



**EBook Gratis**

# APRENDIZAJE ASP.NET

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

**#asp.net**

# Tabla de contenido

Acerca de.....	1
<b>Capítulo 1: Empezando con ASP.NET.....</b>	<b>2</b>
Observaciones.....	2
Examples.....	2
Instalación o configuración.....	2
Descripción general de ASP.NET.....	2
Hola mundo con OWIN.....	3
Introducción simple de ASP.NET.....	4
<b>Capítulo 2: Asp Web Forms Identity.....</b>	<b>5</b>
Examples.....	5
Empezando.....	5
<b>Capítulo 3: ASP.NET - Controles básicos.....</b>	<b>7</b>
Sintaxis.....	7
Examples.....	7
Cuadros de texto y etiquetas.....	7
Casillas de verificación y botones de radio.....	8
Controles de lista.....	9
Lista de botones de radio y lista de casillas de verificación.....	10
Listas con viñetas y listas numeradas.....	11
Control HyperLink.....	11
Control de imagen.....	12
<b>Capítulo 4: ASP.NET - Controles de usuario.....</b>	<b>13</b>
Introducción.....	13
Examples.....	13
Introducción de controles de usuario.....	13
Creación de instancia de control de usuario mediante programación.....	14
Agregar propiedades personalizadas para el control de usuario.....	15
<b>Capítulo 5: ASP.NET - Estado de gestión.....</b>	<b>16</b>
Examples.....	16
Ver estado.....	16

<b>Capítulo 6: ASP.NET - Validadores</b>	<b>18</b>
Sintaxis	18
Examples	18
Controles de validación	18
RequiredFieldValidator Control	18
Control RangeValidator	19
CompareValidator Control	19
RegularExpressionValidator	20
Resumen de validación	21
Grupos de validación	22
<b>Capítulo 7: Asp.net Ajax Controls</b>	<b>25</b>
Examples	25
FileUpload Ajax Toolkit Control	25
<b>Capítulo 8: ASP.NET Caching</b>	<b>27</b>
Examples	27
Caché de datos	27
<b>Capítulo 9: Ciclo de vida de la página</b>	<b>29</b>
Examples	29
Eventos del Ciclo de Vida	29
Ejemplo de código	30
<b>Capítulo 10: DayPilot Scheduler</b>	<b>34</b>
Parámetros	34
Observaciones	34
Examples	34
Información básica	34
Declaración	34
<b>Capítulo 11: Delegación de eventos</b>	<b>36</b>
Sintaxis	36
Observaciones	36
Examples	36
Delegación de eventos del control de usuario a aspx	36

<b>Capítulo 12: Directivas</b>	<b>39</b>
Examples	39
La directiva de aplicación	39
La directiva de control	39
La Directiva de Implementos	40
La Directiva Maestra	40
La directiva de importación	40
La Directiva MasterType	40
La directiva de la página	41
La directiva de outputcache	42
<b>Capítulo 13: El enlace de datos</b>	<b>43</b>
Examples	43
Fuente de datos SQL	43
Recuperando datos	43
Uso básico	44
Fuente de datos del objeto	44
<b>Capítulo 14: Encuentra Control por ID</b>	<b>46</b>
Sintaxis	46
Observaciones	46
Examples	46
Accediendo al Control TextBox en la página aspx	46
Encuentra un control en un GridView, Repeater, ListView, etc.	46
<b>Capítulo 15: Estado de sesión</b>	<b>47</b>
Sintaxis	47
Observaciones	47
Examples	47
Usando el objeto Session para almacenar valores	47
Usando un almacén de sesión SQL	48
Usando un almacén de sesión de Amazon DynamoDB	48
<b>Capítulo 16: Expresiones</b>	<b>50</b>
Examples	50
Valor desde App.Config	50

Expresion Evaluada.....	50
Bloque de código dentro del marcado ASP.....	50
<b>Capítulo 17: Formas Web.....</b>	<b>51</b>
Sintaxis.....	51
Observaciones.....	51
Examples.....	51
Usando un repetidor para crear una tabla HTML.....	51
Agrupación en ListView.....	52
Ejemplo.....	54
Hiperenlace.....	54
<b>Capítulo 18: Gestion de Sesiones.....</b>	<b>56</b>
Examples.....	56
Ventaja y desventaja del estado de sesión, tipos de sesión.....	56
<b>Capítulo 19: httpHandlers.....</b>	<b>57</b>
Examples.....	57
Uso de un httpHandler (.ashx) para descargar un archivo desde una ubicación específica.....	57
<b>Capítulo 20: Katana.....</b>	<b>59</b>
Introducción.....	59
Examples.....	59
Ejemplo.....	59
<b>Capítulo 21: Lista de datos.....</b>	<b>61</b>
Sintaxis.....	61
Examples.....	61
Enlace de datos en asp.net.....	61
<b>Capítulo 22: Manejo de eventos.....</b>	<b>63</b>
Sintaxis.....	63
Parámetros.....	63
Examples.....	63
Eventos de aplicación y sesión.....	63
Eventos de página y control.....	63
Eventos predeterminados.....	64

<b>Capítulo 23: Métodos de página</b>	<b>67</b>
Parámetros	67
Observaciones	67
<b>Más de un parámetro</b>	<b>67</b>
<b>Valor de retorno</b>	<b>67</b>
Examples	67
Como llamarlo	67
<b>Capítulo 24: Middleware</b>	<b>69</b>
Parámetros	69
Observaciones	69
Examples	69
Muestra la ruta de la solicitud y el tiempo que llevó procesarla	69
<b>Capítulo 25: Reloj de repetición</b>	<b>71</b>
Examples	71
Uso básico	71
<b>Capítulo 26: ScriptManager</b>	<b>72</b>
Introducción	72
Sintaxis	72
Examples	72
Trabajando con ScriptManager	72
<b>Capítulo 27: Servicio Web sin Visual Studio</b>	<b>74</b>
Introducción	74
Observaciones	74
Examples	74
Calculadora de servicios web	74
<b>Capítulo 28: UpdatePanel</b>	<b>75</b>
Introducción	75
Sintaxis	75
Observaciones	75
Examples	75
Ejemplo de panel de actualización	75

<b>Capítulo 29: Ver estado</b>	<b>77</b>
Introducción	77
Sintaxis	77
Examples	77
Ejemplo	77
<b>Capítulo 30: Vista en cuadrícula</b>	<b>79</b>
Examples	79
El enlace de datos	79
Encuadernación manual	79
DataSourceControl	79
Columnas	79
GridView fuertemente tipado	80
Manejo de evento de comando	81
Paginacion	82
ObjectDataSource	83
Encuadernación manual	83
Actualizar Gridview en la fila de clic	84
<b>Capítulo 31: web.config&gt; system.webServer / httpErrors &amp; system.web / customErrors en las</b>	<b>87</b>
Introducción	87
Examples	87
¿Cuál es la diferencia entre customErrors y httpErrors?	87
<b>Creditos</b>	<b>88</b>

---

## Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [asp-net](#)

It is an unofficial and free ASP.NET ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ASP.NET.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)



---

# Capítulo 1: Empezando con ASP.NET

## Observaciones

ASP.NET es una colección de tecnologías dentro de .NET Framework que están orientadas al desarrollo de aplicaciones web. Estas tecnologías consisten en:

- WebForms: una plataforma de desarrollo de estilo RAD que utiliza controles web.
- MVC: Una plataforma de desarrollo de Model View Controller.
- SignalR: una plataforma de mensajería en tiempo real para mensajería cliente / servidor.
- Razor: un lenguaje de marcado de front-end con el que puede incrustar comandos del lado del servidor.
- WebAPI: una plataforma para crear aplicaciones de estilo REST API.

## Examples

### Instalación o configuración

De forma predeterminada, todas las bibliotecas necesarias para compilar aplicaciones ASP.NET se incluyen durante la instalación de Visual Studio. Si se lanza una versión más reciente de ASP.NET que no se incluyó con Visual Studio, puede descargar la biblioteca SDK correspondiente de Microsoft, que incluirá todas las bibliotecas necesarias para esa versión.

De manera similar, el sistema operativo Windows viene preinstalado con una versión más reciente de ASP.NET y se registra automáticamente con IIS para la configuración y ejecución. De manera similar, si una versión más nueva de ASP.NET se vuelve disponible, puede instalar el SDK para la versión que necesita y luego usar la herramienta `aspnet_regiis` para registrar el marco con IIS para su uso.

También se debe tener en cuenta que para implementaciones de servidor, también existe un paquete redistribuible de SDK de ASP.NET. Esta versión es una versión optimizada del SDK, con solo las bibliotecas esenciales y no tiene las herramientas e integraciones con Visual Studio.

### Descripción general de ASP.NET

ASP.NET es un modelo de desarrollo web unificado que incluye los servicios necesarios para que pueda crear aplicaciones web de clase empresarial con un mínimo de codificación. ASP.NET es parte de .NET Framework, y al codificar aplicaciones ASP.NET tiene acceso a las clases en .NET Framework.

Puede codificar sus aplicaciones en cualquier idioma compatible con Common Language Runtime (CLR), incluidos Microsoft Visual Basic, C #, JScript .NET y J #. Estos idiomas le permiten desarrollar aplicaciones ASP.NET que se benefician del tiempo de ejecución del idioma común, la seguridad de tipos, la herencia, etc.

ASP.NET incluye:

- Un marco de página y controles.
- El compilador de ASP.NET
- Infraestructura de seguridad
- Instalaciones de gestión estatal.
- Configuración de la aplicación
- Monitorización de la salud y características de rendimiento.
- Soporte de depuración
- Un marco de servicios web XML
- Entorno de hosting extensible y gestión del ciclo de vida de la aplicación
- Un entorno de diseño extensible.

## Hola mundo con OWIN

Utilice el administrador de paquetes para instalar Microsoft.Owin.SelfHost

```
install-package Microsoft.Owin.SelfHost
```

Código para una aplicación web HelloWorld mínima que se ejecuta desde una ventana de consola:

```
namespace HelloOwin
{
    using System;
    using Owin;

    class Program
    {
        static readonly string baseUrl = "http://localhost:8080";

        static void Main(string[] args)
        {
            using (Microsoft.Owin.Hosting.WebApp.Start<Startup>(baseUrl))
            {
                Console.WriteLine("Prease any key to quit.");
                Console.ReadKey();
            }
        }
    }

    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.Run(ctx =>
            {
                return ctx.Response.WriteAsync("Hello World");
            });
        }
    }
}
```

## Introducción simple de ASP.NET

Asp.net es un marco de aplicación web desarrollado por Microsoft para crear aplicaciones web dinámicas basadas en datos y servicios web.

Asp.net es básicamente un subconjunto de un marco .NET más amplio. Un marco no es más que una colección de clases.

En .NET Framework puedes construir una aplicación de consola. Aplicación web, aplicación de ventana, aplicación móvil. Así que para la aplicación web se está utilizando ASP.net.

ASP.NET es el sucesor de ASP clásico (página Active Server).

### ¿Qué es la aplicación web?

Una aplicación web es una aplicación a la que los usuarios acceden mediante un navegador web como:

- Microsoft Internet Explorer.
- Google Chrome
- Mozilla Firefox
- Safari de manzana

Lea Empezando con ASP.NET en línea: <https://riptutorial.com/es/asp-net/topic/836/empezando-con-asp-net>

---

# Capítulo 2: Asp Web Forms Identity

## Examples

### Empezando

#### Empezando

#### Instale los paquetes de NuGet:

1. **Microsoft.AspNet.Identity.EntityFramework**
2. **Microsoft.AspNet.Identity.Core**
3. **Microsoft.AspNet.Identity.OWIN**

#### Registrar acción - Controlador de cuenta

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            AddErrors(result);
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

#### Acción de inicio de sesión - Método SignInAsync

```
private async Task SignInAsync(ApplicationUser user, bool isPersistent)
{
    AuthenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie);

    var identity = await UserManager.CreateIdentityAsync(
        user, DefaultAuthenticationTypes.ApplicationCookie);

    AuthenticationManager.SignIn(
        new AuthenticationProperties() {
```

```
        IsPersistent = isPersistent
    }, identity);
}
```

## Desconectarse

```
// POST: /Account/LogOff
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult LogOff()
{
    AuthenticationManager.SignOut();
    return RedirectToAction("Index", "Home");
}
```

Lea Asp Web Forms Identity en línea: <https://riptutorial.com/es/asp-net/topic/9146/asp-web-forms-identity>

# Capítulo 3: ASP.NET - Controles básicos

## Sintaxis

- `<asp: Button ID = "Button1" runat = "server" onclick = "Button1_Click" Text = "Click" /> <asp: TextBox ID = "txtstate" runat = "server">`
- `</ asp: TextBox> <asp: CheckBox ID = "chkoption" runat = "Server"> </ asp: CheckBox>`
- `<asp: RadioButton ID = "rdboption" runat = "Server"> </ asp: RadioButton>`
- `<asp: ListBox ID = "ListBox1" runat = "server" AutoPostBack = "True" OnSelectedIndexChanged = "ListBox1_SelectedIndexChanged"> </ asp: ListBox>`
- `<asp: DropDownList ID = "DropDownList1" runat = "server" AutoPostBack = "True" OnSelectedIndexChanged = "DropDownList1_SelectedIndexChanged"> </ asp: DropDownList>`
- `<asp: RadioButtonList ID = "RadioButtonList1" runat = "server" AutoPostBack = "True" OnSelectedIndexChanged = "RadioButtonList1_SelectedIndexChanged"> </ asp: RadioButtonList>`
- `<asp: CheckBoxList ID = "CheckBoxList1" runat = "server" AutoPostBack = "True" OnSelectedIndexChanged = "CheckBoxList1_SelectedIndexChanged"> </ asp: CheckBoxList>`
- `<asp: BulletedList ID = "BulletedList1" runat = "server"> </ asp: BulletedList>`
- `<asp: HyperLink ID = "HyperLink1" runat = "server"> HyperLink </ asp: HyperLink> <asp: Image ID = "Image1" runat = "server">`

## Examples

### Cuadros de texto y etiquetas

Los controles de cuadro de texto se utilizan normalmente para aceptar entradas del usuario. Un control de cuadro de texto puede aceptar una o más líneas de texto dependiendo de la configuración del atributo `TextMode`.

Los controles de etiqueta proporcionan una manera fácil de mostrar texto que se puede cambiar de una ejecución de una página a la siguiente. Si desea mostrar texto que no cambia, use el texto literal.

Sintaxis básica del control de texto:

```
<asp:TextBox ID="txtstate" runat="server" ></asp:TextBox>
```

Propiedades comunes del cuadro de texto y etiquetas:

Propiedades	Descripción
Modo de texto	Especifica el tipo de cuadro de texto. <code>SingleLine</code> crea un cuadro de texto estándar, <code>MultiLine</code> crea un cuadro de texto que acepta más de una línea de

Propiedades	Descripción
	texto y la Contraseña hace que los caracteres que se ingresan estén enmascarados. El valor predeterminado es SingleLine.
Texto	El contenido de texto del cuadro de texto.
Longitud máxima	El número máximo de caracteres que se pueden introducir en el cuadro de texto.
Envolver	Determina si el texto se ajusta automáticamente o no para el cuadro de texto de varias líneas; el valor predeterminado es true.
Solo lectura	Determina si el usuario puede cambiar el texto en el cuadro; el valor predeterminado es falso, es decir, el usuario puede cambiar el texto.
Columnas	El ancho del cuadro de texto en caracteres. El ancho real se determina en función de la fuente que se utiliza para la entrada de texto.
Filas	La altura de un cuadro de texto multilínea en líneas. El valor predeterminado es 0, significa un cuadro de texto de una sola línea.

