Departamento de Lenguajes y Sistemas Informáticos

Tema 13. Aplicaciones Web (IV)

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escuela Politécnica Superior
Universidad de Alicante

Indice

1. Cookies
2. Envío de email
3. Ficheros
4. Controles de usuario

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 Ficheros

Ficheros

- En ASP.NET existen un conjunto de clases para trabajar con ficheros
  - System.IO
- Nos permiten leer, escribir y actualizar el contenido de ficheros y directorios
- Tres tipos de clases para:
  - Trabajar con ficheros
  - Trabajar con streams
  - Trabajar con directorios

Ficheros

- Clases para trabajar con ficheros y streams:
  - File: Métodos para trabajar con ficheros
  - FileStream: Representa un stream para leer y escribir en ficheros
  - StreamReader: Lee caracteres de un fichero de texto
  - StreamWriter: Escribe caracteres en un fichero de texto
  - Path: Métodos para manipular un fichero o directorio

Ficheros

- Escribiendo en un archivo:

Página WriteFile.aspx

```
<asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" Rows="10"
   Width="220px"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Escribir"
   onclick="Button1_Click" />
```

Página WriteFile.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
 {
   using (StreamWriter streamwriter = File.CreateText(@"C:\Dpaa\Mitexto.txt"))
    {
            streamwriter.WriteLine(TextBox1.Text);
    }
 }
```

Ficheros

- El método CreateText sobreescribe el archivo cada vez que se invoca
- Alternativa:
  - AppendText: Agrega texto a continuación
- La directiva using permite liberar los recursos utilizados al terminar la escritura

Ficheros

- Leyendo de un archivo

```
string inputString;
TextBox1.Text = "";
using (StreamReader streamReader =
    File.OpenText(@"C:\Dpaa\Mitexto.txt"))
    {
       inputString = streamReader.ReadLine();
       while (inputString != null)
       {
          TextBox1.Text += inputString + "\n";
          inputString = streamReader.ReadLine();
       }
     }
```

Ficheros

- Subiendo archivos
  - Los usuarios pueden subir archivos al servidor
    - Fotos
    - Presentaciones
    - Vídeos
- En ASP.NET
  - Control FileUpload
  - Proporciona un TextBox y un botón para seleccionar los archivos del ordenador del usuario

Ficheros

- Propiedades de FileUpload:
  - HasFile: True si el usuario ha subido un archivo
  - FileName: El nombre del fichero como string
  - FileContent: Stream que puede ser utilizado para leer el contenido
  - FileBytes: Array de bytes que puede ser utilizado para leer el contenido
  - PostedFile: Un objeto HttpPostedFile
    - ContentLength: El tamaño del fichero en bytes
    - ContentType: El tipo MIME (image/gif)
  - SaveAs: Se le pasa un string que contiene la ruta y el nombre del archivo a guardar

Ficheros

Archivo UploadFile.aspx

```
<asp:FileUpload ID="FileUpload1" runat="server" />
<asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="Upload" />
<br />
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

Archivo UploadFile.aspx.cs

```
entArgs e)
{  if (FileUpload1.HasFile)
   {
      string fileName = FileUpload1.FileName;
      FileUpload1.SaveAs(@"C:\Dpaa\"+fileName);
      Label1.Text = "Fichero " + fileName + " adjuntado con éxito.";
    }
    else
      Label1.Text = "No existe fichero adjunto!";
}
```

4

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