

TÍTULO

# **DESARROLLO DE APLICACIONES I**

**Sesión Nro. 10**



## Objetivos de la sesión:

Al culminar la presente podrás:

- Los formularios MasterPage
- Pase de datos entre formularios: Variables de Sesión y el método QueryString
- Envío de archivos al servidor con el control FileUpload
- Conocer controles que permitan la validación de datos
- Incorporar controles AJAX Toolkit a la aplicación



Visual  
Studio



# Temario

1. Los formularios MasterPager
2. Pase de datos entre formularios
3. Envío de archivos al servidor mediante el control FileUpload
4. Los controles de Validación de ASP .Net
5. Empleo de AJAX en los formularios WEB



# Tema Nro. 1: Los Formularios MasterPage

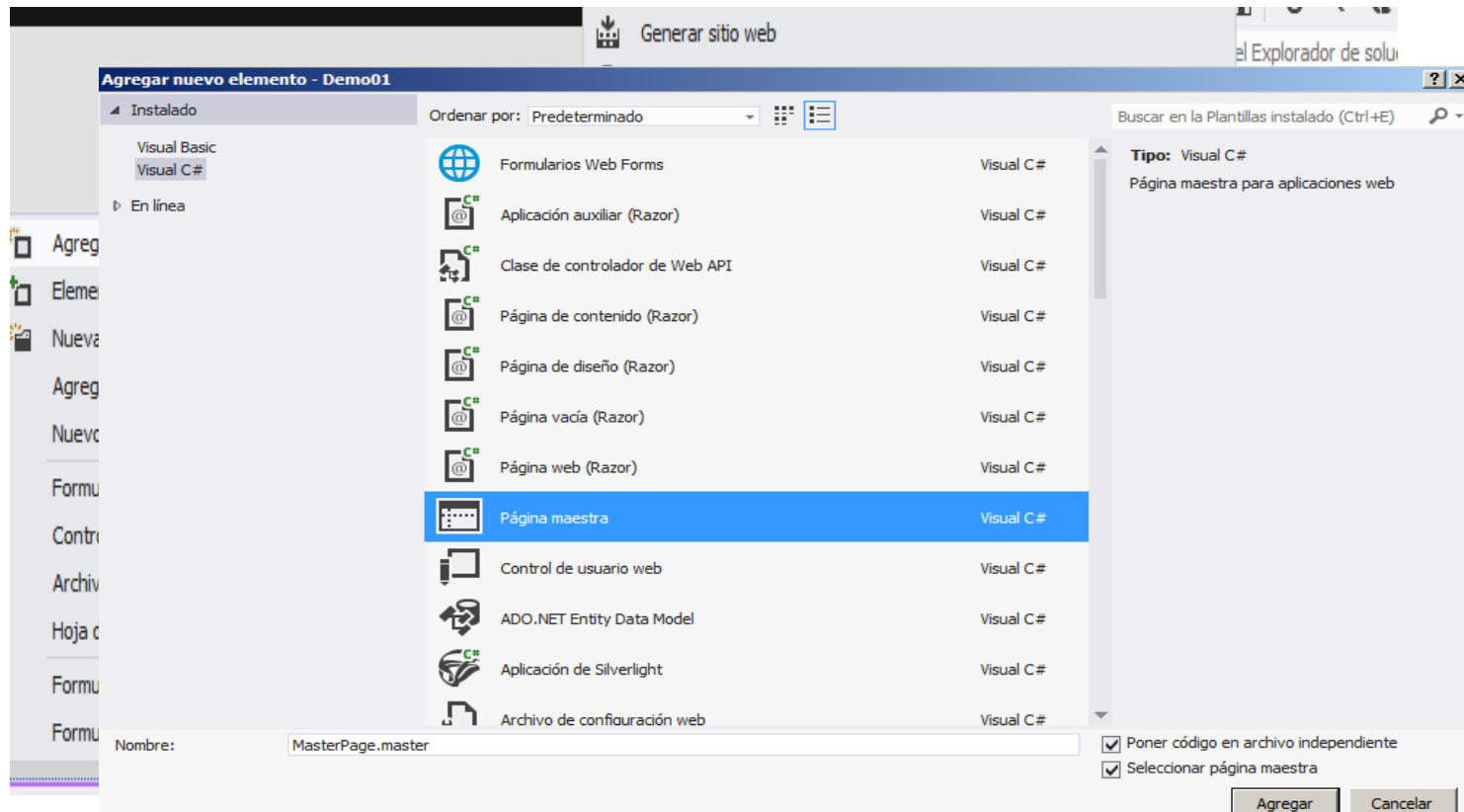


## 1.1 Definición

- Un formulario de Pagina Maestra o MasterPage es un formulario WEB que servirá de plantilla para los demás formularios WEB de la aplicación.
- Podemos diseñar un MasterPage de tal forma que su presentación y funcionalidad puedan heredarse a los demás formularios, ganando tiempo y estandarizando la presentación del producto final



## 1.2 Agregando un formulario MasterPage



## Tema Nro. 2: Pasando datos entre formularios



## 2.1 QueryString

- Podemos pasar datos de un WEB Form a otro mediante la URL

### Envio:

```
protected void btnEnviar_Click(object sender, EventArgs e)
{
    String strURL = "Destino.aspx?CP1=" + TextBox1.Text + "&CP2=" +
    TextBox2.Text;
    Response.Redirect(strURL);
}
```

### Recepción:

```
protected void Page_Load(object sender, EventArgs e)
{
    //Se obtiene parámetros por la URL (Botón enviar)
    Label1.Text = Request.QueryString["CP1"];
    Label2.Text = Request.QueryString["CP2"];
}
```





## 2.2 Variables Session

- Podemos pasar datos de un WEB Form a otro mediante variables de sesión

### Envio:

```