El atributo más utilizado para un control de etiqueta es 'Texto', que implica el texto que se muestra en la etiqueta.

## Casillas de verificación y botones de radio

Una casilla de verificación muestra una única opción que el usuario puede marcar o desmarcar y los botones de radio presentan un grupo de opciones entre las cuales el usuario puede seleccionar solo una opción.

Para crear un grupo de botones de opción, debe especificar el mismo nombre para el atributo GroupName de cada botón de opción en el grupo. Si se requiere más de un grupo en un solo formulario, especifique un nombre de grupo diferente para cada grupo.

Si desea que se seleccione la casilla de verificación o el botón de opción cuando se muestre inicialmente el formulario, establezca su atributo Verificado en verdadero. Si el atributo Comprobado se establece en verdadero para varios botones de radio en un grupo, solo el último se considera verdadero.

Sintaxis básica de la casilla de verificación:

```
<asp:CheckBox ID= "chkoption" runat= "Server"> </asp:CheckBox>
```

Sintaxis básica del botón de radio:

```
<asp:RadioButton ID= "rdboption" runat= "Server"> </asp: RadioButton>
```

Propiedades comunes de casillas de verificación y botones de radio:

Propiedades	Descripción
Texto	El texto que se muestra al lado de la casilla de verificación o botón de radio.
Comprobado	Especifica si está seleccionado o no, el valor predeterminado es falso.
Nombre del grupo	Nombre del grupo al que pertenece el control.

## Controles de lista

ASP.NET proporciona los siguientes controles

- La lista desplegable
- Cuadro de lista
- Lista de botones de radio
- Lista de casillas de verificación
- Lista con viñetas

Estos controles permiten al usuario elegir entre uno o más elementos de la lista. Los cuadros de lista y las listas desplegables contienen uno o más elementos de lista. Estas listas se pueden cargar por código o por el editor ListItemCollection.

Sintaxis básica del control de cuadro de lista:

```
<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged">
</asp:ListBox>
```

Sintaxis básica del control de lista desplegable:

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
</asp:DropDownList>
```

Propiedades comunes del cuadro de lista y listas desplegables:

Propiedades	Descripción
Artículos	La colección de objetos ListItem que representa los elementos en el control. Esta propiedad devuelve un objeto de tipo ListItemCollection.
Filas	Especifica el número de elementos que se muestran en el cuadro. Si la lista real contiene más filas de las que se muestran, se agrega una barra de desplazamiento.
SelectedIndex	El índice del elemento seleccionado actualmente. Si se selecciona más de un elemento, entonces el índice del primer elemento seleccionado. Si no



Propiedades	Descripción
	se selecciona ningún elemento, el valor de esta propiedad es -1.
SelectedValue	El valor del elemento seleccionado actualmente. Si se selecciona más de un elemento, entonces el valor del primer elemento seleccionado. Si no se selecciona ningún elemento, el valor de esta propiedad es una cadena vacía ("").
Modo de selección	Indica si un cuadro de lista permite selecciones individuales o selecciones múltiples.

Propiedades comunes de cada objeto de la lista de objetos:

Propiedades	Descripción
Texto	El texto mostrado para el artículo.
Seleccionado	Un valor de cadena asociado con el elemento.
Valor	Indica si el elemento está seleccionado.

Es importante señalar que:

- Para trabajar con los elementos en una lista desplegable o en un cuadro de lista, utilice la propiedad Items del control. Esta propiedad devuelve un objeto ListItemCollection que contiene todos los elementos de la lista.
- El evento SelectedIndexChanged se genera cuando el usuario selecciona un elemento diferente de una lista desplegable o cuadro de lista.

## Listado de botones de radio y lista de casillas de verificación

Una lista de botones de radio presenta una lista de opciones mutuamente excluyentes. Una lista de casillas de verificación presenta una lista de opciones independientes. Estos controles contienen una colección de objetos ListItem a los que se podría hacer referencia a través de la propiedad Items del control.

Sintaxis básica de la lista de botones de radio:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
    OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">
</asp:RadioButtonList>
```

Sintaxis básica de la lista de casillas de verificación:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack="True"
    OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged">
</asp:CheckBoxList>
```

Propiedades comunes de las listas de casillas de verificación y botones de radio:

Propiedades	Descripción
RepeatLayout	Este atributo especifica si las etiquetas de la tabla o el flujo de HTML normal se usarán al formatear la lista cuando se procesa. El valor predeterminado es Tabla.
Repetir direccion	Especifica la dirección en la que se repetirán los controles. Los valores disponibles son horizontal y vertical. El valor predeterminado es vertical.
RepetirColumnas	Especifica el número de columnas que se utilizarán al repetir los controles; el valor predeterminado es 0.

## Listas con viñetas y listas numeradas

El control de lista con viñetas crea listas con viñetas o listas numeradas. Estos controles contienen una colección de objetos ListItem a los que se podría hacer referencia a través de la propiedad Items del control.

Sintaxis básica de una lista con viñetas:

```
<asp:BulletedList ID="BulletedList1" runat="server">
</asp:BulletedList>
```

Propiedades comunes de la lista con viñetas:

Propiedades	Descripción
BulletStyle	Esta propiedad especifica el estilo y el aspecto de las viñetas o números.
Repetir direccion	Especifica la dirección en la que se repetirán los controles. Los valores disponibles son horizontal y vertical. El valor predeterminado es vertical.
RepetirColumnas	Especifica el número de columnas que se utilizarán al repetir los controles; el valor predeterminado es 0.

## Control HyperLink

El control HyperLink es como el elemento HTML.

Sintaxis básica para un control de hipervínculo:

```
<asp:HyperLink ID="HyperLink1" runat="server">
    HyperLink
</asp:HyperLink>
```

Tiene las siguientes propiedades importantes:

Propiedades	Descripción
URL de la imagen	Ruta de acceso de la imagen a ser mostrada por el control.
NavegarUrl	URL del enlace de destino.
Texto	El texto que se mostrará como el enlace.
Objetivo	La ventana o marco que carga la página enlazada.

## Control de imagen

El control de imagen se utiliza para mostrar imágenes en la página web, o algún texto alternativo, si la imagen no está disponible.

Sintaxis básica para un control de imagen:

```
<asp:Image ID="Image1" runat="server">
```

Tiene las siguientes propiedades importantes:

Propiedades	Descripción
Texto alternativo	Texto alternativo para ser mostrado en ausencia de la imagen.
ImageAlign	Opciones de alineación para el control.
URL de la imagen	Ruta de acceso de la imagen a ser mostrada por el control.

Lea ASP.NET - Controles básicos en línea: <https://riptutorial.com/es/asp-net/topic/6444/asp-net---controles-basicos>

# Capítulo 4: ASP.NET - Controles de usuario

## Introducción

Los controles de usuario son contenedores que pueden rellenarse con el marcado HTML y los controles de servidor con código subyacente de la misma manera que la página ASPX. Están tratadas como unidades más pequeñas reutilizables de una página, por lo que no se puede ejecutar como páginas independientes y no deben tener elementos **HTML**, o la **forma del cuerpo HTML** en ellos.

## Examples

### Introducción de controles de usuario

Los controles de usuario se hacen para la reutilización en las páginas ASP.NET, similar a las páginas maestras. En lugar de compartir el diseño de la página base, los controles del usuario comparten un grupo de controles de servidor integrados HTML / ASP.NET o un diseño de formulario específico, por ejemplo, envío de comentarios o notas de invitados.

Un control de usuario puede contener controles HTML y controles de servidor ASP.NET, incluidos los scripts del lado del cliente.

Los controles de usuario generalmente incluyen `Control` directiva de `Control` sobre su definición:

```
<%@ Control Language="C#" AutoEventWireup="True" CodeFile="UserControl.ascx.cs" %>
```

Al igual que la página ASPX, los controles de usuario consisten en marcas que se pueden asociar con un código detrás del archivo para realizar ciertos eventos y tareas, por lo tanto, todas las etiquetas HTML disponibles en la página ASPX se pueden usar en los controles del usuario excepto `<html>` , `<body>` y `<form>` etiquetas.

Aquí hay un ejemplo para el marcado de control de usuario simple:

```
<%-- UserControl.ascx --%>
<%@ Control Language="C#" AutoEventWireup="True" CodeFile="UserControl.ascx.cs" %>
<div>
    <asp:Label ID="Label1" runat="server" />
    <br />
    <asp:Button ID="Button1" runat="server" Text="Click Here" OnClick="Button1_Click" />
</div>
```

Ejemplo de código detrás:

```
// UserControl.ascx.cs
public partial class UserControl : System.Web.UI.UserControl
{
    protected void Button1_Click(Object sender, EventArgs e)
```

```

    {
        Label1.Text = "Hello World!";
    }
}

```

Antes de insertar un control de usuario en la página ASPX, la directiva de `Register` debe declararse en la parte superior de la página que hace referencia al control de usuario con su URL de origen, nombre de etiqueta y prefijo de etiqueta.

```
<%@ Register Src="UserControl.ascx" TagName="UserControl" TagPrefix="uc" %>
```

Después, puede colocar el control de usuario dentro de la página ASPX como el control de servidor integrado de ASP.NET:

```
<uc:UserControl ID="UserControl1" runat="server" />
```

## Creación de instancia de control de usuario mediante programación

Si desea crear una instancia de un control de usuario dentro del código ASPX detrás de la página, debe escribir la declaración de control de usuario en el evento `Page_Load` siguiente manera:

```

public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)
    {
        Control control1 = LoadControl("UserControl.ascx");
        Page.Controls.Add(control1);
    }
}

```

Tenga en cuenta que el archivo ASCX de control del usuario ya debe estar creado al ejecutar el método `LoadControl`.

Otra forma conocida para declarar controles de usuario mediante programación es mediante `Placeholder`:

```

public partial class Default : System.Web.UI.Page
{
    public Placeholder Placeholder1;
    protected void Page_Load(Object sender, EventArgs e)
    {
        Control control1 = LoadControl("UserControl.ascx");
        Placeholder1.Controls.Add(control1);
    }
}

```

Dependiendo de su necesidad, `Placeholder` coloca los controles del usuario en un contenedor que almacena todos los controles del servidor agregados dinámicamente en la página, donde `Page.Controls` inserta directamente el control del usuario dentro de la página, lo que es más preferido para representar controles literales HTML.

## Agregar propiedades personalizadas para el control de usuario

Al igual que los controles de servidor incorporados estándar de ASP.NET, los controles de usuario pueden tener propiedades (atributos) en su etiqueta de definición. Supongamos que desea agregar un efecto de color en el archivo `UserControl.ascx` como este:

```
<uc:UserControl ID="UserControl1" runat="server" Color="blue" />
```

En este punto, los atributos / propiedades personalizados para los controles de usuario se pueden establecer declarando las propiedades dentro del código de control del usuario detrás de:

```
private String _color;
public String Color
{
    get
    {
        return _color;
    }
    set
    {
        _color = value;
    }
}
```

Además, si desea establecer un valor predeterminado en una propiedad de control de usuario, asigne el valor predeterminado dentro del método constructor del control de usuario.

```
public UserControl()
{
    _color = "red";
}
```

Luego, la marca de control del usuario debe modificarse para agregar el atributo de color como el siguiente ejemplo:

```
<%@ Control Language="C#" AutoEventWireup="True" CodeFile="UserControl.ascx.cs" %>
<div>
    <span style="color:<%= Color %>"><asp:Label ID="Label1" runat="server" /></span>
    <br />
    <asp:Button ID="Button1" runat="server" Text="Click Here" OnClick="Button1_Click" />
</div>
```

Lea ASP.NET - Controles de usuario en línea: <https://riptutorial.com/es/asp-net/topic/6773/asp-net---controles-de-usuario>

# Capítulo 5: ASP.NET - Estado de gestión

## Examples

### Ver estado

El siguiente ejemplo muestra el concepto de almacenar el estado de vista. Mantengamos un contador, que se incrementa cada vez que se publica la página haciendo clic en un botón en la página. Un control de etiqueta muestra el valor en el contador.

El código del archivo de marcado es el siguiente:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="statedemo._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

    <head runat="server">
        <title>
            Untitled Page
        </title>
    </head>

    <body>
        <form id="form1" runat="server">

            <div>
                <h3>View State demo</h3>

                Page Counter:

                <asp:Label ID="lblCounter" runat="server" />
                <asp:Button ID="btnIncrement" runat="server" Text="Add Count"
onclick="btnIncrement_Click" />
            </div>

        </form>
    </body>

</html>
```

El código detrás del archivo para el ejemplo se muestra aquí:

```
public partial class _Default : System.Web.UI.Page
{
    public int counter
    {
        get
        {
            if (ViewState["pcounter"] != null)
            {
```

```
        return ((int)ViewState["pcounter"]);
    }
    else
    {
        return 0;
    }
}

set
{
    ViewState["pcounter"] = value;
}
}

protected void Page_Load(object sender, EventArgs e)
{
    lblCounter.Text = counter.ToString();
    counter++;
}
}
```

Produciría el siguiente resultado:

Ver demostración del estado

### View State demo

Page Counter: 1

Lea ASP.NET - Estado de gestión en línea: <https://riptutorial.com/es/asp-net/topic/6296/asp-net---estado-de-gestion>



# Capítulo 6: ASP.NET - Validadores

## Sintaxis

- **RequiredFieldValidator Control:** `<asp: RequiredFieldValidator ID = "rfvcandidate" runat = "server" ControlToValidate = "ddlcandidate" ErrorMessage = "Por favor, elija un candidato" InitialValue = "Por favor, elija un candidato">  
</ asp: RequiredFieldValidator>`
- **Control RangeValidator:**  
`<asp: RangeValidator ID = "rvclass" runat = "server" ControlToValidate = "txtclass"  
ErrorMessage = "Ingrese su clase (6 - 12)" MaximumValue = "12" MinimumValue = "6" Type  
= "Integer">  
</ asp: RangeValidator>`
- **Control CompareValidator:** `<asp: CompareValidator ID = "CompareValidator1" runat = "server" ErrorMessage = "CompareValidator"> </ asp: CompareValidator>`
- **CustomValidator:**  
`<asp: CustomValidator ID = "CustomValidator1" runat = "server" ClientValidationFunction =  
.cvf_func. ErrorMessage = "CustomValidator">  
  
</ asp: CustomValidator>`
- **Resumen de validación:** `<asp: ValidationSummary ID = "ValidationSummary1" runat = "server" DisplayMode = "BulletList" ShowSummary = "true" HeaderText = "Errores:" />`

## Examples

### Controles de validación

Los controles de validación de ASP.NET validan los datos de entrada del usuario para garantizar que los datos inútiles, no autenticados o contradictorios no se almacenen.

ASP.NET proporciona los siguientes controles de validación:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- Validación resumen

### RequiredFieldValidator Control

El control `RequiredFieldValidator` asegura que el campo requerido no esté vacío. Por lo general, está vinculado a un cuadro de texto para forzar la entrada en el cuadro de texto.

La sintaxis del control es la dada:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
    runat="server" ControlToValidate ="ddlcandidate"
    ErrorMessage="Please choose a candidate"
    InitialValue="Please choose a candidate">

</asp:RequiredFieldValidator>
```

## Control RangeValidator

El control `RangeValidator` verifica que el valor de entrada se encuentre dentro de un rango predeterminado.

Tiene tres propiedades específicas:

Propiedades	Descripción
Tipo	Define el tipo de los datos. Los valores disponibles son: Moneda, Fecha,
Valor mínimo	Especifica el valor mínimo del rango.
Valor máximo	Especifica el valor máximo del rango.

La sintaxis del control es la dada:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
    ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
    MinimumValue="6" Type="Integer">

</asp:RangeValidator>
```

## CompareValidator Control

El control `CompareValidator` compara un valor en un control con un valor fijo o un valor en otro control.

