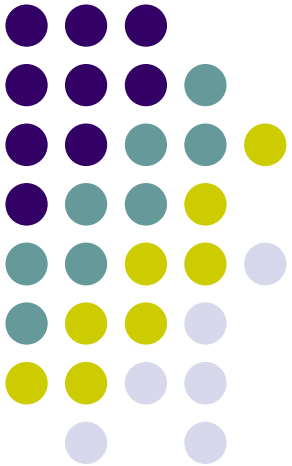
Desarrollo de Aplicaciones Web

Desarrollo Web en Entornos de Servidor



Tema 3

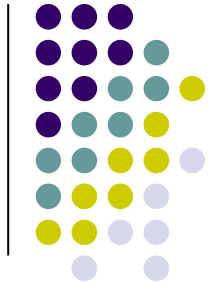
Programación basada en código de servidor

Vicente J. Aracil Miralles

vj.aracilmiralles@edu.gva.es

27/09/2022

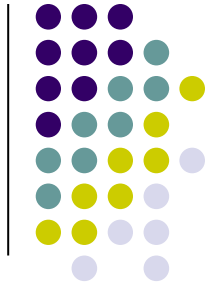
Tema 3
Programación basada en código de servidor embebido



Objetivos

- Conocer y utilizar los mecanismos disponibles para la creación de bloques de código procedural de ejecución en el lado del servidor Web
- Conocer la sintaxis de los lenguajes de programación en entorno servidor que sean capaces de representar la lógica del negocio de las aplicaciones Web
- Aprender a utilizar y definir funciones
- Comprender los modelos de objetos y la utilización de procedimientos de evento
- Comprender los diferentes métodos de recuperación de información enviada por un cliente Web
- Aprender a procesar en el servidor Web la información enviada por un cliente a través de formularios

Tema 3
Programación basada en código de servidor embebido



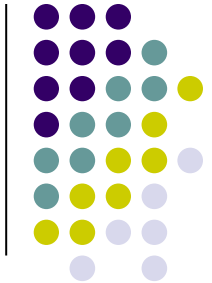
Contenidos

3.1 Agregar código lógico a las páginas de ASP.NET

3.2 Introducción a la programación de páginas de ASP.NET

3.3 Contexto de aplicación y de sesión en aplicaciones Web

3.4 Lenguaje de programación C#



Desarrollo de páginas Web de ASP.NET

- El desarrollo de páginas Web de ASP.NET se divide en dos partes:
  - **El componente visual.** Está compuesto por código de marcado que define los elementos y controles Web que conforman la interfaz de usuario
    - Una página de ASP.NET funciona como un contenedor de los elementos y controles Web que conforman la presentación de la página (capa de presentación)
  - **El componente lógico.** Se compone de código lógico o procedural ejecutado por el servidor y creado para interactuar con la página Web
    - Expresa el procesamiento lógico ejecutado en el servidor para obtener y presentar la información solicitada de forma adecuada (capa de lógica del negocio)
    - Las páginas Web de ASP.NET soportan bloques de código lógico que pueden estar escrito en diversos lenguajes de programación, como Visual C#, Visual Basic, u otros compatibles con .NET Framework

El código lógico incluido en el conjunto de páginas de ASP.NET que conforman una aplicación Web representa la capa de lógica del negocio de la aplicación Web

Modelo de código de ASP.NET

- Una página Web de ASP.NET se compone de:
  - **Elementos de presentación visual**, incluidos el formato de la presentación visual, los controles de servidor Web y el texto estático
  - **Lógica de procesamiento en el servidor**, que incluye la programación de los controladores de eventos para la página y otro tipo de código procedural
    - El modelo de código de ASP.NET proporciona dos formas para administrar el código lógico de una página de ASP.NET:

Se recomienda la utilización del modelo de código subyacente (archivos distintos)

