Departamento de Lenguajes y Sistemas Informáticos

Unit 13. Web Applications (IV)

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escuela Politécnica Superior
Universidad de Alicante

Index

1. Cookies
2. Email sending
3. Files
4. User controls

Cookies

1

Cookies

- For storing data related to a user we can use the Session object
- Problem:
  - Data are deleted when the user closes the broswer
- Solurtion:
  - In order to store data and preserve them we need to use cookies

Cookies

- Cookies are part of the data of a ASP.NET application that can be stored in the client browser for its later gathering
- Cookies are not lost when the user closes the browser (unless the user removes them)

Cookies in ASP.NET

- A cookie is represented by the class HttpCookie
- User cookies are read by using the Cookies property of the Request object
- They are modified with the Cookies property of the Response object
- By default, they expire when the browser is closed
  - We can altern the expiry points (set a certain expiration date)

Cookies in 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 in ASP.NET

- The userCookie variable is initialised as an instance of theHttpCookie class and set to the value of the UserID cookie.
- The existence of the cookie is checked
  - If it does not exist yet a message is displayed
  - The cookie value is set to “Ana López”
  - An expiry date is set
- The cookie is transferred back to thebrowser using the Response.Cookies.Add method.
- If the cookie value already existed, a Welcome Back message is displayed.

Cookies in ASP.NET

- The first time the page is load the next message will be displayed
  - No existe la cookie, creando cookie ahora
- If the page is reloaded, the cookie already exists, and the next message is shown
  - Bienvenida otra vez, Ana López
- Careful!
  - The users can reject the cookies
  - We cannot leave there important aspects of the application

Cookies in ASP.NET

- Deleting cookies
  - The only way to do it is replacing a cookie by another with expiry date that has already passed

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

2 Email

Email

- Let’s suppose we have an oline shop and we want to send a confirmation email of the order done by every client
- Instead of manually writting each email, we can automate it with ASP.NET

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 Files

Files

- In ASP.NET there exist a set of classes for working with files
  - System.IO
- They allow us reading, writing and updating the content of the files and directories
- There are three types of classes, for:
  - Working with files
  - Working with streams
  - Working with directories

Files

- Classes for working with files and streams:
  - File: Methods for working with files
  - FileStream: Represents an stream for reading and writing files
  - StreamReader: Reads characters from a text file
  - StreamWriter: Writes characters in a text file
  - Path: Methods for manipulating a file or directory

Files

- Writing in a file:

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);
    }
 }
```

Files

- The method CreateText overwrites the file each time is invoked
- Alternative:
  - AppendText: Adds text at the end

- The using directive allows releasing the resources when the writting is finished

Files

- Reading from a file

```
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();
       }
     }
```

Files

- Uploading files
  - The users can upload files to the server
    - Photos
    - Presentations
    - Videos
- In ASP.NET
  - Control FileUpload
  - Provides a TextBox and a Button for selecting the files from the user's computer

Files

- Properties of FileUpload:
  - HasFile: True if the user has uploaded a file
  - FileName: The name of the file as string
  - FileContent: Stream that can be used for reading the content
  - FileBytes: Array of bytes that can be used for reading the content
  - PostedFile: A HttpPostedFile object
    - ContentLength: The size of the file in bytes
    - ContentType: The MIME type (image/gif)
  - SaveAs: We pass a string with the path and name of the file to store

Files

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 User controls

What are them?

- We can create reusable and personalized controls in several pages: USER CONTROLS

- They are files with .ascx extension that we will include in our aspx we pages. The changes done in the ascx will be reflected wherever we have put the user control.

- They contain one or more ASP.NET controls with the needed code for doing the desired functionality.

Differences with webforms

- The extension of the file containing the user control is .ascx.
- Instead of the @Page directive the user control contains a @Control directive that defines the configuration and other properties.
- The user controls cannot be executed as independent files. Instead, we need to add them to ASP.NET pages, as we would do with any other control.
- The user control does not have **html**, **body** or **form** elements. This elements should be in the hosting page.

Creation

- Adding a new element to the project, of user control type
- Once created, we work as it is a normal .aspx page
- For adding the control to any place of our Web, we can drag and drop where we want to use it..

Including a user control in a ASP.NET Web page

- We can do it in two ways:

1. Manually adding a directive in the aspx file with the next attributes:
   - **src:** defines the path to the user control
   - **tagprefix**: allows associating a prefix to the user control
   - **Tagname**: defines the name of the user control
2. We can drag the control from our solution explorer and drop it in the page where we want to iclude it and the directive is automatically added.

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
- An example.
- http://csharp-experience.blogspot.com.es/2009/12/aspnet-eventos-de-controles-de-usuarios.html