Célula Académica
UABC-Live .net

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

http://uabc-live-net.spaces.live.com/

Sesión No. 6

Introducción al desarrollo de aplicaciones Web con ASP.NET

Expositor:

Carlos Alberto Cabrera González (carlos5686@hotmail.com)

Fecha: 26 de Octubre de 2006

Programa Microsoft Desarrollador Cinco Estrellas

Estrella 2
Introducción al desarrollo de aplicaciones Web con ASP.NET

Objetivo

Conocer los elementos involucrados en el desarrollo de una aplicación web con Visual Studio 2005 y la plataforma Microsoft .NET, presentando las novedades introducidas al respecto en la nueva versión 2.0

Prerrequisitos

- Haber cursado y aprobado el módulo correspondiente a la Estrella 1 del programa Desarrollador 5 Estrellas
- Para realizar los ejercicios y ver los ejemplos de código adjunto necesita:
  - Herramienta de Desarrollo
    - MS Visual Studio 2005 ó
    - MS Visual Web Developer 2005 Express
  - Base de Datos
    - MS SQL 2005 Express Edition

Temario (1/2)

- Introducción a ASP.NET
- Formularios Web (Web Forms)
- Configuración
- Autenticación
- Como mantener el estado en una aplicación Web

Temario (2/2)

- Master Pages
- Themes y Skins
- Navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear una aplicacion Web en Visual Studio 2005
- Referencias

Temario

- Introducción a ASP.NET
  - Aplicaciones Web
  - ASP.NET
  - Servidor Web
- Formularios Web (Web Forms)
- Configuración
- Autenticación
- Como mantener el estado en una aplicación web

Aplicaciones Web – ASP.NET

- Una aplicación web es un conjunto de páginas HTML que se transmiten por medio del protocolo HTTP de un servidor al cliente y viceversa, brindando distintas funcionalidades a un usuario final.

- ASP.NET es un “Marco” (framework) para programar aplicaciones web, de un modo similar al que se programan las aplicaciones windows. El componente principal son los Web Forms (formularios web) que permiten, entre otras cosas, separar la interfaz del usuario de la funcionalidad de la aplicación.

Aplicaciones Web - Servidor Web

- Un servidor web es un sistema informático conectado a una red, donde se almacenan las páginas, imágenes, etc. (que forman una aplicación web) disponibles para ser visitadas por los usuarios de la red.

- Internet Information Server (IIS), es el servidor Web de Microsoft que corre sobre plataformas Windows. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS

Hypertext Transfer Protocol (HTTP)

- Uno de los protocolos más importantes de Internet
- HTTP define como los navegadores y los servidores Web se comunican uno con otro
- Esta basado en texto y es transmitido sobre conexiones TCP

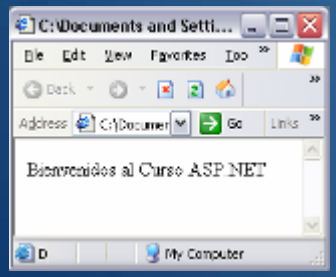
Funcionamiento de HTTP

Cliente

http://www.cursoaspnet.com/inicio.html

Internet DNS

IP=66.45.26.25 Puerto: 80



HTTP Request

HTTP Response

Servidor

www.cursoaspnet.com

IP = 66.45.26.25

```
inicio.html
<html>
<body>
Bienvenidos al
Curso ASP.NET
</body>
</html>
```

Pagina solicitada

inicio.html

```
<html>
    <body>
        Bienvenidos al
        Curso ASP.NET
    </body>
</html>
```

HTTP Request

```
GET /inicio.html HTTP/1.1
Accept: */*
Accept-Language:...
Accept-Encoding:...
If-Modified-Since:...
If-None-Match:...
User-Agent: Mozilla/4.0...
Host: www.cursoaspnet.com
Connection: Keep-Alive
[blank line]
```

HTTP Response

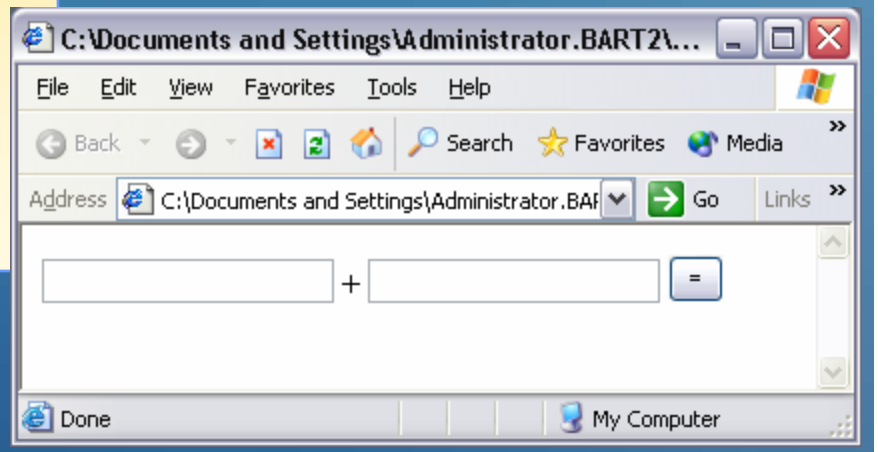
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: ...
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: ...
ETag: ...
Content-Length: 46
[blank line]
<html>
<body>
Bienvenidos al Curso ASP.NET
</body>
</html>
```

HTML Forms

- En el corazón de toda aplicación Web genuina están los HTML Forms
- Un HTML Form es la porción de un documento HTML que aparece entre las etiquetas <form></form>

Suma.html

```
<html>
  <body>
    <form>
      <input type="text" name="op1" />
      +
      <input type="text" name="op2" />
      <input type="submit" value="  =  " />
    </form>
  </body>
</html>
```



HTML Forms (Cont.)

- Un botón **submit** (<input type="submit">) juega un rol especial en un HTML Form:
  - Cuando es pulsado, el navegador envía el HTML Form junto con cualquier entrada de datos del usuario al servidor Web
- Cómo el HTML Form es enviado, dependerá del atributo **Method** del form:
  - Si el atributo **Method** del form no está presente o tiene el valor **GET**, el navegador enviará al servidor un comando **HTTP GET**
  - Si el atributo **Method** del form tiene el valor **POST**, el navegador enviará al servidor un comando **HTTP POST**

HTML Forms (Cont.)

Method = GET

```
<form method="get">
...
</form>
```

```
GET /suma.html?op1=2&op2=2 HTTP/1.1
  .
  .
  .
Connection: Keep-Alive
[blank line]
```

El navegador envia los datos ingresados como una cadena de consulta

Method = POST

```
<form method="post">
...
</form>
```

```
POST /suma.html HTTP/1.1
  .
  .
Content-Type: ...
Content-Length: 11
[blank line]
op1=2&op2=2
```

El navegador envia los datos ingresados en el cuerpo de la solicitud HTTP

Cualquiera sea el método utilizado, es decir **GET** o **POST**, cuando un form es enviado al servidor, decimos que se produjo un **POSTBACK**

Procesamiento en el Servidor

- Construir la parte del cliente es “fácil”, sólo es HTML
- La parte difícil es la construcción de la lógica del lado del servidor. “Algo en el servidor”, tiene que interpretar las entradas del usuario enviadas junto con el form y generar la correspondiente salida.

Suma.html

```
<html>
 <body>
  <form>
   <input type="text" name="op1" />
   +
   <input type="text" name="op2" />
   <input type="submit" value="  =  " />
  </form>
 </body>
</html>
```

Antes del procesamiento

Después del procesamiento

Suma.html