Tiene las siguientes propiedades específicas:

Propiedades	Descripción
Tipo	Especifica el tipo de datos.
ControlToCompare	Especifica el valor del control de entrada para comparar con.
ValueToCompare	Especifica el valor constante a comparar con.

Propiedades	Descripción
ValueToCompare	Especifica el operador de comparación, los valores disponibles son: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual y DataTypeCheck.

La sintaxis básica del control es la siguiente:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ErrorMessage="CompareValidator">

</asp:CompareValidator>
```

## RegularExpressionValidator

El RegularExpressionValidator permite validar el texto de entrada mediante la comparación con un patrón de una expresión regular. La expresión regular se establece en la propiedad ValidationExpression.

La siguiente tabla resume las construcciones de sintaxis comúnmente utilizadas para expresiones regulares:

Escapes de personajes	Descripción
\segundo	Coincide con un retroceso.
\ t	Coincide con una pestaña.
\ r	Coincide con un retorno de carro.
\ v	Coincide con una pestaña vertical.
\F	Coincide con un feed de formulario.
\norte	Coincide con una nueva línea.
\	Personaje de escape.

Aparte de la coincidencia de un solo carácter, se podría especificar una clase de caracteres que se pueden combinar, llamados metacaracteres.

Metacaracteres	Descripción
.	Coincide con cualquier carácter excepto \ n.
[a B C D]	Coincide con cualquier personaje en el conjunto.
[^ abcd]	Excluye cualquier personaje del conjunto.

Metacaracteres	Descripción
[2-7a-mA-M]	Coincide con cualquier carácter especificado en el rango.
\w	Coincide con cualquier carácter alfanumérico y guión bajo.
\W	Coincide con cualquier carácter que no sea palabra.
\s	Coincide con caracteres de espacio en blanco como, espacio, tabulador, nueva línea, etc.
\S	Coincide con cualquier carácter que no sea un espacio en blanco.
\re	Coincide con cualquier carácter decimal.
\RE	Coincide con cualquier carácter no decimal.

Se podrían agregar cuantificadores para especificar el número de veces que podría aparecer un carácter.

Cuantificador	Descripción
*	Cero o más partidos.
+	Uno o más partidos.
?	Cero o uno partidos.
{NORTE}	N partidos.
{NORTE,}	N o más partidos.
{NUEVO MÉJICO}	Entre N y M partidos.

La sintaxis del control es la dada:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
    ValidationExpression="string" ValidationGroup="string">

</asp:RegularExpressionValidator>
```

## Resumen de validación

El control ValidationSummary no realiza ninguna validación pero muestra un resumen de todos los errores en la página. El resumen muestra los valores de la propiedad ErrorMessage de todos los controles de validación que fallaron la validación.

Las siguientes dos propiedades mutuamente exclusivas muestran el mensaje de error:

ShowSummary: muestra los mensajes de error en el formato especificado.

ShowMessageBox: muestra los mensajes de error en una ventana separada.

La sintaxis para el control es la dada:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
    DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

## Grupos de validación

Las páginas complejas tienen diferentes grupos de información provistos en diferentes paneles. En tal situación, podría surgir la necesidad de realizar la validación por separado para un grupo separado. Este tipo de situación se maneja utilizando grupos de validación.

Para crear un grupo de validación, debe colocar los controles de entrada y los controles de validación en el mismo grupo lógico estableciendo su propiedad ValidationGroup.

Ejemplo El siguiente ejemplo describe un formulario que deben llenar todos los estudiantes de una escuela, dividido en cuatro casas, para elegir al presidente de la escuela. Aquí, usamos los controles de validación para validar la entrada del usuario.

Este es el formulario en vista de diseño:

El código del archivo de contenido es el dado:

```
<form id="form1" runat="server">

    <table style="width: 66%; ">

        <tr>
            <td class="style1" colspan="3" align="center">
                <asp:Label ID="lblmsg"
                    Text="President Election Form : Choose your president"
                    runat="server" />
            </td>
        </tr>

        <tr>
            <td class="style3">
                Candidate:
            </td>
```

```

<td class="style2">
    <asp:DropDownList ID="ddlcandidate" runat="server" style="width:239px">
        <asp:ListItem>Please Choose a Candidate</asp:ListItem>
        <asp:ListItem>M H Kabir</asp:ListItem>
        <asp:ListItem>Steve Taylor</asp:ListItem>
        <asp:ListItem>John Abraham</asp:ListItem>
        <asp:ListItem>Venus Williams</asp:ListItem>
    </asp:DropDownList>
</td>

<td>
    <asp:RequiredFieldValidator ID="rfvcandidate"
        runat="server" ControlToValidate ="ddlcandidate"
        ErrorMessage="Please choose a candidate"
        InitialValue="Please choose a candidate">
    </asp:RequiredFieldValidator>
</td>
</tr>

<tr>
    <td class="style3">
        House:
    </td>

    <td class="style2">
        <asp:RadioButtonList ID="rblhouse" runat="server" RepeatLayout="Flow">
            <asp:ListItem>Red</asp:ListItem>
            <asp:ListItem>Blue</asp:ListItem>
            <asp:ListItem>Yellow</asp:ListItem>
            <asp:ListItem>Green</asp:ListItem>
        </asp:RadioButtonList>
    </td>

    <td>
        <asp:RequiredFieldValidator ID="rfvhouse" runat="server"
            ControlToValidate="rblhouse" ErrorMessage="Enter your house name" >
        </asp:RequiredFieldValidator>
        <br />
    </td>
</tr>

<tr>
    <td class="style3">
        Class:
    </td>

    <td class="style2">
        <asp:TextBox ID="txtclass" runat="server"></asp:TextBox>
    </td>

    <td>
        <asp:RangeValidator ID="rvclass"
            runat="server" ControlToValidate="txtclass"
            ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
            MinimumValue="6" Type="Integer">
        </asp:RangeValidator>
    </td>
</tr>

<tr>

```

```

        <td class="style3">
            Email:
        </td>

        <td class="style2">
            <asp:TextBox ID="txtemail" runat="server" style="width:250px">
            </asp:TextBox>
        </td>

        <td>
            <asp:RegularExpressionValidator ID="remail" runat="server"
                ControlToValidate="txtemail" ErrorMessage="Enter your email"
                ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*">
            </asp:RegularExpressionValidator>
        </td>
    </tr>

    <tr>
        <td class="style3" align="center" colspan="3">
            <asp:Button ID="btnsubmit" runat="server" onclick="btnsubmit_Click"
                style="text-align: center" Text="Submit" style="width:140px" />
        </td>
    </tr>
</table>
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
    DisplayMode ="BulletList" ShowSummary ="true" HeaderText="Errors:" />
</form>

```

El código detrás del botón de enviar:

```

protected void btnsubmit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblmsg.Text = "Thank You";
    }
    else
    {
        lblmsg.Text = "Fill up all the fields";
    }
}

```

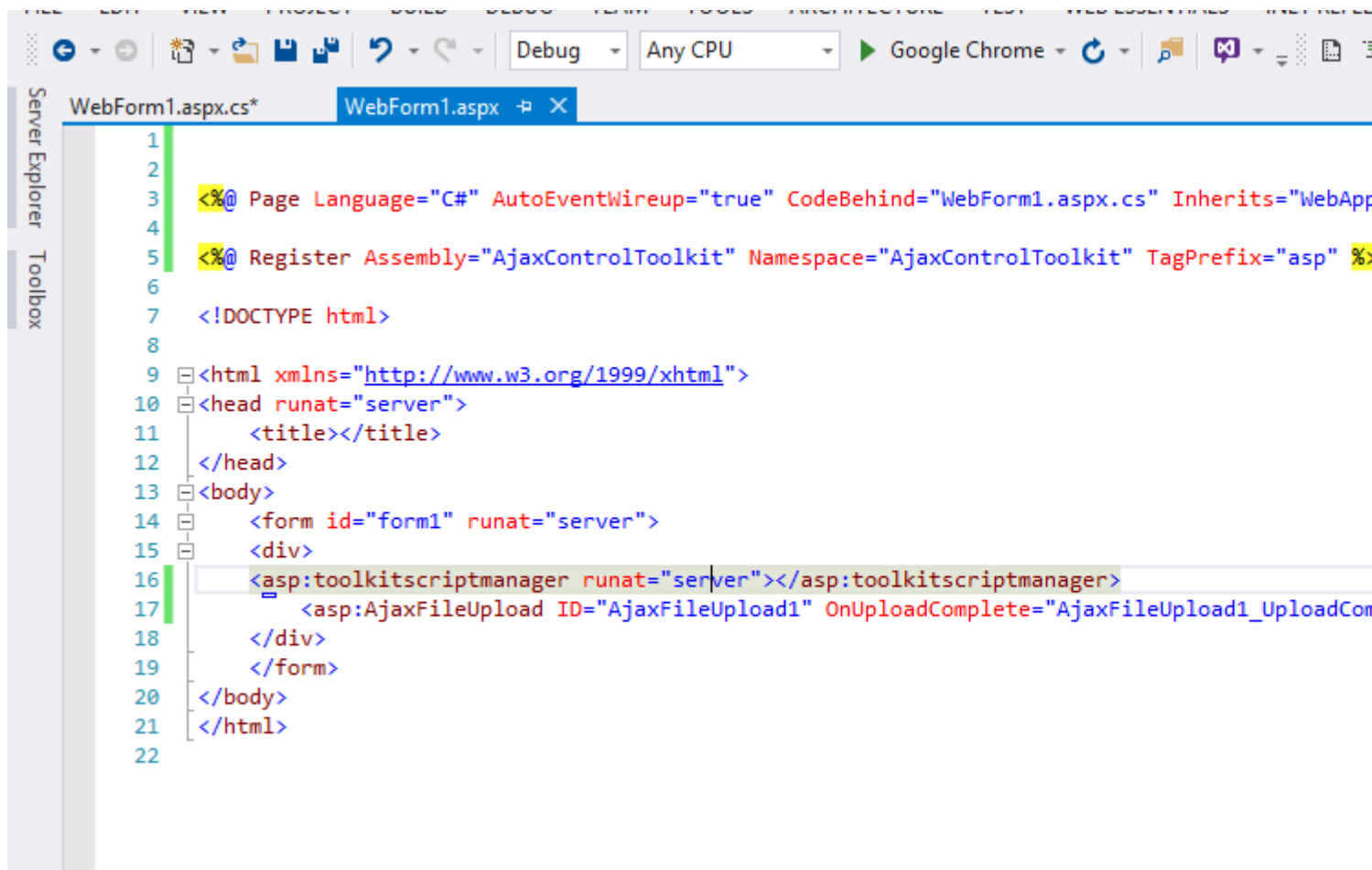
Lea ASP.NET - Validadores en línea: <https://riptutorial.com/es/asp-net/topic/6180/asp-net---validadores>

# Capítulo 7: Asp.net Ajax Controls

## Examples

### FileUpload Ajax Toolkit Control

1. Agregue una referencia de `AjaxToolkitControl.dll` en su proyecto.
2. Luego, arrastre y suelte `Toolkit Script Manager` y `AjaxFileUpload Control` desde la ventana de Visual Studio Toolbox a su página `.aspx` esta manera:



```
1 2
3  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApp
4  <%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit" TagPrefix="asp" %>
5  <!DOCTYPE html>
6
7  <html xmlns="http://www.w3.org/1999/xhtml">
8  <head runat="server">
9      <title></title>
10 </head>
11 <body>
12     <form id="form1" runat="server">
13     <div>
14         <asp:toolkitscriptmanager runat="server"></asp:toolkitscriptmanager>
15         <asp:AjaxFileUpload ID="AjaxFileUpload1" OnUploadComplete="AjaxFileUpload1_UploadCom
16     </div>
17     </form>
18 </body>
19 </html>
20
21
22
```

3. Usa este código en tu archivo `aspx.cs`



```

using System.Web.UI.WebControls;

namespace WebApplication1
{
    1 reference
    public partial class WebForm1 : System.Web.UI.Page
    {
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        0 references
        protected void AjaxFileUpload1_UploadComplete(object sender, AjaxControlToolkit.AjaxFileUploadEventArgs e)
        {
            string fileName = Path.GetFileName(e.FileName);
            AjaxFileUpload1.SaveAs(Server.MapPath("~/Uploads/" + fileName));
        }
    }
}

```

4. Asegúrese de que ha creado la carpeta denominada **Cargas** en el directorio raíz de su proyecto.

Lea Asp.net Ajax Controls en línea: <https://riptutorial.com/es/asp-net/topic/7164/asp-net-ajax-controls>

# Capítulo 8: ASP.NET Caching

## Examples

### Caché de datos

ASP.Net expone la API de caché para almacenar datos en la caché para su posterior recuperación.

#### Empezando

### Cadena de la tienda

```
Cache["key"]="value";
```

### Recuperar cadena

```
var value="";
if (Cache["key"] != null)
    value = Cache["key"].ToString();
```

También puede utilizar los métodos **Agregar** o **Insertar** .

```
protected void Page_Load( object sender, EventArgs e)
{
    if ( this.IsPostBack )
    {
        label1.Text + = "Page is posted back";
    }
    else
    {
        label1.Text + = "Page is created";
    }

    if ( Cache [ "item" ] == null )
    {
        label1.Text + = "New item is created";
        DateTime item = DateTime.Now;
        label1.Text + = "Item is stored";
        Cache.Insert ( "item", item, null );
        DateTime.Now.AddSeconds ( 20 ), TimeSpan.Zero;
    }

    else
    {
        label1.Text + = "Item is accesses";
        DateTime item = ( DateTime) Cache [ "item" ];
        label1.Text + = "Time is: " + item.ToString();
        label1.Text + = <br/>";
    }

    label1.Text + = "<br/>";
}
```

Lea ASP.NET Caching en línea: <https://riptutorial.com/es/asp-net/topic/9148/asp-net-caching>

---

# Capítulo 9: Ciclo de vida de la página

## Examples

### Eventos del Ciclo de Vida

Los siguientes son los eventos del ciclo de vida de la página:

**PreInit** - PreInit es el primer evento en el ciclo de vida de la página. Comprueba la propiedad `IsPostBack` y determina si la página es una devolución de datos. Establece los temas y las páginas maestras, crea controles dinámicos y obtiene y establece valores de propiedades de perfil. Este evento se puede controlar anulando el método `OnPreInit` o creando un controlador `Page_PreInit`.

**Init** : el evento `Init` inicializa la propiedad de control y se construye el árbol de control. Este evento se puede controlar anulando el método `OnInit` o creando un controlador `Page_Init`.

**InitComplete** : el evento `InitComplete` permite el seguimiento del estado de vista. Todos los controles activan el seguimiento de estado de vista.

**LoadViewState** : el evento `LoadViewState` permite cargar información de estado de vista en los controles.

**LoadPostData** : durante esta fase, se procesan los contenidos de todos los campos de entrada con la etiqueta.

**Carga previa** : la carga previa se produce antes de que se carguen los datos de la respuesta posterior en los controles. Este evento se puede controlar anulando el método `OnPreLoad` o creando un controlador `Page_PreLoad`.

