Departamento de Lenguajes y Sistemas Informáticos

Tema 12. Aplicaciones Web (IV)

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escuela Politécnica Superior
Universidad de Alicante

Indice

1. Cookies
2. Envío de email
3. Controles de usuario
4. Introducción a AJAX en .NET

Cookies

1

Cookies

- Para almacenar datos relativos a un usuario se puede utilizar el objeto Session
- Problema:
  - Los datos se borran cuando el usuario cierra la ventana del navegador
- Solución:
  - Para almacenar datos y que éstos se preserven es necesario utilizar las cookies

Cookies

- Las cookies son extractos de datos que una aplicación ASP.NET puede almacenar en el navegador del cliente para su posterior recuperación
- Las cookies no se pierden cuando se cierra el navegador (a no ser que el usuario las borre)

Cookies en ASP.NET

- Una cookie se representa por la clase HttpCookie
- Las cookies del usuario se leen a través de la propiedad Cookies del objeto Request
- Las cookies del usuario se modifican a través de la propiedad Cookies del objeto Response
- Por defecto las cookies expiran cuando se cierra el navegador
  - Se pueden alterar los puntos de expiración (poner una fecha determinada de expiración)

Cookies en ASP.NET

Página default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie userCookie;
    userCookie = Request.Cookies["UserID"];
    if (userCookie == null)
    {
        Label1.Text ="No existe la cookie, creando cookie ahora";
        userCookie = new HttpCookie("UserID", "Ana López");
        userCookie.Expires = DateTime.Now.AddMonths(1);
        Response.Cookies.Add(userCookie);
    }
    else
    {
        Label1.Text = "Bienvenida otra vez, " + userCookie.Value;
    }
}
```

Cookies en ASP.NET

- La variable userCookie se inicializa como una instancia de la clase HttpCookie y se le asigna el valor de la cookie llamada UserID
- Se comprueba la existencia de la cookie
  - Caso de no existir se muestra un mensaje
  - Se le da el valor “Ana López”
  - Se le asigna una fecha de expiración
- La cookie se transfiere al navegador usando el método Response.Cookies.Add
- Si la cookie existe se muestra un mensaje de bienvenida

Cookies en ASP.NET

- La primera vez que se carga la página se mostrará el mensaje
  - No existe la cookie, creando cookie ahora
- Si se recarga de nuevo la página, la cookie ya existirá y se mostrará el mensaje
  - Bienvenida otra vez, Ana López
- Cuidado!
  - Ten en cuenta que los usuarios pueden rechazar las cookies
  - No puedes dejar que recaigan en ellas aspectos importantes de la aplicación

Cookies en ASP.NET

- Borrar cookies
  - La única forma es reemplazarla por una cookie con una fecha de expiración que ya ha pasado

```
HttpCookie userCookie= new HttpCookie("UserID");
userCookie.Expires=DateTime.Now.AddDays(-1);
Response.Cookies.Add(userCookie);
```

2 Email

Email

- Supongamos que tenemos una tienda online y queremos enviar un email de confirmación de pedido a cada cliente

- En lugar de escribir manualmente cada email ASP.NET permite automatizar este proceso

Email

- System.Net.Mail
  - **SmtpClient**
  - **MailMessage**
  - **Attachment**
  - **AttachmentCollection**
  - **MailAddress**
  - **MailAddressCollection**

Email

- **MailMessage**
  - **From**
  - **To**
  - **CC**
  - **Bcc**
  - **Attachments**
  - **Subject**
  - **Body**
  - **IsBodyHTML**

Email

```
SmtpClient smtpClient = new SmtpClient("smtp.gmail.com",587);
MailMessage message = new MailMessage();
try
{
   MailAddress fromAddress = new MailAddress(”irene@dlsi.ua.es", “Alias remitente");
   MailAddress toAddress = new MailAddress(“correo@gmail.com", “Alias destinatario");
   message.Attachments.Add(new Attachment("C:\\imagen1.gif"));
   message.Attachments.Add(new Attachment("C:\\imagen2.jpg"));
   message.From = fromAddress;
   message.To.Add(toAddress);
   message.Subject = “Probando el envío!";
   message.Body = “Este es el cuerpo del mensaje";
   smtpClient.EnableSsl = true;

