THE DEVELOPER'S CONFERENCE

O que esperar do C# 7

Tania Raquel Stormovski de Andrade

Software developer at Trinca

Globalcode

THE DEVELOPER'S CONFERENCE

Repositório da apresentação

https://github.com/TeteStorm/TDC2016Presentation

Contato

email: tanstormandrade@gmail.com

linkedin: https://br.linkedin.com/in/taniastormovski

Globalcode

Objetivo da palestra

THE DEVELOPER'S CONFERENCE

O objetivo principal é apresentar brevemente algumas das novas features previstas para contemplar o C# 7 e que atualmente já estão "bem aceitas" que farão parte da nova release.

Brevemente iremos abordar qual foi o intento de cada umas das versões do C# disponíveis até hoje e também comentar um pouquinho sobre o que guiou o pensamento da equipe de design do C# para chegar a essas features selecionadas.

Globalcode

Agenda

THE DEVELOPER'S CONFERENCE

- **C# 7 visão geral e status da versão**
- **Ambiente de execução**
  VS 15 Preview 4 e o novo Visual Studio Installer
- **Breve overview das versões do C#**
- **Principais focos da nova versão da linguagem**
  - Consumo de dados
  - Aumento de performance
  - Simplificação de código
- **Apresentando algumas novas features:**
  - Pattern Matching
  - Tuples
  - Desconstruction
  - Local Functions
  - Ref return and Locals
  - Out var

Globalcode

Futuro do C#

THE DEVELOPER'S CONFERENCE

A próxima versão da linguagem está sendo projetada com código aberto, os comentários e discussões dos membros da equipe e da comunidade. À medida que vão sendo testadas e validadas o time determina quais recursos devem ser adicionados à linguagem, o legal disso tudo é o acompanhamento real time que podemos ter da evolução das features, bem como as idéias que por alguma razão vão sendo abandonadas.

Globalcode

Documento oficial do status da linguagem no github acessível em

https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md

THE DEVELOPER'S CONFERENCE

Language Feature Status

This document reflects the status, and planned work, for the compiler team. It is a live document and will be updated as work progresses, features are added / removed, and as work on feature progresses.