```
<html>
 <body>
  <form>
    <input type="text" name="op1“ value=“2”/>
    +
    <input type="text" name="op2“ value=“2”>
    <input type="submit" value="  =  " />
    4
  </form>
 </body>
</html>
```

Procesamiento en el Servidor

- Existen varias tecnologías de procesamiento:
  - **CGI** (Common Gateway Interface)
    - Define una API de bajo nivel
    - Popular en entornos UNIX, no tanto en Windows
  - **ISAPI** (Internet Server Application Programming Interface)
    - Son DLL Windows que “corren” bajo IIS. Escritas en C++
    - Mejor performance que CGI
  - **ASP** (Active Server Pages)
    - Simple solución: HTML + Script del lado del servidor
    - Programadas en JScript o VBScript
    - Objetos intrínsecos que abstraen detalles de bajo nivel de HTTP. Objetos Request y Response
    - Permite usar ADO (ActiveX Data Object) para acceso a datos

ASP.NET

- ASP.NET es el framework de programación web dentro de .NET
- Permite desarrollar aplicaciones Web con un modelo “similar” al utilizado para aplicaciones Windows
- El componente fundamental de ASP.NET es el WebForm
- Independencia del cliente (navegador, S.O., dispositivo físico, etc.)
- Permite utilizar cualquier lenguaje .NET
- Permite desarrollar Servicios Web XML

ASP.NET - Ventajas

- La “parte ejecutable” de una aplicación ASP.NET es COMPILADA
- Implementación y actualización de las aplicaciones sin reiniciar el servidor!
- Acceso a toda la .NET Class Library
- Independiente del lenguaje de programación
- Encapsulamiento de funcionalidad a través de controles de servidor y controles de usuario

ASP.NET – Ventajas (Cont.)

- Permite usar ADO.NET para acceso a datos
- Soporta XML, Hojas de estilo CSS, etc.
- Detección automática del navegador cliente, generando el lenguaje de marcas soportado por el mismo
- Mecanismo de Caching incorporado para páginas completa o partes de la misma frecuentemente solicitadas

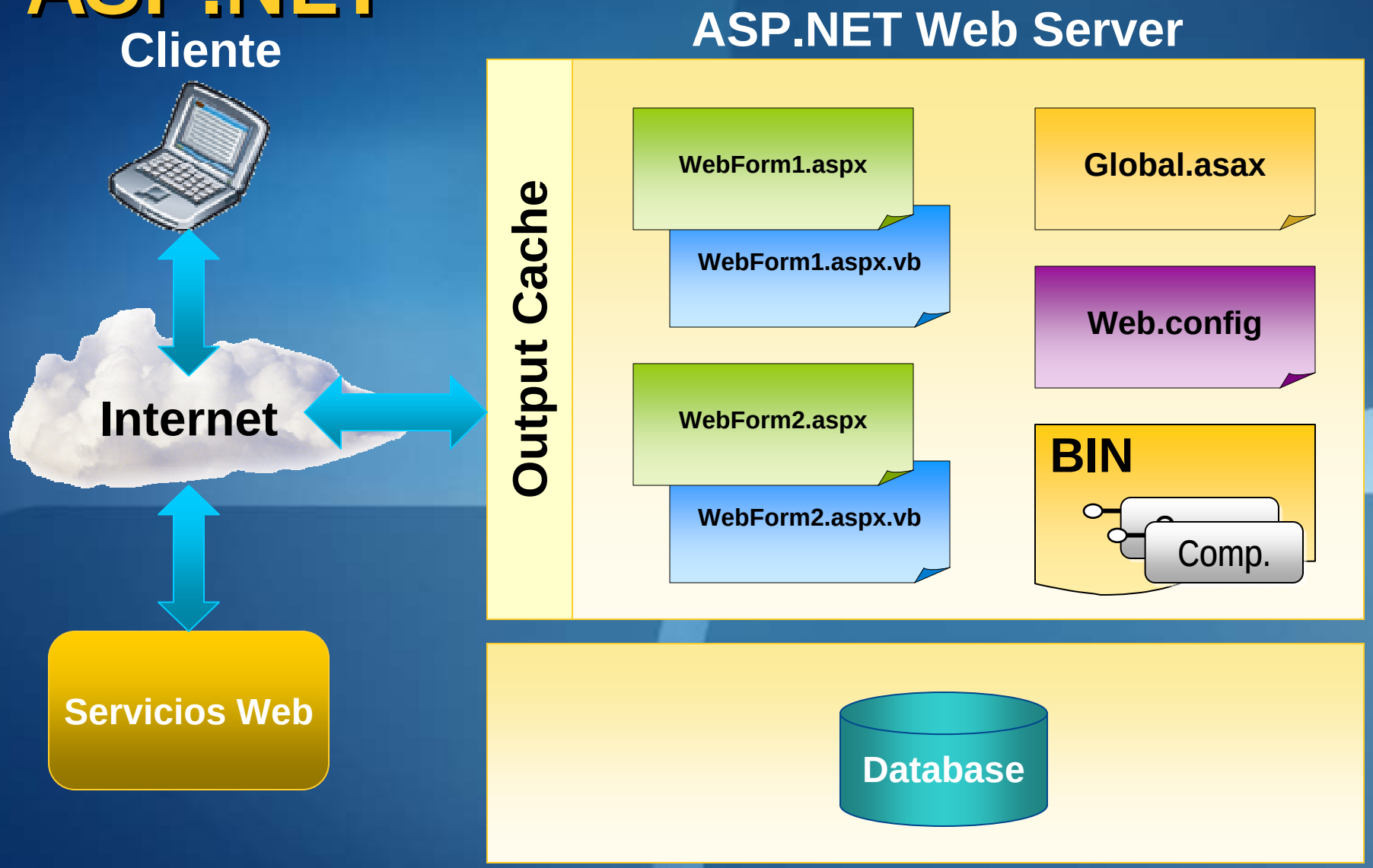
Componentes de una aplicación ASP.NET

- WebForms (Formularios Web)
  - Uno o más archivos con extensión .aspx
- Archivos Code-Behind
  - Archivos asociados a WebForms que contienen código del lado del servidor (Ej. VB.NET, C#, etc.)
- Archivos de configuración con formato XML
  - Un archivo Web.config por c/aplicación
  - Un único archivo Machine.config por servidor
- Global.asax
  - Eventos a nivel de aplicación

Componentes de una aplicación ASP.NET

- Directorio BIN
  - Contiene el assembly de la aplicación (Ej.: MiAplic.dll)
  - Cero o más assemblies (Componentes externos)
- Enlaces a Servicios Web XML
  - Permiten a la aplicación ASP.NET enviar y recibir datos desde Servicios Web

Componentes de una aplicación ASP.NET



Las aplicaciones Web ASP.NET + IIS

- IIS es el servidor Web de la plataforma Windows
- Las aplicaciones Web solo pueden existir en una ubicación que es publicada por IIS como un Directorio Virtual
