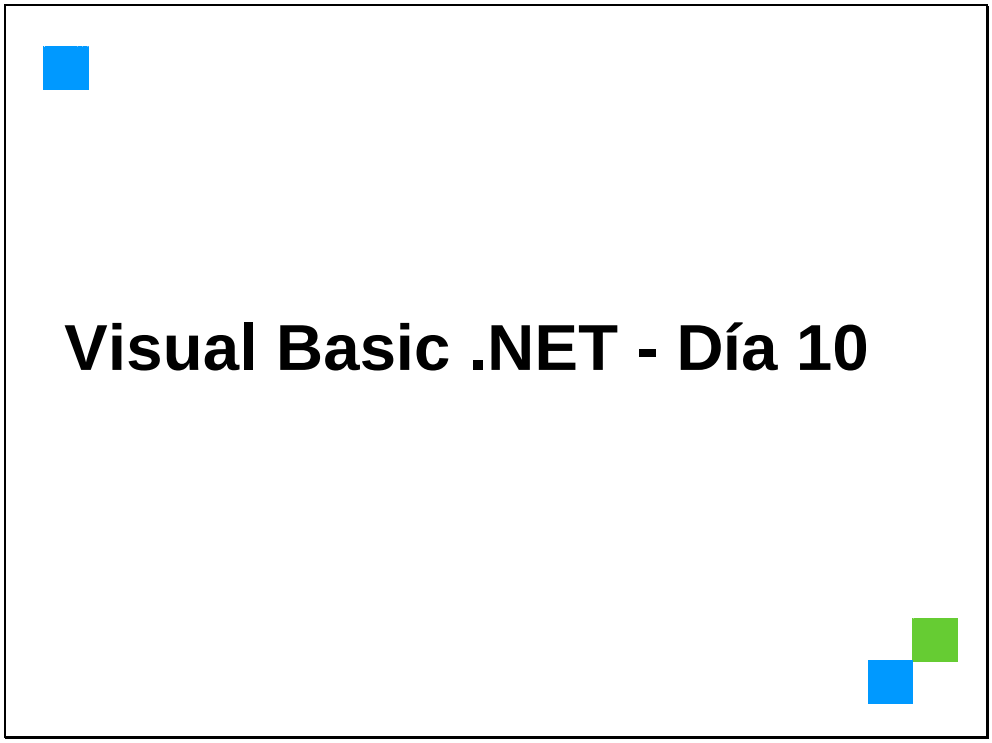
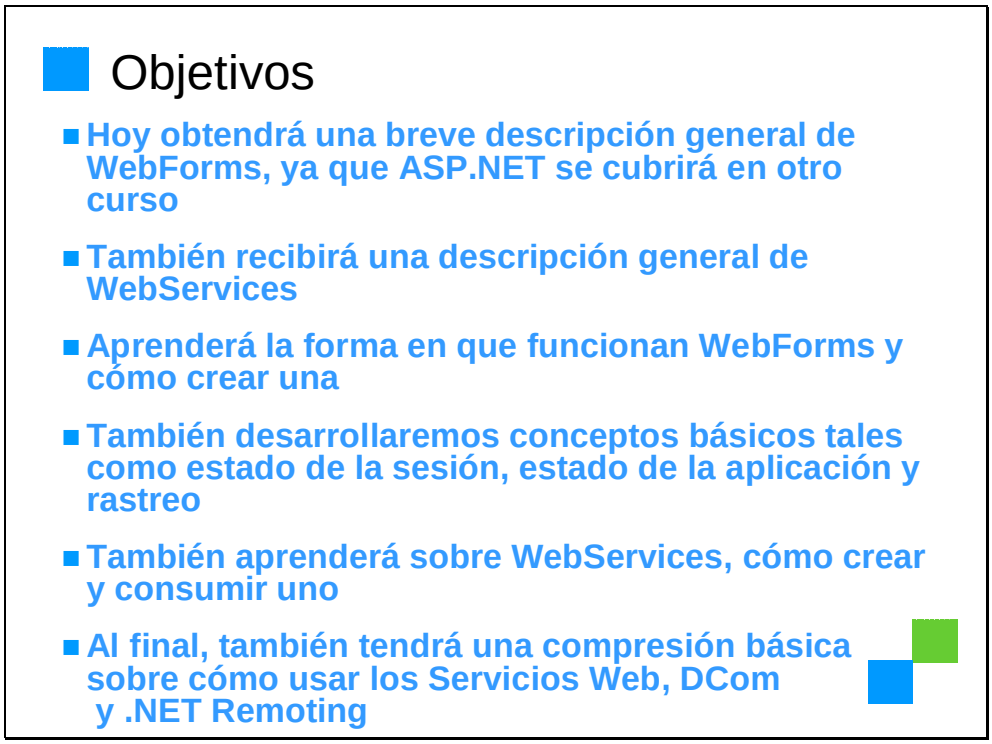
Diapositiva 1



Diapositiva 2

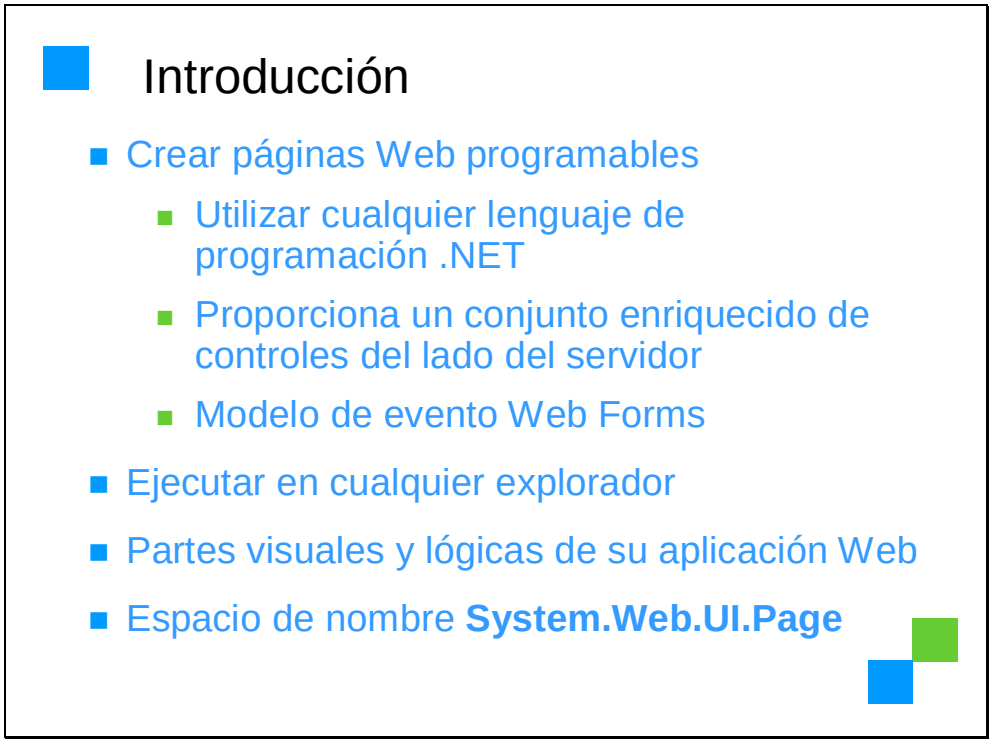


Diapositiva 3

Parte 1/2 WebForms

- Introducción
- Partes de una Web Form
- Flujo de ejecución de las Web Forms
- Crear WebFoms
  - Controles
  - Sintaxis de la página ASP.NET
  - Presentar datos
- Estado de la sesión
- Estado de la aplicación
- Autenticación del usuario
  - Validación personalizada
- Rastreo

Diapositiva 4



¿Qué es una Web Form?

Una página Web Form presenta información para el usuario en cualquier explorador o dispositivo de cliente e implementa la lógica de la aplicación utilizando un código del lado del servidor.

El resultado de una página Web Form puede contener casi cualquier lenguaje que sirve para http, incluyendo HTML, XML, WML y ECMAScript (JScript, JavaScript).

Las páginas Web Forms están basadas en la tecnología ASP.NET de Microsoft, cuyo código se ejecuta en el servidor dinámicamente y genera un resultado de la página Web para el explorador o dispositivo cliente.

El marco de la página ASP.NET elimina los detalles de la implementación de la separación del cliente y del servidor inherentes en las aplicaciones basadas en el Web al elevar un modelo unificado para responder a los eventos del cliente en un código que se ejecuta en el servidor.

System.Web.UI.Page

Cuando se compila la unidad de Web Forms, ASP.NET analiza la página y su código, genera una clase nueva dinámicamente, y luego compila esta clase nueva.

La clase generada dinámicamente se deriva de la clase ASP.NET Page, pero se extiende mediante controles, su código y el texto HTML estático en el archivo .aspx.

Durante el tiempo de ejecución, la clase Page procesa solicitudes entrantes y responde al crear dinámicamente HTML y agilizarlo de vuelta al explorador. Si la página contiene controles Web (como es común), la clase Page derivada actúa como contenedor para los controles y las instancias de los controles se crean durante el tiempo de ejecución, así mismo hace llegar el texto HTML al flujo.

Page contiene los siguientes objetos.

Server: Obtiene el objeto Server, que es una instancia de la clase HttpServerUtility.

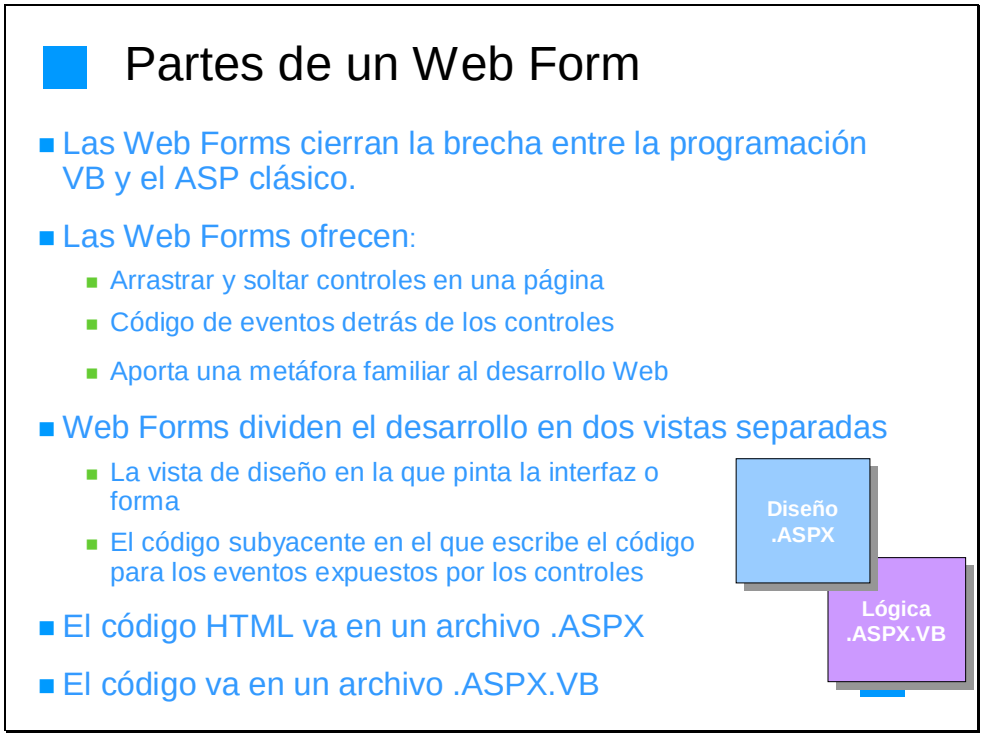
Application: Obtiene el objeto Application para la solicitud Web actual.

Session: Obtiene el objeto Session actual que proviene de ASP.NET.

Request: Obtiene el objeto HttpRequest para la página solicitada. HttpRequest permite que ASP.NET lea los valores de HTTP que envió un cliente durante una solicitud Web.

Response: Obtiene el objeto HttpResponse asociado con Page. Este objeto le permite enviar datos de respuesta HTTP a un cliente, además de que contiene información acerca de la respuesta.

Diapositiva 5



Web Forms cierran la brecha entre la programación VB y el ASP clásico.

Las Web Forms ofrecen

Arrastrar y soltar controles en una página

Código de eventos detrás de los controles

Aporta una metáfora familiar al desarrollo Web

Web Forms dividen el desarrollo en dos vistas separadas

La vista de diseño en la que presenta la interfaz o forma

El código subyacente en el que escribe el código para los eventos expuestos por los controles

HTML va en los archivos .ASPX

El código va en los archivos .ASPX.VB

Después estos archivos se colocan un servidor Web y los ejecuta el servidor Web.

Cuando una solicitud llega al servidor Web, compila la webform en una clase ejecutable desde la compilación de ambos archivos.

Diapositiva 6

Archivo Global.asax

El archivo Global.asax, también conocido como el archivo de aplicación ASP.NET, es un archivo opcional que contiene el código para responder a eventos a nivel de aplicación que surgen a través de ASP.NET o de los módulos HTTP.

El archivo Global.asax reside en el directorio raíz de una aplicación basada en ASP.NET.

Durante ese tiempo se analiza y se compila Global.asax en una clase .NET Framework generada dinámicamente y derivada a partir de la clase base HttpApplication.

El propio archivo Global.asax está configurado de manera que cualquier solicitud URL directa se rechaza automáticamente; los usuarios externos no pueden descargar o ver el código escrito en él.

Cuando guarda los cambios en un archivo activo Global.asax, el marco de la página ASP.NET detecta que se cambió el archivo. Completa todas las solicitudes actuales para la aplicación, envía el evento Application_OnEnd a cualquier escucha, y reinicia el dominio de la aplicación.

En efecto, éste reinicia la aplicación, cerrando todas las sesiones del explorador y desechando toda la información sobre el estado. Cuando llega la siguiente solicitud entrante de un explorador, el marco de la página ASP.NET vuelve a analizar

sintácticamente y a recopilar el Archivo global .asax y lanza el evento Application_OnStart.

