

第36期-集成EFCore数据访问

2020年9月27日 9:14

安装基础结构模块包

```
Install-Package Volo.Abp.EntityFrameworkCore  
Install-Package Volo.Abp.EntityFrameworkCore.SqlServer
```

```
[DependsOn(typeof(AbpEntityFrameworkCoreModule))]  
[DependsOn(typeof(AbpEntityFrameworkCoreSqlServerModule))]
```

创建 DbContext 上下文

```
public class Dog : Entity<int>  
{  
    [Required]  
    public string Name { get; set; }  
}  
  
public class MyDbContext : AbpDbContext<MyDbContext>  
{  
    public DbSet<Dog> Dogs { get; set; }  
  
    public MyDbContext(DbContextOptions<MyDbContext> options) : base(options)  
    {  
    }  
  
    protected override void OnModelCreating(ModelBuilder modelBuilder)  
    {  
        base.OnModelCreating(modelBuilder);  
  
        modelBuilder.ConfigureEntities();  
    }  
}
```

实体数据库表映射配置

支持：数据注解和 Fluent API

```
public static class MyDbContextModelCreatingExtensions  
{  
    public static void ConfigureEntities(this ModelBuilder modelBuilder)  
    {  
        Check.NotNull(modelBuilder, nameof(modelBuilder));  
    }  
}
```

```

        modelBuilder.Entity<Dog>(b =>
        {
            b.ToTable("Dogs", "Foo");
            b.HasKey(x => x.Id);
            b.Property(x => x.Name).HasMaxLength(128);
        });
    }
}

```

可用 `b.ConfigureByConvention()`：配置基类基本属性。

可用 `[ConnectionStringName("MySecondConnString")]`：指定要使用的连接字符串。

向依赖注入注册 DbContext 上下文

为了灵活支持模块化、微服务和多租户的数据库分表分库部署方案，连接字符串应该根据不同租户不同模块甚至不同的微服务动态获取。

```

Configure<AbpDbContextOptions>(options =>
{
    options.UseSqlServer();
});

context.Services.AddAbpDbContext<MyDbContext>();

```

创建通用仓储

```

services.AddAbpDbContext<MyDbContext>(options =>
{
    options.AddDefaultRepositories();
});

```

默认情况下，为每个聚合根实体（从派生的类 `AggregateRoot`）创建一个存储库。如果您也想为其他实体创建存储库，请设置 `includeAllEntities` 为 `true`。

```

services.AddAbpDbContext<MyDbContext>(options =>
{
    options.AddDefaultRepositories(includeAllEntities: true);
});

```

使用通用仓储

```

public class DogAppService : ApplicationService
{
    private readonly IRepository<Dog, int> _dogRepository;
}

```

```

//inject default repository to the constructor
public DogAppService(IRepository<Dog, int> dogRepository)
{
    _dogRepository = dogRepository;
}

public async Task Examples()
{
    Dog dog = new Dog { Name = "mydog" };
    await _dogRepository.InsertAsync(dog);
    await _dogRepository.DeleteAsync(1);
    await _dogRepository.UpdateAsync(dog);
    dog = _dogRepository.Where(d => d.Name == dog.Name).First();
    dog = await _dogRepository.FindAsync(2);
}

}

_dogRepository.GetDbContext();
_dogRepository.GetDbSet();

```