- Directorio Virtual: es un recurso compartido identificado por un alias y que representa una ubicación física en el servidor
- El famoso http://localhost hace referencia al directorio raíz del servidor web
- Por default, http://localhost "apunta" a C:\Inetpub\wwwroot

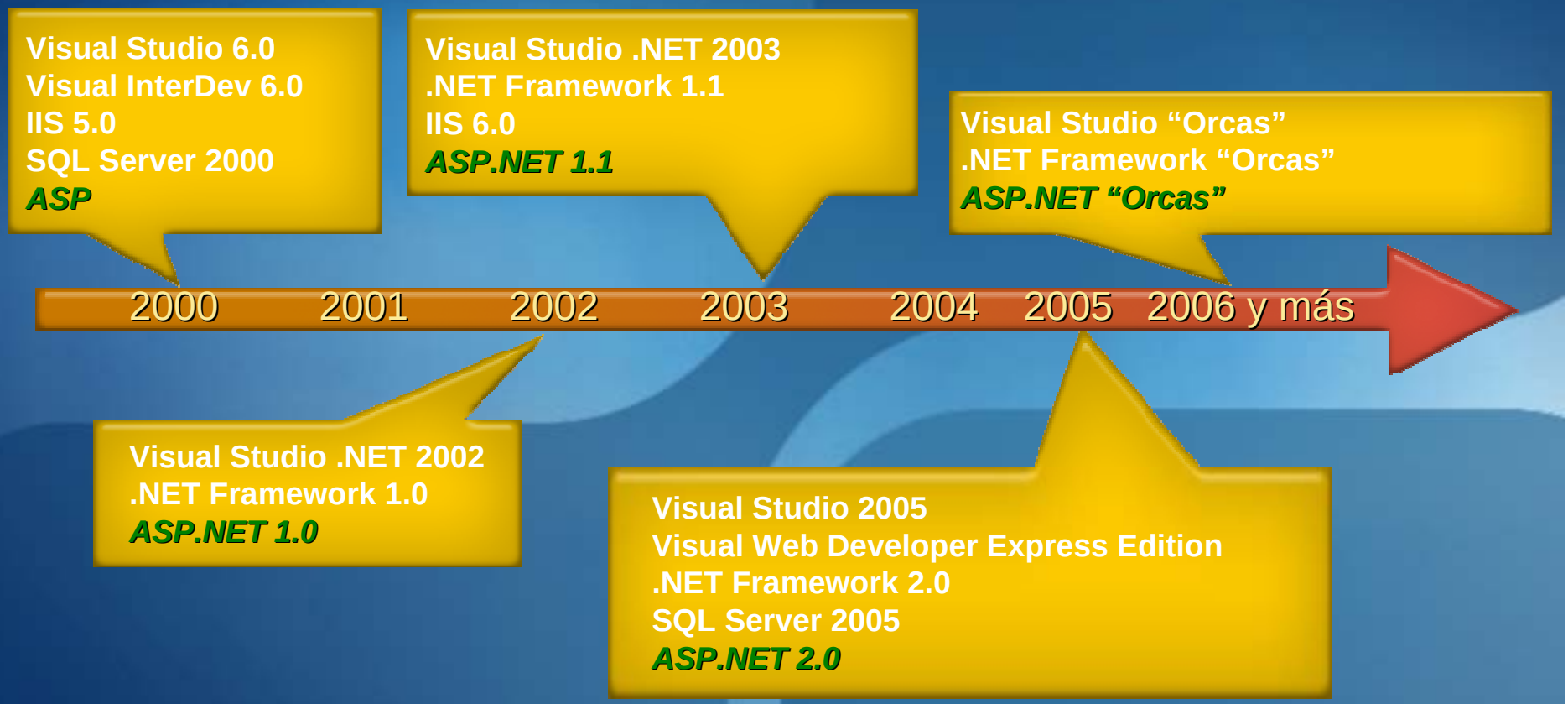
Las aplicaciones Web ASP.NET + IIS

- VS.NET por default crea las aplicaciones web bajo el directorio raíz, ej.:
  - MiAplicacion
    - Virtual:
      http://localhost/MiAplicacion
    - Física:
      C:\Inetpub\wwwroot\MiAplicacion
- Podemos usar IIS para definir un directorio virtual donde alojar nuestras aplicaciones Web, diferente al predeterminado

Estructura de las aplicaciones Web

- El “perímetro” de una aplicación Web es determinada por su estructura de directorios
- Comienza por su directorio raíz, el cual contiene:
  - La página o WebForm de inicio
  - El archivo de configuración Web.config
  - El directorio BIN
- El perímetro de la aplicación termina en su último directorio o cuando se encuentra el directorio raíz de otra aplicación Web

Plataforma de desarrollo Web Microsoft en el tiempo



Temario (1/2)

- Introducción a ASP.NET
- **Formularios Web (Web Forms)**
  - Generalidades
  - Controles Web
  - Eventos de un formulario web
  - Ciclo de Vida de un formulario web
- Configuración
- Autenticación
- Como mantener el estado en una aplicación web

WebForms - Generalidades

- Formulario Web *(ASP.NET web form)*
  - Es una página expresada en lenguaje de marcas que es compilada y ejecutada dinámicamente en el servidor para generar la salida solicitada por el cliente (explorador ó dispositivo).
- Code Behind
  - Es el código que se ejecuta del lado del servidor para lograr el comportamiento deseado en un formulario web.
- Partial Class
  - Un nuevo concepto, que es aplicado en ASP.NET para vincular las páginas aspx (la interfaz del usuario) con su Code Behind (comportamiento).

Controles de Servidor

- Componentes que se ejecutan en el lado del servidor
- Encapsulan partes de la interface de usuarios
- Poseen el atributo runat="server"
- Mantienen su "estado" entre postbacks al servidor – ViewState
- Poseen un modelo de objetos común
  - Ej.: todos tienen las propiedades Id y Text
- Generan HTML especifico según el browser cliente

Tipos de Controles de Servidor

Controles de Servidor HTML

- Por default, los elementos HTML no son accesibles desde código del lado del servidor
- Agregando runat="server" y el atributo id, se convierten en Controles de Servidor HTML

Controles de Servidor Web

- Conocidos como WebControls
- Solo accesibles del lado del servidor
- Muchos tipos
  - Intrínsecos
  - Validación
  - "Ricos"
  - Del tipo lista de datos
- No tienen una relación 1:1 con elem. HTML

Equivalencias de Controles

- Botón HTML “clásico” (No es de Servidor)

