Anexo
Uso de Variables de Sesión con ASP.NET Core MVC

El estado de sesión de ASP.NET Core permite el almacenamiento en memoria de información relativa a la sesión de cada usuario, mientras que el usuario utiliza una aplicación web durante su sesión. Se utiliza un almacén temporal de información para poder conservar la información entre las diferentes solicitudes realizadas desde un mismo programa navegador o cliente. Este almacén temporal de información de la sesión está administrado por la propia aplicación Web, de modo que pueden definirse **variables de sesión** que permiten almacenar información que estará disponible en todas las páginas de la aplicación Web solicitadas durante la sesión del usuario. Las variables de sesión se almacenan en una memoria caché y se consideran datos temporales. Debe tenerse en cuenta que, por razones de rendimiento, la mayor parte de los datos de la aplicación Web deberán almacenarse en bases de datos y, por tanto, solo algunos datos necesarios para realizar determinados procesamientos concretos podrán almacenarse en caché como variables de sesión.

Configurar el estado de la sesión

El paquete *Microsoft.AspNetCore.Session* está incluido implícitamente en ASP.NET Core y proporciona el *middleware* necesario para administrar el estado de sesión. Para habilitar el *middleware* de sesión, el archivo *Program.cs* debe contener:

- Una implementación de la memoria cache distribuida para que sea utilizada como memoria auxiliar para la sesión. Se puede configurar la implementación en memoria *IDistributedCache* como proveedor de sesión en memoria para establecer la memoria auxiliar para la sesión.
- Una llamada al método *AddSession()*.
- Una llamada al método *UseSession()*.

A continuación, se puede observar cómo configurar el proveedor de sesión en memoria, añadiendo el código que aparece resaltado en el archivo *Program.cs*.

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using MvcTienda.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

// Registrar el contexto de la base de datos
builder.Services.AddDbContext<MvcTiendaContexto>(options =>
    options.UseSqlServer(connectionString));

builder.Services.AddDefaultIdentity<IdentityUser>(options =>
options.SignIn.RequireConfirmedAccount = true)
```

```
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddControllersWithViews();

// Configurar el estado de la sesión
builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(20);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

// Configurar el estado de la sesión
app.UseSession();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.MapRazorPages();

app.Run();
```

Debe tener en cuenta que el **orden en la configuración del *middleware*** es importante, de manera que deben respetarse ciertas reglas en cuanto al orden de configuración a la hora de establecer la configuración del estado de la sesión en el archivo de inicio *Program.cs*. Debe llamarse al método *UseSession()* después de llamar al método *UseRouting()*, pero siempre antes de llamar a los métodos *MapControllerRoute()* y *MapRazorPages()*.

El almacenamiento en memoria del contexto del estado de la sesión de usuario, *HttpContext.Session*, está disponible después de configurar el estado de sesión. No se puede acceder al contexto del estado de la sesión *HttpContext.Session* antes de llamar al método *UseSession()*.

En el código anterior puede observarse cómo se establecen las opciones de la sesión en el método *AddSesion()*. Las opciones *Cookie* determina la configuración usada para crear la cookie. Y, la opción *IdleTimeout* indica cuánto tiempo puede estar inactiva la sesión antes de que su contenido de almacenamiento se abandone. Cada acceso a la sesión restablece el tiempo de espera. Este valor solo es aplicable al contenido de la sesión, no a la cookie. Puede apreciarse que se ha establecido un valor a la opción *IdleTimeout* de 20 minutos, que suele ser, además, el valor predeterminado.

Establecer y obtener valores de variables de sesión

La implementación *ISession* proporciona varios métodos de extensión para recuperar y establecer valores enteros y de cadena en variables de sesión. Estos métodos de extensión se aplican al contexto del estado de la sesión *HttpContext.Session* y son los siguientes:

Método	Descripción
Get(String)	Recupera el valor de la variable de sesión *String*
GetInt32(String)	Recupera el valor entero de la variable de sesión *String*
GetString(String)	Recupera el valor de cadena de la variable de sesión *String*
SetInt32(String, Int32)	Establece el valor entero *Int32* a la variable de sesión *String*
SetString(String1, String2)	Establece el valor de cadena *String2* a la variable de sesión *String1*

Los métodos de extensión que comienzan *Get* permiten recuperar el valor de una variable de sesión, mientras que los que comienzan por *Set* permiten establecer el valor a una variable de sesión.

En el siguiente ejemplo se muestra **cómo recuperar o establecer el valor de una variable de sesión**.

