3. PÁGINAS WEB ASP.NET (PÁGINAS WEB DE SERVIDOR).

Las páginas Web ASP.NET se utilizan como la interfaz de usuario programable para su aplicación Web. Este tipo de páginas presenta la información al usuario en cualquier explorador o dispositivo cliente e implementa lógica de aplicación mediante el código de la parte servidor.

Las páginas Web ASP.NET:

- Se basan en la tecnología Microsoft ASP.NET, en la que el código que se ejecuta en el servidor genera de forma dinámica salida de páginas Web en un explorador o dispositivo cliente.

- Son compatibles con cualquier explorador o dispositivo móvil. Las páginas Web ASP.NET representan automáticamente el código HTML adecuado al explorador para funciones tales como estilos, diseño, etc. Como alternativa, se pueden diseñar las páginas Web ASP.NET para ejecutarse en un explorador determinado, como Microsoft Internet Explorer 6 y aprovechar así todas las características de un explorador específico.

- Admiten cualquier lenguaje compatible con Common Language Runtime de .NET, incluidos Microsoft Visual Basic, Microsoft Visual C#, Microsoft J# y Microsoft JScript.NET.

- Se crean en el entorno Microsoft .NET Framework. Esto proporciona todos los beneficios del marco de trabajo, incluidos un entorno administrado, seguridad de tipos y herencia.

- Son flexibles gracias a la posibilidad de incorporar a ellas controles creados por los usuarios y de otros fabricantes.

COMPONENTES DE PÁGINAS WEB ASP.NET

En las páginas Web ASP.NET, la programación de la interfaz de usuario se divide en dos partes: el componente visual y el lógico.

El **elemento visual** está compuesto por un archivo que contiene el marcado estático como HTML o controles de servidor ASP.NET o ambos. La página Web ASP.NET funciona como un contenedor del texto y los controles estáticos que se desea mostrar.

La **lógica de las páginas** Web ASP.NET se compone del código creado para interactuar con la página. El código puede residir en un bloque de script en la página o en una clase independiente. Si el código está en un archivo de clase independiente, a este archivo se le conoce como archivo de código subyacente. El código del archivo de código subyacente se puede escribir en Visual Basic, Visual C#, Visual J# o JScript .NET.

Las páginas Web ASP.NET se compilan en un archivo de biblioteca de vínculos dinámicos (.dll). La primera vez que un usuario examina la página .aspx con el explorador, ASP.NET genera automáticamente un archivo de clase .NET que representa a la página y la compila. El archivo .dll se ejecuta en el servidor y genera dinámicamente la salida HTML para su página.

MODO DE FUNCIONAMIENTO DE PÁGINAS WEB ASP.NET

Las páginas Web de ASP.NET le permiten crear contenido dinámico para su sitio Web. Con una página HTML estática (archivo .htm o .html), el servidor cumple una solicitud Web leyendo el archivo y enviándolo tal como está al explorador. Al contrario, cuando alguien solicita una página Web de ASP.NET (archivo .aspx), la página se ejecuta como un programa en el servidor Web. Mientras la página se está ejecutando, puede realizar cualquier tarea que requiera su sitio Web, incluido el cálculo de valores, la lectura o escritura de información de base de datos o la llamada a otros programas. Como su resultado, la página genera dinámicamente marcado (elementos en HTML u otro lenguaje de marcado) y envía este resultado dinámico al explorador.

Devoluciones de datos y recorridos de ida y vuelta

Las páginas ASP.NET se ejecutan como código en el servidor. Por consiguiente, para procesar la página, ésta se configura para que se envíe al servidor cuando los usuarios hagan clic en botones (u opcionalmente, cuando los usuarios activen casillas o interactúen con otros controles de la página). Una y otra vez la página se envía a sí misma para poder ejecutar su código de servidor y después representar una nueva versión de sí misma al usuario.