Eventos generales en el archivo Global.asax

Application_OnStart: este evento ocurre como el primer evento en la canalización HTTP de la ejecución cuando ASP.NET responde a una solicitud.

Application_OnEnd: este evento ocurre como el último evento en la canalización HTTP de la ejecución cuando ASP.NET responde a una solicitud.

Session_OnStart: este evento ocurre cuando se crea una sesión. Por ejemplo, cuando el usuario abre su explorador y entra a una aplicación ASP.NET, se crea una nueva sesión.

Session_OnEnd: este evento ocurre cuando termina una sesión. Por ejemplo, cuando el usuario cierra su explorador, la sesión concluye.

Archivo Web.Config

La información de configuración se almacena en los archivos de texto basados en XML. Puede usar cualquier editor de texto estándar o analizador XML para crear y editar archivos de configuración ASP.NET.

Web.Config es el archivo de configuración principal a nivel de aplicación.

El sistema de configuración ASP.NET es ampliable. Puede definir nuevos parámetros de configuración y escribir los manejadores de sección de configuración para procesarlos.

ASP.NET protege los archivos de configuración contra acceso externo al configurar Internet Information Services (IIS) y así evitar acceso directo del explorador a los archivos de configuración. El error de acceso 403 (prohibido) de HTTP se devuelve a cualquier explorador que intente solicitar directamente un archivo de configuración.

Ejemplo.

Este ejemplo demuestra cómo acceder a un parámetro de aplicación definido en un archivo Web.Config en la sección respectiva.

Archivo Web.Config

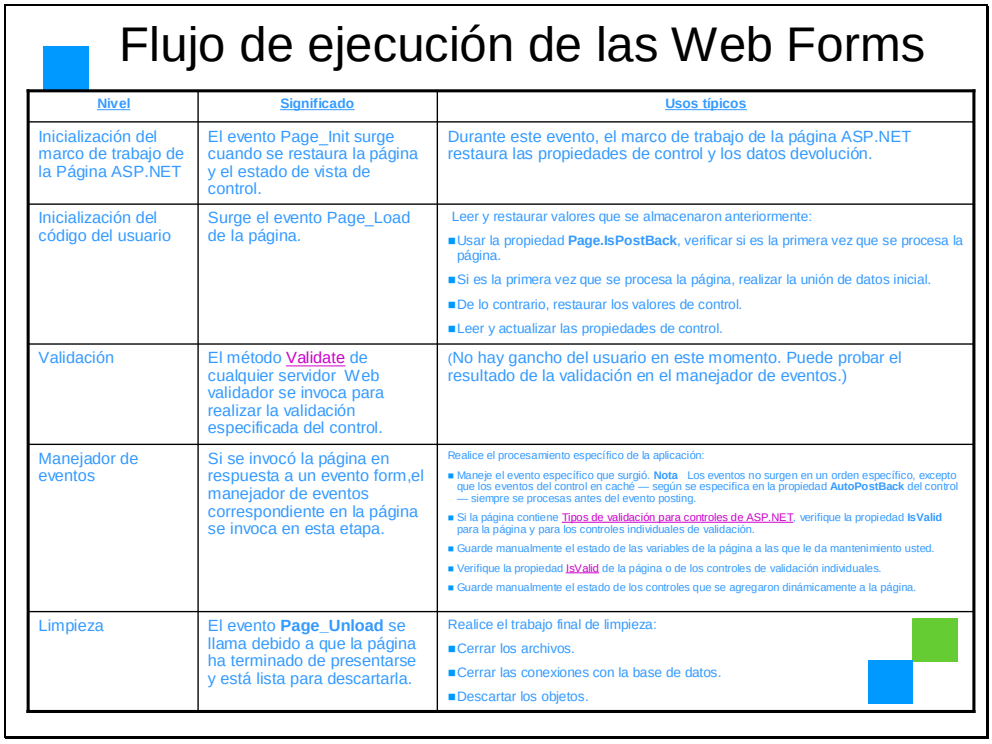
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
        <appSettings>
                <add key="WelcomeMessage" value="Hello World!!!" />
        </appSettings>
…
```

appSettings es una sección predefinida (en el archivo Machine.config) para configurar los parámetros de la aplicación.

HelloWorld.aspx page

```
<%@ Page Language="VB" %>
<script runat="server" language=vb>
 ‘ Insert page code here
   Sub Button1_Click(sender As Object, e As EventArgs)
      Dim msg As String = System.Configuration.ConfigurationSettings.AppSettings(
"WelcomeMessage“)
      Label1.Text = msg
   End Sub
</script>
<html>
        <head>
        </head>
        <body>
                <form runat="server">
                        <asp:Label id="Label1" runat="server">Label</asp:Label>
                        <asp:Button id="Button1" onclick="Button1_Click" runat="server"
Text="Button"></asp:Button>
                        <!-- Insert content here -->
                </form>
        </body>
</html>
```

Diapositiva 7



El marco de trabajo de la página ASP.NET procesa las páginas Web Forms en niveles distintos.

Durante cada nivel de procesamiento Web Forms pueden surgir eventos, y cualquier manejador de eventos que corresponda a las ejecuciones de los mismos. Estos métodos le proporcionan puntos de entrada que le permiten actualizar los contenidos de la página Web Forms.

La tabla enumera los niveles más comunes del procesamiento de página, que suceso dispara cada evento, y usos típicos en cada nivel. Estos niveles se repiten cada vez que se solicita la página. La propiedad Page.IsPostBack le permite comprobar si se está procesando la página por primera vez.

Nota: Existen algunos niveles más de procesamiento de página Web Forms que se enumeran en la tabla.

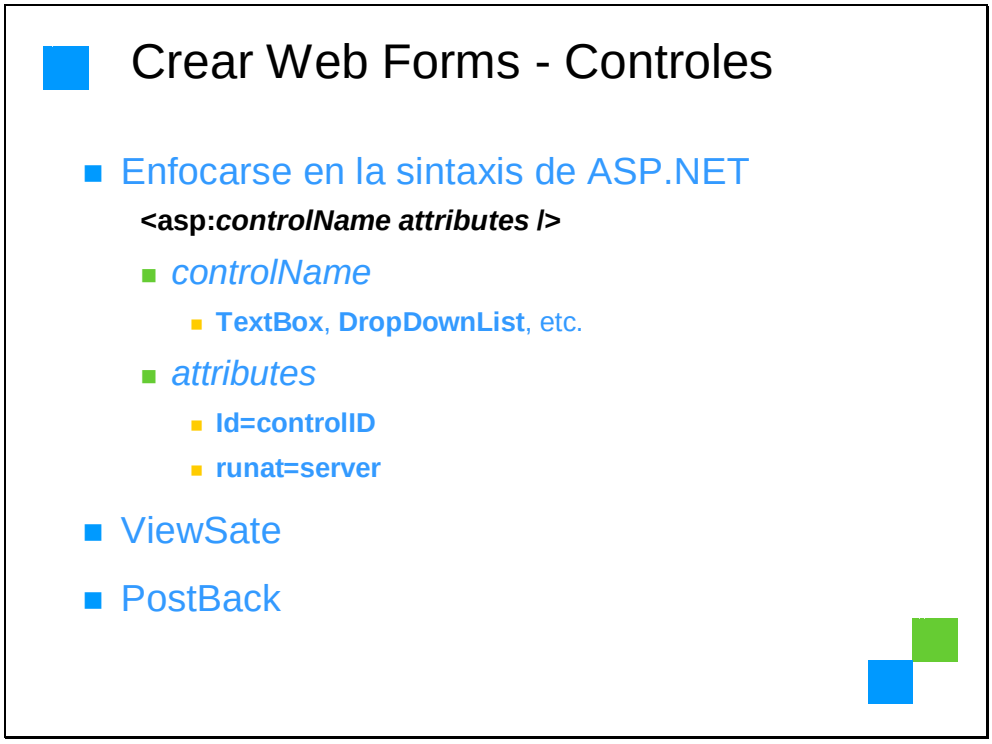
Sin embargo, no se utilizan para la mayoría de los procesamientos de página.

En su lugar, los controles del servidor en la página Web Forms los usan principalmente para iniciar y entregar. Si intenta escribir sus propios controles de servidor ASP.NET, necesita comprender mejor estos niveles.

Nivel	**Significado**	**Usos típicos**
Inicialización del marco de trabajo de la Página ASP.NET	El evento Page_Init surge cuando se restaura la página y el estado de vista de control.	Durante este evento, el marco de trabajo de la página ASP.NET restaura las propiedades de control y los datos devolución.
Inicialización del código del usuario	Surge el evento Page_Load de la página.	Leer y restaurar valores que se almacenaron anteriormente: * Usar la propiedad **Page.IsPostBack**, verificar si es la primera vez que se procesa la página. * Si es la primera vez que se procesa la página, realizar la unión de datos inicial. * De lo contrario, restaurar los valores de control. * Leer y actualizar las propiedades de control.
Validación	El método Validate de cualquier servidor Web validador se invoca para realizar la validación especificada del control.	(No hay referencia del usuario en este momento. Puede probar el resultado de la validación en el manejador de eventos.)

Manejador de eventos	Si se invocó la página en respuesta a un evento form ,el manejador de eventos correspondiente en la página se invoca en esta etapa.	Realice el procesamiento específico de la aplicación: * Maneje el evento específico que surgió. **Nota** Los eventos no surgen en un orden específico, excepto que los eventos del control en caché — según se especifica en la propiedad **AutoPostBack** del control — siempre se procesas antes del evento posting. * Si la página contiene Tipos de validación para controles de ASP.NET, verifique la propiedad **IsValid** para la página y para los controles individuales de validación. * Guarde manualmente el estado de las variables de la página a las que le da mantenimiento usted. * Verifique la propiedad IsValid de la página o de los controles de validación individuales. * Guarde manualmente el estado de los controles que se agregaron dinámicamente a la página.
Limpieza	El evento **Page_Unload** se llama debido a que la página ha terminado de presentarse y está lista para descartarla.	Realice el trabajo final de limpieza: Cerrar los archivos. Cerrar las conexiones con la base de datos. Descartar los objetos.

Diapositiva 8



ViewSate

Cada control en una página Web Forms, incluyendo la página misma, tiene una propiedad ViewState que hereda de la clase base Control.

El marco de la página ASP.NET utiliza el estado de la vista para guardar automáticamente los valores de la página y de cada control justo antes de enviarlos a la página.

Cuando se envía la página, una de las primeras tareas que lleva a cabo el procesamiento de la página es restaurar el estado de la vista.

ViewState es un campo oculto que contiene los valores encriptados de todos los controles del Servidor Web sobre la página.

ViewState se puede deshabilitar de los ajustes de control de la propiedad EnableViewState a Falso.

Ejemplo.

Dada esta definición de la página.

```
…
<body>
        <form id="HelloWorld" method="post" runat="server">
                <asp:Label id="Label1" runat="server"
EnableViewState=“True”>Label</asp:Label>
                <asp:Button id="Button1" runat="server" Text="Button"
EnableViewState=“True”></asp:Button>
        </form>
</body>
…
```

Al invocar esta página obtiene el siguiente resultado una vez que se procesa la página del servidor.

```
…
        <body>
                <form name="HelloWorld" method="post" action="HelloWorld.aspx"
id="HelloWorld">
<input type="hidden" name="__VIEWSTATE"
value="dDw2NjY0NzU4OTc7dDw7bDxpPDE+Oz47bDx0PDtsPGk8MT47PjtsPHQ8
cDxwPGw8VGV4dDs+O2w8SGVsbG8gV29ybGQhISE7Pj47Pjs7Pjs+Pjs+nAC
6oDwHqiXs92PmEYRWSP6gU3s=" />
                        <span id="Label1">Hello World!!!</span>
                        <input type="submit" name="Button1" value="Button"
id="Button1" />
                </form>
        </body>
```

Eventos y evento PostBack

A diferencia de los eventos en las aplicaciones de escritorio, los eventos controlados por el servidor de ASP.NET se presentan y se manejan en el servidor. Cuando una solicitud Web comunica una acción del lado del cliente al servidor, un control puede elevar eventos en el servidor en respuesta a la acción del cliente. La página o sus controles hijo manejan el evento, y ASP.NET envía una respuesta de vuelta al cliente. Esto hace que la experiencia del usuario sea parecida a la de la aplicación del escritorio. Sin embargo, los desarrolladores de control deben comprender que únicamente los eventos del lado del cliente se envían al servidor, lo que constituye el evento postback.

En los controles del servidor Web, algunos eventos, por lo general los eventos de clic, provocan que las formas se envíen de vuelta al servidor.

Los eventos de cambio en los controles del servidor de HTML y los controles del servidor Web, como el control TextBox, se registran, pero no se envían de inmediato. En su lugar, el control los detecta hasta la siguiente vez que ocurre un envío.

Posteriormente, cuando se procesa de nuevo la página en el servidor, todos los eventos pendientes se lanzan y se procesan.

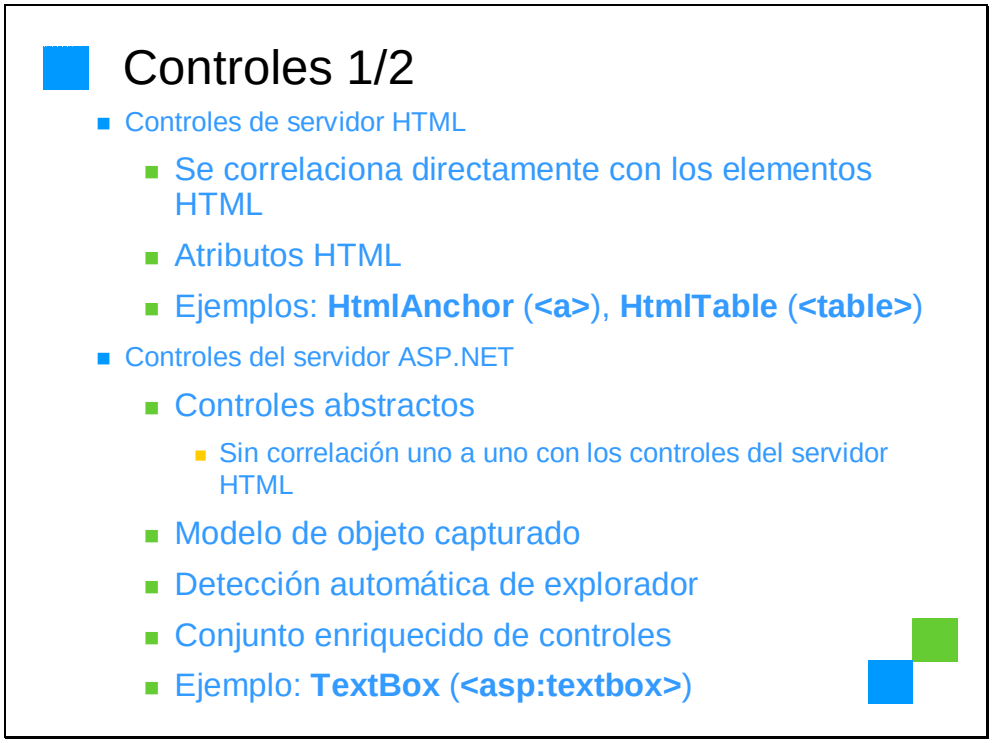
Los controles del servidor Web que soportan un evento de cambio incluyen la propiedad AutoPostBack.

Para detectar un evento PostBack en su código, utilice la propiedad Page.IsPostBack que obtiene un valor que indica si la página se cargó en respuesta a un postback del cliente (Ejemplo: un clic de uso de un Botón de servidor Web en la página), o si se cargó y se accedió por primera vez.

Ejemplo.

