Формат документации API

Фреймворк «Модуль-конструктор» является библиотекой для .NET, а не веб-сервисом, поэтому:

- Документация формируется посредством использования директивы «<GenerateDocumentationFile>true</GenerateDocumentationFile>» в проектных файлах.
- Примеры использования приведены в разделе «Руководство пользователя».

Установка и настройка

Минимальные системные требования:

- CPU: 1 ядро
- RAM: 1 Гб
- HDD: 10 Гб
- Операционная система: Windows 10+/Linux
- .NET SDK 8
- IDE: Visual Studio 2022 / Rider / VS Code

Установка .nuget-пакетов фреймворка на сервер .nuget:

1. Необходимо в периметре Компании развернуть сервер .nuget для хранения пакетов.

2. Скопируйте все пакеты в одну папку и внутри нее выполните стандартные команды для загрузки пакетов на .nuget сервер:

```
dotnet nuget push "*.nupkg" --no-symbols --skip-duplicate --api-key <API Key>

dotnet nuget push "*.snupkg" --skip-duplicate --api-key <API Key>
```

   Пример для GitLab:

```
dotnet nuget add source <.NUGET Server URL>/nuget/index.json" --name taiga --username <UserName> --password <Password> --store-password-in-clear-text

dotnet nuget push "*.nupkg" --no-symbols --source taiga --skip-duplicate --api-key <Password>

dotnet nuget push "*.snupkg" --source taiga --skip-duplicate --api-key <Password>
```

3. Убедитесь в том, что команды отработали без ошибок и пакеты оказались развернуты на сервере .nuget

Руководство пользователя:

Подключение пакеты к проекту:

```
dotnet add package Firewood.Audit
dotnet add package Firewood.Bus.Rebus
dotnet add package Firewood.ORM.EF
dotnet add package Firewood.Service
dotnet add package Firewood.Service.Api
dotnet add package Firewood.UnitTest
```

Пример использования в коде сервиса Service Fabric, взаимодействующего с другими сервисами с помощью шины сообщений:

Файл Program.cs

```
        private static void RegisterInfrastructure(IocContainer container)
        {
            container.AddAmbientContext();
            container.AddUowFactory(new UnitOfWorkFactory<ServiceDbContext>());

            container.AddServiceProviders();

            container.AddProcessorForService();
            container.AddInvokerForServices();

            container.AddNullProjection();

            container.AddProcessorForBus();
            container.AddBus<RebusMessageBrokerV2>();

            container.AddLogger(EventSourceForAudit.Current);
            container.AddLogger(EventSourceForLogicRaw.Current);
            container.AddLogger(EventSourceForLogicWithContext.Current);
            container.AddLogger(EventSourceForRebus.Current);

            container.AddNullTracer();
            container.AddDefaultAudit();

            container.AddSettings(new SettingConfig<IServiceSettings,
ServiceSettings>("ServiceParameters"));
        }
```

Пример использования в коде сервиса Service Fabric, реализующего REST:

Файл Startup.cs

```
    internal class Startup : FirewoodStartup
    {
        public Startup(StartupConfig config) : base(config)
        { }

        protected override void ConfigureServicesAdditional(IServiceCollection services)
        {
            services.AddSingleton<ISwaggerRouteHandler>(new SwaggerRouter());
         }
    }
```