protected void btnEnviar_Click(object sender, EventArgs e)
{
    Session["Usuario"] = TextBox1.Text;
    Session["Password"] = TextBox2.Text;
    Response.Redirect("Destino.aspx");
}
```

### Recepción:

```
protected void Page_Load(object sender, EventArgs e)
{
    //Se obtiene parámetros del Objeto Session
    Label1.Text = Session["Usuario"].ToString ();
    Label2.Text = Session["Password"].ToString();
}
```



## 2.3 Variables Session y Application

- Puede utilizar los objetos **Application** y **Session** para almacenar los valores que son globales en lugar de específicos de páginas, bien para un usuario concreto (el objeto **Session**) bien para todos los usuarios (el objeto **Application**).
- Las variables **Session** y **Application** se almacenan en el servidor. A continuación, los examinadores de cliente se unen a la sesión mediante una cookie. Como resultado, el cliente debe disponer de cookies activadas en el explorador para que las variables **Session** y **Application** funcionen.
- Ejemplos:
  - `Session["UsuarioActual"]="José León"`
  - `Application["TipoCambio"]=3.68`



## LABORATORIO

**Realizar junto a su instructor el laboratorio en referencia al pase de valores entre formularios WEB**



## **Tema Nro. 3: Enviando archivos al servidor . El control FileUpload**



## 3.1 El control FileUpload

- Podemos enviar archivos desde el cliente al servidor mediante el control FileUpload .
- Como vemos en muchas aplicaciones WEB es posible que se envíen archivos de diferentes tipos, como por ejemplo:
  - Documentos Word
  - Libros Excel
  - Imágenes (Fotos)
- Recuerde que las aplicaciones empresariales no tienen el mismo objetivo que las redes sociales, donde los usuarios pueden postear videos, audios o cualquier otro tipo de material multimedia, por lo que hay que validar que tipos de archivos estamos enviando al servidor WEB.



