عنوان: EF Code First #12
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۶ ۱۰:۵۸:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

پیاده سازی الگوی Context Per Request در برنامه‌های مبتنی بر EF Code first

در طراحی برنامه‌های چند لایه مبتنی بر EF مرسوم نیست که در هر کلاس و متدی که قرار است از امکانات آن استفاده کند، یکبار DbContext و کلاس مشتق شده از آن وهله سازی شوند؛ به این ترتیب امکان انجام امور مختلف در طی یک تراکنش از بین می‌رود. برای حل این مشکل الگویی مطرح شده است به نام Session/Context Per Request و یا به اشتراک گذاری یک Unit of work در لایه‌های مختلف برنامه در طی یک درخواست، که در ادامه یک پیاده سازی آن‌را با هم مرور خواهیم کرد.
البته این سشن با سشن ASP.NET یکی نیست. در NHibernate معادل DbContextایی که در اینجا ملاحظه می‌کنید، Session نام دارد.

اهمیت بکارگیری الگوی Unit of work و به اشتراک گذاری آن در طی یک درخواست

در الگوی واحد کار یا همان DbContext در اینجا، تمام درخواست‌های رسیده به آن، در صف قرار گرفته و تمام آن‌ها در پایان کار، به بانک اطلاعاتی اعمال می‌شوند. برای مثال زمانیکه شیءایی را به یک وهله از DbContext اضافه/حذف می‌کنیم، یا در ادامه مقدار خاصیتی را تغییر می‌دهیم، هیچکدام از این تغییرات تا زمانیکه متد SaveChanges فراخوانی نشود، به بانک اطلاعاتی اعمال نخواهند شد. این مساله مزایای زیر را به همراه خواهد داشت:

الف) کارآیی بهتر
در اینجا از یک کانکشن باز شده، حداکثر استفاده صورت می‌گیرد. چندین و چند عملیات در طی یک batch به بانک اطلاعاتی اعمال می‌گردند؛ بجای اینکه برای اعمال هرکدام، یکبار اتصال جداگانه‌ای به بانک اطلاعاتی باز شود.

ب) بررسی مسایل همزمانی
استفاده از یک الگوی واحد کار، امکان بررسی خودکار تمام تغییرات انجام شده بر روی یک موجودیت را در متدها و لایه‌های مختلف میسر کرده و به این ترتیب مسایل مرتبط با ConcurrencyMode عنوان شده در قسمت‌های قبل به نحو بهتری قابل مدیریت خواهند بود.

ج) استفاده صحیح از تراکنش‌ها
الگوی واحد کار به صورت خودکار از تراکنش‌ها استفاده می‌کند. اگر در حین فراخوانی متد SaveChanges مشکلی رخ دهد، کل عملیات Rollback خواهد شد و تغییری در بانک اطلاعاتی رخ نخواهد داد. بنابراین استفاده از یک تراکنش در حین چند عملیات ناشی از لایه‌های مختلف برنامه، منطقی‌تر است تا اینکه هر کدام، در تراکنشی جدا مشغول به کار باشند.

کلاس‌های مدل مثال جاری

در مثالی که در این قسمت بررسی خواهیم کرد، از کلاس‌های مدل گروه محصولات کمک گرفته شده است:

```
using System.Collections.Generic;


namespace EF_Sample07.DomainClasses
{
    public class Category
    {
        public int Id { get; set; }
        public virtual string Name { get; set; }
        public virtual string Title { get; set; }
```

```
        public virtual ICollection<Product> Products { get; set; }
    }
}
```

```
using System.ComponentModel.DataAnnotations;


namespace EF_Sample07.DomainClasses
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }


        [ForeignKey("CategoryId")]
        public virtual Category Category { get; set; }
        public int CategoryId { get; set; }
    }
}
```

در کلاس Product، یک خاصیت اضافی به نام CategoryId اضافه شده است که توسط ویژگی ForeignKey، به عنوان کلید خارجی جدول معرفی خواهد شد. از این خاصیت در برنامه‌های ASP.NET برای مقدار دهی یک کلید خارجی توسط یک DropDownList پر شده با لیست گروه‌ها، استفاده خواهیم کرد.

پیاده سازی الگوی واحد کار

همانطور که در قسمت قبل نیز ذکر شد، DbContext در EF Code first بر اساس الگوی واحد کار تهیه شده است، اما برای به اشتراک گذاشتن آن بین لایه‌های مختلف برنامه نیاز است یک لایه انتزاعی را برای آن تهیه کنیم، تا بتوان آن‌را به صورت خودکار توسط کتابخانه‌های Dependency Injection یا به اختصار DI در زمان نیاز به استفاده از آن، به کلاس‌های استفاده کننده تزریق کنیم. کتابخانه‌ی DI ایی که در این قسمت مورد استفاده قرار می‌گیرد، کتابخانه معروف StructureMap است. برای دریافت آن می‌توانید از Nuget استفاده کنید؛ یا از صفحه اصلی آن در Github : (^).
اینترفیس پایه الگوی واحد کار ما به شرح زیر است:

```
using System.Data.Entity;
using System;


namespace EF_Sample07.DataLayer.Context
{
    public interface IUnitOfWork
    {
        IDbSet<TEntity> Set<TEntity>() where TEntity : class;
        int SaveChanges();
    }
}
```

برای استفاده اولیه آن، تنها تغییری که در برنامه حاصل می‌شود به نحو زیر است:

```
using System.Data.Entity;
using EF_Sample07.DomainClasses;


namespace EF_Sample07.DataLayer.Context
{
    public class Sample07Context : DbContext, IUnitOfWork
```

```
    {
        public DbSet<Category> Categories { set; get; }
        public DbSet<Product> Products { set; get; }


        #region IUnitOfWork Members
        public new IDbSet<TEntity> Set<TEntity>() where TEntity : class
        {
            return base.Set<TEntity>();
        }
        #endregion
    }
}
```

توضیحات:

با کلاس Context در قسمت‌های قبل آشنا شده‌ایم. در اینجا به معرفی کلاس‌هایی خواهیم پرداخت که در معرض دید EF Code first قرار خواهند گرفت.

DbSetها هم معرف الگوی Repository هستند. کلاس Sample07Context، معرفی الگوی واحد کار یا Unit of work برنامه است.

برای اینکه بتوانیم تعاریف کلاس‌های سرویس برنامه را مستقل از تعریف کلاس Sample07Context کنیم، یک اینترفیس جدید را به نام IUnitOfWork به برنامه اضافه کرده‌ایم.

در اینجا کلاس Sample07Context پیاده سازی کننده اینترفیس IUnitOfWork خواهد بود (اولین تغییر).

دومین تغییر هم استفاده از متد base.Set می‌باشد. به این ترتیب به سادگی می‌توان به DbSetهای مختلف در حین کار با IUnitOfWork دسترسی پیدا کرد. به عبارتی ضرورتی ندارد به ازای تک تک DbSetها یکبار خاصیت جدیدی را به اینترفیس IUnitOfWork اضافه کرد. به کمک استفاده از امکانات Generics مهیا، اینبار

```
uow.Set<Product>
```

معادل همان db.Products سابق است؛ در حالتیکه از Sample07Context به صورت مستقیم استفاده شود.

همچنین نیازی به پیاده سازی متد SaveChanges نیست؛ زیرا پیاده سازی آن در کلاس DbContext قرار دارد.

استفاده از الگوی واحد کار در کلاس‌های لایه سرویس برنامه

```
using EF_Sample07.DomainClasses;
using System.Collections.Generic;


namespace EF_Sample07.ServiceLayer
{
    public interface ICategoryService
    {
        void AddNewCategory(Category category);
        IList<Category> GetAllCategories();
    }
}
```

```
using EF_Sample07.DomainClasses;
using System.Collections.Generic;


namespace EF_Sample07.ServiceLayer
{
    public interface IProductService
    {
        void AddNewProduct(Product product);
        IList<Product> GetAllProducts();
    }
}
```

لایه سرویس برنامه را با دو اینترفیس جدید شروع می‌کنیم. هدف از این اینترفیس‌ها، ارائه پیاده سازی‌های متفاوت، به ازای

ORMهای مختلف است. برای مثال در کلاس‌های زیر که نام آن‌ها با Ef شروع شده است، پیاده سازی خاص Ef Code first را تدارک خواهیم دید. این پیاده سازی، قابل انتقال به سایر ORMها نیست چون نه پیاده سازی یکسانی را از مباحث LINQ ارائه می‌دهند و نه متدهای الحاقی همانندی را به همراه دارند و نه اینکه مباحث نگاشت کلاس‌های آن‌ها به جداول مختلف یکی است:

```
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using EF_Sample07.DataLayer.Context;
using EF_Sample07.DomainClasses;


namespace EF_Sample07.ServiceLayer
{
    public class EfCategoryService : ICategoryService
    {
        IUnitOfWork _uow;
        IDbSet<Category> _categories;
        public EfCategoryService(IUnitOfWork uow)
        {
            _uow = uow;
            _categories = _uow.Set<Category>();
        }


        public void AddNewCategory(Category category)
        {
            _categories.Add(category);
        }


        public IList<Category> GetAllCategories()
        {
            return _categories.ToList();
        }
    }
}
```

```
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using EF_Sample07.DataLayer.Context;
using EF_Sample07.DomainClasses;


namespace EF_Sample07.ServiceLayer
{
    public class EfProductService : IProductService
    {
        IUnitOfWork _uow;
        IDbSet<Product> _products;
        public EfProductService(IUnitOfWork uow)
        {
            _uow = uow;
            _products = _uow.Set<Product>();
        }


        public void AddNewProduct(Product product)
        {
            _products.Add(product);
        }


        public IList<Product> GetAllProducts()
        {
            return _products.Include(x => x.Category).ToList();
        }
    }
}
```

توضیحات:
همانطور که ملاحظه می‌کنید در هیچکدام از کلاس‌های سرویس برنامه، وهله سازی مستقیمی از الگوی واحد کار وجود ندارد. این لایه از برنامه اصلا نمی‌داند که کلاسی به نام Sample07Context وجود خارجی دارد یا خیر.
همچنین لایه اضافی دیگری را به نام Repository جهت مخفی سازی سازوکار EF به برنامه اضافه نکرده‌ایم. این لایه شاید در نگاه اول برنامه را مستقل از ORM جلوه دهد اما در عمل قابل انتقال نیست و سبب تحمیل سربار اضافی بی موردی به برنامه می‌شود؛ ORMها ویژگی‌های یکسانی را ارائه نمی‌دهند. حتی در حالت استفاده از LINQ، پیاده سازی‌های یکسانی را به همراه ندارند.
بنابراین اگر قرار است برنامه مستقل از ORM کار کند، نیاز است لایه استفاده کننده از سرویس برنامه، با دو اینترفیس IProductService و ICategoryService کار کند و نه به صورت مستقیم با پیاده سازی آن‌ها. به این ترتیب هر زمان که لازم شد، فقط باید پیاده سازی‌های کلاس‌های سرویس را تغییر داد؛ باز هم برنامه نهایی بدون نیاز به تغییری کار خواهد کرد.

تا اینجا به معماری پیچیده‌ای نرسیده‌ایم و اصطلاحا over-engineering صورت نگرفته است. یک اینترفیس بسیار ساده IUnitOfWork به برنامه اضافه شده؛ در ادامه این اینترفیس به کلاس‌های سرویس برنامه تزریق شده است (تزریق وابستگی در سازنده کلاس). کلاس‌های سرویس ما «می‌دانند» که EF وجود خارجی دارد و سعی نکرده‌ایم توسط لایه اضافی دیگری آن‌را مخفی کنیم. شیوه کار با IDbSet تعریف شده دقیقا همانند روال متداولی است که با EF Code first کار می‌شود و بسیار طبیعی جلوه می‌کند.

استفاده از الگوی واحد کار و کلاس‌های سرویس تهیه شده در یک برنامه کنسول ویندوزی

در ادامه برای وهله سازی اینترفیس‌های سرویس و واحد کار برنامه، از کتابخانه StructureMap که یاد شد، استفاده خواهیم کرد. بنابراین، تمام برنامه‌های نهایی ارائه شده در این قسمت، ارجاعی را به اسمبلی StructureMap.dll نیاز خواهند داشت.
کدهای برنامه کنسول مثال جاری را در ادامه ملاحظه خواهید کرد:

```
using System.Collections.Generic;
using System.Data.Entity;
using EF_Sample07.DataLayer.Context;
using EF_Sample07.DomainClasses;
using EF_Sample07.ServiceLayer;
using StructureMap;


namespace EF_Sample07
{
    class Program
    {
        static void Main(string[] args)
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample07Context,
Configuration>());

            HibernatingRhinos.Profiler.Appender.EntityFramework.EntityFrameworkProfiler.Initialize();
            ObjectFactory.Initialize(x =>
            {
                x.For<IUnitOfWork>().CacheBy(InstanceScope.Hybrid).Use<Sample07Context>();
                x.For<ICategoryService>().Use<EfCategoryService>();
            });


            var uow = ObjectFactory.GetInstance<IUnitOfWork>();
            var categoryService = ObjectFactory.GetInstance<ICategoryService>();


            var product1 = new Product { Name = "P100", Price = 100 };
            var product2 = new Product { Name = "P200", Price = 200 };
            var category1 = new Category
            {
                Name = "Cat100",
                Title = "Title100",
                Products = new List<Product> { product1, product2 }
            };
            categoryService.AddNewCategory(category1);
            uow.SaveChanges();
        }
    }
```

```
}
```

در اینجا بیشتر هدف، معرفی نحوه استفاده از StructureMap است.
ابتدا توسط متد ObjectFactory.Initialize مشخص می‌کنیم که اگر برنامه نیاز به اینترفیس IUnitOfWork داشت، لطفا کلاس SampleO7Context را وهله سازی کرده و مورد استفاده قرار بده. اگر ICategoryService مورد استفاده قرار گرفت، وهله مورد نظر باید از کلاس EfCategoryService تامین شود.
توسط ObjectFactory.GetInstance نیز می‌توان به وهله‌ای از این کلاس‌ها دست یافت و نهایتا با فراخوانی uow.SaveChanges می‌توان اطلاعات را ذخیره کرد.

چند نکته:
- به کمک کتابخانه StructureMap، تزریق IUnitOfWork به سازنده کلاس EfCategoryService به صورت خودکار انجام می‌شود. اگر به کدهای فوق دقت کنید ما فقط با اینترفیس‌ها مشغول به کار هستیم، اما وهله‌سازی‌ها در پشت صحنه انجام می‌شود.
- حین معرفی IUnitOfWork از متد CacheBy با پارامتر InstanceScope.Hybrid استفاده شده است. این enum مقادیر زیر را می‌تواند بپذیرد:

```
public enum InstanceScope
{
        PerRequest = 0,
        Singleton = 1,
        ThreadLocal = 2,
        HttpContext = 3,
        Hybrid = 4,
        HttpSession = 5,
        HybridHttpSession = 6,
        Unique = 7,
        Transient = 8,
}
```

برای مثال اگر در برنامه‌ای نیاز داشتید یک کلاس به صورت Singleton عمل کند، فقط کافی است نحوه کش شدن آن‌را تغییر دهید.
حالت PerRequest در برنامه‌های وب کاربرد دارد (و حالت پیش فرض است). با انتخاب آن وهله سازی کلاس مورد نظر به ازای هر درخواست رسیده انجام خواهد شد.
در حالت ThreadLocal، به ازای هر Thread، وهله‌ای متفاوت در اختیار مصرف کننده قرار می‌گیرد.
با انتخاب حالت HttpContext، به ازای هر HttpContext ایجاد شده، کلاس معرفی شده یکبار وهله سازی می‌گردد.
حالت Hybrid ترکیبی است از حالت‌های HttpContext و ThreadLocal. اگر برنامه وب بود، از HttpContext استفاده خواهد کرد در غیراینصورت به ThreadLocal سوئیچ می‌کند.

استفاده از الگوی واحد کار و کلاس‌های سرویس تهیه شده در یک برنامه ASP.NET MVC

یک برنامه خالی ASP.NET MVC را آغاز کنید. سپس یک HomeController جدید را نیز به آن اضافه نمائید و کدهای آن‌را مطابق اطلاعات زیر تغییر دهید:

```
using System.Web.Mvc;
using EF_Sample07.DomainClasses;
using EF_Sample07.ServiceLayer;
using EF_Sample07.DataLayer.Context;
using System.Collections.Generic;


namespace EF_Sample07.MvcAppSample.Controllers
{
    public class HomeController : Controller
    {
        IProductService _productService;
        ICategoryService _categoryService;
        IUnitOfWork _uow;
```

```
        public HomeController(IUnitOfWork uow, IProductService productService, ICategoryService
categoryService)
        {
            _productService = productService;
            _categoryService = categoryService;
            _uow = uow;
        }


        [HttpGet]
        public ActionResult Index()
        {
            var list = _productService.GetAllProducts();
            return View(list);
        }



        [HttpGet]
        public ActionResult Create()
        {
            ViewBag.CategoriesList = new SelectList(_categoryService.GetAllCategories(), "Id", "Name");
            return View();
        }


        [HttpPost]
        public ActionResult Create(Product product)
        {
            if (this.ModelState.IsValid)
            {
                _productService.AddNewProduct(product);
                _uow.SaveChanges();
            }


            return RedirectToAction("Index");
        }


        [HttpGet]
        public ActionResult CreateCategory()
        {
            return View();
        }


        [HttpPost]
        public ActionResult CreateCategory(Category category)
        {
            if (this.ModelState.IsValid)
            {
                _categoryService.AddNewCategory(category);
                _uow.SaveChanges();
            }


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

نکته مهم این کنترلر، تزریق وابستگی‌ها در سازنده کلاس کنترلر است؛ به این ترتیب کنترلر جاری نمی‌داند که با کدام پیاده سازی خاصی از این اینترفیس‌ها قرار است کار کند.

اگر برنامه را به همین نحو اجرا کنیم، موتور ASP.NET MVC ایراد خواهد گرفت که یک کنترلر باید دارای سازنده‌ای بدون پارامتر باشد تا من بتوانم به صورت خودکار وهله‌ای از آن‌را ایجاد کنم. برای رفع این مشکل از کتابخانه StructureMap برای تزریق خودکار وابستگی‌ها کمک خواهیم گرفت:

```
using System;
using System.Data.Entity;
using System.Web.Mvc;
using System.Web.Routing;
using EF_Sample07.DataLayer.Context;
```

```
using EF_Sample07.ServiceLayer;
using StructureMap;


namespace EF_Sample07.MvcAppSample

{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801


    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }


        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");


            routes.MapRoute(
                "Default", // Route name
                "{controller}/{action}/{id}", // URL with parameters
                new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter
defaults
            );
        }


        protected void Application_Start()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample07Context,
Configuration>());
            HibernatingRhinos.Profiler.Appender.EntityFramework.EntityFrameworkProfiler.Initialize();
            AreaRegistration.RegisterAllAreas();
            RegisterGlobalFilters(GlobalFilters.Filters);
            RegisterRoutes(RouteTable.Routes);
            initStructureMap();
        }


        private static void initStructureMap()
        {
            ObjectFactory.Initialize(x =>
            {
                x.For<IUnitOfWork>().HttpContextScoped().Use(() => new Sample07Context());
                x.ForRequestedType<ICategoryService>().TheDefaultIsConcreteType<EfCategoryService>();
                x.ForRequestedType<IProductService>().TheDefaultIsConcreteType<EfProductService>();
            });