```
<INPUT type="button" value="Buscar">
```

- Control de Servidor HTML

```
<INPUT type="button" value="Buscar"
id="cmdBuscar" runat="server" NAME="button1">
```

- Control de Servidor Web

```
<asp:Button id="cmdBuscar" runat="server"
Text="Buscar"/>
```

Controles de Servidor - Ejemplos

WebControl	HTML equivalente
<asp:button>	<input type=submit>
<asp:checkbox>	<input type=checkbox>
<asp:hyperlink>	
<asp:image>	<img src="…">
<asp:imagebutton>	<input type=image>
<asp linkButton>	
<asp:label>	
<asp:panel>	
<asp:radiobutton>	<input type=radiobutton>
<asp:table>	<table> </table>
<asp:textbox>	<input type=text>
<asp:listbox>	<select size="5"> </select>

Controles de Servidor - Validación

- Son elementos ocultos que validan las entradas de datos contra algún patrón
- El proceso de validación puede ser llevado en:
  - Cliente
    - El browser cliente debe soportar lenguaje script
    - Le da al usuario un feedback inmediato
    - Reduce el número de postbacks
  - Servidor
    - Repite la validación del lado del cliente
    - Permite validar contra datos almacenados por ej. en una base de datos

Controles de Servidor - Validación

- ASP.NET proporciona 6 controles
  - RequiredFieldValidator. Valor requerido.
  - CompareValidator. Valida contra un valor constante o contra otro control.
  - RangeValidator. Valor dentro de un rango de tipos.
  - RegularExpressionValidator. Valida contra un patrón o expresión regular.
  - CustomValidator. Lógica de validación proporcionada por nosotros.
  - ValidationSummary. No es un validador, sino que muestra mensajes de error “agrupados”.

Controles de Servidor - Validación

- En el lado del servidor se puede determinar si TODAS las validaciones fueron exitosas mediante Page.IsValid
- Muy importante!
  - La propiedad IsValid NO está disponible en los eventos Init ni Load del WebForm

Controles de Servidor – Controles "Ricos"

- Controles con lógica de IU compleja encapsulados de forma sencilla
- Ejemplos:
  - AdRotator. Permite mostrar anuncios publicitarios (banners) de una secuencia predeterminada o aleatoria.
  - Calendar. Permite disponer de un calendario altamente personalizable.

Controles de Usuario

- Simplifican la reutilización de código y componentes de la IU dentro de las aplicaciones Web ASP.NET
- Son del servidor definidos en un archivo .ascx
- Contienen HTML pero NO los tags <HTML>, <BODY> o <FORM>

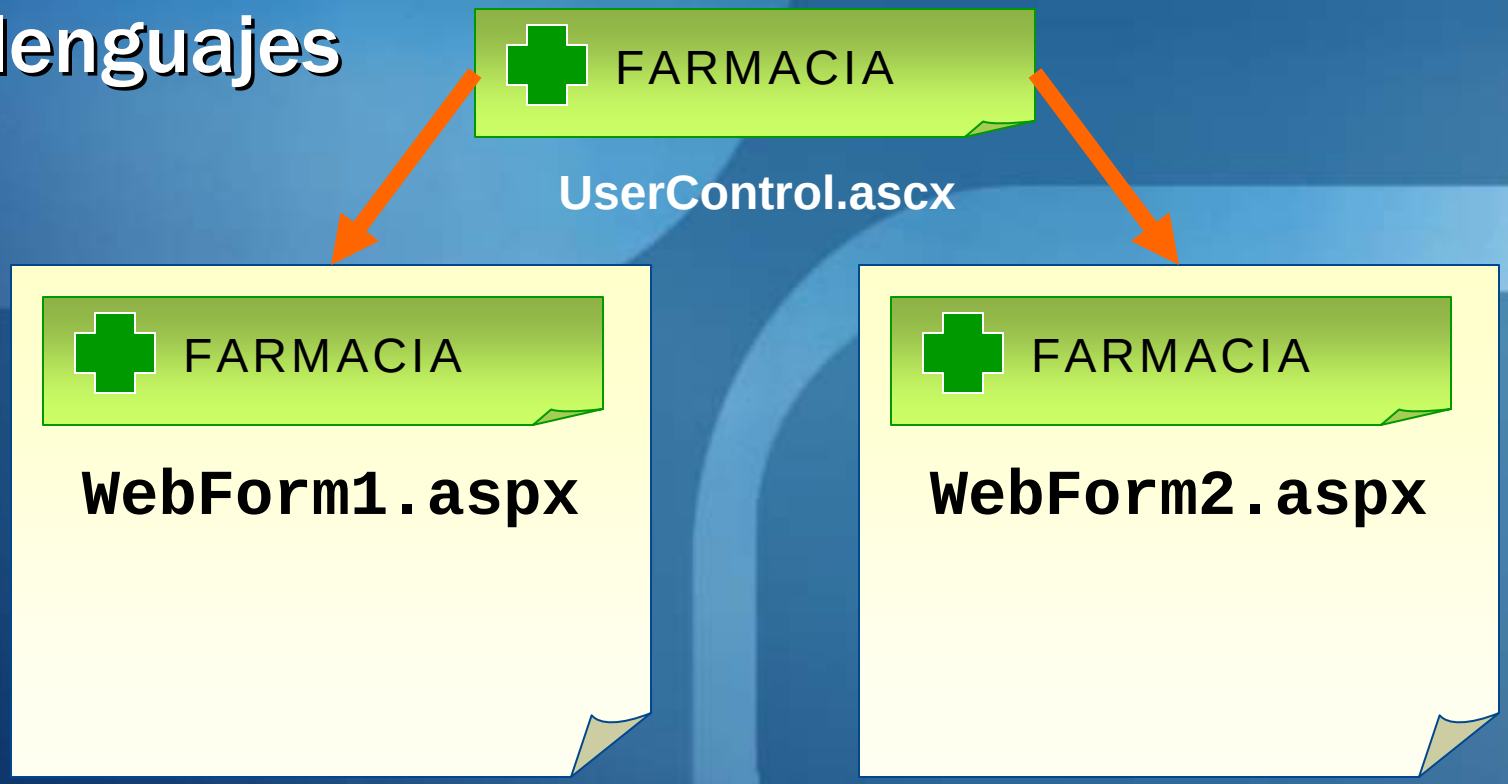
```
<%@ Control Language="vb" %>
```

```
<%@ Control Language="cs" %>
```

- Contiene también código en VB.NET o C#

¿Por qué usar Controles de Usuario?

- Son autocontenidos
- Pueden ser utilizados más de una vez
- Pueden estar escritos en diferentes lenguajes



Agregando Controles de Usuario

- Para usar un control de usuario en un WebForm se usa la directiva @Register

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

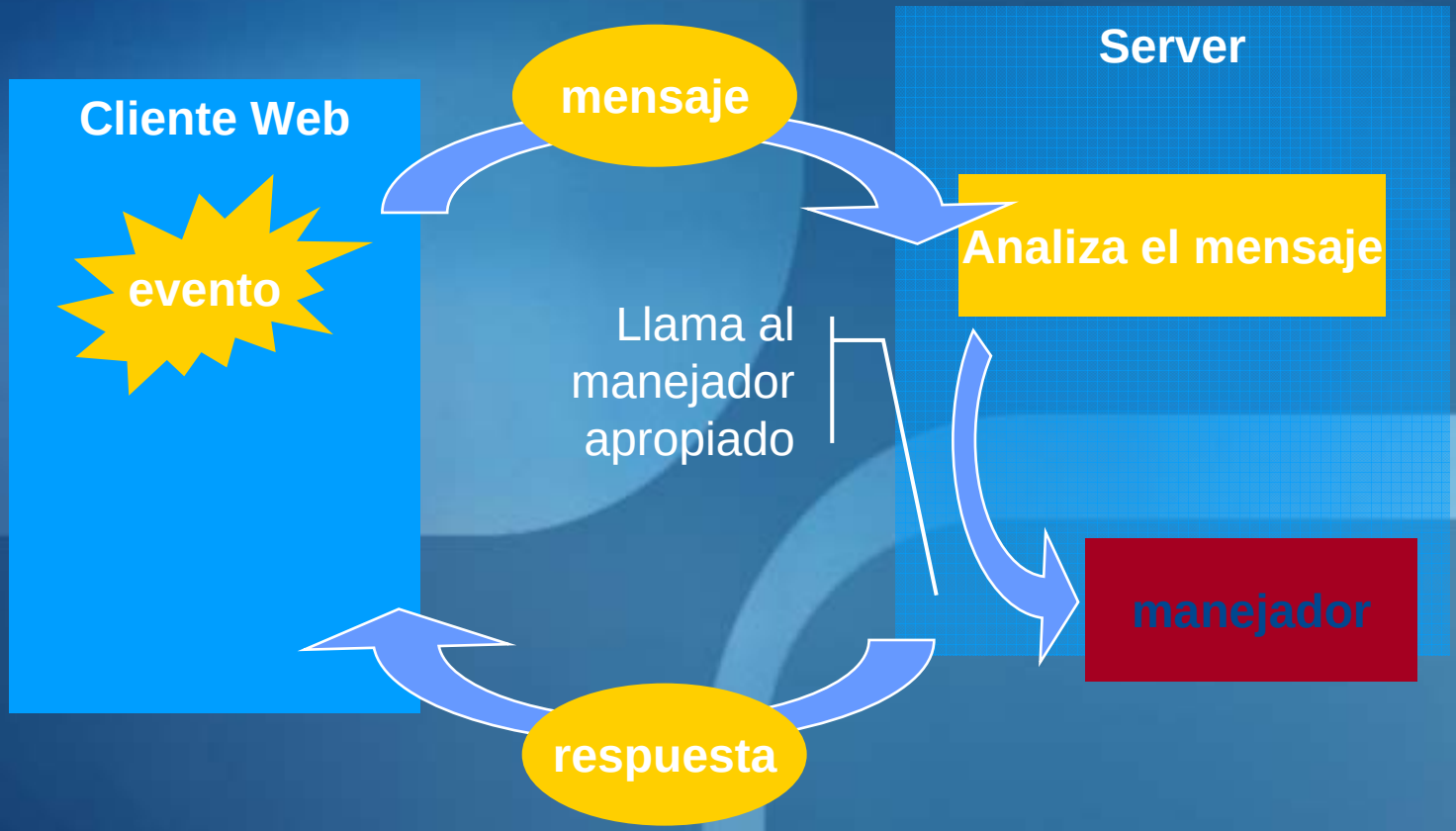
- Dentro del WebForm...

