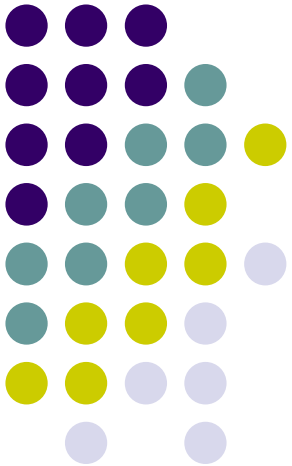
Desarrollo de Aplicaciones Web

Desarrollo Web en Entornos de Servidor



Tema 4

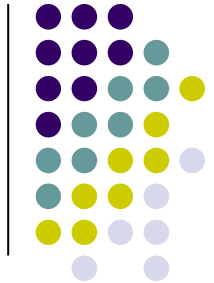
Desarrollo de aplicaciones Web

Vicente J. Aracil Miralles

vj.aracilmiralles@edu.gva.es

06/09/2022

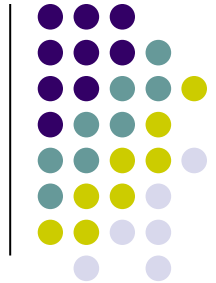
Tema 4
Desarrollo de aplicaciones Web



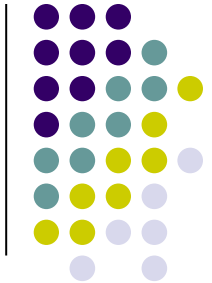
Objetivos

- Reconocer la importancia del planeamiento de las Aplicaciones Web
- Conocer conceptos sobre pruebas, depuración y corrección de errores de programación lógica en páginas Web y aplicaciones Web interactivas
- Utilizar y definir los mecanismos que permiten realizar la validación de la información introducida por el usuario en controles Web a través de formularios
- Utilizar y definir controles Web reutilizables creados por el usuario para su inserción en páginas Web como una parte constitutiva de estas
- Utilizar y definir páginas maestras para la composición visual de páginas Web
- Conocer los métodos para manejar el mantenimiento del estado de la sesión y de la aplicación en aplicaciones Web interactivas
- Conocer los aspectos más significativos sobre globalización y accesibilidad Web
- Comprender la utilización de los modelos de desarrollo de soluciones software basadas en la utilización de patrones de software en el servicio Web

Tema 4
Desarrollo de aplicaciones Web



Contenidos

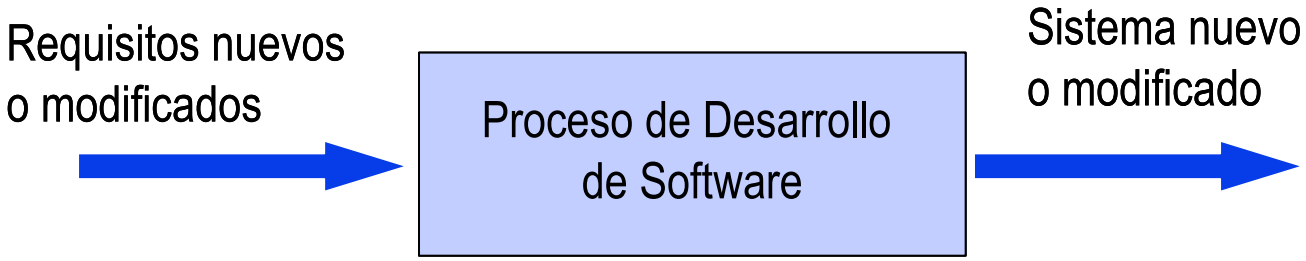


Planeamiento de una aplicación Web

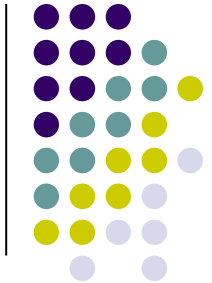
- Planear una aplicación Web significa tomar decisiones sobre diversos aspectos relevantes que se refieren a su desarrollo
  - Estas decisiones tienen por objeto especificar y definir:
    - Tipo de sitio Web: estática, dinámica o interactiva
    - Tecnologías implicadas en el desarrollo
    - Diseño coherente del sitio Web
    - Apariencia visual homogénea para todas las páginas Web que la forman
    - Controles y mecanismos de navegación a emplear
    - Modo de acceso a los datos almacenados en un origen de datos
    - Cumplimiento de las normas de Accesibilidad Web
    - Requisitos de Globalización y Localización para facilitar el funcionamiento de la aplicación con diversas referencias culturales y configuraciones regionales
  - Antes de empezar a crear el sitio Web, es decir, antes de empezar a crear las páginas y escribir el código de un sitio Web, siempre es útil planear el sitio Web

Proceso del desarrollo de software

- En general, tiene como propósito la producción eficaz y eficiente de un sistema software que cumpla y reúna los requisitos del cliente
  - Este proceso es intensamente intelectual



- Un proyecto de desarrollo de software es equiparable en muchos aspectos a cualquier otro tipo de proyecto de ingeniería, aunque incorpora algunas particularidades propias derivadas de:
  - La complejidad en sí del software, lo que impide una verificación exhaustiva
  - La intangibilidad del software, lo que implica un alto nivel de abstracción
  - La inmadurez de la ingeniería del software como disciplina