            //Set current Controller factory as StructureMapControllerFactory
            ControllerBuilder.Current.SetControllerFactory(new StructureMapControllerFactory());
        }


        protected void Application_EndRequest(object sender, EventArgs e)
        {
            ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects();
        }
    }


    public class StructureMapControllerFactory : DefaultControllerFactory
    {
        protected override IController GetControllerInstance(RequestContext requestContext, Type
controllerType)
        {
            return ObjectFactory.GetInstance(controllerType) as Controller;
        }
    }
}
```

توضیحات:

کدهای فوق متعلق به کلاس Global.asax.cs هستند. در اینجا در متد Application_Start، متد initStructureMap فراخوانی شده است.
با پیاده سازی ObjectFactory.Initialize در کدهای برنامه کنسول معرفی شده آشنا شدیم. اینبار فقط حالت کش شدن کلاس Context برنامه را HttpContextScoped قرار داده‌ایم تا به ازای هر درخواست رسیده یک بار الگوی واحد کار وهله سازی شود.
نکته مهمی که در اینجا اضافه شده‌است، استفاده از متد ControllerBuilder.Current.SetControllerFactory می‌باشد. این متد نیاز به وهله‌ای از نوع DefaultControllerFactory دارد که نمونه‌ای از آن‌را در کلاس StructureMapControllerFactory مشاهده می‌کنید. به این ترتیب در زمان وهله سازی خودکار یک کنترلر، اینبار StructureMap وارد عمل شده و وابستگی‌های برنامه را مطابق تعاریف ObjectFactory.Initialize ذکر شده، به سازنده کلاس کنترلر تزریق می‌کند.
همچنین در متد Application_EndRequest با فراخوانی ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects از نشتی اتصالات به بانک اطلاعاتی جلوگیری خواهیم کرد. چون وهله الگوی کار برنامه HttpScoped تعریف شده، در پایان یک درخواست به صورت خودکار توسط StructureMap پاکسازی می‌شود و به نشتی منابع نخواهیم رسید.

استفاده از الگوی واحد کار و کلاس‌های سرویس تهیه شده در یک برنامه ASP.NET Web forms

در یک برنامه ASP.NET Web forms نیز می‌توان این مباحث را پیاده سازی کرد:

```
using System;
using System.Data.Entity;
using EF_Sample07.DataLayer.Context;
using EF_Sample07.ServiceLayer;
using StructureMap;


namespace EF_Sample07.WebFormsAppSample
{
    public class Global : System.Web.HttpApplication
    {
        private static void initStructureMap()
        {
            ObjectFactory.Initialize(x =>
            {
                x.For<IUnitOfWork>().HttpContextScoped().Use(() => new Sample07Context());
                x.ForRequestedType<ICategoryService>().TheDefaultIsConcreteType<EfCategoryService>();
                x.ForRequestedType<IProductService>().TheDefaultIsConcreteType<EfProductService>();


                x.SetAllProperties(y=>
                {
                    y.OfType<IUnitOfWork>();
                    y.OfType<ICategoryService>();
                    y.OfType<IProductService>();
                });
            });
        }


        void Application_Start(object sender, EventArgs e)
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample07Context,
Configuration>());
            HibernatingRhinos.Profiler.Appender.EntityFramework.EntityFrameworkProfiler.Initialize();
            initStructureMap();
        }


        void Application_EndRequest(object sender, EventArgs e)
        {
            ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects();
        }
```

در اینجا کدهای کلاس Global.asax.cs را ملاحظه می‌کنید. توضیحات آن با قسمت ASP.NET MVC آنچنان تفاوتی ندارد و یکی است. البته منهای تعاریف SetAllProperties که جدید است و در ادامه به علت اضافه کردن آن‌ها خواهیم رسید.
در ASP.NET Web forms برخلاف ASP.NET MVC نیاز است کار وهله سازی اینترفیس‌ها را به صورت دستی انجام دهیم. برای این

منظور و کاهش کدهای تکراری برنامه می‌توان یک کلاس پایه را به نحو زیر تعریف کرد:

```
using System.Web.UI;
using StructureMap;


namespace EF_Sample07.WebFormsAppSample
{
    public class BasePage : Page
    {
        public BasePage()
        {
            ObjectFactory.BuildUp(this);
        }
    }
}
```

سپس برای استفاده از آن خواهیم داشت:

```
using System;
using EF_Sample07.DataLayer.Context;
using EF_Sample07.DomainClasses;
using EF_Sample07.ServiceLayer;


namespace EF_Sample07.WebFormsAppSample
{
    public partial class AddProduct : BasePage
    {
        public IUnitOfWork UoW { set; get; }
        public IProductService ProductService { set; get; }
        public ICategoryService CategoryService { set; get; }


        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                bindToCategories();
            }
        }


        private void bindToCategories()
        {
            ddlCategories.DataTextField = "Name";
            ddlCategories.DataValueField = "Id";
            ddlCategories.DataSource = CategoryService.GetAllCategories();
            ddlCategories.DataBind();
        }


        protected void btnAdd_Click(object sender, EventArgs e)
        {
            var product = new Product
            {
                Name = txtName.Text,
                Price = int.Parse(txtPrice.Text),
                CategoryId = int.Parse(ddlCategories.SelectedItem.Value)
            };
            ProductService.AddNewProduct(product);
            UoW.SaveChanges();
            Response.Redirect("~/Default.aspx");
        }
    }
}
```

اینبار وابستگی‌های کلاس افزودن محصولات، به صورت خواصی عمومی تعریف شده‌اند. این خواص عمومی توسط متد SetAllProperties که در فایل global.asax.cs معرفی شدند، باید یکبار تعریف شوند (مهم!).

سپس اگر دقت کرده باشید، اینبار کلاس AddProduct از BasePage ما ارث بری کرده است. در سازند کلاس BasePage، با فراخوانی متد ObjectFactory.BuildUp، تزریق وابستگی‌ها به خواص عمومی کلاس جاری صورت می‌گیرد.
در ادامه نحوه استفاده از این اینترفیس‌ها را جهت مقدار دهی یک DropDownList یا ذخیره سازی اطلاعات یک محصول مشاهده می‌کنید. در اینجا نیز کار با اینترفیس‌ها انجام شده و کلاس جاری دقیقا نمی‌داند که با چه وهله‌ای مشغول به کار است. تنها در زمان اجرا است که توسط StructureMap ، به ازای هر اینترفیس معرفی شده، وهله‌ای مناسب بر اساس تعاریف فایل Global.asax.cs در اختیار برنامه قرار می‌گیرد.

کدهای کامل مثال‌های این سری را از آدرس زیر هم می‌توانید دریافت کنید: (^)

به روز رسانی
کدهای قسمت جاری را به روز شده جهت استفاده از EF 6 و StructureMap 3 در VS 2013، از اینجا می‌توانید دریافت کنید:
EF_Sample07.zip

نظرات خوانندگان

نویسنده: Sh Mjsoft
تاریخ: ۱۳۹۱/۰۲/۲۶ ۱۸:۵۳:۵۱

واقعا ممنون از مطالبت

نویسنده: NTC
تاریخ: ۱۳۹۱/۰۲/۲۶ ۲۲:۳۷:۰۴

این قسمت کمی مشکل بود.کمی گیج شدم
تصویر زیر را ببینید.
در یک Win Application
جای تعاریف درست است؟؟
چرا عبارت "HibernatingRhinos" شناخته نمیشود؟

نویسنده: Sirwan Afifi
تاریخ: ۱۳۹۱/۰۲/۲۶ ۲۲:۴۸:۱۳

سلام استاد خیلی ممنون
یه سوال :

این متد SaveChanges خودش به صورت توکار کار مدیریت کانکشن (Open و Close کردن کانکشن) رو انجام میده مثل متد Fill کلاس SqlDataAdapter در ADO.NET یا روال کار در اینجا به صورت دیگری است؟ در کل منظورم همون بحث Connection هPooling

نویسنده: Sirwan Afifi
تاریخ: ۱۳۹۱/۰۲/۲۶ ۲۲:۵۲:۲۳

استاد راستی برای یادگیری و تسلط به این الگوها شما کتابی خاصی رو میتونید بهم معرفی کنید؟
چون اطلاعاتم در رابطه با این الگو ها کم بود نتونستم از کل مطالبتون استفاده کنم.
در هر حال ممنون از اینکه دانش خودتون رو به اشتراک میذارید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۰۰:۱۴:۰۲

قبلا در این مورد مطلب نوشتم: (^) | (^)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۰۰:۱۵:۳۳

مدیریت اتصالات توسط DbContext مدیریت می‌شود؛ با وهله سازی و سپس dispose آن.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۰۰:۱۶:۵۰

HibernatingRhinos مربوط است به برنامه EF Prodiler که در قسمت 10 توضیح دادم.

نویسنده: amir hosein jelodari
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۱:۰۶:۱۰

فقط میتونم بگم که دمتون گرم ... یعنی ایول :دی

نویسنده: Hossein Raziee
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۱:۱۱:۲۴

با درود
ابتدا سپاس به خاطر مطالب بسیار مفیدی که مینویسید.
در یک پروژه وب که به صورت ماژولار تعریف شده باشه و در ابتدا مشخص نباشه که چه امکاناتی داره و قرار باشه در آینده به پروژه اضافه بشه، نمیتونیم Model های مختلف رو در ابتدا در DbContext تعریف کرد.
بنابر این باید برای هر ماژول dll ای تولید کرد که حاوی DomainClass ها , ServiceLayer ها ، Controller ها و DbContext مربوط به اون ماژول باشه.
به نظر شما برای تعریف این قسمت ها در Application_Start باید چه کار کید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۱:۲۹:۴۱

StructureMap امکان اسکن اسمبلی‌های اضافه شده به سیستم را دارد. باید وقت بگذارید مستندات آن‌را مطالعه کنید: (^)

نویسنده: NTC
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۴:۲۹:۳۰

شرمنده
کتابخانه ی structuremap را هم از NuGet اضافه کردم و هم از Github.
ولی هر دور در زمان اجرا اخطار زیر را میدهند :
"The type or namespace name 'StructureMap' could not be found (are you missing a using directive or an assembly reference?)"

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۶:۰۷:۴۷

StructureMap در حالت استفاده از Client profile کار نمی‌کند و load نمی‌شود. علت هم این است که ارجاعی را به اسمبلی System.Web دارد. به همین جهت به خواص پروژه مراجعه کرده و Client profile را به حالت Full تغییر دهید، مشکل حل می‌شود.

نویسنده: Mona
تاریخ: ۱۳۹۱/۰۲/۳۰ ۰۹:۱۱:۵۸

با سلام خدمت آقای نصیری
با تشکر فراوان از مطالب بسیار حوب و سطح بالای شما.
متاسفانه این مطلب کمی برای من سنگین است.
سوال: اگر به هیچ عنوان قصد تغییر ORM را در برنامه نداشته باشیم و فقط بخواهیم از قابلیت های Session/Context Per Request استفاده کنیم و کلاس واسط Service را پیاده سازی کنیم (فرضا در ASP.NET MVC)، امکان دارد راهنمایی بفرمایید یا نمونه ای را معرفی کنید.
با تشکر
(کمی هنگ کردم)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۳۰ ۱۰:۱۱:۳۴

اصل قضیه در اینجا مدیریت Context در طی یک درخواست Http است که به خوبی توسط StructureMap مدیریت می‌شود؛ فقط

با چند سطر کد نویسی (قسمت HttpContextScoped و بعد هم ReleaseAndDisposeAllHttpScopedObjects). اگر بخواهید اینترفیس‌ها را حذف کنید و از StructureMap استفاده نکنید به چند صد سطر کد برای جایگزینی آن خواهید رسید که ضرورتی ندارد.

نویسنده: Mona
تاریخ: ۱۳۹۱/۰۲/۳۰ ۱۰:۵۰:۰۹

بسیار متشکرم از پاسخ کامل و سریع شما.
البته مشکل اصلی من با تعدد کلاس ها و زمانبر بودن پیاده سازی آنها است.
مثلاً برای کلاس Product یک بار باید خود آن را ساخت، یک بار EfProductService و IProductService
آیا راه ساده تری وجود دارد؟ یا من کلاً قضیه را اشتباه متوجه شدم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۳۰ ۱۱:۰۰:۲۰

ساخت اینترفیس از روی کلاس در ویژوال استودیو ساده است. روی نام کلاس کلیک راست کنید. بعد گزینه Refactor و سپس گزینه Extract Interface را انتخاب کنید. با چند کلیک، یک اینترفیس کامل برای شما تولید خواهد شد.

نویسنده: iman
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۲:۲۶

با سلام خدمت استاد عزیز
اول از همه بی نهایت از مطالب خوبتون تشکر میکنم.مقالات شما در پیشرفت من فوق العاده تاثیر داشت.برای تشکر فقط میتونم بگم واقعاً خسته نباشی و امیدوارم همیشه سلامت باشید.
من این pattern رو در پروژه خودم استفاده کردم و واقعاً سودمند بود.من برای validation سمت سرور از data annotation استفاده کردم و هنگام save changes عمل validation فراخوانی و اجرا میشود.اما چون میخواستم code behind رو کاملاً خلوت کنم ، exception و مقادیر خروجی validation error رو در ایمپلیمنت اینترفیس بررسی کردم و برای اینکار

```
_uow.SaveChanges();
```

رو بجای code behind در ایمپلیمنت اینترفیس نوشتم.این روش درست است؟
آیا شما برای چک کردن validation با code first روش دیگری را پیشنهاد میکنید؟البته با توجه به pattern فوق.
در ضمن پروژه بنده asp.net web form می باشد.
با تشکر از استاد و معذرت بخاطر طولانی شدن سوال

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۲:۳۲

سلام؛ من هم تقریبا همین کار رو انجام می‌دم.

```
public class MyDbContextBase : DbContext, IUnitOfWork
```

یک کلاس پایه MyDbContextBase دارم که پیاده سازی IUnitOfWork و DbContext اصلی خود EF رو داره. به این ترتیب حجم کدهای تکراری من کم میشه و از این کلاس پایه استفاده می‌کنم. داخلش در زمان Save تغییرات می‌شود DbEntityValidationException را هم بررسی کرد و مواردی از این دست.
البته اگر از MVC استفاده کنید این data annotation در سمت کلاینت هم به صورت خودکار اعمال می‌شود.

نویسنده: iman
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۲:۳۹

ممنون استاد.یه سوال دیگه که شاید مربوط به این بخش نباشد.
asp.net web form بر خلاف mvc
attribute برای compare validation ندارد.این مشکل رو چگونه حل کنم؟
حقیقتش خودم خواستم با ایمپلیمنت کلاس validation Attribute اینکار رو انجام بدم ولی نشد.
ممنون میشم کمکم کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۳:۴۳

دستی باید این مساله رو مدیریت کنید. سمت سرور آن: دو فیلد دارید که باید مقایسه شوند. سمت کلاینت آن هم در اینجا بحث شده.

نویسنده: iman
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۴:۷

متشکرم استاد
سمت کلاینت رو با Jquery Validation انجام دادم.مشکلم سمت سرور بود که میخواستم مثل بقیه attribute ها ، واسه این مورد هم از attribue استفاده کنم که انگار راهی نیست و باید دستی انجام داد.
ممنون از وقتی که گذاشتید.هر روز منتظر مطالب پر بارتون هستم

نویسنده: Mona
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۳:۴۶

با سلام خدمت جناب نصیری عزیز
بسیار متشکرم از مطالب شما
آیا امکان دارد روش خودتان را که می‌فرمایید به صورت یک پروژه Open Source ارائه کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۳:۵۶

ترکیبی است از همین مثال سورس باز فوق و قسمت tracking که در مورد یکی کردن ی و ک ارسالی پیشتر بحث شد. مورد بیشتری ندارد. فقط یک try/catch اضافه شده در زمان save نهایی که DbEntityValidationException را بررسی می‌کند.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۷:۳۲

بار دیگر سلام ... سوالی که دارم اینه که آیا استفاده از context (در اینجا sample07context) در لایه‌ی سرویس کار درستی است؟! ... چون بعضی از مواقع به ویژگی‌های کلاس DbContext نیاز میشود مثه Entry !

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۸:۲۰

در طراحی فوق در تمام مواقع در لایه سرویس از IUnitOfWork شرح داده شده استفاده می‌شود؛ که توسط IDbSet های متناظر با اشیاء در معرض دید EF ، با EF ارتباط برقرار می‌کند.

نویسنده: امیرحسین جلوداری

تاریخ: ۱۳۹۱/۰۳/۲۹ ۲۳:۵۰

بنده همین محدودیت رو عرض کردم! ... اصلن چه دلیلی داره که لایه‌ی سرویس رو به استفاده از IunitOfWork محدود کنیم؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۹ ۲۳:۵۳

این محدودیت نیست. IUnitOfWork دسترسی کاملی رو به Context جاری به شما می‌ده. مطلب رو یکبار لطفا از ابتدا مطالعه کنید. اگر از IUnitOfWork استفاده نشه باید به صورت مستقیم Context رو وهله سازی کنید. یعنی هر کلاسی یکبار تراکنش خاص خودش را باید باز کند، یکبار کانکشن خاص خودش را. با استفاده از IUnitOfWork اگر متد جاری شما از 10 کلاس هم استفاده کند، تماما با یک وهله از Context کار می‌کنند (یعنی یک کانکشن و یک تراکنش که نحوه صحیح کار به این صورت است).

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۳/۳۰ ۰:۱۷

یعنی اگه بخوام فرضا از متد Entry تو DbContext استفاده کنم باید اونو تو اینترفیس IUnitOfWork قرار بدم؟

```
public interface IUnitOfWork
    {
        IDbSet<TEntity> Set<TEntity>() where TEntity : class;
        int SaveChanges();
        DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;
    }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۰ ۰:۴۷

بله. اینکار رو میشه انجام داد. در زمان استفاده، نیازی هم به پیاده سازی نداره چون در DbContext تعریف شده و از همان استفاده می‌شود.
مزیت استفاده از اینترفیس در اینجا این است که کتابخانه DI مورد استفاده کار تزریق تنها یک وهله از Context رو به n کلاسی که هم اکنون مثلا در روال جاری کلیک برنامه درگیر هستند و تنها IUnitOfWork رو می‌شناسند به صورت خودکار انجام می‌ده. یک کانکشن؛ یک تراکنش؛ سربار کم و سرعت بالای کار.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱:۱۹

خیلی ممنون ... شیرفهم شدم :دی

نویسنده: iman
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۲:۲۰

با سلام و تشکر از زحمات استاد نصیری
من چند روز پیش مطلبی رو راجع به ساخت attribute برای compare validation در روش code first عنوان کردم(البته در asp.net web form).که در واقع اصلی‌ترین کاربردش همون کاربرد معروف مقایسه رمز عبور و تکرار رمز عبور هست. یه سری کارایی در این زمینه انجام دادم و خواستم در مورد مشکل مربوطه و در کل ، صحیح یا غلط بودن این روش کمکم کنید.