**Cargar** : el evento `Cargar` se genera para la página primero y luego recursivamente para todos los controles secundarios. Se crean los controles en el árbol de control. Este evento se puede controlar anulando el método `OnLoad` o creando un controlador `Page_Load`.

**LoadComplete** : el proceso de carga se completa, los controladores de eventos de control se ejecutan y se lleva a cabo la validación de la página. Este evento se puede controlar anulando el método `OnLoadComplete` o creando un controlador `Page_LoadComplete`

**PreRender** : el evento `PreRender` se produce justo antes de que se genere la salida. Al manejar este evento, las páginas y los controles pueden realizar cualquier actualización antes de que se genere la salida.

**PreRenderComplete** : como el evento `PreRender` se activa recursivamente para todos los controles secundarios, este evento garantiza la finalización de la fase de representación previa.

**SaveStateComplete** : se **guarda el** estado de control de la página. Se guarda la personalización, el estado de control y la información de estado de vista. Se genera el marcado HTML. Esta etapa

se puede manejar anulando el método `Render` o creando un controlador `Page_Render`.

**UnLoad** : la fase `UnLoad` es la última fase del ciclo de vida de la página. Aumenta el evento `UnLoad` para todos los controles de forma recursiva y, por último, para la propia página. Se realiza la limpieza final y se liberan todos los recursos y referencias, como las conexiones de base de datos. Este evento se puede controlar anulando el método `OnUnLoad` o creando un controlador `Page_UnLoad`.

## Ejemplo de código

```
using System;

namespace myProject
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        public string PageSteps = string.Empty;

        //Raised after the start stage is complete and before the initialization stage begins.
        protected void Page_PreInit(object sender, EventArgs e)
        {
            PageSteps += "1 - Page_PreInit<br>";

            //Access to page Controls not available in this step
            //Label1.Text = "Step 1";
        }

        //Raised after all controls have been initialized and any skin settings have been
        applied.
        //The Init event of individual controls occurs before the Init event of the page.
        protected void Page_Init(object sender, EventArgs e)
        {
            PageSteps += "2 - Page_Init<br>";

            Label1.Text = "Step 2";
        }

        //Raised at the end of the page's initialization stage.
        //Only one operation takes place between the Init and InitComplete events: tracking of
        view state changes is turned on.
        //View state tracking enables controls to persist any values that are programmatically
        added to the ViewState collection.
        //Until view state tracking is turned on, any values added to view state are lost
        across postbacks.
        //Controls typically turn on view state tracking immediately after they raise their
        Init event.
        protected void Page_InitComplete(object sender, EventArgs e)
        {
            PageSteps += "3 - Page_InitComplete<br>";

            Label1.Text = "Step 3";
        }

        //Raised after the page loads view state for itself and all controls, and after it
        processes postback data that is included with the Request instance.
        protected override void OnPreLoad(EventArgs e)
        {
            PageSteps += "4 - OnPreLoad<br>";
        }
    }
}
```

```

        Label1.Text = "Step 4";
    }

    //The Page object calls the OnLoad method on the Page object, and then recursively
    does the same for each child control until the page and all controls are loaded.
    //The Load event of individual controls occurs after the Load event of the page.
    protected void Page_Load(object sender, EventArgs e)
    {
        PageSteps += "5 - Page_Load<br>";

        Label1.Text = "Step 5";
    }

    //Use these events to handle specific control events, such as a Button control's Click
    event or a TextBox control's TextChanged event.
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        //Step only visible on PostBack
        PageSteps += "6 - btnSubmit_Click<br>";

        Label1.Text = "Step 6";
    }

    //Raised at the end of the event-handling stage.
    protected void Page_LoadComplete(object sender, EventArgs e)
    {
        PageSteps += "7 - Page_LoadComplete<br>";

        Label1.Text = "Step 7";
    }

    //Raised after the Page object has created all controls that are required in order to
    render the page, including child controls of composite controls.
    //(To do this, the Page object calls EnsureChildControls for each control and for the
    page.)
    protected override void OnPreRender(EventArgs e)
    {
        PageSteps += "8 - OnPreRender<br>";

        Label1.Text = "Step 8";
    }

    //Raised after each data bound control whose DataSourceID property is set calls its
    DataBind method.
    protected override void OnPreRenderComplete(EventArgs e)
    {
        PageSteps += "9 - OnPreRenderComplete<br>";

        Label1.Text = "Step 9";
    }

    //Raised after view state and control state have been saved for the page and for all
    controls.
    //Any changes to the page or controls at this point affect rendering, but the changes
    will not be retrieved on the next postback.
    protected override void OnSaveStateComplete(EventArgs e)
    {
        PageSteps += "10 - OnSaveStateComplete<br><hr><br>";
    }

```

```

        Label1.Text = "Step 10";
    }

    // Render
    //This is not an event; instead, at this stage of processing, the Page object calls
    this method on each control.
    //All ASP.NET Web server controls have a Render method that writes out the control's
    markup to send to the browser.

    //Raised for each control and then for the page.
    //Controls use this event to do final cleanup for specific controls, such as closing
    control-specific database connections
    protected void Page_UnLoad(object sender, EventArgs e)
    {
        //This last PageSteps addition will not be visible on the page
        PageSteps += "11 - Page_UnLoad<br>";

        //Access to page Controls not available in this step
        //Label1.Text = "Step 11";
    }
}
}

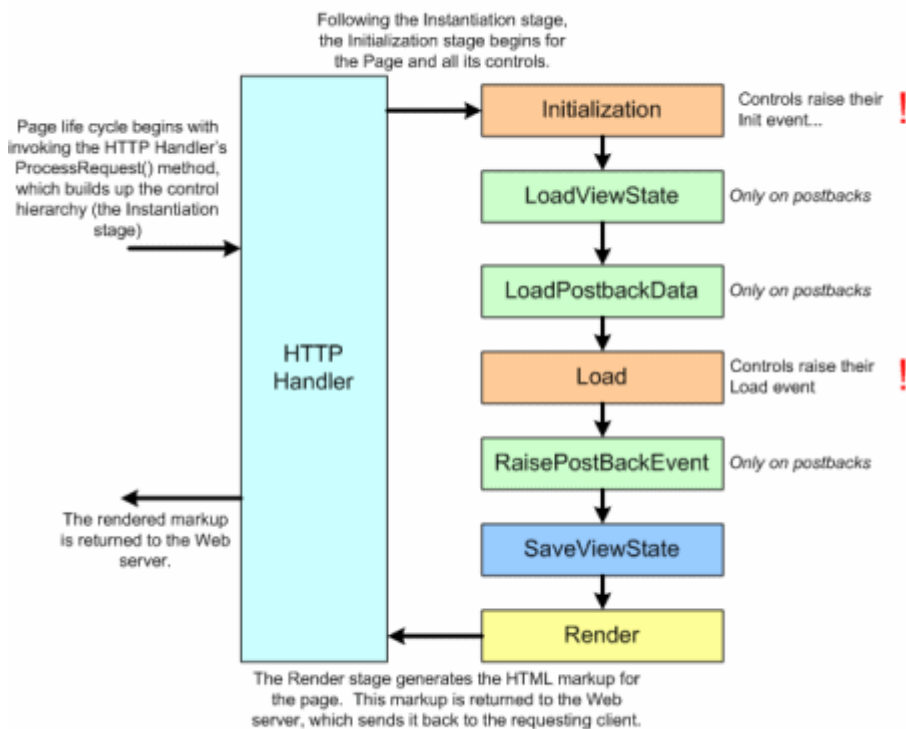
```

Agregue el siguiente código a la página .aspx para visualizar los Pasos en el Ciclo de Vida.

```

<b>Page Life Cycle Visualization:</b>
<br />
<%= PageSteps %>

```



## Más información

- <https://msdn.microsoft.com/en-us/library/ms178472.aspx>
- [https://www.tutorialspoint.com/asp.net/asp.net\\_life\\_cycle.htm](https://www.tutorialspoint.com/asp.net/asp.net_life_cycle.htm)
- <http://www.c-sharpcorner.com/UploadFile/8911c4/page-life-cycle-with-examples-in-Asp-Net/>
- <https://www.codeproject.com/Articles/667308/ASP-NET-Page-Life-Cycle-Events>

Lea Ciclo de vida de la página en línea: <https://riptutorial.com/es/asp-net/topic/4948/ciclo-de-vida-de-la-pagina>



# Capítulo 10: DayPilot Scheduler

## Parámetros

Parámetro	Desc
DataStartField	especifica la columna de origen de datos que contiene el inicio del evento (DateTime)
DataStartField	especifica la columna de origen de datos que contiene el inicio del evento (DateTime)
DataEndField	especifica la columna de origen de datos que contiene el final del evento (DateTime)
DataTextField	Especifica la columna de datos que contiene el texto del evento (cadena).
DataIdField	especifica la columna de origen de datos que contiene el ID de evento (cadena o entero)
DataResourceField	Especifica la columna de datos que contiene la clave externa del recurso de evento (cadena)

## Observaciones

Estos son los conceptos básicos de la programación de DayPilot, que deben explorarse más a fondo.

## Examples

### Información básica

El widget del Programador de DayPilot muestra una línea de tiempo para múltiples recursos. Soporta AJAX y HTML5. Localización automática y manual. Soporte completo de estilo CSS

### Declaración

```
<%@ Register Assembly="DayPilot" Namespace="DayPilot.Web.Ui" TagPrefix="DayPilot" %>
<DayPilot:DayPilotScheduler
  ID="DayPilotScheduler1"
  runat="server"

  DataStartField="eventstart"
  DataEndField="eventend"
  DataTextField="name"
```

```
DataIdField="id"
DataResourceField="resource_id"

CellGroupBy="Month"
Scale="Day"

EventMoveHandling="CallBack"
OnEventMove="DayPilotScheduler1_EventMove" >

</DayPilot:DayPilotScheduler>
```

Lea DayPilot Scheduler en línea: <https://riptutorial.com/es/asp-net/topic/6027/daypilot-scheduler>

# Capítulo 11: Delegación de eventos

## Sintaxis

```
1. public delegate void ActionClick();
```

```
public event ActionClick OnResetClick;
```

## Observaciones

No he encontrado ninguna desventaja en este enfoque, pero hay algunas cosas que hacen que esto sea un poco problemático.

1. Es necesario agregar un controlador de eventos para cada evento. Si no agrega los controladores de eventos en el evento OnInit de la página, puede enfrentar algunos problemas que en la publicación de la página atrás, perderá la asignación del evento (ya que ASP.NET no tiene estado, lo que no es el caso con los controles de Windows) .
2. En este enfoque, debe respetar los eventos del ciclo de vida de la página. Algunas veces, cuando está trabajando en el Diseñador, puede haber un caso en el que el controlador de eventos se pierda sin su notificación.
3. Incluso si no ha agregado el controlador de eventos, no obtendrá ningún error o advertencia. Si tiene varias páginas para realizar la misma acción, no hay garantía de que todos los nombres de los métodos sean iguales; el desarrollador puede elegir sus propios nombres de métodos, lo que reduce la capacidad de mantenimiento del código.

## Examples

### Delegación de eventos del control de usuario a aspx

Normalmente, optamos por este enfoque si queremos una encapsulación completa y no queremos hacer públicos nuestros métodos.

#### Ascx

```
<div style="width: 100%; ">
  <asp:Button ID="btnAdd" runat="server"
    Text="Add" OnClick="btnAdd_Click"></asp:button>
  <asp:button id="btnEdit" runat="server"
    text="Edit" onclick="btnEdit_Click"> </asp:button>
  <asp:button id="btnDelete" runat="server"
    text="Delete" onclick="btnDelete_Click"> </asp:Button>
  <asp:button id="btnReset" runat="server"
    text="Reset" onclick="btnReset_Click"></asp:button>
</div>
```

#### Ascx.cs

```

public delegate void ActionClick();

public partial class EventDelegation : System.Web.UI.UserControl
{
    public event ActionClick OnAddClick;
    public event ActionClick OnDeleteClick;
    public event ActionClick OnEditClick;
    public event ActionClick OnResetClick;
    protected void btnAdd_Click(object sender, EventArgs e)
    {
        if(OnAddClick!= null)
        {
            OnAddClick();
        }
    }

    protected void btnEdit_Click(object sender, EventArgs e)
    {
        if (OnEditClick != null)
        {
            OnEditClick();
        }
    }

    protected void btnDelete_Click(object sender, EventArgs e)
    {
        if(OnDeleteClick!= null)
        {
            OnDeleteClick();
        }
    }

    protected void btnReset_Click(object sender, EventArgs e)
    {
        if(OnResetClick!= null)
        {
            OnResetClick();
        }
    }
}

```

El control de usuario especifica algunos eventos públicos como `OnAddClick` , `OnEditClick` , etc., que declaran un delegado. Cualquier persona que quiera usar estos eventos debe agregar `EventHandler` para que se ejecute cuando se produzca el evento de clic de botón correspondiente.

## Diseño aspx

```

<%@ Register src="Controls/EventDelegation.ascx"
    tagname="EventDelegation" tagprefix="uc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title></title>
</head>
<body>

```

```
<form id="form1" runat="server">
<div>
    <uc1:Direct ID="Direct1" runat="server" />
</div>
</form>
</body>
</html>
```

## Aspx.cs

```
public partial class EventDelegation : System.Web.UI.Page
{
    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);
        ActionControl.OnAddClick += ActionControl_OnAddClick;
        ActionControl.OnDeleteClick += ActionControl_OnDeleteClick;
        ActionControl.OnEditClick += ActionControl_OnEditClick;
        ActionControl.OnResetClick += ActionControl_OnResetClick;
    }

    private void ActionControl_OnResetClick()
    {
        Response.Write("Reset done.");
    }

    private void ActionControl_OnEditClick()
    {
        Response.Write("Updated.");
    }

    private void ActionControl_OnDeleteClick()
    {
        Response.Write("Deleted.");
    }

    private void ActionControl_OnAddClick()
    {
        Response.Write("Added.");
    }
}
```

Lea Delegación de eventos en línea: <https://riptutorial.com/es/asp-net/topic/6927/delegacion-de-eventos>

# Capítulo 12: Directivas

## Examples

### La directiva de aplicación

La directiva de aplicación define atributos específicos de la aplicación. Se proporciona en la parte superior del archivo global.aspx. La sintaxis básica de la directiva de aplicación es:

```
<%@ Application Language="C#" %>
```

Los atributos de la directiva de aplicación son:

Atributos	Descripción
Hereda	El nombre de la clase de la cual heredar.
Descripción	La descripción del texto de la aplicación. Los analizadores y compiladores ignoran esto.
Idioma	El lenguaje utilizado en los bloques de código.

### La directiva de control

La directiva de control se usa con los controles de usuario y aparece en los archivos de control de usuario (.ascx).

La sintaxis básica de la directiva de control es:

```
<%@ Control Language="C#" EnableViewState="false" %>
```

Los atributos de la directiva de Control son:

Atributos	Descripción
AutoEventWireup	El valor booleano que habilita o inhabilita la asociación automática de eventos a los manejadores.
Nombre de la clase	El nombre del archivo para el control.
Depurar	El valor booleano que habilita o inhabilita la compilación con símbolos de depuración.
Descripción	La descripción de texto de la página de control, ignorada por el compilador.

Atributos	Descripción
EnableViewState	El valor booleano que indica si el estado de la vista se mantiene en todas las solicitudes de página.
Explícito	Para el lenguaje VB, le dice al compilador que use la opción modo explícito.
Hereda	La clase de la que se hereda la página de control.
Idioma	El lenguaje para código y script.
Src	El nombre de archivo para la clase de código subyacente.
Estricto	Para el lenguaje VB, le dice al compilador que use la opción modo estricto.