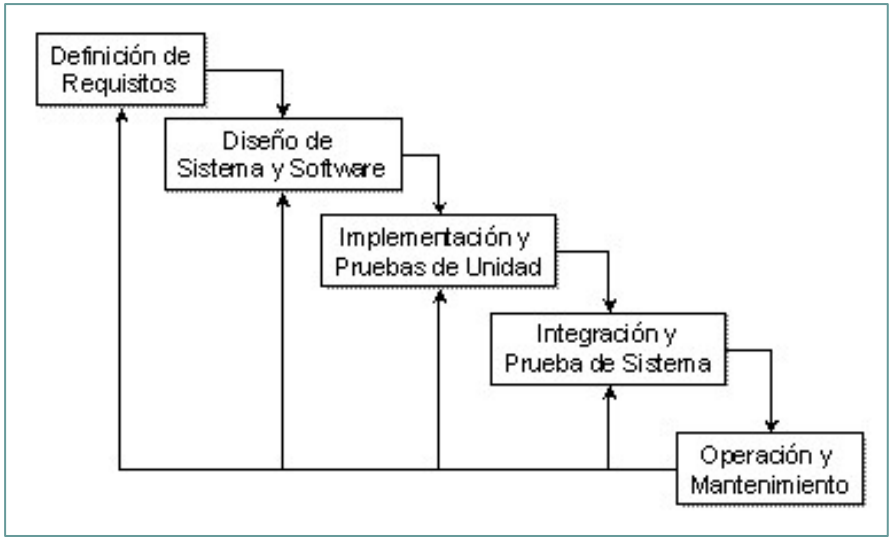


Proceso del desarrollo de software

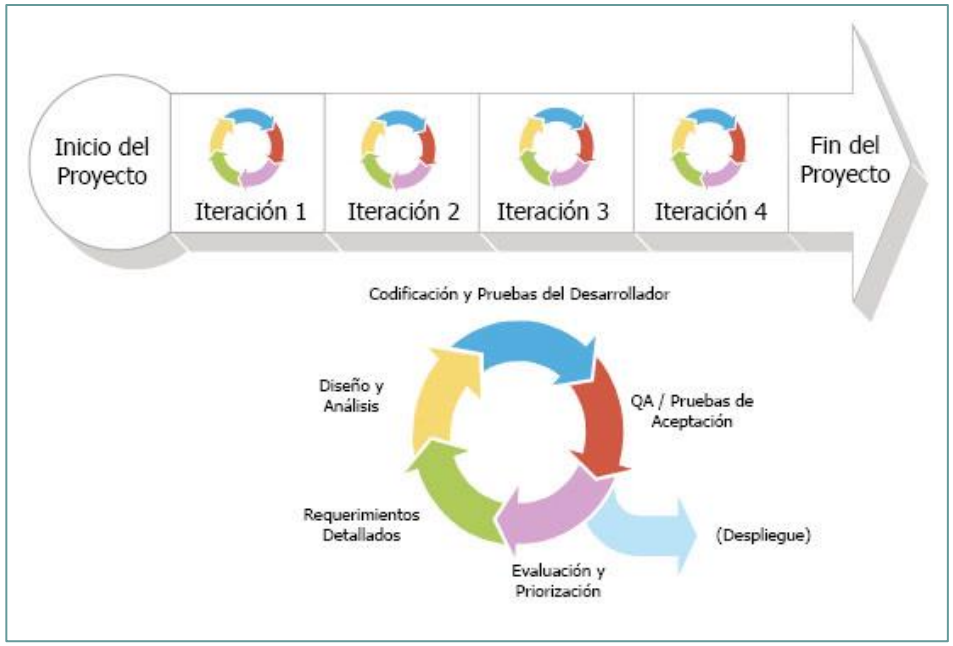
- No existe un proceso universal para el desarrollo de software que sea efectivo para todos los contextos de los proyectos de desarrollo
  - Las diferentes propuestas existentes suelen coincidir al definir las siguientes actividades fundamentales:
    1. **Especificación de software.** Se define la funcionalidad y las restricciones operacionales que debe cumplir el software
    2. **Diseño.** Se diseña el sistema software de acuerdo a la especificación realizada
    3. **Implementación.** Se construye el sistema software de acuerdo con el diseño realizado
    4. **Validación.** El sistema software debe validarse, para asegurar que cumple con los requisitos y la funcionalidad establecida en las especificaciones
    5. **Evolución.** El software debe evolucionar, para adaptarse a futuras necesidades

4.1 Planear el desarrollo de una aplicación Web

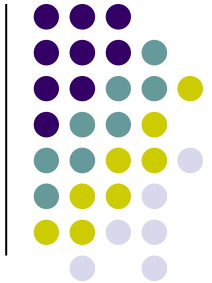
Proceso del desarrollo de software



Modelo de desarrollo de software en cascada (Tradicional)

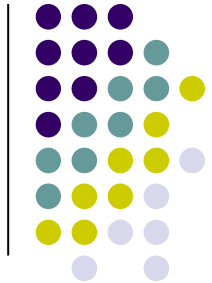


Modelo de desarrollo de software iterativo e incremental *(Agile development)*



Diagnóstico de errores en tiempo de ejecución

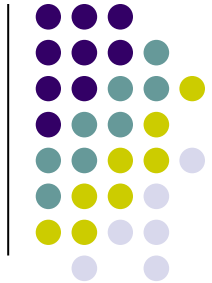
- La tecnología de desarrollo ASP.NET incluye funcionalidades que ayudan a diagnosticar los problemas y errores que pueden surgir en las aplicaciones Web en tiempo de ejecución
- Entre las funcionalidades de diagnóstico disponibles destacan:
  - Utilización de un **depurador** para realizar un seguimiento de la ejecución de una página o de una aplicación Web mientras se ejecuta
  - **Traza de solicitudes de página** para recopilar la información sobre cada uno de los pasos del procesamiento de las páginas de la aplicación Web



Depuración

- El código de las aplicaciones puede contener distintos tipos de errores
  - La mayor parte de los errores de sintaxis se detectan durante la compilación
  - Sin embargo, hay tipos de errores que requieren que se depure el código
- Microsoft Visual Studio incluye una herramienta de depuración denominada **Visual Debugger** que permite examinar el funcionamiento de una aplicación Web mientras se está ejecutando, para poder:
  - Realizar un **seguimiento del código en tiempo de ejecución**, mediante: la definición de puntos de interrupción, el recorrido de ejecución paso a paso y la inspección de los valores almacenados en cada variable
  - **Proporcionar información de los errores** que se puedan producir en tiempo de ejecución

Las técnicas de depuración consisten en realizar un seguimiento de la ejecución de la aplicación Web para comprobar su funcionamiento y poder examinar las causas de los posibles errores



Depuración

- Para poder utilizar el depurador y así poder corregir los errores que se puedan producir en tiempo de ejecución, es necesario habilitar el seguimiento de la ejecución en cualquiera de sus dos tipos:
  - **A nivel de aplicación.** La configuración de una aplicación Web para su compilación en una versión de depuración se realiza en el archivo ***Web.config*** de la aplicación Web

```
<configuration>
    <system.Web>
        <compilation debug="true" targetFramework="4.8"/>
    </system.Web>
</configuration>
```

  - **A nivel de página.** Como alternativa, se puede agregar debug=true a la directiva **@ Page** solo en la página concreta que se desea depurar

```
<%@ Page Language="C#" AutoEventWireup="true“ CodeBehind=“Empleados.aspx.cs"
        Inherits=“Ejemplo.Empleados“ debug="true" %>
```

