Atualizações .NET 6

Atualizações .NET 6

Devido as mudanças no ASP.NET 6 atente-se aos detalhes as mudanças nas classes a seguir conforme implementações nas aulas anteriores.

Classe Program.cs:

```
using AspNetCoreIdentity.Config;
using AspNetCoreIdentity.Extensions;

var builder = WebApplication.CreateBuilder(args);

// Adicionando suporte a Diversos arquivos de configuração por ambiente.
builder.Configuration
    .SetBasePath(builder.Environment.ContentRootPath)
    .AddJsonFile("appsettings.json", true, true)
    .AddJsonFile($"appsettings.{builder.Environment.EnvironmentName}.json",
true, true)
    .AddEnvironmentVariables();

// Adicionando suporte a User Secrets
if (builder.Environment.IsProduction())
{
    builder.Configuration.AddUserSecrets<Program>();
}

// *** Configurando serviços no container ***

// Extension Method de configuração do Identity
builder.Services.AddIdentityConfig(builder.Configuration);

// Extension Method de Authorization (Policies)
builder.Services.AddAuthorizationConfig();

// Extension Method de resolução de DI
builder.Services.ResolveDependencies();

// Extension Method de configuração KissLog
builder.Services.RegisterKissLogListeners();

// Adicionando o MVC com suporte ao filtro de auditoria
builder.Services.AddControllersWithViews(options =>
{
    options.Filters.Add(typeof(AuditoriaFilter));
});

// Adicionando suporte a componentes Razor (ex: Telas do Identity)
builder.Services.AddRazorPages();
```

```
var app = builder.Build();

// *** Configurando o resquest dos serviços no pipeline ***

if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/erro/500");
    app.UseStatusCodePagesWithRedirects("/erro/{0}");
    app.UseHsts();
}

// Força redirect para HTTPS
app.UseHttpsRedirection();

// Adicionando suporte a arquivos (ex: CSS, JS)
app.UseStaticFiles();

// Adicionando suporte a rota
app.UseRouting();

// Autenticacao e autorização (Identity)
app.UseAuthentication();
app.UseAuthorization();

// Rota padrão
app.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");

// Mapeando componentes Razor Pages (ex: Identity)
app.MapRazorPages();

// Adicionando suporte ao KissLog
app.RegisterKissLogListeners(builder.Configuration);

app.Run();
```

Classe LogConfig.cs (configuração do KissLog):

```
using KissLog;
using KissLog.AspNetCore;
using KissLog.CloudListeners.Auth;
using KissLog.CloudListeners.RequestLogsListener;
using KissLog.Formatters;

namespace AspNetCoreIdentity.Config
{
    public static class LogConfig
    {
        public static void RegisterKissLogListeners(this IServiceCollection
services)
        {
```

```
            services.AddHttpContextAccessor();

            // Optional. Register IKLogger if you use KissLog.IKLogger instead
of Microsoft.Extensions.Logging.ILogger<>
            services.AddScoped<IKLogger>((provider) => Logger.Factory.Get());

            services.AddLogging(logging =>
            {
                logging.AddKissLog(options =>
                {
                    options.Formatter = (FormatterArgs args) =>
                    {
                        if (args.Exception == null)
                            return args.DefaultValue;

                        string exceptionStr = new
ExceptionFormatter().Format(args.Exception, args.Logger);

                        return string.Join(Environment.NewLine, new[] {
args.DefaultValue, exceptionStr });
                    };
                });
            });
        }

        public static void RegisterKissLogListeners(this IApplicationBuilder
app, IConfiguration configuration)
        {
            app.UseKissLogMiddleware(options => ConfigureKissLog(options,
configuration));
        }

        private static void ConfigureKissLog(IOptionsBuilder options,
IConfiguration configuration)
        {
            KissLogConfiguration.Listeners
                .Add(new RequestLogsApiListener(new
Application(configuration["KissLog.OrganizationId"],
configuration["KissLog.ApplicationId"]))
                {
                    ApiUrl = configuration["KissLog.ApiUrl"]
                });
        }
    }
}
```

A interface da classe do KissLog mudou afetando as classes:

HomeController.cs

```
public class HomeController : Controller
    {
        // Repare o novo nome da Interface
        private readonly IKLogger _logger;
```

```
        public HomeController(IKLogger logger)
        {
            _logger = logger;
        }
}
```

AuditoriaFilter.cs

```
public class AuditoriaFilter : IActionFilter
    {
        private readonly IKLogger _logger;

        public AuditoriaFilter(IKLogger logger)
        {
            _logger = logger;
        }
}
```