   smtpClient.Credentials = new System.Net.NetworkCredential(“usuario", “password");
   smtpClient.Send(message);
   Label1.Text = “Mensaje enviado.";
}
catch (Exception ex)
{
   Label1.Text = “No se pudo enviar el mensaje!";
}
```

3

Controles de usuario

Qué son?

- Se pueden crear controles personalizados y reutilizables en diferentes páginas : CONTROLES DE USUARIO

- Son archivos con extensión .ascx que incluiremos en nuestras páginas web aspx. Los cambios realizados en el ascx serán reflejados allá donde lo hayamos ubicado.

- Contienen uno o varios controles ASP.NET junto con el código necesario para que los controles realicen la funcionalidad deseada.

Diferencias con webforms

- 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.
- Los controles de usuario no se pueden ejecutar como archivos independientes. En su lugar, debe agregarlos a las páginas ASP.NET, como haría con cualquier otro control.
- El control de usuario no contiene elementos **html**, **body** o **form**. Estos elementos deben estar en la página de alojamiento.

Creación

- Agregar un nuevo elemento al proyecto, de tipo user control
- Una vez creado, trabajamos como si de una página .aspx normal se tratara
- Para añadir el control a cualquier sitio de nuestra Web, no tenemos más que arrastrarlo al lugar que queramos..

Incluir un control de usuario en una página Web ASP.NET

- Se puede hacer de dos maneras:

1. Manualmente añadimos la siguiente directiva en el aspx
   - **Atributo src:** define la ruta al control de usuario
   - **Atributo tagprefix**: permite asociar un prefijo al control de usuario
   - **Atributo Tagname**: asociamos un nombre al control de usuario
2. Arrastramos el control desde nuestro explorador de soluciones a la página donde queramos incluirlo y automáticamente se añadirá esta directiva.

Ejemplo

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="uc" TagName="Spinner"
   Src="~\Controls\Spinner.ascx" %>
<html>
<body>
<form runat="server">
   <uc:Spinner id="Spinner1"
      runat="server"
      MinValue="1"
      MaxValue="10" />
</form>
</body>
```

Links

- http://msdn.microsoft.com/es-es/library/26db8ysc(v=vs.80).aspx
- http://www.subgurim.net/Articulos/asp-net-general/121/como-hacer-un-control-de-usuario-ascx.aspx
- Ejemplo un buscador.
- http://csharp-experience.blogspot.com.es/2009/12/aspnet-eventos-de-controles-de-usuarios.html

4.1

Introducción a AJAX

http DESVENTAJAS

- Período demasiado largo al refrescar la página
- El usuario debe esperar a que la petición (request) finalice para seguir interactuando con la aplicación.

- SOLUCIÓN: Manejo de la petición de un modo asíncrono
- *Llamadas asíncronas que no bloquean la interfaz de usuario mientras se producen.*

AJAX= Asynchronous JavaScript and XML

- Cura para “enfermedades comunes” de las aplicaciones basadas en navegador
  - Usa el objeto **XmlHttpRequest** para recuperar datos del servidor de forma asíncrona y **JavaScript** para actualizar el contenido de la página.
  - Elimina el parpadeo y usa el ancho de banda de manera más eficiente.
- Se utiliza en Google Suggests, Google Maps, Microsoft Virtual Earth, Outlook Web Access… etc

- Tecnología de Microsoft para desarrollar aplicaciones Web basadas en AJAX

- Provee librerías script en el cliente que incorporan tecnologías **JavaScript** y HTML dinámico (**DHTML**), y las integra con la plataforma de desarrollo basada en **ASP.NET**

- Utilizando ASP.NET AJAX, se puede mejorar la experiencia de usuario y la eficiencia de las aplicaciones Web.

Arquitectura AJAX

- AJAX incluye librerías script en el cliente para facilitar las llamadas asíncronas al servidor