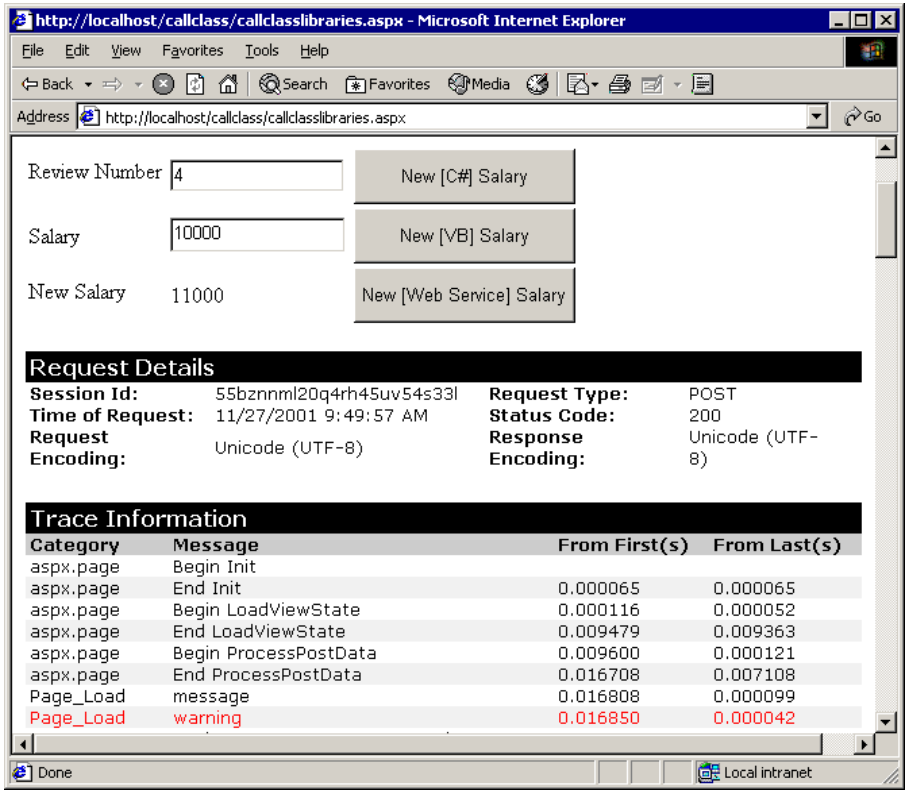
Traza de solicitudes de página

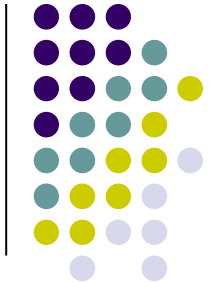
- El objeto **Trace** de ASP.NET facilita la recopilación de información sobre el procesamiento de código en tiempo de ejecución:
  - Información de diagnóstico
  - Mensajes de traza personalizados

La información obtenida de la traza (objeto Trace):

- **Se añade al flujo de salida de la página Web de respuesta que se envía al cliente**
- **Puede ayudar a investigar errores o resultados no deseados mientras ASP.NET procesa una solicitud**

Las instrucciones de traza sólo se procesan y se muestran cuando la traza está habilitada





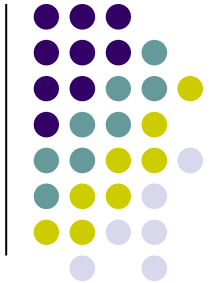
Traza de solicitudes de página

- Para visualizar la información de la traza sobre las páginas Web, es necesario hablitar su uso a nivel de aplicación o a nivel de página:
  - **A nivel de aplicación.** Se puede configurar el archivo de configuración *Web.config* de la aplicación Web para que todas las páginas muestren información de la traza a menos que una página se deshabilite explícitamente. Se habilita mediante el elemento *trace*

```
<configuration>
  <system.Web>
     <compilation debug="true" targetFramework="4.8"/>
     <trace enabled="true" pageOutput="true"/>
  </system.Web>
</configuration>
```

  - **A nivel de página.** Como alternativa, se pueden configurar las páginas individuales para mostrar información de la traza en su contenido resultante.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Empleados.aspx.cs"
         Inherits="Ejemplo.Empleados" trace="true" %>
```

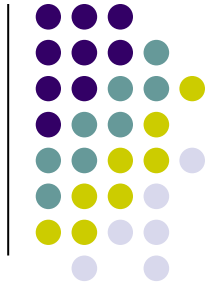


Traza de solicitudes de página

- Información de seguimiento de la traza de la aplicación

Nivel de Página	Nivel de Aplicación	Resultado
trace = true	trace=true ó trace=false	Los resultados de seguimiento se muestran en la página
trace = false	trace=true ó trace=false	Los resultados de seguimiento no se muestran en la página
trace no definido	trace=true	Los resultados de seguimiento se muestran en todas las páginas

- Las instrucciones de seguimiento a nivel de aplicación se muestran en páginas individuales
- Si aparece ***pageOutput="false"*** en el archivo *Web.config*, entonces los resultados de la traza se pueden visualizar en el visor de seguimiento, al cual se accede a través de la dirección web: **http://servidor/proyecto/trace.axd**



Traza de solicitudes de página

- Algunas funcionalidades del objeto Trace desde el código lógico:
  - Insertar mensajes de seguimiento

```
Trace.Write ("category", "message")
Trace.Warn ("category", "message")      // Advertencia (warning)
```

  - Ejecución condicional con Trace.IsEnabled

