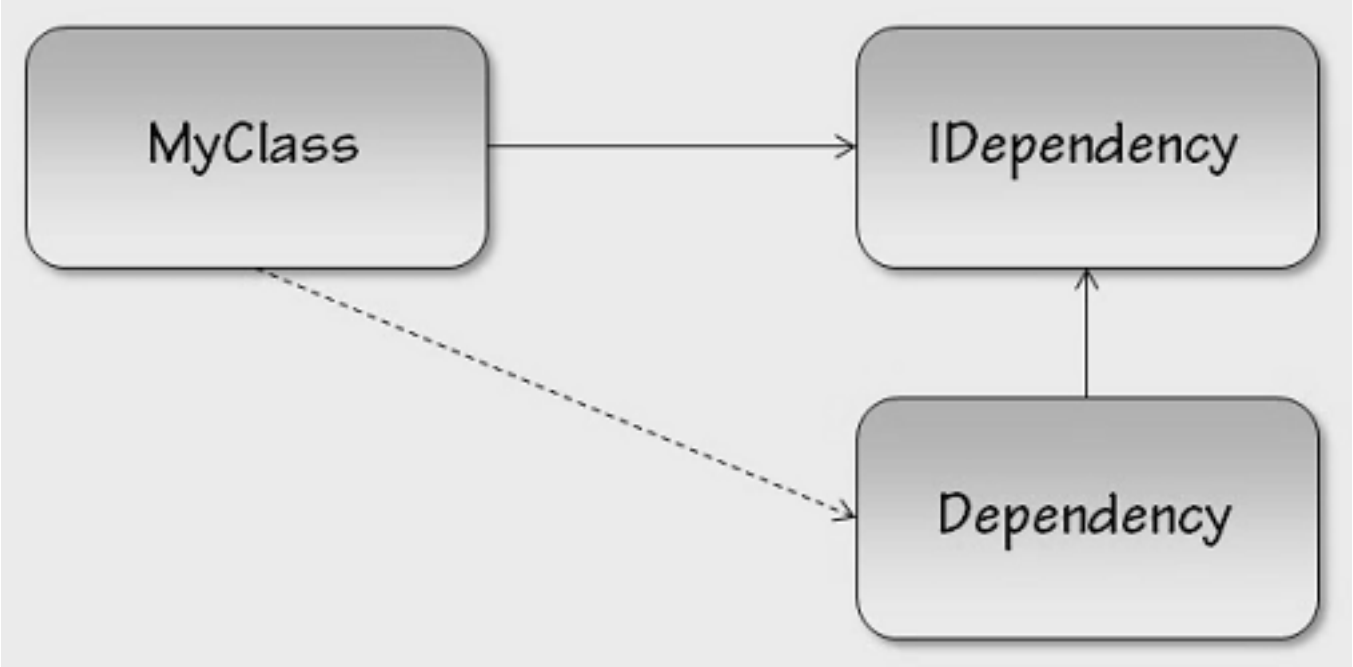
عنوان: **تزریق وابستگی‌ها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۵ ۲۳:۴
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns, Dependency Injection, IoC

خوب! بالاخره بعد از دو قسمت قبل، به بحث اصلی تزریق وابستگی‌ها رسیدیم. بحثی که بسیاری از برنامه نویس‌های تصور می‌کنند همان IoC است که پیشتر در مورد آن بحث شد.