El ciclo de procesamiento de una página Web de ASP.NET es este:

1. El usuario solicita la página. (La página se solicita utilizando un método GET de HTTP.) La página se ejecuta por primera vez, realizando el procesamiento preliminar si la ha programado para hacerlo.
2. La página representa el marcado en el explorador dinámicamente y lo que el usuario ve es una página Web similar a cualquier otra.
3. El usuario escribe la información o la selecciona entre las opciones disponibles y, a continuación, hace clic en un botón. (Si los usuarios hacen clic en un vínculo en lugar de en un botón, la página podría simplemente navegar hasta otra página, sin que tenga lugar ningún procesamiento más en la primera página).
4. La página se manda al servidor Web. (El explorador ejecuta un método POST de HTTP, que en ASP.NET se denomina devolución de datos.) Específicamente, la página se devuelve datos a sí misma. Por ejemplo, si el usuario está trabajando con la página Default.aspx y hace clic en un botón de la página, la devuelve al servidor con un destino de Default.aspx.
5. En el servidor Web, la página se ejecuta de nuevo. La información que el usuario escribió o seleccionó está disponible para la página.
6. La página realiza el procesamiento que tiene programado hacer.
7. La página se representa a sí misma en el explorador.

Este ciclo continúa durante el tiempo que el usuario esté trabajando en la página. Cada vez el usuario hace clic en un botón, la información de la página se manda al servidor Web y la página se ejecuta de nuevo. Cada ciclo se conoce como acción de ida y vuelta. Dado que el procesamiento de páginas se realiza en el servidor Web, cada acción que la página puede hacer requiere un recorrido de ida y vuelta al servidor.

> **Nota:**
> Una página Web de ASP.NET puede ejecutar script de cliente, lo cual no requiere un recorrido de ida y vuelta al servidor y resulta útil para la validación de la entrada del usuario y para algunos tipos de programación de la interfaz de usuario.

Envío entre páginas

Bajo algunas circunstancias, podría desear que una página se mande a otra página diferente, no a sí misma. Esto se denomina envío entre páginas. Por ejemplo, podría estar creando una serie de páginas que procese un orden del cliente. Cada página se puede mandar a la página siguiente de la secuencia.

Período de duración de la página

Al contrario que los formularios de las aplicaciones de sobremesa, una página Web de ASP.NET no se inicia, se ejecuta mientras el usuario trabaja con el formulario y, a continuación, sólo se descarga cuando el usuario hace clic en un botón Cerrar. Esto ocurre porque el Web está desconectado inherentemente. Cuando un explorador solicita una página de un servidor Web, el explorador y el servidor sólo se conectan el tiempo suficiente para procesar la solicitud. Después de que el servidor Web ha representado una página en el explorador, finaliza la conexión. Si el explorador realiza otra solicitud al mismo servidor Web, incluso para la misma página, esta solicitud se procesa como una nueva solicitud.

La naturaleza desconectada del Web dicta la manera en la que se ejecuta la página ASP.NET. Cuando un usuario solicita una página Web ASP.NET, se crea una nueva instancia de la página. La página realiza su procesamiento, representa el marcado en el explorador y se descarta a continuación. Si el usuario hace clic en un botón para realizar una devolución de datos, se crea una nueva instancia de la página, ésta realiza su procesamiento y se descarta de nuevo. Así, cada devolución de datos y cada recorrido de ida y vuelta produce una nueva instancia de la página.

Conservar el estado de la página

En el protocolo HTTP estándar, la única información que el servidor posee de una página es la que el usuario ha especificado mediante los controles de la misma, porque el explorador envía al servidor únicamente esa información cuando se envía la página. Cualquier otro tipo de información, como los valores de las variables y la configuración de las propiedades, se elimina. ASP.NET ayuda a conservar otra información de la página de las maneras siguientes:

- ASP.NET guarda la configuración del control (sus propiedades) entre las acciones de ida y vuelta, lo que recibe el nombre de estado del control.

- ASP.NET proporciona funciones de administración del estado que permiten guardar las variables personalizadas e información específica de la aplicación o de la sesión entre cada acción de ida y vuelta.

- ASP.NET puede detectar cuándo se solicita una página por primera vez y cuándo se envía, lo que permite programar de la manera conveniente. Por ejemplo, podría desear leer información de una base de datos la primera vez se muestre una página, pero no en cada devolución de datos.

  Nota:

  Se puede configurar el servidor para que almacene la información de la página en la memoria caché y así optimizar las páginas, pero para el objetivo de la programación de aplicaciones, es mejor pensar que las páginas se eliminan tan pronto como el servidor ha terminado de procesarlas.