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Field |
AttributeTargets.Parameter, AllowMultiple = true, Inherited = true)]
    public  class CompareAttribute : ValidationAttribute
    {
        public CompareAttribute(string originalProperty, string confirmProperty)
        {
            OriginalProperty = originalProperty;
            ConfirmProperty = confirmProperty;
        }
        public string ConfirmProperty
        {
            get;
```

```
            private set;
        }
        public string OriginalProperty
        {
            get;
            private set;
        }
        public override bool IsValid(object value)
        {
            PropertyDescriptorCollection properties = TypeDescriptor.GetProperties(value);
            return String.Equals(((User)value).Password, ((User)value).RepeatPassword);
        }
    }
```

تا اینجا یه کلاس تعریف شده که خیلی خلاصه Validation Attribute رو پیاده سازی کرده.این هم از کلاس User ، فقط در حد تعریف propertyهای لازم این مثال

```
//[CompareAttribute("Password", "RepeatPassword", ErrorMessage = "Not Compare!")]
    public class User
    {
        public Int64 UserId { get; set; }




        [Required(ErrorMessageResourceName = "Password", ErrorMessageResourceType =
typeof(ErrorMessageResource))]
        public String Password { get; set; }




        [NotMapped]
        [Required(ErrorMessageResourceName = "RepeatPassword", ErrorMessageResourceType =
typeof(ErrorMessageResource))]
       //[CompareAttribute("Password","RepeatPassword" , ErrorMessage = "Not Compare!")]
        public String RepeatPassword { get; set; }
    }
```

مشکل اصلی استفاده از همین compare attribute هست که ساخته شده .
وقتی اونو بالای کلاس User میزارم کار میکنه.ولی خب این یه کار خیلی زشتیه! چون به ازای همه propertyها اجرا میشه.ولی وقتی اونو تو جای اصلیش که همون بالای repeat password هست میزارم کار نمیکنه.
یه جورایی مشکل از اینه که نمیشه انگار مقدار property رو به این سمت پاس داد.ولی در حالتی که بالای کلاس میزارمش چون خود کلاس رو پاس میده ، طبیعتا
اسم propertyها رو هم پیدا میکنه.
ممنون میشم راهنماییم کنید .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۳:۴۵

به این ترتیب باید پیاده سازی بشه. یک حالت عمومی است و به کلاس و شیء خاصی گره نخورده:

```
using System;
using System.ComponentModel.DataAnnotations;
namespace Test
{
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
    public class CompareAttribute : ValidationAttribute
    {
        public CompareAttribute(string originalProperty, string confirmProperty)
        {
            OriginalProperty = originalProperty;
            ConfirmProperty = confirmProperty;
```

```
        }
        public string ConfirmProperty { get; private set; }
        public string OriginalProperty { get; private set; }
        protected override ValidationResult IsValid(object value, ValidationContext ctx)
        {
            if (value == null)
                return new ValidationResult("لطفا فیلدها را تکمیل نمائید");
            var confirmProperty = ctx.ObjectType.GetProperty(ConfirmProperty);
            if (confirmProperty == null)
                throw new InvalidOperationException(string.Format("لطفا فیلد {0} را تعریف نمائید",
ConfirmProperty));
            var confirmValue = confirmProperty.GetValue(ctx.ObjectInstance, null) as string;
            if (string.IsNullOrWhiteSpace(confirmValue))
                return new ValidationResult(string.Format("لطفا فیلد {0} را تکمیل نمائید",
ConfirmProperty));
            var originalProperty = ctx.ObjectType.GetProperty(OriginalProperty);
            if (originalProperty == null)
                throw new InvalidOperationException(string.Format("لطفا فیلد {0} را تعریف نمائید",
OriginalProperty));
            var originalValue = originalProperty.GetValue(ctx.ObjectInstance, null) as string;
            if (string.IsNullOrWhiteSpace(originalValue))
                return new ValidationResult(string.Format("لطفا فیلد {0} را تکمیل نمائید",
OriginalProperty));
            return originalValue == confirmValue ? ValidationResult.Success : new
ValidationResult("مقادیر وارد شده یکسان نیستند");
        }
    }
}
```

نویسنده: iman
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۴:۲۸

مثل همیشه کامل و بی نقص
عالی بود.ممنونم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۸:۲

خواهش می‌کنم.
لطفا از این پس مطالبی رو که در قسمت نظرات عنوان می‌کنید متناسب با عنوان موضوع جاری آن (برای مثال context per request در اینجا) باشد. با تشکر.

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۰:۲۷

سلام؛
من از وهله سازی نوع HybridHttpOrThreadLocalScoped در یک برنامه ویندوزی استفاده کردم.اما فکر میکنم چون حالت ThreadLocal رو برای این حالت انتخاب میکنه ، وهله مربوط به اشیاء ساخته شده تا زمان بستن برنامه در حافظه باقی بمونه.از این رو فکر میکنم چون وهله ساخته شده از بین نمیره ، کانکشن به DataBase تا زمان بسته شدن برنامه باز بمونه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۰:۴۸

توضیحات بالا برای برنامه‌های وب بهینه شده. در آنجا در Application_EndRequest به صورت خودکار کانکشن بسته میشه (با کد ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects که نوشته شد).
در برنامه‌های ویندوزی این مدیریت رو باید خودتون دستی انجام بدید و چنین مکانیزمی در آن طراحی نشده.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۶:۱۶

سلام آقای نصیری . خسته نباشید
همونطور که منو توی یکی از کامنت‌ها به اینجا ارجاع داه بودین، مطلب رو خوندم. ولی متاسفانه کمی گیج شدم. من توی پروژه ام از روش EF Code First استفاده نکردم. روش من اینجوری بود که دیتابیس رو ساختم و بعد از روی اون فایل .edmx رو ساختم و باهاش توی کلاس هام کار کردم. حالا میخواستم بدونم آیا من می‌تونم با توجه به این موضوع از توضیحاتی که در بالا گفتین استفاده کنم یا حتما باید EF Code First باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۶:۳۷

طراحی فوق برای روش database first نیست و بحث اینجا متفاوت است. اینترفیس‌های تعریف شده مطلب فوق و زیرساخت آن متفاوت است با database first.
ضمن اینکه از ef code first برای کار با بانک اطلاعاتی موجود هم می‌شود استفاده کرد. روش و ابزار مهندسی معکوس آن وجود دارد: (^)

نویسنده: میثم
تاریخ: ۱۳۹۱/۰۴/۱۶ ۲۲:۵۷

سلام و خسته نباشید استاد نصیری

سوال بنده اینه که استفاده از لایه سرویس ضرورتی داره ؟ مشکلی پیش میاد اگه این لایه رو به طور کامل خذف کنیم و مستقیم با UnitOfWork کار کنیم؟

چون با استفاده از این لایه به ازای یک عمل مانند ذخیره کردن یک شی در پایگاه داده باید 2 شی (یکی از لایه مدل و دیگری از لایه سرویس) تعریف بشه ضمن آنکه وقتی تعداد کلاس‌ها زیاد بشه متد initStructureMap() پیچیده میشه

تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۶ ۲۳:۴۰

منهای تمام مباحثی که عنوان شد، یکی دیگر از مزایای استفاده از لایه سرویس، جدا سازی منطقی قسمت‌های مختلف برنامه از هم است. به این ترتیب الان شما دقیقا می‌دونید اعمال کار با یک موجودیت دقیقا در کدام کلاس قرار گرفته و مرتبا در قسمت‌های مختلف برنامه پراکنده و تکرار نشده. اگر مشکلی وجود داشته باشد، در یکجا باید اصلاح شده و اثرش به صورت خودکار به تمام برنامه اعمال می‌شود.

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۴/۲۳ ۱۴:۵۸

سلام مهندس
توی پیاده سازی قسمت MVC شما کد زیر توی فایل global فراخوانی کردین

```
private static void initStructureMap()
        {
            ObjectFactory.Initialize(x =>
            {
```

```
            x.For<IUnitOfWork>().HttpContextScoped().Use(() => new Sample07Context());
            x.ForRequestedType<ICategoryService>().TheDefaultIsConcreteType<EfCategoryService>();
            x.ForRequestedType<IProductService>().TheDefaultIsConcreteType<EfProductService>();
        });
        //Set current Controller factory as StructureMapControllerFactory
        ControllerBuilder.Current.SetControllerFactory(new StructureMapControllerFactory());
    }
```

اینجا تعداد Entityهای ما از قبل ثابت و مشخصه اگر خواستیم به این لیست Entityهای جدیدی اضافه بشه چکار باید بکنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۳ ۱۶:۳۵

امکان اسکن اسمبلی‌های اضافه شده به سیستم هم وجود دارد: (^)

نویسنده: میثم
تاریخ: ۱۳۹۱/۰۴/۲۴ ۱۳:۱۰

سلام و خسته نباشید استاد
آیا نحوه استفاده که برای asp.net MVC و asp.netمعرفی نمودید کامل است ، یعنی با ناتمام ماندن یک درخواست از سمت کلاینت (به هر دلیلی) منابع به طور کامل آزاد میگردند و یا نیاز است تا با استفاده از HttpModule اشیا نابود بشوند.
شبیه چیزی که تو NH انجام میدادیم یا اینجا ذکر شده است
همیشه دعایتان می‌کنم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۴ ۱۳:۳۲

HttpModule همان اعمال قابل تنظیم در global.asax را در اختیار شما قرار می‌دهد. فقط اینبار کدهای شما اندکی نظم بهتری پیدا می‌کنند؛ به علاوه تعدادی روال رخدادگردان بیشتر نیز دراختیار شما خواهند بود.
در global.asax.cs دارید:

```
protected void Application_BeginRequest
```

در یک ماژول خواهید داشت:

```
public void Init(HttpApplication context)
{
    context.BeginRequest += beginRequest;
}
```

و الی آخر.

نویسنده: بابک
تاریخ: ۱۳۹۱/۰۴/۳۱ ۳:۲۶

در این روش شما نحوه ویرایش رکورد را چطور انجام می‌دهید؟ می‌خواهم در متد add در صورتیکه رکورد موجود باشد update شود و اگر نباشد در دیتابیس اینزرت شود یا اینکه 2 متد جدا به اسم add. edit در لایه سرویس داشته باشم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۳۱ ۸:۳۷

این یک سری به هم پیوسته است.
در مورد add or update
در مورد نحوه صحیح به روز رسانی اطلاعات و اشتباهات متداول مرتبط

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۴/۳۱ ۱۴:۱

ممنون از پاسخ شما
اینجا هم نمونه خوبی از Scan گذاشته

نویسنده: بابک
تاریخ: ۱۳۹۱/۰۵/۰۱ ۰:۱۶

من از EF 4.3.1 استفاده می‌کنم ولی متد AddOrUpdate ندارد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۱ ۰:۲۱

در فضای نام System.Data.Entity.Migrations قرار دارد.

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۰:۴۷

سلام
مهندس نصیری این لینک یه نگاه بندازین بد نیست ظاهرا که کامل کار کرده روی موضوع شما

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۰:۵۰

اینجا هم نمونه خوبی از Scan کردن اسمبلی‌ها گذاشته

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱:۰

مربوط است به db first و این مشکلات را دارد:
- کلاس واحد کار رو استاتیک تعریف کرده. این مورد در یک برنامه asp.net یعنی به اشتراک گذاری واحد کار جاری با تمام کاربران سایت.
- از StructureMap استفاده کرده اما چون درک درستی از تزریق وابستگی‌ها نداشته از الگوی service locator آن (ObjectFactory.GetInstance) برای وهله سازی استفاده کرده (از این مورد فقط در حالت‌های ناچاری مانند تهیه یک role provider سفارشی که وهله سازی آن در کنترل ما نیست و راسا مدیریت می‌شود باید استفاده کرد)
- از StructureMap استفاده کرده اما نمی‌دونسته که این کتابخانه خودش می‌تونه در پایان درخواست‌های وب اشیاء مورد استفاده رو dispose کنه و کار اضافی انجام داده.
و ....

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۹:۲۴

ممنونم مهندس

مثل همیشه کامل و بی نقص

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۵/۲۶ ۱۱:۴۷

من همین پیاده سازی رو انجام دادم و در متد Seed هم دیتای اولیه رو قرار دادم ولی هر دفعه حین اجرای برنامه این متد فراخوانی میشه و دیتای تکراری وارد بانک میکنه، میشه جلوی این کار رو گرفت یا نه، مگر این تنظیم

```
void Application_Start(object sender, EventArgs e)
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample07Context,
Configuration>());
```

برای بروز کردن بانک نیست

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۶ ۱۱:۵۲

به این ترتیب طراحی شده. نظر یکی از اعضای تیم EF در این مورد: (^). اگر می‌خواهید رکورد تکراری ثبت نشود از متد AddOrUpdate استفاده کنید.
استفاده‌ای که من از متد seed می‌کنم در عمل، تعریف قیودی مانند unique است با sql نویسی (داخل try/catch البته).

نویسنده: عرفان رضایی
تاریخ: ۱۳۹۱/۰۶/۰۷ ۱۲:۱۲

سلام آقای نصیری،
یه سوال داشتم ازتون:
در عمل ما تو یه برنامه‌ی وب (مثلا mvc) به متدهای زیادی فقط برای سرویس دهی به موضوعات احتیاج داریم،مثلاً
getTopCategories
getLastCategories
(getCategoryByID(int Id
(getCategoriesByDate(Datetime date
()getProductCategories
و ...
حالا سوالم اینه پیاده سازی‌های این متدها باید تو کدوم لایه انجام بشه؟
-1مثلا باید لیست موضوعات،(همین متد GetAllCategories مربوط به سرویس شما)، رو از سرویس برگردوند و داخل controller کویری‌های Linq رو روش اجرا کرد و اطلاعاتی که میخوایم رو ازش بکشیم بیرون و نمایش داد؟
-2یا باید توی اینترفیس ICategoryService **تک تک متد هایی که احتیاج داریم** رو تعریف و بعد توی EFCategoryService تو لایه سرویس اونارو پیاده سازی کنیم و فقط نتیجه رو به controller برگردوند و ازش استفاده کرد(یعنی تو controller فقط پارامترهای مورد نیاز متدهای لایه سرویس رو بهش پاس کنیم)؟
امیدوارم مفهوم رو رسونده باشم.
ممنون به خاطر زحمت هایی که می‌کشید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۰۷ ۱۲:۱۸

حالت دوم صحیح است. تمام پیاده سازی‌ها باید در لایه سرویس باشند. استفاده نهایی در یک کنترلر یا code behind و امثال آن.

نویسنده: عرفان رضایی

تاریخ: ۱۳۹۱/۰۶/۰۷ ۱۲:۲۵

واقعاً ممنونم از سرعتتون

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۰:۱۸

سلام آقای نصیری،
شما تو این مقاله گفتید که:
"همچنین نیازی به پیاده سازی متد SaveChanges نیست؛ زیرا پیاده سازی آن در کلاس DbContext قرار دارد."
ولی این متد رو تو اینترفیس IUnitOfWork ذکر کردید،اینجوری که اگه پیاده سازیه این متد رو انجام ندیم ارور میده!

بالاخره این متد باید توی اینترفیس IUnitOfWork ذکر بشه و توی Context هم پیاده سازی بشه
یا
توی اینترفیس IUnitOfWork ذکر نشه که در اینصورت نیاز باشه توی Context هم پیاده سازی بشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۰:۳۷

- پیاده سازی متد SaveChanges در کلاس پایه DbContext که توسط تیم EF ارائه شده، انجام شده و وجود دارد.
- ذکر یک متد در اینترفیس (یک قرار داد) به جهت امکان استفاده از آن است. شما نهایتا با متدهای تعریف شده در طراحی IUnitOfWork در لایه سرویس قرار است کار کنید نه مستقیما با کلاس مشتق شده از DbContext.
زمانیکه می‌نویسید:

```
public class MyContext : DbContext, IUnitOfWork
```

چند کار با هم انجام می‌شود:
MyContext به صورت خودکار امکان دسترسی به متد SaveChanges موجود در DbContext را پیدا می‌کند. کتابخانه StructureMap می‌تونه زمانیکه نیازی به یک وهله پیاده ساز IUnitOfWork بود، از MyContext استفاده کنه. همچنین چون الان SaveChanges با امضایی که در اینترفیس IUnitOfWork وجود دارد در کلاس MyContext هم قابل دسترسی است، نیازی به پیاده سازی مجدد آن نیست.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۰:۴۴

آخه برای بنده ارور میده که این متد حتماً باید پیاده سازی بشه!
منظور شما این نیست که باید تو پیاده سازی متد save changes اینترفیس IUnitOfWork فقط کد زیر رو بنویسیم؟

```
Return Base.SaveChanges();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۰:۵۱

کدهای کامل و قابل کامپایل مورد نظر این قسمت از این آدرس قابل دریافت است. می‌تونید وقت بذارید و با کدهای خودتون مقایسه‌اش کنید.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۱

ببخشید آقای نصیری من این کدا رو دیدم،یه سوالم پیش اومد،اینکه همونطور که اینجا عنوان شد ما باید پیاده سازی اینترفیس IUnitOfWork رو تو کلاس MyDbContextBase انجام بدیم،حالا پیاده سازی متد SaveChanges اینترفیس IUnitOfWork تو اونجا باید به صورت زیر باشه؟

```
try
{
applyCorrectYeKe();


auditFields();


//and another methods ...


Return base.SaveChanges();
}
catch (DbEntityValidationException validationException)
{
   //...
}
catch (DbUpdateConcurrencyException concurrencyException)
{
   //...
}
catch (DbUpdateException updateException)
{
   //...
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۱۹

بله. اگر این‌ها رو بخواهید با هم ترکیب کنید همین شکلی است.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۲۵

نمیدونم چطور خوبیتونو جبران کنم ...
فقط میتونم بگم ممنونم ...

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۳۹

بازم ببخشید آقای نصیری،
در اینصورت به پیاده سازی متد SaveChanges اینترفیس IUnitOfWork باید **Overrides** هم اضافه بشه که کامپایلر بدونه متد SaveChanges اینترفیس IUnitOfWork متد SaveChanges کلاس DbContext رو تحریف کرده ،درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۴۳

اگر این متد، return base.SaveChanges را بر می‌گرداند نیازی به ذکر override نیست و می‌تونید برای متد SaveChanges حتی پارامتر هم تعریف کنید (البته اینترفیس هم باید به همین شکل اصلاح شود):

```
public int SaveChanges(string userName, bool updateAuditFields = true)
```

این شکلی است که من خودم ازش استفاده می‌کنم.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۱:۵۳

به خدا گیج شدم،این هشدارها رو برای من میده:

```
Warning   1   function 'SaveChanges' shadows an overridable method in the base class 'DbContext'. To
override the base method, this method must be declared 'Overrides'.
```

در صورتیکه شما گفتید "اگر این متد، return base.SaveChanges را بر می‌گرداند نیازی به ذکر override نیست"!

```
Warning 2   Function 'SaveChanges' doesn't return a value on all code paths. Are you missing a 'Return'
statement?
```

اینم پیاده سازی متد SaveChanges اینترفیس IUnitOfWork من هستش:

```
Public Function SaveChanges() As Integer Implements IUnitOfWork.SaveChanges
            Try
                ApplyCorrectYeKe()
                'auditFields()
                Return MyBase.SaveChanges()
            Catch validationException As DbEntityValidationException
                '...
            Catch concurrencyException As DbUpdateConcurrencyException
                '...
            Catch updateException As DbUpdateException
                '...
            End Try
        End Function
```

به نظر شما مشکل چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۲:۰

- من تمام مطالبی رو که اینجا عنوان کردم در مورد سی شارپ بود و الان در کارهای خودم دارم ازش استفاده می‌کنم. نمونه قابل کامپایل هم در سایت گذاشتم که لینکش رو دادم.
- این متد SaveChanges آخری با امضای جدید آن، دیگر متد SaveChanges کلاس پایه رو مخفی نمی‌کنه. به همین جهت نیازی به override نداره. بحث من در این مورد بود. نهایتا شما قراره با IUnitOfWork کار کنید. نام این متد رو اصلا تغییر بدید به ApplyChanges بعد هم داخل آن کارهای خودتون رو قرار بدید و دست آخر return base.SaveChanges بازگشت داده شود.
ضرورتی ندارد حتما در این اینترفیس از نام SaveChanges استفاده شود. این یک انتخاب بود، بر اساس قسمت 12 جاری که ترکیبی نیست از چند قسمت دیگر. به این صورت می‌شد مبحث رو ساده‌تر و طبیعی‌تر توضیح داد.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۲۲:۶

ظاهراً تو VB باید **Overrides** هم اضافه بشه.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۵:۴۵

بنظرتون میشه این قسمتی که مربوط به StructureMap هست رو بیرون از پروژه سرویس نگهداری کرد یا نه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۶:۱۶

در مثال (^) فوق، تنظیمات StructureMap فقط در فایل Global.asax.cs برنامه‌های MVC و WebForms تعریف شدند نه در لایه سرویس.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۶:۴۱

ممنون از جوابتون، تو نگاه به این پروژه بنظرم اومد شاید بشه این تنظیمات رو توی یک پروژه جداگانه نگهداری کرد!

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۰۶/۲۶ ۱۱:۴۰

سلام؛
کد زیر که درون IUow تعریف شده آیا برای NH هم قابل استفاده است؟ آیا مستقل از Orm است؟

```
IDbSet<TEntity> Set<TEntity>() where TEntity : class;
```

درون Wpf من نیاز به خاصیت Local دارم که ازنوع ObservableCollection است . آیا درست است که از لایه Service این نوع را برگردانم ؟ آیا برای پیاده سازی NH نیز این قابل استفاده است ؟ آخه من درون Vm مجبورم