```
Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Not Page.IsPostBack Then
        Response.Write("First time")
    Else
        Response.Write("PostBack event")
    End If
End Sub
```

Diapositiva 9



Los elementos HTML dentro de un archivo ASP.NET se tratan como texto literal y son inaccesibles de manera programática durante el procesamiento de servidor de la página.

Para hacer programáticamente accesible a estos elementos, puede indicar que un elemento HTML se debe analizar y tratar como un control de servidor al añadir un atributo **runat="server"**. El atributo único **id** le permite referirse de manera programática al control.

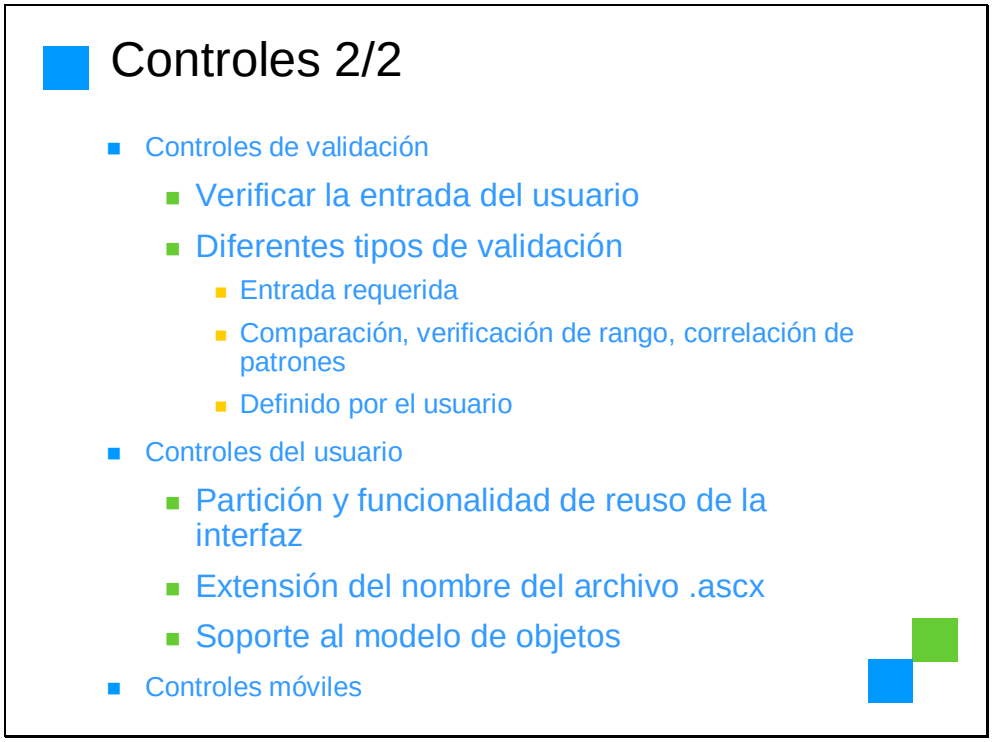
Los atributos se utilizan para declarar los argumentos de la propiedad y los enlaces del evento en instancias del control del servidor. Los controles del servidor HTML deben residir dentro de una etiqueta que contenga **<form>** con el atributo **runat="server"**.

El espacio de nombre System.Web.UI.WebControls es un conjunto de clases que le permite crear controles de servidor Web en una página Web.

Los controles del servidor Web se ejecutan en el servidor e incluyen controles de la forma como botones y cuadros de texto. También pueden incluir controles de propósitos especiales como un calendario.

Dado que los controles del servidor Web se ejecutan en el servidor, usted puede controlar de manera programática estos elementos. Los controles del servidor Web son más abstractos que los controles del servidor HTML.

Diapositiva 10



¿Qué es un control de validación?

Los Controles de validación del servidor son un conjunto de controles que le permiten validar un control de servidor asociado con las entradas, como un TextBox, y muestra un mensaje personalizado cuando falla la validación. Cada control de validación lleva a cabo un tipo de validación específico.

Los controles de validación llevan a cabo verificaciones de entrada en un código de servidor. Cuando el usuario envía una forma al servidor, los controles de validación se invocan para revisar las entradas del usuario, control por control. Si ocurre un error en cualquiera de los controles de entrada, la misma página establece un estado inválido de manera que pueda poner a prueba la validez antes de ejecutar sus códigos.

Si el usuario está trabajando con un explorador que soporta DHTML, los controles de validación también pueden llevar a cabo validaciones utilizando secuencias de comandos de cliente. Esto puede mejorar sustancialmente el tiempo de respuesta en la página; los errores se detectan de inmediato y los mensajes de error se muestran tan pronto como el usuario deja el control que contiene el error.

Tipos de controles de validación

Entrada requerida

RequiredFieldValidator: asegura que el usuario no se salte una entrada requerida.

Ejemplo.

```
…
<body>
        <form runat=server ID="Form1">
                <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator id="RequiredFieldValidator1"
runat="server" ErrorMessage="Field is required"
ControlToValidate="TextBox1"></asp:RequiredFieldValidator>
        </form>
</body>
```

Comparación con un valor

CompareValidator: compara una entrada del usuario contra un valor constante, o un valor de propiedad de otro control, utilizando un operador de comparación (menor que, igual a, mayor que, etc.).

Ejemplo.

```
…
<body>
        <form runat=server ID="Form1">
                <asp:TextBox id="TextBox2" runat="server"></asp:TextBox>
                <asp:CompareValidator id="CompareValidator1" runat="server"
ErrorMessage="Control must be equal to 43." ControlToValidate="TextBox2"
ValueToCompare="43"></asp:CompareValidator></P>
        </form>
</body>
```

Verificación de rango

RangeValidator: verifica que la entrada del usuario se encuentre entre los límites inferior y superior especificados. Puede verificar los rangos dentro de pares de números, caracteres alfabéticos y fechas.

Ejemplo.

```
…
<body>
        <form runat=server ID="Form1">
                <asp:TextBox id="TextBox3" runat="server"></asp:TextBox>
                <asp:RangeValidator id="RangeValidator1" runat="server"
ErrorMessage="Field values between 1 and 12." ControlToValidate="TextBox3"
MaximumValue="1" MinimumValue="12"></asp:RangeValidator>
        </form>
</body>
```

Correlación de patrones

RegularExpressionValidator: verifica que la entrada concuerde con el patrón definido por una expresión regular. Este tipo de validación le permite verificar las secuencias predecibles de caracteres, tales como aquellas en los números de seguro social, direcciones electrónicas, números telefónicos, códigos postales, etc.

Ejemplo.

```
…
<body>
        <form runat=server ID="Form1">
                <asp:TextBox id="TextBox4" runat="server"></asp:TextBox>
                <asp:RegularExpressionValidator id="RegularExpressionValidator1"
runat="server" ErrorMessage="Enter the valid email address"
ControlToValidate="TextBox4" ValidationExpression="\w+([-+.]\w+)*@\w+([-
.]\w+)*\.\w+([-.]\w+)*"></asp:RegularExpressionValidator>
        </form>
</body>
```

Definido por el usuario

CustomValidator: verifica la entrada del usuario utilizando la lógica de validación que codificó usted mismo. Este tipo de validación le permite verificar los valores derivados durante el tiempo de ejecución.

Ejemplo.

```
...
<body>
        <form runat=server ID="Form1">
                <asp:TextBox id="TextBox5" runat="server"></asp:TextBox>
                <asp:CustomValidator id="CustomValidator1" runat="server"
ErrorMessage="Error Message." ControlToValidate="TextBox5"
ClientValidationFunction="ClientValidationFunction“
OnServerValidate="ServerValidation" ></asp:CustomValidator></P>
        </form>
</body>
<script runat=server language=vb>
        Sub ServerValidation (source As Object, args As ServerValidateEventArgs)
                try
                        ‘ execute your validation
                        args.IsValid = true
                catch
                        args.IsValid = false
                End Try
        End Sub
</script>
<script language="vbscript">
<!--
        Sub ClientValidationFunction(source, arguments)

                ' execute your validation
                If ( Some Validation is True ) Then
                        arguments.IsValid=true
                Else
                        arguments.IsValid=false
                End If
        End Sub
-->
</script>
```

Controles del usuario

Los controles del usuario del Web son fáciles de definir, pero puede ser poco conveniente utilizarlos en escenarios avanzados.

Usted desarrolla controles de usuario del Web casi de la misma manera que como desarrolla las páginas Web Forms. Sin embargo, ya que los controles del usuario del Web se compilan dinámicamente durante el tiempo de ejecución, no se pueden agregar a la Caja de herramientas, y están representados por un sencillo símbolo cuando se agrega a la página.

Los controles personalizados del Web son códigos compilados, que los hace más fáciles de utilizar, pero más difíciles de crear; los controles personalizados del Web deben estar escritos en código. Resulta imposible instalar una copia sencilla de los controles personalizados del Web en el caché global de ensamblados (GAC) y compartirlo entre las aplicaciones, lo que hace más fácil su mantenimiento.

Controles móviles

El Tiempo de ejecución de los controles móviles de Internet permite que los desarrolladores se enfoquen en una gran variedad de dispositivos móviles, incluyendo teléfonos celulares habilitados para el Web, localizadores y PDA como la Pocket PC.

El kit de herramientas Internet Mobile de Microsoft proporciona el mismo modelo de aplicación enriquecido del Web que encuentra en las aplicaciones ASP.NET de Microsoft, enfocados en los exploradores de escritorio.

Esta sección del documento sobre el kit de herramientas describe las extensiones que se añadieron a Web Forms de ASP.NET.

Diapositiva 11

ASP.NET – Sintaxis de la página 1/3

- Directrices
  - <%@ Page language="vb" […] %>
- Bloque de declaración de códigos
  - <script runat="server" […]>
    [lines of code]<s
    </script>
- Bloques de entrega de código
  - <%
    [inline code or expression] [
    %>
- Sintaxis del control HTML
  - <HTMLelement runat="server" [attribute(s)]>
    </HTMLelement>

Directrices de @ Page

Define los atributos específicos de la página (archivo .aspx) que utilizan el analizador y el compilador de ASP.NET.

Ejemplo

El siguiente código instruye al compilador de la página ASP.NET para utilizar VB como el lenguaje código en línea y establece el ContentType MIME predeterminado de HTTP que se trasmite al cliente para "text/xml".

```
<%@ Page Language="vb" ContentType="text/xml" %>
```

Bloque de declaración de códigos

Define las variables de miembros y los métodos que se compilan en las clases Page generadas dinámicamente.

Sintaxis

```
<script runat="server" language="codelanguage" Src="pathname">
  Code goes here…
</script>
```

Runtat = “server”

Determina dónde se procesa el código.

Src

Especifica la ruta y el nombre de un archivo de secuencia de comandos para cargar. Cuando se utiliza este atributo, no se ignora ningún otro código en el bloque de declaración.

Ejemplo.