PROGRAMACIÓN DE PÁGINAS WEB ASP.NET

El código de servidor de las páginas Web ASP.NET se puede crear con una serie de lenguajes en .NET Framework, incluidos Visual Basic, C# y J#. Las páginas Web de ASP.NET pueden contener script de cliente que se ejecuta dentro del explorador. Algunas funciones ASP.NET generan script de cliente y lo insertan en la página. En ese caso, ASP.NET siempre genera ECMAScript (JavaScript) para obtener una mejor funcionalidad entre los distintos exploradores. Además, puede agregar su propio script de cliente para conseguir una funcionalidad personalizada. Si hace, puede utilizar cualquier lenguaje de script de cliente que sea compatible con los exploradores que desee utilizar.

Controles de servidor

Como cualquier página Web, las páginas Web de ASP.NET pueden contener texto estático. Sin embargo, normalmente lo que hará será agregar controles a la página, por ejemplo cuadros de texto, casillas y botones. Estos controles permiten al usuario interactuar con la página y enviar información al servidor cuando ésta se devuelve.

ASP.NET proporciona una colección de controles que se conocen como controles de servidor Web. Los controles de servidor ASP.NET pueden ser similares a los elementos de formato HTML correspondientes. Por ejemplo, el control ASP.NET TextBox es similar a una etiqueta HTML <input type="text">. Sin embargo, los controles de servidor ASP.NET proporcionan una experiencia de programación más rica que los elementos HTML. Hay también controles de servidor ASP.NET para una gama de funciones mucho más amplia que la que proporcionan los elementos HTML. Entre los controles de servidor que puede utilizar en una página Web de ASP.NET existe un control de calendario, controles enlazados a datos que muestran listas o cuadrículas, un control de inicio de sesión para agregar seguridad a su sitio y mucho más.

Eventos de controles de página y de servidor

Una característica importante de ASP.NET es que permite programar páginas Web utilizando un **modelo basado en eventos** similar al de las aplicaciones de cliente. Como ejemplo sencillo, se puede agregar un botón a una página Web ASP.NET y, a continuación, escribir un controlador de eventos para el evento de clic del botón. Aunque esto es habitual en páginas Web que funcionan exclusivamente con una secuencia de comandos de cliente (que controla el evento onclick del botón en HTML dinámico), ASP.NET traslada este modelo al procesamiento basado en servidor.

Incluso la propia página provoca eventos de ciclo de vida cuando se inicializa, eventos tales como Page_Init y Page_Load, lo que permite ejecutar código cuando la página se inicia. (Recuerde que la página se crea y reinicializa con cada recorrido de ida y vuelta).

Por tanto, buena parte de la lógica de la página implica responder a los eventos iniciados por los controles.

Los eventos producidos por los controles de servidor ASP.NET funcionan de manera diferente a los eventos de las páginas HTML tradicionales o de las aplicaciones Web basadas en el cliente. La diferencia se basa principalmente en la *separación existente entre el propio evento y el lugar donde se controla el evento*. En las aplicaciones basadas en cliente, los eventos se producen y controlan en el cliente. Sin embargo, en las páginas Web ASP.NET, los eventos asociados a los controles de servidor se originan en el cliente (explorador) pero los controla la página ASP.NET en el servidor Web.

El modelo de control de eventos Web ASP.NET necesita que la información del evento se capture en el cliente y que se transmita un mensaje de evento al servidor mediante un envío HTTP. La página debe interpretar el envío para determinar el evento ocurrido y, a continuación, llamar al método apropiado del código del servidor para controlar dicho evento.

ASP.NET controla la tarea de capturar, transmitir e interpretar el evento. Al crear controladores de eventos en una página Web ASP.NET, no es necesario saber capturar la información del evento y hacer que esté disponible para el código.

La mayoría de los controles de servidor de ASP.NET sólo admiten unos pocos eventos que el usuario puede controlar en el código del servidor. Para procesar un evento, la página debe realizar una acción de ida y vuelta de manera que la elección del usuario se pueda enviar a la página para su procesamiento. Los controles de servidor no exponen eventos que se producen con mucha frecuencia, como onmouseover, porque cada vez que se provoca un evento, se produce una nueva acción de ida y vuelta en el servidor, lo que afecta considerablemente al tiempo de respuesta de la página. Sin embargo, los controles de servidor ASP.NET se pueden configurar para provocar eventos, como onmouseover, en el cliente. En ese caso, los controles no se devuelven al servidor y el usuario crea el script de cliente para responder a los eventos.