Modelo de código en un archivo

Ejemplo3-1.aspx

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

<!DOCTYPE html>

<script runat="server">
    protected void btnEnviar_Click(object sender, EventArgs e)
    {
        if (txtNombre.Text != "")
        {
            lblTexto.Text = "Bienvenido a ASP.NET " + txtNombre.Text;
        }
        else
        {
            lblTexto.Text = "Debe introducir su nombre en el cuadro de texto";
        }
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

Modelo de código en un archivo

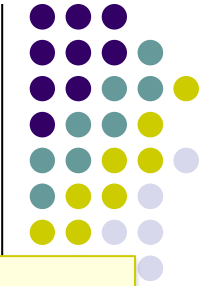
Ejemplo3-1.aspx (continuación)

```
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
       <div>
          <h1>Ejemplo 3-1</h1> <hr />
          <br />
          <asp:Label ID="Label1" runat="server" Text="Label">
                Introduzca su nombre: </asp:Label>
          <asp:TextBox ID="txtNombre" runat="server"></asp:TextBox> <br /> <br />
          <asp:Button ID="btnEnviar" runat="server" Text="Enviar“
                      OnClick="btnEnviar_Click"/> <br /> <br />
          <asp:Label ID="lblTexto" runat="server" Text=""></asp:Label>
       </div>
    </form>
</body>
</html>
```

Modelo de código subyacente

Ejemplo3-2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Ejemplo3-2.aspx.cs"
         Inherits="EjemplosTeoriaT3.Ejemplo3_2" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h1>Ejemplo 3-2</h1> <hr />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label">
                  Introduzca su nombre: </asp:Label>
            <asp:TextBox ID="txtNombre" runat="server"></asp:TextBox> <br /> <br />
            <asp:Button ID="btnEnviar" runat="server" Text="Enviar“
                        OnClick="btnEnviar_Click"/> <br /> <br />
            <asp:Label ID="lblTexto" runat="server" Text=""></asp:Label>
        </div>
    </form>
</body>
</html>
```



Modelo de código subyacente

Ejemplo3-2.aspx.cs

```
using System;

namespace EjemplosTeoriaT3
{
    public partial class Ejemplo3_2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void btnEnviar_Click(object sender, EventArgs e)
        {
            if (txtNombre.Text != "")
            {
                lblTexto.Text = "Bienvenido a ASP.NET " + txtNombre.Text;
            }
            else
            {
                lblTexto.Text = "Debe introducir su nombre en el cuadro de texto";
            }
        }
    }
}
```

Modelo de código subyacente

- Referencia al archivo de código subyacente vinculado con la página de ASP.NET

Ejemplo3-2.aspx

```
<%@ Page
    Language="C#"
    AutoEventWireup="true“
    CodeBehind ="Ejemplo3-2.aspx.cs“
    Inherits="EjemploT3.Ejemplo3_2“ %>
<!DOCTYPE html PUBLIC
. . .
```

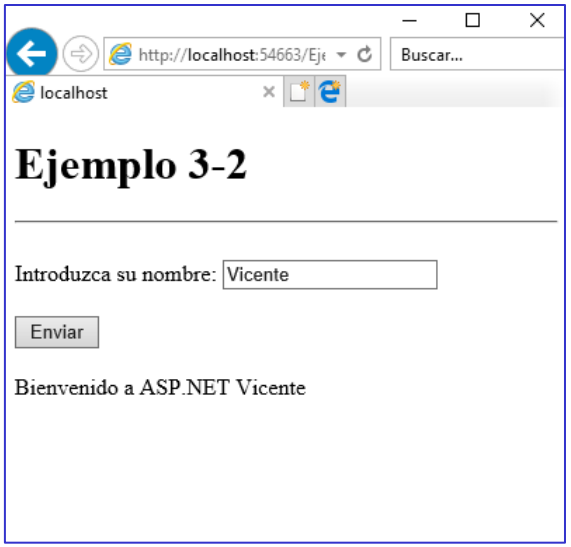
Ejemplo3-2.aspx.cs

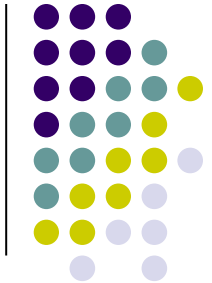
```
namespace EjemploT3
{
    public partial class Ejemplo3_2:
        System.Web.UI.Page
    {
        protected void Page_Load
            (object sender, EventArgs e)
        {
            . . .
        }

    . . .

