Building .Net Applications – Using Interfaces

When building a .Net application, most programmers have elected to separate the GUI (graphical user interface), business rules (calculations and algorithms) and data into separate components. This permits independent development of these components using appropriate, possibly-different programming languages, programmers and time-lines. ***Interfaces*** in .Net are also designed and created so that each of application system's components can interact properly.

.Net interfaces are classes, or structures, which define themselves using the interface keyword. Interfaces may incorporate.Net methods, properties, events and indexers which are automatically public members of the interface. An interface does not implement the actions associated with its members, but instead provides a definition of the formal structure of these members. Thus an interface is an abstraction of the methods, properties, events and indexers of the data and data access methods associated with the application system.

Other .Net classes, or structures, may inherit an interface. If a class inherits an interface, all members of the inherited interface must be implemented by the inheriting class, or structure.

The interfaces for an application system are generally developed by the programmer immediately so that they are available to all other application system components as they are developed. In this way each application system component can be assured that it has a full understanding of the formal structure of any interface available to all components of the application system. Using an interface also permits the physical implementation of the methods, properties, events and indexers of an interface, via an inheriting class, to be performed on its own time-line and by a separate programmer, if necessary.

Of course, any .Net language, such as C#, VisualAPL or VB.Net can create, use and share interfaces.

Using interfaces make good sense when designing a .Net application system. In addition an interface is essential in certain circumstances, for example when exposing a .Net assembly as COM/ActiveX.

To learn more about interfaces, go to:

- http://msdn.microsoft.com/en-us/library/ms173156.aspx
- http://msdn.microsoft.com/en-us/library/ms173157.aspx
- http://msdn.microsoft.com/en-us/library/44a9ty12.aspx
- http://msdn.microsoft.com/en-us/library/4taxa8t2.aspx

To see examples of a .Net interfaces in action, go to:

- http://www.c-sharpcorner.com/UploadFile/rmcochran/csharp_interrfaces03052006095933AM/csharp_interrfaces.aspx?ArticleID=cd6a6952-530a-4250-a6d7-54717ef3b345
- http://www.codeproject.com/KB/cs/cs_interfaces.aspx

An excellent reference book, with significant material on .Net interfaces in "Part 3 Advanced C# Programming Constructs: Chapter 9 Working with Interfaces", is:

- http://www.apress.com/book/view/1590598849