01

Extrair Variável

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://github.com/alura-cursos/csharp-refatorando-codigo/archive/cd427ca8e8906feed44bd954ba103de632c722b7.zip) completo do projeto do capítulo anterior e continuar seus estudos.

Veremos a terceira técnica de refatoração: **extrair variável**. Para isso, veremos como o projeto `caelum-stella-csharp` faz a avaliação de documentos, como o CPF, o CNPJ e o Título Eleitoral. Todos esses documentos têm um formato, uma validação específica e dois dígitos verificadores. Eles também compartilham uma regra de validação, que é feita no projeto `caelum-stella-csharp` .

No Visual Studio, dentro do projeto `caelum-stella-csharp` , encontraremos a pasta `Validation` . Dentro dela teremos três classes: `CNPJValidator.cs` , `CPFValidator.cs` e `TituloEleitoralValidator.cs` . Essas classes herdam de uma classe base chamada `BaseValidator.cs` , que abriremos a seguir.

Localizaremos o método que faz a validação desses documentos, o `GetInvalidValues()` . Esse método retorna uma *lista de strings* de todos os possíveis erros de validação do documento que está sendo validado. No final desse mesmo método, encontramos uma condição um pouco complicada.

```
if (unformattedDocument != documentSubstring
        + GetDigitoVerificador(documentSubstring).ToString()
        + GetDigitoVerificador(documentSubstring + GetDigitoVerificador(documentSubstring).
        errors.Add(DocumentError.InvalidCheckDigits);
```

Esse código compara um documento formatado, com um outro concatenado com os dígitos verificadores. Agora, modificaremos o código para facilitar a leitura e facilitar a manutenção ou para adição de alguma funcionalidade. Quebraremos essa expressão complexas em expressões mais simples. Para isso, usaremos a refatoração chamada **Extração de Variáveis**.

Perceberemos que há dois pedaços idênticos nesse trecho. Para encontrá-los, usaremos o atalho "Ctrl + F" e colamos a expressão `GetDigitoVerificador(documentSubstring).ToString()` e assim faremos a busca. Repare que o Visual Studio encontrará a mesma expressão logo na linha abaixo. Isso indica que resolveremos a duplicação extraindo-a para uma variável local.

Depois de selecionar o trecho e fazer a pesquisa, o Visual Studio nos mostrou uma lâmpada do lado esquerdo para nos auxiliar. Essa lâmpada nos traz algumas opções de refatoração. A primeira delas extrairemos para uma variável somente dessa expressão que selecionamos. Na segunda, a extração ocorrerá em todas as ocorrências dessa expressão. Vamos escolher a segunda.

Assim como fizemos com o *extrair método,* o Visual Studio também pede para renomear a nova variável. Como nós já conhecemos o código, sabemos que essa expressão representa o primeiro digito verificador do documento. Essa variável se chamará `digito1` .

```
string digito1 = GetDigitoVerificador(documentSubstring).ToString();
    if (unformattedDocument != documentSubstring
            + digito1
            + GetDigitoVerificador(documentSubstring + digito1).ToString())
            errors.Add(DocumentError.InvalidCheckDigits);
```

Automaticamente, a expressão foi substituída pela variável `digito1`. Sabemos também que a segunda concatenação representa o segundo dígito verificador. Vamos selecionar o trecho `GetDigitoVerificador(documentSubstring + digito1).ToString()` e clicar na lâmpada que aparece à esquerda.

O próximo passo será escolher a opção na qual a extração será feita em todas as ocorrências da expressão. Novamente, daremos o nome de `digito2` para essa variável.

Conseguimos resolver a duplicação de código. Repare que o `if ()` ficou bem mais simples:

```
string digito1 = GetDigitoVerificador(documentSubstring).ToString();
string digito2 = GetDigitoVerificador(documentSubstring + digito1).ToString();
    if (unformattedDocument != documentSubstring
            + digito1
            + digito2)
            errors.Add(DocumentError.InvalidCheckDigits);
```

Este `if()` verifica se os dois dígitos verificadores dos documentos batem com aquilo que foi calculado pelo algoritmo do `BaseValidator.cs`. Depois de refatorado, rodaremos os testes para verificar se nenhuma funcionalidade foi quebrada com essa refatoração.

Clicamos no menu "Test > Run > All Tests" e aguardamos. Ao lado esquerdo, será aberto um painel de testes, e veremos que todos passaram com sucesso.