```
_uow.Set<Person>().Local
```

استفاده کنم.
در صورت امکان راهنمایی کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۱۲:۳

- IDbSet یک اینترفیس است؛ پیاده سازی نیست. اینترفیسی به همین نام را برای هر ORM دلخواه دیگری طراحی و پیاده سازی کنید. کدهای شما قابل انتقال خواهند بود.
- بله. دسترسی به خاصیت Local داخل لایه سرویس صورت گرفته و کپسوله شده. به همین جهت امضای متدی که ارائه شده به هر ORM دیگری نیز قابل انتقال است. فقط در آنجا یک تبدیل لیست به ObservableCollection را در پیاده سازی داخلی لایه سرویس خود خواهید داشت. اما استفاده کننده نهایی فقط با اینترفیس و قراردادهای تعریف شده در آن هست که کار می‌کند و کاری به جزئیات پیاده سازی لایه سرویس شما ندارد.
به همین جهت است که در اینجا کار با اینترفیس‌ها و قراردادها ترویج شده؛ تا جزئیات پیاده سازی لایه سرویس از دید استفاده کننده مخفی باقی بماند.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۰۶/۲۶ ۱۴:۵

متشکرم آقای نصیری.
الان حدود 3 ماه میشه هر روز به سایتتون سر میزنم، این مطالب منو بیشتر به برنامه نویسی علاقه مند کرد. مطالب عالی شما .
نمی‌دونم چطوری تشکر کنم از شما، ولی می‌دونم که شناسایی حق در حق شناسیست ، ممنون.

نویسنده: رضا

تاریخ: ۱۳۹۱/۰۷/۰۱ ۱۱:۳۰

آقای نصیری من دوتا سوال داشتم:
اول اینکه معادل دستور زیر، در صورتی که بخواهیم از این روش استفاده کنیم چه چیزی میشود؟

```
db.Entry(post).State = EntityState.Modified;
```

و سوال بعدیم اینکه برای DI چه کتابخوانه ای رو توصیه میکنید؟ همین StructureMap یا Ninject و ...؟
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۱ ۱۲:۱۵

- نیاز خواهید داشت یک چنین تعریفی را اضافه کنید:

```
public interface IUnitOfWork
{
   //...
   DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;
}
```

+ در این حالت این IUnitOfWork آنچنان قابل انتقال و تعویض نخواهد بود چون DbEntityEntry کلاس خاصی است در EF و در سایر ORMها معادلی ندارد. اما اگر فقط با EF کار می‌کنید و قصد تعویض ORM را ندارید، این روش کار می‌کنه و مناسب است.
- از این موارد زیاد است. فرقی هم نمی‌کند (سرعت هم به تنهایی ملاک نیست). با هر کدام که راحت هستید، همان مطلوب است.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۷/۰۵ ۹:۵۱

با سلام و خسته نباشید
آیا برای مسائلی نظیر transaction commit و transaction rollback در الگوی UOW راهی وجود داره؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۰:۳

طراحی EF Code first مبتنی است بر روان بودن و سادگی؛ هر چند پشت صحنه ساده‌ای ندارد. هر وهله از DbContext به صورت خودکار یک تراکنش را تشکیل می‌دهد و در زمان بسته شدن و dispose، این تراکنش را commit می‌کند (همچنین متد SaveChanges در پشت صحنه از تراکنش‌ها بهره می‌گیرد). هر استثنایی این بین رخ دهد، تراکنش rollback خواهد شد. به همین جهت الگوی واحد کار مطرح شده تا تعداد وهله‌های زیادی از DbContext هربار ایجاد نشوند و کل عملیات در یک DbContext یا یک تراکنش انجام شود.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۰:۵۱

ممنون از پاسخ کاملتون.
فقط برای روشنتر شدن موضوع این سوال را میپرسم.
پس یعنی وقتی که ما در پیاده سازی متد یک Interface ، درج و حذف و بروزرسانی‌های مختلفی در آن متد داریم و در نتیجه چند بار از saveChanges استفاده میکنیم.

در این حالت کلیه این عملیات در قالب یک transaction به حساب می‌آید و در صورت بروز استثنا ، کل عملیات داخل متد rollback خواهد شد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۱:۳۶

خیر. به ازای هر SaveChanges یک تراکنش خاتمه یافته و تراکنش جدیدی آغاز می‌شود (این موارد رو می‌تونید با SQL Server Profiler دقیقا مشاهده کنید).
+ ضرورتی ندارد در یک تراکنش، از چندین و چند SaveChanges استفاده کنید؛ از این جهت که EF از مکانیزم Tracking برخوردار است و می‌تواند با یک SaveChanges ، چندین و چند عملیات insert و update را (بهینه‌ترین حالتی را که محاسبه کرده) با هم در طی یک تراکنش بر اساس مواردی که ردیابی کرده، انجام دهد.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۱:۵۲

ممنون مهندس بابت پاسخ کاملتون
من ذهنم به سمت TransactionScope رفته بود و به Tracking در EF توجه نکرده بودم.
متشکرم.

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۷/۰۹ ۱۲:۵۱

با عرض سلام
چه موقع نیاز است از UnitOfWork در EF استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۹ ۱۳:۵

در تمام برنامه‌های واقعی که حالت مثال نداشته باشند.

نویسنده: kia
تاریخ: ۱۳۹۱/۰۷/۰۹ ۱۷:۲

سلام و ممنون از این سری codeFirst. یک سوال؟

چرا توی لایه **UI** مستقیم به لایه **DL دسترسی** پیدا کردین؟ فقط چون مثال هست اینکارو کردین؟
چون چیزی که هست گفته می‌شه اینکارو نکنین.
و در یک پروژه واقعی به چه صورت باید کار کرد؟ چجوری می‌شه UOW (که در DL هست) رو مخفی کرد از دید UI؟ یا اصلا همچین کاری باید کرد؟ (منظور اینکه راه درست استفاده در سمت UI وقتی که قراره با این دید بریم جلو به چه صورت است؟ چون در نهایت باید یک Container ای باشه ک موجودیتها در داخل اون باشن و اون کانتینر بتونه کارایی مثل Transaction رو انجام بده.
من روی WinForm الان می‌خواستم پیاده کنم که با این مسئله به مشکل برخوردم)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۹ ۱۷:۱۶

خیر. زمانیکه از EF استفاده می‌کنید، DAL همان EF است و تنظیمات آن.
پیشنهاد من این است که یکبار پروژه مربوطه رو دریافت کنید و بعد پروژه‌های DataLayer و همچنین ServiceLayer و غیره آن‌را بررسی کنید.
در این مثال‌ها فقط از اینترفیس‌های ServiceLayer (و نه DataLayer مجزای آن) به کمک ترزیق وابستگی‌ها در لایه نمایشی

استفاده شده.

نویسنده: kia
تاریخ: ۱۳۹۱/۰۷/۰۹ ۲۰:۵۵

اینکه DAL همان EF هست درست، من پروژه رو اجرا و بررسی کردم و مشکلی با روالها ندارم الانم مطالبی می‌خونم از این پست و کامنتاش و البته جاهای دیگه راجع به ابهاماتی که برام هست هنوز اما راستش هنوز جواب سوالمو نگرفتم:
شما می‌گین که

در این مثال‌ها فقط از اینترفیس‌های ServiceLayer (و نه DataLayer مجزای آن) به کمک ترزیق وابستگی‌ها در لایه نمایشی استفاده شده.
اما چرا (مثلا) در پروژه Console EF_Sample07 (یا همون لایه نمایشی **UI**) **رفرنسی به DataLayer** زده شده و از UOW که اینترفیسی در لایه DL هست استفاده شده؟ ایا اینکار یکسری قراردادها رو نقض نمی‌کنه؟ +

```
using EF_Sample07.DataLayer.Context;
```

```
_uow.SaveChanges();
```

درسته ک یکسری قرارداد هست این چیزا ولی هرچی خوندم بیشتر در مورد این بود که از سمت UI هیچ دسترسی ای به DL نباید باشه و UI با ServiceLayer یا BL در تعامل باید باشه.
مثلا در برنامه ای که بصورت nTier قراره اجرا بشه اینکار مشکل ساز خواهد بود و شاید اصلا مجوز قرارگیری DAL روی سیستمی که UI هست داده نشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۹ ۲۱:۱۲

- من در مورد الگوی مخزن در قسمت 11 این سری بحث کردم (کامنت‌های آن‌را هم بخوانید)؛ همچنین این مباحث در مورد EF Code first است و نه db first و نه EF 4. به علاوه این لینک‌هایی که مطرح کردید مثلا نمونه code project، داخل به اصطلاح BLL خودش، پر است از وهله سازی Context و من در این مطلب توضیح دادم که چرا اینکار غلط است و چگونه استفاده از یک تراکنش برای چندین عملیات مرتبط را زیر سؤال می‌برد.
- اون اینترفیس IUnitOfWork مطرح شده در مثال جاری، وابستگی خاصی به DataLayer نداره. می‌تونه در لایه سرویس هم تعریف بشه (منظور این است می‌تونه در یک اسمبلی و پروژه جداگانه هم قرار بگیره و مشکلی نیست). اما DataLayer باید بتونه در حین تزریق وابستگی‌ها وهله‌ای از IUnitOfWork رو فراهم کنه تا به اون معنا ببخشه؛ به همین جهت Context برنامه باید آن‌را پیاده سازی کند تا توسط StructureMap قابل شناسایی و استفاده شود.
اما نهایتا وهله سازی اینترفیس یاد شده توسط DAL صورت می‌گیره. uow به خودی خود موجودیتی نداره. در اینجا مثلا EF هست که به اون معنا می‌بخشه و سبب وهله سازی آن خواهد شد. هرچند به ظاهر برنامه با اینترفیس‌ها کار می‌کند اما تزریق وابستگی‌ها است که به این اینترفیس‌ها موجودیت می‌بخشد و سبب دسترسی به وهله‌ای که قرار داد ارائه شده توسط آن‌ها را پیاده سازی کرده می‌شود.
- در یک سیستم nTier هم مباحث ذکر شده در این قسمت، جاری است. مثلا یک WCF Service قرار گرفته روی یک سرور مجزا هم می‌تونه از DataLayer و ServiceLayer مثال جاری استفاده کند. استفاده کننده نهایی برای نمایش آن در UI با هیچکدام از دو مورد ذکر شده کاری ندارد و فقط با قراردادهای WCF Service کار می‌کنه.

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۷/۱۱ ۱۱:۳۲

با عرض سلام
با اجرای این sample خطای زیر رخ داد. علت خطای زیر چیست؟

CreateDatabase is not supported by the provider. **Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and .where it originated in the code

Exception Details: System.Data.ProviderIncompatibleException: CreateDatabase is not supported by the provider

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۱ ۱۱:۵۵

سطر «HibernatingRhinos.Profiler» را حذف کنید. اطلاعات بیشتر در اینجا .

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۷/۱۸ ۱۶:۳۱

با تشکر از مطالبتون
در هنگام دیباگ نسخه کنسول این مثال و ردگیری کوئری‌ها در EFProf به نظر میاد که کوئری‌های Insert درست بعد از uow.SaveChanges اجرا نمی‌گردند؛ بلکه این امر در انتهای کار و پس از از بین رفتن ابجکت uow صورت می‌پذیرد. قاعدتا با توجه به توضیحاتتان نباید این اتفاق می‌افتاد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۸ ۲۲:۴۸

علت این است که **یک** SaveChanges در اینجا تعریف شده (ضمنا زمان اجرای این‌ها مهم نیست. بحث ما در مورد تعدد تراکنش‌ها بود). چند سطر زیر را اضافه کنید بعد از سطر آخر و مجددا تست کنید:

```
//...
uow.SaveChanges();
//...
var product3 = new Product { Name = "P300", Price = 300 };
var category2 = new Category
{
                Name = "Cat200",
                Title = "Title200",
                Products = new List<Product> { product3 }
};
categoryService.AddNewCategory(category2);
uow.SaveChanges();
```

من دو تراکنش مجزا رو در برنامه مطمئن و تست شده SQL Server Profiler مشاهده می‌کنم (به ازای هربار فراخوانی .(SaveChanges

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۷/۱۹ ۰:۱۲

آقای نصیری این پیاده سازی (Context per request) مناسب برنامه‌های Silverlight هست ؟ یا لزومی دارد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۹ ۰:۱۹

ORMها کلا در سیلورلایت مستقیما قابل استفاده نیستند چون سیلورلایت سمت کاربر اجرا می‌شود و دسترسی کاملی هم به کل دات نت ندارد. سیلورلایت از طریق سرویس‌های WCF می‌تونه با سرور ارتباط برقرار کنه و این مباحث در سرویس‌های WCF هم قابل استفاده است.
البته برای سیلورلایت WCF RIA Services تعریف شده که روش مرجح است و در آن امکان دسترسی به EF Code first وجود دارد .

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۷/۱۹ ۷:۵۴

با سپاس
با SQL Server Profiler تست شد و نتیجه درست بود.
احتمالا مسئله بر می‌گرده به عدم آشنایی من با طرز کار دقیق EFProf.

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۱/۰۷/۲۶ ۲۱:۳۵

وقت بخیر مهندس نصیری. خسته نباشید.
یک سوال.
در لایه سرویس اگر یک عملیات مشترک باشد (به عنوان مثال درهم سازی (Hash) کلمه عبور کاربر) به نظر شما بهتر است در کجا قرار گیرد.
1) به عنوان مثال اگر در Ef.....Service قرار گیرد خیلی جالب، زیبا و مربوط نیست.
2) میشه در یک بخش دیگر (مثلا مشترک) قرار گیره، که خوب بازم مسئله اینه که این متد همیشه به بخش کاربران سرویس میده و عملا نباید جدا باشه.
3) میشه از یک کلاس میانی انتزاعی استفاده کرد و متدهای مشترک در تمام انواع سرویس (EF، Fake، و یا ....) در دسترس باشه.
ممنون میشم که راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۶ ۲۲:۳۲

من در تمام پروژه‌هام یک class library به نام MyProject.Common ایجاد می‌کنم برای قرار دادن توابعی که می‌تونه در پروژه‌های دیگر بدون وابستگی خاصی به پروژه جاری مورد استفاده قرار گیرد. دیدم جاهای دیگر اسمش رو گذاشتند Application framework من اسمش رو گذاشتم MyProject.Common. مثلا تابع SHA1 می‌تونه در چندین و چند پروژه بدون وابستگی خاصی استفاده شود و بین این‌ها مشترک است یا مثلا تابع فشرده سازی یک فایل هم به همین صورت و الی آخر.
سرویس‌های برنامه هم می‌تونند از این کتابخانه مشترک استفاده کنند و سرویس دهند.

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۱/۰۷/۲۷ ۰:۲۴

ممنون.

نویسنده: حسینی
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۱:۴۱

با سلام
از مطالب مفیدتون بینهایت سپاسگزارم.
من برای فهم بهتر این مطلب به نمونه ویندوزی اون احتیاج دارم.
با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۲:۱

نمونه ویندوزی آن برنامه کنسولی است که به همراه کد این قسمت ارائه شده: (^)

نویسنده: حسینی
تاریخ: ۱۳۹۱/۰۸/۰۵ ۱۵:۱۶

با سلام
از پاسختون ممنون
در برنامه ای که یکی از دوستان زحمت کشیده بود؛ علاوه بر اینترفیس هایی که به ازای هر کلاس تعریف کرده بودند؛ یک اینترفیس هم تعریف کرده بودند و تمام متدها رو در اون تعریف کرده بودند. همه اینترفیس‌ها از اون ارث بری کرده بودند و تمام متدها رو به اون منتقل کرده بودند. داخل خودشون هیچ متدی باقی نمونده بود.
-1 این روش مورد تائید شما میباشد؟
-2 تعریف اینترفیسی که در روش بالا عرض کردم به شکل زیر است:
منظور از IDisposable چیست؟
-3 در صورت تائید روش ذکر شده, چه لزومی به تعریف مابقی اینترفیس‌ها است در صورتی که همه EFClassها میتوانند مستقیما از اون اینترفیس ارث بری کنند.

```
public interface IGenericService<T>: IDisposable where
 T : class
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۵ ۱۶:۱۳

به صورت خلاصه: شما می‌تونید یک سری متد عمومی رو در یک base class جنریک هم قرار بدید و از اون ارث بری کنید. به این ترتیب حجم کد نویسی کمتری خواهید داشت. اما این چند متد عمومی پاسخگوی نیاز یک برنامه واقعی نیستند. به همین جهت نیاز است مابقی اینترفیس‌ها و کلاس‌ها هم به صورت مجزا تعریف شوند.

نویسنده: فریدون غلامی
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۰:۱۶

سلام؛
من اگر بخوام Structure map را در پروژه Windows Application استفاده بکنم باید چکار کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۰:۲۵

- لطفا نظرات این مطلب را یکبار مطالعه کنید. پیشتر به این سؤال پاسخ داده شده:
خلاصه آن: قسمت «استفاده از الگوی واحد کار و کلاس‌های سرویس تهیه شده در یک برنامه کنسول ویندوزی» عنوان شده در مطلب فوق، یک برنامه ویندوزی است. سورس کامل این سری هم در دسترس است (لینک داده شده در پایان مطلب). شبیه به برنامه‌های وب که یک سری روال مانند شروع و پایان درخواست را دارند، در اینجا شروع یک فرم، پایان یک فرم، شروع و پایان مثلا یک کلیک را دارید.

نویسنده: فریدون غلامی
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۰:۳۱

بله شما درست می‌فرمایید ولی شما یه جا فرمودید که باید دستی کانکشن را از بین برد.
این چه طوری هستش و باید در کدام قسمت انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۰:۵۷

بحث ASP.NET متفاوت است با Windows Forms یا WPF. در برنامه‌های وب یک زیر ساخت آماده برای آغاز و پایان درخواست‌ها و مدیریت خودکار این مسایل وجود دارد. معادل آن مثلا در یک برنامه مبتنی بر MVVM، مدیریت طول عمر یک context در طول عمر ViewModel برنامه است. علت این مساله هم به stateless بودن برنامه‌های وب و state-full بودن برنامه‌های ویندوزی بر می‌گردد. در یک برنامه وب در پایان درخواست، تمام اشیاء یک فرم در سمت سرور تخریب می‌شوند. اما در یک برنامه ویندوزی تا زمانیکه یک فرم باز است، اشیاء آن تخریب نخواهند شد. بنابراین مدیریت context در برنامه‌های ویندوزی «دستی» است. در زمان شروع فرم context شروع خواهد شد، زمان تخریب/بستن آن، با بستن یا dispose یک context، خودبخود اتصالات هم قطع خواهند شد.
در متد Program.Main هم می‌تونید تنظیمات اولیه ObjectFactory رو قرار بدید (شبیه به Application_Start برنامه‌های وب).
بنابراین در برنامه‌های وب «context/session per http request» داریم؛ در برنامه‌های ویندوزی «context per operation or per form». یعنی می‌تونید بسته به معماری برنامه ویندوزی خود، context را در سطح یک فرم تعریف کنید و مدیریت؛ و یا در سطح یک عملیات کوتاه مانند یک کلیک. تمام مباحث uow.SaveChanges ، ObjectFactory.GetInstance و یا Dispose آن هم دستی است و زیر ساختی برای مدیریت خودکار آن‌ها همانند برنامه‌های مثلا ASP.NET MVC وجود ندارد. حداکثر اینکه یک سری base class را شبیه به مثال Web forms زده شده تهیه کنید، تا میزان کدهای تکراری را کاهش بدید.

نویسنده: فریدون غلامی
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۱:۲

واقعا ممنون از توضیحات جامعتون

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۱/۱۰/۲۸ ۲۳:۵۰

وقت بخیر
یک سوال
اگر من به دلایلی لازم باشه از چند DbContext استفاده کنم (فرض کنید یکی برای اطلاعات اصلی و یکی برای فایل‌ها و ...) در این حالت به چه شکلی می‌توان از این الگو استفاده کرد؟
آیا لازم است که چند نوع UOW ایجاد شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۹ ۰:۱۳

بله. به ازای هر DbContext یک سری تنظیمات مجزا نیاز است. ردیابی تغییرات در یک context کار می‌کند. همچنین مباحث migrations هم به ازای یک context عمل خواهند کرد.