## La Directiva de Implementos.

La directiva de Implementación indica que la página web, la página maestra o la página de control del usuario deben implementar la interfaz de .Net framework especificada.

La sintaxis básica para la directiva de implementos es:

```
<%@ Implements Interface="interface_name" %>
```

## La Directiva Maestra

La directiva maestra especifica un archivo de página como la página principal.

La sintaxis básica de la directiva MasterPage de muestra es:

```
<%@ MasterPage Language="C#" AutoEventWireup="true" CodeFile="SiteMater.master.cs"
Inherits="SiteMaster" %>
```

## La directiva de importación

La directiva Importar importa un espacio de nombres a una página web, la página de control de usuario de la aplicación. Si la directiva de importación se especifica en el archivo global.asax, entonces se aplica a toda la aplicación. Si está en una página de la página de control de usuario, entonces se aplica a esa página o control.

La sintaxis básica para la directiva de importación es:

```
<%@ namespace="System.Drawing" %>
```

## La Directiva MasterType

La directiva MasterType asigna un nombre de clase a la propiedad maestra de una página, para que sea fuertemente tipada.

La sintaxis básica de la directiva MasterType es:

```
<%@ MasterType attribute="value"[attribute="value" ...] %>
```

## La directiva de la página

La directiva Page define los atributos específicos del archivo de página para el analizador de páginas y el compilador.

La sintaxis básica de la directiva Page es:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" Trace="true" %>
```

Los atributos de la directiva de la página son:

Atributos	Descripción
AutoEventWireup	El valor booleano que habilita o inhabilita los eventos de página que se están vinculando automáticamente a los métodos; por ejemplo, Page_Load.
Buffer	El valor booleano que habilita o inhabilita el búfer de respuesta HTTP.
Nombre de la clase	El nombre de la clase para la página.
ClientTarget	El navegador para el que controla el servidor debe mostrar contenido.
CodeFile	El nombre del código detrás del archivo.
Depurar	El valor booleano que habilita o inhabilita la compilación con símbolos de depuración.
Descripción	La descripción de texto de la página, ignorada por el analizador.
EnableSessionState	Habilita, deshabilita o hace que el estado de la sesión sea de solo lectura.
EnableViewState	El valor booleano que habilita o inhabilita el estado de vista en las solicitudes de página.
ErrorPage	URL para la redirección si se produce una excepción de página no controlada.
Hereda	El nombre del código detrás u otra clase.



Atributos	Descripción
Idioma	El lenguaje de programación para el código.
Src	El nombre del archivo del código detrás de la clase.
Rastro	Habilita o deshabilita el rastreo.
TraceMode	Indica cómo se muestran los mensajes de seguimiento y se ordenan por tiempo o categoría.
Transacción	Indica si las transacciones son compatibles.
ValidateRequest	El valor booleano que indica si todos los datos de entrada se validan con una lista de valores codificada.

## La directiva de outputcache

La directiva OutputCache controla las políticas de almacenamiento en caché de salida de una página web o un control de usuario.

La sintaxis básica de la directiva OutputCache es:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```

Lea Directivas en línea: <https://riptutorial.com/es/asp-net/topic/2255/directivas>

# Capítulo 13: El enlace de datos

## Examples

### Fuente de datos SQL

Los controles que pueden vincularse con datos pueden hacer uso de los controles `SqlDataSource`. El control `SqlDataSource` no solo le permite recuperar datos de una base de datos, sino también editar y ordenar los datos.

## Recuperando datos

Procedimiento almacenado:

```
<asp:SqlDataSource ID="SqlDataSourceEmployees"
    runat="server"
    ConnectionString="<%= ConnectionStrings:MyConnectionString %>"
    SelectCommand="sp_GetEmployees"
    SelectCommandType="StoredProcedure">
</asp:SqlDataSource>
```

Consulta SQL:

```
<asp:SqlDataSource ID="SqlDataSourceEmployees"
    runat="server"
    ConnectionString="<%= ConnectionStrings:MyConnectionString %>"
    SelectCommand="SELECT
        EmployeeID,
        EmployeeFirstName,
        EmployeeLastName
    FROM
        dbo.Employees">
</asp:SqlDataSource>
```

Parámetros:

```
<asp:SqlDataSource ID="SqlDataSourceEmployees"
    runat="server"
    ConnectionString="<%= ConnectionStrings:MyConnectionString %>"
    SelectCommand="SELECT
        EmployeeID,
        EmployeeFirstName,
        EmployeeLastName
    FROM
        dbo.Employees
    WHERE
        DepartmentID = @DepartmentID;">
<SelectParameters>
    <asp:ControlParameter ControlID="ddlDepartment"
        Name="DepartmentID"
        PropertyName="SelectedValue" />
</SelectParameters>
```

```
</SelectParameters>
</asp:SqlDataSource>
```

Tenga en cuenta la opción `CancelSelectOnNullParameter` , que si se establece en verdadero (predeterminado) detendrá el enlace de datos si cualquier parámetro es NULL

## Uso básico

Vista en cuadrícula:

```
<asp:GridView ID="GridViewEmployees"
    runat="server"
    AutoGenerateColumns="false"
    DataSourceID="SqlDataSourceEmployees">
    <Columns>
        <asp:BoundField DataField="EmployeeID" HeaderText="Employee ID" />
        <asp:BoundField DataField="EmployeeFirstName" HeaderText="First Name" />
        <asp:BoundField DataField="EmployeeLastName" HeaderText="Last Name" />
    </Columns>
</asp:GridView>
```

## Fuente de datos del objeto

```
<asp:ObjectDataSource ID="ObjectDataSourceEmployees" runat="server"
    TypeName="MyPackage.MyDataAccessClass"
    DataObjectTypeName="MyPackage.Employee"
    SelectMethod="GetEmployees"
    UpdateMethod="SaveEmployee"
    InsertMethod="SaveEmployee">
</asp:ObjectDataSource>
```

En el código detrás

La clase de acceso a datos

```
public class MyDataAccess
{
    public static List<Employee> GetEmployees()
    {
        List<Employee> results = new List<Employee>()
        {
            new Employee(){ Id=1, Name="John Smith" },
            new Employee(){ Id=2, Name="Mary Jane" }
        };

        return results;
    }

    public static void SaveEmployee(Employee e)
    {
        // Persist Employee e to the DB/cache etc. here
    }
}
```

## La clase de empleado

```
public class Employee
{
    public Int32 EmployeeId { get; set; }
    public string Name { get; set; }
}
```

Lea El enlace de datos en línea: <https://riptutorial.com/es/asp-net/topic/2245/el-enlace-de-datos>

---

# Capítulo 14: Encuentra Control por ID

## Sintaxis

```
1. control.FindControl("Id Of The Control To Be Found")
```

## Observaciones

- `FindControl` no es recursivo, solo busca a los hijos inmediatos del control
- Hay una sobrecarga `FindControl(String, int)` que no tiene sangría para uso público
- Si no se encuentra nada, `FindControl` devuelve `null`, por lo que a menudo es una buena idea verificar el resultado por no ser `null`

## Examples

### Accediendo al Control TextBox en la página aspx

```
TextBox txt = (TextBox)FindControl(yourtxt_Id);
```

### Encuentra un control en un GridView, Repeater, ListView, etc.

Si el control tiene filas.

```
TextBox tb = GridView1.Rows[i].FindControl("TextBox1") as TextBox;
```

O si tiene elementos.

```
TextBox tb = Repeater1.Items[i].FindControl("TextBox1") as TextBox;
```

Lea Encuentra Control por ID en línea: <https://riptutorial.com/es/asp-net/topic/6894/encuentra-control-por-id>

---

# Capítulo 15: Estado de sesión

## Sintaxis

- `Session ["Session_Key"] = Obj_Value;`

## Observaciones

HTTP es sin estado. El estado de la sesión ASP.NET es un marco que facilita el mantenimiento del estado entre las solicitudes de página HTTP.

La sesión difiere de las variables de nivel de clase en su capacidad de permanecer disponible a través de post-backs y diferentes páginas. Por ejemplo, una variable de sesión creada en `Page1.aspx` estará disponible si el usuario se redirige a `Page2.aspx` posteriormente, dentro de la misma aplicación.

Además, a diferencia de las variables estáticas declaradas a nivel de página, las variables de sesión son independientes para diferentes usuarios. Es decir, cambiar el valor de la variable de sesión de un usuario no afectará el valor de la misma variable para otros usuarios.

Si bien `ViewState` se puede usar para almacenar los datos del usuario temporalmente, no permite guardar datos en varias páginas. Además, el estado de `viewstate` es parte de la página y se envía al cliente. Como resultado, cualquier información crítica relacionada con el usuario no se puede guardar en `ViewState`, y ahí es donde las variables de la sesión se vuelven útiles.

## Examples

### Usando el objeto `Session` para almacenar valores

El objeto `System.Web.SessionState.HttpSessionState` proporciona una forma de conservar los valores entre las solicitudes HTTP. En el siguiente ejemplo, la preferencia de un usuario por las advertencias se guarda en la sesión. Más adelante, mientras se entrega otra solicitud al usuario, la aplicación puede leer esta preferencia de la sesión y ocultar las advertencias.

```
public partial class Default : System.Web.UI.Page
{
    public void LoadPreferences(object sender, EventArgs args)
    {
        // ...
        // ... A DB operation that loads the user's preferences.
        // ...

        // Store a value with the key showWarnings
        HttpContext.Current.Session["showWarnings"] = false;
    }

    public void button2Clicked(object sender, EventArgs args)
    {
```

```

    // While processing another request, access this value.
    bool showWarnings = (bool)HttpContext.Current.Session["showWarnings"];
    lblWarnings.Visible = false;
}
}

```

Tenga en cuenta que las variables de sesión no son comunes para todos los usuarios (al igual que las cookies), y se conservan en varias respuestas posteriores.

La sesión funciona configurando una cookie que contiene un identificador para la sesión de los usuarios. De forma predeterminada, este identificador se almacena en la memoria del servidor web, junto con los valores almacenados en su contra.

Aquí hay una captura de pantalla del conjunto de cookies en el navegador del usuario para realizar un seguimiento de la sesión:

Name	Value
ASP.NET_SessionId	3235CC720E020D5F045

## Usando un almacén de sesión SQL

Si descubre que tiene varios servidores que necesitan compartir el estado de la sesión, almacenarlo en la memoria de proceso de ASP.NET no funcionará. Por ejemplo, puede implementarse en un entorno de granja de servidores web con un equilibrador de carga que distribuye solicitudes de forma rotatoria. En este entorno, las solicitudes de un solo usuario pueden ser atendidas por múltiples servidores.

En el archivo web.config puede configurar un almacén de sesión del servidor SQL.

```

<configuration>
  <system.web>
    <sessionState
      mode="SQLServer"
      sqlConnectionString="Data Source=localhost;Integrated Security=SSPI"
      cookieless="true"
      timeout="30" />
    </system.web>
  </configuration>

```

Para crear el esquema sql use la herramienta aspnet\_regsql. [SampleSqlServerName] es el nombre de host del servidor SQL. -ssadd le dice a la herramienta que cree la base de datos de estado de sesión. -sstype p le dice a la herramienta que cree una nueva base de datos con el nombre predeterminado ASPState.

```
aspnet_regsql.exe -S [SampleSqlServerName] -U [Username] -P [Password] -ssadd -sstype p
```

## Usando un almacén de sesión de Amazon DynamoDB

Si no desea utilizar el servidor SQL, puede usar la base de datos nosql de Dynamo DB alojada de

Amazon como almacén de sesiones.

Necesitarás el SDK de AWS. Para instalar esto desde la consola del administrador de paquetes nuget de Visual Studio, use el siguiente comando

```
Install-Package AWSSDK
```

Luego puede configurar su proveedor de sessionState para usar un proveedor personalizado. Debe especificar la región y las credenciales, ya sea un perfil o una combinación de clave secreta y acceso IAM. De forma predeterminada, esto creará una tabla llamada ASP.NET\_SessionState.

```
<configuration>
  <system.web>
    <sessionState
      timeout="20"
      mode="Custom"
      customProvider="DynamoDBSessionStoreProvider">
      <providers>
        <add name="DynamoDBSessionStoreProvider"
          type="Amazon.SessionProvider.DynamoDBSessionStateStore"
          AWSProfileName="[PROFILE]"
          Region="[REGION]"
          CreateIfNotExist="true"
          />
      </providers>
    </sessionState>
  </system.web>
</configuration>
```

Lea Estado de sesión en línea: <https://riptutorial.com/es/asp-net/topic/3864/estado-de-sesion>



---

# Capítulo 16: Expresiones

## Examples

### Valor desde App.Config

```
<asp:Literal runat="server" text="<%$ AppSettings:MyAppSettingName %>"/>
```

### Expresion Evaluada

```
<div>
  The time is now <%= DateTime.Now.ToString() %>
</div>
```

### Bloque de código dentro del marcado ASP

```
<div>
  <form id="form1" runat="server">
    <%
      for (int i = 1; i <= 10; j++)
      {
        Response.Write(i) + " ";
      }
    %>
  </form>
</div>
```

Lea Expresiones en línea: <https://riptutorial.com/es/asp-net/topic/6326/expresiones>

# Capítulo 17: Formas Web

## Sintaxis

- `<asp: TextBox runat = "server" ID = "" TextMode = "" Text = "" />`
- `<asp: Repeater runat = "server" ID = "" OnItemDataBound = "">`  
    `<HeaderTemplate></HeaderTemplate>`  
    `<ItemTemplate></ItemTemplate>`  
    `<FooterTemplate></FooterTemplate>`  
    `</asp:Repeater>`

## Observaciones

Todos los controles de ASP.Net WebForm requieren `runat="server"` para comunicarse con el CodeBehind.

## Examples

### Usando un repetidor para crear una tabla HTML

Cuando el repetidor está enlazado, para cada elemento de los datos, se agregará una nueva fila de la tabla.

```
<asp:Repeater ID="repeaterID" runat="server" OnItemDataBound="repeaterID_ItemDataBound">
  <HeaderTemplate>
    <table>
      <thead>
        <tr>
          <th style="width: 10%">Column 1 Header</th>
          <th style="width: 30%">Column 2 Header</th>
          <th style="width: 30%">Column 3 Header</th>
          <th style="width: 30%">Column 4 Header</th>
        </tr>
      </thead>
    </HeaderTemplate>
    <ItemTemplate>
      <tr runat="server" id="rowID">
        <td>
          <asp:Label runat="server" ID="mylabel">You can add ASP labels if you
want</asp:Label>
        </td>
        <td>
          <label>Or you can add HTML labels.</label>
        </td>
        <td>
          You can also just type plain text like this.
        </td>
        <td>
          <button type="button">You can even add a button to the table if you
want!</button>
        </td>
      </tr>
    </ItemTemplate>
  </asp:Repeater>
```

```

</ItemTemplate>
<FooterTemplate>
    </table>
</FooterTemplate>
</asp:Repeater>

```

El método `ItemDataBound` es opcional, pero útil para formatear o `ItemDataBound` datos más complicados. En este ejemplo, el método se utiliza para dar dinámicamente a cada `<tr>` una ID única. Esta ID se puede usar en JavaScript para acceder o modificar una fila específica. Tenga en cuenta que el `tr` no mantendrá su valor de ID dinámico en `PostBack`. El texto de cada fila `<asp:Label>` también se estableció en este método.