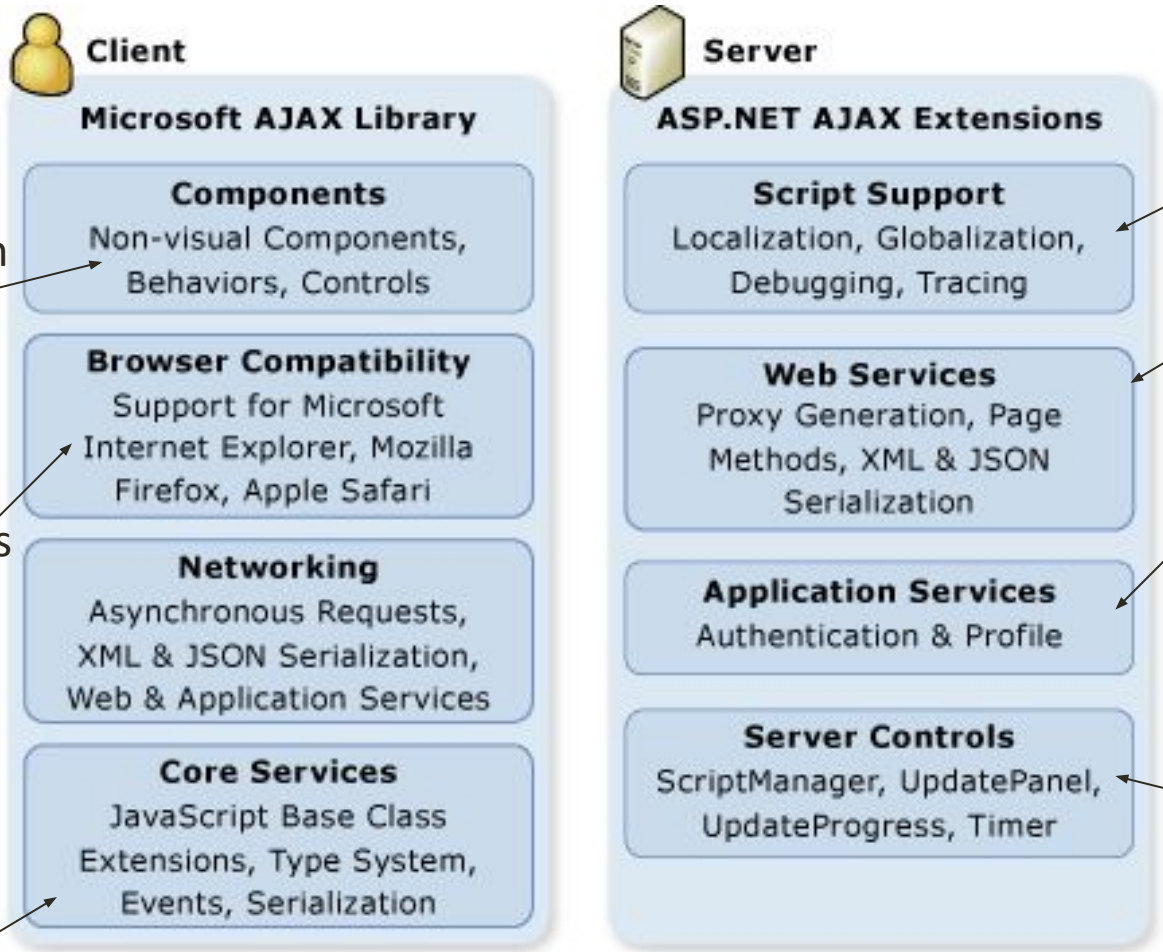
- También incluye componentes básicos en el servidor para soportar esas llamadas asíncronas del cliente