```
<html>
  <script language=“vb" runat="server">
     Sub EnterBtn_Click(Src As Object, e As EventArgs)
        Message.Text = "Hi " & Name.Text & ", welcome to ASP.NET!"
     End Sub
  </script>
  <body>
   <form runat="server" ID="Form1">
    Enter your name: <asp:textbox id="Name" runat=server/>
                  <asp:button text="Enter" Onclick="EnterBtn_Click" runat="server"
ID="Button1" NAME="Button1"/>
       <p>
       <asp:label id="Message" runat=server/>
    </form>
  </body>
</html>
```

Bloques de entrega de código

Define códigos en línea o expresiones en línea que se ejecutan cuando se entregan a la página. Existen dos estilos: código en línea y expresiones en línea. Utilice el código en línea para definir los bloques de códigos auto-contenidos o los bloques de flujo de control.

Código en línea: utilice el código en línea para definir bloques de códigos auto-contenidos o bloques de flujo de control.

Sintaxis

```
<% inline code %>
```

Ejemplo

```
<%
        …
        Response.Write(“Text");
%>
```

El método Response.Write escribe una cadena para un flujo de contenido de resultado HTTP.

Expresiones en línea: utilice expresiones en línea como un acceso directo para invocar el método HttpResponse.Write.

Sintaxis

```
<% inline expression %>
```

Ejemplos de atributos:

Inherits

Define una clase con código subyacente para que la página la herede.

Debug

Indica si la página se debe compilar con los símbolos de depuración.

EnableViewState

Indica si el estado de vista se mantiene en todas las solicitudes de la página.

Transaction

Indica si las transacciones se soportan en la página.

Los valores posibles son Disabled, NotSupported, Supported, Required y RequiresNew.

El atributo de transacciones declarativas especifica cómo participa un objeto en una transacción, y cómo se configura de manera programática.

Microsoft Transaction Server (MTS) proporciona los servicios de transacción, COM+ 1.0 que es un servicio del sistema operativo de Windows.

Diapositiva 12

ASP.NET – Sintaxis de la página 2/3

- Sintaxis del contro del servidor Web
  - Identificadro del control Button
    - **<ASP:Button id="MyButton" runat="server">**
  - Propiedad del control Button
    - **<ASP:Button text="Click Me!" runat="server">**
  - Unión de datos del control Button
    - **<ASP:Button onclick="MyClick" runat="server">**

El control Web Server es un control que se puede incluir en una página ASPX que se procesa en un servidor y que, por lo general, contiene una interfaz gráfica. Cuando se invoca una página, el servidor procesa la solicitud y todos los controles contenidos en la página solicitada y el resultado HTML se envía al explorador, por ejemplo, el Botón del control del servidor Web presenta un botón de presión a un usuario una vez que se procesa.

Sintaxis

```
<tagprefix:tagname id="OptionalID" attributename="value"
eventname="eventhandlermethod" runat="server" />
```

O

```
<tagprefix:tagname id="OptionalID" runat="server" > </tagprefix:tagname>
```

tagprefix

Un alias para el espacio de nombre totalmente calificado del control. Los alias para los controles escritos por el usuario se declaran con la directiva @ Register. Por ejemplo, "<**ASP**: …".

tagname

El nombre de la clase que encapsula el control. Por ejemplo, "<ASP:**Button** …".

id

Un identificador único que habilita la referencia programática al control. Por ejemplo, “<ASP:Button **id=“MyButton”**… ”.

attributename

El nombre del atributo. Por ejemplo, “<ASP:Button id=“MyButton” **text**=“Click Me!!!”…”.

value

El valor asignado al atributo. Por ejemplo, “<ASP:Button id=“MyButton” text=“**Click Me!!!**”…”.

eventname

El nombre del evento del control.

eventhandlermethod

El nombre del método del manejador del evento definido en el código para la página Web Forms.

Por ejemplo, “<ASP:Button id=“MyButton” text=“Click Me!!!”

onclick=“**Click_EventHandler**”…”.

Ejemplo.

Este ejemplo muestra una página Web con los controles Label, TextBox y Button.

```
<%@ Page language=“vb" Codebehind="Controls.aspx.vb" AutoEventWireup="false" Inherits="WebApplicationvb.Controls" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
	<HEAD>
		<title>Controls</title>
		<meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
		<meta name="CODE_LANGUAGE" Content=“vb">
		<meta name="vs_defaultClientScript" content="JavaScript">
		<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
	</HEAD>
	<body>
		<form id="Controls" method="post" runat="server">
			<asp:Label id="lblName" runat="server">Name:</asp:Label>
			<asp:TextBox id="txtName" runat="server"></asp:TextBox>
			<asp:Button id="cmdSearch" runat="server" Text="Search"></asp:Button>
		</form>
	</body>
</HTML>
```

Diapositiva 13

ASP.NET – Sintaxis de la página 3/3

- Expresión de unión de datos
  - <asp:label
    text=<'<%# **databinding expression** %>'
    runat=s"server" />
- Etiquetas del objeto del lado del servidor
  - <object id="id" runat="server"
    identifier=e"idName" />
- Directrices Include del lado del servidor
  - <!-- #include pathtype = filename -->
- Comentarios del lado del servidor
  - <%-- comment block --%>
- Eventos
  - ASP.NET
    <asp:ImageButton id=btnNext runat="server"
    imageurl="..." onclick="brtnNext_Click"/>
  - VB Protected Sub btnNext_Click(S As Object,_
    E As ImageClickEventArgs)
    [… do something ...]
    End Sub

Directrices Include del lado del servidor

Inserta los contenidos de un archivo específico en cualquier parte dentro de una página ASP.NET.

Sintaxis.

```
<!-- #include file | virtual = filename -->
```

Ejemplo.

El siguiente ejemplo muestra cómo puede utilizar las sintaxis de directrices Include del lado del servidor para invocar archivos que crearán un encabezado y una nota al pie de página y una página ASP.NET. Ambos utilizan rutas relativas.

```
<html>
  <body>
    <!-- #Include virtual=".\include\header.inc" -->
    Here is the main body of the .aspx file.
    <!-- #Include virtual=".\include\footer.inc" -->
  </body>
</html>
```

Comentarios del lado del servidor

Le permite incluir comentarios sobre el código en el cuerpo de un archivo .aspx. Cualquier contenido entre las etiquetas de abertura y cierre de los elementos del

comentario del lado del servidor, ya sea un texto de código literal ASP.NET, no se procesarán en el servidor ni se enviarán a la página resultante.

Sintaxis.

```
<%-- commented out code or content --%>
```

Ejemplo.

El siguiente ejemplo muestra un control HtmlButton que se comentó.

```
<%--
        <button runat="server" id="MyButton" OnServerClick="MyButton_Click">
                Click here for enlightenment!
        </button>
--%>
```

Eventos

En aplicaciones basadas en el cliente, los eventos se elevan y se manejan en el cliente. Por otro lado, en las páginas Web Forms, los eventos asociados con los controles del servidor se elevan en el cliente, pero se manejan en el servidor Web a través del marco de la página ASP.NET.

Ejemplo.

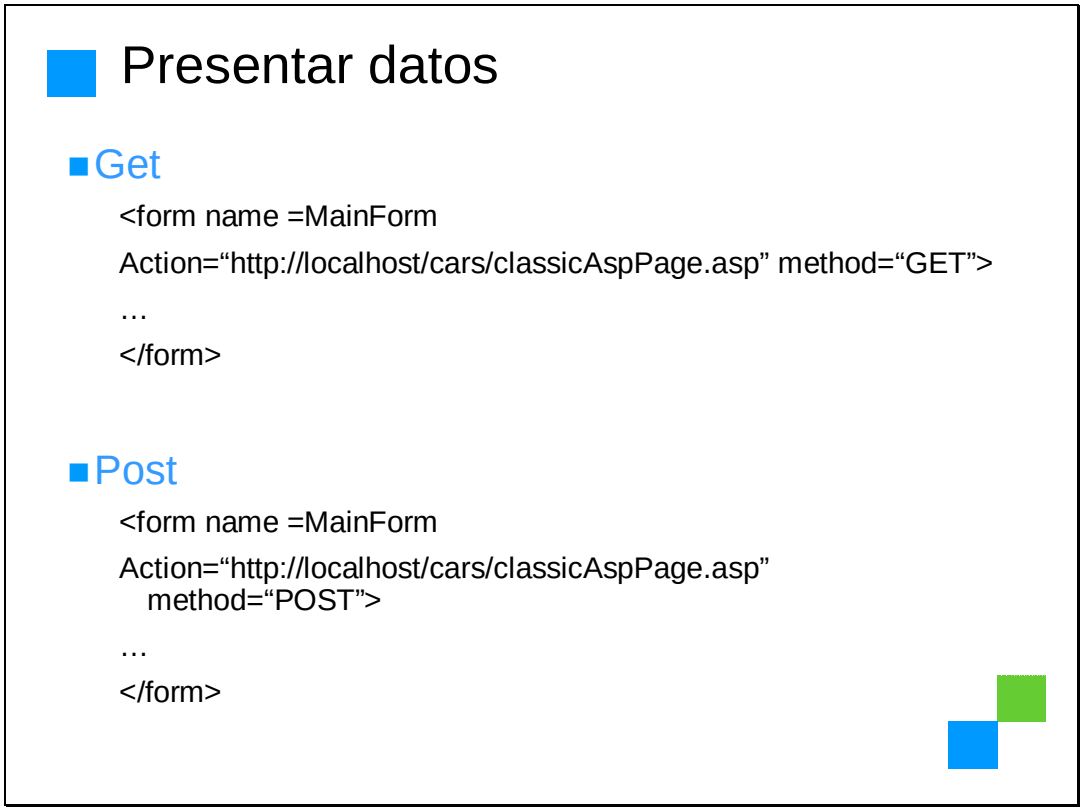
Control declaration.

```
….
<asp:ImageButton id=btnNext runat=“server“ imageurl=“...“
onclick=“EventHandler“/>
...
```

Server script

```
....
   Protected Sub btnNext_Click(ByVal S As Object, _
         ByVal E As ImageClickEventArgs)
        [ ... do something ... ]
   End Sub
```

Diapositiva 14



Presentar datos

Cuando desarrolla una página HTML, típicamente proporciona un atributo de acción para especificar al receptor de los datos entrantes.

Los receptores posibles incluyen los servidores de correo, otros archivos HTML, una Active Server Page, etc.

En los ejemplos de la diapositiva les diremos que utilicen los métodos **GET** o **POST** de adjuntar datos.

GET – Cuando se elige este modo de transmisión de datos, se toman los datos del formulario, usando la cadena de consulta como un conjunto de pares de nombre/valor y se envía.

POST – Cuando se elige este modo de transmisión de datos, no se toman los datos del formulario en la cadena de consulta, sino que se escriben en una línea por separado enviada por el encabezado HTTP.

De esta forma, los datos del formulario no son visibles directamente al mundo externo y por lo tanto son más seguros.

Lo más importante es que el envío no está limitado por la longitud de los caracteres.

Diapositiva 15

¿Qué es una sesión?

Una sesión se define como el periodo de tiempo durante el cual un usuario único interactúa con una aplicación Web.

Los desarrolladores de Active Server Pages (ASP) que desean conservar los datos para las sesiones de usuario único pueden utilizar una función intrínseca conocida como estado de sesión.

De manera programática, el estado de la sesión no es más que la memoria en forma de un diccionario o una tabla de revisión, por ejemplo pares de valores clave, que se pueden establecer y leer durante una sesión del usuario.

Las páginas Web se vuelven a crear cada vez que la página se envía al servidor. En una programación Web tradicional, esto significaría comúnmente que toda la información asociada con la página y los controles de la página se perderían en cada viaje de ida y vuelta. Por ejemplo, si un usuario registra la información en un cuadro de texto, la información se perdería en el viaje de ida y vuelta del explorador o dispositivo cliente al servidor.

ASP.NET le permite guardar los valores utilizando el estado de sesión, que es una instancia de la clase HttpSessionState para cada sesión de aplicación Web activa.

Ejemplo.

Para guardar valores en la sesión.

```
Dim firstName As String = "John"
Dim lastName As String = "Smith"
Dim city As String = "Seattle"
Session("FirstName") = firstName
Session("LastName") = lastName
Session("City") = city
```

Para recuperar los valores desde una sesión.

```
Dim firstName As String = CType(Session.Item("FirstName"), String)
Dim lastName As String = CType(Session.Item("LastName"), String)
Dim city As String = CType(Session.Item("City"), String)
```

Eventos de sesión

ASP.NET proporciona dos eventos a nivel de sesión, **Session_OnStart** ocurre cuando se crea una sesión, por ejemplo, cuando un usuario navega a un sitio, **Session_OnEnd** ocurre cuando una sesión termina, por ejemplo, cuando un usuario cierra un explorador.

Estos eventos se ubican en el archivo Global.asax que se ubican en la raíz de la aplicación ASP.NET.

Declaración de eventos.

```
Protected Sub Session_Start(sender As Object, e As EventArgs)
    'do something
End Sub
Protected Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
    'do something
End Sub
```

Configuración Web.Config

Los atributos de la configuración a nivel administración en la sesión en la aplicación Web.

```
<sessionState mode="Off|Inproc|StateServer|SQLServer"
        cookieless="true|false"
        timeout="number of minutes"
        stateConnectionString="tcpip=server:port"
        sqlConnectionString="sql connection string" />
```

Modo

Especifica dónde almacenar el estado de la sesión.

Off: Indica que el estado de la sesión no está habilitado.

Inproc: Indica que el estado de la sesión está almacenado localmente.

StateServer: Indica que el estado de la sesión está almacenado en un servidor remoto.

SQLServer: Indica que el estado de la sesión se almacena en el Servidor SQL.

Cookieless

Especifica si las sesiones sin *cookies* se deben utilizar para identificar las sesiones del cliente. El sessionId se envía a la parte de la cadena consultada.

Ejemplo.

