04

Validando CNPJ e Título Eleitoral

Transcrição

Agora que fizemos a validação do CPF, você pode se perguntar: "E se eu quisesse fazer a validação utilizando o valor booleano?" ou até mesmo: "Se eu não quisesse tratar uma exceção para saber se o CPF é válido ou não?" Então, um `try catch` não seria necessário.

Felizmente a biblioteca **Stella** tem o método `IsValid()` que faz essa validação Booleana e retorna um bool.

Cuidado! Se fazemos isso, precisamos mudar a nossa estratégia, não podemos usar o `try catch`, e sim um `if e else`.

Então, faremos assim:

```
if (new CPFValidator().IsValid(cpf))
{
    Debug.WriteLine("CPF válido: " + cpf);
}
else
{
    Debug.WriteLine("CPF inválido: " + cpf);
}
```

Agora sim! CPF válido ou inválido dependendo do resultado do retorno do método `IsValid()`. Rodaremos a aplicação para ver como ele se comporta.

```
CPF válido: 86288366757
CPF inválido: 98745366797
CPF inválido: 22222222222
```

Como podemos ver, não é apresentado para nós alguma mensagem de erro, somente se o CPF é válido ou não.

Agora, faremos a validação de dois tipos de documento: o **CNPJ** e o **Título Eleitoral**.

O CNPJ (Código Nacional de Pessoa Jurídica) identifica empresas, instituições, e outros tipos de empresa. Utilizaremos a mesma biblioteca **Stella** para a validação.

Para validar alguns CNPJs, precisamos primeiro entrar em um site que gera esses CNPJs (http://www.geradordecnpj.org/) para nós.

Vamos copiar o CNPJ gerado, e depois, colaremos em uma variável em nosso projeto que a chamaremos de "cnpj1".

```
static void Main(string[] args)
{
    string cpf1 = "86288366757";
    string cpf2 = "98745366797";
    string cpf3 = "22222222222";

    ValidarCPF(cpf1);
    ValidarCPF(cpf2);
    ValidarCPF(cpf3);

    string cnpj1 = "51241758000152";
}
```

Pegaremos mais um CNPJ que chamaremos de "cnpj2":

```
static void Main(string[] args)
{
    string cpf1 = "86288366757";
    string cpf2 = "98745366797";
    string cpf3 = "22222222222";

    ValidarCPF(cpf1);
    ValidarCPF(cpf2);
    ValidarCPF(cpf3);

    string cnpj1 = "51241758000152";
    string cnpj2 = "14128481000120";
}
```

Para fazer a validação dos CNPJs, seguiremos com a mesma estratégia utilizando uma nova classe de validador: `CNPJValidator()`.

```
    string cnpj1 = "51241758000152";
    string cnpj2 = "14128481000120";

    new CNPJValidator().IsValid(cnpj1);
```

Só que esse `IsValid()`, retorna um booleano. Por isso, essa linha deve ficar dentro de um bloco `if`.

```
string cnpj1 = "51241758000152";
string cnpj2 = "14128481000120";

if (new CNPJValidator().IsValid(cnpj1))
{

}
```

Dentro do `if`, entraremos em uma condição para exibir uma mensagem dizendo que o CNPJ é válido:

```
if (new CNPJValidator().IsValid(cnpj1))
{
    Debug.WriteLine("CNPJ válido: " + cnpj1);
}
else
{
    Debug.WriteLine("CNPJ inválido: " + cnpj1);
}
```

Muito bem! Antes de testar o segundo CNPJ, temos que extrair o método `CNPJValidator`. Selecionando todo o bloco, clicando com o direito, clicamos em "Quick Actions and Refactorings...", e por fim clicaremos em "Extract Method"! Vamos chamar esse método de `ValidarCNPJ()`.

```
static void Main(string[] args)
{
    string cpf1 = "86288366757";
    string cpf2 = "98745366797";
    string cpf3 = "22222222222";

    ValidarCPF(cpf1);
    ValidarCPF(cpf2);
    ValidarCPF(cpf3);

    string cnpj1 = "51241758000152";
    string cnpj2 = "14128481000120";

    ValidarCNPJ(cnpj1);
    ValidarCNPJ(cnpj2);

}

private static void ValidarCNPJ(string cnpj)
{
    if (new CNPJValidator().IsValid(cnpj))
    {
        Debug.WriteLine("CNPJ válido: " + cnpj);
    }
    else
    {
        Debug.WriteLine("CNPJ inválido: " + cnpj);
```

```
        }
    }
```

Rodando a aplicação novamente, temos os dois CNPJs válidos no *output*.

Veremos como validar um título eleitoral. Para isso, será necessário gerar (http://4devs.com.br/gerador_de_titulo_de_eleitor) um novo código para título eleitoral.

Entrando no link acima, clicamos em "Gerar Título de Eleitor", e copiamos o número que foi gerado. Vamos armazená-lo em uma variável `string titulo1`. Depois pegaremos mais um título de eleitor.

```
    string titulo1 = "041372570132";
    string titulo2 = "774387480132";
```

Criaremos um novo validador:

```
    string titulo1 = "041372570132";
    string titulo2 = "774387480132";

    new TituloEleitoralValidator().IsValid(titulo1);
```

Como você pode imaginar, envolveremos esse novo validador em um `if`.

```
    if (new TituloEleitoralValidator().IsValid(titulo1))
    {
        Debug.WriteLine("Título válido: " + titulo1);
    }
    else
    {
        Debug.WriteLine("Título inválido: " + titulo1);
    }
```

Se o "titulo1" for válido ou inválido, vamos exibir no console. Vamos rodar a aplicação para ver se o nosso título será validado ou não.

Legal! Foi mostrado para nós que o título é **válido**. Para testar o segundo título, faremos a refatoração e extração do método.

Selecionamos todo o `if`, "Quick Actions and Refactorings..." > "Extract Method" para um método chamado de `ValidarTitulo()`.

```
{
    // main()...
    string titulo1 = "041372570132";
    string titulo2 = "774387480132";

    ValidarTitulo(titulo1);
    ValidarTitulo(titulo2);
}
```

```
private static void ValidarTitulo(string titulo)
{
    if (new TituloEleitoralValidator().IsValid(titulo))
    {
        Debug.WriteLine("Título válido: " + titulo);
    }
    else
    {
        Debug.WriteLine("Título inválido: " + titulo);
    }
}
```

Muito bem! No console aparecerá para nós que os dois títulos são válidos! E nesse vídeo, aprendemos a utilizar a API *Stella* da Caelum, validando CPFs, CNPJs e Títulos Eleitorais. Mais pra frente, veremos como formatar documentos e mais coisas interessantes dessa biblioteca.