```
<uc:encabezado id="Enc1" runat="server">
    </uc:encabezado>
```

- Podemos acceder y crear propiedades como con cualquier otro control u objeto

Eventos en un WebForm (1/4)

Modelo de Eventos Web Form



Eventos en un WebForm (2/4)

- Eventos del lado del servidor

  Cuando se trabaja con controles ASP.NET, estos generan eventos en el servidor para responder a las peticiones del usuario, produciéndose **PostBack**.

- PostBack

  A instancias de un formulario web mostrado en el cliente cada evento sucedido en él genera un POST hacia el servidor y una respuesta. Este ida y vuelta dentro de un mismo formulario web se llama postback.

- View State

  Es un mecanismo que permite mantener el estado de los controles del formulario web entre postbacks. El estado de los controles viaja en el view state por cada postback.(ver diapositiva 36)

Eventos en un WebForm (3/4)

Declaración de eventos en un control del lado del cliente:

```
<asp:Button ID="btnEjemplo" runat="server" Text="Aceptar"
      onclick="btnEjemploClick" />
```

Atención del evento en el servidor (code behind)

Ejemplo en C#:

```
protected void btnEjemploClick(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(txtEjemplo.Text))
    {
        lblEjemplo.Text = txtEjemplo.Text;
    }
}
```

Eventos en un WebForm (4/4)

Atención del evento en el servidor (code behind) .

Ejemplo en Visual Basic:

```
Protected Sub btnEjemploClick(ByVal sender As Object, _
    ByVal e As EventArgs)
    If Not String.IsNullOrEmpty(txtEjemplo.Text) Then
       lblEjemplo.Text = txtEjemplo.Text
    End If
End Sub
```

Ciclo de vida de un WebForm

Sucesos mas importantes del ciclo de vida de una página:

- Inicialización de objetos
- Carga View State
- Procesa datos del post
- Carga de la página
- Eventos Post Back
- Graba View State
- Render
- UnLoad

Temario (1/2)

- Introducción a ASP.NET
- Formularios Web (Web Forms)
- Configuración
  - Conceptos principales
  - Accediendo desde el código al web.config
- Autenticación
- Como mantener el estado en una aplicación web

Conceptos principales (1/2)

- Archivo Web.Config
  - Es un archivo xml, donde se guarda información de configuración común a toda la aplicación como ser: cadenas de conexión, tipo de autenticación, etc.
- Arquitectura de configuración jerárquica
  - El archivo de configuración posee una estructura jerárquica (xml) que permite una lectura rápida y facilita su modificación.

Conceptos principales (2/2)

- Secciones y grupos de secciones
  - En ASP.NET 2.0 se incorporan nuevos grupos de secciones, como por ejemplo “connectionStrings”. Facilitando de este modo el acceso a las conexiones de datos y simplificando el código.
- Herramientas administrativas
  - Snap-in de MMC para ASP.NET
  - Herramienta de administración del sitio web (Web Site Administration Tool)

Accediendo desde el código al web.config (1/2)

- Secciones configuración más simples

```
<configuration>
 <connectionStrings>
   <add name="northwind"
      connectionString="server=(local);database=Northwind;Integrated
      Security=SSPI" providerName="System.Data.SqlClient" />
 </connectionStrings>
</configuration>
```

Accediendo desde el código al web.config (2/2)

- Acceso de lectura/escritura a especificaciones de configuración.

  Ejemplo en C#:

```
SqlConnection connection = new SqlConnection(
    ConfigurationManager.ConnectionStrings[
    "ADVENTUREWORKSConnectionString"].ConnectionString);
```

  Ejemplo en Visual Basic

```
Dim connection As SqlConnection = New _
    SqlConnection(ConfigurationManager.ConnectionStrings( _
    "ADVENTUREWORKSConnectionString").ConnectionString)
```

Temario (1/2)

- Introducción a ASP.NET
- Formularios Web (Web Forms)
- Configuración
- **Autenticación**
  - Generalidades
  - Tipos de Autenticación
  - Autenticación por Formularios
  - Controles de Login
- Como mantener el estado en una aplicación web

Generalidades

- ¿Qué es Autenticación?
  - Es el mecanismo que permite afirmar que la persona que esta ingresando al sistema es quien dice ser.
- ¿Cómo Funciona?
  - Se aceptan las credenciales ingresadas por el usuario (usuario – contraseña) y se validan contra una base de datos, el sistema operativo, un servicio web, u otro mecanismo definido según el tipo de autenticación.

Tipos de Autenticación

- Basada en Windows
  - Basada en Windows e IIS
  - La solicitud de la página pasa por IIS
  - Si IIS valida exitosamente la credencial, entonces se devuelve la página solicitada
- Basada en Formularios
  - Las solicitudes no autenticadas son redireccionadas a un formulario de login
  - Después de validar la credencial se envía al cliente una cookie de autenticación
- Basada en Microsoft Passport
  - Servicio de autenticación centralizado
  - Passport es un Web Service

Configurando la autenticación

```
<system.web>
    <authentication mode="Forms">
        <forms loginUrl="login.aspx"></forms>
    </authentication>
    <authorization>
        <deny users="?"/>
    </authorization>
</system.web>
```

Autenticación por Formularios

- La autenticación por formularios se utiliza para validar a los usuarios contra bases de datos relaciones, Servicios Web, etc.
- ¿Cómo funciona?
  - Si las credenciales son válidas, ASP.NET graba un ticket de autenticación en la cookie que contiene la identidad del usuario.
  - Si el usuario es anónimo, redirecciona las peticiones a una página predeterminada para validar las credenciales del usuario.

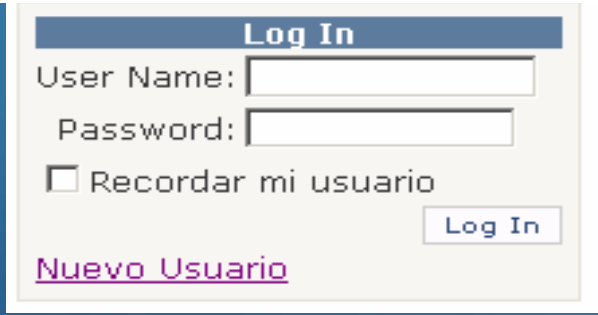
Autenticación por Formularios

Controles de Login (1/2)

- Control Log In

```
<asp:Login ID="Login1" runat="server“
    CreateUserUrl="~/CrearUsuario.aspx"
                CreateUserText="Nuevo Usuario“
                RememberMeText="Recordar mi usuario" BackColor="#F7F6F3"
                BorderColor="#E6E2D8" BorderPadding="4" BorderStyle="Solid" BorderWidth="1px"
     Font-      Names="Verdana" Font-Size="0.8em" ForeColor="#333333" >
    <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True" Font-Size="0.9em" ForeColor="White" />
    <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
    <TextBoxStyle Font-Size="0.8em" />
    <LoginButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC" BorderStyle="Solid"
                BorderWidth="1px"  Font-Names="Verdana" Font-Size="0.8em" ForeColor="#284775"
     />
</asp:Login>
```

Ejemplo del control



Controles de Login (2/2)

Control de Creación de Usuarios

