

Tutorial ASP.NET: Estructura de clases y objetos

ASP.NET

Muchas veces necesitamos reutilizar código en nuestros proyectos, utilizar la misma funcionalidad en varias partes diferentes de nuestro proyecto. La solución a este problema es introducir código en la carpeta `App_Code` donde podemos crear clases con métodos que pueden utilizarse desde todo el proyecto y se compila en tiempo de ejecución.



Fernando Giardina

enero 27 2011

La carpeta `App_Code` puede contener todos los archivos y subcarpetas que necesitemos. Se puede organizar el código fuente de la forma que creamos más conveniente, ya que ASP.NET lo compilará en un solo ensamblado al que podrá tener acceso el código de cualquier parte de la aplicación Web.

Código compartido

En los ejemplos anteriores tuvimos la necesidad de contar con código compartido. Por ejemplo, el método `IsNumeric` lo utilizamos en el [Capítulo 7: Controles de usuarios reutilizables](#) y en el [Capítulo 11: Acceso a datos](#) (consultar y guardar información desde WebForms).

- **Estructura de una clase:** Una clase está compuesta por Atributos, Propiedades, Métodos y Eventos. Aunque no necesariamente deba tener todos sus componentes declarados. Podemos crear clases que sólo tengan métodos o que sólo tengan atributos y propiedades.
- **Ejemplo de una clase básica:** El primer evento de nuestra clase va a ser el constructor,

al instanciar la clase (Objeto) lo primero que se ejecuta es el constructor.

```
1 | public class Herramientas
2 | {
3 |     public Herramientas()
4 |     {
5 |     }
6 | }
```

?

Ahora agregaremos un método que nos facilite la reutilización de código, como por ejemplo: `IsNumeric()`.

```
01 | using System;
02 | using System.Collections.Generic;
03 | using System.Web;
04 | public class Herramientas
05 | {
06 |     public Herramientas()
07 |     {
08 |     }
09 |     public bool IsNumeric(object Expression)
10 |     {
11 |         bool isNum;
12 |         double retNum;
13 |         isNum = Double.TryParse(Convert.ToString(Expression), S
14 |         return isNum;
15 |     }
16 | }
```

?

Ya tenemos creada nuestra clase `Herramientas.cs` en la carpeta `App_Code`. ¿Cómo podemos utilizar el método `IsNumeric()` desde cualquier parte de nuestra solución Web?

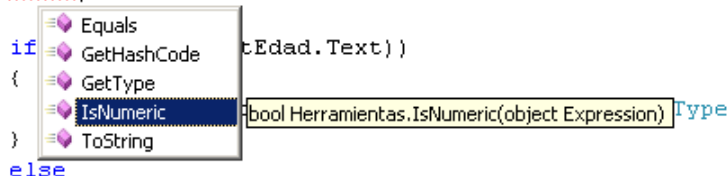
Lo primero que debemos hacer es instanciarla.

```
1 | Herramientas herr = new Herramientas();
```

?

Ahora podemos ver dentro de nuestro objeto `herr` los métodos públicos.

```
Herramientas herr = new Herramientas();
herr.
if {
    GetType
    IsNumeric
    ToString
else
```



Este método nos devuelve un Booleano: verdadero o falso y nos sirve para evaluar. Vamos a llamar a la clase Herramientas desde el método **Escribir** del capítulo anterior.

```

1 | Herramientas herr = new Herramientas();
2 | if (herr.IsNumeric(txtEdad.Text))
3 | {
4 | }
```

De esta forma nos evitamos tener que codificar el evento **IsNumeric** en cada parte del código donde lo necesitemos.

Clases estáticas:

Las clases estáticas son clases que no tienen un constructor invocable, por consiguiente no hay que instanciarlas y se pueden acceder a los métodos en forma directa.

Veamos el ejemplo anterior pero declarando la clase como estática y su utilización.

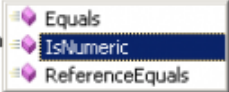
```

1 | public static class Herramientas
2 | {
3 |     static Herramientas()
4 |     {
5 |     }
6 |     public static bool IsNumeric(object Expression)
7 |     {
8 |     }
9 | }
```

Al no tener la necesidad de instanciarla la llamada al método **IsNumeric** lo hacemos en forma directa de la siguiente forma.

```

if (Herramientas.IsNumeric(txtEdad.Text))
{
    insertComm = IsNumeric(txtEdad.Text) ? OleDbType.Integer : OleDbType.VarChar;
}
else
{
    throw new Exception("La edad debe ser numerica");
}
```



Capítulos del Tutorial

- [Capítulo 1: Tutorial de desarrollo web con ASP.NET](#)
- [Capítulo 2: Primera aplicación web en ASP.NET](#)

- [Capítulo 3: Ejecutar código JavaScript en ASPNET](#)
- [Capítulo 4: Archivos Web.Config y Global.asax](#)
- [Capítulo 5: Utilización de Código detrás del modelo o código en línea](#)
- [Capítulo 6: Controles de servidor y eventos](#)
- [Capítulo 7: Controles de usuarios reutilizables](#)
- [Capítulo 8: Crear una página de login, autenticación y seguridad.](#)
- [Capítulo 9: Utilización de estilos en ASP.NET \(CSS\)](#)
- [Capítulo 10: Trabajando con XML y Web Services](#)
- [Capítulo 11: Acceso a datos, consultar y guardar información desde WebForms](#)
- [Capítulo 12: Estructura de clases y objetos](#)



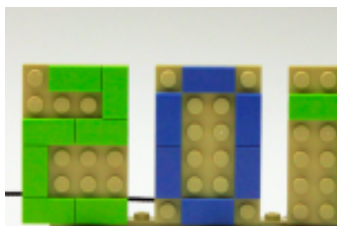
Fernando
Giardina

Fernando Giardina

En 1997 comienzo a desarrollar en lenguaje C y Oracle programando IVRs, automatización de procesos y sistemas de CRM. Continuo con JAVA, Perl y PHP hasta que en el año 2002 me inicio en la tecnología .NET desarrollando aplicaciones web, desktop y mobile.

<http://www.metalstudio.com.ar>

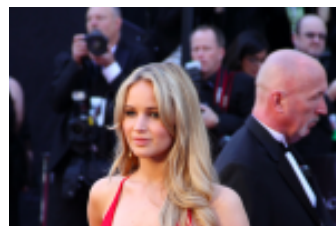
Otros artículos que podrían interesarte



[10 Posts inspiradores de Maestros para guardar en PDF](#)



["Search Experience Optimization" SEO para humanos, no robots](#)



[Fotos de famosos desnudos: un modelo de negocio emergente en Internet](#)



[Cómo iniciar una startup: conoce los secretos de los más exitosos](#)



Añade un comentario...

☐ Publicar también en Facebook

Publicar como **Javier J Solis Flores** ▼

[Comentar](#)

Plug-in social de Facebook