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 toda 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 la base de datos y que 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 el *framework* de ASP.NET Core y proporciona el *middleware* necesario para administrar el estado de sesión. Para habilitar el middleware de sesión, el archivo *Startup.cs* debe contener:

- Añadir la implementación de la memoria cache distribuida para que sea utilizada como memoria auxiliar para la memoria de la sesión de usuarios en el método *ConfigureServices()*.
- Una llamada al método *AddSession()* en el método *ConfigureServices()*.
- Una llamada a *UseSession()* en el método *Configure()*.

A continuación, se puede observar cómo configurar el proveedor de sesión en memoria, añadiendo el código que aparece resaltado en los métodos *ConfigureServices()* y *Configure()* del archivo *Startup.cs*.

```
. . .

public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
    services.AddDefaultIdentity<IdentityUser>(options =>
options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();

    // Configurar el estado de la sesión.
    services.AddDistributedMemoryCache();
    services.AddSession(options =>
    {
        options.IdleTimeout = TimeSpan.FromMinutes(20);
        options.Cookie.HttpOnly = true;
        options.Cookie.IsEssential = true;
    });
```

```
      services.AddControllersWithViews();

. . .

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
      if (env.IsDevelopment())
      {
          app.UseDeveloperExceptionPage();
          app.UseDatabaseErrorPage();
      }
      else
      {
          app.UseExceptionHandler("/Home/Error");
          app.UseHsts();
      }
      app.UseHttpsRedirection();
      app.UseStaticFiles();

      app.UseRouting();

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

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

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

 . . .
```

Es importante tener en cuenta que el **orden en la configuración del *middleware*** es importante. **Debe llamarse al método UseSession() después de llamar a UseRouting() y a app.UseAuthentication() y app.UseAuthorization(), pero antes de la llamada a UseEndpoints()**. Para saber más sobre el orden del middleware se puede visitar la dirección web: Middleware de ASP.NET Core | Microsoft Docs.

El almacenamiento del estado de la sesión de usuario, *HttpContext.Session*, solo está disponible después de configurar el estado de sesión. No se puede acceder a *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()*. La opción *Cookie* determina la configuración usada para crear la cookie. Y, la opción *IdleTimeout* indica cuánto tiempo puede estar inactiva la sesión antes de que se abandone su contenido. 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 establecer y recuperar valores de cadena y enteros en variables de sesión. Estos métodos de extensión se encuentran en el espacio de nombres *Microsoft.AspNetCore.Http*. Son los siguientes:

- *Get(String)*
- *GetInt32(String)*
- *GetString(String)*
- *SetInt32(String, Int32)*
- *SetString(String, String)*

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 objetos pedido y detalle a agregar
    Pedido pedido = new Pedido();
    Detalle detalle = new Detalle();

    // Crear nuevo pedido, si el carrito está vacío
    // 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)
    {
        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 Id del cliente que corresponde al usuario
                                  // actual. ATENCIÓN: Pruebas sobre el cliente Id=2
        pedido.EstadoId = 1;      // Estado: "Pendiente" (Sin confirmar) tiene Id=1

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

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

    // Agregar producto al detalle de un pedido existente
    string strNumeroPedido = HttpContext.Session.GetString("NumPedido");
    detalle.PedidoId = Convert.ToInt32(strNumeroPedido);

    // El valor de id tiene el Id del producto a agregar
    detalle.ProductoId = id;
    detalle.Cantidad = 1;
    detalle.Precio = producto.Precio;
    detalle.Descuento = 0;
    if (ModelState.IsValid)
    {
        _context.Add(detalle);
        await _context.SaveChangesAsync();
    }

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

Referencias:

- Configurar el estado de la sesión https://docs.microsoft.com/es-es/aspnet/core/fundamentals/app-state?view=aspnetcore-3.1#configure-session-state

- Establecer y obtener valores de variables de sesión https://docs.microsoft.com/es-es/aspnet/core/fundamentals/app-state?view=aspnetcore-3.1#set-and-get-session-values

- Orden del middleware https://docs.microsoft.com/es-es/aspnet/core/fundamentals/middleware/?view=aspnetcore-3.1#order.