Compatibilidad del explorador

Dado que el procesamiento de una página Web de ASP.NET se realiza en el servidor Web, las páginas Web de ASP.NET son compatibles con cualquier explorador o dispositivo móvil. Una página Web representa automáticamente el marcado compatible con el explorador (XHTML u otro lenguaje de marcado) para características como estilos y diseño. Alternativamente, puede crear páginas Web con controles diseñados específicamente para representar el resultado de unos dispositivos concretos, como los teléfonos móviles.

SINTAXIS DE PÁGINAS WEB ASP.NET.

Aunque las páginas Web ASP.NET se crean de forma similar a las páginas Web HTML estáticas (páginas que no contienen procesamiento basado en servidor), incluyen elementos adicionales que ASP.NET reconoce y procesa cuando se ejecuta la página. Las características que distinguen las páginas Web ASP.NET de las páginas HTML estáticas (u otras) son las siguientes:

- La extensión de nombre de archivo **.aspx** en lugar de .htm, .html u otras extensiones. La extensión de nombre de archivo .aspx hace que ASP.NET procese la página.

  Nota:

  La asignación de extensiones de nombre de archivo a ASP.NET se realiza en Internet Information Services (IIS). De forma predeterminada, ASP.NET ejecuta las páginas .aspx, no las páginas .htm y .html.

- Una directiva **@ Page** u otra directiva opcional, según convenga para el tipo de página que se está creando.

- Un elemento **form** que está configurado correctamente para ASP.NET. El elemento form sólo es necesario si la página contiene controles cuyos valores se deben utilizar durante el procesamiento de páginas.

- Controles de servidor Web.

- Código del servidor si agrega su propio código a la página.

  Nota:

  Si desea que las páginas cumplan los estándares XHTML, deberá incluir elementos adicionales, como el elemento DOCTYPE.

Si lo desea, puede cambiar el nombre de las páginas HTML y asignarles la extensión de nombre de archivo .aspx para que se ejecuten como páginas Web ASP.NET. No obstante, si la página no requiere procesamiento en el servidor, no es necesario que tenga la extensión .aspx, ya que al hacerlo se sobrecarga el procesamiento de la página.

Elementos de Formulario

Si la página incluye controles que permiten a los usuarios interactuar con la página y enviarla, ésta debe incluir un elemento **form**. Aunque se utiliza el elemento form de HTML estándar, se deben cumplir ciertas reglas. Las reglas para utilizar el elemento form son las siguientes:

- La página sólo puede contener un elemento form.
- El elemento form debe contener el atributo runat establecido con el valor server. Este atributo permite hacer referencia al formulario y los controles de la página mediante programación en código del servidor.
- Los controles de servidor que pueden realizar devolución de datos deben estar dentro del elemento form.
- La etiqueta de apertura no debe contener ningún atributo action. ASP.NET establece estos atributos dinámicamente cuando se procesa la página y reemplaza cualquier configuración que se pueda establecer.

Controles de Servidor Web

En la mayoría de las páginas ASP.NET deberá agregar controles que permitan al usuario interactuar con la página, como botones, cuadros de texto, listas, etc. Estos controles de servidor Web son similares a los botones y los elementos input de HTML. Sin embargo, éstos se procesan en el servidor, lo que permite utilizar el código del servidor para establecer sus propiedades. Estos controles también provocan eventos que se pueden controlar en el código del servidor.

Los controles de servidor utilizan una sintaxis especial que ASP.NET reconoce cuando se ejecuta la página. En el ejemplo de código siguiente se muestran algunos de los controles de servidor Web que se emplean típicamente.

> **Nota de seguridad:**
>
> TextBox acepta los datos proporcionados por el usuario, lo que puede suponer una amenaza para la seguridad. De forma predeterminada, las páginas Web ASP.NET validan los datos escritos por el usuario para comprobar que no incluyen scripts ni elementos HTML.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Click" OnClick="Button1_Click" />
```

El nombre de la etiqueta de los controles de servidor ASP.NET comienza con un prefijo, en este caso, **asp**:. El prefijo podría ser diferente si el control no forma parte de .NET Framework. Los controles de servidor ASP.NET también incluyen el atributo runat="server" y, opcionalmente, un identificador cuyo uso permite hacer referencia al control en el código del servidor.

Cuando se ejecuta la página, ésta identifica los controles de servidor y ejecuta el código que está asociado a dichos controles. Muchos controles representan código HTML u otro formato en la página. Por ejemplo, el control asp:textbox representa un elemento input con el atributo type="text" en una página. Sin embargo, no hay necesariamente una asignación uno a uno entre un control de servidor Web y un elemento HTML. Por ejemplo, el control asp:calendar representa una tabla HTML. Algunos controles no representan nada en el explorador; en lugar de ello, sólo se procesan en el servidor y proporcionan información a otros controles.

Elementos HTML como controles de servidor

En lugar, o además, de utilizar controles de servidor ASP.NET, puede utilizar elementos HTML ordinarios como controles de servidor. Puede agregar el atributo runat="server" y un atributo ID a cualquier elemento HTML de la página. Cuando se ejecuta la página, ASP.NET identifica el elemento como control de servidor y lo pone a disposición del código del servidor. Por ejemplo, puede agregar los elementos necesarios a un elemento body de HTML, tal como se muestra en el ejemplo de código siguiente.