```
if (Trace.IsEnabled)  {
   strMsg = "Tracing is enabled!";
   Trace.Write("myTrace", strMsg);
}
```

  - Cambiar dinámicamente el estado del seguimiento

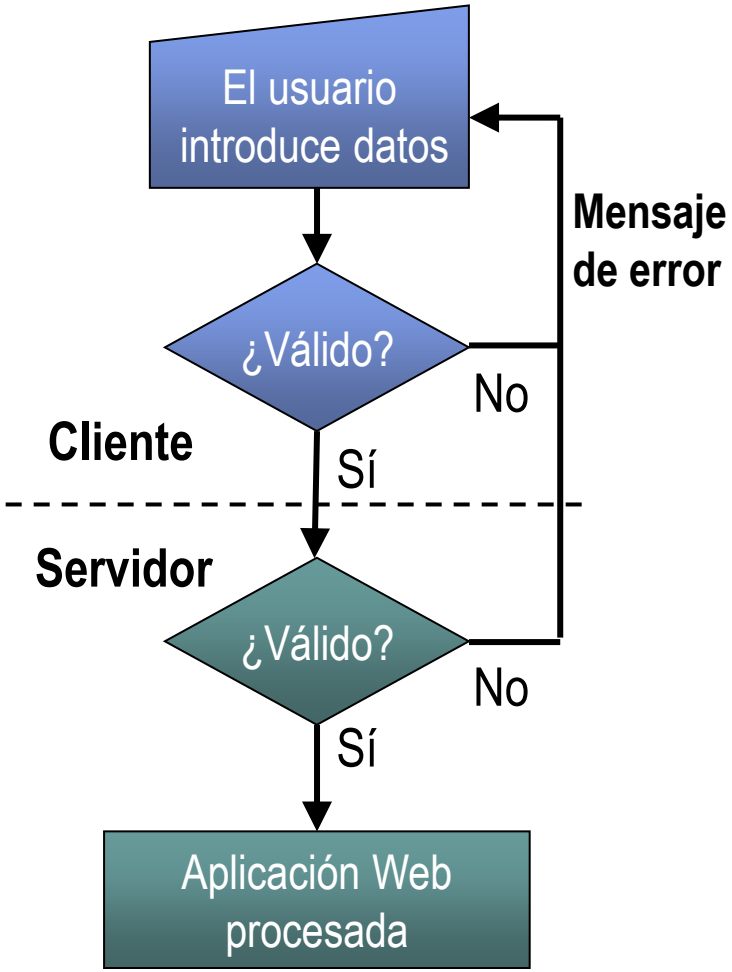
```
Trace.IsEnabled = false
```

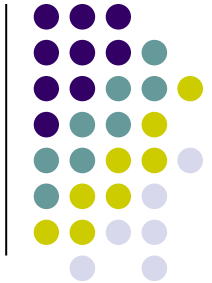
Controles de validación de ASP.NET

- Los **controles de validación** permiten comprobar los valores que introducen los usuarios en los controles de servidor de ASP.NET y de HTML que forman parte de una página
  - Cada control de validación hace referencia a un control de servidor de la página
  - Al enviar la página, cada control de validación comprueba los valores y establece si la información introducida por el usuario es válida
  - De forma predeterminada, **se bloquea el procesamiento de la página** hasta que los valores introducidos en todos los controles de servidor de la página sean válidos
    - Puede inhibirse este comportamiento predeterminado asignando el valor *False* a la propiedad **CausesValidation** de algunos controles

Validación en el lado del cliente y en lado del servidor

- ASP.NET puede generar validación en el lado del cliente y en el lado del servidor
- Validación en el lado del cliente
  - Dependiente de la configuración de seguridad del navegador
  - El proceso de validación puede resultar complejo
  - Reduce los ciclos de *Post-Back*
- Validación en el lado del servidor
  - Puede validar contra datos almacenados
  - Mayor seguridad
  - Por razones de seguridad, debe repetirse en el servidor la misma validación que se haya incluido en el lado del cliente

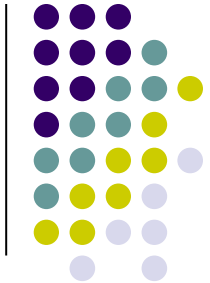




Controles de validación

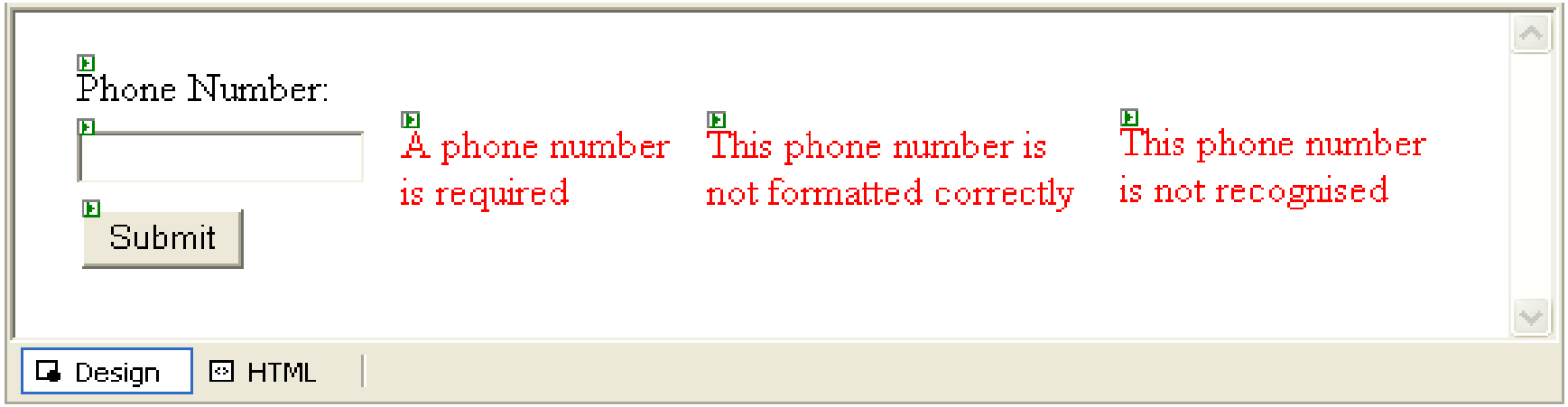
Tipo de validación	Utilizar el Control de validación
Entrada requerida	**RequiredFieldValidator**
Comparación con un valor	**CompareValidator**
Comprobación con un rango o intervalo de valores	**RangeValidator**
Coincidencia con un modelo de expresión regular o patrón	**RegularExpressionValidator**
Definida por el usuario	**CustomValidator**
Muestra toda la información de errores de validación de una página en forma de resumen	**Validation summary**

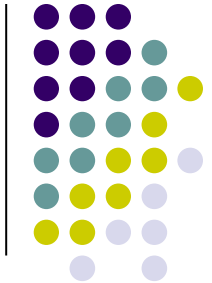
- Para incluir controles de validación a una página de ASP.NET:
  1. Agregar el tipo de control de validación adecuado
  2. Seleccionar el control a validar, mediante la propiedad **ControlToValidate**
  3. Establecer las propiedades de validación adecuadas: ErrorMessage, Text, etc.



Validación compuesta sobre un mismo control

- Pueden existir múltiples controles de validación sobre un único control de servidor de entrada de datos
  - Únicamente el control de validación *RequiredFieldValidator* comprueba los controles vacíos



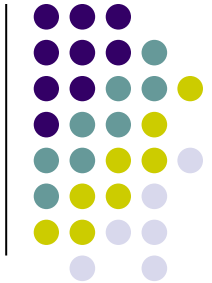


Validez de la página mediante código lógico

- Utilizando código lógico puede comprobarse el estado de validación de la página y de los controles individuales

