Mantener el estado en una aplicación ASP.NET

Concepto de estado en ASP.NET

Los formularios ASP:NET no funcionan de igual manera que un Windows Forms. Por ejemplo, si observamos el ejemplo Acumulador vemos que la variable que se desea incrementar pierde su valor entre postback y postback.

Este se debe a que cuando el cliente recibe la página transferida desde el servidor esta ya ha sido destruida en el mismo.

Elementos que permiten mantener el estado

- Web.Config
- Cookies
- ViewState
- Variables de sesión
- Variables de aplicación

Web.Config

Es posible guardar valores en el Web.Config. Para ello existe una clave denominada <configuration> y dentro de ella una llamada <appSettings>.

Por ejemplo:

```
<appSettings>
  <add key="Empresa" value="Acme" />
</appSettings>
```

Podemos leer esta u otras claves con código similar a:

```
string Empresa = System.Configuration.ConfigurationManager.AppSettings["Empresa"];
```

Siempre se recupera valores del tipo string. Si deseamos guardar por ejemplo un valor entero deberemos castear dicha string a entero.

Es posible también modificar el valor por código fuente en modo de ejecución:

```
Configuration webConfigApp =
```

```
System.Web.Configuration.WebConfigurationManager.OpenWebConfiguration("~");
webConfigApp.AppSettings.Settings["Empresa"].Value = "Acme";
webConfigApp.Save();
```

Normalmente se guardan el el Web.Config valores que no deberían cambiar frecuentemente y sobre todo en tiempo de ejecución. El ejemplo típico es un token o contraseña, cadena de conexión, etc.

Cookies

Son archivos de texto que se alojan el el dispositivo del cliente que consulta la Web.

Las cookies pueden ser de sesión o persistentes.

Una cookie de sesión "vive" hasta que finaliza la sesión que la origino. En cambio,una cookie persistente queda almacenada físicamente en el cliente hasta que este las elimina o estas expiran, ya que es posible especificar en el momento de crearlas el tiempo de vida que tendrá.

Además existen cookies simples y cookies multivalor. Las cookies multivalor pueden almacenar varios valores en una única cookie.

Por ejemplo, el siguiente código crea una cookie de sesión:

```
Response.Cookies["Fecha"].Value = DateTime.Now.ToString();
```

Podemos recuperala con:

```
Request.Cookies["Fecha"].Value
```

Para crear una cookie persistente especificamos cuando debe expirar:

```
Response.Cookies["Fecha"].Value = DateTime.Now.ToString();
Response.Cookies["Fecha"].Expires = DateTime.Now.AddYears(1);
```

Su valor se recupera igual que una cookie de sesión.

No es posible borrar físicamente una cookie en el cliente. Para lograr que no sean tenidas en cuenta por nuestra aplicación debemos logran que estas expiren. Por ejemplo:

```
Response.Cookies["Fecha"].Expires = DateTime.Now.AddDays(-1);
```

Una cookie multivalor se crea de la siguiente manera:

```
Response.Cookies["FechaHora"]["Fecha"] = DateTime.Now.Date.ToString();
```

```
Response.Cookies["FechaHora"]["Hora"] = DateTime.Now.Hour.ToString();
Response.Cookies["FechaHora"].Expires = DateTime.Now.AddYears(1);
```

Para recuperar sus valores:

```
Request.Cookies["FechaHora"]["Fecha"]
Request.Cookies["FechaHora"]["Hora"]
```

Se recomienda un uso moderado de cookies ya que estas tienen varias desventajas:

- el usuario puede inhabilitar el uso de cookies
- puede borrar las cookies cuando desee
- si cambia de dispositivo, los valores guardados en el dispositivo anterior no estarán disponibles.
- pueden afectar su privacidad

ViewState

Básicamente el ViewState es un campo oculto codificado en base 64 que genera el el servidor de ASP.NET para mantener el estado de los controles.