```
// POST: Escaparate/AgregarCarrito/5
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AgregarCarrito(int id)
{
    // Cargar datos de producto a añadir al carrito
    var producto = await _context.Productos
                                 .FirstOrDefaultAsync(m => m.Id == id);
    if (producto == null)
    {
        return NotFound();
    }

    // Crear nuevo pedido, si el carrito está vacío y, por tanto, no existe pedido actual
    // La variable de sesión NumPedido almacena el número de pedido del carrito
    //if (string.IsNullOrEmpty(HttpContext.Session.GetString("NumPedido")) )
    if (HttpContext.Session.GetString("NumPedido") == null)
    {
        // Crear objeto pedido a agregar
        Pedido pedido = new Pedido();

        pedido.Fecha = DateTime.Now;
        pedido.Confirmado = null;
        pedido.Preparado = null;
        pedido.Enviado = null;
        pedido.Cobrado = null;
```

```
        pedido.Devuelto = null;
        pedido.Anulado = null;
        pedido.ClienteId = 2;   // Asignar el cliente correspondiente al usuario actual
                                // Pruebas sobre el cliente Id=2
        pedido.EstadoId = 1;    // Estado: "Pendiente" (Sin confirmar)

        if (ModelState.IsValid)
        {
            _context.Add(pedido);
            await _context.SaveChangesAsync();
        }

        // Se asigna el número de pedido a la variable de sesión
        // que almacena el número de pedido del carrito
        HttpContext.Session.SetString("NumPedido", pedido.Id.ToString());
    }

    // Crear objeto detalle para agregar el producto al detalle del pedido del carrito
    Detalle detalle = new Detalle();

    string strNumeroPedido = HttpContext.Session.GetString("NumPedido");
    detalle.PedidoId = Convert.ToInt32(strNumeroPedido);
    detalle.ProductoId = id;      // El valor id tiene el id del producto a agregar
    detalle.Cantidad = 1;
    detalle.Precio = producto.Precio;
    detalle.Descuento = 0;

    if (ModelState.IsValid)
    {
        _context.Add(detalle);
        await _context.SaveChangesAsync();
    }

    return RedirectToAction(nameof(Index));
}
```

En el código anterior puede apreciarse el uso que se hace de la variable de sesión *NumPedido* que permite almacenar el valor del número del pedido que se encuentra actualmente en el carrito de la compra. Este número de pedido sirve para identificar el pedido al cual se agregará cada producto que el usuario desee comprar durante la sesión.

Si el valor de la variable de sesión *NumPedido* es *null*, entonces significa que no existe ningún pedido en el carrito de la compra actual, porque aún no se añadido productos. En este caso, en primer lugar, será necesario crear un nuevo pedido en la tabla *Pedidos*. Una vez creado el nuevo pedido, se asignará el valor del pedido creado a la variable de sesión *NumPedido* para almacenar el número del pedido que se encuentra en el carrito de la compra actualmente. Y, finalmente, se creará una nueva fila en la tabla *Detalles* para añadir el producto que se desea comprar al pedido que se encuentra actualmente en el carrito de la compra.

Si el valor de la variable de sesión *NumPedido* es distinto de *null*, entonces significa que ya existe un pedido en el carrito de la compra y, por tanto, solo es necesario crear una nueva fila en la tabla *Detalles* para añadir el producto que se desea comprar al pedido que se encuentra en el carrito de la compra.

Eliminar una variable de sesión

Conviene eliminar el valor de una variable de sesión cuando se deja de ser utilizada en una aplicación Web. Para ello, se puede utilizar el método de extensión *Remove()* de la forma que puede apreciarse en la siguiente línea de código de ejemplo:

```
HttpContext.Session.Remove("NumPedido");
```

Al eliminar una variable de sesión, su valor será *null* cuando vuelva a ser utilizada.

Eliminar todas las variables de sesión

Conviene eliminar el valor de todas las variables de sesión cuando se cierra la sesión de un usuario en una aplicación Web. Para ello, se puede utilizar el método de extensión *Clear()* de la siguiente forma:

```
HttpContext.Session.Clear();
```

Esto significa que será eliminado el valor almacenado en todas y cada una de las variables de sesión existentes en la sesión actual. El valor de todas las variables de sesión será *null* cuando vuelvan a ser utilizadas.

Referencias:

- Configurar el estado de la sesión

  https://learn.microsoft.com/es-es/aspnet/core/fundamentals/app-state?view=aspnetcore-6.0#configure-session-state

- Establecer y obtener valores de variables de sesión

  https://learn.microsoft.com/es-es/aspnet/core/fundamentals/app-state?view=aspnetcore-6.0#set-and-get-session-values

- Administración del estado de la sesión

  https://learn.microsoft.com/es-es/aspnet/core/fundamentals/app-state?view=aspnetcore-6.0

- Orden del middelware
  https://learn.microsoft.com/es-es/aspnet/core/fundamentals/middleware/?view=aspnetcore-6.0#middleware-order