## 3.2 El control FileUpload

### CONTROL FILEUPLOAD

#### Ejemplo de FileUpload

<input type="text"/>	<input type="button" value="Browse..."/>	<input type="button" value="Subir"/>
<div>[lblMensaje1]</div>		

#### Ejemplo2 de FileUpload: Validando extension

<input type="text"/>	<input type="button" value="Browse..."/>	<input type="button" value="Subir"/>
<div>[lblMensaje2]</div>		

#### Ejemplo 3 de FileUpload: Subiendo imagenes con restriccion de tamaño

<input type="text"/>	<input type="button" value="Browse..."/>	<input type="button" value="Subir"/>
----------------------	--	--------------------------------------



## LABORATORIO

**Realizar junto a su instructor el laboratorio en referencia al envío de archivos al servidor empleando el control FileUpload**



## Tema Nro. 4: Controles de Validación ASP. Net





## 4.1 Los controles de validación ASP .Net

- ASP. Net nos abastece de controles que permiten implementar validaciones diversas
- Estos controles son:
  - RequeryFieldValidator
  - RangeValidator
  - CompareValidator
  - CustomValidator
  - RegularExpressionValidator
  - ValidatorSummary
- Revisaremos algunos casos de aplicacion



## 4.2 RequyFieldValidator

- Permite establecer la obligatoriedad de un campo dentro del WEB Form
- Su propiedad básica es ControlToValidate, a la cual se le debe asignar el nombre del control que va a validar que no este en blanco al momento de hacer un envío

**EJEMPLOS DE REQUIREDFIELDVALIDATOR**

Nombre	<input type="text"/>	<span style="color: red;">Ingresar Nombre</span>
Apellido	<input type="text"/>	<span style="color: red;">Ingresar Apellido</span>
Direccion	<input type="text"/>	<span style="color: red;">Ingresar Direccion</span>
Telefono	<input type="text"/>	<span style="color: red;">Ingresar Telefono</span>
E_mail	<input type="text"/>	

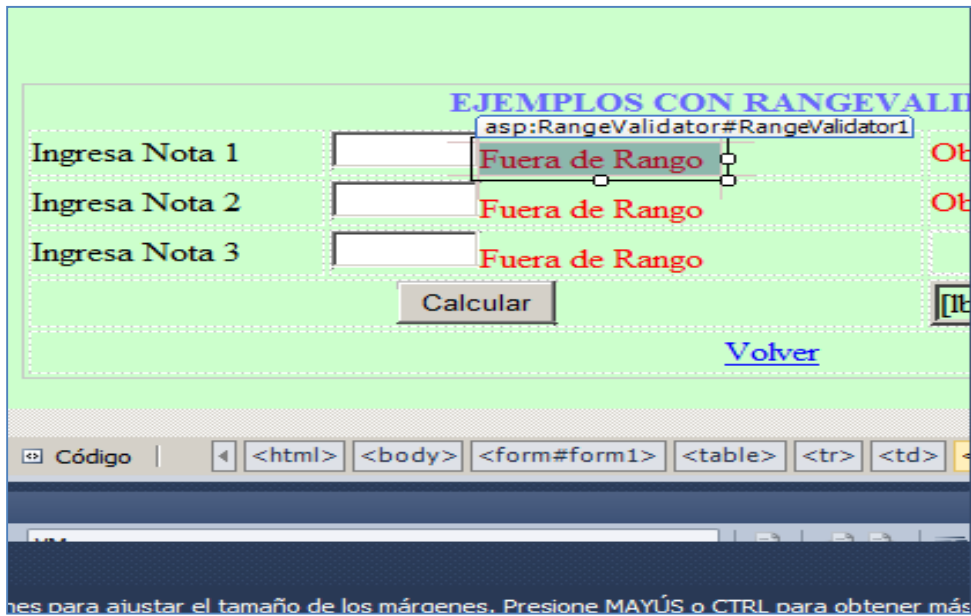
asp:requiredfield...#RequiredField...

AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToValidate	<b>TextBox1</b>
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	<b>Ingresar Nombre</b>
Font	



## 4.3 RangeValidator

- Permite validar un campo dentro de un rango
- Sus propiedades básicas son:
- ControlToValidate: Nombre del control a validar
- MaximumValue y MinimumValue: Los valores máximo y mínimo que el control va a soportar
- Type: Para definir que tipo de validación se aplicara.