```
<body runat="server" id="body">
```

A continuación puede hacer referencia al elemento body en el código del servidor; por ejemplo, para establecer el color de fondo del elemento body en tiempo de ejecución en respuesta a los datos proporcionados por el usuario o la información proveniente de una base de datos.

Código del servidor

La mayoría de las páginas ASP.NET incluyen código que se ejecuta en el servidor cuando se procesa la página. ASP.NET admite un gran número de lenguajes, entre los que se encuentran C#, Visual Basic, J#, Jscript y otros.

ASP.NET admite dos modelos a la hora de escribir el código del servidor para una página Web. En el modelo de un solo archivo, el código de la página se encuentra en un elemento script cuya etiqueta de apertura incluye el atributo runat="server".

Opcionalmente, puede crear el código para la página en un archivo de clase independiente al que se hace referencia como modelo de código subyacente. En este caso, la página Web ASP.NET generalmente no contiene código del servidor. En su lugar, la directiva @Page incluye información que vincula la página .aspx con su archivo de código subyacente asociado. En el ejemplo de código siguiente se muestra una directiva @ Page típica para una página con un archivo de código subyacente.

```
<%@ Page Language="VB" CodeFile="Default.aspx.vb" Inherits="Default" %>
```

El atributo CodeFile especifica el nombre del archivo de clase independiente y el atributo Inherits especifica el nombre de la clase del archivo de código subyacente que corresponde a la página.

Nota:

Las páginas Web ASP.NET también pueden incluir scripts de cliente que se ejecutan en el explorador como respuesta a los eventos del cliente. Una página ASP.NET puede incluir scripts de cliente y código del servidor.

MODELO DE CÓDIGO DE LAS PÁGINAS WEB ASP.NET.

Una página Web ASP.NET se compone de dos partes:

- Elementos visuales, incluidos el formato, los controles de servidor y el texto estático.
- Lógica de programación para la página, que incluye controladores de eventos y otro tipo de código y que el servidor ejecuta durante la fase de representación de la página.

ASP.NET proporciona dos modelos para administrar el código y los elementos visuales: el modelo de página un solo archivo y el modelo de página de código subyacente. Los dos modelos funcionan de la misma manera y se pueden utilizar los mismos controles y el mismo código para ambos modelos.

Modelo de página de un solo archivo

En este modelo de página, el formato de la página y su código de programación están el mismo archivo .aspx físico.

El código de programación se encuentra incrustado en línea y mezclado junto con el marcado de la página o en un bloque de script que contiene el atributo runat="server", lo que lo identifica como código que debe ejecutar ASP.NET.

El código en línea debe incluirse entre los delimitadores **<% %>** y puede utilizarse la sintaxis **<% = expression %>** para resolver una expresión y devolver su valor al bloque.

```
<%@ Page Language="VB" %>
<html>
<body>
    <form id="form1" runat="server">
    <% For i As Integer = 0 To 5 %>
         <% Response.Write("<br>" & i.ToString())%>
    <% Next%>
    </form>
</body>
</html>
```

