عنوان: **آشنایی با الگوی Adapter**
نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۲/۰۹/۰۴ ۲۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: Design patterns

قبل از آشنایی با الگوی Adapter،ابتدا با تعریف الگوهای ساختاری آشنا می‌شویم که به شرح ذیل می‌باشد:

الگوهای ساختاری (Structural Patterns):

از الگوهای ساختاری برای ترکیب کلاسها و اشیاء (Objects)،در جهت ایجاد ساختارهای بزرگتر استفاده می‌شود.به بیان ساده‌تر الگوهای ساختاری با ترکیب کلاسها و آبجکتها،قابلیت‌های کلاسهای غیر مرتبط را در قالب یک Interface(منظور ظاهر) در اختیار Client (منظور کلاس یا متد استفاده کننده می‌باشد) قرار می‌دهند.الگوهای ساختاری با استفاده از ارث بری به ترکیب Interfaceها پرداخته و آنها را پیاده سازی می‌نمایند.
استفاده از الگوهای ساختاری برای توسعه کتابخانه هایی (Library) که مستقل از یکدیگر می‌باشند،اما در کنار هم مورد استفاده قرار می‌گیرند،بسیار مفید است.

در ادامه به الگوی Adapter که یکی از الگوهای ساختاری است،می پردازیم. الگوی Adapter انواع مختلفی دارد که فهرست آنها به شرح ذیل می‌باشد:
1- Class Adapter 2- Object Adapter 3- Two way Adapter 4- Pluggable Adapter

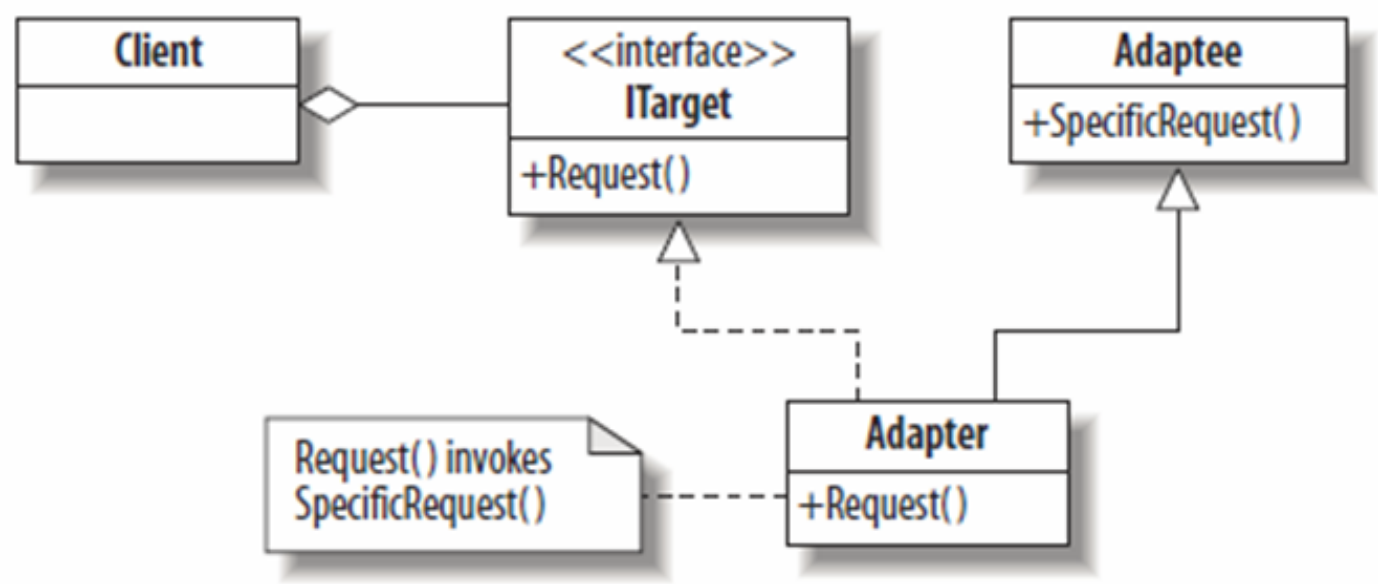
در این مقاله Class Adapter و Object Adapter را مورد بررسی قرار می‌دهیم و اگر عمری باقی باشد در مقاله بعدی Two-way Adapter و Pluggable Adapter را بررسی می‌کنیم.
قبل از پرداختن به هر یک از Adapterها با یکسری واژه آشنا می‌شویم،که در سرتاسر مقاله ممکن است از آنها استفاده شود.
Interface: منظور از Interface در اینجا، ظاهر یا امکاناتی است که یک کلاس می‌تواند ارائه دهد.
Client: منظور متد یا کلاسی است که از Interface مورد انتظار،استفاده می‌نمایید.

