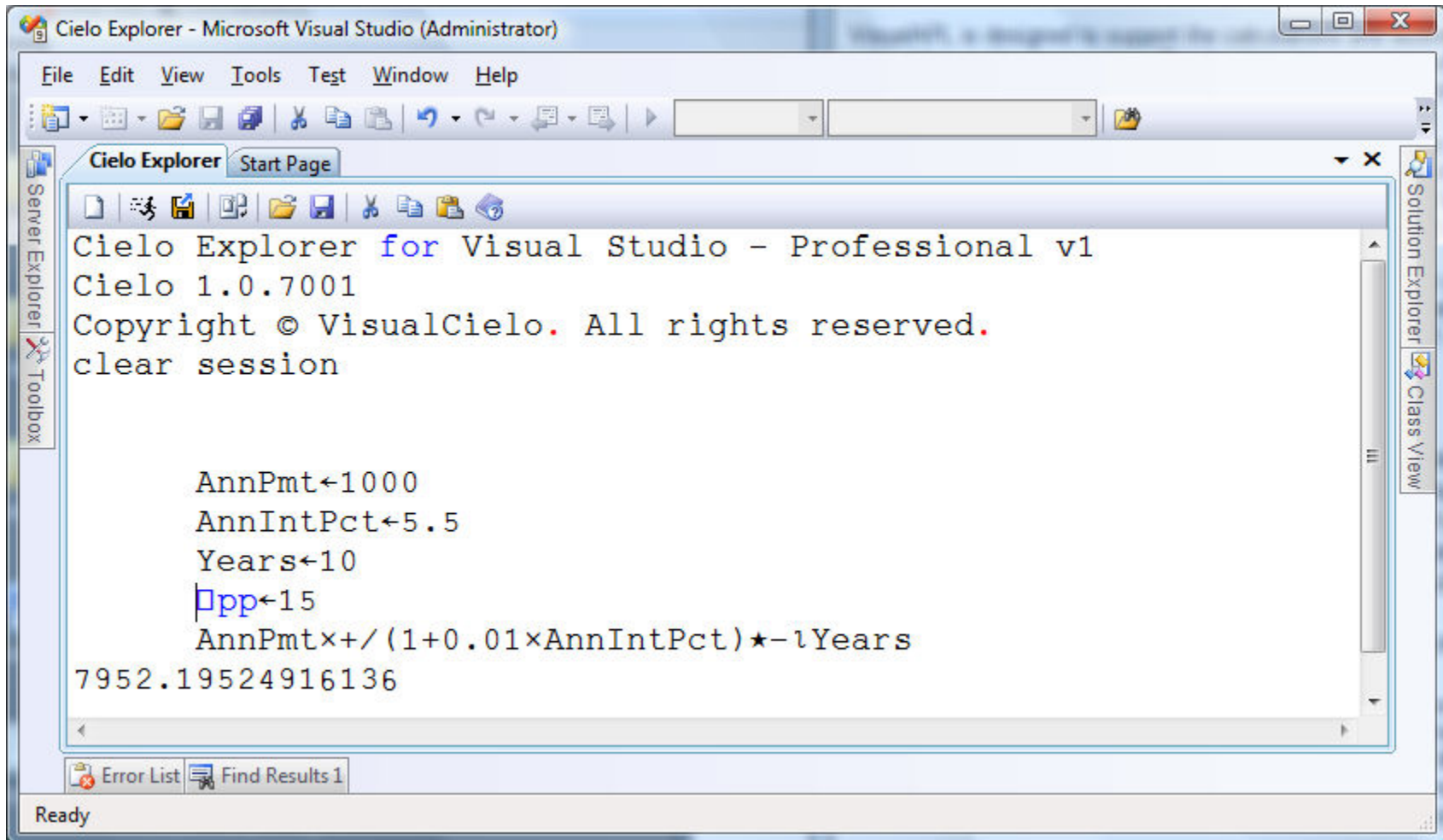
Save the entire solution and then debug it to start the WCF web service and present the WCF Test Client surface:



Double click the CalcPVAnnDue() operation on the WCF Test Client surface to test it. Enter reasonable values for the AnnIntPct, years and AnnPmt fields and click the Invoke button to see the result as it would be provided to a web-client which used this web service:



This result can be quickly checked in the Cielo Explorer (the VisualAPL interactive session):



```
Cielo Explorer for Visual Studio - Professional v1
Cielo 1.0.7001
Copyright © VisualCielo. All rights reserved.
clear session

AnnPmt<-1000
AnnIntPct<-5.5
Years<-10
Dpp<-15
AnnPmt*+/(1+0.01*AnnIntPct)*-1Years
7952.19524916136
```

Now that the WCF web service solution has been completed there are two additional steps to facilitate production use of the project:

- Publishing the WCF service to the Microsoft Internet Information Services (IIS) web server which is described here: <http://msdn.microsoft.com/en-us/library/ms733766.aspx>. This involves:
 - Creating the appropriate virtual paths (directories) in IIS
 - Modifying the web.config file of the WCF project to use IIS
- Creating the appropriate web client. Some possibilities include:
 - Microsoft WPF (Windows Presentation Foundation) and 1-click install technology.
 - Microsoft Silverlight , a lightweight version of WPF exclusively for web-client GUI development
 - PDF form technology
 - HTML form technology (possibly with Javascript support)

For a description of how to create a WCF client see: <http://msdn.microsoft.com/en-us/library/ms733133.aspx>.