Arquitectura AJAX

Soporte para crear componentes en el cliente

compatibilidad con navegadores

componentes para manejar la IU y el flujo de una aplicación y gestionar eventos…



Soporte de scripts

Servicios Web

Servicios de aplicación

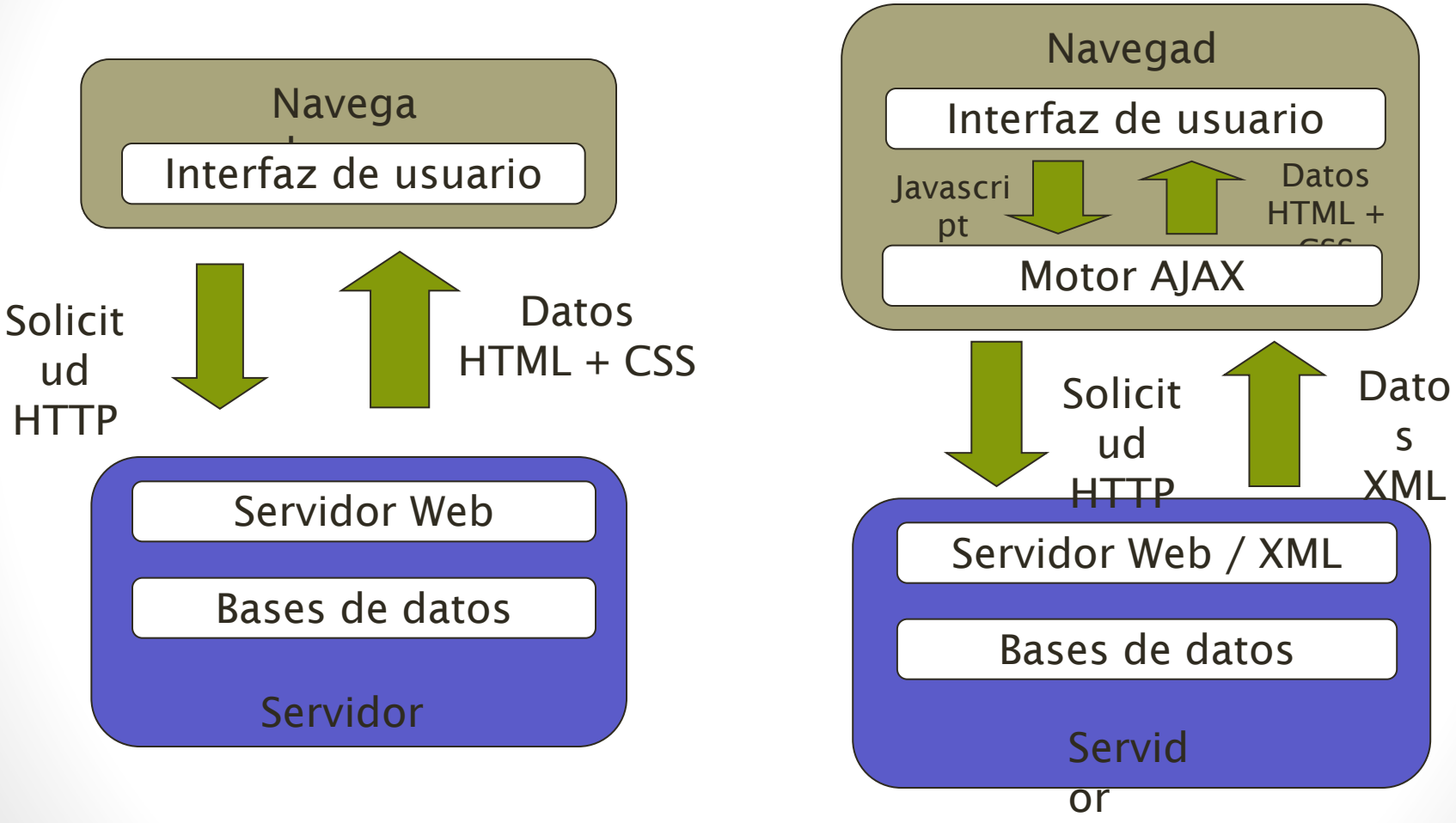
Controles de servidor

¿Por qué usar AJAX?

- Eficiencia mejorada ejecutando partes significantes de una página Web en el navegador.
- Elementos de IU familiares, típicas de una aplicación de escritorio, como ventanas pop-up, indicadores de progreso y tooltips.
- Actualizaciones parciales de la página que refrescan sólo las partes de la página Web que han cambiado.
- Soporte para los navegadores más populares y utilizados, como Microsoft Internet Explorer, Mozilla Firefox, y Apple Safari.

Funcionamiento de AJAX

Modelo Clásico de aplicaciones web vs AJAX



Una aplicación Ajax

- Se elimina la naturaleza *start-stop-start-stop* de la interacción de la Web introduciendo un intermediario — **un motor Ajax** — entre el usuario y el servidor.

- En lugar de cargar una página Web, al inicio de la sesión, el navegador carga el **motor Ajax** — escrito en JavaScript , usualmente en un marco oculto.

Motor Ajax

- Este motor es responsable de presentar la interfaz al usuario y de comunicar con el servidor de parte del usuario.
- Permite que la interacción del usuario con la aplicación sea **asíncrona**.
- **Cada acción del usuario** que normalmente generaría una petición HTTP se convierte en una **llamada JavaScript al motor Ajax**.

Motor Ajax (II)

- Cualquier respuesta a una acción del usuario que no requiera volver al servidor es manejada por el motor.
  - como validación de datos simple, editar datos en memoria, y incluso alguna navegación

- Si el motor necesita algo del servidor para elaborar la respuesta el motor hace estas peticiones de manera **asíncrona**, usualmente utilizando XML, sin paralizar la interacción del usuario con la aplicación.