```
private void cmdSubmit_Click(object s, System.EventArgs e)
{
    if (Page.IsValid)
    {
        Message.Text = “¡La página es válida!";

        // Incluir aquí la lógica del proceso a realizar

    }
}
```

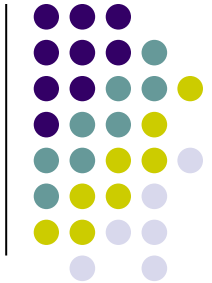
  - De forma predeterminada, si una comprobación de validación produce errores, se omite todo el procesamiento de código lógico en el servidor y se devuelve la página al usuario



Controles propios del desarrollador

- En ocasiones, es posible que se precise cierta funcionalidad específica en un control que:
  - No está incluida en los controles de servidor de ASP.NET integrados
  - Pueda ser reutilizada en varias páginas Web de ASP.NET
- En estos casos, se pueden crear controles Web propios:
  - **Controles de usuario.** Son contenedores de elementos HTML y controles de servidor. Una vez creado, un control de usuario se puede tratar como una unidad y definir sus propiedades y métodos
  - **Controles personalizados.** Es una clase de código escrito que deriva de la clase Control o WebControl

Los controles de usuario son más fáciles de crear que los controles personalizados, ya que es posible reutilizar los controles de servidor ya existentes para crear con controles complejos y reutilizables

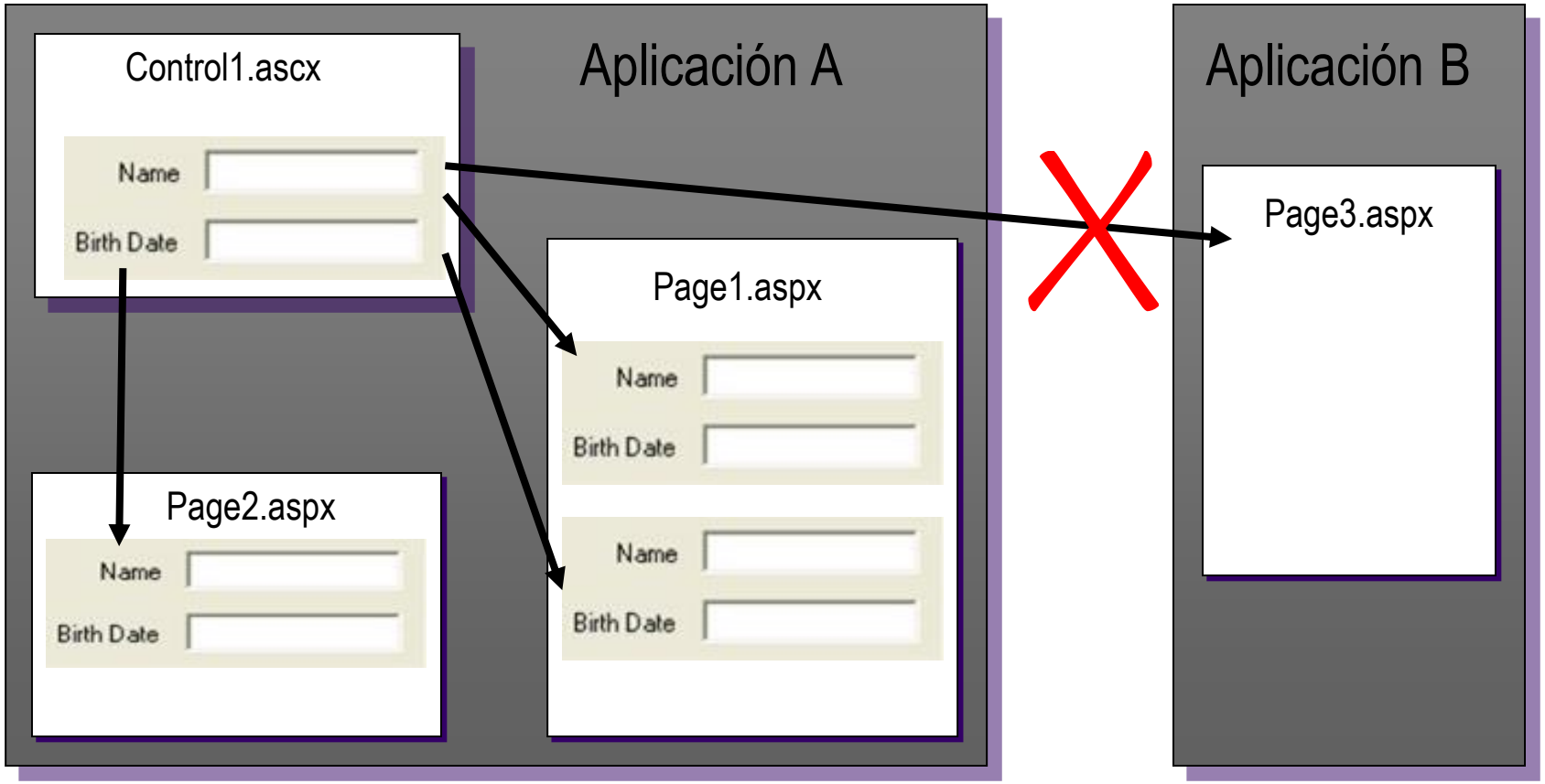


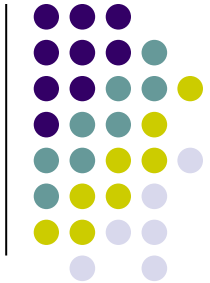
Creación de un control de usuario

- El proceso de creación del control de usuario es muy similar al proceso de creación de una página ASP.NET
  - Incluye una página de interfaz de usuario y un archivo de código subyacente asociado
- Un control de usuario se diferencia de una página Web de ASP.NET en los siguientes aspectos:
  - La extensión de nombre de archivo para el control de usuario es **.ascx**
  - En lugar de una directiva @ Page, el control de usuario contiene una directiva **@ Control** que define la configuración y otras propiedades.
  - El control de usuario no contiene elementos html, body o form. Estos elementos estarán definidos en la página de huésped
  - Los controles de usuario **no se pueden ejecutar de forma independiente**. Deben ser agregados a las páginas ASP.NET de la aplicación Web como cualquier otro control de servidor

Reutilización de controles de usuario

- Los controles de usuario permiten reutilizar la interfaz de usuario y el código lógico en el ámbito de una aplicación Web





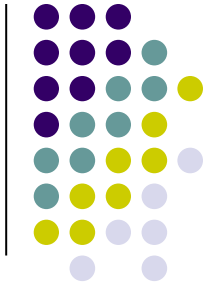
Agregar un control de usuario a una página de ASP.NET

- Es necesario definir la directiva **@ Register** en la página de ASP.NET que actúa como huésped *(host)* para especificar el control de usuario a reutilizar