```

protected void repeaterID_ItemDataBound(object sender, RepeaterItemEventArgs e)
{
    if (e.Item.ItemType == ListItemType.Item || e.Item.ItemType ==
    ListItemType.AlternatingItem)
    {
        MyItem item = (MyItem)e.Item.DataItem;

        var row = e.Item.FindControl("rowID");
        row.ClientIDMode = ClientIDMode.Static;
        row.ID = "rowID" + item.ID;

        Label mylabel = (Label)e.Item.FindControl("mylabel");
        mylabel.Text = "The item ID is: " + item.ID;
    }
}

```

Si planeas hacer mucha comunicación con el `CodeBehind`, deberías considerar usar `GridView`. Sin embargo, los repetidores, en general, tienen menos sobrecarga que `GridView` y, con la manipulación de ID básica, pueden realizar las mismas funciones que `GridView`.

## Agrupación en ListView

`asp:ListView` introducido en ASP.NET WebForms framework 3.5 es el más flexible de todos los controles de presentación de datos en el marco. Un ejemplo de agrupación usando `ListView` (que será útil como galería de imágenes)

**Objetivo :** Mostrar tres imágenes en una fila usando `asp:ListView`

### Margen

```

<asp:ListView ID="SportsImageList" runat="server"
    GroupItemCount="3">
    <LayoutTemplate>
        <span class="images-list">
            <ul id="groupPlaceholder" runat="server"></ul>
        </span>
    </LayoutTemplate>
    <GroupTemplate>
        <ul>
            <li id="itemPlaceholder" runat="server"></li>
        </ul>
    </GroupTemplate>

```

```
<ItemTemplate>
    <li>
        <img src='<%# Container.DataItem %>' />
    </li>
</ItemTemplate>
</asp:ListView>
```

## Código detrás

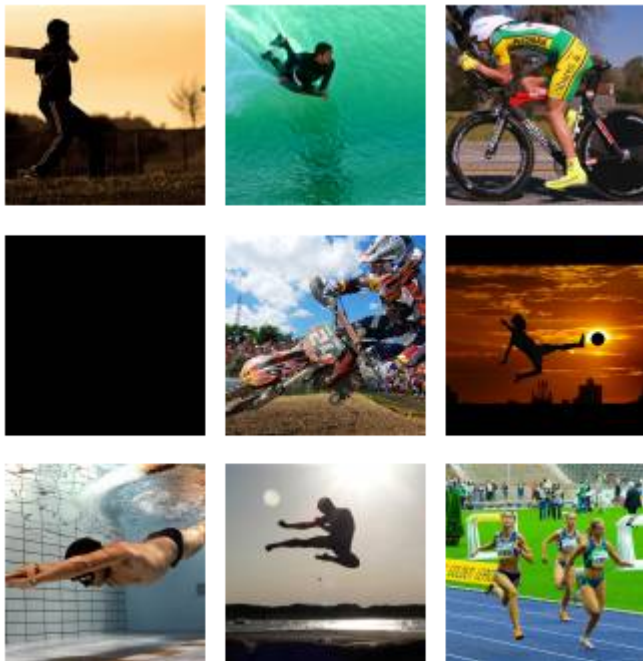
```
protected void Page_Load(object sender, EventArgs e)
{
    if(!IsPostBack)
    {
        SportsImageList.DataSource = GetImages();
        SportsImageList.DataBind();
    }
}

private static IEnumerable<string> GetImages()
{
    var images = Enumerable.Range(1, 9) //get numbers 1 to 9
        .Select(i =>
            string.Format("http://lorempixel.com/100/100/sports/{0}/", i)
        ); //convert the numbers to string
    return images;
}
```

## CSS

```
.images-list ul{
    clear: both;
    list-style-type: none;
}
.images-list ul li{
    float: left;
    padding: 5px;
}
```

## Salida renderizada



## Ejemplo

```
<script language="VB" runat="server">

    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        Label1.Text = "Text1.Text = " & Text1.Text
    End Sub

</script>
```

```
<h3><font face="Verdana">TextBox Sample</font></h3>

<form runat="server">

    <asp:TextBox id="Text1" Text="Copy this text to the label" Width="200px" runat="server"/>

    <asp:Button OnClick="SubmitBtn_Click" Text="Copy Text to Label" Runat="server"/>

    <p>

    <asp:Label id="Label1" Text="Label1" runat="server"/>

</form>
```

## Hiperenlace

El control HyperLink se utiliza para navegar desde el cliente a otra página.

```
<html>

<script language="VB" runat="server">

    Sub Page_Load(sender As Object, e As EventArgs)
```

```
' Set hyperlink to "~", which indicates application root.
HyperLink1.NavigateUrl = "~"
End Sub

</script>

<body>

  <h3><font face="Verdana">Simple asp:hyperlink Sample</font></h3>

  <form runat=server>

    <p>

      <asp:hyperlink id=HyperLink1 runat="server">
        Go To QuickStart
      </asp:hyperlink>

    </p>

  </form>

</body>

</html>
```

Lea Formas Web en línea: <https://riptutorial.com/es/asp-net/topic/5394/formas-web>

# Capítulo 18: Gestion de Sesiones

## Examples

### Ventaja y desventaja del estado de sesión, tipos de sesión

The advantages of using Session State are

- 1) Better security
- 2) Reduced bandwidth

The disadvantages of using Session state are

- 1) More resource consumption of server.
- 2) Extra code/care if a Web farm is used (we will discuss this shortly)

**\*\*Session State Modes\*\***

1) InProc mode, which stores session state in memory on the Web server. This is the default.

2) StateServer mode, which stores session state in a separate process called the ASP.NET state service. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm.

3) SQLServer mode stores session state in a SQL Server database. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm.

4) Custom mode, which enables you to specify a custom storage provider.

Off mode, which disables session state.

Lea Gestion de Sesiones en línea: <https://riptutorial.com/es/asp-net/topic/4180/gestion-de-sesiones>

# Capítulo 19: httpHandlers

## Examples

### Uso de un httpHandler (.ashx) para descargar un archivo desde una ubicación específica

Cree un nuevo httpHandler dentro de su proyecto ASP.NET. Aplique el siguiente código (VB) al archivo del controlador:

```
Public Class AttachmentDownload
    Implements System.Web.IHttpHandler

    Sub ProcessRequest(ByVal context As HttpContext) Implements IHttpHandler.ProcessRequest

        ' pass an ID through the query string to append a unique identifier to your
        downloadable fileName

        Dim fileUniqueId As Integer = CInt(context.Request.QueryString("id"))

        ' file path could also be something like "C:\FolderName\FilesForUserToDownload

        Dim filePath As String = "\\ServerName\FolderName\FilesForUserToDownload"
        Dim fileName As String = "UserWillDownloadThisFile_" & fileUniqueId
        Dim fullFilePath = filePath & "\" & fileName
        Dim byteArray() As Byte = File.ReadAllBytes(fullFilePath)

        ' prompt the user to download the file

        context.Response.Clear()
        context.Response.ContentType = "application/x-please-download-me" ' "application/x-
unknown"
        context.Response.AppendHeader("Content-Disposition", "attachment; filename=" &
fileName)
        context.Response.BinaryWrite(byteArray)
        context.Response.Flush()
        context.Response.Close()
        byteArray = Nothing

    End Sub

    ReadOnly Property IsReusable() As Boolean Implements IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

End Class
```

Puede llamar al controlador desde el código que está detrás, o desde un idioma del lado del cliente. En este ejemplo, estoy usando un javascript que llamará al controlador.

```
function openAttachmentDownloadHandler(fileId) {
```



```
// the location of your handler, and query strings to be passed to it

var url = "..\\_Handlers\\AttachmentDownload.ashx?";
url = url + "id=" + fileId;

// opening the handler will run its code, and it will close automatically
// when it is finished.

window.open(url);

}
```

Ahora adjunte la asignación de la función javascript a un evento de clic de botón en un elemento seleccionable en su formulario web. Por ejemplo:

```
<asp:LinkButton ID="lbtnDownloadFile" runat="server"
OnClientClick="openAttachmentDownloadHandler(20);">Download A File</asp:LinkButton>
```

O también puedes llamar a la función javascript desde el código que está detrás:

```
ScriptManager.RegisterStartupScript(Page,
    Page.GetType(),
    "openAttachmentDownloadHandler",
    "openAttachmentDownloadHandler(" & fileId & ");",
    True)
```

Ahora, cuando haga clic en su botón, el httpHandler llevará su archivo al navegador y le preguntará al usuario si desea descargarlo.

Lea httpHandlers en línea: <https://riptutorial.com/es/asp-net/topic/3476/httphandlers>

---

# Capítulo 20: Katana

## Introducción

**¿Qué es Katana?** Katana es un conjunto de componentes de código abierto para crear y alojar aplicaciones web basadas en OWIN, mantenidas por Microsoft Open Technologies Group. Katana proporciona una implementación de la especificación OWIN y, de hecho, se usa en un número creciente de plantillas de proyectos ASP.NET. . Además, Katana proporciona una amplia variedad de componentes de middleware listos para usar, listos para usar en una aplicación basada en OWIN.

## Examples

### Ejemplo

#### Aplicación básica de KatanaConsole

```
namespace KatanaConsole
{
    // use an alias for the OWIN AppFunc:
    using AppFunc = Func<IDictionary<string, object>, Task>;

    class Program
    {
        static void Main(string[] args)
        {
            WebApp.Start<Startup>("http://localhost:8080");
            Console.WriteLine("Server Started; Press enter to Quit");
            Console.ReadLine();
        }
    }

    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            var middleware = new Func<AppFunc, AppFunc>(MyMiddleWare);
            app.Use(middleware);
        }

        public AppFunc MyMiddleWare(AppFunc next)
        {
            AppFunc appFunc = async (IDictionary<string, object> environment) =>
            {
                // Do something with the incoming request:
                var response = environment["owin.ResponseBody"] as Stream;
                using (var writer = new StreamWriter(response))
                {
                    await writer.WriteAsync("<h1>Hello from My First Middleware</h1>");
                }
                // Call the next Middleware in the chain:
                await next.Invoke(environment);
            };
        }
    }
}
```

```
        return appFunc;  
    }  
}  
}
```

Lea Katana en línea: <https://riptutorial.com/es/asp-net/topic/8236/katana>

# Capítulo 21: Lista de datos

## Sintaxis

1. **ItemTemplate** : Potencia el contenido y el diseño de los elementos dentro de la lista. Esto es obligatorio.
2. **AlternatingItemTemplate** : si se menciona, determina el contenido y el diseño de los elementos alternos. Si no se menciona, se utiliza ItemTemplate.
3. **Plantilla de separador** : si se menciona, se representa entre elementos (y elementos alternos). Si no se menciona, no se renderiza un separador.
4. **SelectedItemTemplate** : si se menciona, determina el contenido y el diseño del elemento seleccionado. Si no se menciona, se usa ItemTemplate (AlternatingItemTemplate).
5. **EditItemTemplate** : si se menciona, determina el contenido y el diseño del elemento que se está editando. Si no se menciona, se usa ItemTemplate (AlternatingItemTemplate, SelectedItemTemplate).
6. **HeaderTemplate** : si se menciona, determina el contenido y el diseño del encabezado de la lista. Si no se menciona, el encabezado no se representa.
7. **Plantilla de pie de página** : si se menciona, determina el contenido y el diseño del pie de página de la lista. Si no se menciona, el pie de página no se renderiza.

## Examples

### Enlace de datos en asp.net

#### Aspx

```
<asp:DataList runat="server" CssClass="sample" RepeatLayout="Flow" ID="dlsamplecontent"
RepeatDirection="Vertical" OnItemCommand="dlsamplecontent_ItemCommand">
    <ItemStyle CssClass="tdContainer" />
    <ItemTemplate>
        //you code
    </ItemTemplate>
</asp:DataList>
```

#### Aspx.cs

```
public void GetSamplingContentType()
{
    try
    {
        ErrorLogger.gstrClientMethodName = this.GetType().FullName + "_" +
        System.Reflection.MethodBase.GetCurrentMethod().Name + " : ";

        DataTable dt = new DataTable();
        dlsamplecontent.DataSource = dt;
        dlsamplecontent.DataBind();
    }
}
```

```
        catch (Exception ex)
        {
            ErrorLogger.ClientErrorLogger(ex);
        }
    }
```

## Elemento de comando y recuperación de identificación utilizando el argumento de comando

```
protected void dlsamplecontent_ItemCommand(object source, DataListCommandEventArgs e)
{
    try
    {
        int BlogId = Convert.ToInt32(e.CommandArgument.ToString());
        if (e.CommandName == "SampleName")
        {
            //your code
        }
    }
    catch (Exception ex)
    {
        ErrorLogger.ClientErrorLogger(ex);
    }
}
```

Lea Lista de datos en línea: <https://riptutorial.com/es/asp-net/topic/7041/lista-de-datos>

# Capítulo 22: Manejo de eventos

## Sintaxis

- Nombre de evento vacío privado (remitente del objeto, EventArgs e);

## Parámetros

Parámetro	Detalles
objeto remitente	remitente se refiere al objeto que invocó el evento que activó el controlador de eventos. Esto es útil si tiene muchos objetos utilizando el mismo controlador de eventos.
EventArgs e	EventArgs es algo así como una clase base ficticia. En sí mismo, es más o menos inútil, pero si se deriva de él, puede agregar cualquier información que necesite pasar a sus manejadores de eventos.

## Examples

### Eventos de aplicación y sesión

Los eventos de aplicación más importantes son:

**Application\_Start** : se genera cuando se inicia la aplicación / sitio web.

**Application\_End** : se genera cuando se detiene la aplicación / sitio web.

Del mismo modo, los eventos de sesión más utilizados son:

**Session\_Start** : se **genera** cuando un usuario solicita una página de la aplicación por primera vez.

**Session\_End** - Se **genera** cuando finaliza la sesión.

### Eventos de página y control

Los eventos comunes de página y control son:

**Enlace de datos** : se genera cuando un control se enlaza con un origen de datos.

**Disposed** : se genera cuando se suelta la página o el control.

**Error** : es un evento de página, se produce cuando se lanza una excepción no controlada.

**Inicial** : se genera cuando se inicializa la página o el control.

**Cargar** : se genera cuando se carga la página o un control.

**PreRender** : se **genera** cuando se va a representar la página o el control.

**Descargar** : se genera cuando la página o el control se descargan de la memoria.

## Eventos predeterminados

El evento predeterminado para el objeto Page es el evento Load. Del mismo modo, cada control tiene un evento predeterminado. Por ejemplo, el evento predeterminado para el control de botón es el evento Click.

El controlador de eventos predeterminado podría crearse en Visual Studio, simplemente haciendo doble clic en el control en la vista de diseño. La siguiente tabla muestra algunos de los eventos predeterminados para controles comunes:

Controlar	Evento predeterminado
AdRotator	AdCreated
Lista con viñetas	Hacer clic
Botón	Hacer clic
Calandrar	SelectionChanged
Caja	CheckedChanged
CheckBoxList	SelectedIndexChanged
Cuadrícula de datos	SelectedIndexChanged
Lista de datos	SelectedIndexChanged
La lista desplegable	SelectedIndexChanged
Hiperenlace	Hacer clic
Botón de imagen	Hacer clic
Mapa de imagen	Hacer clic
LinkButton	Hacer clic
Cuadro de lista	SelectedIndexChanged
Menu MenuItem	Hacer clic
Boton de radio	CheckedChanged
RadioButtonList	SelectedIndexChanged

**Ejemplo** Este ejemplo incluye una página simple con un control de etiqueta y un control de botón en él. A medida que ocurren los eventos de la página, como Page\_Load, Page\_Init, Page\_PreRender, etc., se envía un mensaje, que se muestra mediante el control de etiqueta. Cuando se hace clic en el botón, se levanta el evento Button\_Click y también se envía un mensaje para que se muestre en la etiqueta.

Cree un nuevo sitio web y arrastre un control de etiqueta y un control de botón desde la caja de herramientas de control. Usando la ventana de propiedades, establezca las ID de los controles como .lblmessage. y .btnclick. respectivamente. Establezca la propiedad de Texto del control Button como 'Clic'.

El archivo de marcas (.aspx):

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="eventdemo._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

    <head runat="server">
        <title>Untitled Page</title>
    </head>

    <body>
        <form id="form1" runat="server">
            <div>
                <asp:Label ID="lblmessage" runat="server" >

                </asp:Label>

                <br />
                <br />
                <br />

                <asp:Button ID="btnclick" runat="server" Text="Click" onclick="btnclick_Click" />
            </div>
        </form>
    </body>

</html>
```