```
<asp:CreateUserWizard ID="createUserWizard" runat="server"
    CreateUserButtonText="Crear Usuario"
        CompleteSuccessText="Usuario Creado"
ContinueButtonText="Finalizar"
        ContinueDestinationPageUrl="~/Default.aspx"
    FinishDestinationPageUrl="~/Default.aspx">
      <WizardSteps>
        <asp:CreateUserWizardStep ID="createUserWizardStep"
runat="server">
        </asp:CreateUserWizardStep>
        <asp:CompleteWizardStep ID="comple
          Title="Usuario Creado"
           AllowReturn="true">
        </asp:CompleteWizardStep>
      </WizardSteps>
</asp:CreateUserWizard>
```

Ejemplo del control

Temario (1/2)

- Introducción a ASP.NET
- Formularios Web (Web Forms)
- Configuración
- Autenticación
- Como mantener el estado en una aplicación web
  - Generalidades
  - Global.asax
  - Estado de la aplicación
  - Estado de la sesión
  - View State

Generalidades

- Las páginas html de una aplicación web se transmiten por medio del protocolo HTTP, como se mencionó anteriormente. Este protocolo es un protocolo “sin estado”.

  Así, una vez que el usuario ingreso datos en el navegador, **si no se mantiene el estado** mediante algún mecanismo, **se pierden los datos ingresados**.

  Por este motivo, ASP.Net proporciona mecanismos para mantener el estado de sus variables a través de las distintas peticiones de páginas.

- Entre estos mecanismos se encuentran:
  - **Application State** : mecanismo de almacenamiento global accesible desde todas las páginas de la aplicación Web
  - **Session State** : mecanismo de almacenamiento limitado a la sesión actual del navegador
  - **View State** :Mantiene valores entre múltiples solicitudes a la misma página

Administración de estados

Sin Adm. De Estados

Login.aspx

Ingrese sus datos de inicio de sesión
Nombre
Tania
Contraseña
****
Ingresar

Inicio.aspx

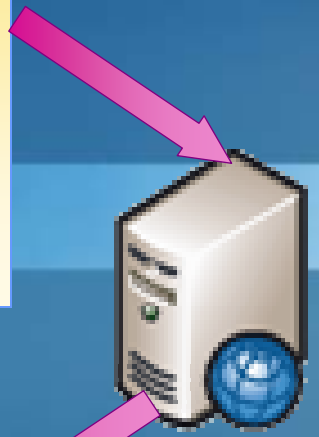
Hola

Olvide quien es Ud!

Con Adm. De Estados

Login.aspx

Ingrese sus datos de inicio de sesión
Nombre
Tania
Contraseña
****
Ingresar



Inicio.aspx

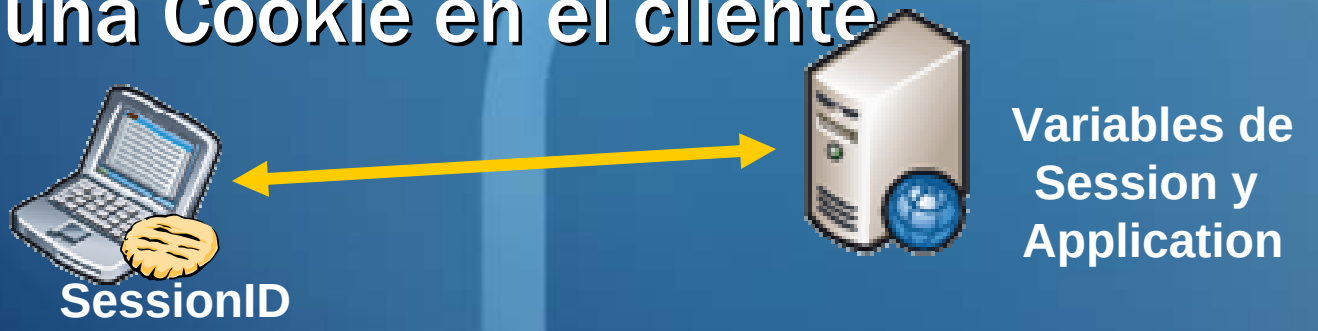
Hola Tania

Tipos de administración de estado

Del lado del servidor	Del lado del cliente
Application state Información disponible para todos los usuarios de la aplicación Web.	**Cookies** Archivos de texto que guardan información de estado en la PC cliente
Session state Información disponible únicamente para un usuario de una sesión específica	**ViewState** Mantiene valores entre múltiples solicitudes a la misma página
Database En algunos casos se utiliza una Base de Datos para guardar la información de estado	**Query strings** Información anexada al final de la URL

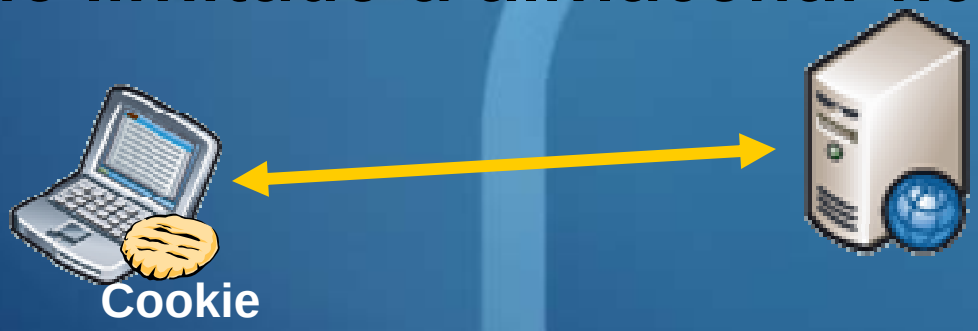
Administración de estados - Servidor

- Application State es un mecanismo de almacenamiento global accesible desde todas las páginas de la aplicación Web
- Seesion State está limitada a la sesión actual del browser
- Las sesiones ASP.NET se identifican con una cadena de caracteres ASCII y se guarda como una Cookie en el cliente



Administración de estados - Cliente

- Cookies para mantener el estado:
  - Temporarias
  - Persistentes
- Problema: el usuario puede borrarlas o deshabilitarlas
- Problemas de seguridad
- Espacio limitado a almacenar no más de 4KB



Global.asax

- Administra eventos a nivel de aplicación y sesión.
- Los eventos referentes a la manutención del estado de aplicaciones web, son:
  - Application_Start
  - Application_End
  - Session_Start
  - Session_End

Estado de la aplicación(1/2)

- Permite almacenar información a nivel de aplicación, común a todas las sesiones. Esta información se almacena en una colección llamada **Application**.
- El acceso a esta información se realiza mediante el objeto intrínseco “Application”

Ejemplo en C#

```
protected void Application_Start() {
     DataSet ds = new DataSet();
     try {
        FileStream fs = new
                FileStream(Server.MapPath("schemadata.xml"),
                FileMode.Open,FileAccess.Read);
        StreamReader reader = new StreamReader(fs);
        ds.ReadXml(reader);
     }
     finally { fs.Close(); }
     DataView view = new DataView(ds.Tables[0]);
     Application["Sucursal"] = view;
 }
```

Estado de la aplicación(2/2)

Ejemplo en Visual Basic