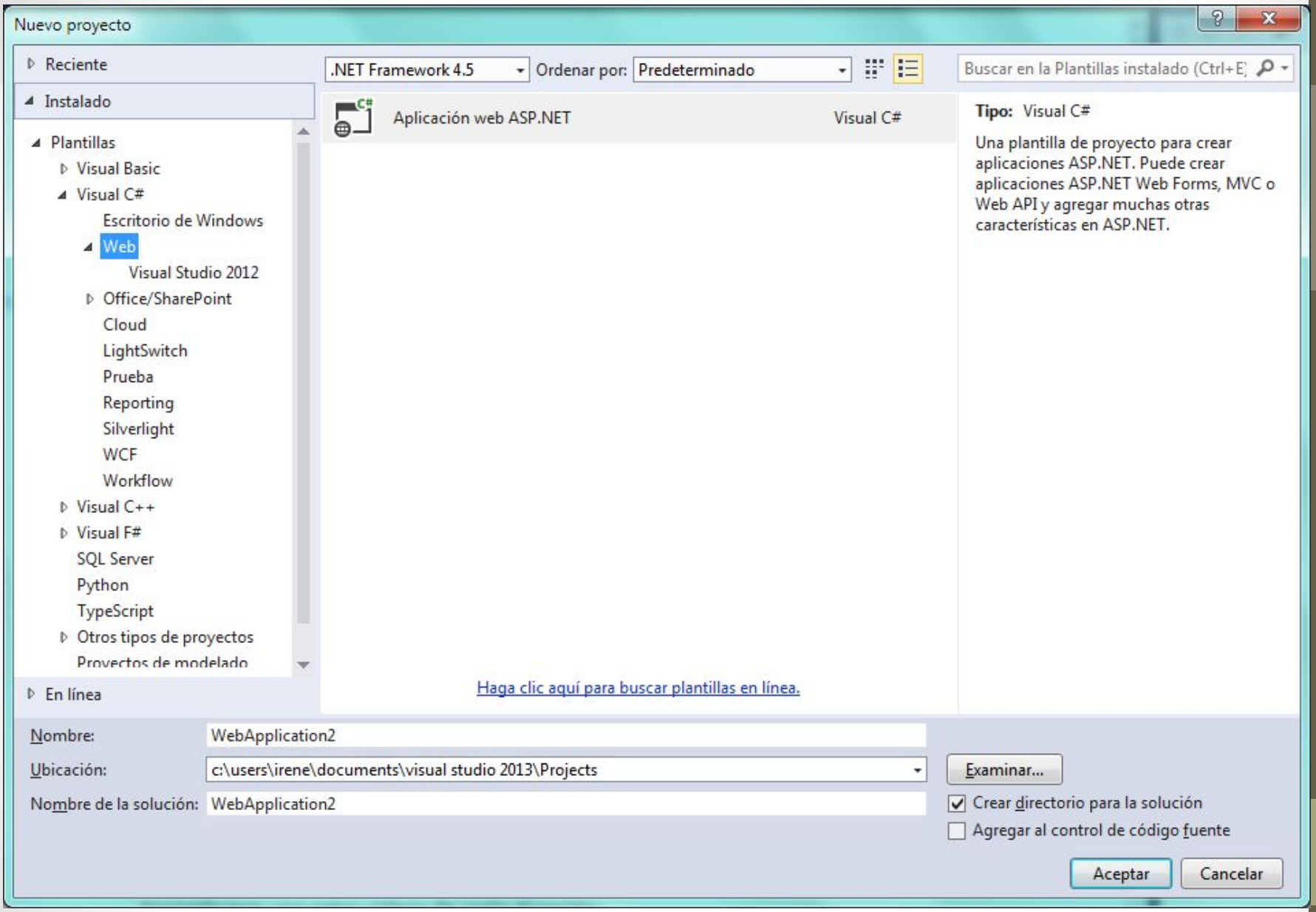
ASP.Net

4.2

Creación de una aplicación Web AJAX

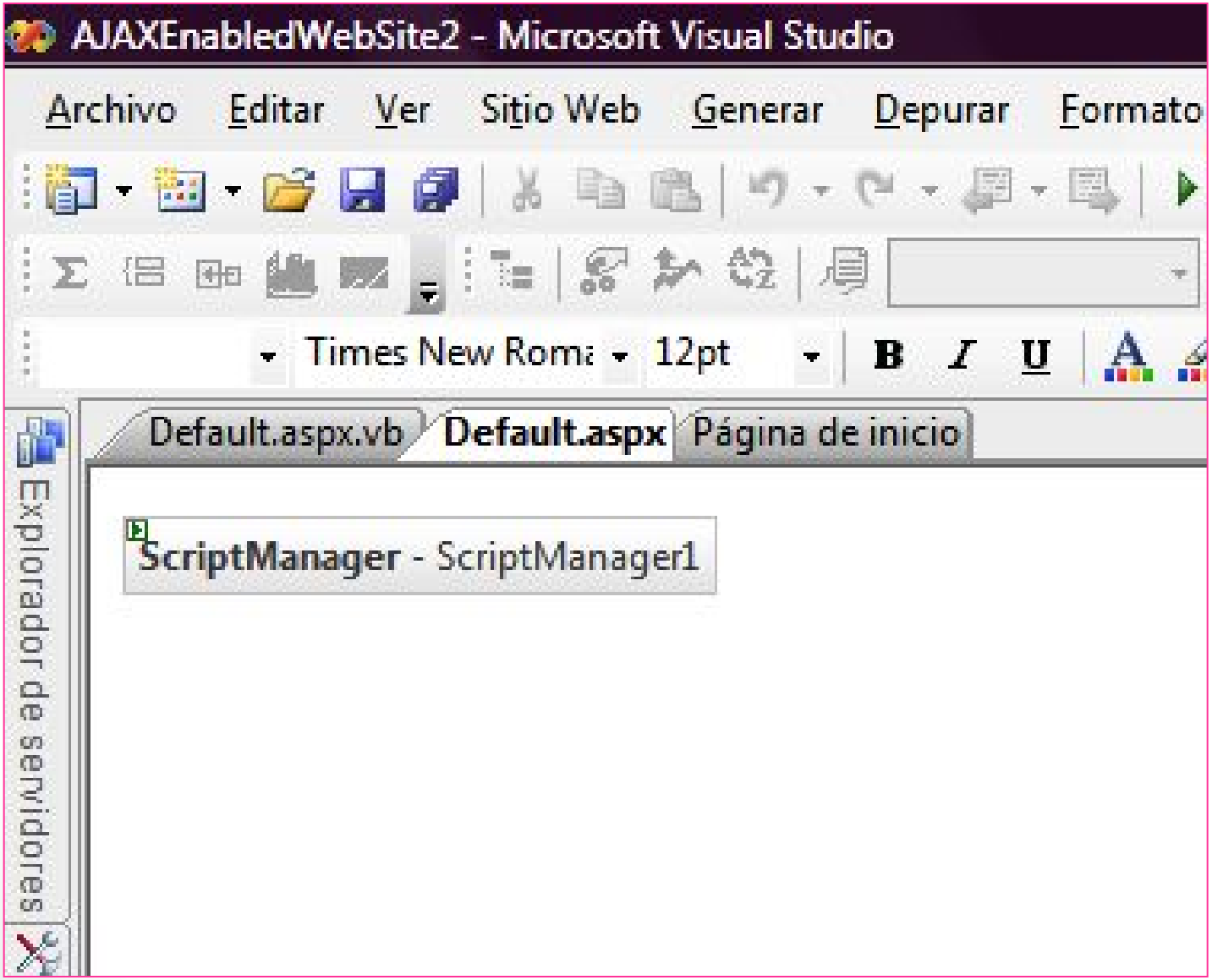
Controles de servidor ASP.net AJAX

- Consisten en código de cliente y de servidor que se integran para producir el comportamiento AJAX.
- **ScriptManager**
  - Registra el script de la librería de microsoft AJAX en la página. Esto permite al script de cliente admitir características como la representación parcial de páginas y las llamadas a servicios web. **El control ScriptManager es necesario para utilizar los demás controles.**


