عنوان: **نحوه پیاده سازی عملیات Undo و Redo با استفاده از الگوی طراحی Command**
نویسنده: وحید فرهمندیان
تاریخ: ۱۳۹۳/۱۱/۲۹ ۱۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: Design patterns

اگر با الگوهای طراحی آشنا باشید، یکی از مناسب‌ترین الگوهای طراحی برای پیاده سازی عملیات Undo و Redo استفاده از الگوی طراحی Command هست (مطالعه بیشتر).
در این الگو یک کلاینت دارم که مشخص می‌کند چه کاری قرار است انجام شود. یک Command داریم که می‌گوید هر کاری را چه کسی انجام دهد و یک Receiver داریم که می‌گوید هر کاری چطور انجام می‌شود. **قدم اول:** کلاینت می‌خواهد عملیات Undo و Redo انجام شود. من اضافه‌بر این دو عملیات، عملیات Execute را هم اضافه می‌کنم. پس کلاینت می‌خواهد که سه کار Undo و Redo و Execute را انجام دهد.

```
public class Client
    {
        public delegate string Invoker();
        public static Invoker Execute;//اضافه کردن یک آیتم جدید
        public static Invoker Redo;//حرکت به جلو
        public static Invoker Undo;//حرکت به عقب
    }
```

قدم دوم: Command باید مشخص کند که هر کاری را چه کسی باید انجام دهد:

```
public class Command
    {
        public Command(Receiver receiver)
        {
            Client.Execute = receiver.Action;
            Client.Redo = receiver.Foreward;
            Client.Undo = receiver.Reverse;
        }
    }
```

Command در سازنده‌ی خود ورودی از نوع Receiver دارد (در ادامه پیاده سازی خواهد شد) و در واقع می‌خواهد کارها را به Receiver محول نماید. **قدم سوم:** بایدمشخص شود هر کاری قرار است چگونه انجام شود:

```
public class Receiver
    {
        private readonly List<string> build = new List<string>();
        private readonly List<string> oldBuild = new List<string>();

        public string Action()
        {
            if (build.Count > 0)
                oldBuild.Add(build.LastOrDefault());
            build.Add(build.Count.ToString(CultureInfo.InvariantCulture));
            return build.LastOrDefault();
        }

        public string Reverse()
        {
            string last = oldBuild.LastOrDefault();
            if (last == null)
                return "EMPTY";
            oldBuild.Remove(last);
            return last;
        }

        public string Foreward()
        {
            string oldIndex = oldBuild.LastOrDefault();
            int index = oldIndex == null ? -1 : build.IndexOf(oldIndex);
            if ((index + 1) == build.Count)
                return "END";
            oldBuild.Add(build.ElementAt(index + 1));
            return oldBuild.LastOrDefault();
        }
    }
```

اگر روش بهتری برای پیاده سازی Undo و Redo و Execute دارید، میتوانید جایگزین کنید. این اولین روشی بود که به ذهنم رسید!

قدم‌های لازم برای پیاده کردن الگوی Command تا اینجا به پایان می‌رسند. حالا کافی‌است از آن استفاده کنیم:

```
new Command(new Receiver());
            Console.WriteLine(Client.Execute());
            Console.WriteLine(Client.Execute());
            Console.WriteLine(Client.Undo());
            Console.WriteLine(Client.Undo());
            Console.WriteLine(Client.Undo());
            Console.WriteLine(Client.Redo());
            Console.WriteLine(Client.Redo());
            Console.WriteLine(Client.Redo());
            Console.WriteLine(Client.Execute());
```

در این روش ما از delegate استفاده کردیم و به کمک آن یک واسط را بین کلاینت و Command ساختیم (Invoker). مطالعه بیشتر در مورد delegate