Documentação: Auto Injection de ID nas APIs

1. Classe para a Camada Command

A classe a seguir inicializa os handlers da camada de Command.

```
public static class InicializeCommand
{
    public static IServiceCollection AddCommandHandlers(this IServiceCollection services)
    {
        var assembly = Assembly.GetExecutingAssembly();

        var handlerTypes = assembly.GetTypes()
            .Where(t => t.GetInterfaces().Any(i =>
                i.IsGenericType &&
                i.GetGenericTypeDefinition() == typeof(IRequestHandler<,>)))
            .ToList();

        foreach (var handlerType in handlerTypes)
        {
            services.AddMediatR(handlerType.Assembly);
        }

        return services;
    }
}
```

2. Classe para a Camada Query

A classe abaixo é responsável pela inicialização dos handlers da camada de Query.

```
public static class InicializeQuery
{
    public static IServiceCollection AddQueryHandlers(this IServiceCollection services)
    {
        var assembly = Assembly.GetExecutingAssembly();

        var handlerTypes = assembly.GetTypes()
            .Where(t => t.GetInterfaces().Any(i =>
                i.IsGenericType &&
                i.GetGenericTypeDefinition() == typeof(IRequestHandler<,>)))
            .ToList();

        foreach (var handlerType in handlerTypes)
        {
            services.AddMediatR(handlerType.Assembly);
```

```
        }

        return services;
    }
}
```

3. Classe para CommandStore

Esta classe inicializa os CommandStores que finalizam com o sufixo "CommandStore".

```
public static class InicializeCommandStore
{
    public static IServiceCollection AddCommandStores(this IServiceCollection services)
    {
        var assembly = Assembly.GetExecutingAssembly();

        services.Scan(scan => scan
            .FromAssemblies(assembly)
            .AddClasses(classes => classes.Where(type => type.Name.EndsWith("CommandStore")))
            .AsSelf()
            .WithScopedLifetime());

        return services;
    }
}
```

4. Classe para QueryStore

A classe a seguir inicializa os QueryStores que finalizam com o sufixo "QueryStore".

```
public static class InicializeQueryStore
{
    public static IServiceCollection AddCommandStores(this IServiceCollection services)
    {
        var assembly = Assembly.GetExecutingAssembly();

        services.Scan(scan => scan
            .FromAssemblies(assembly)
            .AddClasses(classes => classes.Where(type => type.Name.EndsWith("CommandStore")))
            .AsSelf()
            .WithScopedLifetime());

        return services;
    }
}
```

5. Camada de API

A camada de API será responsável por injetar os comandos e queries. O código abaixo configura os serviços para a API.

```
public static class DependencyInjectionExtensions
{
    public static IServiceCollection AddApplicationServices(this IServiceCollection services)
    {
        var assemblies = new Assembly[]
        {
            typeof(Command.InicializeCommand).Assembly,
            typeof(Query.InicializeQuery).Assembly,
            typeof(CommandStore.InicializeCommandStore).Assembly,
            typeof(QueryStore.InicializeQueryStore).Assembly,
        };

        services.Scan(scan => scan
            .FromAssemblies(assemblies)
                        .AddClasses(classes  =>  classes.Where(type  =>  type.Name.EndsWith("CommandStore")  ||
type.Name.EndsWith("QueryStore")))
            .AsSelf()
            .WithScopedLifetime());

        return services;
    }
}
```

6. Comando na Classe Program.cs ou Startup.cs

Por fim, para garantir que os serviços sejam adicionados corretamente no pipeline da aplicação, insira o seguinte comando na classe Program.cs ou Startup.cs:

```
builder.Services.AddApplicationServices();
```