VisualAPL for Visual Studio 2008
New Project Templates

A significant difference between the Visual Studio 2005 and Visual Studio 2008 versions of VisualAPL is that the Visual Studio 2005 version of VisualAPL included a project template for creating 'Windows Forms' GUI applications and the Visual Studio 2008 version of VisualAPL no longer provides that project template.

In Visual Studio 2008 Microsoft deprecated 'Windows Forms' GUI methodology and introduced Windows Presentation Foundation (WPF) for GUI development. This replacement of 'Windows Forms' with WPF technology by Microsoft is even more dramatic when one observes that the Visual Studio 2010 IDE is itself constructed using WPF technology, whereas the Visual Studio 2005 and 2008 IDEs were developed using Windows Forms technology.

The VisualAPL development team, recognizing the benefits of WPF over 'Windows Forms' GUI technology and understanding that the strengths of the APL programming language arise from its functional programming style and array-based operations, realized that it was not beneficial to the VisualAPL product to continue to incorporate Microsoft-deprecated technology in VisualAPL. In fact with the plethora of GUI building technologies available today, it no longer makes sense to construct the GUI of an application system using APL.

The VisualAPL development team recommends that the application system design use a 'multi-tier' approach containing separate but interacting .Net components for GUI, business rules and calculations and data. These components are developed as separate Visual Studio projects. These projects are contained in or referenced by a Visual Studio solution. Because all .Net programming languages, e.g. VB.Net, C# and VisualAPL, produce projects which seamlessly interoperate, the .Net programming language best-suited may be independently selected to construct each 'tier' of the application system. Each tier of the application system and each .Net programming language play a supporting role in the solution, rather than developing the entire solution in one programming language.

The options for GUI development include:

- Html and javascript for a browser-based GUI
- Legacy Windows Forms using C#
- WPF Windows using C# or VB.Net
- Microsoft SilverLight (as subset of WPF)
- Legacy []wi (a subset of Windows Form) using VisualAPL
- Non-.Net methodologies such as Adobe Forms

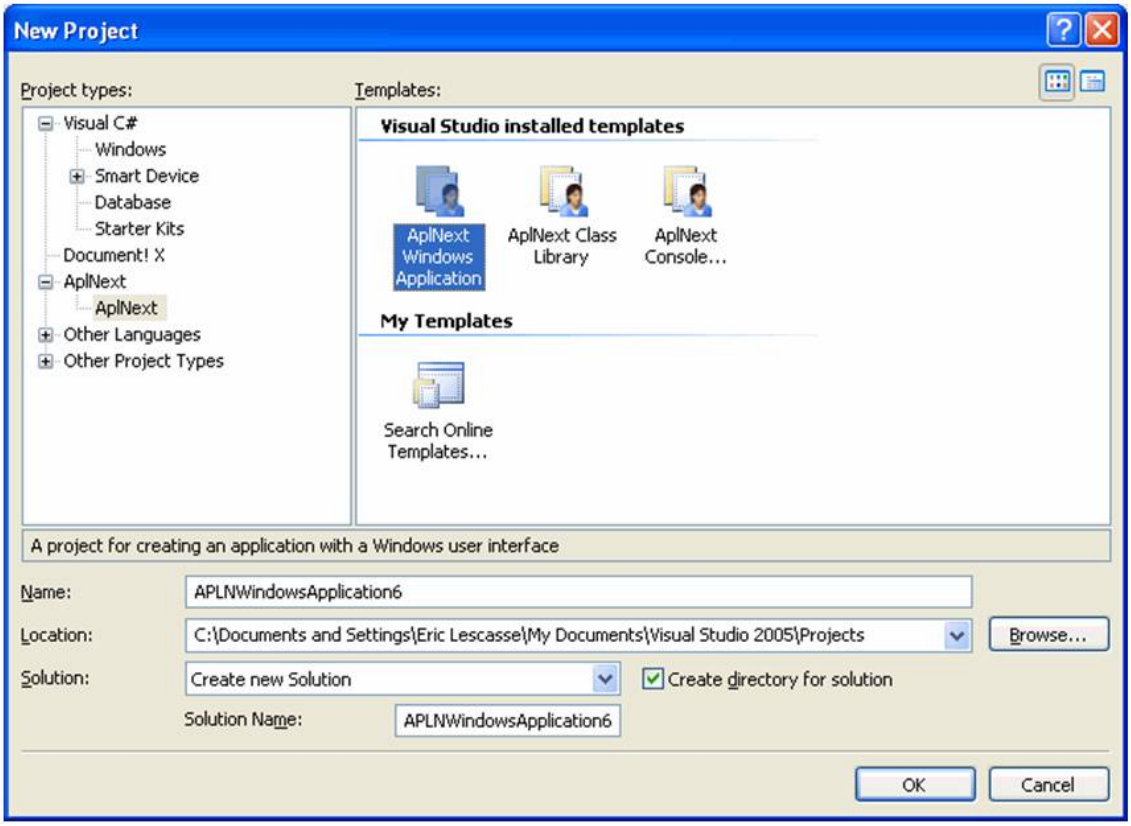
Often VisualAPL is the best programming language choice when implementing the business rules and calculations tier of the application system. The VisualAPL methods, properties and events are contained in one or more VisualAPL Class Llibraries which are then either referenced by the GUI project or are deployed on a web server using Windows Communication Foundation (WCF) technology.

The implementation of the data tier of the application will depend on the target user base for the application system data. Legacy APL component files, native files and scalable, enterprise-level databases, e.g. Microsoft SQL Server, IBMDB2, or Oracle, accessed using ADO.net technology are just a few of the possibilities.

