Alunos: Jaime Lay e Lucas Dunhão de Carvalho
Professor: Francisco Sant’anna

Introdução

- **Desenvolvida no ano 2000 por um time de desenvolvedores da Microsoft liderada por Anders Hejlsberg.**
- **Faz parte da plataforma .NET**
  - Plataforma única de desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET pode ser executado em qualquer dispositivo que possua tal framework.
- **Baseado em linguagens como C, C++, JAVA e VB.**
- **Possui código fonte compilado para Common Intermediate Language (CIL) e que é interpretada pela máquina virtual Common Language Runtime(CLR).**

Classificação

- **Orientada a Objetos**
- **Fortemente Tipada**
- **Fácil de aprender mas complexa.**
- **Linguagem de propósito geral**
  - **Desenvolvimento de Jogos**
  - **Desenvolvimento Web**
  - **Desenvolvimento Mobile**
  - **IoT**
  - **entre outros….**

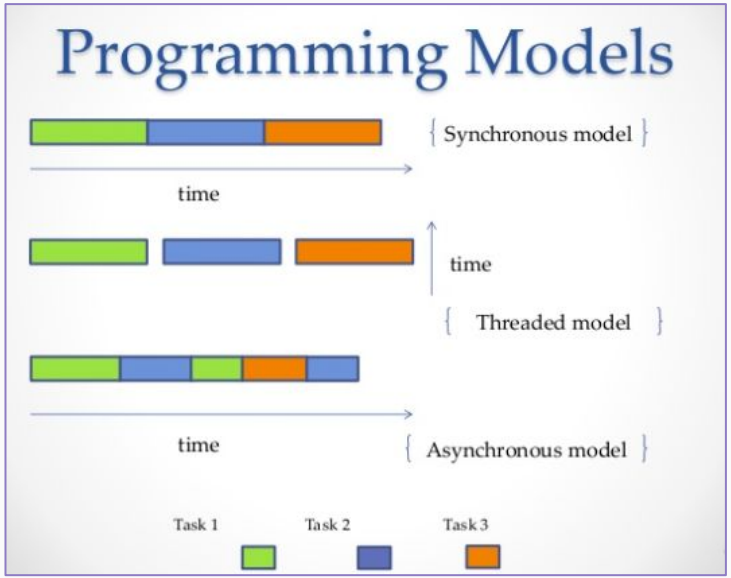
Funcionalidade: Garbage Collection

É um recurso que oferece o gerenciamento de memória. Com ele é possível recuperar uma área inutilizada por um programa,evitando problemas de vazamento. Em linguagens que não tem o GC como C e C++, a memória dinâmica é alocada e desalocada explicitamente.

No caso do C#, toda vez que você cria um novo objeto, o CLR aloca memória para ele. Quando não há nenhuma referência à esse objeto, o GC recupera a parte da memória que estava sendo usada.

Funcionalidade: Async/Await

Para podermos fazer tarefas simultaneamente, sem que o nosso programa pare e aguarde cada tarefa ser finalizada para realizar a próxima tarefa, temos que usar funções assíncronas. Como por exemplo, quando fazemos uma requisição a algum serviço web, ela certamente é mais demorada que qualquer operação local, e se ela for feita de maneira síncrona a página ficará congelada enquanto a resposta não chegar. Cada linguagem tem uma maneira diferente de implementar o assincronismo, e o C# utiliza Tasks e Async/Await.