C# 7.0 and VB 15

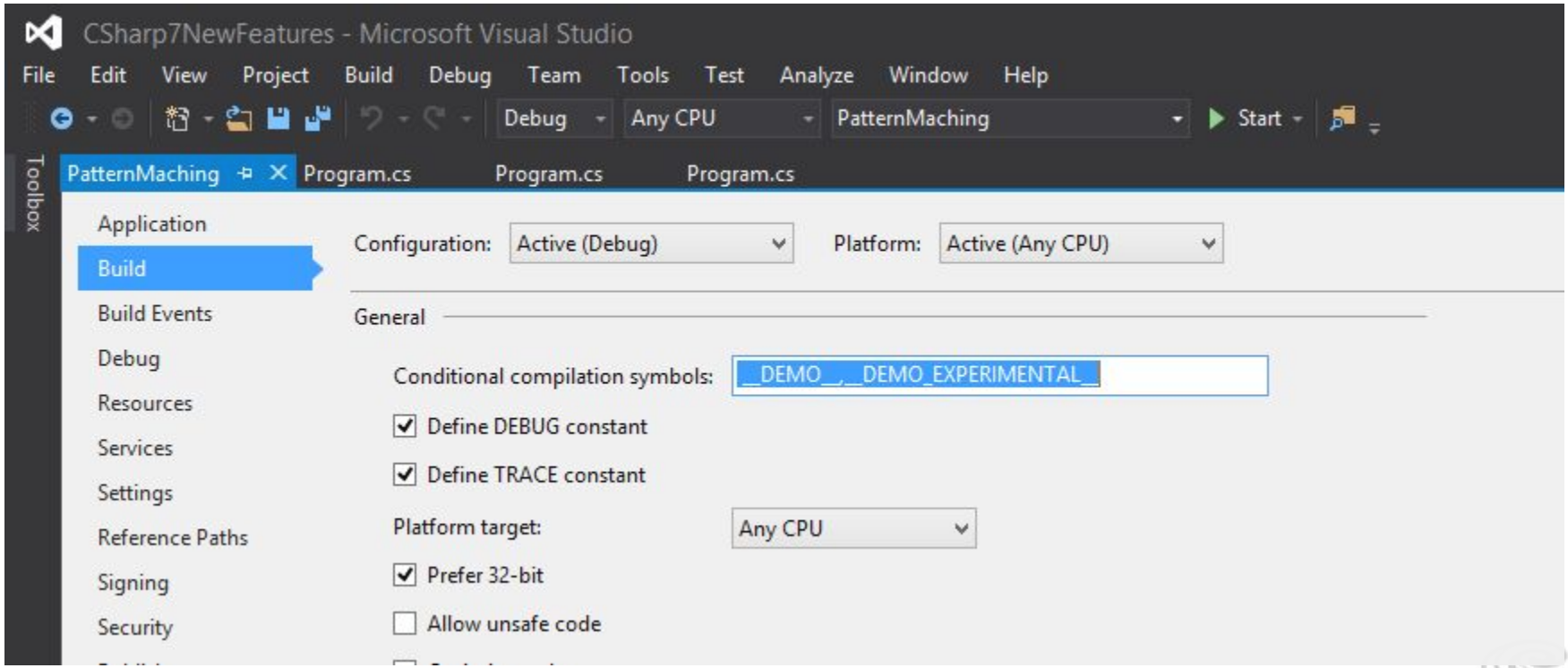
Feature	Branch	State	Owners	LDM Champ
Binary Literals	master	Finishing		gafter
Digit Separators	master	Finishing		gafter
Local Functions	master	Finishing	agocke, jaredpar, vsadov	gafter
Type switch	master	Finishing	gafter, alekseyts, agocke	gafter
Ref Returns	master	Finishing	vsadov, agocke, jaredpar	vsadov
Tuples	master	Finishing	vsadov, jcouv	madstorgersen
Out var	master	Finishing	alekseyts	gafter
ValueTask	master	Finishing	alekseyts	lucian
Throw Expr	features/throwexpr	Prototyping	gafter, agocke, tyoverby	gafter
Expression-Bodied Everything	features/exprbody	Prototyping	MgSam, gafter	madstorgersen

Globalcode

Ambiente de execução

THE DEVELOPER'S CONFERENCE

Até antes do lançamento da release do Visual Studio 15 preview 4, que aconteceu dia 22 de agosto de agosto dependíamos de configurações adicionais, no caso símbolos de compilação condicionais, para testarmos as novas features propostas no C#. Agora com o advento da nova release grande parte das features finalizadas já estão disponíves sem precisarmos dos simbolos condicionais.