```
Sub Application_Start(ByVal sender As Object, ByVal e As _ EventArgs)
        Dim ds As DataSet = New DataSet
        Try
            Dim fs As FileStream = New _
                FileStream(Server.MapPath("schemadata.xml"), _
                FileMode.Open, FileAccess.Read)
            Dim reader As StreamReader = New StreamReader(fs)
            ds.ReadXml(reader)
        Finally
                fs.Close
        End Try
        Dim view As DataView = New DataView(ds.Tables(0))
        Application("Sucursal") = view
 End Sub
```

Diapositiva 70

CAW3 finally

Carlos Walzer, 12/2/2005

Estado de la sesión (1/2)

- Una sesión es una interacción entre un navegador y un servidor web (comprende varios Requests a lo largo del tiempo)
- Es posible almacenar información únicamente relevante para una sesión.
- El acceso a esta información se realiza mediante el objeto intrínseco “Session”

Ejemplo C#

```
Protected void Session_Start( object sender, EventArgs e ) {
     //...
     try
    {
        conn = new SqlConnection(
        ConfigurationManager.ConnectionStrings[
        "ADVENTUREWORKSConnectionString"].ConnectionString);
        command.Connection = conn;
        conn.Open();
        object result = command.ExecuteScalar();
        Session["codigo"] = result;
     }
     finally {
       conn.Close();
    }
 }
```

Estado de la sesión (2/2)

Ejemplo en Visual Basic

```
Sub Application_Start(ByVal sender As Object, ByVal e As _
    EventArgs)
   Dim ds As DataSet = New DataSet
    Dim command As SqlCommand = New SqlCommand( _
              "Select codigo From clientes Where name = " & nombre )
    Try
       conn = New SqlConnection( _
       ConfigurationManager.ConnectionStrings( _
       "ADVENTUREWORKSConnectionString").ConnectionString)
        command.Connection = conn
        conn.Open()
        Dim result As Object = command.ExecuteScalar()
        Session("codigo") = result
    Finally
        conn.Close()
    End Try
End Sub
```

View State

- Mantiene el estado de los controles, entre postback de una página.
- El View State se implementa mediante un campo oculto en el html generado y viaja en cada POST

Temario (2/2)

- Master Pages
  - Generalidades
  - Páginas Maestras
  - Páginas de Contenido
- Themes y Skins
- Navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear un sitio Web
- Referencias

Generalidades

- Logran herencia visual para las páginas Web
- Permite manejar áreas comunes de un sitio de manera consistente
- En ASP.NET 1.1, el problema de la herencia visual implicaba
  - Copiar & Pegar
  - Includes
  - Herencia + User Controls
- Se basan en Templates (Master Page) y en páginas de contenido (Content Page).
- Permiten incluir menús, encabezados, navegaciones, etc.

Páginas Maestras

- En lugar de la directiva @Page, utiliza la directiva @Master:

```
<%@ Master Language="C#“
    CodeFile="MasterPage.master.cs"
    Inherits="MasterPage" %>
```

- Se trata como cualquier formulario web, con la extensión .master, y debe incluir el siguiente control:

```
<asp:contentplaceholder id="contenedor" runat="server“>
    contenido por defecto
</asp:contentplaceholder>
```

Páginas de contenido

- Al crear un página de contenido, hay que elegir la página maestra. La directiva @page de la página sería:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
    CodeFile="Default.aspx.cs" Inherits="_Default" Title="Home" %>
```

- No repiten los tags de la página maestra, pero se pueden acceder desde el código.
- Debe contener el control

```
<asp:Content ID="cntDefault" ContentPlaceHolderID="contenedor"
Runat="Server"> contenido de la página </asp:Content>
```

Temario

- Master Pages
- Themes y Skins
  - Generalidades
  - Ejemplo del contenido de un “Skin”
- Navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear un sitio Web
- Referencias

Themes y Skins - Generalidades

- **Skins:** Son definiciones de formato y estilos que se aplican a los controles de servidor y se guardan en archivos de extensión .skin
- Themes: Son “paquetes” de Skins, también pueden contener hojas de estilo en cascada e imágenes asociados.
- ASP.NET 2.0 incluye una nueva carpeta virtual para la organización de los temas (APP_Themes).
- Se pueden configurar a nivel de maquina (en el machine config) a nivel de aplicación (en el web.config) o bien a nivel de página, dentro de la directiva Page.

Contenido de un archivo Skin

- Los archivos *.skin contienen declaraciones de estilo y formato de los controles de ASP.NET

```
<asp:LoginName  runat="server"
                BorderWidth="1"
                BorderColor="#FF9900"
                ForeColor="navy"
                Font-Names="verdana" />
```

Temario

- Master Pages
- Themes y Skins
- Navegación
  - Menú
  - Control de navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear un sitio Web
- Referencias

Navegación - Menú

```
<asp:menu id="Menu"
    datasourceid="SiteMapDataSource1“ disappearafter="500"
  staticdisplaylevels="2" staticsubmenuindent="20" orientation="Vertical"
    font-names="Trebuchet MS, Arial" DynamicMenuItemStyle-Width="150"
    Width="150“ runat="server">
    <staticmenuitemstyle backcolor="RoyalBlue" forecolor="WhiteSmoke" horizontalpadding="5"
    verticalpadding="2" />
    <statichoverstyle backcolor="CornflowerBlue" forecolor="White" borderstyle="Solid" borderwidth="1px" />
    <dynamicmenuitemstyle backcolor="RoyalBlue" forecolor="WhiteSmoke" horizontalpadding="5"
    verticalpadding="2" />
    <dynamichoverstyle backcolor="CornflowerBlue" forecolor="White" borderstyle="Solid" borderwidth="1px"
    />
</asp:menu>
```

Ejemplo del control

Control de Navegación

- Este control esta basado en el modelo de proveedores. (Se configura el proveedor en el web.config)
- A diferencia de otros controles de navegación no posee una propiedad “DataSource”.
- Al proveedor se le configura el archivo (xml) que posee el mapa del sitio, por ejemplo:
  - siteMapFile="web.sitemap"

Temario

- Master Pages
- Themes y Skins
- Navegación
- **Acceso a Datos**
  - Controles de enlace a datos
  - Controles visualizadores de datos
- Compilación e Instalación
- Como crear un sitio Web con Visual Studio 2005
- Referencias

Controles de enlace de Datos

- Permiten realizar el enlace entre un control que muestra datos (gridview, detailsview, etc) y la lógica que los administra

  - Control ObjectDataSource

    Enlaza los controles con una clase de la capa de negocios.

  - Control SqlDataSource

    Enlaza los controles con una base de datos relacional.

  - Control XmlDataSource

    Enlaza los controles con datos en formato xml.

Controles visualizadores (1/2)

DataGridView

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
      DataKeyNames="ProductID" AllowPaging="True" AllowSorting="True">
      <Columns>
        <asp:CommandField DeleteText="Excluir“ CancelText="Cancelar“
            UpdateText="Atualizar" EditText="Editar"/>
        <asp:BoundField ReadOnly="True" HeaderText="Cod" DataField="ProductID"/>
        <asp:BoundField HeaderText="Produto" DataField="ProductName"/>
        <asp:BoundField HeaderText="Estoque" DataField="UnitsInStock"
                        DataFormatString="{0:n0}"/>
</Columns>
  </asp:GridView>
  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:myConnection %>"
    UpdateCommand="UPDATE [Products] SET [ProductName] = @ProductName "
    SelectCommand="SELECT [ProductID], [ProductName] ... FROM [Products]"
    InsertCommand="INSERT INTO [Products] ([ProductName] ..."
    DeleteCommand="DELETE FROM [Products] WHERE [ProductID] ...">
  </asp:SqlDataSource>
```

Controles visualizadores (2/2)

DetailsView

