

This Free E-Book is brought to you by Natural-Aging.com.

**[100% Effective Natural Hormone Treatment](#)
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!**

Understanding Common Type System in .Net Framework

By Balaji

Understanding Common Type System in .Net Framework by Balaji

Understanding Common Type System in .Net Framework

As .Net Framework is language independent and support over 20 different programming languages, many programmers will write data types in their own programming language.

For example, an integer variable in C# is written as int, whereas in Visual Basic it is written as integer. Therefore in .Net Framework you have single class called System.Int32 to interpret these variables. Similarly, for the ArrayList data type .Net Framework has a common type called System.Collections.ArrayList. In .Net Framework, System.Object is the common base type from where all the other types are derived.

This system is called Common Type System. The types in .NET Framework are the base on which .NET applications, components, and controls are built. Common Type System in .Net Framework defines how data types are going to be declared and managed in runtime. The Common Type System performs the following functions:

- Automatically adapts itself in a framework that enables integration of multiple languages, type safety, and high performance code execution.
- Provides an object-oriented model.
- Standardizes the conventions that all the languages must follow.
- Invokes security checks.
- Encapsulates data structures.

There are two general types of categories in .Net Framework that Common Type System support. They are value types and reference types. Value types contain data and are user-defined or built-in. they are placed in a stack or in order in a structure. Reference types store a reference of the value's memory address. They are allocated in a heap structure. You can determine the type of a reference by the values of self-describing types. Reference types can be categorized into self-describing types, pointer types, or interface types.

Understanding Common Type System in .Net Framework

There are many other types that can be defined under Value types and Reference types. In .Net Framework, the System namespace is the root for all the data types. This namespace consists of classes such as Object, Byte, String, and Int32 that represents base data types. These base data types are used by all applications. During runtime a type name can be classified into two: the assembly name and the type's name within the assembly. The runtime in .Net Framework uses assemblies to find and load types.

To access online version of the above article, go to <http://www.dotnet-guide.com/commontype.html>

Visit <http://www.dotnet-guide.com> for a complete introduction to .NET framework. Learn about

ASP.NET, VB.NET, C# and other related technologies.

Understanding Code Behind in .Net Framework

By Balaji

Understanding Code Behind in .Net Framework by Balaji

Understanding Code Behind in .Net Framework

The ASP.NET Code Behind feature in .Net Framework allows developers to separate the server-side code from the presentation layer. This concept makes the server-side code to store in one file and the presentation code, that is, HTML code in another file. When you compile the ASP.NET page both these files get compiled as a single entity. In the traditional ASP model, this could not be achieved which often leads to intermingling of the code and the design.

The biggest advantage, in ASP.NET, is that the presentation code will be in .aspx file and the server-side code will be in any .Net compatible language such as Visual Basic.Net, C#, or J#. You can also do away with the presentation layer because you can give this role to the web designers. This saves time and you can concentrate only on the coding part of the application. In addition, you can create a class for your code and inherit this class from the ASP.NET Page object. By this way the class can access the page intrinsic and also interact with the postback architecture. After this you can create the ASP.NET page and apply a page directive to inherit from this new class.

But before you create an ASP.NET Code Behind class, you have to reference it to a namespace. The namespace could be System.Web.UI or System.Web.UI.WebControls. Next you have to inherit the class from the Page object. You must declare some public instances of server controls using the name for the variables that are similar to the web controls. This procedure will create a link between the ASP.NET Code Behind class and the server controls.

You can use the ASP.NET Code Behind feature in various web applications development tools such as Visual Studio.Net and ASP.NET Web Matrix. They provide very easy ways to use the ASP.NET Code Behind. After dragging and dropping the server control from the Toolbox to the web page you can just

right click on it to view the ASP.NET Code Behind page.

To access online version of the above article, go to <http://www.dotnet-guide.com/codebehind.html>

Visit <http://www.dotnet-guide.com> for a complete introduction to .NET framework. Learn about ASP.NET, VB.NET, C# and other related technologies.



This Free E-Book has been brought to you by Natural-Aging.com.

[100% Effective Natural Hormone Treatment](#)
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!