}
```

Ciclo del procesamiento de una página de ASP.NET

1) Se solicita una página de ASP.NET o *Web Form* desde el navegador
2) La página de ASP.NET se procesa en el servidor Web y se envía la página HTML de respuesta hacia el navegador Web (cliente)
3) El navegador presenta el código HTML de la página de respuesta
4) El usuario introduce o selecciona información de entre las opciones disponibles en el formulario y, acaba haciendo clic en el botón de envío
5) En ese momento, se produce el envío de los datos del formulario hacia el servidor Web. En ASP.NET, esta acción se denomina **devolución de datos**. De forma predeterminada, el destino de los datos de un formulario para su procesamiento en el servidor es la misma página de ASP.NET
6) La página de ASP.NET se procesa de nuevo en el servidor Web, utilizando la información recibida a través del formulario
7) Finalizado el procesamiento, **el servidor envía la página de respuesta** al cliente, incluyendo los resultados del procesamiento en el servidor Web
8) La página de repuesta HTML se reconstruye en el navegador

Este ciclo continúa durante todo el tiempo que el usuario esté utilizando la página de ASP.NET. Cada vez que el usuario hace clic en un botón de envío, u otro control activo, la información de la página se manda al servidor y la página se procesa con los nuevos datos del formulario. Cada uno de estos ciclos se conoce como **recorrido de ida y vuelta o *Post-Back***

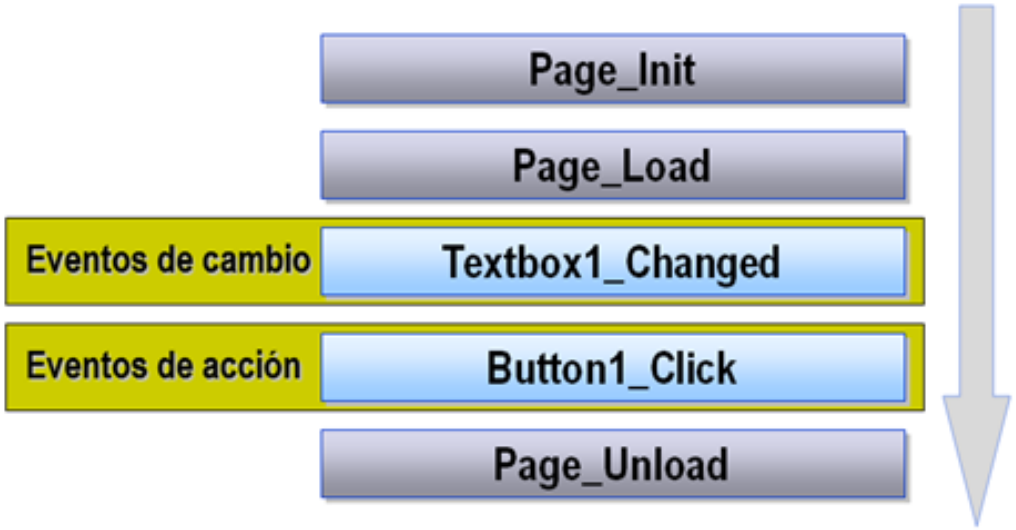




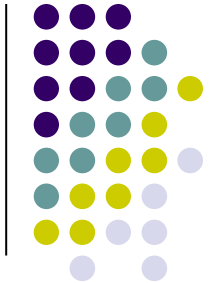
Mantenimiento del estado de la página

- ASP.NET ayuda a conservar la información de una página:
  - Se denomina estado de la página al conjunto de los valores de las propiedades de todos los controles de servidor incluidos en la página
  - De forma predeterminada, **los valores de las propiedades de los controles de servidor de la página se conservan entre las acciones de ida y vuelta**