```
<asp:DetailsView AutoGenerateRows="False" DataKeyNames="au_id"
    DataSourceID="SqlDataSource3" HeaderText="Author Details" ID="DetailsView1"
    runat="server" Width="275px">
     <Fields>
       <asp:BoundField DataField="au_id" HeaderText="au_id" SortExpression="au_id" />
    <asp:BoundField DataField="au_lname" HeaderText="au_lname" />
       <asp:CheckBoxField DataField="contract" HeaderText="contract" />
    </Fields>
    </asp:DetailsView>
```

FormView

```
<asp:FormView ID="FormView1" runat="server" DataSourceID="ObjectDataSource1“>
    <ItemTemplate>
     <asp:Label ID="CaptionLabel" runat="server" Text='<%# Eval("Caption") %>' /><br />
     <asp:Image ID="Image1" runat="server" ImageUrl='<%# Eval("FileName", "images/{0}") %>'
         /> <br />
          <asp:HyperLink ID="HyperLink1" Text=“Volver" NavigateUrl='<%# Eval( "AlbumID",
          "PhotosDataList.aspx?ID={0}") %>' runat="server" />
    </ItemTemplate> </asp:FormView>
```

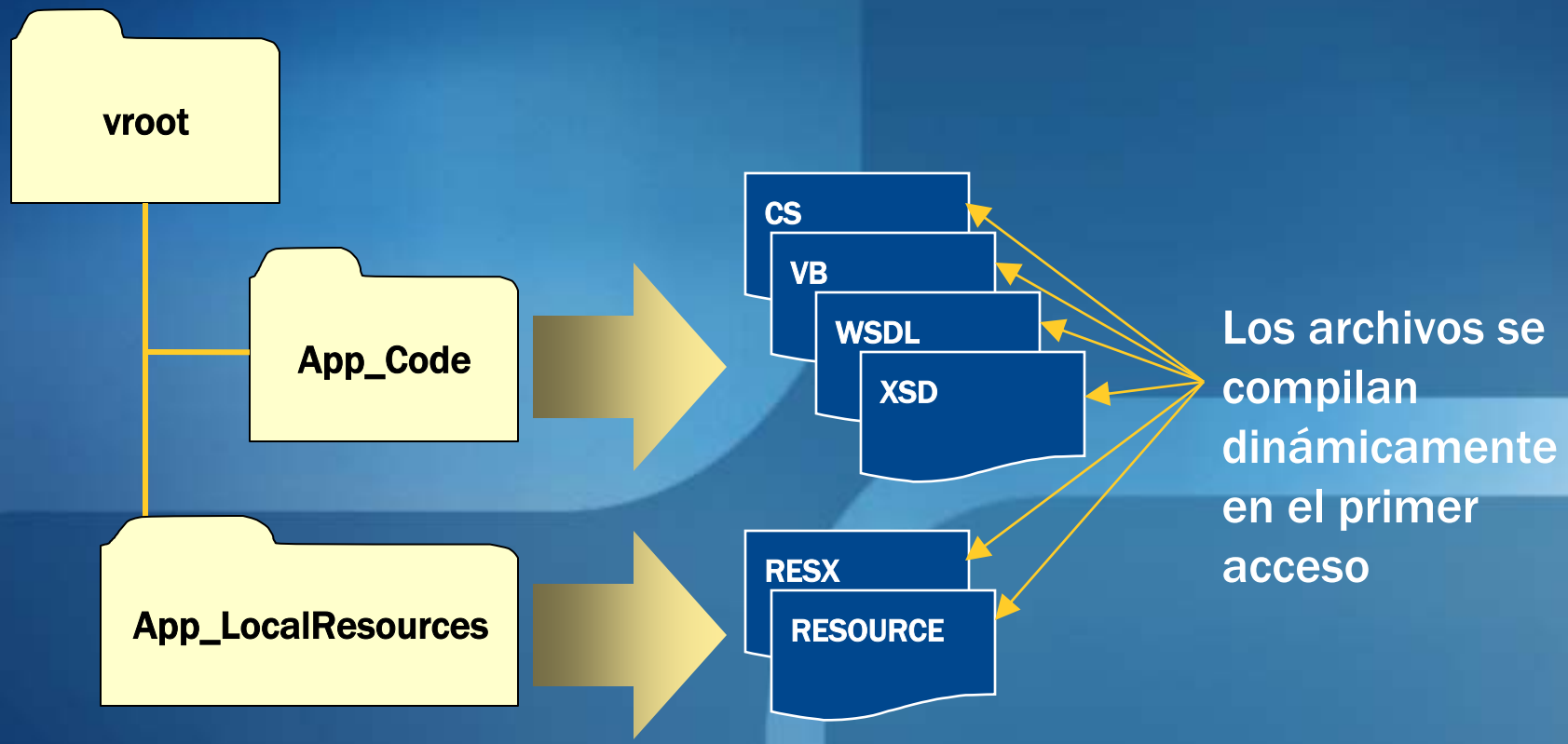
Temario

- Master Pages
- Themes y Skins
- Navegación
- Acceso a Datos
- Compilación e Instalación
  - Generalidades
  - Compilación Dinámica
- Como crear un sitio Web con Visual Studio 2005
- Referencias

Compilación e Instalación - Generalidades

- Compilación dinámica de:
  - Aspx, asmx, ascx
  - Vb, cs, resx
  - Sólo es necesario poner los archivos en los directorios específicos
- Precompilación e implementación sin código fuente
  - *Aspnet_compiler.exe* precompila sitios y los instala sin el código fuente

Compilación dinámica



Temario

- Master Pages
- Themes y Skins
- Navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear un sitio web con Visual Studio 2005
  - Generalidades
  - Creación y Acceso a Proyectos
- Referencias

Generalidades

- Independencia de IIS
  - Visual Studio 2005 invluye el **ASP.NET Development Server**, un servidor de HTTP local que permite trabajar en una PC de desarrollo sin tener instalado IIS (Microsoft Internet Information Server)
- Front Page Server Extensions
  - Las extensiones de servidor de Front Page no son necesarias, ahora se puede seleccionar el directorio donde se van a alojar las páginas del sitio y comenzar a desarrollar.

Creación y Acceso a Proyectos

- **File System**
  - Permite desarrollar un sitio en cualquier carpeta de la PC.
- **IIS Local**
  - Permite desarrollar localmente una aplicación web en un directorio virtual de IIS.
- **Sitio FTP**
  - Permite editar y modificar proyectos web remotos utilizando el protocolo FTP.
- **Sitio Remoto**
  - Se puede sincronizar el proyecto de desarrollo local con la aplicación instalada en producción.

Temario

- Master Pages
- Themes y Skins
- Navegación
- Acceso a Datos
- Compilación e Instalación
- Como crear un sitio Web con Visual Studio 2005
- Referencias

Referencias (1/2)

- Tutorial de ASP.NET 2.0

  http://www.ASP.NET/Tutorials/quickstart.aspx
- Estado de la Aplicación
  http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art175.asp
- Call Back
  http://msdn.microsoft.com/msdnmag/issues/05/01/CuttingEdge/default.aspx
- Sitio Oficial de ASP.NET

  http://www.ASP.NET

Referencias (2/2)

- Ciclo de Vida

  http://msdn2.microsoft.com/en-us/library/ms178472.aspx
- Web Parts
  http://msdn.microsoft.com/msdnmag/issues/05/09/WebParts/default.aspx
- Modelo de Proveedores

  http://msdn.microsoft.com/ASP.NET/default.aspx?pull=/library/en-us/dnaspp/html/ASPNETProvMod_Intro.asp
- *Libro:* Introducing Microsoft ASP.NET 2.0

  Autor: Dino Esposito

  http://www.microsoft.com/mspress/books/6962.asp

Microsoft®