02

Introduzir Extensão Local

Transcrição

Para fechar esse capítulo, falaremos sobre a técnica de refatoração **Introduzir Extensão Local**:

Abriremos o Visual Studio mais uma vez, acessaremos o projeto `refatoracao`. Em "Aula08 > R17.IntroduceLocalExtension > depois", temos o arquivo `Exemplo.cs`.

Encontraremos também a classe `Exemplo` que armazenará a funcionalidade das regras de negócio da aplicação. Vemos que existem três métodos para obter o *último dia do mês*, o *primeiro dia do mês* e um outro para saber se *é fim de semana* a partir de uma data.

```
public Exemplo()
{
    var data = DateTime.Today;
    var ultimoDiaDoMes = UltimoDiaDoMes(data);
    var primeiroDiaDoMes = PrimeiroDiaDoMes(data);
    var ehFimDeSemana = EhFimDeSemana(data);
}
```

Os três métodos recebem um `DateTime` como argumento, e também podemos perceber que já foi aplicada a refatoração feita anteriormente, referente à introdução ao método estrangeiro. Entretanto, esses métodos estão se acumulando na classe `Exemplo`. Poderíamos organizá-los melhor se estes fossem extraídos para uma classe que fosse extensão do `DateTime`.

Chamamos a esse tipo de classe de **extensão local**, na qual estenderemos a classe `DateTime` para melhorar a organização. No fim do mesmo arquivo, criaremos uma classe. Quando criamos uma *classe extensão*, geralmente colocamos o nome da classe **alvo** e, colocamos a palavra **Extensions** no final:

```
class Exemplo{...}

static class DateTimeExtensions
{
```

```
}
```

Essa nova classe foi marcada como *static* e, em seguida copiaremos para dentro dela os *métodos estrangeiros* que estão na classe `Exemplo`. Depois de colado, ainda faremos com que eles sejam públicos:

```
static class DateTimeExtensions
{
    public DateTime PrimeiroDiaDoMes(DateTime data)
    {
        return new DateTime(data.Year, data.Month, 1);
    }

    public DateTime UltimoDiaDoMes(DateTime data)
    {
        return new DateTime(data.Year, data.Month, DateTime.DaysI.......);
    }

    public bool EhFimDeSemana(DateTime data)
    {
        return data.DayOfWeek == DayOfWeek.Saturday
                || data.DayOfWeek == DayOfWeek.Sunday;
    }
}
```

O próximo passo é marcá-los como *static*.

```
    public static DateTime PrimeiroDiaDoMes(DateTime data){...}

    public static DateTime UltimoDiaDoMes(DateTime data){...}

    public static bool EhFimDeSemana(DateTime data){...}
```

Agora podemos utilizar esse código a partir da classe `Exemplo`. Acessaremos o `UltimoDiaDoMes` como se esse método fizesse parte do `DateTime`. No lugar do parâmetro `data`, não vamos passar nada.

```
    var ultimoDiaDoMes = data.UltimoDiaDoMes();
```

Ao realizarmos essa mudança em todo o trecho, o código ficará assim:

```
public Exemplo()
{
    var data = DateTime.Today;
    var ultimoDiaDoMes = data.UltimoDiaDoMes();
    var primeiroDiaDoMes = data.PrimeiroDiaDoMes();
    var ehFimDeSemana = data.EhFimDeSemana();
}
```

Entretanto, o Visual Studio ainda está reclamando pois ele ainda não os reconhece como se fossem parte do `DateTime`. Será preciso fazer uma outra mudança no código da classe de extensão, que é *marcar* o parâmetro `DateTime` com a

palavra `this` . Assim será reconhecido que estamos estendendo a classe `DateTime` :

```
public static DateTime PrimeiroDiaDoMes(this DateTime data){...}

public static DateTime UltimoDiaDoMes(this DateTime data){...}

public static bool EhFimDeSemana(this DateTime data){...}
```

Podemos remover os três métodos da classe `Exemplo` e ela ficará muito mais clara:

```
class Exemplo
{
    public Exemplo()
    {
        var data = DateTime.Today;
        var ultimoDiaDoMes = data.UltimoDiaDoMes();
        var primeiroDiaDoMes = data.PrimeiroDiaDoMes();
        var ehFimDeSemana = data.EhFimDeSemana();
    }
}
```

Inclusive, conseguimos colocar a classe `DateTimeExtensions` em uma outra biblioteca para que a aplicação fique ainda mais organizada.