```
<%@ Register TagPrefix="demo“
      TagName="validNum" Src="numberbox.ascx" %>
```

- Además, la inserción de un control de usuario sobre la página de Asp.NET se realiza mediante código de marcado en el archivo .aspx de la página de la siguiente forma

```
<demo:validNum id="num1" runat="server"/>
```



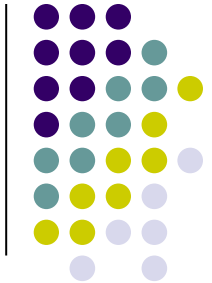
Métodos get y set para el acceso a las propiedades

- La página de ASP.NET, que actúa como huésped *(host)*, puede interactuar con el control de usuario utilizando los descriptores de acceso **get** (obtener) y **set** (establecer) para tener acceso a las propiedades. **Ejemplo**:
  - Para especificar el acceso a las propiedades en el archivo de código subyacente de la página de ASP.NET donde se ha incluido el control de usuario, se haría:

```
x = num1.pNum;              // utiliza Get
num1.pNum = 5;              // utiliza Set
```

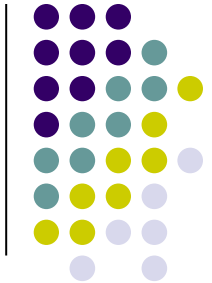
  - El código que define los descriptores de acceso get y set en el archivo de código subyacente del control de usuario (.ascx.cs), sería:

```
public int pNum
{
   get {  return Convert.ToInt32(txtNum.Text);  }
   set {  txtNum.Text = Convert.ToString(value);  }
}
```



Concepto

- Las páginas maestras de ASP.NET permiten crear un diseño coherente y una apariencia visual homogénea para todas las páginas de una aplicación Web
  - Una página maestra permite definir el aspecto, el diseño y el comportamiento estándar que se desea tengan todas las páginas, o un grupo de páginas, de la aplicación Web
  - Una vez creada una página maestra, se pueden crear páginas de contenido que incluyen el contenido específico que desea mostrar en cada página concreta
- Cuando los usuarios solicitan las páginas de contenido:
  - Las páginas de contenido se combinan con la página maestra con la que están asociada para generar una salida que **combina el diseño de la página maestra con la página de contenido**



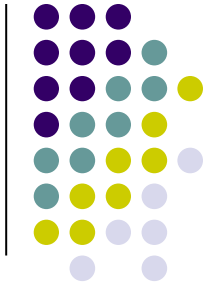
Páginas maestras

- Una página maestra es un archivo de ASP.NET que especifica un diseño predefinido que puede incluir texto estático, elementos HTML y controles de servidor
- Sus características principales son:
  - Es un archivo de ASP.NET con la extensión: **.master**. Por ejemplo *MySite.master*
  - Se identifican mediante una directiva **@ Master** que reemplaza a @ Page. La directiva @ Master incluye el nombre del archivo de código subyacente asociado y asigna el nombre de clase a la página maestra.

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

  - Incluye todos los elementos HTML de nivel superior, como html, head, body y form
  - Puede incluir cualquier elemento HTML y cualquier control de ASP.NET
  - Siempre incluye uno o varios controles **ContentPlaceHolder** que son **marcadores de posición de contenido remplazable**

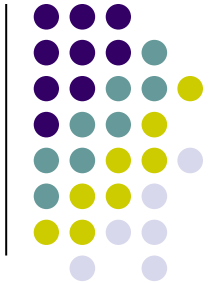
Marcadores de posición de contenido reemplazable



- Además del contenido común, una página maestra incluye uno o varios controles **ContentPlaceHolder**, que definen las **regiones de contenido reemplazable**

```
<%@ Master Language="C#" AutoEventWireup="true“
           CodeBehind="MasterPage.master.cs" Inherits=“Ejemplo.MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div class=“Titulo”> Aplicación Web GESPROM </div>
    <div>
        <asp:ContentPlaceHolder id="contenido" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Marcador de posición de contenido reemplazable
(El contenido se define en las páginas de contenido)