BorderWidth	
ControlToValidate	<b>txtN1</b>
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
<b>ErrorMessage</b>	<b>Fuera de Rango</b>
Font	
ForeColor	Red
Height	
MaximumValue	<b>20</b>
MinimumValue	<b>0</b>
SetFocusOnError	<b>True</b>
SkinID	
TabIndex	0
Text	
ToolTip	
Type	<b>Integer</b>

## 4.4 CompareValidator

- Permite validar un campo comparándolo con un valor o con otro control
- Sus propiedades básicas son:
- **ControlToValidate**: Nombre del control a validar
- **CompareToValidate**: El nombre del control sobre el cual se efectuara la comparación
- **ValueToCompare**: Si desea puede hacer la comparación sobre un valor determinado en esta propiedad.
- **Operator**: Que tipo de operador de relación se aplicara en la comparación
- **Type**: Para definir que tipo de validación se aplicara.



## 4.5 CompareValidator

### Ejemplo de CompareValidator

Valor 1

Valor 2

asp:CompareValidator#CompareValida...

El valor debe ser menor en el textbox2

El textbox1 debe ser mayor que 100

Calcular

[lblResto]

[Volver](#)

Design Split Source <asp:Content#Content2> <body> <div> <asp:CompareValidator#Com...>

Error List

0 Errors 3 Warnings 0 Messages

Search Error List

Solution Explorer

- favicon.ico
- Global.asax
- packages.config
- Site.master
- Test1.aspx
- Test2.aspx
- Test3.aspx

Model Browser

Properties

**CompareValidator1** System.Web.UI.WebControls.CompareValidator

BorderWidth	
ClientIDMode	Inherit
ControlToCompare	TextBox1
ControlToValidate	TextBox2
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	El valor debe ser menor en el textbox2
Font	



## 4.6 CustomValidator

- Permite implementar una regla de validación personalizada la cual se aplica en el evento ServerValidate del control.
- Sus propiedad básica es :
  - ControlToValidate: Nombre del control a validar
- Veamos un ejemplo del código que implementa una validación con este control



## 4.6 CustomValidator

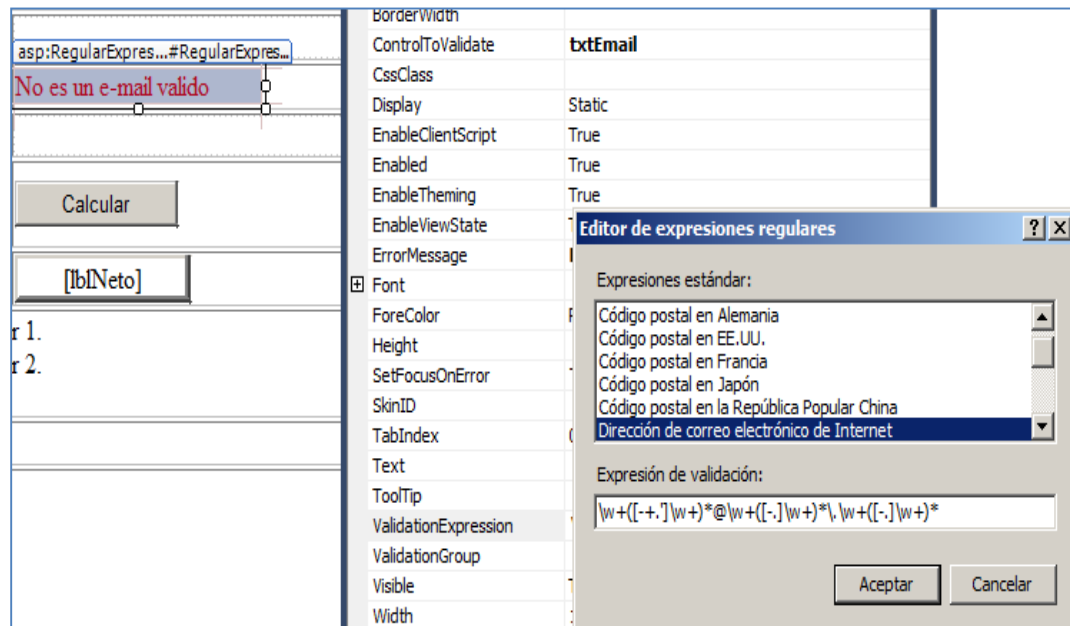
```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (txtCod.Text == String.Empty | txtCod.Text.StartsWith("L") | txtCod.Text.StartsWith("E") )
        // Si el valor ingresado en la caja de texto txtCodigo esta vacio o empieza con "L" o "E"
        // se da como valido el ingreso
    {
        args.IsValid = true;
    }
    else
        // de lo contrario se invalida
    {
        args.IsValid = false;
    }
}

protected void LinkButton1_Click(object sender, EventArgs e)
{
    if (Page.IsValid )
        // Si la pagina es valida
    {
        lblMensaje.Text = "Código grabado!!";
    }
    else
        // de lo contrario
    {
        lblMensaje.Text = "No se grabo la informacion";
    }
}
```