```
<%@ Page Language="VB" %>
<script runat=server>
Protected Function GetTime() As String
    Return DateTime.Now.ToString("t")
End Function
</script>
<html>
<body>
    <form id="form1" runat="server">
       Current server time is <% =GetTime()%>.
    </form>
</body>
</html>
```

Los bloques de código incrustados se deben escribir en el lenguaje predeterminado de la página. Por ejemplo, si la directiva @Page de la página contiene el atributo language="VB", la página utilizará el compilador de Visual Basic para compilar el código de los bloques de secuencias de comandos marcados con runat="server" y el código en línea incluido entre los delimitadores <% %>.

Modelo de página de código subyacente

Este modelo permite mantener el formato en un archivo (el archivo .aspx) y el código de programación en otro. El nombre del archivo de código varía según el lenguaje de programación que se esté utilizando.

Nota:

No todos los lenguajes de programación de .NET permiten crear archivos de código subyacente para las páginas Web ASP.NET. Los lenguajes deben admitir clases parciales. Por ejemplo, J# no admite las clases parciales y, por consiguiente, no permite la creación de archivos de código subyacente para las páginas ASP.NET.

Por ejemplo, si está trabajando con una página denominada SamplePage, el formato se encuentra en el archivo SamplePage.aspx y el código se encuentra en un archivo denominado SamplePage.aspx.vb (en Visual Basic), SamplePage.aspx.cs (en C#), etc.

Hay dos diferencias entre la página .aspx del modelo de un solo archivo y la del modelo de código subyacente. En el modelo de código subyacente, no hay un bloque script con el atributo runat="server". (La página puede contener bloques script que carecen del atributo runat="server" si desea escribir script del cliente en la página.) La segunda diferencia está en que la directiva @Page del modelo de código subyacente contiene atributos que hacen referencia a un archivo externo (SamplePage.aspx.vb o SamplePage.aspx.cs) y una clase. Estos atributos vinculan la página .aspx a su código.

El archivo de código subyacente contiene las declaraciones de clase completas en el espacio de nombres predeterminado. Sin embargo, la clase se declara con la palabra clave partial, que indica que no está incluida totalmente en un archivo. En lugar de ello, cuando se ejecuta la página, el compilador lee la página .aspx y el archivo al que hace referencia en la directiva @ Page, los ensambla en una sola clase y, a continuación, los compila como una unidad en una sola clase.

CONTEXTO DE PÁGINA Y APLICACIÓN EN ASP.NET.

Cuando se ejecuta una aplicación Web, ASP.NET conserva información sobre la aplicación actual, cada sesión de usuario, la solicitud HTTP actual, la página solicitada, etc. ASP.NET contiene una serie de clases para encapsular esta información de contexto.

ASP.NET hace que instancias de estas clases estén disponibles como objetos intrínsecos a los que se puede tener acceso desde el código. La tabla siguiente enumera estos objetos intrínsecos y las clases de las que son instancias.

Nombre de objeto	Descripción	Clase ASP.NET
Response	Proporciona acceso a la secuencia de salida de la página actual. Puede utilizar esta clase para insertar texto en la página, escribir cookies y mucho más.	HttpResponse
Request	Proporciona acceso a la solicitud de página actual, incluidos los encabezados de solicitud, las cookies, el certificado de cliente, la cadena de consulta, etc. Puede utilizar esta clase para leer lo que ha enviado el servidor.	HttpRequest
Context	Proporciona acceso a todo el contexto actual (incluido el objeto de la solicitud). Puede utilizar esta clase para compartir información entre páginas.	HttpContext
Server	Expone métodos de utilidad que puede utilizar para transferir el control entre páginas, obtener información sobre el error más reciente, codificar y descodificar texto HTML, y mucho más.	HttpServerUtility
Application	Proporciona acceso a métodos y eventos de aplicación para todas las sesiones. También proporciona acceso a una caché de aplicación que puede utilizar para almacenar información.	HttpApplicationState
Session	Proporciona información de la sesión de usuario actual. También proporciona acceso a una caché de sesión que puede utilizar para almacenar información, junto con los medios para controlar la administración de la sesión.	HttpSessionState
Trace	Proporciona un modo de mostrar mensajes de diagnóstico de seguimiento personalizados y del sistema en el resultado de la página HTTP.	TraceContext

RECUPERACIÓN DE LA INFORMACIÓN INTRODUCIDA POR LOS USUARIOS.

> **Nota**: Tenga presente que La propiedad **ViewState** proporciona un objeto de diccionario para conservar valores entre las distintas solicitudes de una misma página. Éste es el método predeterminado que la página utiliza para conservar los valores de las propiedades de la propia página y sus controles entre recorridos de ida y vuelta.