```
http://localhost/samplecookieless/(lvyrrzzwljpjtl75d4mwyy59)/login.aspx
```

TimeOut

Especifica el número de minutos que puede permanecer inactiva una sesión antes de que se abandone. El valor predeterminado es 20.

stateConnectionString

Especifica el nombre del servidor y el puerto en donde se almacena remotamente el estado de sesión. Por ejemplo, "tcpip=127.0.0.1:42424".
Este atributo se requiere cuando el mode es StateServer.

sqlConnectionString

Especifica la cadena de conexión para un Servidor SQL.
Por ejemplo, "data source=127.0.0.1;user id=sa; password=".
Este atributo se requiere cuando mode es SQLServer.
El código en sus aplicaciones es independiente del método que se utiliza para almacenar los datos en la sesión.

Identificar una sesión

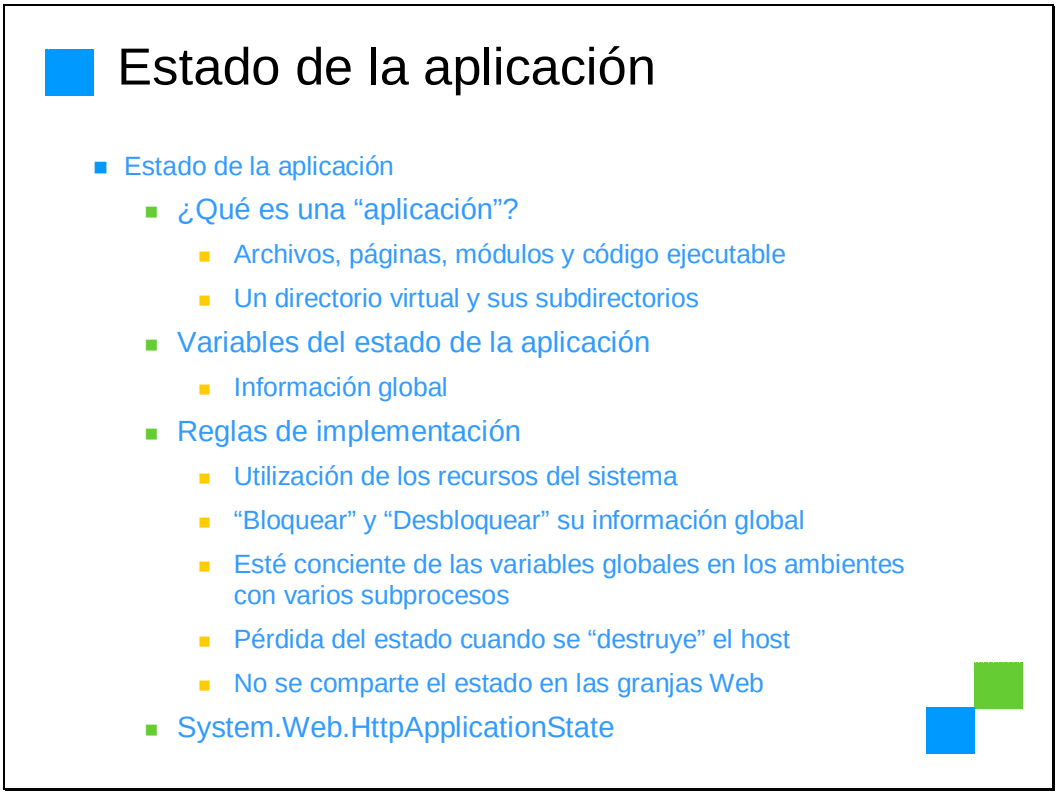
Cada sesión ASP.NET activa se identifica y rastrea utilizando una cadena SessionID de 120 bits que contiene únicamente los caracteres ASCII que se permiten en las direcciones URL.
Los valores SessionID se generan utilizando un algoritmo que garantiza que sean únicos, de manera que las sesiones no choquen, y que sean aleatorios de manera que los usuarios maliciosos no puedan utilizar una nueva SessionID para calcular la SessionID de una sesión existente.

Las cadenas SessionID se comunican entre las solicitudes del servidor del cliente, ya sea a través de una *cookie* de HTTP o una URL modificada con una cadena SessionID incrustada, dependiendo de cómo configura los ajustes de la aplicación.

Ejemplo:

```
Private Sub Page_Load(sender As Object, e As System.EventArgs)
        If Not Page.IsPostBack Then
            Session("Key1") = "Some text"
            Session("Key2") = 123
        Else
            CType(Session.Item("FirstName"), String)()
            Label1.Text = "Key1: " & CType(Session("Key1"), String)
            Label2.Text = "Key2: " & CType(Session("Key2"), String)
        End If
    End Sub
```

Diapositiva 16



ASP.NET le permite guardar valores usando el estado de la aplicación (una instancia de la clase HttpApplicationState) para cada aplicación Web activa.

El estado a la aplicación es un mecanismo de almacenaje global accesible desde todas las páginas en aplicación Web y, por lo tanto, es útil para almacenar información que se necesita mantener durante los viajes de ida y vuelta entre el servidor y las páginas.

Las diferencias principales entre los estados de Sesión y Aplicación son que la **Sesión** permite almacenar datos por usuario y la **Aplicación** es común para todos los usuarios dentro de una aplicación.

Los procesos múltiples dentro de una aplicación pueden acceder simultáneamente a valores almacenados en el estado de la aplicación. Por lo tanto, cuando vea algo que necesita acceso a valores de estado-aplicación, siempre se debe asegurar de que el objeto del estado de aplicación esté libre de procesos y lleve a cabo su propia sincronización interna o que lleve a cabo los pasos de la sincronización manual para protegerlos contra las violaciones de acceso.

La clase **HttpApplicationState** proporciona dos métodos, **Lock** y **Unlock,** que sólo le permite realizar un subproceso a la vez para acceder a las variables del estado de aplicación.

No se comparte el estado en las granjas Web (Web Farms)

El estado de aplicación no se comparte en las granjas Web (en cuya aplicación se alojan servidores múltiples, o un grupo de servidores.

Las variables almacenadas en un estado de aplicación, en cualquiera de estos escenarios, son globales únicamente para el proceso particular en el que se ejecuta la aplicación.

Cada proceso de aplicación puede tener diferentes valores. Por lo tanto, no puede confiar en el estado de la aplicación para almacenar valores únicos o actualizar contadores globales, por ejemplo, en escenarios de granjas Web o grupo de servidores.

Almacenar valores en el objeto de la aplicación

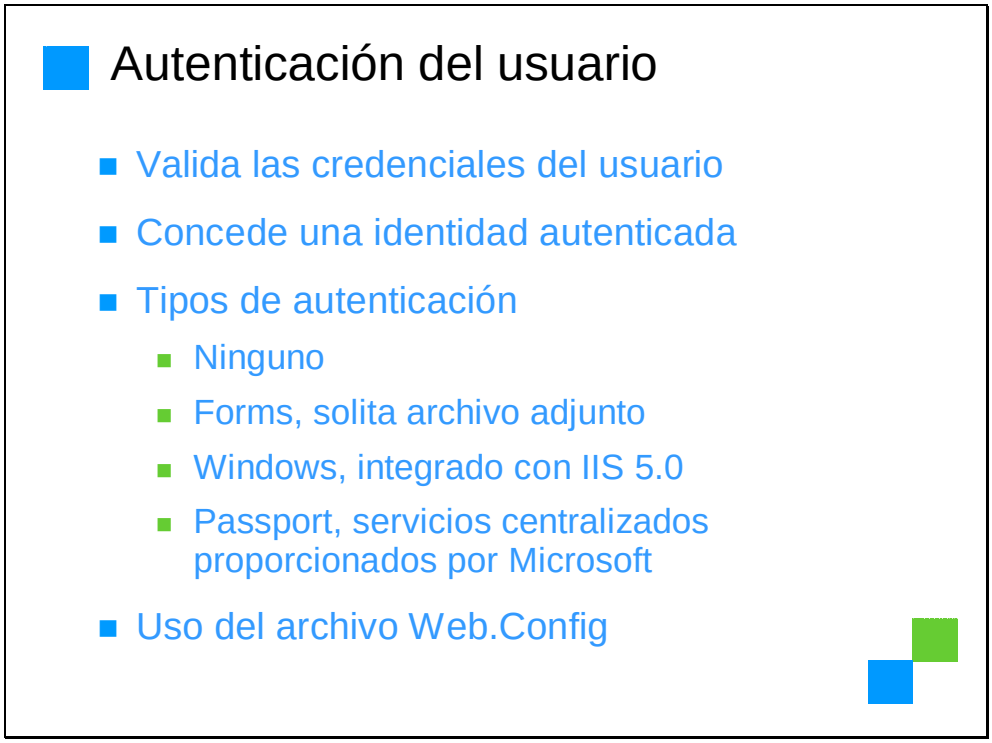
```
'VB code from within a page, a handler, or Global.asax.
Application("Message") = " MyMsg"
Application("AppStartTime") = DateTime.Now
```

Invocar Lock en el objeto Application provoca que ASP.NET bloquee los intentos a través de la ejecución del código en otro subproceso del trabajador para acceder a cualquier estado de aplicación. Estos subprocesos se desbloquean únicamente cuando el subproceso que invocó Lock invoca el método Unlock correspondiente en el objeto Application.

```
'VB code from within a page, a handler, or Global.asax.
Application.Lock()
Application("SomeGlobalCounter") =
CType(Application("SomeGlobalCounter"), Integer) + 1
Application.UnLock()
```

Si no invoca explícitamente a **Unlock**, .NET Framework eliminará automáticamente el bloqueo cuando se complete la solicitud, cuando la solicitud llegue a su fin o cuando ocurra un error no manejado durante la ejecución de la solicitud provoque que éste falle. Este desbloqueo automático evita que la aplicación quede “congelada”.

Diapositiva 17



¿Qué es una autenticación?

La autenticación confirma que los usuarios sean quienes dicen que son. Por ejemplo, un usuario proporciona un nombre de usuario y una contraseña que se verifica contra una autoridad (una base de datos, por ejemplo, o un servidor de dominio Windows).

Autenticación por formularios

La autenticación por formularios, por lo general, se refiere a un sistema de solicitudes no autenticadas que se desvían hacia un formulario HTML utilizando un redireccionador del lado del cliente HTTP.

Los formularios de autenticación son una buena opción si su aplicación necesita recolectar sus propias credenciales de usuario durante el tiempo de inicio a través de formularios HTML.

El usuario proporciona las credenciales y envía la página.

Si una aplicación autentica la solicitud, el sistema emite una cookie que contiene las credenciales o una clave para volver a adquirir la identidad.

Las solicitudes subsiguientes se emiten con la cookie en los encabezados de solicitud. Las solicitudes se autentican y autorizan a través de un manejador de eventos ASP.NET utilizando cualquier método de validación que especifique la aplicación.

Autenticación Windows

ASP.NET utiliza la autenticación Windows junto con la autenticación IIS.

La autenticación se realiza a través de IIS en una de las tres siguientes formas: **básica**, **resumen** o **autenticación Windows integrada**.

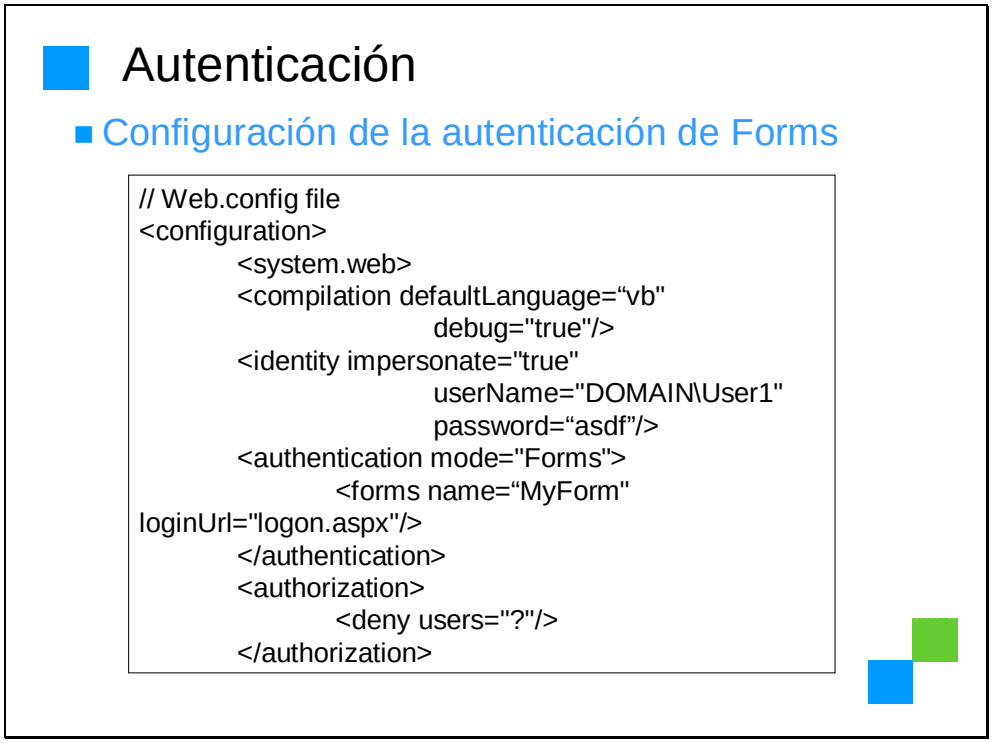
Cuando se completa la autenticación IIS, ASP.NET utiliza la identidad autenticada para autorizar el acceso.

Configuración Web.config

El nodo de autenticación en Web.Config debe estar configurado con el método de autenticación que desee utilizarse.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
        <system.web>
                <authentication mode= "[Windows/Forms/Passport/None]">
</authentication>
…
```

Diapositiva 18



El archivo de configuración de la aplicación ASP.NET está contenido en el archivo Web.config.

ASP.NET aplica los ajustes de la configuración a los recursos de una manera jerárquica.

Los archivos Web.config sustituyen la información a la configuración con los directorios en los que se ubican y con todos los directorios hijo.

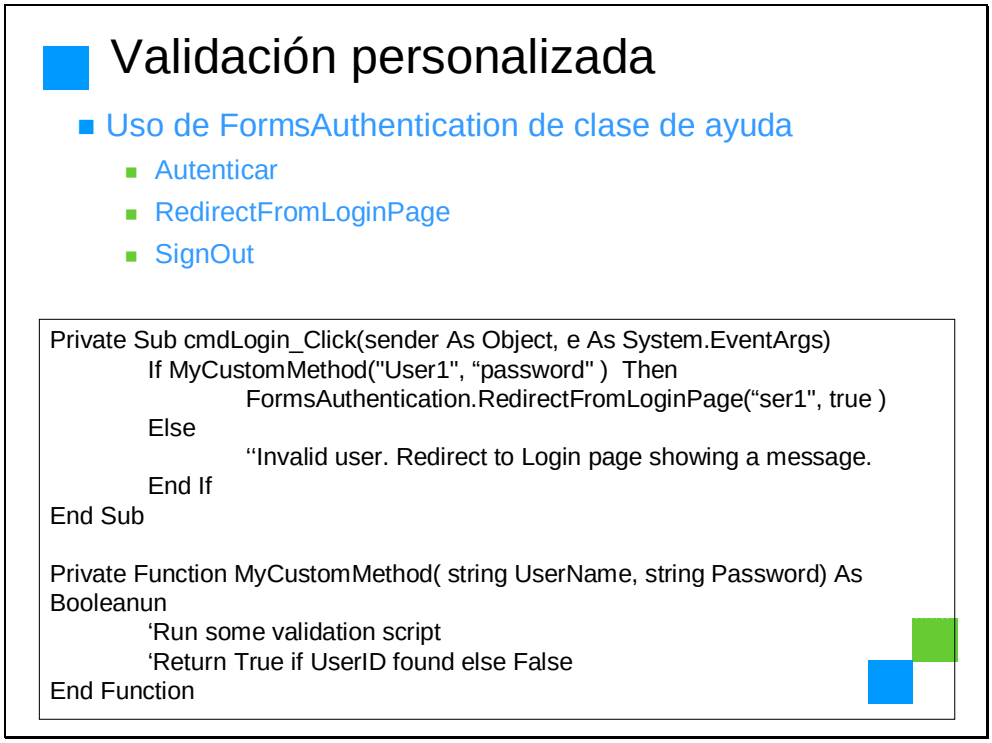
En este ejemplo, la aplicación ASP.NET adquiere la identidad de DOMAIN/User1.