نویسنده: حسین
تاریخ: ۱۳۹۱/۱۰/۲۹ ۹:۲۷

سلام. من توی برنامه WPF MVVM ای که نوشتم یه Context توی هر ViewModel ایجاد می‌کنم میکنم و در پایان توی بستن ویومدل Context رو Dispose می‌کنم. قبلاً از using برای مدیریت اتصال به دیتابیس استفاده می‌کردم ولی وقتی از using استفاده می‌کنم دیگه تغییراتی که اعمال می‌کنم (حذف، ویرایش، افزودن) UI متوجه نمیشه.
میخواستم بدونم این مشکل با استفاده از الگوی Context Per Request حل میشه یا نه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۹ ۹:۴۸

خیر. این الگو خارج از توضیحات مطلب فوق در مورد «اهمیت بکارگیری الگوی Unit of work و به اشتراک گذاری آن در طی یک درخواست» کار دیگری را انجام نمی‌دهد.

نویسنده: فریدون غلامی
تاریخ: ۱۳۹۱/۱۱/۰۳ ۹:۹

سلام

structureMap را در پروژه ویندوزی Add کردم اما خطایی زیر را دارد:

Error 36 The type or namespace name 'StructureMap' could not be found

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۳ ۹:۱۹

قبلا بحث شده . لطفا نظرات را یکبار کامل مطالعه کنید.

نویسنده: علی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۶:۳۰

سلام، در مثالی که برای MVC ذکر کردید، آیا امکان داره که تزریق وابستگی به این صورت زیر انجام بشه:

```
public static class DataFactory
{
    public static IUnitOfWork UnitOfWork
    {
        get { return ObjectFactory.GetInstance<IUnitOfWork>(); }
    }

    public static ICategoryService CategoryService
    {
        get { return ObjectFactory.GetInstance<IUnitOfWork>(); }
    }

    public static IProductService ProductService
    {
        get { return ObjectFactory.GetInstance<IUnitOfWork>(); }
    }
}

...

 public HomeController()
 {
 _productService = DataFactory.ProductService;
 _categoryService = DataFactory.CategoryService;
 _uow = DataFactory.UnitOfWork;
 }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۷:۲۳

خیر. برنامه‌های وب چند کاربری هستند. ProductService *استاتیک* یعنی به اشتراک گذاری یک وهله بین تمام کاربران سایت .

نویسنده: علی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۲۱:۲۵

ممنون، ولی مگر

```
ObjectFactory.GetInstance<IProductService>()
```

هربار یک وهله از کلاس پیاده سازی شده‌ی آن (ProductService) ارائه نمی‌ده؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۶ ۹:۱۳

بله. مسایل همزمانی رو چطور مدیریت می‌کنید؟ زمانی که یک وهله استاتیک در اختیار برنامه قرار دادید آیا می‌تونید تضمین کنید که از بین مثلا 100 نفری که دارند از سایت استفاده می‌کنند، هیچکدام به صورت اتفاقی در آن واحد به همان وهله استاتیک دریافتی دسترسی پیدا نمی‌کنند؟ این وهله به اشتراک گذاشته شده می‌تونه اطلاعات مدیریتی باشه که نباید در اختیار یک کاربر با سطح دسترسی معمولی قرار بگیره.
ضمن اینکه در EF وهله DbContext به صورت Thread safe طراحی نشده و امکان به اشتراک گذاری آن بین چندین ترد وجود ندارد. به ازای هر ترد باید یک وهله جداگانه از آن تهیه شود تا شاهد تخریب اطلاعات نباشید.

نویسنده: سجاد
تاریخ: ۱۳۹۱/۱۱/۰۶ ۱۳:۲۷

با سلام؛
با این شرایط باید متد های add ,edit,delete ,.... رو در لایه سرویس برای همه‌ی کلاس‌ها بصورت جداگانه تعریف کرد امکانش وجود ندارد که لایه سرویس مون رو به صورت جنریک برای همه کلاس هامون داشته باشیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۶ ۱۳:۴۸

خیر. هستند یک سری الگوی مخزن عمومی به این شکل که در قسمت 11 سری EF نقد شدند و دارای مشکلات زیادی بوده که نیازی به تکرار آن در اینجا نیست. به علاوه دنیای واقعی با چند مورد متد ساده عمومی مدل نمی‌شود. عموما جمع چند عملیات هست که در قالب یک متد مشخص، خروجی یک سرویس را تشکیل می‌دهد. این عملیات هم می‌تواند مرتبط به چندین موجودیت باشد در آن واحد. تمام این موارد باید به صورت بسته بندی شده در قالب یک متد در اختیار لایه‌های دیگر قرار گیرد.

نویسنده: نوید
تاریخ: ۱۳۹۱/۱۲/۱۰ ۲۳:۴۱

با سلام
من در حین اجرای نمونه کدهای این مقاله در بخش MVC به خطای Value Cannot be null در کلاس

```
public class StructureMapControllerFactory : DefaultControllerFactory
    {
        protected override IController GetControllerInstance(RequestContext requestContext, Type
controllerType)
        {
            return ObjectFactory.GetInstance(controllerType) as Controller;
        }
    }
```

مواجه شدم که با اضافه کردن :

```
routes.IgnoreRoute("{*favicon}", new { favicon = @"(.*/)?favicon.ico(/.*)?" });
```

به متد Register Route برطرف شد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۱ ۰:۶

برای استفاده در شرایط واقعی:

```
public class StructureMapControllerFactory : DefaultControllerFactory
    {
        protected override IController GetControllerInstance(RequestContext requestContext, Type
controllerType)
        {
            if (controllerType == null)
                throw new InvalidOperationException(string.Format("Page not found: {0}",
requestContext.HttpContext.Request.Url.AbsoluteUri.ToString(CultureInfo.InvariantCulture)));
            return ObjectFactory.GetInstance(controllerType) as Controller;
        }
    }
```

نویسنده: Masoud
تاریخ: ۱۳۹۱/۱۲/۲۸ ۲۱:۲۲

سلام

با توجه به گفته دوستمون (بنابر این باید برای هر ماژول dll ای تولید کرد که حاوی DomainClassها , ServiceLayerها ، Controllerها و DbContext مربوط به اون ماژول باشه)

اگر از این الگو برای طراحی نرم افزار استفاده شود خوب برای ارتباط بین ماژول‌ها که باید رفرنس همدیگر را در خود درج نمایند و در این صورت با circular dependency روبه رو میشویم

برای جلوگیری ای این خطا چه راه حلی پیشنهاد میدهید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۸ ۲۱:۳۷

- من هر نوع طراحی رو تائید نمی‌کنم. چرا یک برنامه باید چندین DbContext داشته باشد؟ نیازی نداره. چرا باید چندین ماژول کنترلر داشته باشه؟
- سؤال شما خارج از موضوع بحث است (در اینجا بحثی در مورد طراحی «افزونه پذیر» مطرح نشده). برای طراحی افزونه پذیر می‌تونید به مباحث زیر مراجعه کنید:
ابتدا فقط و فقط یک DbContext مرکزی را در کل برنامه تعریف کنید. بعد تنظیمات نگاشت‌ها را به صورت پویا یافته و به آن اضافه کنید. سپس موجودیت‌های مهیا را به صورت پویا یافته و به Context مرکزی اضافه نمائید.
+ در EF نمی‌تونید در عمل چندین DbContext داشته باشید مرتبط با یک دیتابیس. Change tracking در EF بر مبنای یک DbContext کار می‌کند. اگر قرار باشد چندین وهله از DbContextهای مختلف مثلا در طی یک درخواست وجود داشته باشند، یعنی چندین اتصال باز شده به دیتابیس و چندین تراکنش مجزا در حال انجام است (کل بحث جاری از ابتدا). به علاوه قابلیت کار کردن با چندین موجودیت را به صورت همزمان در طی یک تراکنش از دست می‌دهید.
- برای اینکه در حین کار با Structure Map خطای Circular dependency را مشاهده نکنید، نیاز است یک کتابخانه یا حتی یک کلاس واسط طراحی کنید تا مشترکات در آن قرار گیرند.

نویسنده: behrouz
تاریخ: ۱۳۹۲/۰۱/۱۲ ۱۸:۳۲

با سلام
به نظر می‌رسد با توجه به معماری که ارائه دادید منطق سیستم (BLL) و کدهای دستکاری داده‌ها یعنی کد هایی که اعمال CRUD را در دیتابیس انجام می‌دهند (DAL) را در یک لایه به نام لایه سرویس قرار دادید
به نظر شما برای خوانایی بیشتر بهتر نیست این دو از یکدیگر جدا شوند؟

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۰۱/۱۲ ۱۸:۴۸

لایه‌ی Data Access در این معماری همان ORM مورد استفاده هست.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۱/۱۳ ۲:۵۸

نظر شما در مورد این پیاده سازی الگوی کار چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۳ ۹:۳۸

- ابتدای کار یک روش غلط رو شروع کرده، بعد اواسط کار اون رو نقد کرده و رد.
- چون از لایه سرویس استفاده نکرده کل منطق کار رو برده داخل کنترلرها.
- از تزریق وابستگی‌ها استفاده نکرده، بنابراین برخلاف شکلی که در ابتدای کار گذاشته، کنترلرهاش رو نمی‌تونه مستقل از یک سری کلاس کاملا مشخص، تست کنه.
- الگوی مخزنی که ارائه داده در این مثال ساده کار می‌کنه اما اگر قرار باشه با چند موجودیت کار کرد و نتیجه رو ترکیب، کارآیی خوبی نداره چون خیلی از قابلیت‌های ذاتی EF مثل کوئری‌های به تاخیر افتاده (deferred LINQ queries) در اینجا قابل پیاده سازی نیست. اگر هم بخوان این رو اضافه کنن باید به لایه مخزن خروجی IQueryable اضافه کنن که به یک طراحی نشتی دار خواهند رسید چون انتهای کار با خروجی IQueryable کاملا باز باقی می‌ماند (نمونه‌اش متد Get ایی است که طراحی کرده).
یا یکی دیگر از اهداف ظاهری لایه مخزن، امکان تعویض آن در صورت نیاز است و مثلا کوچ به یک ORM دیگر. دنیای واقعی: include ایی که اینجا تعریف شده فقط در EF وجود خارجی دارد یا یک سری از نکات دیگر بکار گرفته شده در این الگوی مخزن. (در قسمت 11 سری EF سایت بحث شده)
- در مثالی که زده باگ امنیتی وجود دارد. متد Updateاش به دلیل عدم استفاده از ViewModel آسیب پذیر هست. (در این مورد در سری MVC بحث شده)

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۱/۱۴ ۸:۴۱

سلام
عالی مثل همیشه.
مهندس شما فرمودین:
الگوی مخزنی که ارائه داده در این مثال ساده کار می‌کنه اما اگر قرار باشه با چند موجودیت کار کرد و نتیجه رو ترکیب، کارآیی خوبی نداره چون خیلی از قابلیت‌های ذاتی EF مثل کوئری‌های به تاخیر افتاده (deferred LINQ queries) در اینجا قابل پیاده سازی نیست. اگر هم بخوان این رو اضافه کنن باید به لایه مخزن خروجی IQueryable اضافه کنن که به یک طراحی نشتی دار خواهند رسید چون انتهای کار با خروجی IQueryable کاملا باز باقی می‌ماند (نمونه‌اش متد Get ایی است که طراحی کرده).

البته (البته چندین جای دیگه هم گفتین) در مورد نشتی حافظه، کاربرد IQueryable پس توی کدام لایه از کار ما می‌تونه باشه با توجه به انعطاف پذیری که به کار ما میده؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۰:۵۷

نشتی طراحی مد نظر بوده؛ نه نشتی حافظه. نشتی طراحی به این معنا است که اگر متد شما خروجی IQueryable داشته باشه، در لایه‌های دیگر می‌شود کلا مقصود آن‌را تغییر شکل داد به فرم دیگری که اصلا شاید هدف اولیه این نبوده (چون IQueryable یک عبارت است و نه اجرای یک فرمان). به همین جهت باید در لایه سرویس و بدنه متدهای آن از IQueryable استفاده شود و نهایتا این متدها باید IList یا IEnumerable را بازگشت دهند. به این ترتیب حد و مرز یک لایه مشخص می‌شود.

نویسنده: صابر فتح الهی

تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۱:۸

آخه بعضا دیده شده (مثلا متدی مانند GetAll) کل رکوردهارو به صورت یکجا از بانک واکشی می‌کنه، اما ما می‌خواهیم قسمتی از اونها واکشی بشه مثلا 20 رکورد اول، با این تفاسیر در صورتی که خروجی از نوع IList (یا هر نوعی شبیه این) باشه اون وقت یکبار واکشی میشه کل رکوردها و بعد متد ما روی اون عمل انتخاب انجام میده.
-1 ایا این باعث عدم کارایی نمیشه؟
-2خروجی نوع IQueryable کجا به کار ببریم؟
-3در کدام لایه تبدیل IQueryable به IList (یا انواع مشابه) باید انجام بشه.

معذرت دیگه زیادی دارم بحث کش میدم و می‌دونم اینجا جای پرسش و پاسخ نیست، بازم به بزرگواری و تجربتون من را ببخشید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۱:۱۸

لایه سرویس شما می‌تونه متد Paging دار هم داشته باشه. مثلا:

```
public IList<BlogPost> GetLatestBlogPosts(int pageNumber, int recordsPerPage = 4)
        {
            var skipRecords = pageNumber * recordsPerPage;
            return _blogPosts
                        .OrderByDescending(x => x.Id)
                        .Skip(skipRecords)
                        .Take(recordsPerPage)
                        .ToList();
        }
```

در این حالت در بدنه این متد لایه سرویس از IQueryable استفاده شده اما خروجی آن یک لیست مشخص است.

نویسنده: محسن
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۱:۱۹

به این مطلب مراجعه کنید.یه پیاده سازی کوچیکه که کلی از ابهامات را رفع میکنه.

نویسنده: عبدی
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۳:۸

سلام و تبریک سال نو

آقای نصیری نظر شما در مورد انتخاب بین IList یا IEnumerable را برای خروجی لایه سرویس می‌خوام بدونم. معمولا خودتون کدوم را استفاده و توصیه می‌فرمایید. ممنون میشم یه توصیه و توضیحی راجع به این مورد بدید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۳:۲۳

- این مساله در لابلای قسمت‌های مختلف سری EF در سایت بحث شده. قسمت 12 رو قسمت آخر تلقی کنید نه قسمت شروع.
- IList هم دقیقا از IEnumerable مشتق شده و یک سری قابلیت مانند افزودن آیتم به آن و همچنین دسترسی بر اساس ایندکس به آن اضافه شده.
- اگر در حین کار با خروجی لیستی یک متد، فراخوانی‌های بعدی نتیجه آن، فقط یکبار است، از IEnumerable استفاده کنید. اگر بیشتر از یکبار است از IList استفاده کنید. چون در EF هربار مراجعه به نتیجه یک IEnumerable مساوی است با واکشی دوباره اطلاعات از سرور. در حالت استفاده از IList کار یکبار انجام شده و تمام می‌شود.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۰۱/۱۸ ۱۱:۴۹

اگر در یک سناریو نیاز باشد رشته‌ی اتصال در زمان اجرا عوض شود (مثلا در یک برنامه‌ی ویندوزی که طراحی به گونه ای هست که به ازای هر یک از یک موجودیت خاص یک دیتابیس ایجاد شود.) ، یک روش پاس دادن آن در زمان وهله سازی DbContext به سازنده هست.
در این روش که DbContext به صورت مستقیم در دسترس نیست ، چه روشی برای انجام این کار پیشنهاد می‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۸ ۱۲:۴۳

- DbContext در زمان شروع برنامه در دسترس است. مانند کلاس Sample07Context مثال این قسمت.
- اگر قرار است به ازای هر موجودیت یک دیتابیس داشته باشید یعنی چندین کلاس DbContext مجزا نیاز هست با تعاریف مختلف DbSetها در ابتدای کار. هر کدام را باید جداگانه در شروع برنامه با روش زیر مدیریت کرد. مثلا:

```
public partial class MyCtx1 : DbContext
{
    public MyCtx1(string connectionString) : base(connectionString) { }
}
```

و در این حالت بدیهی است هر کدام در Context مختلفی کار می‌کنند و بحث اعمال الگوی واحد کار در یک Context معنا دارد نه در چندین Context. در EF مباحث Tracking فقط در یک Context کار می‌کنند.
و اگر قرار است تمام موجودیت‌ها در یک کلاس DbContext باشند، خوب ... همگی نهایتا در زمان فعال سازی migrations باید در دیتابیس مشترکی قرار گیرند و گرنه برنامه آغاز نخواهد شد یا اینکه migrations را کلا از کار انداخت.
- به علاوه در همین مثال جاری در زمان تزریق وابستگی‌ها می‌شود نوشت:

```
x.For<IUnitOfWork>().HttpContextScoped().Use(() =>
                {
                    var ctx = new Sample07Context();
                    ctx.Database.Connection.ConnectionString = "...";
                    return ctx;
                });
```

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۰۱/۱۸ ۱۴:۳۳

متشکرم.
منظور من از داشتن دیتایس به ازای هر موجودیت بد بیان شد.
یک DbContext داریم حاوی DbSet ها.
در یک برنامه‌ی ویندوزی کاربر می‌تواند دیتابیس‌های مختلف با نام‌های مختلف با یک طراحی داشته باشد.
در واقع به دلایلی مثل محدودیت SQL Server Express در ذخیره سازی (10 گیگ) طراحی به گونه ای انجام شده که برنامه بتواند به ازای هر پروژه یک دیتابیس داشته باشد.
برنامه به یک دیتابیس وصل است و زمانی که کاربر قصد ایجاد یک پروژه‌ی جدید دارد باید یک دیتابیس جدید ایجاد شود و Connection String عوض شود. و از این پس DbContext باید به دیتابیس جدید متصل شود.