Nuevo proyecto
Reciente
Instalado
Plantillas
Visual Basic
Visual C#
Escritorio de Windows
Web
Visual Studio 2012
Office/SharePoint
Cloud
LightSwitch
Prueba
Reporting
Silverlight
WCF
Workflow
Visual C++
Visual F#
SQL Server
Python
TypeScript
Otros tipos de proyectos
Proyectos de modelado
En línea
.NET Framework 4.5
Ordenar por:
Predeterminado
Buscar en la Plantillas instalado (Ctrl+E)
Aplicación web ASP.NET
Visual C#
Tipo: Visual C#
Una plantilla de proyecto para crear aplicaciones ASP.NET. Puede crear aplicaciones ASP.NET Web Forms, MVC o Web API y agregar muchas otras características en ASP.NET.
Haga clic aquí para buscar plantillas en línea.
Nombre:
WebApplication2
Ubicación:
c:\users\irene\documents\visual studio 2013\Projects
Examinar...
Nombre de la solución:
WebApplication2
Crear directorio para la solución
Agregar al control de código fuente
Aceptar
Cancelar

Default.aspx

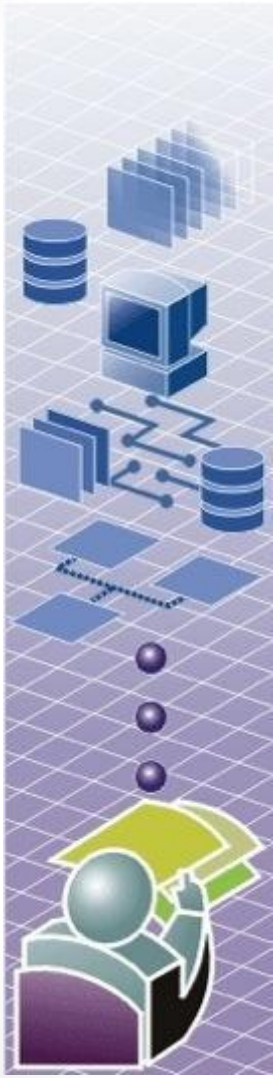
```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
        </div>
    </form>
</body>
</html>
```


AJAXEnabledWebSite2 - Microsoft Visual Studio
Archivo
Editar
Ver
Sitio Web
Generar
Depurar
Formato
Times New Roma
12pt
Default.aspx.vb
Default.aspx
Página de inicio
ScriptManager - ScriptManager1
Explorador de servidores

Controles AJAX

- UpdatePanel
  - Permite refrescar las partes seleccionadas de la página, en lugar de refrescar la página completa utilizando un postback síncrono.
- UpdateProgress
  - Provee información de estado sobre actualizaciones parciales de la página en controles **UpdatePanel.**
- Timer
  - Ejecuta postbacks en intervalos de tiempo definidos. Puede utilizarse para enviar la página completa, o usarse junto al control **UpdatePanel** para realizar actualizaciones parciales de la página en un intervalo definido.

Ejercicio



Ejercicio

- En un formulario Web vamos a crear dos listas desplegadas (listBox) a las que vamos a insertar elementos desde un cuadro de texto.
- La primera lista se actualizará SIN recarga de la página.
- La segunda lista se actualizará normalmente.

Añadir elemento

elemento 1

añadir sin recarga

Añadir con recarga

no se recarga la página...

elemento 1

se recarga la página...