Globalcode

VS 2015 Preview 4 - Nova experiência de instalação

THE DEVELOPER'S CONFERENCE

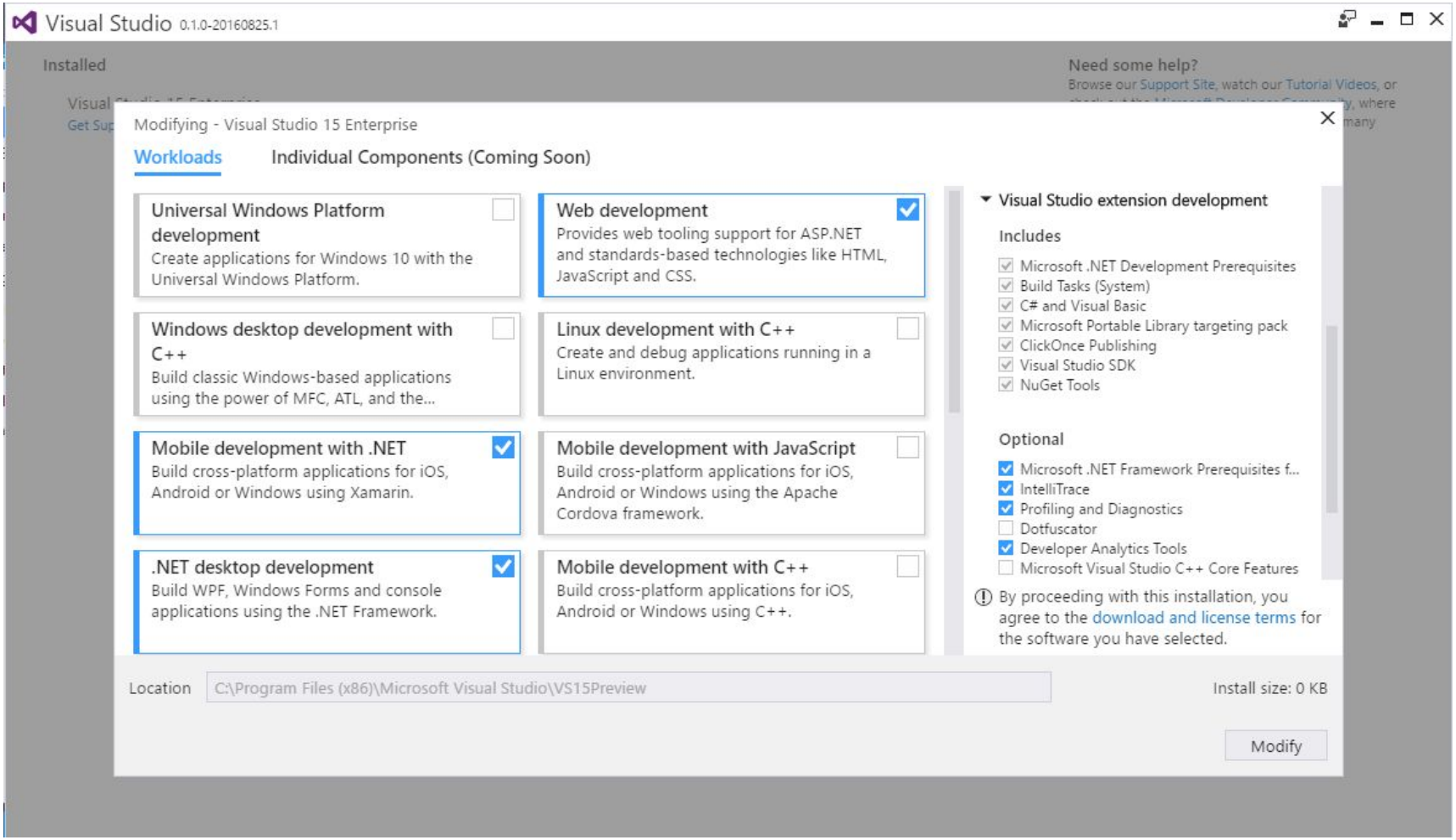
Como primeiro impacto nos deparamos com um nova experiência de instalação. É um proposta diferenciada das versões anteriores promete ser uma versão mais leve e otimizada..

Conforme a Microsoft a nova versão tem os seguintes principais propósitos:

- **Reduzir o volume mínimo de memória do Visual Studio.**
- **Instalar mais rapidamente e com menos impacto no sistema, e desinstalar de forma limpa.**
- **Facilitar a seleção e instalação apenas dos recursos necessários.**