```
public static void ChangeDatabase(string name)
{
    var sqlConnectionStringBuilder =
        new SqlConnectionStringBuilder(ConfigHelper.ActiveConnection);
    sqlConnectionStringBuilder["Database"] = name
    ConfigHelper.ActiveConnection = sqlConnectionStringBuilder.ToString();
    Database.DefaultConnectionFactory =
        new
System.Data.Entity.Infrastructure.SqlConnectionFactory(ConfigHelper.ActiveConnectionString());

    Database.SetInitializer(
        new MigrateDatabaseToLatestVersion<TestContext, MigrationConfiguration>());
    using (var context = new TestContext())
```

```
    {
        context.Database.Initialize(true);
    }
}
```

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۰۱/۱۸ ۱۶:۲۴

با استفاده از این لینک و پاس دادن رشته‌ی اتصال هنگام تزریق وابستگی مشکلم حل شد.

نویسنده: حامد
تاریخ: ۱۳۹۲/۰۲/۰۷ ۱۰:۵۱

ممنون آقای نصیری
یه سوال مهم دارم
من اگه بخوام پروژه سیلورلایت انجام بدم با استفاده از معماری MVVM ، ترکیب اون با این مدل 5 لایه به چه شکل خواهد شد؟
میشه یه مثال خوب هم در این زمینه معرفی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۷ ۱۱:۸

- مباحث سمت سرور آن تفاوتی نمی‌کند؛ از این جهت که سیلورلایت نهایتا با استفاده از سرویس‌های سمت سرور WCF و یا WCF RIA Services قرار است به بانک اطلاعاتی دسترسی پیدا کند و برای نمونه امکان استفاده از EF Code first در WCF RIA Services مدتی هست که فراهم شده .
- ضمن اینکه سیلورلایت آینده مشخصی نداره ؛ بهتره روی ASP.MVC سرمایه گذاری کنید.

نویسنده: سجاد
تاریخ: ۱۳۹۲/۰۲/۰۸ ۲۳:۱۷

با سلام
وقتی از IOC استفاده می‌کنم کد زیر به درستی کار نمیکنه و خطای
No parameterless constructor defined for this object رو میده
با تشکر

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetAllProducts" TypeName="ServiceLayer.EfProductService">
        </asp:ObjectDataSource>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۸ ۲۳:۲۳

مراجعه کنید به مطلب صفحه بندی اطلاعات در ListView .

نویسنده: debugger
تاریخ: ۱۳۹۲/۰۲/۲۲ ۲۰:۵۱

با سلام

"در پاسخ یکی از سوالهافرمودید با استفاده از IUnitOfWork اگر متد جاری شما از 10 کلاس هم استفاده کند، تماما با یک وهله از Context کار می‌کنند"

موقعی که از context یک وهله می‌گیریم و از Using استفاده می‌کنیم آیا به ازا هر کلاسی که در این scope هست connection جدیدی استفاده میشه ؟

آیا هنگام استفاده از Using موارد مربوط به همزمانی و transaction مدیریت نمیشه ؟

اگر این طور نیست مزیت روش بالا نسبت به استفاده از Using چیست ؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۲ ۲۲:۱۰

مزیت این روش، استفاده از یک IoC Container برای مدیریت طول عمر DbContext در طول یک درخواست است.
در برنامه‌های وب، کار صرفا به یک کلیک ساده ختم نمی‌شود که در همان لحظه، یک Context آغاز و پایان یابد. در طی یک درخواست وب، قسمتی از صفحه لیست گروه‌ها، قسمتی دیگر لیست نویسندگان، قسمتی دیگر گزارش درصد استفاده از مرورگرها و قسمتی دیگر لیست آخرین مطالب را نمایش می‌دهد. تمام این‌ها هم در طی یک درخواست رخ می‌دهند و هرکدام، ماژول ماژول طراحی شده‌اند و از هم جدا.
اینجا است که ارزش استفاده از قابلیت‌های مدیریت طول عمر IoC containers برای به اشتراک گذاری یک DbContext در طی یک درخواست بهتر مشخص می‌شود. به این ترتیب می‌شود به سربارى کم و سرعتی بالا دست یافت چون مدام به ازای قسمت‌های مختلف برنامه Context ایجاد و تخریب نمی‌شود.

نویسنده: سجـــــاد ف
تاریخ: ۱۳۹۲/۰۲/۲۸ ۸:۱۴

با سلام
من از یک سری توابع جنریک استفاده میکنم که نیاز به دسترسی DbContext داره آیا تعریف این توابع در IUnitOfWork درسته؟
یک نمونه

```
public interface IUnitOfWork
    {
        IDbSet<TEntity> Set<TEntity>() where TEntity : class;
        int SaveChanges();
        void Update<TEntity>(TEntity entity) where TEntity : class;
    }


 public class MyContext : DbContext, IUnitOfWork
    {
        public void Update<TEntity>(TEntity entity) where TEntity : class
        {
            var fqen = GetEntityName<TEntity>();

            object originalItem;
            EntityKey key = (IObjectContextAdapter)this).ObjectContext.CreateEntityKey
                        (fqen, entity);
            if (((IObjectContextAdapter)this).ObjectContext.TryGetObjectByKey(key, out
                                        originalItem))
            {
               ((IObjectContextAdapter)this).ObjectContext.ApplyCurrentValues
                  (key.EntitySetName, entity);
            }
               ((IObjectContextAdapter)this).ObjectContext.ApplyCurrentValues
                        (key.EntitySetName, entity);
        }
```

```
        private string GetEntityName<TEntity>() where TEntity : class
        {
            return string.Format("{0}.{1}", ((IObjectContextAdapter)this).ObjectContext.
            DefaultContainerName, _pluralizer.Pluralize(typeof(TEntity).Name));
        }

        #region IUnitOfWork Members
        public new IDbSet<TEntity> Set<TEntity>() where TEntity : class
        {
            return base.Set<TEntity>();
        }
        #endregion
    }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۸ ۹:۰

نیازی به این پیچ و تاب‌ها در EF Code first نیست. تابع استاندارد Find ایی که در آن اضافه شده همین کار ابتدا مراجعه به کش و بعد مراجعه به دیتابیس رو انجام می‌ده.

نویسنده: داود
تاریخ: ۱۳۹۲/۰۲/۲۸ ۹:۱۳

با سلام
ما چرا در mvc موجودیت هامون رو تو Model تعریف نکردیم و تو Domain Classes تعریف کردیم حالا اگر نیاز به اعتبارسنجی بود آیا باید تو همون DomainClasses اینکار رو با Metadata که فرموده بودید انجام بدیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۸ ۹:۱۸

نیاز است با یک سری پیشنیاز مانند ViewModel و همچنین « مقابله با مشکل امنیتی Mass Assignment در حین کار با Model binders » آشنا باشید تا علت جدا سازی این موارد از هم مشخص بشه.

نویسنده: داود
تاریخ: ۱۳۹۲/۰۲/۲۸ ۱۰:۴۲

با سلام و تشکر از پاسختون
من با ViewModel آشنایی دارم و مطالب Mvc شما رو کامل خوندم آیا درست متوجه شدم جهت اعتبار سنجی باید یک کلاس در ViewModel بسازم و اعتبارسنجی رو انجام بدیم و بعد کلاس ViewModel رو تبدیل به کلاس اصلی کرده و برای انجام عملیات مربوطه به Service بفرستیم؟ یا نه باید یک کلاس مشابه کلاس موجود در DomainClasses در Model بسازم و اعتبارسنجی رو اونجا انجام بدم و بعد ارسال به Service
یا باید به صورت زیر توی همون domainClasses انجام بدم

```
[MetadataType(typeof(CustomerMetadata))]
public partial class Customer
{
    class CustomerMetadata
     {
     }
}
public partial class Customer : IValidatableObject
{
```

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۰۲/۲۸ ۱۰:۵۹

ViewModel متناظر است با اشیاء یک View که الزاما تطابق یک به یکی با Domain Model و موجودیت‌های بانک اطلاعاتی ندارند. مثلا یک صفحه تعویض پسورد هست و فقط یک فیلد پسورد داره. اینجا در معرض دید قرار دادن کل موجودیت کاربر در یک برنامه وب MVC اشتباه است چون به سادگی مورد حمله واقع خواهد شد. خلاصه هر دو مورد ViewModel و Domain model نیاز به اعتبارسنجی دارند؛ به هر روشی که صلاح می‌دونید.
نهایتا اطلاعات ViewModel در حالت Post، به اطلاعات Model انتساب داده میشه. یا دستی و یا مثلا توسط AutoMapper ؛ در این حالت هم هر طور که راحت هستید عمل کنید. قانون یا روش بهتری برای این نوع انتساب‌ها وجود نداره.

نویسنده: مهدی
تاریخ: ۱۳۹۲/۰۳/۰۱ ۱۵:۳۰

سلام
نمونه ای از پیاده سازی روش بالا در WPF با MVVM سراغ دارید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۱ ۱۶:۲۶

پیشنیاز تئوری قسمت 12، دوره‌ای است به نام « بررسی مفاهیم معکوس سازی وابستگی‌ها و ابزارهای مرتبط با آن » در حدود 11 قسمت.
بعد از مطالعه آن، خودتان به سادگی می‌توانید این مباحث را در الگوهای مختلف پیاده سازی کنید.

نویسنده: m.d
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۲:۳۵

با تشکر از معرفی دوره آموزشی شما
متاسفانه هنوز متوجه نشدم چگونه uow را در ViewModelهای برنامه استفاده کنم
ترکیب StructureMap , MVVM Light ، EF Code First چگونه پیاده سازی میشود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۲:۴۲

دوره جدیدی به سایت اضافه شده تحت عنوان « طراحی یک فریم ورک برای کار با WPF و EF Code First توسط الگوی MVVM ».
دسترسی به آن فقط برای نویسندگان سایت با حداقل یک مطلب ارسالی در طی یک ماه قبل است. البته همه‌ی کاربران عضو ، می‌توانند مشارکت کنند و در سایت مطلب ارسال کنند. از این لحاظ محدودیتی وجود ندارد.

نویسنده: m.d
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۴:۲۳

یعنی برای استفاده از این دوره می‌بایست یک مطلب به سایت ارسال کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۴:۲۷

بله. البته پس *از* تائید محتوای آن و انتشار در صفحه اول سایت، بلافاصله دسترسی شما باز خواهد شد.

نویسنده: پویا امینی
تاریخ: ۱۳۹۲/۰۳/۲۹ ۰:۱۵

با سلام، من کد شما را به صورت زیر تغییر دادم
ابتدا یک اینترفیس به صورت زیر ایجاد کردم

```
namespace Service.Interfaces
{
    public interface IGenericService<T>
    {
        void AddOrUpdate(T entity);
        void Delete(T entity);
        T Find(Func<T, bool> predicate);
        T GetLast(Func<T, bool> predicate);
        IList<T> GetAll();
        IList<T> GetAll(Func<T, bool> predicate);
        IList<T> GetAll(Expression<Func<T, object>> orderby);
        IList<T> GetAll(Func<T, bool> predicate, Expression<Func<T, object>> orderby);

        Task<List<T>> GetAllAsync();
        Task<List<T>> GetAllAsync(Func<T, bool> predicate);
        Task<List<T>> GetAllAsync(Expression<Func<T, object>> orderby);
        Task<List<T>> GetAllAsync(Func<T, bool> predicate, Expression<Func<T, object>> orderby);

        int Count();
        int Count(Func<T, bool> predicate);
    }
}
```

و تمام اینترفیس‌های دیگر از این به صورت زیر به ارث برده شده اند

```
public interface IBookGroupService:IGenericService<BookGroup>
    {
    }
```

و در قسمت Servise یک کلاس ایجاد کردم که اینترفیس IGenericService را پیاده سازی می‌کند که کدهای آن به صورت زیر است

```
public class EFGenericService<TEntity> : IGenericService<TEntity>
         where TEntity : class
    {
        protected IUnitOfWork _uow;
        protected IDbSet<TEntity> _tEntities;

        public EFGenericService(IUnitOfWork uow)
        {
            _uow = uow;
            _tEntities = _uow.Set<TEntity>();
        }


        public void AddOrUpdate(TEntity entity)
        {
             _tEntities.AddOrUpdate(entity);
        }

        public virtual void Delete(TEntity entity)
        {
            _tEntities.Remove(entity);
        }

        public virtual TEntity Find(Func<TEntity, bool> predicate)
        {
            return _tEntities.Where(predicate).FirstOrDefault();
        }

        public virtual TEntity GetLast(Func<TEntity, bool> predicate)
        {
            return _tEntities.Where(predicate).Last();
        }


        public virtual IList<TEntity> GetAll()
        {
            return _tEntities.ToList();
```

```
        }

        public virtual IList<TEntity> GetAll(Func<TEntity, bool> predicate)
        {
            return _tEntities.Where(predicate).ToList();
        }

        public virtual IList<TEntity> GetAll(Expression<Func<TEntity, object>> @orderby)
        {
            return _tEntities.OrderBy(@orderby).ToList();
        }

        public virtual IList<TEntity> GetAll(Func<TEntity, bool> predicate, Expression<Func<TEntity,
object>> @orderby)
        {
            return _tEntities.OrderBy(@orderby).Where(predicate).ToList();
        }

        public async Task<List<TEntity>> GetAllAsync()
        {
           return await Task.Run(() => _tEntities.ToList());
        }

        public async Task<List<TEntity>> GetAllAsync(Func<TEntity, bool> predicate)
        {
          return await Task.Run(() => _tEntities.Where(predicate).ToList());
        }

        public async Task<List<TEntity>> GetAllAsync(Expression<Func<TEntity, object>> @orderby)
        {
           return await Task.Run(() => _tEntities.OrderBy(@orderby).ToList());
        }

        public async Task<List<TEntity>> GetAllAsync(Func<TEntity, bool> predicate,
Expression<Func<TEntity, object>> @orderby)
        {
           return await Task.Run(()=> _tEntities.OrderBy(@orderby).Where(predicate).ToList());
        }

        public virtual int Count()
        {
            return _tEntities.Count();
        }

        public virtual int Count(Func<TEntity, bool> predicate)
        {
            return _tEntities.Count(predicate);
        }
    }
```

و بقیه کلاس‌ها از کلاس بالا به ارث می‌برند.

```
public class EFBorrowService:EFGenericService<Borrow>,IBorrowService
    {
        public EFBorrowService(IUnitOfWork uow) : base(uow)
        {
        }
    }
```

حال سوال من اینه که این پیاده سازی از لحاظ پیاده سازی مشکلی ندارد؟ و می‌توانم در پروژه هام از این روش استفاده کنم یا خیر؟

ممنونم

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۰۴/۰۱ ۹:۲۱

با سلام.
من کل پوشه Controllers را درون یک اسمبلی جداگانه قرار دادم. ولی هنگام اجرای برنامه خطای زیر رخ میدهد.

```
public class StructureMapControllerFactory : DefaultControllerFactory
    {
        protected override IController GetControllerInstance(RequestContext requestContext, Type
controllerType)
        {
            if (controllerType == null)
            {
                throw new InvalidOperationException(string.Format("Page not found: {0}",
requestContext.HttpContext.Request.Url.AbsoluteUri.ToString(CultureInfo.InvariantCulture)));
            }
            return ObjectFactory.GetInstance(controllerType) as Controller;
        }
    }

StructureMap Exception Code:  202
No Default Instance defined for PluginFamily MyProject.Controllers.HomeController,
MyProject.Controllers, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
```

با تشکر.

نویسنده: اکبر بابامرادی
تاریخ: ۱۳۹۲/۰۴/۲۰ ۰:۴

با سلام و خسته نباشید

آقای نصیری با توجه به توضیحات شما اگه نیاز به متدهای (مثلا: where, Skip,take,...) داشته باشیم آیا این متدها رو هم باید پیاده سازی کنیم؟

یا اینکه لیستی از مقادیر موجود رو بخونیم و بعد با استفاده از linq کوری مورد نظر رو استخراج کنیم!؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۰ ۰:۱۳

- منطق تجاری مورد نظر رو باید به صورت یک متد با حد و مرز مشخص، کپسوله و داخل این متد از Skip و Take و سایر امکانات LINQ استفاده کنید.
- ضمنا نباید لیست تهیه کنید و بعد روی آن Take انجام دهید؛ چون کارآیی پایینی داره:
« تفاوت بین IQueryable و IEnumerable در حین کار با ORMs »

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۲/۰۴/۲۷ ۰:۲

با سلام
یکی از دوستان در این قسمت نظری به شرح {در حین اجرای نمونه کدهای این مقاله در بخش MVC به خطای Value Cannot be null مواجه شدم } بیان کردند و راه حل نیز ارائه کردند و شما نیز پاسخی برای نظر ایشان ذکر کردید میشه در مورد وقوع این خطا بیشتر توضیح بدید ؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۷ ۰:۳۸

علت وقوع خطا، در استثنای صادره ذکر شده: Page not found . یعنی کاربر یا حتی مرورگر درخواست آدرسی رو کرده که در سایت شما موجود نیست (مثلا کروم اولین کاری که انجام می‌ده، وجود فایل استاندارد آیکن سایت رو به صورت خودکار بررسی می‌کنه) . بنابراین امکان وهله سازی کنترلر معادل آن صفحه یا آدرس یافت نشده، وجود ندارد.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۲/۰۴/۲۷ ۴:۲۴

این یعنی به ازاء تمام درخواستهایی که سمت سرور ارسال میشه context تشکیل میشه (نیاز به context باشه یا نه ، مثلا یه تصویر قرار هست در صفحه نمایش داده شود و نیاز به کنترلر هم نداره. خطای گزارش شده نمایش آیکون بود) ؟
آیا از لحاظ Performance مشکلی نداره ؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۷ ۸:۵

- Context زمانی تشکیل خواهد شد که کار وهله سازی کنترلر متناظر و موجودی با موفقیت انجام شده باشد.
+ زمانیکه runAllManagedModulesForAllRequests در وب کانفیگ به true تنظیم شده تمام درخواست‌ها به موتور ASP.NET نگاشت می‌شوند. می‌شود این وضعیت را کنترل کرد؛ مراجعه کنید به قسمت « تنظیمات ثانویه پس از فعال سازی RouteExistingFiles »

در کل تهیه یک برنامه ASP.NET MVC نیاز به رعایت یک سری موارد دارد که پیشتر در این سایت بحث شده: « چک لیست تهیه یک برنامه ASP.NET MVC ». یکی از نکات آن هم «پوشه‌های Content و Scripts از سیستم مسیریابی تعریف شده در Global.asax خارج شوند» است.

نویسنده: hossein101211
تاریخ: ۱۳۹۲/۰۵/۰۵ ۲۱:۸

با سلام و خسته نباشید
من یه کلاس استاتیک دارم میخواستم ببینم لزومی داره داخل متدهای استاتیک هم الگوی unit of work استفاده بشه یا نه؟
چون به نظرم متدی که استاتیک تعریف میشه یه جورایی الگوهایی که باید رعایت بشه(مخصوصا unit of work) رو دور میزنه!
نمیدونم تا چه حد درست فکر میکنم یانه؟

```
private  static  readonly ICatHotellService _catHotellService;
  private static readonly ICatTourismService _catTourismService;
  private static readonly ICatTourService _catTourService;
  private static readonly IUnitOfWork _uow;
  public DropDownList(ICatHotellService CatHotellService, IUnitOfWork ouw, ICatTourService
CatTourService, ICatTourismService CatTourismService)
  {
_uow=ouw;
_catHotellService = CatHotellService;
_catTourismService = CatTourismService;
_catTourService = CatTourService;
  }
```

ولی دیگه نمیشه داخل سازنده DropDownList اونا رو بهم نسبت داد
لطفا کمی راهنمایی کنید با تشکر

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۰۵/۰۵ ۲۱:۴۰

اگر برنامه وب است، به هیچ عنوان نباید از سرویس‌هایی که به صورت یک فیلد استاتیک تعریف شدند استفاده کنید:
بررسی واژه کلیدی static
متغیرهای استاتیک و برنامه‌های ASP.NET

- اگر برنامه دسکتاپ است و نیاز دارید که اطلاعات یک سرویس خاص را در طول عمر برنامه زنده نگه دارید، برای نمونه در StructureMap حالت طول عمر Singleton هم وجود دارد برای مدیریت این نوع سرویس‌ها و نیازی نیست باز هم متغیر استاتیک تعریف کنید. (یک نمونه آن در دوره « طراحی یک فریم ورک برای کار با WPF و EF Code First توسط الگوی MVVM » بحث شده به همراه مثال کاربردی)
- ضمنا زنده نگه داشتن اطلاعات یک سرویس در طول عمر یک برنامه، باید با آگاهی کامل صورت گیرد. در اینجا و در حالت استفاده از EF، به این ترتیب Context ایجاد شده Dispose نخواهد شد و همین مساله مشکلات زیادی مانند خطاهای ثبت اطلاعات جدیدی که پیشتر در صفحه‌ای دیگر به Context وارد شدن را سبب می‌شود. همچنین در محیط‌های چندکاربری مانند وب، یک Context به اشتراک گذاشته بین تمام کاربران (مفهوم متغیرهای استاتیک)، thread safe نیست و مشکلات تداخل اطلاعات و یا حتی تخریب آن‌ها را شاهد خواهید بود.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۲/۰۵/۱۳ ۱۵:۹

سلام
در مثال مربوط به MVC برای controllerها از یک کلاس به عنوان base استفاده کردم و متد ExceuteCore رو جهت تنظیمات Globalization تحریف کردم . در این حالت متد ExceuteCore اجرا نشد .چند تا سوال داشتم ممنون میشم راهنمایی کنید
1- آیا تحریف متد ExcuteCore برای تنظیمات Globalization جای مناسبی هست ؟
2- مشکل مربوط به اجرا نشدن ExecuteCore رو با توجه به اینکه StructureMap وهله جدیدی از controller ایجاد میکنه رو چه طور میشه برطرف کرد .
ممنون

نویسنده: امیر
تاریخ: ۱۳۹۲/۰۵/۱۴ ۱۹:۱۱

سلام. اگر بخواهیم با استفاده از الگوی واحد کار یک کوئری دستی ایجاد کنیم روش کار به چه صورتی خواهد بود؟ یعنی من به صورت عادی از کد زیر استفاده میکنم:

```
using (var context = new MyContext())
{
context.Database.SqlQuery......
}
```

حالا با استفاده از UnitOfWork چگونه باید این کار رو انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۵ ۹:۵۸

شما محدود نیستید به چند متد اولیه‌ای که در اینترفیس Uow ذکر شده. یک متد با امضای اجرای SQL به آن اضافه کنید و پیاده سازی آن‌را در کلاس Context خود که از اینترفیس Uow مشتق می‌شود، انجام دهید (مانند this.Database الی آخر). بعد کلاس‌های استفاده کننده از Uow به آن دسترسی خواهند داشت.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۵ ۱۳:۲۸

- خیر. ASP.NET MVC یک فریم ورک AOP سر خود است. این مسایل رو باید با فیلترها پیاده سازی کنید.