Haga doble clic en la vista de diseño para moverse al código detrás del archivo. El evento Page\_Load se crea automáticamente sin ningún código en él. Escriba las siguientes líneas de código autoexplicativas:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;

using System.Web;
using System.Web.Security;
using System.Web.UI;
```



```

using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

namespace eventdemo {

    public partial class _Default : System.Web.UI.Page {

        protected void Page_Load(object sender, EventArgs e) {
            lblmessage.Text += "Page load event handled. <br />";

            if (Page.IsPostBack) {
                lblmessage.Text += "Page post back event handled.<br/>";
            }
        }

        protected void Page_Init(object sender, EventArgs e) {
            lblmessage.Text += "Page initialization event handled.<br/>";
        }

        protected void Page_PreRender(object sender, EventArgs e) {
            lblmessage.Text += "Page prerender event handled. <br/>";
        }

        protected void btnclick_Click(object sender, EventArgs e) {
            lblmessage.Text += "Button click event handled. <br/>";
        }
    }
}

```

Ejecutar la página. La etiqueta muestra la carga de la página, la inicialización de la página y los eventos de renderización previa de la página. Haga clic en el botón para ver el efecto:



Lea Manejo de eventos en línea: <https://riptutorial.com/es/asp-net/topic/2347/manejo-de-eventos>

# Capítulo 23: Métodos de página

## Parámetros

Parámetro	Detalle
límite	El parámetro del método C #. Usted proporciona el argumento a través del método de la página.
sobre el éxito	La función de JavaScript que se ejecuta cuando la llamada al método de página tiene éxito.
en error	La función de JavaScript que se ejecuta cuando hay un error en la llamada al método de página.

## Observaciones

## Más de un parámetro

En el ejemplo, la función C # solo solicita un parámetro, si necesita pasar más de uno, puede hacerlo, solo ordénelos en su llamada JS y estará listo. Ej.

```
//C#
public static int SumValues(int num1, int num2, int num3, ..., int numN)

//JS
PageMethods.SumValues(num1, num2, num3, ..., numN, onSuccess, onError);
```

## Valor de retorno

En la función `onSuccess` , el resultado será el valor de retorno de la función C #. En la función `onError` el resultado será el error.

## Examples

### Como llamarlo

Simplemente agregue el `using` al principio y el decorador `[WebMethod]` al método `static` que se llamará en la página `aspx`:

```
using System.Web.Services;
```

```
public partial class MyPage : System.Web.UI.Page
{
    [WebMethod]
    public static int GetRandomNumberLessThan(int limit)
    {
        var r = new Random();
        return r.Next(limit);
    }
}
```

En su archivo .aspx agregue un asp: ScriptManager habilitando los métodos de página:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true">
</asp:ScriptManager>
```

Entonces puedes llamarlo de JS así:

```
var limit= 42 // your parameter value
PageMethods.GetRandomNumberLessThan(limit, onSuccess, onError);
function onSuccess(result) {
    var randomNumber = result;
    // use randomNumber...
}
function onError(result) {
    alert('Error: ' + result);
}
```

Lea Métodos de página en línea: <https://riptutorial.com/es/asp-net/topic/1411/metodos-de-pagina>

# Capítulo 24: Middleware

## Parámetros

Parámetro	Detalles
<code>IDictionary&lt;string, object&gt; environment</code>	Esta es la única recopilación en la que OWIN comunica información durante una llamada. Todas las claves se pueden encontrar en <a href="https://docs.asp.net/en/latest/fundamentals/owin.html#owin-keys">https://docs.asp.net/en/latest/fundamentals/owin.html#owin-keys</a>

## Observaciones

El tipo `AppFunc` es solo un alias para el tipo `Func<IDictionary<string, object>, Task>` para acortar las firmas de métodos, de manera muy similar a `typedef` en C ++.

## Examples

Muestra la ruta de la solicitud y el tiempo que llevó procesarla.

```
//define a short alias to avoid chubby method signatures
using AppFunc = Func<IDictionary<string, object>, Task>;

class RequestTimeMiddleware
{
    private AppFunc _next;

    public RequestTimeMiddleware(AppFunc next)
    {
        _next = next;
    }

    public async Task Invoke(IDictionary<string, object> environment)
    {
        IOwinContext context = new OwinContext(environment);

        var path = context.Request.Path;
        var sw = Stopwatch.StartNew();
        //Queue up the next middleware in the pipeline
        await _next(environment);
        //When the request comes back, log the elapsed time
        Console.WriteLine($"Request for {path} processed in {sw.ElapsedMilliseconds}ms");
    }
}

public static class RequestTimeMiddlewareExtensions
{
    //Extension method as syntactic sugar, to get a meaningful way
    //in adding the middleware to the pipeline
}
```

```
public static void UseRequestTimeMiddleware(this IApplicationBuilder app)
{
    app.Use<RequestTimeMiddleware>();
}

public class Startup
{
    public void Configuration(IApplicationBuilder app)
    {
        //add the Middleware as early as possible
        app.UseRequestTimeMiddleware();
        //Queue up every other module
        app.Use(async (environment, next) =>
        {
            await environment.Response.WriteAsync("Hello from the console world");
            await next();
        });
    }
}
```

Lea Middleware en línea: <https://riptutorial.com/es/asp-net/topic/6607/middleware>

---

# Capítulo 25: Reloj de repetición

## Examples

### Uso básico

Este ejemplo crea un repetidor simple de 1 columna que muestra una lista de números, uno por elemento de repetidor.

Margen:

```
<asp:Repeater ID="Repeater1" runat="server">
  <ItemTemplate>
    <%# Container.DataItem.ToString() %>
  </ItemTemplate>
</Repeater>
```

Código detrás:

```
protected void Page_Load(object sender, EventArgs e)
{
    List<int> numbers = new List<int>{1, 2, 3, 4, 5};
    Repeater1.DataSource = numbers;
    Repeater1.DataBind();
}
```

Lea Reloj de repetición en línea: <https://riptutorial.com/es/asp-net/topic/2635/reloj-de-repeticion>

---

# Capítulo 26: ScriptManager

## Introducción

El control ScriptManager registra el script para la biblioteca de Microsoft AJAX con la página. Esto habilita las funciones de soporte de script del cliente, como la representación parcial de página y las llamadas de servicio web.

## Sintaxis

1. <asp: ScriptManager ID = "smPop" runat = "server"> </ asp: ScriptManager>
2. ScriptManager.RegisterStartupScript (Control, Type, String, String, Boolean);

## Examples

### Trabajando con ScriptManager

Debe usar un control ScriptManager en una página para habilitar las siguientes características de ASP.NET AJAX:

1. Funcionalidad de script de cliente de la biblioteca de Microsoft AJAX y cualquier script personalizado que desee enviar al navegador.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Page.ClientScript.RegisterStartupScript(
        this.GetType(), "myscript", "alert('hello world!');");
}
```

2. Representación de página parcial, que permite que las regiones de la página se actualicen de forma independiente sin devolución. Los controles ASP.NET AJAX UpdatePanel, UpdateProgress y Timer requieren un control ScriptManager para admitir el procesamiento de páginas parciales.

3. Clases proxy de JavaScript para servicios web, que le permiten usar el script del cliente para acceder a los servicios web al exponer los servicios web como objetos fuertemente tipados.

```
[WebMethod]
public int Add(int a, int b) { return a + b; }

function CallAdd()
{
    // method will return immediately
    // processing done asynchronously
    WebService.Add(0,6, OnMethodSucceeded, OnMethodFailed);
}
```

4. Clases de JavaScript para acceder a los servicios de autenticación y perfil de ASP.NET.

```
Sys.Services.AuthenticationService.login  
Sys.Services.AuthenticationService.logout  
  
<script type="text/javascript">  
    function MyMethod(username, password)  
    {  
        Sys.Services.AuthenticationService.login(username,  
            password, false, null, null, null, null, "User Context");  
    }  
</script>
```

Ver más en <https://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx>

Lea ScriptManager en línea: <https://riptutorial.com/es/asp-net/topic/10077/scriptmanager>



---

# Capítulo 27: Servicio Web sin Visual Studio

## Introducción

Un ejemplo muy básico de ASP.Net del código mínimo para crear un servicio web.

## Observaciones

En una publicación separada de Documentación de StackOverflow, analizaremos el consumo de este servicio web de calculadora.

## Examples

### Calculadora de servicios web

```
<%@ WebService Language="C#" Class="Util" %>
using System;
using System.Web.Services;

public class Util: WebService
{
    [WebMethod]
    public int CalculatorAdd(int operandA, int operandB)
    {
        return operandA + operandB;
    }

    [WebMethod]
    public int CalculatorSubtract(int operandA, int operandB)
    {
        return operandA - operandB;
    }

    [WebMethod]
    public long CalculatorMultiply(int operandA, int operandB)
    {
        return operandA * operandB;
    }

    [WebMethod]
    public long CalculatorDivide(int operandNumerator, int operandDenominator)
    {
        if (operandDenominator == 0)
            return System.Int64.MaxValue;    // Should really do better error handling overall
        & return an error
        else
            return operandNumerator / operandDenominator;
    }
}
```

Lea Servicio Web sin Visual Studio en línea: <https://riptutorial.com/es/asp-net/topic/8859/servicio-web-sin-visual-studio>

# Capítulo 28: UpdatePanel

## Introducción

Este tema describe cómo agregar soporte de actualización de página parcial a una página web utilizando dos controles de servidor Ajax de Microsoft: el control ScriptManager y el control UpdatePanel. Estos controles eliminan el requisito de actualizar toda la página con cada devolución, lo que mejora la experiencia del usuario.

## Sintaxis

- `<asp:UpdatePanel ID = "UpdatePanel1" runat = "server">`  
`</asp:UpdatePanel>`

## Observaciones

Se debe agregar un ScriptManager a la página para que funcione UpdatePanel.

## Examples

### Ejemplo de panel de actualización

Paso 1: Agrega ScriptManager a tu página

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
```

Paso 2: agregue UpdatePanel a su página justo después de ScriptManager.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate></ContentTemplate>
</asp:UpdatePanel>
```

Paso 3: Después de agregar contenido a la plantilla de contenido de UpdatePanels, su página aspx debe tener un aspecto similar al siguiente:

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <style type="text/css">
        #UpdatePanel1 {
            width:300px; height:100px;
```

```

    }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div style="padding-top: 10px">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <fieldset>
                <legend>UpdatePanel</legend>
                <asp:Label ID="Label1" runat="server" Text="Panel created."></asp:Label><br />
                <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button"
            />
            </fieldset>
        </ContentTemplate>
    </asp:UpdatePanel>
    <br />
    </div>
    </form>
</body>
</html>

```

#### Paso 4: Agrega esta parte a tu página de C #:

```

protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "Refreshed at " +
        DateTime.Now.ToString();
}

```

#### Paso 5: Ahora ejecuta tu aplicación.

#### Resultado Esperado:

El contenido del panel cambia cada vez que hace clic en el botón, pero no se actualiza toda la página. De forma predeterminada, la propiedad `ChildrenAsTriggers` de un control `UpdatePanel` es verdadera. Cuando esta propiedad se establece en `true`, los controles dentro del panel participan en las actualizaciones de páginas parciales cuando cualquier control en el panel provoca una devolución de datos.

Lea `UpdatePanel` en línea: <https://riptutorial.com/es/asp-net/topic/10075/updatepanel>

# Capítulo 29: Ver estado

## Introducción

View State es el método para preservar el valor de la página y los controles entre viajes de ida y vuelta. Es una técnica de gestión de estado a nivel de página. View State está activado de forma predeterminada y normalmente serializa los datos en todos los controles de la página, independientemente de si realmente se utilizan durante una devolución posterior.

## Sintaxis

- ViewState ["NameofViewstate"] = "Value";

## Examples

### Ejemplo

#### ASPX

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>ViewState</title>
</head>
<body>
<form id="form1" runat="server">
<asp:TextBox runat="server" id="NameField" />
<asp:Button runat="server" id="SubmitForm" onclick="SubmitForm_Click" text="Submit
& set name" />
<asp:Button runat="server" id="RefreshPage" text="Just submit" />
<br /><br />
Name retrieved from ViewState: <asp:Label runat="server" id="NameLabel" />
</form>
</body>
</html>
```

### Código detrás

```
using System;
using System.Data;
using System.Web;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (ViewState["NameOfUser"] != null)
```

```
        NameLabel.Text = ViewState["NameOfUser"].ToString();
    else
        NameLabel.Text = "Not set yet...";
}

protected void SubmitForm_Click(object sender, EventArgs e)
{
    ViewState["NameOfUser"] = NameField.Text;
    NameLabel.Text = NameField.Text;
}
}
```

Lea Ver estado en línea: <https://riptutorial.com/es/asp-net/topic/8234/ver-estado>

# Capítulo 30: Vista en cuadrícula

## Examples

### El enlace de datos

Hay dos formas de enlazar un GridView. Puede hacerlo manualmente configurando la propiedad `DataSource` y llamando a `DataBind()` , o puede usar un `DataSourceControl` como un `SqlDataSource` .

## Encuadernación manual

Crea tu GridView:

```
<asp:GridView ID="gvColors" runat="server"></asp:GridView>
```

Primero cree o recupere los datos de origen para el GridView. A continuación, asigne los datos a la propiedad `DataSource` de GridView. Finalmente, llame a `DataBind()` .

```
List<string> colors = new List<string>();
colors.Add("Red");
colors.Add("Green");
colors.Add("Blue");

gvColors.DataSource = colors;
gvColors.DataBind();
```

## DataSourceControl

Crea tu DataSourceControl:

```
<asp:SqlDataSource ID="sdsColors"
    runat="server"
    ConnectionString="<%%$ MyConnectionString %>"
    SelectCommand="SELECT Color_Name FROM Colors">
</asp:SqlDataSource>
```

Crea tu GridView y establece la propiedad `DataSourceID` :

```
<asp:GridView ID="gvColors"
    runat="server"
    DataSourceID="sdsColors">
</asp:GridView>
```

## Columnas

Hay siete tipos de columnas diferentes que pueden usarse dentro de un GridView.