Intent (هدف)
هدف از ارائه الگوی Adapter ،تبدیل Interface یک Class به Interface ی که مورد انتظار Client است، می‌باشد.در واقع الگوی Adapter روشی است که بوسیله آن می‌توان کلاسهای با Interface متفاوت را در یک سیستم کنار یکدیگر مورد استفاده قرار داد. به بیان ساده‌تر هرگاه بخواهیم از کلاسهای ناهمگون یا نامنطبق (کلاسهای غیر مرتبط) در یک سیستم استفاده کنیم،راه حل مناسب استفاده از الگوی Adapter می‌باشد.

Adapter را به عنوان Wrapper می‌شناسند.الگوی Adapter از سه Component مهم تشکیل شده است،که عبارتند از:
Target،Adapter و Adaptee.
Target:کلاس یا Interface ی است که توسط Client مورد استفاده قرار می‌گیرد، و Client از طریق آن درخواستهای خود را بیان می‌کند. در واقع Functionality موجود در کلاس Target به جهت پاسخگویی به درخواست‌های Client فراهم گردیده است.
Adaptee: کلاسی است، دارای قابلیتهای مورد نیاز Client بطوریکه Interface اش با Interface مورد انتظار Client (یعنی Target)سازگار نیست. و Client برای استفاده از امکانات کلاس Adaptee و سازگاری با Interface مورد انتظارش نیاز به یک Wrapper همانند کلاسAdapter دارد.
Adapter: کلاسی است که قابلیتها و امکانات کلاس Adaptee را با Interface مورد انتظار Client یعنی Target سازگار می‌کند، تا Client بتواند از امکانات کلاس Adaptee جهت رفع نیازهای خود استفاده نماید. به بیان ساده‌تر Adapter کلاسی هست که برای اتصال دو کلاس نامتجانس (منظور دو کلاسی که هم جنس نمی‌باشند یا از نظر Interface بطور کامل با یکدیگر غیر مرتبط هستند) مورد استفاده قرار می‌گیرد.

در ادامه به بررسی اولین الگوی Adapter یعنی Class Adapter می‌پردازیم:
Class Adapter:

در این روش کلاس Adapter از ارث بری چند گانه استفاده می‌کند و Interface مرتبط به Adaptee را به Interface مرتبط به Target سازگار می‌نماید.
برای درک تعریف بالا مثالی را بررسی می‌کنیم، در ابتدا شکل زیر را مشاهده نمایید:



در شکل ملاحظه می‌کنید، متد SpecificationRequet واقع در Adaptee می‌تواند نیاز Client را برطرف نماید، اما Client،چیزی را که مشاهده می‌کند اینترفیس Itarget می‌باشد، به عبارتی Client بطور مستقیم نمی‌تواند با Adaptee ارتباط برقرار کند، بنابراین اگر بخواهیم از طریق Itarget نیاز Client را برطرف نماییم، لازم است کلاسی بین Itarget و Adaptee به جهت تبادل اطلاعات ایجاد کنیم، که Adapter نامیده می‌شود. حال در روش Class Adapter، کلاس Adapter جهت تبادل اطلاعات بین ITarget و Adaptee هر دو را در خود Implement می‌نماید، به عبارتی از هر دو مشتق (Inherit) می‌شود.
در ادامه شکل بالا را بصورت کد پیاده سازی می‌نماییم.

```
class Adaptee
    {
        public void SpecificationRequest()
        {
            Console.WriteLine("SpecificationRequest() is called");
        }
    }
```