## 4.7 RegularExpressionValidator

- Permite implementar validaciones en base a expresiones regulares o de formato preestablecido (por ejemplo un correo electrónico)
- Sus propiedades básicas son :
  - ControlToValidate: Nombre del control a validar
  - ValidationExpression: Que permite establecer la expresión a emplear para la validación





## 4.8 ValidationSummary

- Permite hacer un sumario de las validaciones que no se han cumplido
- Sus propiedades basicas son :
  - DisplayMode: Indica el modo de visualización de los mensajes del sumario
  - ShowMessageBox: Que permite indicar si se mostraran o no los mensajes de error de cada validación errada.
  - ShowSummary:: Que permite indicar si se mostraran o no un resumen o sumario de las validaciones erradas..



## Recuerde....

- Si ha trabajado en un sitio WEB vacío, y ha empleado controles de validación debe agregar a su archivo de configuración las siguientes líneas:

```
---  
Para obtener más información sobre cómo configurar la aplicación ASP.NET, visite  
https://go.microsoft.com/fwlink/?LinkId=169433  
-->  
<configuration>  
  <appSettings>  
    <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />  
  </appSettings>  
</configuration>  
<system.web>  
  <compilation debug="true" targetFramework="4.6.1"/>  
  <httpRuntime targetFramework="4.6.1"/>  
</system.web>  
</httpRuntime>
```