Si observamos el código fuente que llega al cliente de una página ASP.NET encontraremos algo similar a:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="UTbcvPwN9NZEasZGwtQslLSrs26aGrZS75TeluEuO+OayTAc1+476H/z/btsJGj3/izAmpkVYMfY6sQe3j
DZfBfQWEOmUr7GBl8/i9T+m1/LPp8Goe0I2J1OwhGARtn/8tgPhvav9DZ3FkK0dwqmJD5jB55k5LJh/832qKX
wgSza1DmpS76AiTVAs3uMc2yL5hn+VoLQhM2IMIP9t4gM3w==" />

</div>
```

El servidor recupera dicha información cuando la página regresa luego de un PostBack de tal manera que pueda detectar los cambios de estado que se han producido.

Podemos aprovechar este mecanismo para generar almacenar nuestros propios valores en le ViewState y recuperarlos cuando deseamos.

Por ejemplo, el siguiente código almacena el estado de un objeto en el ViewState:

```
Persona obj = new Persona("José");
ViewState["Persona"] = obj;
```

Para recuperar el objeto almacenado podemos utilizar:

```
Persona obj = (Persona)ViewState["Persona"];
```

Es importante destacar que el objeto debe ser serializable.

Además hay que tener en cuenta que no es posible compartir el ViewState entre páginas.

Variables de sesión y variables de aplicación

Las variables de sesión y las variables se aplicación se almacenan en la memoria del servidor.

Las variables de sesión están asociadas a cada sesión (o sea, cada sesión puede mantener sus propias variables) mientras que las variables de aplicación son compartidas por todos las sesiones.

Debajo observamos cómo crear tanto una variable se sesión cómo una variable de aplicación:

```
Persona Jose = new Persona("José");
Session["VariableSesion"] = Jose;

Persona Maria = new Persona("Maria");
Application["VariableAplicacion"] = Maria;
```

Al recuperarlas hay que castearlas al tipo de datos que fue almacenado:

```
Persona Jose = (Persona)Session["VariableSesion"];
Persona Maria = (Persona)Application["VariableAplicacion"];
```

Global Application Class

El Global Application Class o Global.asax es una clase que una vez incorporada a nuestro proyecto nos permite capturar eventos que de otra forma es imposible capturar.

Por ejemplo:

```
protected void Application_Start(object sender, EventArgs e){}

protected void Session_Start(object sender, EventArgs e) {}

protected void Session_End(object sender, EventArgs e){}
```

El evento Application_Start() es útil para inicializar por ejemplo variables de aplicación. El evento Session_Start() para inicializar variables de aplicación.

Transferencia de datos entre páginas

Evidentemente es posible transferir datos entre páginas utilizando variables de sesión o cookies.

Sin embargo, existen otras formas que según el contexto pueden ser más adecuadas:

- Campos ocultos (método POST)
- Parámetros en la URL (método GET)

Transferencia de datos utilizando el método POST

Este método consiste básicamente en recuperar los controles de servidor y sus valores de la página origen en la página destino.

Para ello es necesario buscar el control deseado dentro de la jerarquía de controles de la página origen y castear el objeto recuperado a la clase que corresponde al control de servidor.

Para ello debemos utilizar el método FindControl(). Ejemplo:

```
PreviousPage.Master.FindControl("ContentPlaceHolder").FindControl("txtNombre") != null
```

En el código anterior se busca un control llamado "txtNombre" que se encuentra dentro de un control de servidor "ContentPlaceHolder" que a su vez se encuentra en la Master Page de la página origen.

Transferencia de datos utilizando el método GET

Este método consiste en pasar parámetros de una página a otra a través de la URL de la página destino.

El primer parámetro se coloca luego de la dirección de la página destino precedido por un signo de interrogación (?). Si se pasan más parámetros deben separarse por un ampersand (&). Por ejemplo:

```
Response.Redirect("~/PaginaDestino.aspx?nombre=" + this.txtNombre.Text + "&apellido=" + this.txtApellido.Text);
```

Para recuperar los valores desde la página destino se utiliza el método Request.QueryString():

```
string Nombre = Request.QueryString["nombre"]
```