- StructureMap وهله جدیدی را ایجاد می‌کند، اما ... کار استفاده (یا عدم استفاده) از آن به عهده ASP.NET MVC است و StructureMap دخالتی در آن ندارد.
- این مورد (عدم فراخوانی ExecuteCore تحریف شده) تغییری است که در MVC4 اعمال شده

```
public class MyBaseController : Controller
    {
        /// <summary>
        /// from
http://forums.asp.net/t/1776480.aspx/1?ExecuteCore+in+base+class+not+fired+in+MVC+4+beta
        /// </summary>
        protected override bool DisableAsyncSupport
        {
            get { return true; }
        }

        protected override void ExecuteCore()
        {
            base.ExecuteCore();
        }
    }
```

باید DisableAsyncSupport را اضافه کنید.

نویسنده: محمد تیموری بادله
تاریخ: ۱۳۹۲/۰۸/۲۱ ۱۹:۴۰

با سلام و تشکر از مطالب مفید تون

اگر بخواهیم تحت هر شرایطی به کلاس DbContext دسترسی داشته باشیم به چه شکلی خواهد بود؟
در این حالت اگر بخواهیم یکی از Entityها را ویرایش کنیم به چه صورتی هست؟من هر کاری م کنم با ارور برخورد می‌کنم و فکری غیر از اینکه DbContext را داخل uow بیارم به ذهنم نمی‌رسه!

به چه شکلی می‌شود این مشکل را حل کرد؟

با احترام.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۱ ۲۱:۵۳

سیستم مدیریت محتوای IRIS از الگوی واحد کار استفاده کرده. از کدهاش برای انجام یک پروژه واقعی ایده بگیرید. برای برنامه‌های دسکتاپ هم دوره طراحی یک فریم ورک برای کار با WPF و EF Code First توسط الگوی MVVM از الگوی واحد کار استفاده می‌کنه.

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۰۹/۰۲ ۱۶:۱۲

با خوندن مطالب فوق این طور بر میاد که در Ef یک Context به صورت Singleton ایجاد بشه تا هم بهینه باشه و هم مباحث مدیریت Transactionها و غیره به راحتی مدیریت بشه.

اما در اینجا StackOverflow در این خصوص خوندم که بهتره برای هر thread یک Context مجزا ایجاد کرد و سیستم Pools کانکشن تکراری و ارتباط متعدد را چک می‌کند.

بنده اشتباه برداشت کردم و یا سیستم Pool استفاده دیگره ای دارد .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۰۲ ۱۸:۴۶

- یک Context در EF باید طول عمر کوتاهی داشته باشد. سینگلتون تعریف کردن آن یعنی زنده نگه داشتن آن در طول عمر برنامه. در یک برنامه وب به این ترتیب اگر جایی کاربری به مشکلی برخورد، چون یک Context در این حالت بیشتر وجود نخواهد داشت، تمام خطاهای آن کاربر به سایر کاربران نیز منعکس می‌شود. همچنین Context به صورت thread safe طراحی نشده و به این ترتیب به مشکلات تخریب اطلاعات در برنامه‌های چند کاربره نیز برخواهید خورد.
- در مطلب فوق به ازای هر درخواست یک Context ایجاد می‌شود (مقدمه بحث). Context در طول عمر یک درخواست کاربری خاص، زنده نگه داشته شده و بعد در پایان کار آن درخواست Dispose می‌شود. نه فقط بحث مدیریت اتصالات در اینجا مطرح است، بحث مدیریت یک تراکنش واحد در طول عمر یک درخواست نیز باید درنظر گرفته شود.

چند پیشنهاد:
- یکبار مطلب جاری را مطالعه کنید.
- یکبار وقت بگذارید نظرات آن‌را هم بررسی کنید. در مورد سینگلتون یا حتی Contextهای استاتیک و امثال آن قبلا بحث شده.
- یکبار دوره پیشنیاز این مطلب را مطالعه کنید: « بررسی مفاهیم معکوس سازی وابستگی‌ها و ابزارهای مرتبط با آن »
- یکبار دوره خاص برنامه‌های دسکتاپ طراحی شده با این الگو را هم مطالعه کنید: « طراحی یک فریم ورک برای کار با WPF و EF Code First توسط الگوی MVVM »
- نگاهی هم به یک پروژه کامل ASP.NET MVC که با درنظر گرفتن الگوی مطرح شده در این بحث تهیه شده، داشته باشید: « سیستم مدیریت محتوای IRIS »

این مبحث باز شده‌اش بالای 20 قسمت است.

نویسنده: reza110
تاریخ: ۱۳۹۲/۰۹/۱۸ ۱۷:۱۲

یکی از اشکالات repository این بود که در عمل و دنیای واقعی، قابلیتی برای تعویض ORM ایجاد نمی‌کرد و گفتید در صورتی می‌توان این کار را کرد که دست و پای ORM را ببندیم و از مشترکات استفاده کنیم. اما در الگویی که معرفی کردید یعنی IUnitOfWork هم ظاهرا دست و پایمان برای پیاده سازی متدهای خاص بسته است مانند متد ساده زیر

```
public void ChangeState<T>(T entity, EntityState state) where T : class
{
     Entry<T>(entity).State = state;
}
```

یا متدی که در بالا ذکر شد یعنی:

```
DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;
```

اینطور نیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۸ ۱۸:۲۸

برای «پیاده سازی متدهای خاص» متد اضافه کن؛ اینترفیس رو تغییر بده. در *مثالی* که زده شده، من دو تا متد تعریف کردم. شما بسطش بده. مثلا:

```
public interface IUnitOfWork
{
   //...
```

```
   DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;
}
```

نویسنده: reza110
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۴:۲۰

در مدلی که ارایه کردید هم نمی‌توان متدهای خاص که از کلاسهای خاصی استفاده کرده اند را بکار برد مثل DbEntityEntry یا EntityState

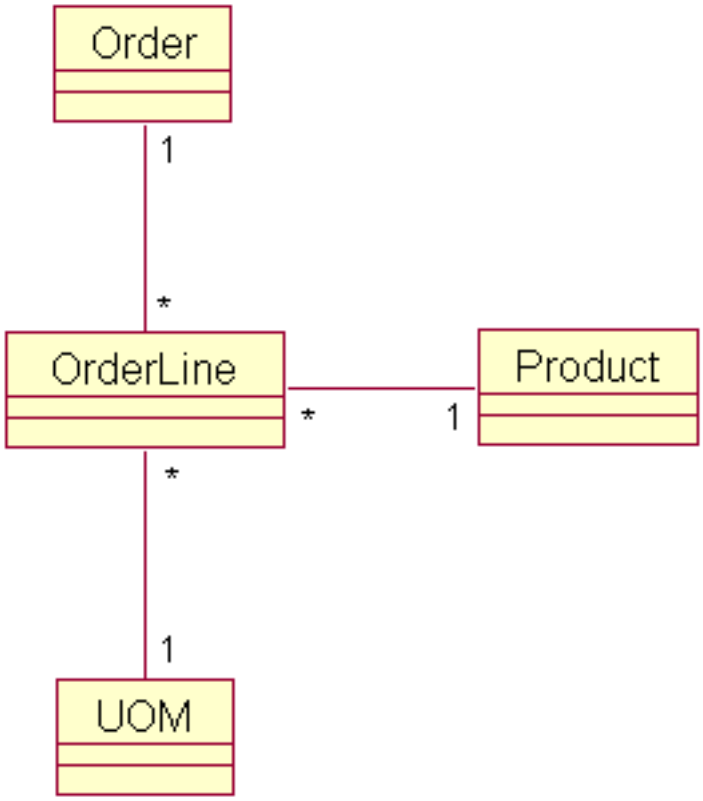
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۵:۳۱

عرض کردم، اینترفیس را بسط دهید؛ مثلا مانند کدهای زیر. DbEntityEntry را داخل یک متد مانند MarkAsChanged هم می‌شود محصور کرد با خروجی void. به عبارتی نحوه کار با base.Entry را بهتر می‌شود از دید مصرف کننده مخفی کرد تا حتما او نیازی نداشته باشد خودش مستقیما base.Entry(entity).State = EntityState.Modified را در کدهای نهایی مورد استفاده قرار دهد. فقط کافی باشد تا متد عمومی MarkAsChanged را که در پشت صحنه از base.Entry(entity).State استفاده می‌کند، بکارگیرد.

```
namespace EF_Sample07.DataLayer.Context
{
    public interface IUnitOfWork
    {
        IDbSet<TEntity> Set<TEntity>() where TEntity : class;
        int SaveAllChanges();
        void MarkAsChanged<TEntity>(TEntity entity) where TEntity : class;
    }
}

namespace EF_Sample07.DataLayer.Context
{
    public class Sample07Context : DbContext, IUnitOfWork
    {
        public DbSet<Category> Categories { set; get; }
        public DbSet<Product> Products { set; get; }

        public new IDbSet<TEntity> Set<TEntity>() where TEntity : class
        {
            return base.Set<TEntity>();
        }

        public int SaveAllChanges()
        {
            return base.SaveChanges();
        }

        public void MarkAsChanged<TEntity>(TEntity entity) where TEntity : class
        {
            base.Entry<TEntity>(entity).State = EntityState.Modified;
        }
    }
}
```

نویسنده: مسعود2
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۴:۴۴

آیا امکان ثبت یک گراف از اشیائ مرتبط، بصورت یکجا در طی یک درخواست وجود دارد؟(در یک برنامه multi-tier وب یا ویندوزی) منظورم این است که فرضا مدلی داریم به شکل زیر:



آیا امکان این وجود دارد که کاربر یک سفارش را بصورت زیر ویرایش کند:

ویرایش تاریخ سفارش در کلاس Order

اضافه نمودن یک OrderLine جدید به Order
حذف یکی از OrderLineهای موجود
ویرایش Product مربوط به یک OrderLine

و سپس دکمه Save را بزند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۵:۶

اخیرا کتابی منتشر شده به نام Entity Framework 6 Recipes . این کتاب فوق العاده است. جزو معدود کتاب‌های چند سال اخیر است که ارزش یکبار خواندن را دارد. فصل 9 آن دقیقا مرتبط است به موضوع «Using the Entity Framework in N-Tier Applications».

نویسنده: حسین
تاریخ: ۱۳۹۲/۱۰/۰۵ ۲۰:۳۳

سلام
من دقیقا طبق همین الگویی که عرض کردید درحال نوشتن پروژه هستم ، در بعضی از نقاط پروژه باید از jquery ajax استفاده کنم ، به عنوان مثال ورود کاربر ، حالا مسئله ای که هست من باید در webmethod مربوطه یک method static رو صدا بزنم ، توی این الگو چطور می‌تونم مثلا ثبت یک رکورد رو بصورت یک method static انجام بدم ؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۵ ۲۱:۱

- باید از الگوی Service locator استفاده کنید در این موارد خاص فناوری‌های قدیمی که برای تزریق وابستگی‌ها طراحی نشده‌اند. پیشنیاز این بحث دوره « بررسی مفاهیم معکوس سازی وابستگی‌ها و ابزارهای مرتبط با آن » است.
- ضمن اینکه الان با بودن ASP.NET Web API که هم با وب فرم‌ها سازگار است و هم با MVC، دلیلی برای استفاده از وب متدهای استاتیک عهد عتیق وجود ندارد. ASP.NET Web API طوری طراحی شده تا تزریق وابستگی‌ها در آن ممکن و آزمون پذیری آن بالا باشد.

نویسنده: مسعود2
تاریخ: ۱۳۹۲/۱۰/۰۶ ۲۲:۹

فرض کنید که بخواهیم در این مثال این کارها رو انجام بدیم:
در یک صفحه لیست کالاها و دسته بندی اونها رو نشون بدیم.

کاربر قادر باشه در همون صفحه مشخصات یک کالا شامل گروه کالا یا نام کالا رو ویرایش کنه.

در این حالت برای ویرایش آیا بایستی از همون وهله DbContext که جهت گرفتن لیست کالاها استفاده شده، استفاده بشه؟ یا برای ویرایش بایستی یک وهله جدید DbContext ساخته بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۶ ۲۳:۴۰

برنامه وب هست یا برنامه ویندوز؟ اگر برنامه وب هست که پس از پایان نمایش صفحه Context شما Dispose شده در سرور.
اگر برنامه ویندوزی هست، بله می‌توانید از همان وهله استفاده کنید. چون تا زمانیکه فرم باز است، آن وهله هم می‌تواند باز باشد، یا زنده نگه داشته شود. نمونه آن در مثال « طراحی یک فریم ورک برای کار با WPF و EF Code First توسط الگوی MVVM » مطرح شده. یکبار مثال آن‌را اجرا کنید. «لطفا»

نویسنده: محمدرضا برنتی
تاریخ: ۱۳۹۲/۱۰/۰۷ ۲۳:۱۵

```
private static void initStructureMap()
        {
            ObjectFactory.Initialize(x =>
            {
                x.For<IUnitOfWork>().HttpContextScoped().Use(() => new Sample07Context());
                x.ForRequestedType<ICategoryService>().TheDefaultIsConcreteType<EfCategoryService>();
                x.ForRequestedType<IProductService>().TheDefaultIsConcreteType<EfProductService>();
            });
            //Set current Controller factory as StructureMapControllerFactory
            ControllerBuilder.Current.SetControllerFactory(new StructureMapControllerFactory());
        }
```

دقیقا در هر خط چه کاری انجام می‌دهند ؟ امکان داره یه مثال حقیقی بزنید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۷ ۲۳:۲۳

در دوره « بررسی مفاهیم معکوس سازی وابستگی‌ها و ابزارهای مرتبط با آن » مفصل بحث شده و جزئیات آن کاملا بررسی شده‌اند.

نویسنده: vici
تاریخ: ۱۳۹۲/۱۱/۰۴ ۱۵:۲۴

ممنون؛ در برنامه ای که از linq to sql استفاده شده. بخواهیم همین روش (Unit of work) رو پیاده کنیم چه کاری باید انجام بشه؟ با تشکر

نویسنده: vici
تاریخ: ۱۳۹۲/۱۱/۰۵ ۲۲:۴۱

سلام؛ من از روش شما استفاده کردم در این خط کد

```
x.For<IUnitOfWork>().CacheBy(InstanceScope.Hybrid).Use<Context>();
```

یه هشدار میده که به این صورت

```
Warning 1 '....CacheBy(StructureMap.InstanceScope)' is obsolete:
```

این جوری که من متوجه شدم میگه این روش absolute شده درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۵ ۲۲:۵۰

- ویژگی absolute یعنی متد CacheBy در نگارش بعدی احتمالا حذف خواهد شد؛ نه اینکه در نگارش فعلی قابل استفاده نیست.
- مبحث تزریق وابستگی‌ها و به روز شده‌ی این مطالب در دوره‌ای به همین نام در سایت ارائه شده

```
x.For<IUsersService>().HybridHttpOrThreadLocalScoped().Use<UsersService>();
```

نویسنده: ناظم
تاریخ: ۱۳۹۲/۱۱/۱۲ ۲۳:۷

سلام؛ در اینترفیس IUnitofWork ما 2 تا متد زیر روداریم

```
IDbSet<TEntity> Set<TEntity>() where TEntity : class;
int SaveChanges();
```

که هر 2 تا تو dbcontext پیاده سازی شدن، ولی تو context خودمون متد Set دوباره پیاده سازی شده
که همون متد dbcontext رو اجرا میکنه، چه نیازی به این کار بود؟ و با متد savechanges چرا اینکارو نکردیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۲ ۲۳:۱۳

جهت نیازهای آموزشی مفاهیم ارث بری در کلاس‌ها؛ مثلا اگر متدی هم نام با یکی از متدهای کلاس پایه DbContext را خواستید بازنویسی کنید، چکار باید کرد و امثال آن. این بازنویسی‌ها ممکن است به همراه یک سری کد اضافی از طرف شما هم باشند. مثلا متد Save کلاس پایه را بازنویسی کنید و قبل از آن خودتان اعتبارسنجی خاصی را اضافه کنید. ضمنا ضرورتی هم در بکارگیری متدهای هم نام با کلاس پایه، نیست. الان حداقل مشاهده کرده‌اید که با کدام متدهای کلاس پایه باید کار کرد و به چه صورتی.

نویسنده: ح مرادف
تاریخ: ۱۳۹۲/۱۱/۲۵ ۱۵:۷

سلام، با تشکر از مقاله عالیتون. بنده 2 تا علامت سوال توی ذهنم پدید اومده که اگر کمکم کنین ، بسیار ممنون میشم :
-1 در تمامی کدها خبری از try catch نیست ، آیا نیاز نیست ؟ یعنی اگر جایی مشکلی پیش بیاد کاربر صفحه ارور معروف asp رو

نمی‌بینه ؟
اگر نیاز هست ، آیا در داخل کنترلر باید استفاده شود ؟
{تا جایی که من می‌دونم گویا در لایه سرویس باید اطلاعات رو درست در نظر بگیریم و بررسی اطلاعات ورودی و مدیریت خطا باید در کنترلر باشه}

-2 در بخش زیر ، برای بنده که از First Database استفاده می‌کنم ، باید چکار کنم ؟

```
public class Sample07Context : DbContext, IUnitOfWork
```

آیا باید یک کلاس دیگه بسازم و از کانتکس اصلیم ارث بدم ؟
با تشکر از وقتتون.
یا حق

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۵ ۱۷:۳۳

- نیازی نیست. کرش بسیار پدیده‌ی نیکویی است و مشخصه‌ی وجود مشکل در سیستم. ELMAH را به پروژه اضافه کنید برای ثبت خودکار جزئیات استثناهای رخ داده و همچنین صفحه‌ی عمومی نمایش خطایی رخ داده‌است (کاربر به دلایل امنیتی نباید هیچ صفحه‌ی دیگری را در این حالت مشاهده کند). همین کافی است. پس از یک مدت کار کردن با ELMAH به ارزش آن در بالا بردن کیفیت برنامه پی‌خواهید برد.
- من راه حل آماده‌ای برای سایر حالات EF ندارم. این سری فقط code first بوده.

نویسنده: مجتبی فخاری
تاریخ: ۱۳۹۲/۱۲/۱۰ ۱۱:۳۱

با سلام
در زمان پیاده سازی الگوی واحد کار در برنامه‌های MVC زمانی که در کنترلر می‌خواهیم یک view بسازیم اگر گزینه create a strongly typed view انتخاب شود از بخش model Class باید کدام مدل را انتخاب کنیم؟
فایل‌های داخل پروژه Model که به عنوان ViewModel تعریف شده اند یا ...! ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۰ ۱۲:۱۶

این مساله ارتباطی به الگوی واحد کار ندارد. شما به عنوان برنامه نویس باید پس از بررسی تشخیص دهید که آیا خطر mass assignment در حین کار با شیء در حال دریافت از کاربر (هر نامی که دارد)، برنامه را تهدید می‌کند یا خیر. همچنین آیا View در حال استفاده نیاز به چند Model برای کار کردن دارد یا خیر. در این حالات استفاده از ViewModel توصیه می‌شود. در غیراینصورت استفاده از Domain modelها نه مشکل امنیتی را به همراه خواهند داشت و نه برای صرفا گزارش گیری، کم و کسری دارند.

نویسنده: رضا شش
تاریخ: ۱۳۹۲/۱۲/۱۸ ۱۷:۱۹

با الگوی uow ظاهرا می‌توان کانتکس‌های مختلف تعریف کرد اما سوال من این است که مثلا به دلایلی کلاس مدل من مثل Order به دلیل ساختار مختلف دیتابیس در سالهای مختلف فرق می‌کند قبلا برای رفع این مشکل از الگوی Factory استفاده می‌کردم اما در EF CodeFirst چگونه باید این تنظیمات در کانتکس تعریف شود. ظاهرا همه این کارها در پشت صحنه و زمان اجرا انجام می‌شود و خودکار DbSetها پر می‌شوند.

