عنوان: **آشنایی با الگوی طراحی Bridge**
نویسنده: وحید فرهمندیان
تاریخ: ۱۳۹۳/۱۲/۰۳ ۱۹:۱۵
آدرس: www.dotnettips.info
گروه‌ها: C#, OOP, DesignPatterns

سناریو زیر را در نظر بگیرید:
قصد دارید تا در برنامه‌ی خود ارسال پیام از طریق پیامک و ایمیل را راه اندازی کنید. هر کدام از این روش‌ها نیز برای خود راه‌های متفاوتی دارند. برای مثال ارسال پیامک از طریق وب سرویس یا یک API خارجی و غیره.
کاری را که می‌توان انجام داد، بشرح زیر نیز می‌توان بیان نمود:
ابتدا یک Interface ایجاد می‌کنیم (IBridge) و در آن متد Send را قرار می‌دهیم. این متد یک پارامتر ورودی از نوع رشته می‌گیرد و به کمک آن میتوان اقدام به ارسال پیامک یا ایمیل یا هر چیز دیگری نمود. کلاس‌هایی این واسط را پیاده سازی می‌کنند که یکی از روش‌های اجرای کار باشند (برای مثال کلاس WebService که یک روش ارسال پیامک یا ایمیل است).

```
public interface IBridge
    {
        string Send(string parameter);
    }
    public class WebService: IBridge
    {
        public string Send(string parameter)
        {
            return parameter + " sent by WebService";
        }
    }
    public class API: IBridge
    {
        public string Send(string parameter)
        {
            return parameter + " sent by API";
        }
    }
```

سپس در ادامه به مکانیزمی نیاز داریم تا بتوانیم از طریق آن پیامک یا ایمیل را ارسال کنیم. خوب می‌خواهیم ایمیل ارسال کنیم؛ اولین سوالی که مطرح می‌شود این است که چگونه ارسال کنیم؟ پس باید در مکانیزم خود زیرساختی برای پاسخ به این سوال آماده باشد.

```
public abstract class Abstraction
    {
        public IBridge Bridge;
        public abstract string SendData();
    }
    public class SendEmail : Abstraction
    {
        public override string SendData ()
        {
            return Bridge.Send("Email");
        }
    }
    public class SendSMS: Abstraction
    {
        public override string SendData ()
        {
            return Bridge.Send("SMS");
        }
    }
```

در کد فوق یک کلاس انتزاعی ایجاد کردیم و در آن یک object از نوع واسط خود قرار دادیم. این object به ما کمک می‌کند تا به طریق آن شیوه‌ی ارسال ایمیل یا پیامک را مشخص سازیم و به سوال خود پاسخ دهیم. سپس در ادامه متد SendData آورده شده است که به کمک آن اعلام می‌کنیم که قصد ارسال ایمیل یا پیامک را داریم و نهایتا هر یک از کلاس‌های ایمیل یا پیامک، این متد را برای خود پیاده سازی کرده‌اند.

قبل از ادامه اجازه دهید کمی در مورد بدنه‌ی یکی از متدهای SendData صحبت کنیم. در این متد با کمک Bridge متد Send موجود

در واسط صدا زده شده است. از آنجا که این object از نظر سطح دسترسی عمومی می‌باشد، لذا از بیرون از کلاس قابل دسترسی است. این باعث می‌شود تا قبل از فراخوانی متد SendData موجود در کلاس ایمیل یا پیامک اعلام کنیم که Bridge از چه نوعی است (به چه روشی می‌خواهیم ارسال رخ دهد).

```
Abstraction ab1 = new Email();
  ab1.Bridge = new WebService();
  Console.WriteLine(ab1.SendData ());
```

```
ab1.Bridge = new API();
  Console.WriteLine(ab1.SendData ());
```

```
Abstraction ab2 = new SMS();
   ab2.Bridge = new WebService();
   Console.WriteLine(ab2.SendData ());
```

```
ab2.Bridge = new API();
    Console.WriteLine(ab2.SendData ());
```

نهایتا در کد فوق ابتدا بیان می‌کنیم که قصد ارسال ایمیل را داریم. سپس اعلام می‌داریم که این ارسال را به کمک WebService می‌خواهیم انجام دهی. و نهایتا ارسال را انجام می‌دهیم.

به کل این الگویی که ایجاد کردیم، الگوی Bridge گفته می‌شود.

حال فکر کنید قصد ارسال MMS دارید. در اینصورت فقط کافیست یک کلاس MMS ایجاد کنید و تمام؛ بدون اینکه کدی اضافی را بنویسید یا برنامه را تغییر دهید. یا فرض کنید روش ارسال جدیدی را می‌خواهید اضافه کنید. برای مثال ارسال به روش XYZ. در اینصورت فقط کافیست یک کلاس XYZ را ایجاد کنید که IBridge را پیاده سازی می‌کند.