La autenticación para la aplicación utiliza el método **Forms** y en específico la página Web **logon.aspx** para validar las solicitudes de inicio. Los usuarios deben estar autenticados como usuarios anónimos ("?") o de otra manera se les negará el acceso.

"Web.config file

```
<configuration>
        <system.web>
        <compilation defaultLanguage="vb"
                        debug="true"/>
        <identity impersonate="true"       userName="DOMAIN\User1"
password=""/>
        <authentication mode="Forms">
                <forms name="MyForm" loginUrl="logon.aspx"/>
        </authentication>
        <authorization>
                <deny users="?"/>
        </authorization>
…
```

Diapositiva 19



Clase de asistencia para FormsAuthentication

Proporciona métodos estáticos que sustituyen las utilidades del asistente para manipular los tickets de autenticación.

Método Authenticate

Intenta validar las credenciales en el almacén de credenciales configuradas, en el archivo Web.Config o desde una base de datos.

Método RedirectFromLoginPage

Redirecciona un usuario autenticado de vuelta a la URL solicitada originalmente.

Método SignOut

Elimina los tickets de autenticación al configurar la *cookie* de autenticación a un valor vacío. Esto elimina tanto las *cookies* duraderas como las de la sesión.

Ejemplo.

Configuración del archivo Web.Config.

```
<configuration>
        <system.web>
        <compilation defaultLanguage="vb"       debug="true"/>
        <authentication mode="Forms">
                <forms name="MyForm" loginUrl="logon.aspx"/>
        </authentication>
        <authorization>
                <deny users="?"/>
        </authorization>
…
```

Código Login.aspx page

Cuando se presiona el botón Login se ejecuta el siguiente código.

```
…
Private Sub cmdLogin_Click(sender As Object, e As System.EventArgs)
      If MyCustomMethod("User1", "password") Then
                FormsAuthentication.RedirectFromLoginPage( "User1", true )
      Else
         'Invalid user. Redirect to Login page showing a message.
      End If
   End Sub
Private Function MyCustomMethod( string UserName, string Password) As Boolean
      'Run some validation script
      'Return True if UserID found else False
   End Function
```

Acceso a otros recursos sin autenticación.

Cuando cualquier usuario trata de acceder a recursos sin una autenticación previa se redirecciona automáticamente a la página configurada en el archivo Web.Config, hasta que el usuario obtiene con éxito la autenticación.

La autenticación exitosa significa que se invocó el método FormsAuthentication.

Sign Out

El método SignOut elimina el ticket de autenticación.

Ejemplo.

```
Private Sub cmdSignOut_Click(sender As Object, e As System.EventArgs)
      FormsAuthentication.SignOut()
   End Sub
```

Diapositiva 20

Parte 2/2 Servicios Web

- Conceptos
  - Servicios Web
  - SOAP
  - WSDL
- Crear un Servicio Web
  - Atributo WebService
  - Atributo WebMethod
- Consumir Servicios Web
  - Get
  - Post
  - Soap
- Servicios Web, DCOM y .NET Remoting

Diapositiva 21

Un Servicio Web es una entidad programable que proporciona un elemento de funcionalidad en particular, como una lógica de aplicación, y es accesible a cualquier número de sistemas potencialmente distintos utilizando estándares de Internet ubicuos, como XML y HTTP.

Los Servicios Web dependen en gran medida de la amplia aceptación de XML y de otros estándares de Internet para crear una infraestructura que soporte la interoperabilidad de aplicaciones a un nivel que resuelva varios de los problemas que dificultaban anteriormente tales intentos.

Diapositiva 22

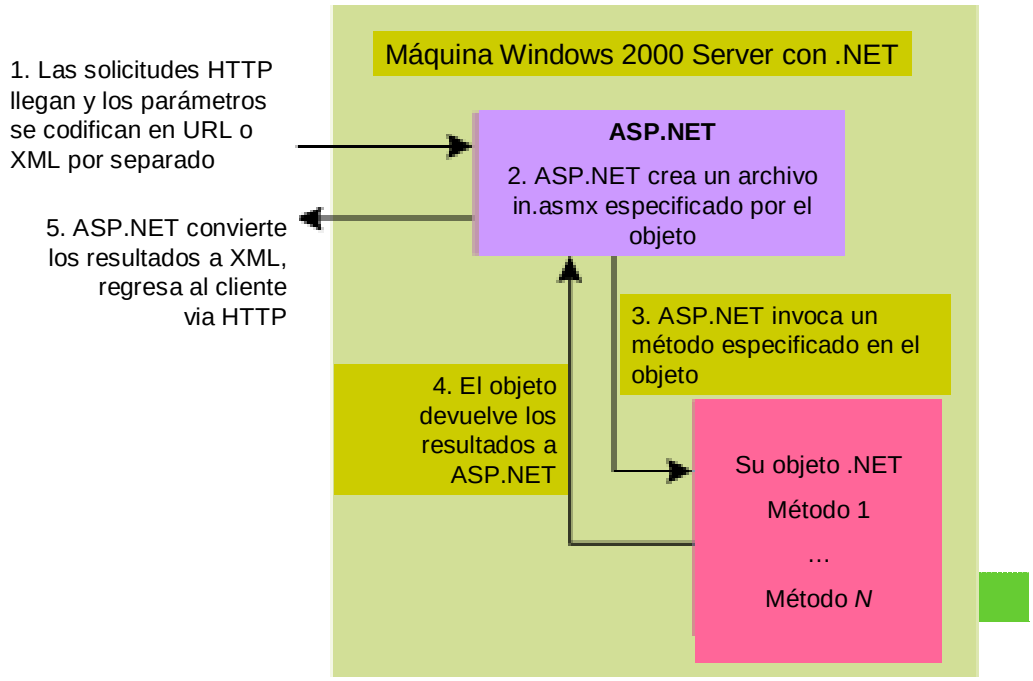
Un Servicio Web se puede utilizar ya sea de manera interna por parte de una aplicación única o exponerse externamente sobre el Internet para su uso por parte de cualquier número de aplicaciones.
Ya que es accesible a través de un protocolo estándar, un Servicio Web permite que sistemas heterogéneos trabajen en conjunto como un Web único de computación.
Los Servicios Web de ASP.NET son uno de los componentes centrales del Framework .Net y utiliza todos los servicios que proporciona el Servidor Web Internet Information Server (IIS) como la seguridad, autenticación, autorización y el proceso de control.

Unido libremente

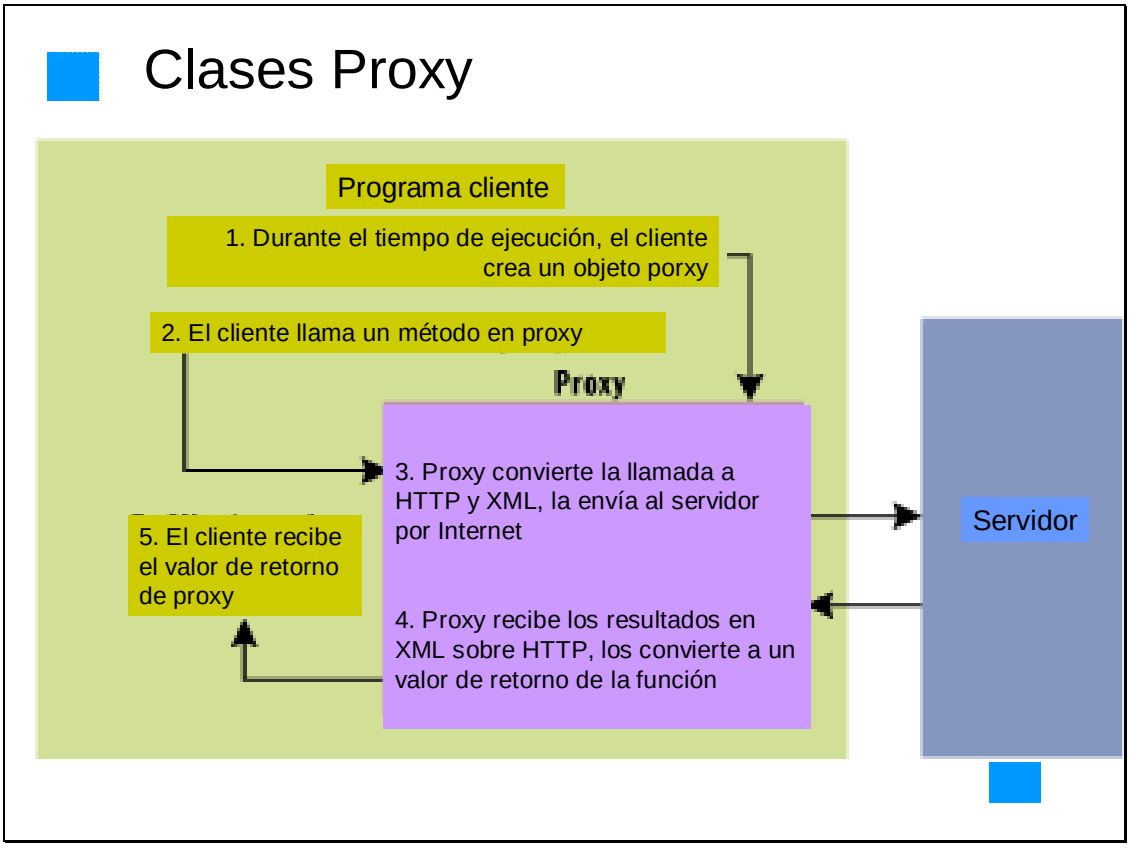
Se considera que dos sistemas están unidos libremente si el único mandato que se le impone a ambos sistemas es comprender los mensajes antes mencionados de auto-descripción y basados en texto.

Diapositiva 23

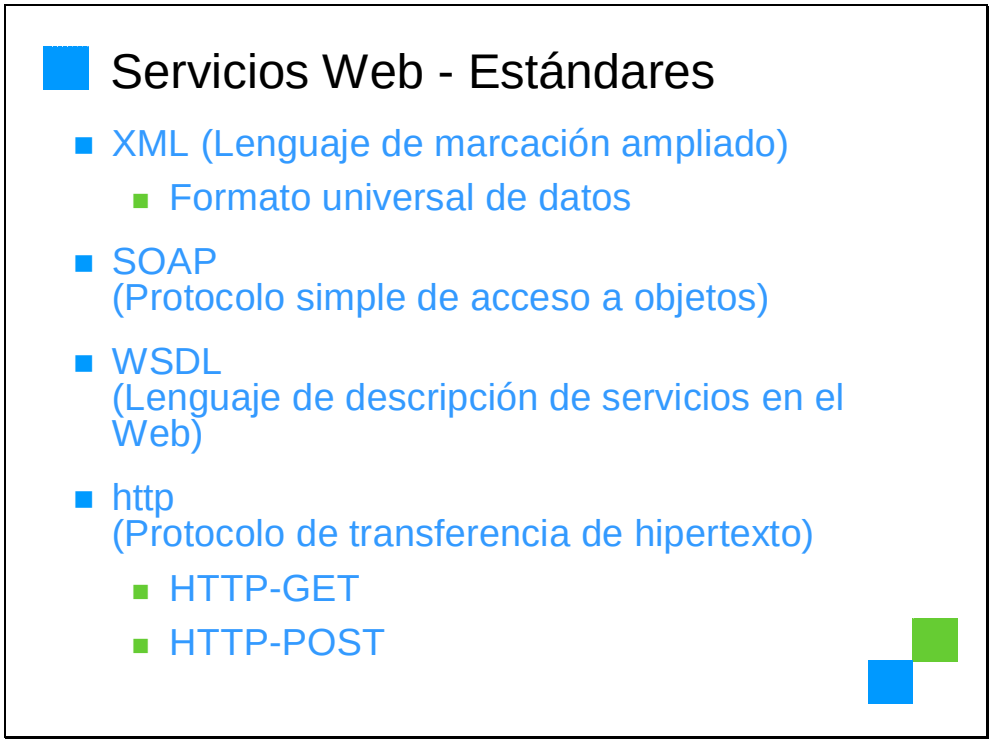
Solicitudes HTTP



Diapositiva 24



Diapositiva 25



XML

XML (Lenguaje extensible de marcadores) es una manera flexible de crear formatos de información común y compartir tanto el formato como los datos en el World Wide Web, los intranets y en cualquier parte.

Al adoptar los estándares abiertos existentes sobre los métodos de comunicación propietarios de circuito cerrado, cualquier sistema que soporta los mismos estándares abiertos puede comprender los Servicios Web.

SOAP

SOAP es un protocolo sencillo y ligero basado en XML para intercambiar información estructurada y de tipo en el Web.

La meta de diseño general de SOAP es mantenerlo tan sencillo como sea posible, y proporcionar un mínimo de funcionalidad. El protocolo define un marco de mensajes que no contiene semántica de aplicación o de transporte.

Como resultado, el protocolo es modular y muy extensible.

SOAP es una manera para que un programa que se ejecuta en un tipo de sistema operativo (como Windows 2000) se comunique con un programa en el mismo tipo u otro sistema operativo (como Linux) al utilizar el Protocolo de transferencia de hipertexto (HTTP) del World Wide Web y su Lenguaje extensible de marcadores (XML) como los mecanismos para el intercambio de información.

WSDL

El Lenguaje de descripción de servicios Web (WSDL) es un lenguaje basado en XML que se utiliza para describir los servicios que ofrece un componente y para proporcionar una forma, tanto a las personas como a otras aplicaciones, de acceder a esos servicios de manera electrónica.

HTTP

El Protocolo de transferencia de hipertexto (HTTP) es un conjunto de reglas para intercambiar archivos (texto, imágenes gráficas, sonido, video y otros archivos multimedia) en el World Wide Web.