Páginas de contenido

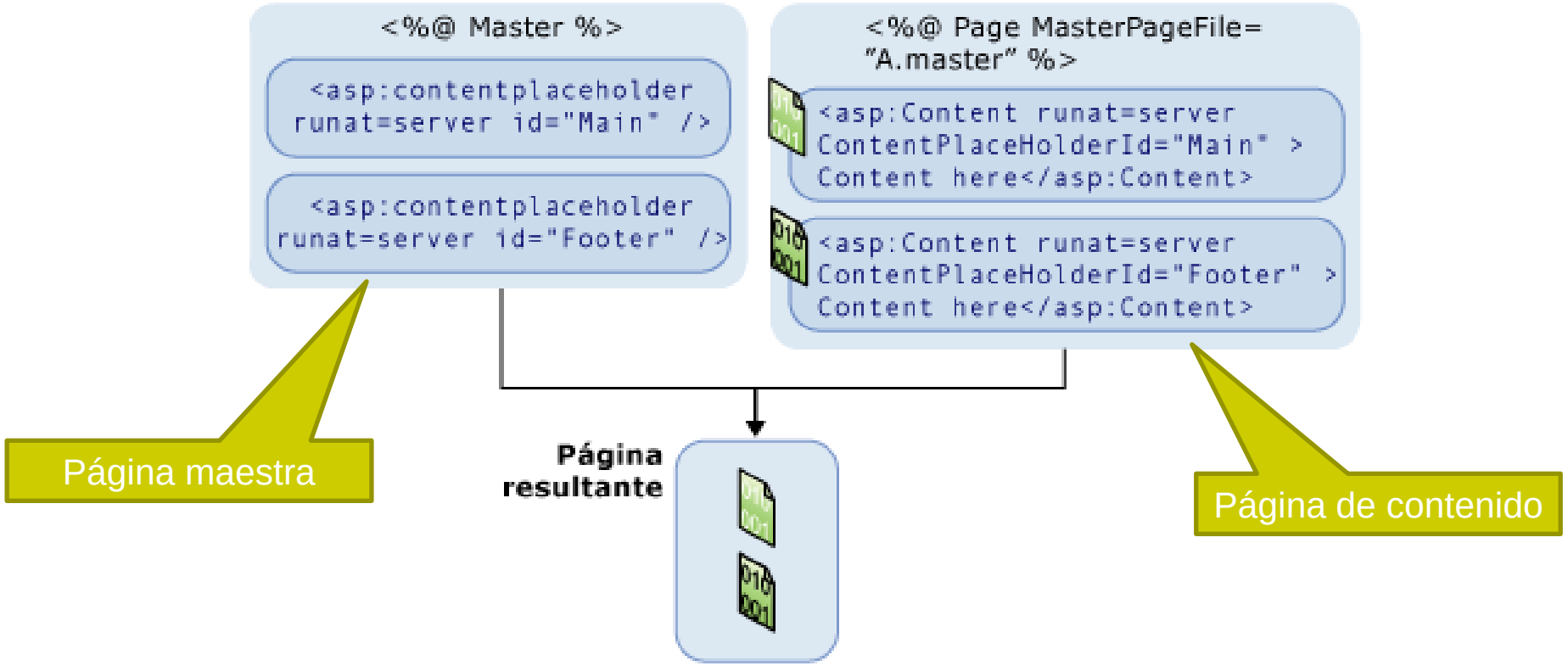
- El contenido de los controles **ContentPlaceHolder** de la página maestra, se define en las páginas de contenido que incluyen el contenido específico de esa página:
  - Las páginas de contenido son páginas de ASP.NET **(.aspx)** que están enlazadas con una página maestra concreta. Sus características son:
    - El enlace con Página Maestra se establece mediante el atributo *MasterPageFile* de la directiva **@ Page** en la página de contenido
    - En la página de contenido se agregan controles **Content** que se asignan a los controles ContentPlaceHolder de la página maestra
    - El contenido específico de la página se define dentro de las etiquetas de los controles **Content**

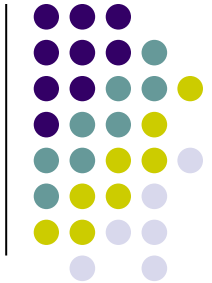
```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
    AutoEventWireup="true" CodeBehind="Default2.aspx.cs" Inherits="Ejemplo.Default2"%>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="contenido" Runat="Server">

</asp:Content>
```

Contenido específico de la página

Combinación de páginas maestras y de contenido

- **Desde la perspectiva del usuario**
  - La combinación de una página maestra con la página de contenido en tiempo de ejecución da como resultado una única página



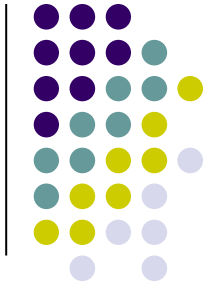


Combinación de páginas maestras y de contenido

- **Desde la perspectiva del desarrollador**
  - Las páginas maestras y las páginas de contenido actúan como contenedores diferentes para sus respectivos controles
  - La página de contenido actúa como un contenedor de la página maestra
    - Así, la página maestra pasa a formar parte de la página de contenido
    - El **árbol de objetos** resultante de la combinación es:

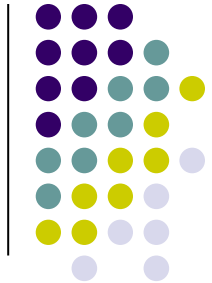
```
Page
      Master Page
      (Controles y marcado de la página maestra)
      ContentPlaceHolder
            (Controles y marcado de la página de contenido)
      (Controles y marcado de la página maestra)
```

    - Desde el código lógico de la página de contenido no se puede obtener acceso directamente a los controles de la página maestra
      - Se puede hacer referencia a los valores de los controles de la página maestra desde el código lógico de las páginas de contenido, a través de los descriptores de acceso get y set de la página maestra, de forma similar a cómo se realiza en el caso de los controles de usuario



Información de estado en el servidor

- ASP.NET proporciona la posibilidad de **mantener la información del estado en el servidor**, en lugar de conservarla en el cliente
  - La administración del estado basada en servidor puede reducir la cantidad de información que se envía al cliente. Sin embargo, puede suponer un uso excesivo de los recursos del servidor, especialmente de la memoria
- Características de administración del estado basada en servidor:
  - **Estado de aplicación**
    - Es un repositorio de datos de tipo identificador-valor disponible para todas las páginas de la aplicación Web. Almacena información global del **ámbito de la aplicación** que es accesible por todos los usuarios y sesiones activas y que se mantiene en las acciones de *Post-Back*
  - **Estado de sesión**
    - Es un repositorio de datos de tipo identificador-valor disponible para cada usuario en el ámbito de su sesión. El ámbito de la sesión de usuario activa viene determinado por la instancia de ejecución del navegador en el que se ejecuta la aplicación Web
    - Es útil para almacenar información común relativa a la sesión de cada usuario que se conserva en las acciones de *Post-Back* y entre las solicitudes de las páginas



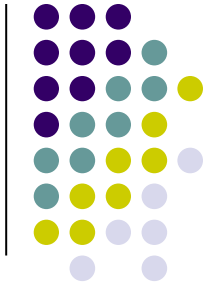
Variables de aplicación

- Para escribir un valor en el estado de aplicación
  - Se asigna el valor a una variable de aplicación desde el código lógico. Por ejemplo:

```
Application["Message"] = " Bienvenidos al sitio Web.";
Application["PageRequestCount"] = 0;
```

- Para leer un valor desde el estado de aplicación
  - Se suele comprobar si la variable de aplicación existe y, a continuación, se convierte al tipo adecuado si se tiene acceso a ella
  - El estado de aplicación almacena datos de tipo Object, por lo que es necesario convertir el valor al tipo adecuado al recuperarlo

```
if (Application["AppStartTime"] != null)
{
    DateTime myAppStartTime = (DateTime)Application["AppStartTime"];
}
```