Globalcode

New Visual Studio Installer



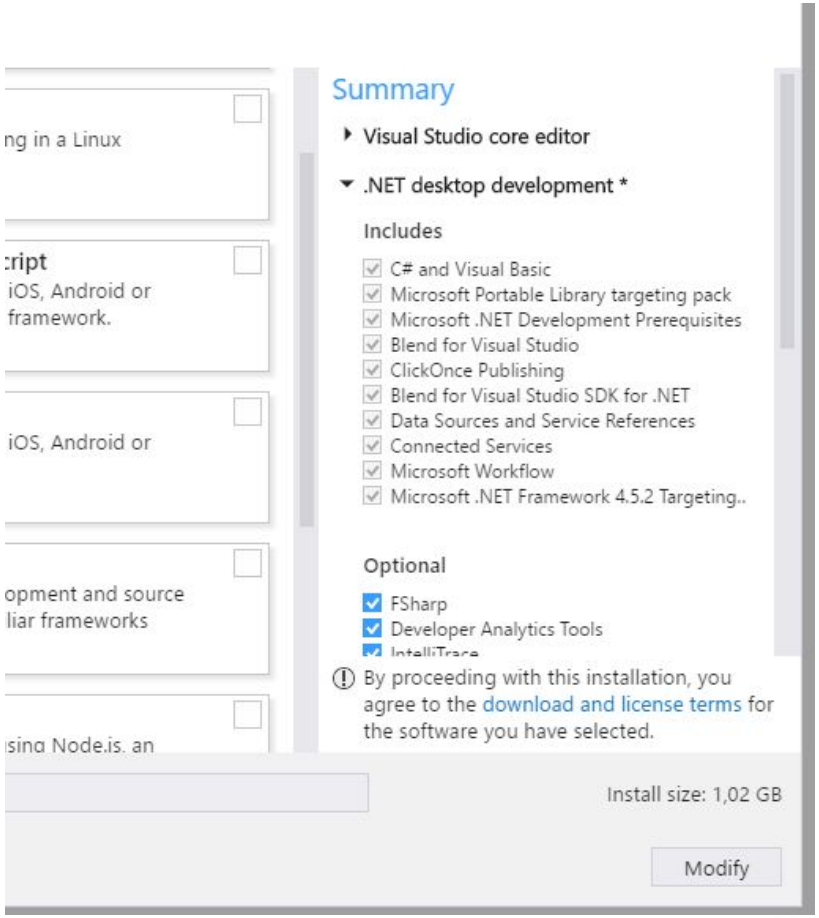
New Visual Studio Installer

THE DEVELOPER'S CONFERENCE

A instalação mais básica, que contempla somente o core editor para execução via shell tem 418 MB.

Se adicionarmos o pacote básico para desktop completo a instalação pula para 1,02 GB.

É possível e bem fácil selecionar itens adicionais e ver tudo que está sendo instalado



Globalcode

Versões do C#

THE DEVELOPER'S CONFERENCE

As novas funcionalidades em cada versão do C #, até agora (com excepção do C # 6.0 talvez) giraram em torno de um tema específico:

C # 2.0 introdução do generics.

C # 3.0 habilitar o LINQ como extension method, expressões lambda, tipos anônimos (anonymous types) e outros recursos relacionados.

C # 4.0 interoperabilidade com linguagens dinâmicas e não tipadas.

C # 5.0 simplificar a programação assíncrona com advento do async e await.

C # 6.0 teve o seu compilador completamente reescrito do zero e introduziu uma variedade de pequenos recursos e melhorias que ficaram mais fáceis de ser implementadas agora na versão 7..

Globalcode

THE DEVELOPER'S CONFERENCE

De acordo com o leader de design do C#, Mads Torgersen o foco das novas features adicionadas na versão são:

"Data consumption, code simplification and performance."

Vamos ver brevemente o que guiou o time de design da linguagem nesse sentido:

Globalcode

Consumo de dados

THE DEVELOPER'S CONFERENCE

Conforme a equipe de desenvolvimento da linguagem aqui se concentram as maiores features da nova versão do C#. Foi pensando na demanda cada vez maior do uso de serviços na web e na necessidade de facilitar o trabalho com interfaces de dados externos várias novas features foram desenvolvidas para ajudar nesse sentido como **Pattern matching** e **Tuples, Descontruction**.