## LABORATORIO

**Realizar junto a su instructor el laboratorio en referencia al empleo de controles de validación en ASP Net**

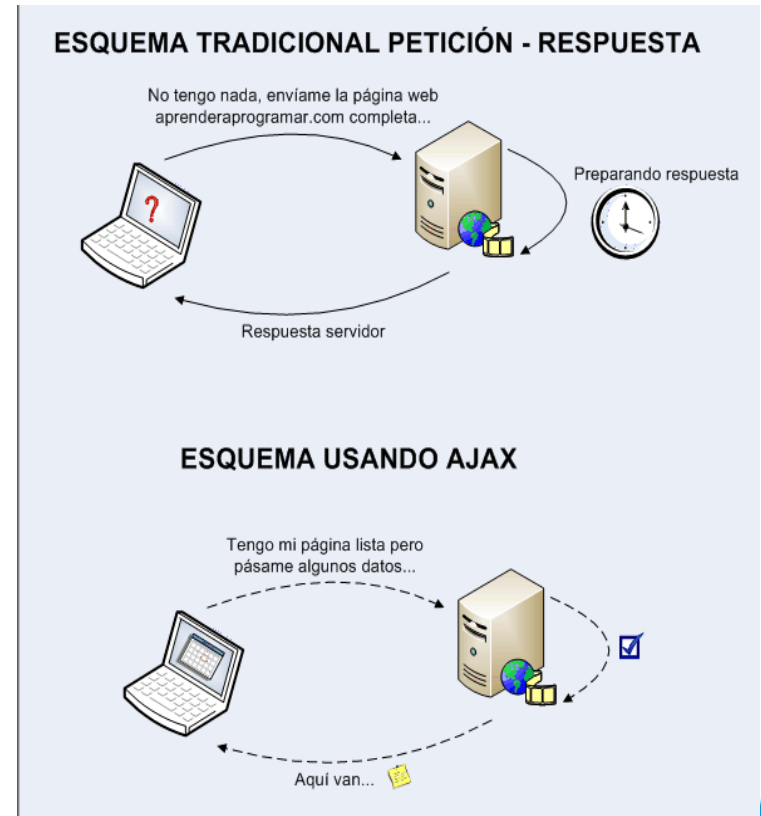


## **Tema Nro. 5: Empleando controles AJAX para el ingreso eficiente de los datos**



# Definición de AJAX

- **AJAX** es el acrónimo de *Asynchronous Javascript and XML*, es decir, **Javascript y XML Asíncrono**.
- El principal objetivo del AJAX, es intercambiar información entre el servidor y el cliente (navegadores) sin la necesidad de recargar la página. De esta forma, ganamos en usabilidad, experiencia y productividad del usuario final.



## Ventajas de AJAX

- Rapidez en las operaciones.
- Menos carga del servidor (menos transferencia de datos cliente/servidor).
- Menos ancho de banda.
- Soportada por la mayoría de navegadores.
- Interactividad (El usuario no tiene que esperar hasta que lleguen los datos del servidor).
- Portabilidad
- Usabilidad
- Velocidad (Debido a que no hay que recargar la página nuevamente)



## Desventajas de AJAX

- Se pierde el concepto de “volver a la página anterior”.
- Problemas con navegadores antiguos.
- No funciona si el usuario tiene desactivado el Javascript en su navegador.
- Se requieren conocimiento sobre las tecnologías que forman AJAX.
- Problemas SEO, los buscadores no indexan la información recibida vía AJAX.





## Agregando la librería de AJAX Toolkit

- Si desea agregar los controles de AJAX Toolkit lo debe hacer mediante la opción de Administrar Paquetes Nuget, haciendo referencia al paquete mostrado a continuación e instalándolo:

Administrador de paquetes NuGet: ClienteWEB

Examinar Instalado Actualizaciones 1

AjaxControlToolkit x ☐ Incluir versión preliminar

Origen del paquete: nuget.org

**AjaxControlToolkit** por CodePlex Foundation, 7.48M descargas 20.1.0  
Build AJAX applications in Web Forms using more than 40 controls in AjaxControlToolkit.

**AjaxControlToolkit.HtmlEditor.Sanitizer** por CodePlex Foundation, 212K descargas 20.1.0  
This package provides HTML sanitization (to prevent XSS attacks) for the AjaxControlToolkit HtmlEditor extender.