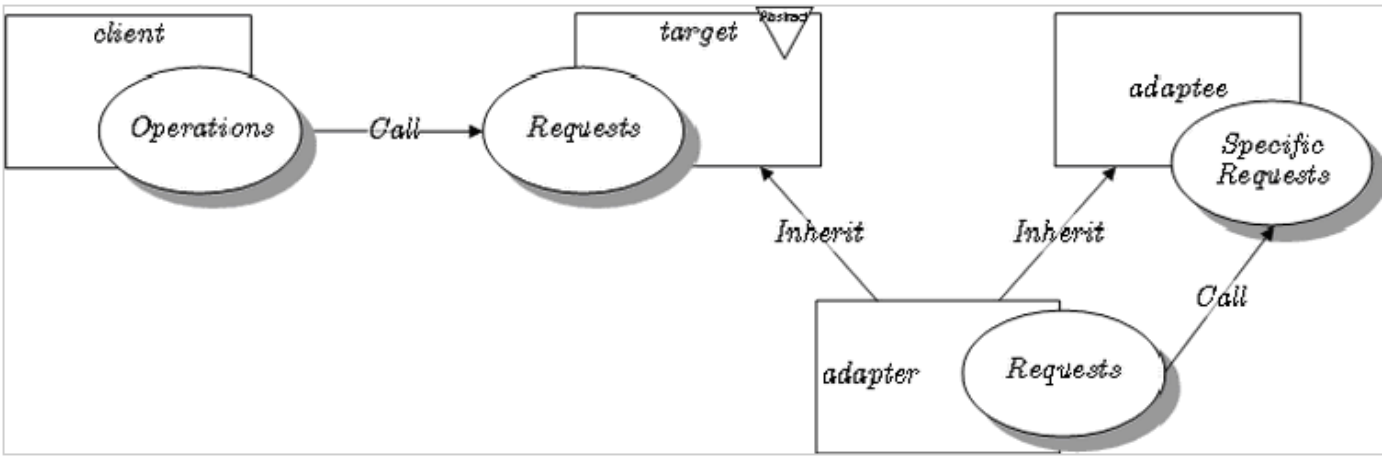
```
interface ITarget
    {
        void Request();

    }
```

```
class Adapter:Adaptee, ITarget
    {
        public void Request()
        {
            SpecificationRequest();
        }
    }
```

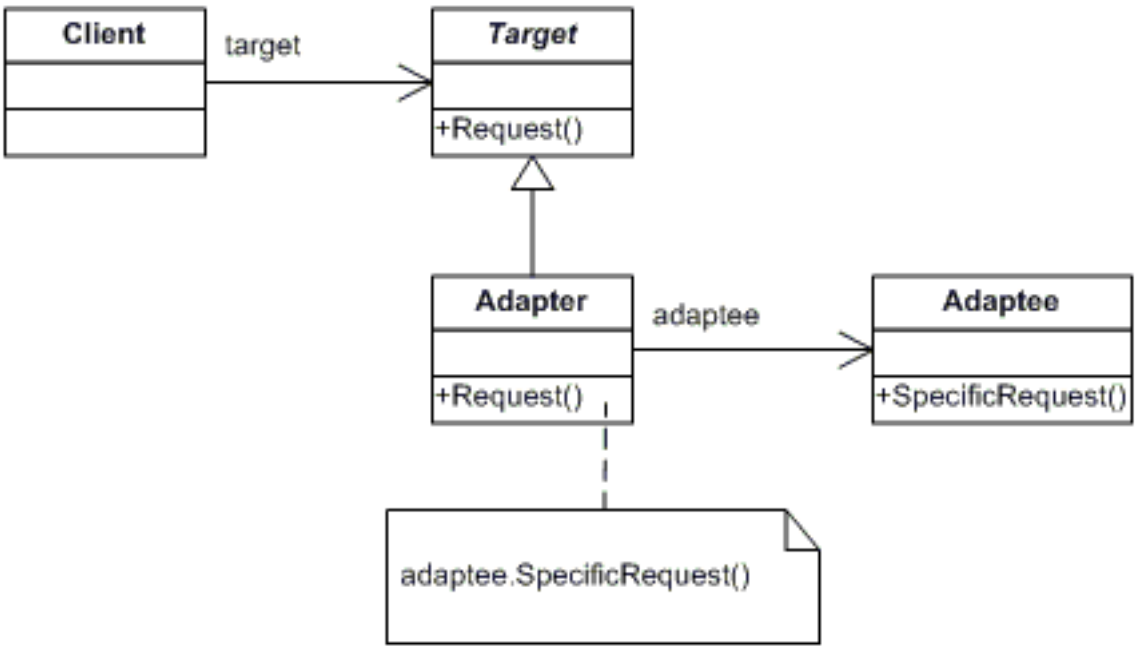
```
class MainApp
{
    static void Main()
    {
        ITarget target = new Adapter();
        target.Request();
```

```
        Console.ReadKey();
    }
}
```

