04

Entendendo melhor o Async

Transcrição

Algo muito legal de notarmos antes de continuarmos o projeto do ByteBank é uma dúvida muito comum entre os que estão estudando o `AsyncAwait`: por que é necessário adicionar um modificador à função, se o `await` (uma palavra chave, reservada) por si só já é forte o suficiente para indicar ao compilador que as variáveis `inicio`, `resultado` e `fim` precisam fazer parte daquele "jogo" com *task* que encadeia outra, recuperando o contexto...? Por quê o modificador `Async` é necessário?

Vamos pensar em um código antigo que não tem nada a ver com o ByteBank, simulando-o em um C# `4`, em que não existia `AsyncAwait` ainda. Usaremos outra função antiga, que retornava o método `AlgumWifiAleatorioInjetandoTrojans()`. Neste exemplo, o programa faz análises de redes que, ao ver um perigo cria um alerta.

```
private void CodigoAntigo()
{
  var await = AlgumWifiAleatorioInjetandoTrojans();
  await.ToString();
}

private object AlgumWifiAleatorioInjetandoTrojans()
{
  return null;
}
```

Utilizando este código, como criamos uma variável para armazená-lo? Para isto, costumamos usar as letras capitais, as maiúsculas. Neste caso, teríamos portanto a `await`. Ué, `await` não é uma palavra reservada? Uma palavra chave do C# `5`? Quer dizer que este código criado em minha empresa, sendo executada, em produção, desenvolvida com C# `4`, não poderá ser rodado em C# `5`?

Sim, é possível executá-lo! Uma grande preocupação do time da Microsoft na construção do C# é mantê-lo sempre **retrocompatível**, o que significa que se escrevemos um código com C# `1`, `2` ou `3`, em uma linguagem antiga, ele será executado da mesma forma, mantendo-se o mesmo comportamento das versões subseguintes.

O compilador (Visual Studio), até pinta de azul a palavra `await`, pois não se trata de uma palavra chave apenas, e sim uma palavra chave contextual, de uma função assíncrona. E nós a marcamos com este modificador `async`. No código antigo criado acima, ao tentarmos incluir `async` para tentarmos utilizar o recurso do *array* e a facilidade de assincronicidade do C# `5`, não será possível manter variáveis com o nome `await`, teremos que mudar este código.

No entanto, o importante é que código feito em versões antigas não quebra nas mais recentes. Legal? Vamos para o próximo vídeo para continuarmos melhorando o ByteBank.