عنوان: **الگوی وضعیت State Pattern**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۵/۰۱/۲۵ ۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: Design patterns

الگوی وضعیت، یکی از الگوهای رفتاری Gang Of Four است و بسیار شبیه به الگوی Strategy می‌باشد؛ ولی با کپسوله سازی بیشتر. در الگوی استراتژی تغییر وضعیت از بیرون کلاس اعمال می‌د ولی در الگوی وضعیت، بر اساس تغییر وضعیت درونی خودش صورت می‌گیرد.
یکی از استفاده‌های این الگو برای مثال در پلیرهاست که وضعیت پخش را چون Play,Pause و ... در خود دارند. در اینجا هم از این مثال استفاده می‌کنیم:
ابتدا یک اینترفیس برای وضعیت خود بسازید که آرگومان ورودی متد آن را در مرحله بعد تعریف میکنیم:

```
public interface IState
{
    void PressPlay(MP3PlayerContext context);
}
```

سپس نوبت ایجاد کلاس اصلی یا همان دستگاه پخش که به آن Context می‌گوییم می‌رسد تا تغییر وضعیت الگو را به آن بسپاریم:

```
public class MP3PlayerContext
{
    public MP3PlayerContext()
    {
        this.CurrentState = new StandbyState();
    }

    public MP3PlayerContext(IState state)
    {
       this.CurrentState = state;
    }

    public IState CurrentState { get; set; }

    public void Play()
    {
        this.CurrentState.PressPlay(this);
    }
}
```

سپس کلاس‌های مختلف خود را بر اساس اینترفیس بالا می‌سازیم:

```
public class StandbyState : IState
{
    public void PressPlay(MP3PlayerContext context)
    {
        context.CurrentState = new PlayingState();
    }
}

public class PlayingState : IState
{
    public void PressPlay(MP3PlayerContext context)
    {
        context.CurrentState = new StandbyState();
    }
}
```

در کدهای بالا، کلاس‌های Playing و StandBy در واقع شبیه سازی از عمل کلید پخش هستند که با هر بار فشردن آن، پخش به طور موقت توقف کرده و یا پخش خود را از سر می‌گیرد. کلاس Context نیز باید در ابتدا به طور پیش فرض با یکی از این مقادیر پر شود و برای دکمه پخش مشخص است که کلاس PlayingState می‌باشد.
بدین ترتیب در اولین اجرای متد Play در کلاس Context، کلاس PlayingState اجرا می‌شود و وضعیت، به StandbyState تغییر می‌کند و هر بار که مجددا متد Play اجرا گردد، تعویض بین این دو کلاس صورت می‌گیرد.