سادگی کد، روش Class Adapter را قابل درک می‌نماید،نکته مهم در کد بالا،متد Request در کلاس Adapter و نحوه فراخوانی متد SpecificationRequest در آن می‌باشد.
شکل زیر که از سایت Wikipedia گرفته شده است،به خوبی نحوه فراخوانی را مشخص می‌نماید:



روش Object Adapter:
می دانیم در زبان برنامه نویسی C# هر کلاس فقط می‌تواند از یک کلاس دیگر Inherit شود، به طوری که هر کلاس نمی‌تواند بیش از یک کلاس Parent داشته باشد، بنابراین اگر Client شما بخواهد از امکانات و قابلیت‌های چندین کلاس Adaptee استفاده نماید، روش Class Adapter نمی‌تواند پاسخگوی نیازتان باشد، بلکه می‌بایست از روش Object Adapter استفاده نمایید.
شکل زیر بیانگر روش Object Adapter می‌باشد:



همانطور که در شکل ملاحظه می‌کنید، در این روش کلاس Adapter به جای Inherit نمودن از کلاس Adaptee، آبجکتی از کلاس Adaptee را در خود ایجاد می‌نماید، بنابراین با این روش شما می‌توانید به چندین Adaptee از طریق کلاس Adapter دسترسی داشته باشید.

پیاده سازی کدی شکل بالا به شرح ذیل می‌باشد:

```
class Adaptee
    {
        public void SpecificRequest()
        {
            MessageBox.Show("Called SpecificRequest()");
        }
    }
```

```
interface ITarget
    {
        void Request();

    }
```

```
class Adapter: ITarget
    {
        private Adaptee _adptee = new Adaptee();

        public void Request()
        {
            _adptee.SpecificationRequest();
        }
    }
```

```
class MainApp
{
    static void Main()
    {
        ITarget target = new Adapter();
        target.Request();

        Console.ReadKey();
```

```
    }
}
```

برای درک تفاوت Class Adapter و Object Adapter ، پیاده سازی کلاس Adapter را مشاهده نمایید، که در کد بالا به جای Inherit نمودن از کلاس Adaptee ، آبجکت آن را ایجاد نمودیم. واضح است که Object Adapter انعطاف پذیرتر نسبت به Class Adapter می‌باشد.

امیدوارم مطلب فوق مفید واقع شود

نظرات خوانندگان

نویسنده: حسین کهزادی
تاریخ: ۱۳۹۲/۱۱/۱۴ ۱۵:۸

باتشکر از مطلب بسیار خوبتون
اگر مطلب رو مانند الگوی composite با یک مثال ساده و کوچک توضیح میدادید بسیار قابل فهم‌تر میشد