Globalcode

Performance

THE DEVELOPER'S CONFERENCE

Principalmente o aumento da participação de dispositivos móveis foi o que fez do desempenho uma consideração importante novamente. Existem recursos planejados para C # 7.0 que permitem otimizações de desempenho, que anteriormente não eram possíveis no framework .NET tais como **Local functions**, **Ref Returns** e **Out var**

Globalcode

Simplificação do código

THE DEVELOPER'S CONFERENCE

Aqui entra uma evolução da versão anterior, com várias pequenas alterações adicionais construídas sobre o trabalho feito para C # 6.0 para permitir uma maior simplificação do código escrito como **Binary Literals** e **Digit Separator**.

Globalcode

Pattern Matching

THE DEVELOPER'S CONFERENCE

C # 7.0 introduz a noção de padrões, que, abstratamente falando, são elementos sintáticos que pode testar se um valor tem uma certa "forma" ou padrão, e extrair informações a partir do valor. Neste momento, estão restritos a valores constantes, tipos e var .Pode ser utilizado de duas maneiras: utilizados em conjunto com o operador “is” e nos “case”s de uma declaração “switch.

Pode ter restrições com when nos cases e vale lembrar no switch: a ordem importa, o default vai ser sempre avaliado por último e o null no final não é unreachable, pode ser tratado.

Globalcode

Constant pattern

THE DEVELOPER'S CONFERENCE

```
  public static void CalculateBodyFat(BodyFatSkinfoldProtocol bodyFatProtocol)
 {
        decimal bodyFat = 0;
        var bodyFatPercent = 0D;
        //Utilização do "is"
        // Constant pattern
        // Antes podíamos validar se o valor era nulo ou se o valor era de um determinado tipo
        // if (bodyFatProtocol == null) { ... }
        // if (bodyFatProtocol is BodyFatSkinfoldProtocol) { ... }
        // Agora podemos validar diretamente em constantes (ou números)
        if (bodyFatProtocol is null)
              throw new ArgumentNullException("Protocol must not be null");
}
```

Globalcode

Type pattern

THE DEVELOPER'S CONFERENCE

```
public static void CalculateBodyFat(BodyFatSkinfoldProtocol bodyFatProtocol)
{
     decimal bodyFat = 0;
    // Type pattern
     // Aqui além de testar o tipo extraímos o valor em uma nova variável do tipo proposto
     if (bodyFatProtocol is Guedes protocol)
     {
          //{ ... }
          bodyFat = protocol.Height;
     }
     else if (bodyFatProtocol is Faulkner protocol)
     {
          //{ ... }
          bodyFat = protocol.Height;
     }
 }
```

Globalcode

Var pattern

THE DEVELOPER'S CONFERENCE

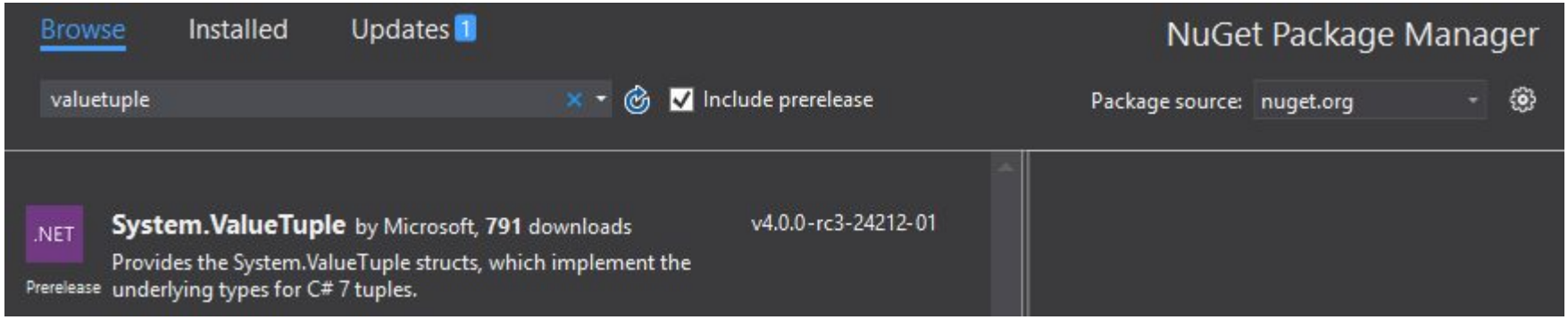
```
public static void CalculateBodyFat(BodyFatSkinfoldProtocol bodyFatProtocol)
{
     decimal bodyFat = 0;

     // Var pattern
     // Aqui sempre vai dar match, a comparação simplemente coloca o valor obtido
      //numa nova variável que será do tipo do input
     if (bodyFatPercent is var bf)
     {
          Console.WriteLine(bf);
     }
 }
```