Una de las acciones efectuadas más habitualmente en el desarrollo Web de las páginas ASP consiste en pasar datos de una página a otra para el procesamiento. Esto implica dos técnicas: una utiliza el método POST; y la otra, el método GET. Aunque existen niveles más profundos de complejidad sobre estos enfoques, en una primera aproximación, solo se muestra cómo aplicar estas ideas.

> **Nota**: Solo la página de destino debe ser una página .aspx, puesto que realmente sólo estas páginas contienen código de servidor. Las páginas de origen pueden ser .html o .aspx.

Utilizar el método POST

Las páginas ASP utilizan el método POST, como método por defecto, para enviar los valores de los controles de su único formulario a la página de destino. En el caso de que la página inicial sea una página HTML, los datos son enviados a la página (especificada en el atributo action del tag form) de acuerdo al método especificado (post o get) en su propiedad method.

En este caso la página final puede recuperar los valores de la colección de controles del formulario, enviada desde un explorador, utilizando la propiedad **HttpRequest.Form**.

La propiedad HttpRequest.Form proporciona una colección de variables de formulario, en la que cada par de nombre/valor en la colección representa un control en el formulario y su valor.

Utilizar el método GET

Con el método GET, los valores de los controles se envían a la página como una colección de variables de cadena formando parte de la dirección URL de la petición de la página de destino.

La dirección URL se monta automáticamente si se ha especificado el method=" get" en el formulario HTML o manualmente, como por ejemplo en una cadena de consulta de un hipervínculo.

La colección de variables puede ser recuperada con la propiedad **HttpRequest.QueryString**.

DIRECTIVAS PARA LAS PÁGINAS WEB ASP.NET.

Normalmente las páginas ASP.NET contienen directivas que permiten especificar las propiedades de la página y la información de configuración para ésta. ASP.NET utiliza las directivas como instrucciones sobre el modo en que se debe procesar la página, pero no se representan como parte del formato que se envía al explorador.

En ASP.NET las directivas son especificadas en bloques **<%@ %>**.

La directiva que se utiliza normalmente es **@Page**, que permite especificar muchas opciones de configuración para la página, entre las que se encuentran las siguientes:

- Lenguaje de programación del servidor para el código de la página.
- Si se trata de una página en la que se ha incluido directamente el código del servidor, denominada página de un solo archivo, o si se trata de una página en la que el código se ha incluido en un archivo de clase independiente, denominada página de código subyacente.
- Opciones de depuración y seguimiento.
- Si la página tiene una página maestra asociada y, por consiguiente, debe tratarse como una página de contenido.

Si no se incluye ninguna directiva @Page en la página o si la directiva no incluye una configuración específica, la configuración se hereda del archivo de configuración de la aplicación Web (archivo Web.config) o del archivo de configuración del sitio (archivo Machine.config).

Algunos tipos de archivos ASP.NET utilizan otras directivas distintas de @Page. Por ejemplo, las páginas maestras ASP.NET utilizan la directiva **@Master** y los controles de usuario ASP.NET utilizan la directiva **@Control**. Cada directiva permite especificar opciones distintas adecuadas para el archivo.

Directiva	Descripción
@Assembly	Vincula de forma declarativa un ensamblado a la página o control de usuario actuales.
@Control	Define atributos específicos del control utilizados por el analizador y compilador de páginas ASP.NET y sólo se puede incluir en archivos .ascx (controles de usuario).
@Implements	Indica de forma declarativa que en una página o control de usuario se implementa la interfaz de .NET Framework especificada.
@Import	Importa un espacio de nombres en una página o un control de usuario explícitamente.
@Master	Identifica una página como página principal y define los atributos utilizados por el analizador y compilador de páginas ASP.NET; sólo se puede incluir en archivos .master.
@MasterType	Define la clase o ruta de acceso virtual utilizada para escribir la propiedad **Master** de una página.
@OutputCache	Controla de forma declarativa las directivas de almacenamiento en la caché de resultados de una página o control de usuario.
@Page	Define atributos específicos de la página utilizados por el analizador y compilador de páginas ASP.NET; sólo se puede incluir en archivos .aspx.
@PreviousPageType	Crea una referencia con establecimiento inflexible de tipos a la página de origen a partir del destino de una publicación en varias páginas.
@Reference	Vincula mediante declaración una página, un control de usuario o un control COM a la página o control de usuario actuales.
@Register	Asocia alias a espacios de nombres y clases, lo que permite la representación de controles de usuario y controles de servidor personalizados cuando se incluyen en una página o un control de usuario solicitados.