تزریق وابستگی‌ها یا DI چیست؟

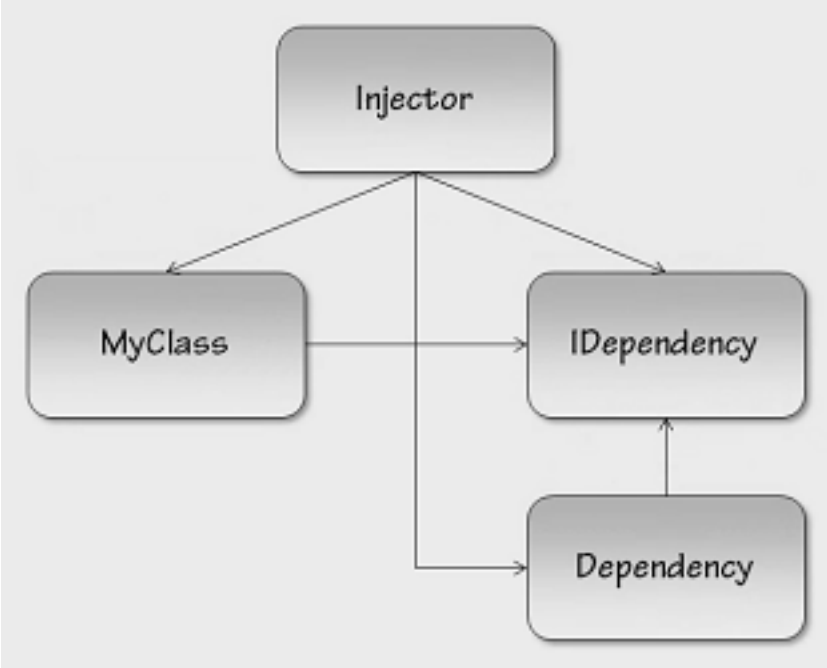
تزریق وابستگی‌ها یکی از انواع IoC بوده که در آن ایجاد و انقیاد (binding) یک وابستگی، در خارج از کلاسی که به آن نیاز دارد صورت می‌گیرد. روش‌های متفاوتی برای ارائه این وابستگی وهله سازی شده در خارج از کلاس به آن وجود دارد که در ادامه مورد بررسی قرار خواهند گرفت.
یک مثال: بسته غذایی را که با خود به سر کار می‌برید درنظر بگیرید. تمام اجزای مورد نیاز تشکیل دهنده یک بسته غذا عموما داخل آن قرار گرفته و حمل می‌شوند. حالت عکس آن زمانی است که در محل کار به شما غذا می‌دهند. این دقیقا همان حالتی است که تزریق وابستگی‌ها کار می‌کند؛ یک سری سرویس‌های خارجی، نیازهای کلاس جاری را برآورده می‌سازند.



در تصویر فوق یک طراحی مبتنی بر معکوس سازی کنترل‌ها را مشاهده می‌کنید. وابستگی‌های یک کلاس توسط اینترفیسی مشخص شده و سپس کلاسی وجود دارد که این وابستگی را پیاده سازی کرده است.
همچنین در این تصویر یک خط منقطع از کلاس MyClass به کلاس Dependency رسم شده است. این خط، مربوط به حالتی است که خود کلاس، مستقیما کار وهله سازی وابستگی مورد نیاز خود را انجام دهد؛ هر چند اینترفیسی نیز در این بین تعریف شده باشد.
بنابراین اگر در بین کدهای این کلاس، چنین کدی مشاهده شد:

```
IDependency dependency= new Dependency();
```

تعریف dependency از نوع IDependency به معنای معکوس سازی کنترل نبوده و عملا همان معماری سابق و متداول بکارگرفته شده است. برای بهبود این وضعیت، از تزریق وابستگی‌ها کمک خواهیم گرفت:



در اینجا یک کلاس دیگر به نام Injector اضافه شده است که قابلیت تزریق وابستگی‌ها را به کلاس نیازمند آن به روش‌های مختلفی دارا است. کار کلاس Injector، وهله سازی MyClass و همچنین وابستگی‌های آن می‌باشد؛ سپس وابستگی را دراختیار MyClass قرار می‌دهد.

تزریق وابستگی‌ها در سازنده کلاس

یکی از انواع روش‌های تزریق وابستگی‌ها، Constructor Injection و یا تزریق وابستگی‌ها در سازنده کلاس می‌باشد که متداول‌ترین نوع آن‌ها نیز به‌شمار می‌رود. در این حالت، وابستگی پس از وهله سازی، از طریق پارامترهای سازنده یک کلاس، در اختیار آن قرار می‌گیرند.
یک مثال:

```
public class Shopper
{
   private readonly ICreditCard _creditCard;
   public Shopper(ICreditCard creditCard)
   {
      _creditCard = creditCard;
   }
}
```

در اینجا شما یک کلاس خریدار را مشاهده می‌کنید که وابستگی مورد نیاز خود را به شکل یک اینترفیس از طریق سازنده کلاس دریافت می‌کند.

```
ICreditCard creditCard = new MasterCard();
Shopper shopper = new Shopper(creditCard);
```

برای نمونه، وهله‌ای از مستر کارتی که ICreditCard را پیاده سازی کرده باشد، می‌توان به سازنده این کلاس ارسال کرد. کار وهله سازی وابستگی و واژه کلیدی new به خارج از کلاس استفاده کننده از وابستگی منتقل شده است. بنابراین همانطور که ملاحظه می‌کنید، این مفاهیم آنچنان پیچیده نبوده و به همین سادگی قابل تعریف و اعمال هستند.