```
<asp:GridView ID="GridView1" runat="server">
    <Columns>
        ...
    </Columns>
</asp:GridView>
```

## BoundField:

```
<asp:BoundField DataField="EmployeeID" HeaderText="Employee ID" />
```

## ButtonField:

```
<asp:ButtonField ButtonType="Button" HeaderText="Select Employee" Text="Select"/>
```

## CheckBoxField:

```
<asp:CheckBoxField DataField="IsActive" HeaderText="Is Active" />
```

## CommandField:

```
<asp:CommandField ShowDeleteButton="true"
    ShowEditButton="true"
    ShowInsertButton="true"
    ShowSelectButton="true" />
```

## HyperLinkField:

```
<asp:HyperLinkField HeaderText="Employee Profile"
    DataNavigateUrlFields="EmployeeID"
    DataNavigateUrlFormatString="EmployeeProfile.aspx?EmployeeID={0}" />
```

## ImageField:

```
<asp:ImageField HeaderText="Photo"
    DataImageUrlField="EmployeeID"
    DataImageUrlFormatString="/images/{0}" />
```

## TemplateField:

```
<asp:TemplateField>
    <HeaderTemplate>
        Name
    </HeaderTemplate>
    <ItemTemplate>
        <asp:Label ID="lblEmployeeName"
            runat="server"
            Text='<&# Eval("EmployeeName") %>'></asp:Label>
    </ItemTemplate>
</asp:TemplateField>
```

## GridView fuertemente tipado

Comenzando con Asp.net 4.5, los controles web pueden aprovechar el enlace fuertemente tipado para obtener soporte de IntelliSense y errores de tiempo de compilación.

Crea una clase que contenga tu modelo:

```
public class Album
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Artist { get; set; }
}
```

Defina el control GridView en su página:

```
<asp:GridView ID="Grid" runat="server" AutoGenerateColumns="false"
ItemType="YourNamespace.Album">
    <Columns>
        <asp:TemplateField HeaderText="Id">
            <ItemTemplate>
                <asp:Label ID="lblName" runat="server" Text="<%# Item.Id %>"></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Name">
            <ItemTemplate>
                <asp:Label ID="lblName" runat="server" Text="<%# Item.Name %>"></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Artist">
            <ItemTemplate>
                <asp:Label ID="lblCity" runat="server" Text="<%# Item.Artist %>"></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

Cargar los datos y enlazarlos:

```
var albumList = new List<Album>
{
    new Album {Id = 1, Artist = "Icing (a Cake cover band)", Name = "Toppings Vol. 1"},
    new Album {Id = 2, Artist = "Fleetwood PC", Name = "Best of Windows"},
    new Album {Id = 3, Artist = "this.Bandnames", Name = "TH_ (Pronounced \"Thunderscore\")"},
};

Grid.DataSource = albumList;
Grid.DataBind();
```

## Manejo de evento de comando

GridViews permite enviar comandos desde una fila de GridView. Esto es útil para pasar información específica de una fila a un controlador de eventos como argumentos de comando.

Para suscribirse a un evento de comando:



```
<asp:GridView ID="GridView1" ... OnRowCommand="GridView1_RowCommand">
```

Los botones son la forma más común de elevar comandos. También admiten una forma de especificar argumentos de comando. En este ejemplo, el argumento es un `ID` del elemento que representa la fila.

```
<TemplateField>
  <ItemTemplate>
    <asp:LinkButton ID="LinkButton1" runat="server"
      CommandName="SampleCmd"
      CommandArgument='<%# Eval("ID") %>'>
    </asp:LinkButton>
  </ItemTemplate>
</TemplateField>
```

Alternativamente, uno puede usar una plantilla de columna `CommandField` que proporciona los controles de comando más comunes.

Manejo del evento en código detrás:

```
protected void GridView1_RowCommand(object source, GridViewCommandEventArgs e)
{
    if (e.CommandName == "SampleCmd")
    {
        var id = e.CommandArgument;
    }
}
```

Tenga en cuenta que el `CommandName` utilizado en este ejemplo es arbitrario y es una opción del desarrollador. Sin embargo, hay un conjunto de nombres predefinidos que el propio `GridView` reconoce. Los eventos correspondientes se generan cuando se activan estos comandos.

Nombre del comando	Eventos planteados
Cancelar	RowCancelingEdit
Borrar	RowDeleting, RowDeleted
Editar	RowEditing
Página	PageIndexChanging, PageIndexChanged
Seleccionar	SelectedIndexChanging, SelectedIndexChanged
Ordenar	Clasificación, Clasificación
Actualizar	RowUpdating, RowUpdated

## Paginacion

# ObjectDataSource

Si usas un ObjectDataSource, casi todo se maneja por ti, simplemente dile a GridView a AllowPaging y dale un PageSize .

```
<asp:GridView ID="gvColors"
    runat="server"
    DataSourceID="sdsColors"
    AllowPaging="True"
    PageSize="5">
</asp:GridView>

<asp:SqlDataSource ID="sdsColors"
    runat="server"
    ConnectionString="<%= MyConnectionString %>"
    SelectCommand="SELECT Color_ID, Color_Name FROM Colors">
</asp:SqlDataSource>
```

Color_ID	Color_Name	Color_ID	Color_Name	Color_ID	Color_Name
1	Red	6	Orange	11	Pink
2	Blue	7	Black	12	Turquoise
3	Green	8	White	13	Maroon
4	Yellow	9	Gray		
5	Purple	10	Brown		
123		123		123	

## Encuadernación manual

Si se vincula manualmente, debe manejar el evento PageIndexChanging . Simplemente configure DataSource y PageIndex y vuelva a enlazar el GridView.

```
<asp:GridView ID="gvColors"
    runat="server"
    AllowPaging="True"
    PageSize="5"
    OnPageIndexChanging="gvColors_PageIndexChanging">
</asp:GridView>
```

DO#

```
protected void gvColors_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvColors.DataSource = // Method to retrieve DataSource
    gvColors.PageIndex = e.NewPageIndex;
    gvColors.DataBind();
}
```

VB.NET

```
Protected Sub gvColors_PageIndexChanging(sender As Object, e As GridViewPageEventArgs)
{
```

```

gvColors.DataSource = // Method to retrieve DataSource
gvColors.PageIndex = e.NewPageIndex
gvColors.DataBind()
}

```

## Actualizar Gridview en la fila de clic

Las vistas en cuadrícula son más útiles si podemos actualizar la vista según nuestras necesidades. Considere una vista con una función de bloqueo / desbloqueo en cada fila. Se puede hacer como:

Añadir un panel de actualización:

```

<asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode="Conditional"> </asp:UpdatePanel>

```

Agregue un ContentTemplate y un Trigger dentro de su UpdatePanel:

```

<asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
    </ContentTemplate>

    <Triggers>
    </Triggers>
</asp:UpdatePanel>

```

Agregue su GridView dentro de ContentTemplate:

```

<ContentTemplate>
<asp:GridView ID="GridView1" runat="server">
    <Columns>
        <asp:TemplateField>
            <ItemTemplate>
                <asp:ImageButton ID="imgDownload" runat="server" OnClientClick="return
confirm('Are you sure want to Lock/Unlock ?');"
                CommandName="togglelock"
                CommandArgument='<%#Container.DataItemIndex%>' />

            </ItemTemplate>
        </asp:TemplateField>
    </Columns>

</ContentTemplate>

```

Aquí estamos dando a nuestra columna constante GridView1, para el botón de bloqueo. Eso sí, el databind no ha tenido lugar hasta ahora.

Tiempo para DataBind: (en PageLoad)

```

using (SqlConnection con= new SqlConnection(connectionString))
{
    SqlCommand sqlCommand = new SqlCommand(" ... ", con);
    SqlDataReader reader = sqlCommand.ExecuteReader();
    GridView1.DataSource = reader;
}

```

```

        GridView1.DataBind();
    }

```

La imagen de bloqueo / desbloqueo será diferente según el valor de una determinada columna en su GridView. Considere un caso en el que su tabla contenga un atributo / columna titulado "Estado de bloqueo". Ahora desea (1) ocultar esa columna justo después de DataBind y justo antes de la representación de la página y (2) asignar diferentes imágenes a cada fila en base a ese valor de columna oculto, es decir, si el estado de bloqueo de una fila es 0, asígnele el "bloqueo". jpg ", si el estado es 1, asignalo "unlock.jpg ". Para hacer esto, OnRowDataBound opción OnRowDataBound de GridView, se mezcla con su GridView, justo antes de representar cada fila a la página HTML.

```

<ContentTemplate>
<asp:GridView ID="GridView1" runat="server" OnRowDataBound="GridView1_RowDataBound"> ...

```

En archivo cs

```

protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Cells[8].Visible = false; //hiding the desired column which is column number
6 in this case
        GridView1.HeaderRow.Cells[8].Visible = false; //hiding its header
        ImageButton imgDownload = (ImageButton)e.Row.FindControl("imgDownload");
        string lstate = ((CheckBox)e.Row.Cells[8].Controls[0]).Checked.ToString();
        if (lstate == "True")
        { imgDownload.ImageUrl = "images/lock.png"; }
        else
        {
            imgDownload.ImageUrl = "images/unlock.png";
        }
    }
}

```

Ahora el GridView se procesará como queramos, ahora implementemos eventos de clic de botón en el botón Bloquear / Desbloquear imagen. Comprenda que para realizar una operación específica en una fila específica, se debe dar un comando a esa fila y GridView nos proporciona la misma funcionalidad llamada OnRowCommand .

```

<ContentTemplate>
<asp:GridView ID="GridView1" runat="server" OnRowDataBound="GridView1_RowDataBound"
OnRowCommand="GridView1_RowCommand">
...
</ContentTemplate>

```

e.CommandArgument una función en el archivo cs que toma un object sender y GridViewCommandEventArgs e Con e.CommandArgument podemos obtener el índice de la fila que dio el comando Punto que se debe señalar aquí es que, una fila puede tener varios botones y el cs el código necesita saber qué botón de esa fila dio el comando. Así que usaremos CommandName

```
<asp:ImageButton ID="imgDownload" runat="server" OnClientClick="return confirm('Are you sure  
want to Lock/Unlock ?');"  
CommandName="togglelock"  
CommandArgument='<%#Container.DataItemIndex%>' />
```

Ahora en el backend uno puede distinguir comandos de diferentes filas y diferentes botones.

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)  
{  
    if (e.CommandName == "togglelock")  
    {  
        using (SqlConnection con= new SqlConnection(connectionString))  
        {  
            int index = Convert.ToInt32(e.CommandArgument);  
            SqlCommand sqlCommand = new SqlCommand(" ... ", con);  
            SqlDataReader reader = sqlCommand.ExecuteReader();  
            GridView1.DataSource = reader;  
            GridView1.DataBind();  
        }  
    }  
}
```

Agregue `<asp:PostBackTrigger ControlID="GridView1"/>` al `Trigger` y actualizará el `GridView` una vez que se complete el `DataBind`.

Use `HorizontalAlign="Center"` para colocar `GridView` en el centro de la página.

Lea Vista en cuadrícula en línea: <https://riptutorial.com/es/asp-net/topic/1680/vista-en-cuadrícula>

---

# Capítulo 31: web.config> system.webServer / httpErrors & system.web / customErrors en las secciones

## Introducción

CustomErrors es un elemento heredado (compatible con versiones anteriores), utilizado por Visual Studio Development Server (también conocido como VSDS o Cassini).

httpErrors es el nuevo elemento que solo usa IIS7.

## Examples

### ¿Cuál es la diferencia entre customErrors y httpErrors?

**Ambos se utilizan para definir el manejo de errores para un sitio web, pero un software diferente se refiere a diferentes elementos de configuración.**

customErrors es un elemento heredado (compatible con versiones anteriores), utilizado por Visual Studio Development Server (también conocido como VSDS o Cassini).

httpErrors es el nuevo elemento que solo usa IIS7.

Esto resalta el posible problema al desarrollar sitios web ASP.NET al usar VSDS en lugar del IIS local.

También, [consulte esta publicación](#) de mi mismo sobre cómo manejar los mensajes de error con IIS7, si desea tener un control total de la salida de error.

Resumen:

1. Desarrollando en VSDS - use customErrors
2. Publicación del sitio en IIS6 - use customErrors
3. Publicación del sitio en IIS7 - use httpErrors.
4. y si desarrolla con VSDS pero publica en IIS7, supongo que necesitará ambos.

Lea [web.config> system.webServer / httpErrors & system.web / customErrors en las secciones en línea](#): <https://riptutorial.com/es/asp-net/topic/10103/web-config-gt--system-webserver---httperrors--amp--system-web---customerrors-en-las-secciones>

# Creditos

S. No	Capítulos	Contributors
1	Empezando con ASP.NET	<a href="#">Ahmed Abdelhameed</a> , <a href="#">Aristos</a> , <a href="#">Community</a> , <a href="#">demonplus</a> , <a href="#">Dillie-O</a> , <a href="#">Josh E</a> , <a href="#">khawarPK</a> , <a href="#">Marco</a> , <a href="#">Matt</a> , <a href="#">Muhammad Awais</a> , <a href="#">Satinder singh</a> , <a href="#">wintersolider</a>
2	Asp Web Forms Identity	<a href="#">tatigo</a>
3	ASP.NET - Controles básicos	<a href="#">khawarPK</a>
4	ASP.NET - Controles de usuario	<a href="#">Tetsuya Yamamoto</a>
5	ASP.NET - Estado de gestión	<a href="#">khawarPK</a>
6	ASP.NET - Validadores	<a href="#">khawarPK</a>
7	Asp.net Ajax Controls	<a href="#">Saurabh Srivastava</a>
8	ASP.NET Caching	<a href="#">tatigo</a>
9	Ciclo de vida de la página	<a href="#">Abdul</a> , <a href="#">mbenegas</a> , <a href="#">Srikar</a> , <a href="#">VDWWD</a>
10	DayPilot Scheduler	<a href="#">Abdul</a>
11	Delegación de eventos	<a href="#">Webruster</a>
12	Directivas	<a href="#">khawarPK</a> , <a href="#">Tot Zam</a>
13	El enlace de datos	<a href="#">j.f.</a> , <a href="#">Ryan</a>
14	Encuentra Control por ID	<a href="#">Andrei</a> , <a href="#">VDWWD</a> , <a href="#">Webruster</a>
15	Estado de sesión	<a href="#">Luke Ryan</a> , <a href="#">Naveen Gogineni</a> , <a href="#">Nisarg Shah</a>
16	Expresiones	<a href="#">Ryan</a>

17	Formas Web	<a href="#">Big Fan</a> , <a href="#">jignesh</a> , <a href="#">naveen</a> , <a href="#">Tot Zam</a>
18	Gestion de Sesiones	<a href="#">Jasmin Solanki</a>
19	httpHandlers	<a href="#">Taylor Brown</a>
20	Katana	<a href="#">jignesh</a>
21	Lista de datos	<a href="#">Webruster</a>
22	Manejo de eventos	<a href="#">khawarPK</a> , <a href="#">Tot Zam</a>
23	Métodos de página	<a href="#">Enrique Zavaleta</a> , <a href="#">wazz</a> , <a href="#">XIII</a>
24	Middleware	<a href="#">Marco</a>
25	Reloj de repetición	<a href="#">Andrei</a>
26	ScriptManager	<a href="#">Naveen Gogineni</a>
27	Servicio Web sin Visual Studio	<a href="#">George 2.0 Hope</a>
28	UpdatePanel	<a href="#">Naveen Gogineni</a>
29	Ver estado	<a href="#">jignesh</a>
30	Vista en cuadrícula	<a href="#">Andrei</a> , <a href="#">Asif.Ali</a> , <a href="#">j.f.</a> , <a href="#">Marco</a> , <a href="#">Ritwik</a>
31	web.config> system.webServer / httpErrors & system.web / customErrors en las secciones	<a href="#">Naveen Gogineni</a>