VisualAPL for Visual Studio 2005 installed three template:

- APLNext Class Library (same as the VisualAPL Class Library discussed below)
- APLNext Console Application (same as the VisualAPL Console Application discussed below)
- APLNext Windows Application which supported the'Windows Forms' GUI technology.

Here is a screen capture of the Visual APL for Visual Studio 2005 template options:

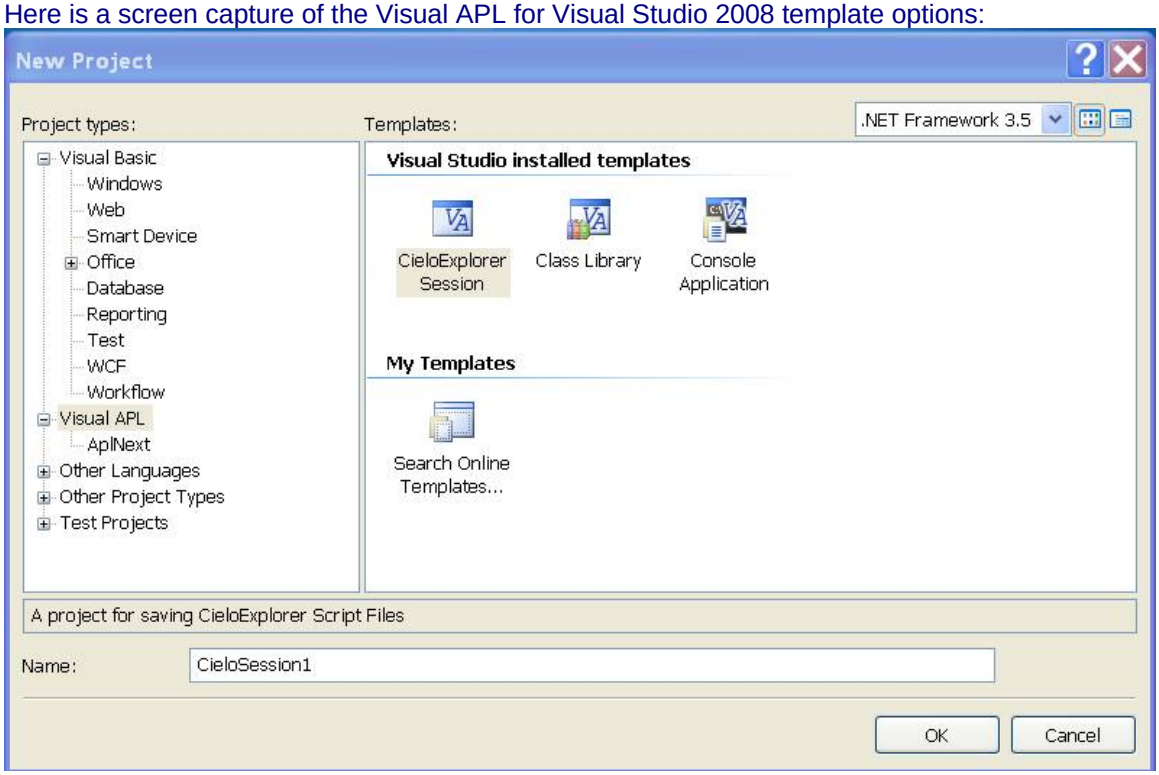


Note that VisualAPL for Visual Studio 2005 is no longer available from APLNext since it has been replaced by VisualAPL for Visual Studio 2008.

VisualAPL for Visual Studio 2008 installed three templates:

- VisualAPL Class Library (discussed below)
- VisualAPL Console Application (discussed below)
- VisualAPL Cielo Explorer Session (discussed below)

Here is a screen capture of the Visual APL for Visual Studio 2008 template options:



VisualAPL Class Library
The fundamental strengths of the APL programming language are its functional programming style and array-based operations. These features are often used to effectively support the 'business rules' and calculations of the application system. The VisualAPL Class Library is the fundamental container for the VisualAPL functions which comprise the mathematical and logical portions of the application system. A VisualAPL Class Library is a fully-managed .Net assembly which seamlessly interoperates with any .Net project using any .Net programming language. The file extension of a VisualAPL Class Library is .dll.

After the application system developer creates a VisualAPL Class Library it can be 'referenced' by any other .Net project in a Visual Studio solution. The VisualAPL Class Library has a namespace and class identity. Within the class the programmer may define methods (analogous to APL functions), properties and events. Those methods, properties and events which are designated by the programmer as public are available to the referencing project in order to interact in an object oriented manner with the VisualAPL Class Library.

Deploying a VisualAPL Class Library on a web server is also a viable possibility using the Windows Communication Foundation (WCF) technology to expose the methods, properties and events of the VisualAPL Class Library to clients via the Internet.

VisualAPL Console Application
The console application is a traditional Visual Studio method to create a simple 'test harness' to investigate .Net features. A console application runs within the 'Command Prompt' window of the operating system. The file extension of a console application is .exe.

Typically a console project will request test input using the Console.ReadLine() method and output information to the Command Prompt window using the Console.WriteLine() method. A console project is a compiled executable, so that the entire source code of the console project must be entered before any results are available.

VisualAPL Cielo Explorer Session
The VisualAPL Cielo Explorer Session is a superior alternative to the console application for investigating the .Net universe. The Cielo Explorer Session supports the entry of individual lines of C# or VisualAPL code which is immediately executed. Thus the Cielo Explorer is the analog of the traditional APL 'session'.

In addition to the line-by-line execution of programmer-entered code, the Cielo Explorer also supports 'scripting'. Scripts are Unicode text files which incorporate one or more lines of C# or VisualAPL source code. A script file can be saved, opened, edited and run from within the Cielo Explorer Session. A script file can also be incorporated into a VisualAPL or C# project and dynamically-loaded and executed.