**AjaxControlToolkit** nuget.org

Versión: Versión estable más reciente 20.1.0 **Instalar**

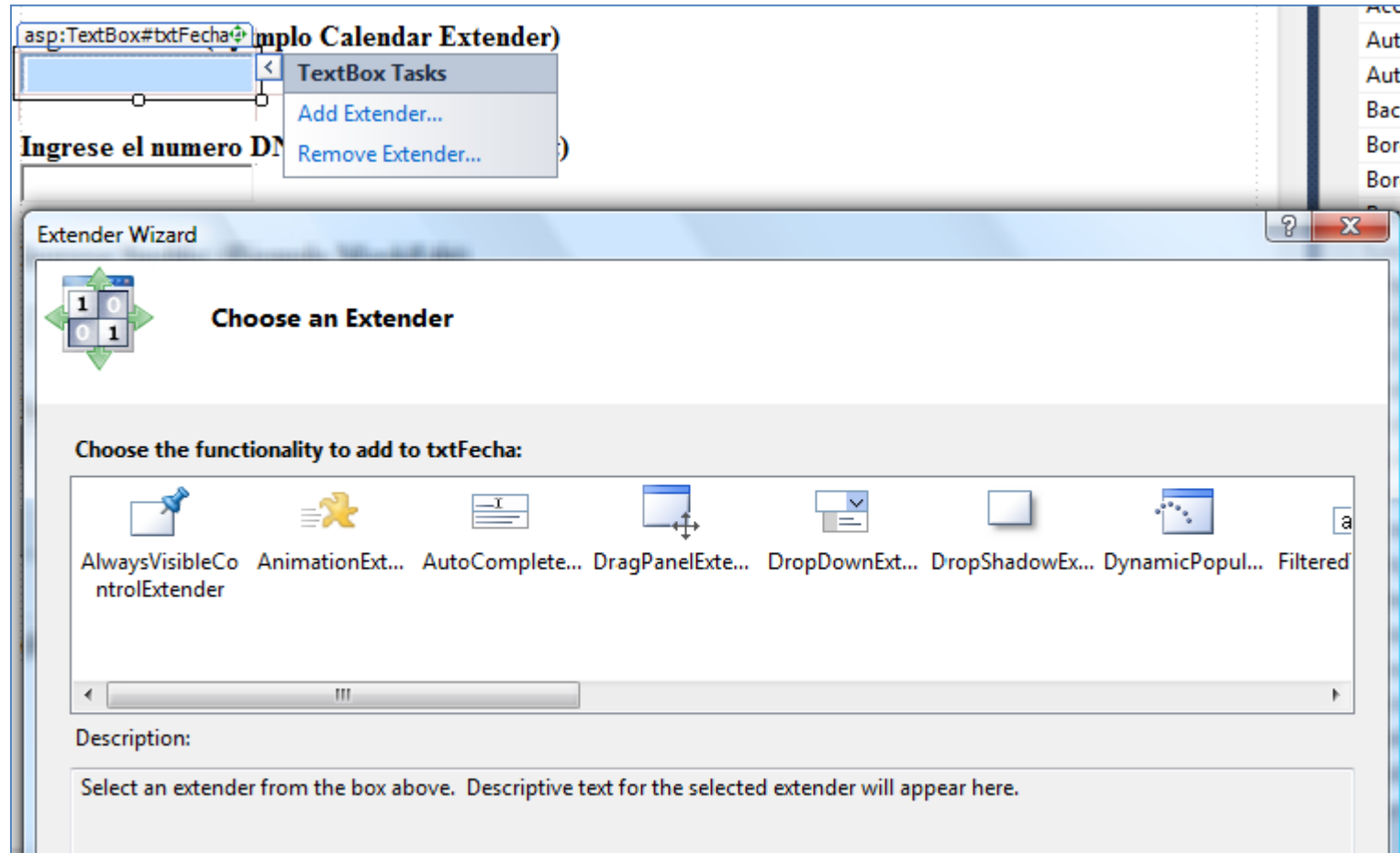
Opciones

Descripción

- Tras la instalación compile su proyecto para poder disponer en su cuadro de herramientas los controles de AJAX Toolkit. Si a pesar que compile no puede visualizarlos en su cuadro de herramientas, pruebe grabando su proyecto, cerrarlo y volviéndolo abrir.



# Agregando un extensor AJAX



# Empleando controles AJAX Toolkit para optimizar el ingreso de datos

**Ingreso fecha: (Ejemplo Calendar Extender)**


August, 2012						
Su	Mo	Tu	We	Th	Fr	Sa
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

**Ingreso Sueldo: (Ejemplo MaskEdit)**

S/. \_.\_|\_

**Ingreso correo electronico: (Ejemplo ValidationCalloutExtender)**

Enviar



Escriba su correo electronico

**Ingreso correo electronico: (Ejemplo ValidationCalloutExtender)**

pjleon69@gmail.com

Enviar

**(Ejemplo mensaje de confirmación)**

Mensaje de página web

?

Seguro de enviar los datos??

Aceptar Cancelar



## LABORATORIO

**Realizar junto a su instructor el laboratorio en referencia al empleo de AJAX en aplicaciones WEB ASP Net**



## Conclusiones de la sesión:

- Entender los formularios MasterPage
- Comprendiste como se pueden pasar datos entre formularios
- Validación de envío de archivos al servidor
- Es importante la funcionalidad de validación de los datos desde formularios WEB.
- El hecho de incluir AJAX en nuestras aplicaciones garantiza una enorme mejora en la capa de presentación de los Web Form