Variables de sesión

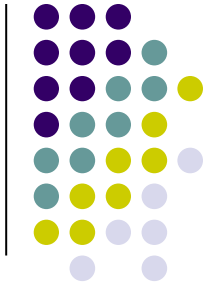
- Para escribir valores en el estado de sesión
  - Se asigna el valor a una variables de sesión desde el código lógico

```
string strNombre = "Smith";
string strCiudad = "Seattle";
Session[“Nombre"] = strNombre;
Session["Ciudad"] = strCiudad;
```

- Para leer los valores almacenados en el estado de sesión
  - Es necesario convertirlos al tipo de datos adecuado, porque son de tipo Object:

```
string strNombre = (string)(Session[“Nombre"]);
string strCiudad = (string)(Session["Ciudad"]);
```

- El estado de la sesión puede expirar y perderse la información almacenada
  - El tiempo predeterminado de duración de la sesión es de 20 minutos de inactividad
  - Se configura con el atributo *timeout* de la sección de configuración *sessionState*

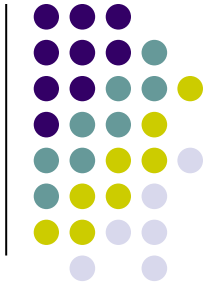


Establecer el idioma y la referencia cultural

- Globalización
  - Proceso de diseño y desarrollo de aplicaciones que pueden funcionar con diversas referencias culturales
- Localización
  - Proceso por el cual se personaliza una aplicación para una referencia cultural y configuración regional determinadas. Consiste, principalmente, en la traducción de la interfaz de usuario
  - La tecnología ASP.NET permite establecer dos valores de referencia cultural:
    - Propiedad **UICulture.** Su valor determina el idioma de la interfaz de usuario, es decir, qué recursos de la interfaz de usuario se cargan para la página
    - Propiedad **Culture.** Su valor determina los resultados de las funciones que dependen de la referencia cultural: formato de fecha, número y moneda, etc. Evita que se identifique un símbolo de moneda con un idioma. Ejemplos:

      El valor “en-US” de Culture define inglés en Estados Unidos, el valor “en-GB” define inglés en Gran Bretaña y el valor “es-MX” define español en Mexico, etc.

  Ambas propiedades se definen mediante valores de cadena estándar



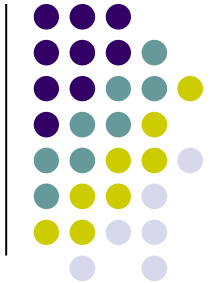
Establecer el idioma y la referencia cultural

- No es necesario que Culture y UICulture tengan el mismo valor
  - En un sitio de subastas Web, la propiedad UICulture podría cambiar para cada navegador Web (usuario), mientras que Culture se podría mantener constante para que los precios siempre se mostrarían en la misma moneda y formato
- Se puede establecer el idioma de la interfaz de usuario y la referencia cultural a nivel de aplicación o a nivel de página:
  - **De todas las páginas de una aplicación Web.** Se establecen los atributos uiculture y culture en la sección globalization del archivo de configuración *Web.config*:

```
<globalization uiculture="es" culture="es-MX" />
```

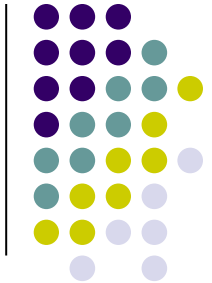
  - **De una página concreta de ASP.NET.** Se establecen los atributos Culture y UICulture de la directiva @ Page de la página de ASP.NET:

```
<%@ Page UICulture="es" Culture="es-MX" %>
```



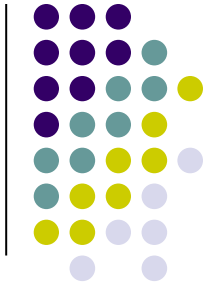
Concepto de accesibilidad Web

- A algunas personas les resulta difícil examinar la Web
  - Dificultades: visuales, manejo del ratón, leer información o entender cómo navegar por un sitio Web complejo, etc. Un caso típico es el de las personas mayores que desarrollan una combinación de estas discapacidades
- Algunas discapacidades solo se pueden superar empleando las denominadas, tecnologías de asistencia
  - Por ejemplo: software de lector de pantalla para personas ciegas
- Actualmente, las normas de accesibilidad Web están pensadas para contribuir a que los sitios Web **sean más entendibles y fáciles de usar para todas las personas desde cualquier tipo de dispositivo**
  - Beneficia a todos los usuarios, no solo a los usuarios con discapacidad
- Las normas de accesibilidad Web más utilizadas son las **directrices WCAG *(Web Content Accessibility Guidelines)*** redactadas por W3C



Características de accesibilidad en ASP.NET

- Los controles de ASP.NET están diseñados para que sean accesibles de forma predeterminada
  - Los controles de servidor de ASP.NET generan automáticamente código de marcado HTML que cumple con las pautas de accesibilidad
  - Los controles de servidor de ASP.NET basados en plantillas podrían generar código HTML que podría no cumplir las normas de accesibilidad
    - Debe configurarse manualmente el código de marcado en las plantillas para que el código HTML generado cumpla las pautas de accesibilidad
- La documentación de ASP.NET proporciona información sobre:
  - Las consideraciones de accesibilidad relativas a cada control
  - Técnicas para crear páginas Web accesibles que cumplan cada una de las directrices WCAG



Concepto

- **Patrón de software**
  - Es un esqueleto de soluciones software a problemas comunes en el desarrollo de software
    - Proporciona una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.
    - Constituye una forma de documentar las mejores prácticas
    - Facilita la creación de algunos tipos de aplicaciones Web
  - Existen varios tipos de patrones de software
    - Los patrones de software pueden ser utilizados en tareas de análisis, diseño y construcción de una aplicación Web
- Para que una solución software sea considerada como patrón
  - Debe haberse comprobado su **efectividad** resolviendo problemas similares
  - Debe ser **reutilizable** para poder aplicarlo a contextos similares en distintas circunstancias
  - Debe disponer de una **documentación** que facilite su comprensión y aplicabilidad
- Existe diversos sitios Web de patrones de ASP.NET: https://www.codeproject.com