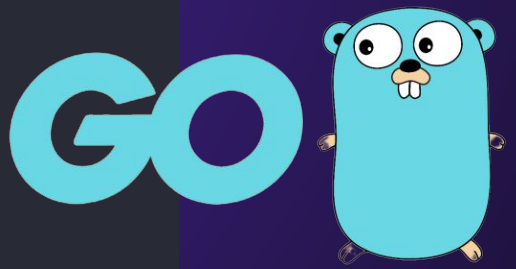


```csharp
using System;
using System.Net.Http;
using System.Threading.Tasks;

namespace codigos{
    0 references
    class Program{
        2 references
        private const string URL = "https://docs
        0 references
        static void Main(string[] args){
            doSyncTask();
            var someTaskAsync = doAsyncTask();
            doSyncTaskWhileAwait();
            someTaskAsync.Wait(); // Isso é uma "blocking call", faz com que o bloco não termine até a Tarefa ser realizada.
        }
        1 reference
        static void doSyncTask(){ Console.WriteLine("1. Fazendo alguma tarefa síncrona..."); }
        1 reference
        static async Task doAsyncTask(){
            Console.WriteLine("2. Tarefa assíncrona começou...");
            await getTotalSizeOfContent();
        }
        1 reference
        static async Task getTotalSizeOfContent(){
            using (var httpClient = new HttpClient()){
                string result = await httpClient.GetStringAsync(URL); // execução para aqui até o GetStringAsync finalizar
                // Apartir dessas linhas abaixo, a execução só irá acontecer após o término da requisição
                Console.WriteLine($"3. O tamanho do conteúdo da página {URL} é de {result.Length} caracteres");
            }
        }
        1 reference
        static void doSyncTaskWhileAwait(){
            Console.WriteLine("4. Enquanto a tarefa assíncrona não é terminada, podemos fazer outras tarefas simultaneamente...");
            for (var i = 1; i <= 3; i++)
                Console.WriteLine($"Tarefa {i} realizada");
        }
    }
}
```

```
PS C:\Users\Jaime\Desktop\Github\UERJ\EDL\artigo_csharp\codigos> dotnet run
1. Fazendo alguma tarefa síncrona...
2. Tarefa assíncrona começou...
4. Enquanto a tarefa assíncrona não é terminada, podemos fazer outras tarefas simultaneamente...
Tarefa 1 realizada
Tarefa 2 realizada
Tarefa 3 realizada
3. O tamanho do conteúdo da página https://docs.microsoft.com/en-us/dotnet/csharp/csharp é de 40172 caracteres
```

```go
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
)

func doAsyncTask(url string, ch chan<- string) {
    resp, _ := http.Get(url)
    body, _ := ioutil.ReadAll(resp.Body)
    ch <- fmt.Sprintf("3. O tamanho do conteúdo da página %s é de %d caracteres", url, len(body))
}

func doSyncTask() {
    fmt.Println("1. Fazendo alguma tarefa síncrona...")
}

func doSyncTaskWhileAwait() {
    fmt.Println("4. Enquanto a tarefa assíncrona não é terminada, podemos fazer outras tarefas simultaneamente......")
    for i := 1; i <= 3; i++ {
        fmt.Println("Tarefa ", i, " realizada")
    }
}

func main() {
    doSyncTask()
    ch := make(chan string)
    fmt.Println("2. Tarefa assíncrona começou...")
    go doAsyncTask("https://docs.microsoft.com/en-us/dotnet/csharp/csharp", ch)
    doSyncTaskWhileAwait()
    fmt.Println(<-ch)
}
```

```
PS C:\Users\Jaime\Desktop\Github\UERJ\EDL\artigo_csharp\golang> ./golang-async
1. Fazendo alguma tarefa síncrona...
2. Tarefa assíncrona começou...
4. Enquanto a tarefa assíncrona não é terminada, podemos fazer outras tarefas simultaneamente......
Tarefa  1  realizada
Tarefa  2  realizada
Tarefa  3  realizada
3. O tamanho do conteúdo da página https://docs.microsoft.com/en-us/dotnet/csharp/csharp é de 40172 caracteres
```



```
Memoria alocada apos a criacao dos elementos: 255988

Elementos da lista:

Elem1
Elem2
Elem3
Elem4
Elem5

Memoria alocada depois do GC: 128824
```

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace Rextester
{
    public class Program
    {
        public static void Main(string[] args)
        {

            LinkedList<String> Lista = new LinkedList<String>();
            Lista.AddLast("Elem1");
            Lista.AddLast("Elem2");
            Lista.AddLast("Elem3");
            Lista.AddLast("Elem4");
            Lista.AddLast("Elem5");

            Console.WriteLine("Memoria alocada apos a criacao dos elementos: {0}",
            GC.GetTotalMemory(false));
            Console.WriteLine();


            Console.WriteLine("Elementos da lista:");
            Console.WriteLine();
            foreach(string str in Lista) {
              Console.WriteLine(str);
            }
            Console.WriteLine();


            GC.Collect(0);
            Console.WriteLine("Memoria alocada depois do GC: {0}", GC.GetTotalMemory(false))
            ;
        }
    }
}
```

```
Elementos da lista:

Elem5
Elem4
Elem3
Elem2
Elem1
```

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef struct elem {
   string data;
   struct elem *next;
}elem;
struct elem* head = NULL;
void insert(string new_data) {
   elem *new_node = (struct elem*) malloc(sizeof(struct elem));
   new_node->data = new_data;
   new_node->next = head;
   head = new_node;
}
void print() {
   elem *ptr;
   ptr = head;
   while (ptr != NULL) {
      cout<< ptr->data << endl;
      ptr = ptr->next;
   }
}
void deletar() {
   elem *ptr , *deleta;
   ptr = head;
   deleta = ptr;
   while (ptr != NULL) {
      ptr=ptr->next;
      free (deleta);
      deleta = ptr;
   }
}
int main() {
   insert("Elem1");
   insert("Elem2");
   insert("Elem3");
   insert("Elem4");
   insert("Elem5");
   cout<<"Elementos da lista:" << endl << endl;
   print();
   deletar();
   return 0;
}
```

Referências

1. https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb
2. https://www.youtube.com/watch?v=NXVQasys0B8
3. https://www.quora.com/Where-is-C-used
4. https://www.codingame.com/playgrounds/4240/your-ultimate-async-await-tutorial-in-c/introduction
5. https://docs.microsoft.com/pt-br/dotnet/standard/garbage-collection/fundamentals
6. https://www.codingame.com/playgrounds/6179/garbage-collection-and-c
7. https://docs.microsoft.com/pt-br/dotnet/api/system.gc?view=netframework-4.8
8. https://www.devmedia.com.br/garbage-collection-entendendo-e-otimizando-parte-1/24082
9. https://www.geeksforgeeks.org/linked-list-implementation-in-c-sharp/
10. https://www.codementor.io/codementorteam/a-comprehensive-guide-to-implementation-of-singly-linked-list-using-c_plus_plus-ondlm5azr
11. https://blog.narenarya.in/concurrent-http-in-go.html