Globalcode

Tuples e Descontruction

THE DEVELOPER'S CONFERENCE

Para utilizarmos ambos recursos é necessário instalarmos o pacote do nuget System.ValueTuple



Globalcode

Tuples e Descontruction

THE DEVELOPER'S CONFERENCE

O novo conceito de Tupla inserido no C# 7 não tem a ver com o atual tipo tupla que se trata de um reference type, e claro vamos querer evitar usá-lo em códigos onde desempenho seja imprescindível. Se tratando de um tipo imutável qualquer mudança vai exigir que realoquemos um novo objeto.

O C# 7 oferece um novo tipo de tupla como valuer type, mutável e mais eficiente em termos de performance.

A ideia básica é permitir declarar tipos como uma tupla para que uma declaração possa conter mais de um valor e, de forma semelhante, os métodos possam retornar mais de um valor.

O caso mais comum é para permitir múltiplos valores de retorno de uma função sem precisar de parâmetros "out "(**tuple types)**. Mas também teremos literais de tuplas (**tuple literals)**, permitindo criar uma variável tipada como uma tupla.

Tuples

THE DEVELOPER'S CONFERENCE

```
// Retorna 3 valores dentro de um elemento tupla que podem ser acessados atráves das propriedades
 Item1, Item2, Item3
static (decimal, decimal, string) GetProtocolValues(BodyFatSkinfoldProtocol bodyFatProtocol)
{
   return (bodyFatProtocol.Weight, bodyFatProtocol.BodyFatPercent, bodyFatProtocol.ProtocolName);
}

 // Retorna 3 valores dentro de um elemento tupla mas de maneira nomeada com variáveis explícitas
 // que poderão ser acessadas atráves dos nomes das propriedades
 static (decimal Weight, decimal TotalBodyFat, string Protocol)
 GetNamedProtocolValues(BodyFatSkinfoldProtocol bodyFatProtocol)
 {
    var returnValues = (bodyFatProtocol.Weight, bodyFatProtocol.BodyFatPercent,
 bodyFatProtocol.ProtocolName);
    return returnValues;
 }
```

Globalcode

Descontruction

THE DEVELOPER'S CONFERENCE

Outra maneira de consumir tuplas é utilizando o conceito de deconstruct, nada mais é do que uma sintaxe para dividir uma tupla (ou outro objeto, não é uma funcionalidade exclusiva para o uso com tuplas) em suas partes e atribuir as partes individualmente para variáveis.

Globalcode

Descontruction

THE DEVELOPER'S CONFERENCE

```
public static void Deconstruct(this BodyFatSkinfoldProtocol bodyFatProtocol, out string
  protocolName, out string formuleDescription)
{
     protocolName = bodyFatProtocol.ProtocolName;
     formuleDescription = bodyFatProtocol.FormuleDescription;
}
 // Automaticamente chama o método ou extension method Desconstruct implementado no objeto
public static void PrintDesconstruction()
{
            var guedes = new Guedes() { Height = 167, Weight = 58 };
            (var name, var description) = guedes; //calls method Deconstruct
            Console.WriteLine($"Protocol name: {name}");
            Console.WriteLine($"Protocol description: {description}");
            string x;
            guedes.Deconstruct(out x);
            Console.WriteLine($"Protocol and description: {x}");
        }
```

Globalcode

Local functions

THE DEVELOPER'S CONFERENCE

Permite a declaração de funções auxiliares aninhados dentro de outras funções. Isto não só irá reduzir o seu escopo, mas também permite o uso de variáveis declaradas em no escopo principal, sem alocar memória adicional na pilha.