تزریق در خواص عمومی کلاس

Setter Injection و یا تزریق در خواص عمومی یک کلاس، یکی دیگر از روش‌های تزریق وابستگی‌ها است (setter در اینجا منظور همان get و set یک خاصیت می‌باشد).
در حالت تزریق وابستگی‌ها در سازنده کلاس، امکان وهله سازی آن کلاس بدون ارسال وابستگی‌ها به سازنده آن ممکن نیست؛ اما در اینجا خیر و امکان وهله سازی و استفاده از یک کلاس، پیش از اعمال وابستگی‌ها نیز وجود دارد. بنابراین امکان تغییر و تعویض وابستگی‌ها پس از وهله سازی کلاس نیز میسر است.

```
public class Shopper
{
   public ICreditCard CreditCard { get; set; }
}

ICreditCard creditCard = new MasterCard();
Shopper shopper = new Shopper();
shopper.CreditCard = creditCard;
```

نمونه‌ای از این روش را در مثال فوق مشاهده می‌کنید. در این کلاس، وابستگی مورد نیاز از طریق یک خاصیت عمومی دریافت شده است.

تزریق اینترفیس‌ها

تزریق اینترفیس‌ها نیز یکی دیگر از روش‌های تزریق وابستگی‌ها است؛ اما آنچنان استفاده گسترده‌ای برخلاف دو روش قبل نیافته است.
در این روش نه از سازنده کلاس استفاده می‌شود و نه از یک خاصیت عمومی. ابتدا یک اینترفیس که بیانگر ساختار کلاسی که قرار است تزریق شود ایجاد می‌گردد. سپس این اینترفیس را در کلاس وابستگی مورد نظر پیاده سازی خواهیم کرد. در این اینترفیس نیاز است متد خاصی تعریف شود تا کار تزریق وابستگی را انجام دهد.
یک مثال:

```
public interface IDependOnCreditCard
{
   void Inject(ICreditCard creditCard);
}

public class Shopper : IDependOnCreditCard
{
  private ICreditCard creditCard;
  public void Inject(ICreditCard creditCard)
  {
     this.creditCard = creditCard;
  }
}

ICreditCard creditCard = new MasterCard();
Shopper shopper = new Shopper();
((IDependOnCreditCard)shopper).Inject(creditCard);
```

در اینجا یک اینترفیس به نام IDependOnCreditCard تعریف گردیده و متد خاصی را به نام Inject تدارک دیده است که نوعی از کردیت کارد را دریافت می‌کند.
در ادامه کلاس خریدار، اینترفیس IDependOnCreditCard را پیاده سازی کرده است. به این ترتیب کلاس Injector تنها کافی است بداند تا این متد خاص را باید جهت تنظیم و تزریق وابستگی‌ها فراخوانی نماید. این روش نسبتا شبیه به روش Setter injection است.
این روش بیشتر می‌تواند جهت فریم ورک‌هایی که قابلیت یافتن کلیه کلاس‌های مشتق شده از IDependOnCreditCard را داشته و سپس می‌دانند که باید متد Inject آن‌ها را فراخوانی کنند، مناسب است.

نکاتی که باید حین کار با تزریق وابستگی‌ها درنظر داشت

الف) حین استفاده از تزریق وابستگی‌ها، وهله سازی به تاخیر افتاده وابستگی‌ها میسر نیست. برای مثال زمانیکه یک وابستگی قرار است در سازنده کلاسی تزریق شود، وهله سازی آن باید پیش از نیاز واقعی به آن صورت گیرد. البته امکان استفاده از اشیاء Lazy دات نت 4 برای مدیریت این مساله وجود دارد اما در حالت کلی، دیگر همانند قبل و روش‌های متداول، وهله سازی تنها زمانیکه نیاز به آن وابستگی خاص باشد، میسر نیست. به همین جهت باید به تعداد وابستگی‌هایی که قرار است در یک کلاس استفاده شوند نیز جهت کاهش مصرف حافظه دقت داشت.

ب) یکی از مزایای دیگر تزریق وابستگی‌ها، ساده‌تر شدن نوشتن آزمون‌های واحد است. زیرا تهیه Mocks در این حالت کار با اینترفیس‌ها بسیار ساده‌تر است. اما باید دقت داشت، کلاسی که در سازنده آن حداقل 10 اینترفیس را به عنوان وابستگی دریافت می‌کند، احتمالا دچار مشکلاتی در طراحی و همچنین مصرف حافظه می‌باشد.

نظرات خوانندگان

نویسنده: فرشید علی اکبری
تاریخ: ۱۳۹۲/۰۱/۲۶ ۱۱:۵۶

سلام و درود بر شما آقای نصیری

من چون توی دات نت تازه کار هستم دوتا سئوال خدمتتون دارم:

من توی برنامه ای که دارم می‌نویسم از الگوی UnitOfWork مطابق با آموزش‌های شما استفاده کردم درضمن برای تزریق وابستگی هم از StructerMap استفاده می‌کنم، برنامه Win Form هستش وتوی Main یک کلاس Configur ation رو که کارش Register کردن کلیه Interface وکلاس هاست رو فراخونی کردم.

