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 la ocasión pueden ser más adecuadas:

- con **Campos ocultos** (método POST)
- con **Parámetros en la URL** (método GET)
- desde el **objeto Context**

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 (DOM) y convertir 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"]
```

Transferencia de datos utilizando el objeto CONTEXT

Podríamos usar el objeto **Session** para almacenar datos de la sesión y recuperarlos entre las distintas páginas de la aplicación web a medida que navegamos en ella. También podemos usar el objeto **Context**.

El objeto de **Context** de asp.net permite transferir datos de una página a otra, y se lo usa en conjunto con el método **Server.Transfer** que permite la redirección a otra página web (navegación entre páginas). Existe otra forma de redirección usando el método **Response.Redirect** pero esta forma no funciona con el objeto **Context** , el mismo será nulo no pudiendo transferir datos de una página a otra.
La redirección con el método **Server.Transfer** se inicia en el servidor.
La redirección con el método **Response.Redirect** se inicia en el cliente.

Las principales diferencias entre el objeto de **Context** y el objeto **Session** que representa la sesión del usuario, son:

- Con el objeto **Context** debemos usar el método Server.Transfer para redirigir a otra página.
- Si usamos el método Response.Redirect con Context, el valor de Context será nulo en otra página.
- Con el objeto **Session** usamos ambos métodos para redireccionar.

Para asignar un valor al Context y navegar a otra página:

```
Context.Items["Nombre"] = txtNombre.Text;
Server.Transfer("SiguientePagina.aspx");
```

Para recuperar el valor de **Context** en la página destino y en el evento **Page_Load**:

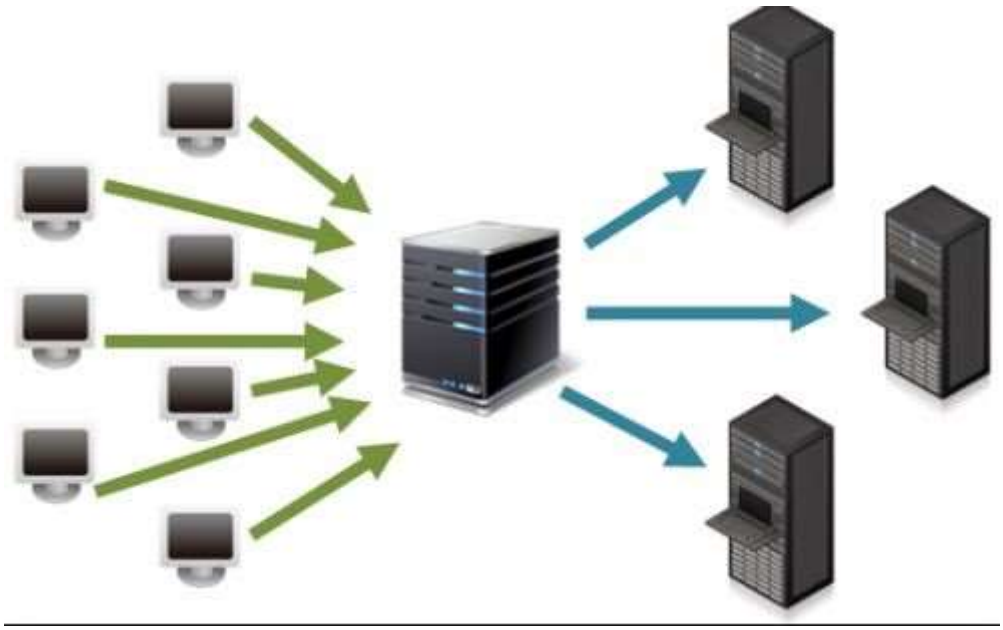
```
        protected void Page_Load(object sender, EventArgs e)
{
            Label1.Text = "Hola = " + Context.Items["Nombre"].ToString();
        }
```

Estado de sesión o variables de aplicación en una granja de Servidores

Modos de Estado de sesión y Concepto de Granja (Farm)

Una **torre de servidores** es un grupo de servidores, normalmente mantenidos por una empresa o universidad para ejecutar tareas que van más allá de la capacidad de una sola máquina corriente, como alternativa, generalmente más económica, a un superordenador.
También hace posible la distribución de tareas, de forma que el sistema gana cierta tolerancia a fallos, ya que si uno de los servidores se estropea, el sistema continúa trabajando, notando únicamente una pérdida de rendimiento.
El término usado en inglés es *server farm* y también podrá encontrarlo con su traducción literal: **granja de servidores**

Generalmente trabajan bajo el lema BALANCEO DE CARGA. Donde un Load Balancer o Balanceador de carga (punto de entrada) se ocupa de determinar que servidor de la granja responderá la solicitud.



El estado de sesión de ASP.NET es compatible con distintas opciones de almacenamiento de los datos de la sesión. Cada opción se identifica mediante un valor en la enumeración SessionStateMode . En la lista siguiente se describen los **modos de estado de sesión** disponibles:

- Modo InProc, que almacena el estado de sesión en memoria en el servidor Web. Éste es el valor predeterminado.
- Modo StateServer, que almacena el estado de sesión en un proceso distinto denominado "servicio de estado de ASP.NET". Este modo garantiza que el estado de sesión se mantiene si se reinicia la aplicación Web y que esté disponible también para varios servidores Web en una batería de servidores Web.
- Modo SQLServer, que almacena el estado de sesión en una base de datos de SQL Server. Este modo garantiza que el estado de sesión se mantiene si se reinicia la aplicación Web y que esté disponible también para varios servidores Web en una batería de servidores Web.
- Modo Custom, que permite especificar un proveedor de almacenamiento personalizado.
- Modo Off, que deshabilita el estado de sesión.

Se puede especificar el modo que desea que utilice el estado de sesión de ASP.NET asignando valores de enumeración SessionStateMode al atributo mode del elemento sessionState en el archivo Web.config de la aplicación. Todos los modos, excepto InProc y Off, requieren parámetros adicionales, como valores de cadena de conexión, que se tratan a continuación en este tema. Puede ver el estado de sesión seleccionado actualmente; para ello, consulte el valor de la propiedad HttpSessionState.Mode.

Navegación entre páginas de una aplicación

Response.Redirect () Vs Server.Transfer ()

Tanto **Response** como **Server** son objetos de ASP.Net y ambos se utilizan para redirigir a un usuario de una página a otra. **Response.Redirect** y **Server.Transfer** ambos métodos se utilizan para redirigir al usuario de una página a otra.

```
Response.Redirect(“Default.aspx”);
Server.Transfer(“Default.aspx”);
```

Podemos usar **Response.Redirect** para enviar al usuario al mismo servidor o a algún otro servidor web.

- **Response.Redirect()** se puede utilizar para redirigir a sitios web externos.
- Puede usar el objeto **Session** con **Response.Redirect()** .
- No se puede usar el objeto de **Context** con **Response.Redirect()** porque el valor del objeto de contexto será nulo mientras la página vaya al servidor.
- **Response.Redirect()** se puede usar tanto para páginas aspx como html.
- Con **Response.Redirect()** se actualiza el historial del navegador.
- **Response.Redirect()** cambia la URL en la barra de direcciones del navegador.

Podemos usar **Server.Transfer** para enviar al usuario al mismo formulario web del servidor, no podemos enviar al usuario a otro servidor.

- Podemos usar **Server.Transfer()** para redirigir solo al mismo sitio web.
- Puede usar **Context** con **Server.Transfer()** para enviar un valor de una página a otra.
- **Server.Transfer()** solo se usa para páginas aspx, no funcionará para páginas html.
- Con **Server.Transfer()** el historial del navegador no se actualiza.
- **Server.Transfer()** no cambia la URL en la barra de direcciones del navegador.