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۱۲/۱۸ ۱۸:۴

« استفاده از چندین Context در EF 6 Code first »

نویسنده: سنائی
تاریخ: ۱۳۹۲/۱۲/۱۹ ۷:۲۵

چنانچه بخواهیم از BoundedContext استفاده کنیم، نحوه استفاده از الگوی واحد کار به چه صورت است؟ فرضا چنانچه SaleDbContext, ShippmentDbContext داشته باشیم که هر دو از IUnitOfWork به ارث رفته اند و در یک سرویس بخواهیم عملیاتی را در هر دو Context و طی یک transaction انجام دهیم، Ioc container هنگام وهله سازی IUnitOfWork، چه کلاسی را بایستی new کند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۹ ۹:۳۷

« آشنایی با TransactionScope » مطلب + نظرات
« استفاده از چندین Context در EF 6 Code first » نکات قسمت داشتن چندین Context در برنامه و مدیریت تراکنش‌ها
« Shrink EF Models with DDD Bounded Contexts » برای روش مدیریت بانک اطلاعاتی

نویسنده: رضا شش
تاریخ: ۱۳۹۲/۱۲/۲۰ ۹:۴۶

با عرض معذرت ، من بخش چندین Context را مطالعه کردم ولی هنوز به پیاده سازی درست نرسیدم. نمونه مثال ساده ای که گذاشته ام فکر می‌کنم سوالم را واضح‌تر کند.

```
//search for person with ID = 1 in year 92.
            using (var context = new TestContextNew())
            {
                // در اینجا هم باید بنحوی بتوان با مشخص کردن سال مورد نظر اطلاعات از جدول مربوطه لود
شود
                //Info_92 مثلا برای سال 92 از جدول
                var result = from h in context.Info_News where h.ID == 1 select h;

                dataGridView1.DataSource = result.ToList();
            }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۰:۲۵

- می‌شود به ازای هر سال یک Context مجزا با Entityهای مجزا درست کرد. فایل مثالی که با دو Context کار می‌کند در نظرات همان مطلب « استفاده از چندین Context در EF 6 Code first » پیوست شده‌است: Sample25.cs
ولی این روش سبب خواهد شد مجبور شوید به ازای هر سال، کوئری‌های LINQ مختلفی را هم بنویسید. یعنی لایه سرویس برنامه را باید هربار بازنویسی کنید، فقط برای اینکه نمی‌خواهید ساختار بانک اطلاعاتی را به روز کنید. چرا؟
- EF با استفاده از امکانات Migration به سادگی ساختار بانک‌های اطلاعاتی را به صورت خودکار می‌تواند به روز کند. باید هم اینکار را انجام بدهید چون کوئری‌های مختلف LINQ شما نهایتا به SQL ترجمه شده و چون یک سری از فیلدها در بانک اطلاعاتی سال قبل حضور ندارند، عملا برنامه کار نخواهد کرد. یعنی قسمت عمده‌ای از برنامه شما (کل لایه سرویس) از کار می‌افتد. کامپایل شدن برنامه در این حالت مهم نیست. آیا مثلا تنها کوئری GetAll ایی که تهیه شده، بر روی تمام سال‌ها و با ساختارهای مختلف اجرا می‌شود؟ خیر.
- سپس برای کار با بانک‌های اطلاعاتی دارای یک ساختار و مربوط به سال‌های مختلف، امکان تعیین رشته اتصالی به ازای هر Context هست:

```
context.Database.Connection.ConnectionString = "...";
```

نویسنده: reza

تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۷:۱۰

من می‌خواهم در پروژه ام از uow استفاده کنم و نیاز به متد زیر دارم

```
context.Database.SqlQuery(type, sql, parameters)
```

ولی نوعی که برمی گرداند از نوع DbRawSqlQuery می‌باشد. چگونه باید متدی سازگار با متد فوق را در uow بازنویسی کنم که uow وابسته به EF خاصی نشود.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۷:۲۵

یک ToList به آخر آن اضافه کنید:

```
public class MyContext : DbContext, IUnitOfWork
    {
        // ...
        public IList<T> GetRows<T>(string sql, params object[] parameters) where T: class
        {
            return this.Database.SqlQuery<T>(sql, parameters).ToList();
        }
    }
```

نویسنده: reza
تاریخ: ۱۳۹۳/۰۱/۰۱ ۱:۳۸

ضمن تبریک سال نو.
منظورم متد غیر جنریک SqlQuery بود . نوع انتیتی در زمان اجرا مشخص می‌شود. این متد ToList ندارد.
کلا آیا روشی وجود دارد که بتوان در درون متد غیر جنریک نوع جنریک آن را صدا زد. مثلا همین SqlQuery هم جنریک دارد و هم غیر جنریک
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۱ ۹:۴۰

در این حالت خاص، خروجی متد را کلمه‌ی کلیدی dynamic قرار دهید. یک مثال

نویسنده: میرزایی
تاریخ: ۱۳۹۳/۰۱/۰۱ ۱۳:۵۱

با سلام و عرض تشکر فراوان به خاطر مطلب مفید تون
سوالی که با خواندن این مطلب برای من پیش آمده اینه که فرض بگیرید بعد از مدتی شما تصمیم می‌گیرید ORM تون رو عوض کنید و تبدیل کنید به Nhibernate ولی شما چون در همه جا از جمله درUI و Application Service از IUnitOfWork استفاده کرده اید چه طور می‌خواهید این کار را انجام دهید
با تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۱ ۱۳:۵۸

- من قصد ندارم چنین تعویضی را انجام دهم. علتش را در اینجا توضیح دادم.
- در لایه UI فقط از اینترفیس‌های لایه سرویس استفاده شده و این لایه از جزئیات پیاده سازی‌ها بی‌اطلاع است. کلاس‌های مورد نیاز از طریق تزریق وابستگی‌ها در اختیار آن قرار می‌گیرند. هر زمان که نیاز به تعویض بود، فقط پیاده سازی‌های لایه سرویس را

تغییر دهید.

نویسنده: behrouz
تاریخ: ۱۳۹۳/۰۱/۰۲ ۲۰:۴۰

سلام
در طراحی لایه سرویس شما کدامیک را پیشنهاد می‌کنید؟
به ازای هر موجودیت در لایه دومین یک کلاس سرویس داشته باشیم
یا
به ازای چندین موجودیت به هم وابسته در دومین یک کلاس سرویس.
آیا استانداردی در این زمینه وجود دارد؟

نویسنده: مجتبی فخاری
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۲:۵۲

با سلام
آیا نحوه کار با StructureMap در VS 2013 و MVC5 با .NET 4.5 متفاوت است؟
آخه من از نوگت StructureMap را نصب نمودم و در فایل global نیز کدهای شما را وارد نمودم ولی در این خط
;(()x.For<IUnitOfWork>().HttpContextScoped().Use(() => new Sample07Context

به HttpContextScoped و در این خط ;()ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects
به ReleaseAndDisposeAllHttpScopedObjects گیر میده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۳:۳

خیر. نگارش سوم structure map تغییراتی داشته که در انتهای مطلب « تزریق خودکار وابستگی‌ها در برنامه‌های ASP.NET Web forms » به صورت یک نکته‌ی تکمیلی مستند شده. مطلب جاری برای نگارش *2.6* تهیه شده بود.

نویسنده: مجتبی فخاری
تاریخ: ۱۳۹۳/۰۱/۲۰ ۰:۳۳

با سلام
الان باید اول structuremap.web , structuremap رو نصب کنم وبعد هم کدهای زیر را در فایل global بنویسم:

```
private static void initStructureMap()
        {
            ObjectFactory.Initialize(x =>
            {
                x.For<IPhoneTypeService>().Use<EFPhoneTypeService>();

                x.Policies.SetAllProperties(y =>
                {
                    y.OfType<IPhoneTypeService>();
                });
            });
            //Set current Controller factory as StructureMapControllerFactory
            ControllerBuilder.Current.SetControllerFactory(new StructureMapControllerFactory());
        }

        protected void Application_EndRequest(object sender, EventArgs e)
        {
            HttpContextLifecycle.DisposeAndClearAll ();
        }
    }

    public class StructureMapControllerFactory : DefaultControllerFactory
    {
        protected override IController GetControllerInstance(RequestContext requestContext, Type
```

```
controllerType)
        {
            return ObjectFactory.GetInstance(controllerType) as Controller;
        }
    }
```

نویسنده: رضایی
تاریخ: ۱۳۹۳/۰۱/۲۴ ۲۳:۱

با سلام؛ موقع اجرا با خطای زیر مواجه می‌شم:

```
StructureMap Exception Code:  202
No Default Instance defined for PluginFamily Iris.Datalayer.Context.IUnitOfWork, baran.Datalayer,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۴ ۲۳:۱۶

«No Default Instance» یعنی در تنظیمات اولیه IoC Container مورد استفاده، برای اینترفیس خاصی، کلاس پیاده سازی کننده‌ای را تعریف نکرده‌اید. برای مشاهده بحث مشابهی در این مورد به نظرات مطلب « تزریق خودکار وابستگی‌ها در برنامه‌های ASP.NET MVC » مراجعه کنید.

نویسنده: سارا کیانی
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۳:۹

سلام آقای نصیری من در حال نوشتن یک پروژه ویندوزی با روشهای گفته شده در این سایت هستم، منظور استفاده از EF Code First و StructureMap و ... می‌باشد. Base برنامه نوشته شده و شامل یک پروژه مرکزی- پروژه حسابداری مالی - حقوق دستمزد- خرید وفروش و.... می‌باشد، کاربران فقط و فقط از طریق اجرای سیستم مرکزی قادر به ورود به سیستم‌های نرم افزاری حوزه خود می‌باشند، با توجه به مطالب فوق الذکر و پرسش و پاسخ‌های کلیه دوستان، متوجه شدم که فقط باید از یک Context استفاده کرد، درسته؟ در حالیکه از بین این چند نرم افزار شاید شرکتی تنها قصد خرید واستفاده سیستم مالی وشرکتی دیگر سیستم مالی و n تای دیگر از برنامه‌ها رو خرید و مورد استفاده قرار دهد، و چون فعلا کلیه عملیات مرتبط با بانک اطلاعاتی در یک Context ودر MainProject انجام میشه، شرکتی که از یک نرم افزار من استفاده خواهد کرد همان ساختار جداول و بانک اطلاعاتی را دارد که شرکتی دیگر از چند نرم افزار دیگر از همین پروژه استفاده میکند. درکل نکته مبهم برام اینست که با چه تکنیک و روشی می‌توان از uow استفاده و پیروی کرد ولی هر پروژه Context خودش رو داشته باشد که با ورود به آن زیرسیستم، عملیات ساخت یا ویرایش DataBase و جداول مرتبط فقط بر روی آن زیر سیستم انجام شود و هیچ تاثیری رو جداول سیستم‌های دیگر نداشته باشد؟ (درضمن کلیه سیستم‌های از یک DataBase استفاده خواهند کرد). با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۳:۱۴

مطلب + نظرات « استفاده از چندین Context در EF 6 Code first »

نویسنده: بهنام
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۴:۶

با عرض سلام و خسته نباشید
من از روش ذکر شده در اجرای یک پروژه استفاده کردم و خیلی مفید بود.
به همین خاطر روش‌های مشابه رو سرچ کردم و به دو تا لینک زیر رسیدم : http://www.prOg33k.com/blog/2003/Getting-Started-MVC-4-Entity-Framework-5-Code-First-and-Unity
http://geekswithblogs.net/danemorgridge/archive/2010/06/28/entity-framework-repository-amp-unit-of-work-t4-template-on.aspx
میخواستم ببینم که آیا این‌ها الگوی واحد کار رو نقض میکنن همونطور که گفته بودید یا خیر؟

و اینکه بین این دو کدوم روش بهتر هست؟ (در صورت عدم نقض الگوی واحد کار)
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۴:۲۴

هیچکدام. قسمت 11 را ابتدا مطالعه کنید (در مورد اینکه چرا نیازی به استفاده از الگوی مخزن با EF نیست).
سپس در سایت به مطالب تکمیلی زیر مراجعه کنید:
به الگوی Repository در لایه DAL خود نه بگویید!
پیاده سازی generic repository یک ضد الگو است
ایجاد Repositories بر روی UnitOfWork
نگاهی به generic repositories
بدون معکوس سازی وابستگی‌ها، طراحی چند لایه شما ایراد دارد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۱ ۹:۶

به روز رسانی
کدهای قسمت جاری را به روز شده جهت استفاده از EF 6 و StructureMap 3 در VS 2013، از اینجا می‌توانید دریافت کنید:
EF_Sample07.zip

نویسنده: ارم وب
تاریخ: ۱۳۹۳/۰۷/۱۲ ۱۶:۱۹

سلام؛ تو این قسمت تمام کارهای که شما گفتید را انجام دادم با Structuremap جوای میگیرم ولی با Ninject فقط اطلاعات اولیه را که در متد Seed وارد کردم نشون میده یعنی برای نمایش جواب میده ولی زمانی که می‌خواهم عملیات درج را انجام بدم انجام نمیده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۲ ۱۷:۵۰

مفاهیم یکی هست. فقط تنظیمات اولیه IoC Containerها متفاوت است. برای Ninject یک افزونه‌ی خاص MVC تهیه شده: در اینجا . پوشه‌ی MVC3 آن تا MVC 5 را هم پوشش می‌دهد. این افزونه یک مثال آماده هم دارد.
ضمنا تنظیم طول عمر یک وهله از UoW در طول یک درخواست توسط متد InRequestScope آن انجام می‌شود.

نویسنده: امیر
تاریخ: ۱۳۹۳/۰۷/۱۳ ۱۸:۴۵

در صورتی که بخواهیم Context را به ازای هر ویندوز فرم زنده نگه داریم در WinForm، پیاده سازی زیر صحیح است:

```
static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Database.SetInitializer<LoanContext>(null);

            // تنظیمات اولیه برنامه که فقط یکبار باید در طول عمر برنامه انجام شود
            ObjectFactory.Initialize(x =>
            {
                x.For<IUnitOfWork>().Use<LoanContext>();
                x.For<IEmployeeService>().Use<EmployeeService>();
```

```
            });

            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmMain());
        }
    }
```

و در frmMain به شکل زیر پیاده سازی را انجام دهیم:

```
public partial class frmMain : Form
    {
        private IContainer _container;
        private IUnitOfWork _uow;

        public frmMain()
        {
            InitializeComponent();

            _container = ObjectFactory.Container.GetNestedContainer();
            _uow = _container.GetInstance<IUnitOfWork>();
        }

        private void btnHomeLoanRequest_Click(object sender, System.EventArgs e)
        {
            var employeeService = _container.GetInstance<IEmployeeService>();
            .
            .
            .
            _uow.SaveChanges();
        }
        private void btnEditLoan_Click(object sender, System.EventArgs e)
        {
            var categoryService = container.GetInstance<ICategoryService>();
            .
            .
            .
            _uow.SaveChanges();
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۳ ۱۹:۲۹

تزریق وابستگی‌های اصولی تا حد امکان از وجود خود IoC Container خالی است (^). یک نمونه پیاده سازی آن برای WinForms: (^)

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۳/۰۷/۱۳ ۲۲:۱۹

باتوجه به توضیح شما، پیاده سازی صحیح‌تر بصورت زیر است؟

```
Program.cs:

Application.Run(ObjectFactory.GetInstance<frmMain>());
```

```
frmMain:
       private IContainer _container;
       private IUnitOfWork _uow;

       public frmMain(IUnitOfWork uow, IContainer container)
       {
           InitializeComponent();

           _container = container;
           _uow = uow;
       }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۳ ۲۲:۵۲

وابستگی‌ها (سرویس‌های مورد نیاز)، از طریق سازنده کلاس دریافت می‌شوند و نیازی نیست از طریق IContainer یا ObjectFactory که در اصل یک وابستگی اضافی داخل کلاسی خاص هستند، دریافت شوند. کلاس‌های شما باید *تا حد امکان* از GetInstance ها خالی باشند. نباید *تا جایی که ممکن است* بالای یک کلاس StructureMap using مشاهده شود (^ و ^) .

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۳/۰۷/۱۴ ۸:۸

یک سوال:
در نگارش سوم Structure map، شما از دستور زیر استفاده کردید:

```
private void btnHomeLoanRequest_Click(object sender, System.EventArgs e)
        {
            using (var container = ObjectFactory.Container.GetNestedContainer()) // کانتکست را به صورت خودکار دیسپوز می‌کند
            {
                var uow = container.GetInstance<IUnitOfWork>();
                var employeeService = container.GetInstance<IEmployeeService>();
```

که با توجه به توضیح شما (^) نیازی به استفاده آن در متد فوق نیست و با تزریق frmMain در program.cs این وابستگی‌ها بوسیله container تزریق و تخریب می‌شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۴ ۹:۱۲

این نکته برای برنامه‌های کنسول و سرویس ویندوز است. در برنامه‌های WinForms و WPF با بسته شدن فرم، به صورت خودکار این اشیاء هم تخریب می‌شوند.

```
public class MyContext : DbContext
    {
        protected override void Dispose(bool disposing)
        {
            Debug.WriteLine("MyContext Dispose() is called.");
            base.Dispose(disposing);
        }
    }
```

برای بررسی بهتر این موضوع، متد فوق را به کلاس Context برنامه اضافه کنید. یک breakpoint روی آن قرار دهید و بعد فرم را باز کرده، با بانک اطلاعاتی کار کنید و سپس فرم را ببندید.

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۳/۰۷/۱۴ ۱۸:۳۴

آیا لازم است در ویندوز فرم‌ها بصورت صریح حین معرفی IUnitOfWork از متد CacheBy با پارامتر InstanceScope.ThreadLocal استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۴ ۱۸:۳۹

« نکته‌ای در مورد مدیریت طول عمر اشیاء در حالت HybridHttpOrThreadLocalScoped در برنامه‌های دسکتاپ »

نویسنده: رضا میر داماد
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۹:۵۰

عرض سلام. در مثالی که قرار دادین در متد جنریک Set در کلاس Context عمل رفع Shadowing انجام شده :

```
public new IDbSet<TEntity> Set<TEntity>() where TEntity : class
```

ولی متد SaveChanges عمل Shadowing نشده!
چرا ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۲۰:۳۱

چون نام متد آن Save **All** Changes است و با نام متد کلاس پایه یکی نیست.

نویسنده: رضا میر داماد
تاریخ: ۱۳۹۳/۰۹/۰۳ ۲۰:۵۶

- اینطوری متد SaveChanges کلاس DbContext در لایه بالاتر(بعنوان مثال DataLayer) قابل دسترسی است , ولی اگر متد SaveChanges را Shadowing کنیم در کلاس Context , لایه بالاتر به متد SaveChanges کلاس DbContext دسترسی ندارد و فقط به متد SaveChanges در کلاس Context دسترسی دارد.
- چرا باید لایه بالاتر به متد SaveChanges کلاس DbContext دسترسی داشته باشد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۲۱:۳۳

- نیازی به بازنویسی SaveChanges اصلی نیست؛ چون تعریف و پیاده سازی آن در کلاس پایه DbContext قرار دارد.
- در اینجا متد جدید SaveAllChanges تعریف شد تا بدانید اگر خواستید پیش از SaveChanges اصلی، مثلا کار اعتبارسنجی دستی را انجام دهید (برای نمونه در برنامه‌های دسکتاپ مثلا)، چگونه می‌توان به آن دسترسی داشت.
- DbContext در DataLayer قرار دارد. زمانیکه با یک ORM کار می‌کنید، خود ORM همان DataLayer شما است. لایه‌ای که از آن استفاده می‌کند (لایه سرویس)، صرفا با اینترفیس IUnitOfWork کار می‌کند و تعاریف موجود در آن و نه چیزی بیشتر از آن.
- زمانیکه با IUnitOfWork کار می‌کنید، در هیچ قسمتی از برنامه قرار نیست مستقیما new MyContext مشاهده شود. تامین این وهله صرفا از طریق تزریق وابستگی‌ها صورت می‌گیرد (کل بحث جاری یا همان پیاده سازی الگوی Context Per Request بر همین مبنا است؛ یک وهله از Context در طول یک درخواست. نه اینکه هر جایی علاقمند بودیم یک new MyContext دستی نوشته شود).