    - Lo que permite que los valores de los controles de servidor de la página sean accesibles desde el código lógico de procesamiento en el servidor Web en todo momento
    - Esta particularidad de la tecnología ASP.NET es posible debido a que:
      - Se utilizan controles de servidor Web para crear la página Web de ASP.NET o Web Form
      - Una página de ASP.NET se devuelve datos a sí misma, de forma predeterminada
  - Se puede **detectar cuándo se solicita una página por primera vez y cuándo se envían datos hacia el servidor *(Post-Back)***, lo que permite especificar la lógica requerida en estos casos
    - Por ejemplo, se podría desear leer cierta información de una base de datos la primera vez que se muestre una página, pero no en cada devolución de datos

Eventos de página y eventos de controles de servidor



- **Una página Web de ASP.NET y los controles de servidor que incluye, admiten un modelo de desarrollo controlado por eventos**
  - Es posible especificar el código lógico a ejecutar por el servidor Web (controlador de evento) como respuesta a un evento (Click) de un **control de servidor** (Button)
  - Además, desde el código lógico de una página de ASP.NET puede tenerse también acceso a la clase **Page**. El objeto que instancia la clase Page:
    - Representa a la página de ASP.NET actual
    - Sirve como contenedor de todos los controles que constituyen la página



Eventos de controles de servidor

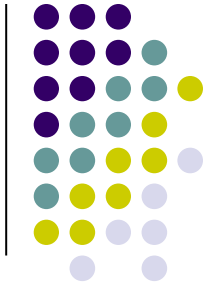
- Los **controles de servidor** contenidos en una página de ASP.NET pueden provocar sus propios eventos, atendiendo a los sucesos que realicen los usuarios al interactuar con la página de ASP.NET
  - Los controladores de evento se establecen en el código lógico del archivo de código subyacente (.aspx.cs) a través de procedimientos de evento