اول اینکه : برای آزاد سازی منابع و استفاده بهینه از حافظه در حال استفاده از StructerMap شما چه پیشنهاد یا روشی رو معرفی می‌کنین؟

دوم اینکه : با ازدیاد و تعدد کلاسها واینترفیس‌ها در حالیکه در ابتدای برنامه کلیه اونها رو با StructerMap رجیستر می‌کنیم ودر هرجا که لازم باشه فقط از اونها یک نمونه میسازیم، اشکالی در روند عملیاتی وکاربری با اون نرم افزار پیش مشتری پیش نمی‌یاد؟ (مخصوصا در سیستم‌های یکپارچه و بزرگ از نظر حافظه).

سوم اینکه: آیا بایدها ونبایدهایی هم در استفاده از StructerMap وجود داره ؟

سپاسگزار شما هستم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۶ ۱۲:۲۴

- کار آزاد سازی منابع را به صورت خودکار GC انجام خواهد داد. فقط وهله سازی در اینجا توسط IoC Container انجام می‌شود. مابقی مسایل مانند قبل است.
- خیر. برای نمونه همین سایت جاری از StructerMap استفاده می‌کند و مشکلی هم از لحاظ میزان مصرف حافظه وجود ندارد. اساسا ایجاد چند شیء در سازنده یک کلاس آنچنان حافظه‌ای را مصرف نمی‌کنند مگر اینکه سازنده‌های این کلاس‌ها دارای تخصیص منابع قابل توجهی باشند. بنابراین سازنده‌های تعریف شده را سبک درنظر بگیرید و طراحی کنید.
- یک قسمت را به StructerMapاختصاص خواهم داد؛ به صورت جداگانه.

نویسنده: فرشید علی اکبری
تاریخ: ۱۳۹۲/۰۱/۲۶ ۱۲:۲۷

سپاسگزار سرعت جوابگوئی شما هستم
امیدوارم هرچه زودتر شاهد مقاله مربوط به StructerMap از شما باشم.

نویسنده: محمد آزاد
تاریخ: ۱۳۹۲/۰۴/۲۱ ۰:۲۸

تزریق وابستگی رو تا چه سطحی باید انجام داد؟ یعنی رعایت کردن اون تو تمام سطوح نرم افزار باید انجام بشه؟ برای مثال کلاس زیر رو در نظر بگیرید که در لایه Entity وجود داره

```
class Parent
{
```

```
    public IChild child {get;set;}
    public Parent(Ichild child)
   {
      this.child =child;
    }
}
```

آیا با اینکه کلاس پدر و فرزند در یک لایه مشترک هستند ، در اینجا ارزش داره که تزریق وابستگی رو انجام بدیم ؟حجم کد و کار بالا بالا نمیره؟ یا اینکه پیچیدگی زیاد نمیشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۱ ۰:۳۴

در دو مطلب زیر به سؤال شما پاسخ داده شده:
- الگوی معکوس سازی کنترل چیست؟ (آیا هرجایی باید اینترفیس تعریف کرد؟ و کجا؟ اصلا این معکوس سازی چی هست و چه هدفی رو دنبال می‌کنه)
- مراحل Refactoring یک قطعه کد برای اعمال تزریق وابستگی‌ها (در کدهای موجود چطور باید این الگو را پیاده سازی کرد و نحوه تشخیص آن به چه صورتی است)

نویسنده: مسعود2
تاریخ: ۱۳۹۲/۰۵/۲۱ ۱۲:۲۰

" حین استفاده از تزریق وابستگی‌ها، وهله سازی به تاخیر افتاده وابستگی‌ها میسر نیست "
فکر کنم منظورتان فقط هنگام تزریق وابستگی از طریق constructor است؟ چون چنانچه از روش تزریق setter استفاده کنیم این مشکل حل میشود. درست برداشت کرده ام؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۲۱ ۱۲:۴۶

- خیر. در حالت setter هم IoC Container (حین استفاده از ابزارهای متداول تزریق وابستگی‌ها) تمام وابستگی‌ها را در حین وهله سازی کلاس ایجاد می‌کند (چه استفاده شوند یا خیر؛ چون اگر اینکار را انجام ندهد استفاده کننده خطای null reference exception را حتما دریافت خواهد کرد. از این جهت که فقط در حالت استفاده از کلاس‌های Lazy است که یک محصور کننده، ارجاع واقعی به شیء را مدیریت می‌کند و به تاخیر می‌اندازد). در حالت به تاخیر افتاده ، یک وابستگی فقط زمانی وهله سازی می‌شود که ارجاعی به آن در مسیر منطقی اجرای برنامه وجود داشته باشد؛ در غیراینصورت از وهله سازی آن وابستگی استفاده نشده، صرفنظر خواهد شد.
- برای آزمایش این مساله یک break point را در سازنده کلاس‌های مورد استفاده قرار دهید.