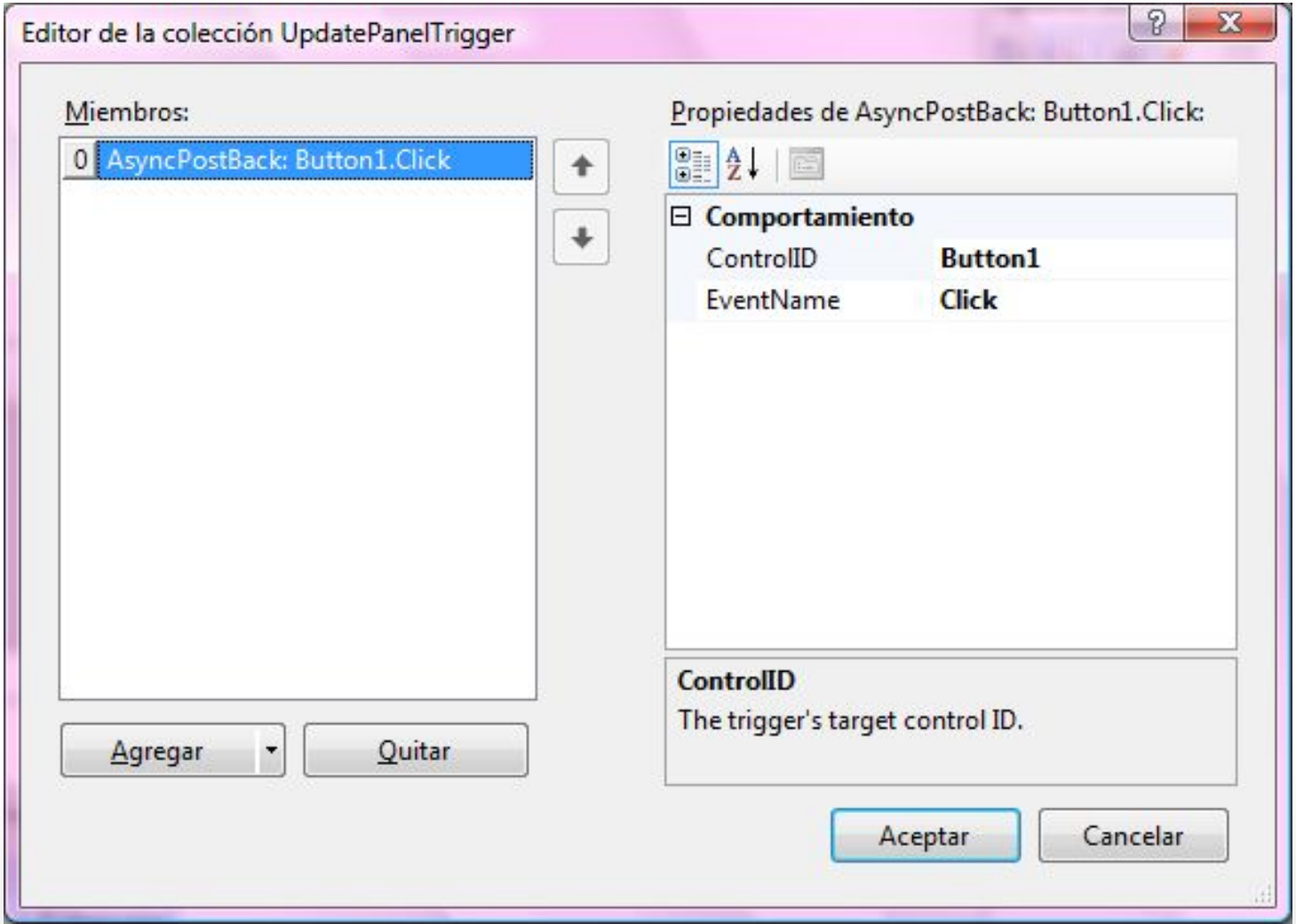
elemento 1

Código asociado

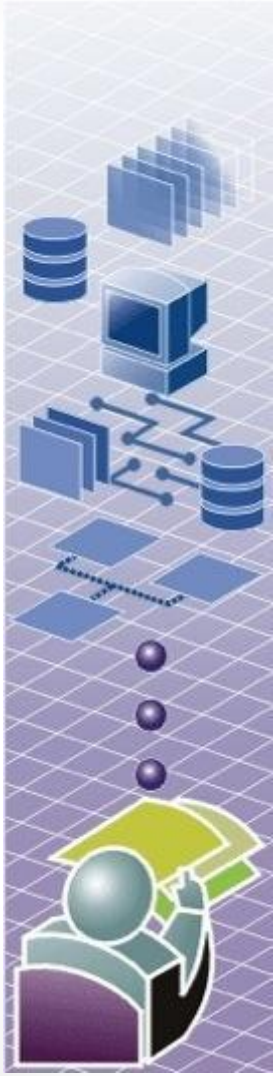
```
protected void Button1_Click(object sender, EventArgs e)
    {
       ListBox1.Items.Add(TextBox1.Text);
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
       ListBox2.Items.Add(TextBox1.Text);
    }
```

Update Panel

- Propiedad Triggers



Ejercicio



Ejercicio

- Vamos a añadir ahora un control UpdateProgress.
- En este control añadimos el texto “Actualizando…” y la imagen indicator.gif (cv).
- Vamos a añadir un retardo de 2 segundos para hacerlo realista.

Añadir elemento

elemento 2

Actualizando...

añadir sin recarga

Añadir con recarga

no se recarga la página...

elemento 1

se recarga la página...

elemento 1

Código a añadir…

```
protected void Button1_Click(object sender, EventArgs e)
   {
      ListBox1.Items.Add(TextBox1.Text);
      System.Threading.Thread.Sleep(2000);
   }
```