HTTP-GET y **HTTP-POST** son protocolos estándar que utilizan verbos de HTTP (Protocolo de transferencia de hipertexto) para codificar y pasar parámetros como pares de nombres / valores, junto con la semántica asociada.

Diapositiva 26

Web.Config

Todos los ajustes de la configuración están almacenados en los archivos basados en XML llamados Web.Config.

Los desarrolladores y administradores que utilicen editores de texto pueden editarlos fácilmente.

Controla las configuraciones de los Servicios Web creados utilizando ASP.NET.

```
<configuration>
  <system.web>
    <webServices>
```

Puede almacenar diferentes archivos Web.Config en el servidor. Cada archivo de configuración afecta al directorio en el que están colocados y a todos los subdirectorios. A esto se le llama una arquitectura con configuración jerárquica.

Diapositiva 27

Archivo ASMX

Los Servicios Web tienen la extensión .asmx. Están almacenados en la ruta virtual de su aplicación Web ASP.NET. Un archivo .asmx puede ser parte de una solución existente o un archivo individual. Cuando desea ubicar un Servicio Web, debe conocer la ubicación del archivo .asmx.

Microsoft Visual Studio.NET

Microsoft Visual Studio.NET le proporciona soporte al desarrollar Servicios Web únicos y aplicaciones Web completas. Puede crear los Servicios Web utilizando el proyecto de plantillas de Servicios Web VB.NET.

Declarar Servicios XML utilizando directiva

La primera línea de su archivo .asmx debe ser la directiva WebService, en la que especifica el lenguaje de programación y el nombre de la clase del Servicio Web.

Ejemplo:

```
<%@ WebService Language="vb" Codebehind="Service1.asmx.vb"
Class="WebService1.Service1" %>
```

Definir los métodos Web XML

Para exponer una clase completa como un Servicio Web, se debe derivar de la clase base System.Web.WebServices.WebService, pero si desea que un método único de su clase esté disponible al público, debe asignar el atributo WebMethod a éste.

Ejemplo:

```
<WebMethod()> _
    Public Function HelloWorld() As String
    Return "Hello World"
End Function
```

Diapositiva 28

Directiva <%@ WebService%>

Agregue una directiva @ WebService a la parte superior de un archivo con una extensión .asmx, especificando la clase que implementa el Servicio Web y el lenguaje de programación utilizado en la implementación.

Ejemplo:

```
<%@ WebService Language="vb" Class="MyMath" %>
Imports System.Web.Services
<WebService(Namespace:="http://msdn.microsoft.com/vbasic/")>
Public Class MyMath
    Inherits System.Web.Services.WebService
    <WebMethod(Description:="Description")> _
     Public Function Add(ByVal num1 As Integer, ByVal num2 As Integer) As Integer
        Return num1 + num2
    End Function
End Class
```

En línea

Si la directiva WebService y el atributo WebMethod residen en el mismo archivo, debe agregar una directiva @ WebService en la parte superior de un archivo con una extensión .asmx, especificando la clase que implementa el Servicio Web y el lenguaje de programación utilizado en la implementación, parecido al que aparece en la diapositiva.

Ejemplo:

```
<%@ WebService Language=“vb“ Class=“MyMath“ %>
Imports System.Web.Services
<WebService(Namespace:="http://msdn.microsoft.com/vbasic/")>
Public Class MyMath
   Inherits System.Web.Services.WebService
   <WebMethod(Description:="Description")> Public Function Add(ByVal num1 As
Integer, ByVal num2 As Integer) As Integer
      Return num1 + num2
   End Function
```

Descripción

La clase de Servicio Web que define puede estar incluida directamente en el archivo .asmx o en un archivo por separado. Si utiliza un archivo por separado, éste debe estar compilado en un ensamble.

La clase de Servicio Web correspondiente tiene que estar almacenada dentro del directorio /bin de su aplicación Web.

La siguiente directiva WebService es la única línea en un archivo con una extensión .asmx.

```
<%@ WebService Language=“vb"
Class="MyNameSpace.MyWebService,MyAssembly" %>
```

MyNamespace = espacio de nombre de la clase.

MyWebservice = nombre de la clase del servicio Web.

MyAssembly = nombre del archive del servicio Web.

Diapositiva 29

Atributo <WebService>

Se utiliza para agregar información adicional a un Servicio Web, como una cadena que describe su funcionalidad.

El WebServiceAttribute no se requiere para publicar ni ejecutar un Servicio Web.

Propiedades de este atributo

Description: Un mensaje descriptivo para el Servicio Web.

Name: Nombre del Servicio Web.

Namespace: El espacio de nombre XML predeterminado para el Servicio Web.

TypeId: Cuando se implementa en una clase derivada, obtiene un identificador único para este Atributo.

Atributo <WebMethod>

Todos los Servicios Web se deben heredar desde la clase base WebService. Para hacer que un método único de su clase de Servicio Web esté disponible al público, asigne el atributo WebMethod a éste.

Propiedades de este atributo

BufferResponse: Obtiene o establece si la respuesta para esta solicitud está almacenada en el búfer.

CacheDuration: Obtiene y establece el número de segundos en la que se debe mantener la respuesta en el caché.

Description: Un mensaje descriptivo sobre el método del servicio Web.

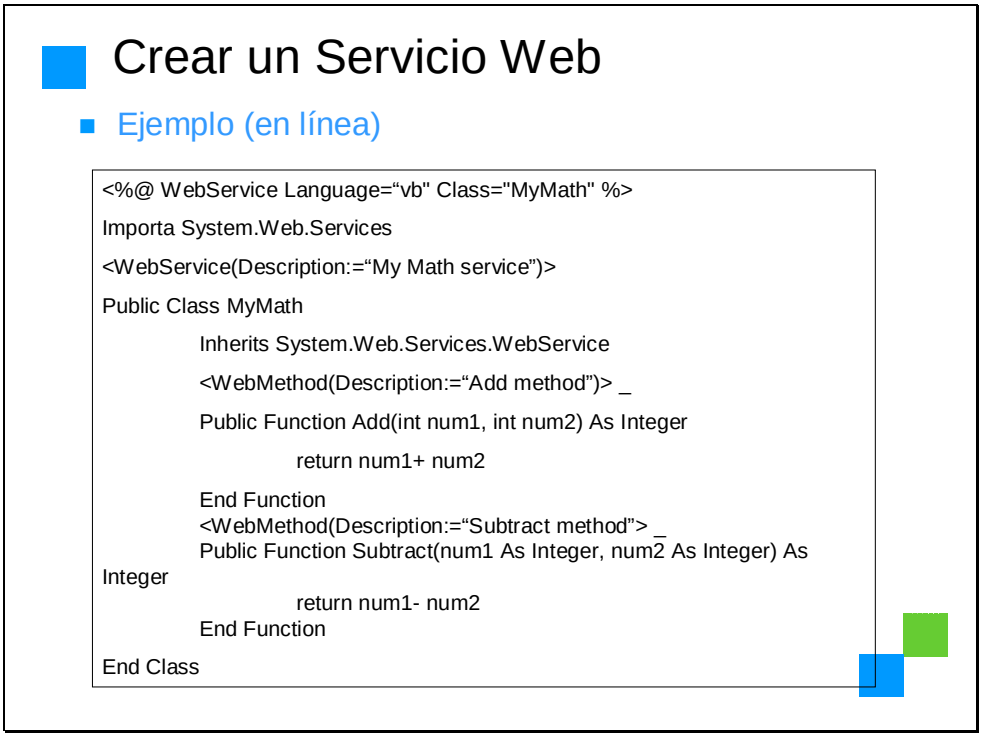
EnableSession: Indica si el estado de la sesión está habilitado para un método de Servicio Web.

MessageName: El nombre que se utiliza para el método del servicio Web en los datos que se enviaron y devolvieron de un método de servicio Web.

TransactionOption: Indica el soporte a la transacción de un método de Servicio Web.

TypeId: Cuando se implementa en una clase derivada, obtiene un identificador único para este Atributo.

Diapositiva 30



Este ejemplo muestra el Servicio Web MyMath.

Este Servicio Web contiene dos métodos Web Add y Subtract.

El atributo WebService proporciona una descripción del Servicio Web.

El atributo WebMethod proporciona una descripción del método Web e indica que el método puede invocarse de manera remota.

```
<%@ WebService Language="vb" Class="MyMath" %>
Imports System.Web.Services
<WebService(Description:="My Math service")>
Public Class MyMath
   Inherits System.Web.Services.WebService
        <WebMethod(Description:="Add method")> _
        Public Function Add(int num1, int num2) As Integer
      Return num1 + num2
   End Function
        <WebMethod(Description:="Subtract method")> _
        Public Function Subtract(num1 As Integer, num2 As Integer) As Integer
                return num1- num2
        End Function
End Class
```

Diapositiva 31

Implementar un Servicio Web

Implementar un Servicio Web involucra copiar el archivo .asmx y cualquier ensamble utilizado por el Servicio Web, al servidor Web.

Crear una clase Proxy

Cualquier desarrollador que desee utilizar un Servicio Web publicado debe crear una clase Proxy para incluirla en el programa o para utilizarla como un ensamble externo.

Si utiliza Visual Studio.Net, la clase Proxy y su implementación se generan automáticamente cuando utiliza el asistente Add Web Reference. De lo contrario, la aplicación wsdl.exe incluida en el SKD de .Net Framework generará la clase Proxy por usted, con base en el documento WSDL del Servicio Web.

Contrato de publicación WSDL

No tiene que implementar código adicional o configurar su servidor para publicar una descripción de su Servicio Web y de su funcionalidad. Si un cliente conoce el URL base de su servicio Web, se puede recibir una descripción HTML sencilla del servicio al introducir el URL sin ningún nombre de método o parámetro.

Diapositiva 32

Dirigible a un URL

Los Servicios Web se pueden dirigir a un URL, lo que significa que se pueden invocar al utilizar solicitudes HTTP. El método y sus argumentos se incluyen en la solicitud.

Protocolos

Los Servicios Web se pueden invocar utilizando los siguientes esquemas de protocolo.

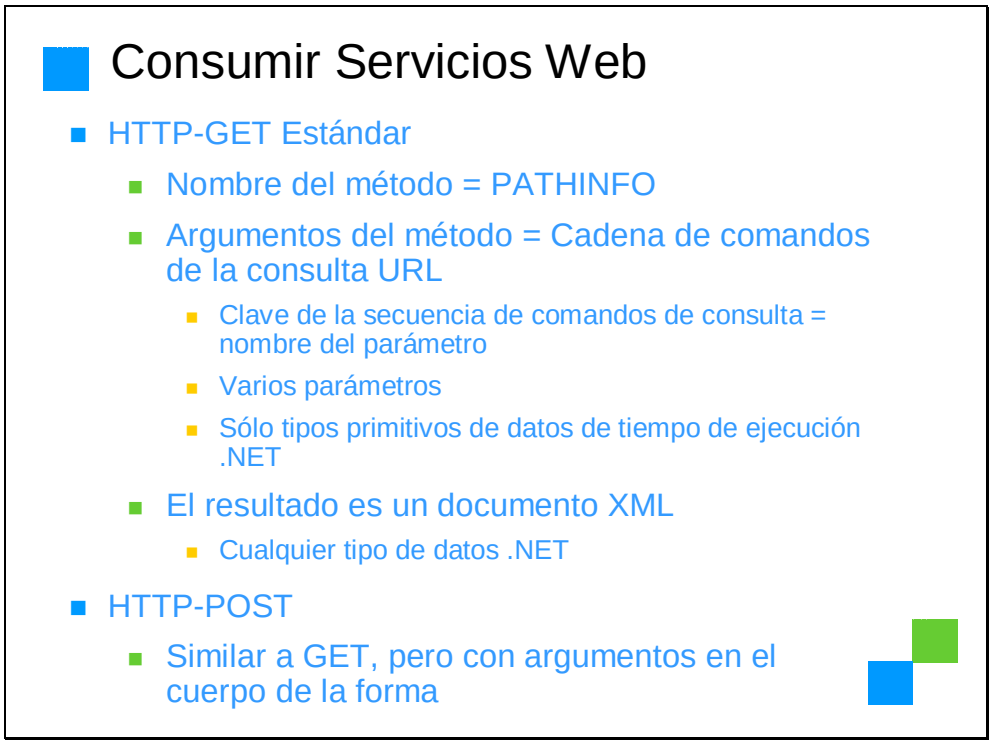
- **HTTP-GET**
- **HTTP-POST**
- **HTTP-SOAP**

Cuando utiliza HTTP-GET, el nombre del método y los argumentos son parte del URL.

Al utilizar HTTP-POST, el nombre del método es también parte del URL. Sin embargo, los argumentos son parte del cuerpo del POST.

SOAP define la gramática de XML para identificar el nombre del método, envolver sus argumentos y devolver los resultados

Diapositiva 33



Sintaxis de ubicación: http://servername/apppath/webservicename.asmx

ServerName = Nombre de servidor Web.

Apppath = Nombre del directorio virtual.

Webservicename = Nombre del archivo que contiene un método Web.

Usted invoca un método Web Service. PATHINFO del mensaje de solicitud identifica el nombre del método. PATHINFO es la información adicional de la ruta para un recurso con una extensión URL. Se envían argumentos como valores de cadenas de consulta, donde la clave de cadena de consulta especifica el nombre del parámetro. Se pueden enviar múltiples parámetros del tipo de datos de tiempo de ejecución .NET primitivos.

Ejemplo

```
http://servername/apppath/webservicename.asmx/WebMethodName?param1=valu
e1&paramN=valueN...
```

URL = http://servername/apppath/webservicename.asmx

PATHINFO = /WebMethodName

PARAMETERS = param1=value1&paramN=valueN...

Para invocar el Servicio Web MyMath debe utilizar esta oración.

http://servername/apppath/webservicename.asmx**/Add?num1=4&num2=5**

HTTP-POST y HTTP-GET

HTTP-GET envía sus parámetros en forma de un texto codificado en URL (urlencoding) utilizando el tipo *application/x-www-form-urlencoded* de MIME, que se incluye en el URL del servidor que maneja la solicitud.