```
protected void btnEnviar_Click(object sender, EventArgs e)
{
    lblValores.Visible = true;
    lblValores.Text = "VALORES DEL FORMULARIO" +
           "<br/> Código Empleado: " + txtCodEmp.Text +
           "<br/> Departamento: " + ddlDepEmp.Text;
}
```

  - La asociación entre el controlador de evento y el control se establece en el código de marcado (.aspx), a través de un atributo circunstancial definido en la etiqueta del control

```
<asp:Button ID="btnEnviar" runat="server" Text="Enviar“
            onclick="btnEnviar_Click" />
```

En el ejemplo, al atributo circunstancial ***onclick*** se le asigna el valor ***“btnEnviar_Click”***

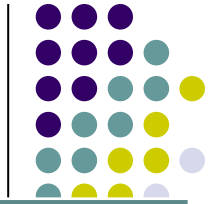


Eventos de página

- El **evento Page_Load** se invoca en cada solicitud de la página de ASP.NET, sin embargo el uso de la propiedad ***IsPostBack*** del objeto ***Page*** permite establecer la lógica para evita la recarga en cada *Post-Back*

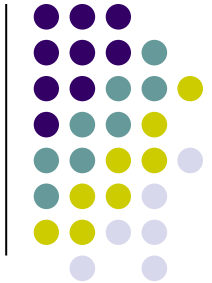
```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        // Se ejecuta solo en la carga inicial del Web Form
    }
    else
    {
        // Este código se ejecuta cuando es una devolución de datos
    }
    // Este código se ejecuta en cada petición
}
```

De este modo se puede detectar cuándo se solicita una página de ASP.NET por primera vez y cuándo se envía como consecuencia de una devolución de datos en una acción de ida y vuelta *(Post-Back),* lo que permite especificar la lógica requerida en cada caso



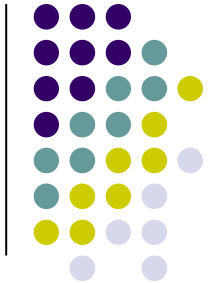
Clases de ASP.NET

Objeto *Clase de ASP.NET*	Descripción
Response ***HttpResponse***	Representa la **respuesta desde el servidor** hacia el cliente. Proporciona acceso al flujo de salida de la página actual hacia el cliente. Se puede utilizar esta clase para insertar texto en la página, escribir cookies, etc.
Request ***HttpRequest***	Representa la **petición desde el cliente**. Se puede utilizar esta clase para tener acceso a la solicitud de página actual: leer la información que se ha enviado desde el navegador, la cadena de consulta, los certificados de cliente, etc.
Context ***HttpContext***	Representa la **ejecución de la aplicación Web**. Proporciona acceso al contexto actual. Se puede utilizar esta clase para compartir información entre páginas
Server ***HttpServerUtility***	Representa el **servidor Web**. Expone métodos de utilidad que puede utilizar para transferir el control entre páginas, obtener información sobre el error más reciente, codificar y descodificar texto HTML, etc.
Application ***HttpApplicationState***	Representa la **aplicación Web**. Proporciona acceso a métodos y eventos de aplicación Web para todas las sesiones activas. También proporciona acceso a una caché de la aplicación que se puede utilizar para almacenar información
Session ***HttpSessionState***	Representa la **sesión de un usuario**. Proporciona acceso a métodos y eventos de la sesión actual de un usuario. También proporciona acceso a una caché de sesión que se puede utilizar para almacenar información durante la sesión de un usuario.
Trace ***TraceContext***	Representa la **ejecución de la página**. Proporciona un modo de mostrar mensajes de diagnóstico de seguimiento personalizados y del sistema en el resultado de la página HTTP



Objetos Application y Session de ASP.NET

- Una característica interesante de la tecnología ASP.NET es que proporciona funcionalidades de **administración del estado de la aplicación y de la sesión a través de los objetos Application y Session de ASP.NET**
  - Permiten guardar variables personalizadas e información del contexto de la aplicación, que afectan a todos los usuarios, o de la sesión, que afectan a todas las páginas de la sesión de un determinado usuario
  - El **estado de la aplicación** se inicia cuando se realiza la primera solicitud a una página de una aplicación Web y finaliza después de que el último usuario deje de utilizarla
    - El objeto **Application** es común a todos los usuarios activos de la aplicación Web
  - El **estado de la sesión** se inicia cuando un usuario realiza la primera solicitud a una página de una aplicación Web y finaliza después de que ese usuario deje de utilizarla
    - El objeto **Session** es común a todas las páginas de ASP.NET que utiliza un usuario durante su sesión, por lo que suele utilizar para almacenar información del contexto de la sesión que es compartida por todas páginas durante la duración de la sesión actual de un usuario



Generalidades

- Es uno de los lenguajes más utilizados. Sus principales características son:
  - Es un lenguaje **Orientado a Objetos**
  - Es un lenguaje que incorpora un modelo de **Programación Controlada por Eventos**
  - Es un lenguaje de programación ampliamente utilizados. Su sintaxis es similar a otros lenguajes como C, C++ o Java
  - Distingue entre mayúsculas y minúsculas en el código
  - Incorpora las estructuras de control típicas, tales como: secuencia, selección y repetición
  - Permite una gestión estructurada de excepciones, con la que controlar los errores o las situaciones que puedan ocurrir durante la ejecución

Un archivo anexo incorpora, a modo de resumen, los aspectos más relevantes de la sintaxis del lenguaje de programación C#