Globalcode

THE DEVELOPER'S CONFERENCE

```
public  void CalculateBodyFatMass(decimal weight, decimal height)
        {
            if (weight <= 0)
                throw new NullReferenceException("weight cannot be null");
            if (height <= 0)
                throw new NullReferenceException("height cannot be null");

            void CalculateBodyFat()
            {
                this.TotalBodyFat = weight + (height * height) /2;
            }
            CalculateBodyFat();
        }

}
```

Globalcode

Ref Returns and locals

THE DEVELOPER'S CONFERENCE

Esta proposta adiciona suporte em C # para devolver valores a partir de métodos de referência. Além disso, as variáveis locais podem ser declaradas como variáveis "ref". Um método poderia retornar uma referência para uma estrutura de dados interna em vez de retornar uma cópia do valor.

Globalcode

Ref Returns and locals

THE DEVELOPER'S CONFERENCE

```
public class BodyWeight
  {
      public BodyWeight(int currentWeight)
      {
          _weight = currentWeight;
      }
      private int _weight = 0;
      public ref int GetWeight()
      {
          return ref _weight;
      }
  }
```

Globalcode

Ref Returns and locals

THE DEVELOPER'S CONFERENCE

```
BodyWeight myWeight = new BodyWeight(58);
// Mostra valor atual e incrementa 2kg usando a maneira padrão de incrementar
// que é método Increment
Console.WriteLine("Current weight value: ");
Console.WriteLine(myWeight.GetWeightValue());
Console.WriteLine("Increment 2 kilogramans using the public method Increment");
myWeight.Increment(2);
Console.WriteLine("Current weight value after increment: ");
Console.WriteLine(myWeight.GetWeightValue());

// Pega a propriedade disponibilizada por referência
// e modifica diretamente a propriedade interna do objeto
ref int teste = ref myWeight.GetWeight();
// incrementa em 1 kg utilizando diretamente a prop weight obtida através do método ref int GetWeight();
teste++;
Console.WriteLine("Current weight value after external increment: ");
Console.WriteLine(myWeight.GetWeightValue());
Console.ReadKey();
```

Globalcode

Out var

THE DEVELOPER'S CONFERENCE

Remove a necessidade de variáveis out serem pré declaradas e também permite o uso de var para declarar o tipo da variável que será inferido pelo compilador (não funciona em casos onde o compilador não consegue inferir a tipagem por causa de algum overload)

Globalcode

Out var

THE DEVELOPER'S CONFERENCE

```
// Deconstruct in new Out mode
        public static bool DesconstructionInOutVariables(out string name, out string description)
        {
            var guedes = new Guedes() { Height = 167, Weight = 58 };
            (name, description) = guedes; //calls method Deconstruct
            return true;
        }
```

Chamada:

```
         //Como ainda existe restrição de escopo para conseguir acessar as variávies precisamos
  criar um para poder acessá-las
            if(DesconstructionInOutVariables(out string name, out string description))
            {
                Console.WriteLine($"Protocol name: {name}");
                Console.WriteLine($"Protocol description: {description}");
            }
```

Globalcode

Out var

THE DEVELOPER'S CONFERENCE

```
// Out Var
  public static void PrintOutVars(string weight)
  {
      if (int.TryParse(weight, out var intWeight))
      {
          Console.WriteLine(new string('*', intWeight));
      }
  }
```

Globalcode