Urlencoding es una manera de codificar los caracteres para asegurar que los parámetros que se enviaron sean consistentes con el texto que los conforma, como codificar un espacio como **%20**.

Los parámetros que se agregan también se conocen como una cadena de consulta (QueryString).

De manera similar, los parámetros HTTP-GET, HTTP-POST también son urlencoded. Sin embargo, en lugar de enviarlos como parte del URL, los pares nombre / valor se envían dentro del mensaje de solicitud HTTP real.

Diapositiva 34

Gramática XML

SOAP proporciona una gramática de XML para especificar el método de Servicio Web, enviando sus parámetros y devolviendo los resultados.

Soporta todos los tipos estándar de datos .NET y las clases de valor

Todos los tipos que están soportados por HTTP-GET y POST también están soportados con SOAP. Además, las clases, estructuras y conjuntos de datos se pueden enviar como argumentos.

Clasificación de clases y estructuras

Todos los campos y propiedades públicos de una clase están serializados en formato XML. Esta es la manera en que se proporciona el orden de las clases y de las estructuras.

Serialización

La serialización es el proceso de convertir un objeto en una forma que se pueda transportar fácilmente.

Por ejemplo, puede serializar un objeto y transportarlo a través de Internet utilizando HTTP entre un cliente y un servidor. Por otro lado, la deserialización reconstruye el objeto desde el flujo.

Ejemplo:

```
Public Class OrderForm
        public DateTime OrderDate
End Class
```

Cuando una instancia de esta clase se serializa, debe parecerse a lo siguiente:

```
<OrderForm>
        <OrderDate>12/12/01</OrderDate>
</OrderForm>
```

Diapositiva 35

HTTP-GET

Con el ejemplo MyMath debe invocar el Servicio Web utilizando este formato:

Solicitud de servicio

```
http://MyHost/MyVirtualDirectory/Math.asmx/Add?num1=2&num2=2
```

Respuesta del servicio

```
<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">4</int>
```

HTTP-POST

Con el mismo ejemplo.

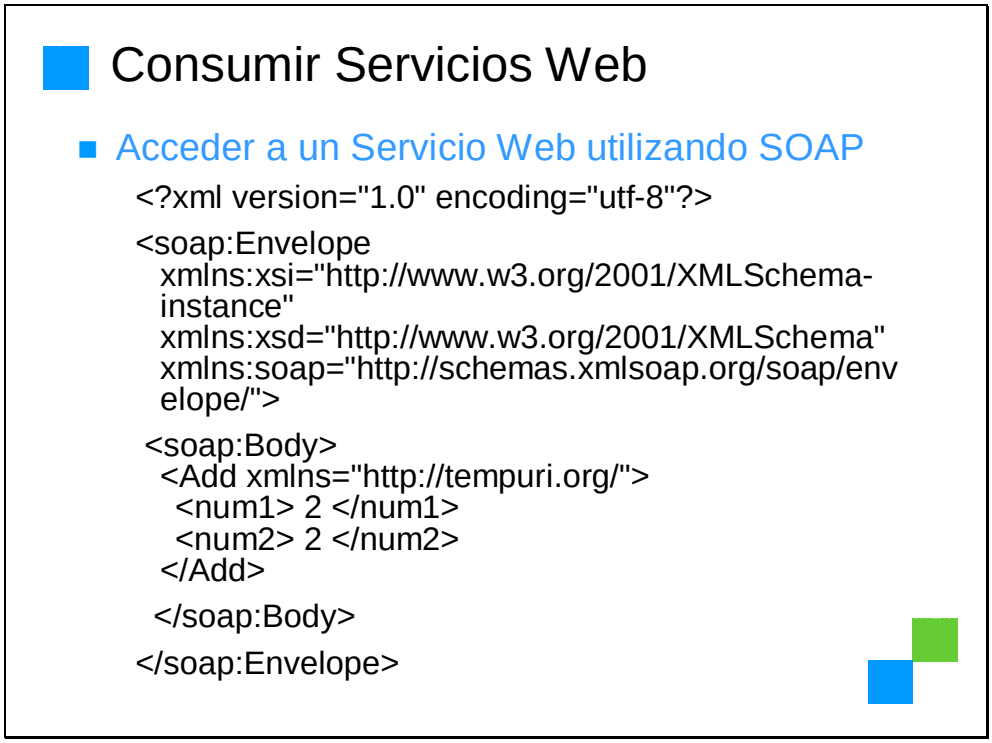
Solicitud de servicio

```
<form method=POST action='http://MyHost/MyVirtualDirectory/Math.asmx/Add'>
        <input type="text" size="3" name='num1'\"></td> -
        <input type="text" size="2" name='num2'\"></td> =
        <input type=submit value="Add"> </td>
</form>
```

Respuesta del servicio

```
<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">1</int>
```

Diapositiva 36



HTTP-SOAP

Comunicarse con un método de Servicio Web utilizando SOAP sigue un formato estándar.

Parte de este formato son los datos que están codificados en un documento XML.

El documento XML consiste en una etiqueta raíz *Envelope*, que a su vez consiste en un elemento *Body* requerido y un elemento opcional *Header*.

El elemento Body consta de datos específicos para el mensaje.

El elemento opcional Header puede contener información adicional que no está directamente relacionada con el mensaje en particular. Cada elemento hijo del elemento Header recibe el nombre de encabezado de SOAP.

Visual Studio .Net extrae los detalles de los Servicios Web de consumo ya que utiliza una clase Proxy que encapsula la funcionalidad de un servicio Web.

Con el Servicio Web MyMath.

Solicitud de servicio

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://tempuri.org/">
      <num1>2</num1>
      <num2>2</num2>
    </Add>
  </soap:Body>
</soap:Envelope>
```

El mensaje SOAP es un documento XML que consiste en una envoltura SOAP obligatoria, un encabezado opcional SOAP y un cuerpo obligatorio SOAP.

El nodo Add en Body denota el método Web y encierra los parámetros.

Respuesta del servicio

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>4</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

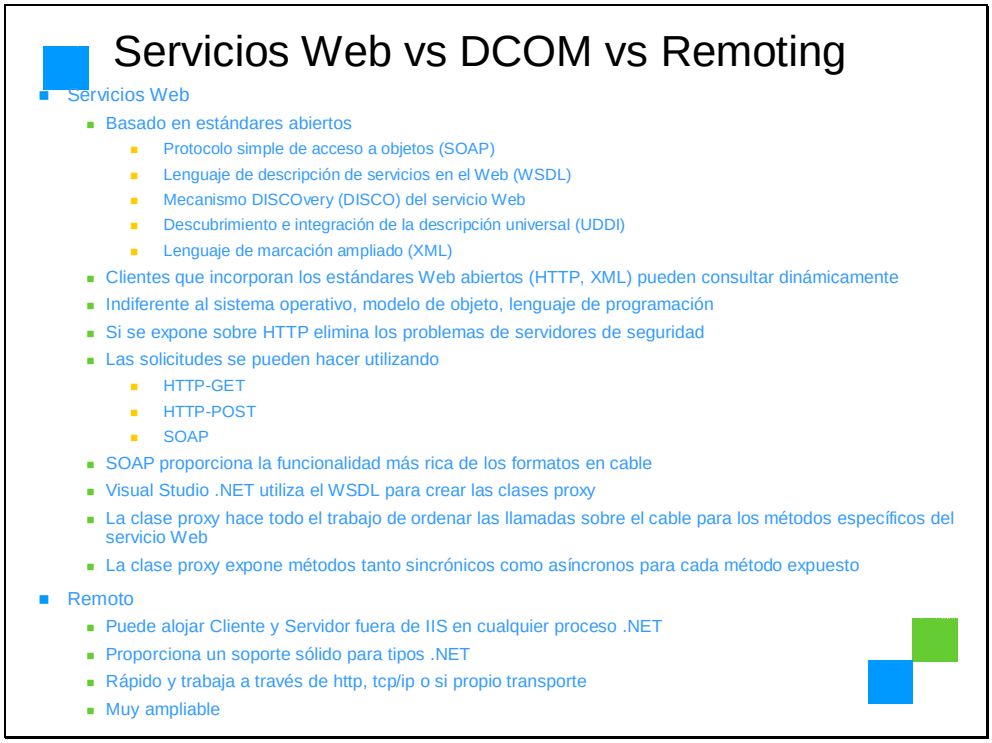
Diapositiva 37

Servicios Web vs DCOM vs Remoting

DCOM

- Basado en DCOM en el estándar Ambiente de cómputo distribuido (DCE) de la Llamada de procedimiento remoto (RPC)
- Permite la creación de objetos de servidor en las máquinas remotas
- DCOM es una extensión del Modelo de objeto de componentes (COM)
- DCOM se basa en el registro y en las bibliotecas COM para información de la biblioteca de tipo
- Sin mecanismo para actualizar y unirse dinámicamente a la información de biblioteca de tipo. DLL HELL!
- Normalmente, la configuración se hace en el Cliente y Servidor
- COM+ y MTS facilitan las cosas (activación, proxy)
- DCOM es un protocolo “chatty”
- Servidores de seguridad problemáticos
- DCOM sigue siendo ineficiente, voluminoso de implementar y requiere una cantidad importante de mantenimiento manual

Diapositiva 38



Servicios Web

- Basado en estándares abiertos
  - Protocolo simple de acceso a objetos (SOAP)
  - Lenguaje de descripción de servicios en el Web (WSDL)
  - Mecanismo DISCOvery (DISCO) del servicio Web
  - Descubrimiento e integración de la descripción universal (UDDI)
  - Lenguaje de marcación ampliado (XML)
- Clientes que incorporan los estándares Web abiertos (HTTP, XML) pueden consultar dinámicamente
- Indiferente al sistema operativo, modelo de objeto, lenguaje de programación
- Si se expone sobre HTTP elimina los problemas de servidores de seguridad
- Las solicitudes se pueden hacer utilizando
  - HTTP-GET
  - HTTP-POST
  - SOAP

- SOAP proporciona la funcionalidad más rica de los formatos en cable
- Visual Studio .NET utiliza el WSDL para crear las clases proxy
- La clase proxy hace todo el trabajo de ordenar las llamadas sobre el cable para los métodos específicos del servicio Web
- La clase proxy expone métodos tanto sincrónicos como asíncronos para cada método expuesto

Remoto

- Puede alojar Cliente y Servidor fuera de IIS en cualquier proceso .NET
- Proporciona un soporte sólido para tipos .NET
- Rápido y trabaja a través de http, tcp/ip o si propio transporte
- Muy ampliable

Diapositiva 39

Servicios Web vs DCOM vs Remoting

- La siguiente tabla resalta las diferencias y capacidades entre las dos tecnologías:

Capacidad	Servicio Web	Remoto
Invoca un método único en un objeto	Sí	Sí
Invoca varios métodos en un objeto con estado completo	No	Sí
Hace que todos los clientes invoquen los métodos en el *mismo* objeto del lado del servidor	No	Sí
Pasa a través de servidor de seguridad	Sí	Sí
Utiliza HTTP para comunicación	Sí	Sí
Utiliza el socket TCP sin procesar para comunicación	No	Sí
Utiliza IIS como host	Sí	Sí
Permite un host personalizado	No	Sí
Utiliza un formato de datos que cumple con SOAP	Sí	Sí
Utiliza formateo binario más pequeño de datos	No	Sí
Recupera una copia parcial de los *datos* de los objetos complejos	Sí	Sí
Recupera una copia *completa* de los objetos complejos	No	Sí

Capacidad	Servicio Web	Remoto
Invoca un método único en un objeto	Sí	Sí
Invoca varios métodos en un objeto con estado completo	No	Sí
Hace que todos los clientes invoquen los métodos en el *mismo* objeto del lado del servidor	No	Sí
Pasa a través de servidor de seguridad	Sí	Sí
Utiliza HTTP para comunicación	Sí	Sí
Utiliza el socket TCP sin procesar para comunicación	No	Sí
Utiliza IIS como host	Sí	Sí
Permite un host personalizado	No	Sí
Utiliza un formato de datos que cumple con SOAP	Sí	Sí
Utiliza formateo binario más pequeño de datos	No	Sí
Recupera una copia parcial de los *datos* de los objetos complejos	Sí	Sí
Recupera una copia *completa* de los objetos complejos	No	Sí

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet10232001.asp

Diapositiva 40

Resumen

- Los servicios Web XML proporcionan flexibilidad
- Visual Studio .NET facilita crear y consumir los Servicios Web
- Acceder a un método de Servicio Web es tan fácil como con COM, sin DLL Hell
- Remoting es el reemplazo de DCOM
- .NET Remoting es otra alternativa al exponer que ninguno de los clientes .NET es importante.

Diapositiva 41

Lecturas recomendadas

- Walther. Inicio de ASP.NET. Sams, 2003
- Beres, Evjen. *Visual Basic .NET Bible*. Wiley, 2002
- Walther. ASP.NET Unleashed. Sams, 2002
- Tagliaferri. Visual Basic.NET Codemaster's Library. Sybex, 2002.
- Short. Building XML Web Services for the Microsoft .NET Platform. Microsoft Press, 2002.
- WWW.ASP.NET
- WWW.GOTDOTNET.COM
- WWW.DOTNETJUNKIES.COM
- Especificación XML
  http://www.w3.org/XML
- Especificación SOAP
  http://www.w3.org/2000/xp/Group
- WSDL
  http://www.w3.org/2002/ws/desc
- WebForms
  http://msdn.microsoft.com/library/en-us/vbcon/html/vboriintrotowebforms.asp?frame=true

- Servicios Web
  http://www.w3.org/2002/ws
- Portal de la industria de servicios Web
  http://www.webservices.org
- Administración de estados
  http://msdn.microsoft.com/library/en-us/vbcon/html/vbconchoosingserverstateoption.asp?frame=true
- Seguridad
  http://msdn.microsoft.com/library/en-us/cpguide/html/cpconaspnetwebapplicationsecurity.asp?frame=true
- Remoting
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet10232001.asp
- Remoting
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnadvnet/html/vbnet05272003.asp