NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH C#

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

THREAD

Thread

- Là một luồng trong chương trình.
- Mặc định, chương trình chỉ sử dụng một luồng duy nhất.
- Việc sử dụng nhiều luồng được gọi là đa luồng.
- Trong C# hỗ trợ nhiều phương pháp để sử dụng luồng: ***Threading***, ***BackgroundWorker*** và ***Async-Await***

Threading

- *Ví dụ 1: Sử dụng Threading để tạo luồng*

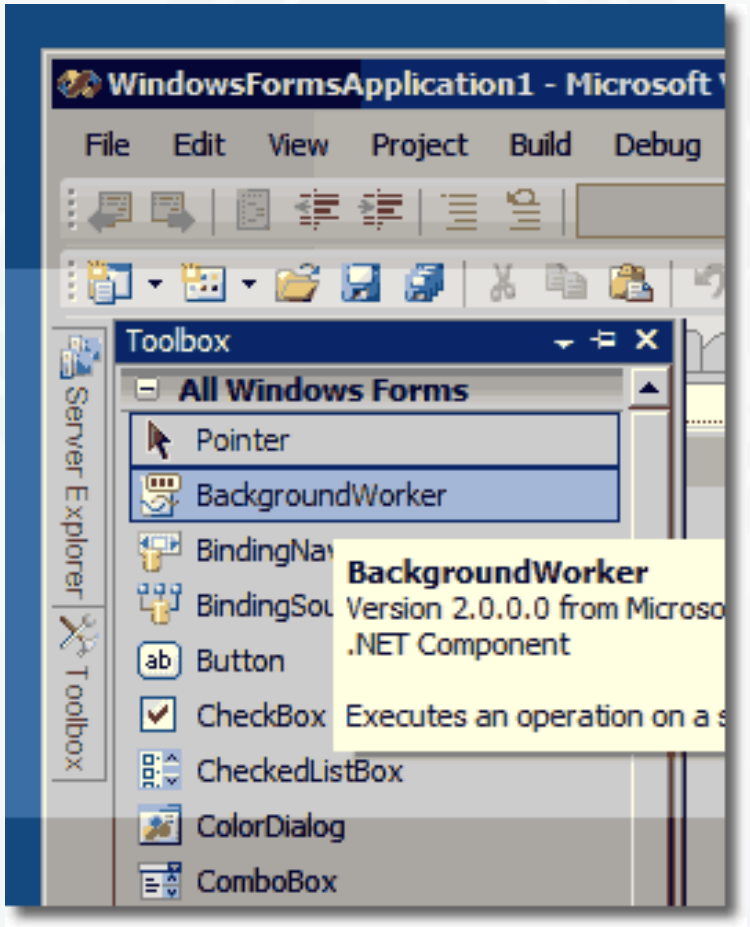
```
class Program
{
    static void Main()
    {
        Thread thread1 = new Thread(new ThreadStart(A));
        Thread thread2 = new Thread(new ThreadStart(B));
        thread1.Start();
        thread2.Start();
        thread1.Join();
        thread2.Join();
    }

    static void A()
    {
        Thread.Sleep(100);
        Console.WriteLine('A');
    }

    static void B()
    {
        Thread.Sleep(1000);
        Console.WriteLine('B');
    }
}
```

BackgroundWorker

- *<u>Ví dụ 2:</u> Sử dụng BackgroundWorker*



Async-Await

- *<u>Ví dụ 3:</u> Sử dụng Async-Await*

```
static void Main()
{
    // Tạo task và khởi động nó.
    // ... Chờ nó hoàn tất.
    Task task = new Task(ProcessDataAsync);
    task.Start();
    task.Wait();
    Console.ReadLine();
}

static async void ProcessDataAsync()
{
    Task<int> task = HandleFileAsync("C:\\enable1.txt");

    Console.WriteLine("Please wait patiently " +
        "while I do something important.");

    // Chờ cho phương thức xử lý hoàn thành.
    // ... Hiển thị kết quả.
    int x = await task;
    Console.WriteLine("Count: " + x);
}
```

Async-Await

```
static async Task<int> HandleFileAsync(string file)
{
    Console.WriteLine("HandleFile enter");
    int count = 0;

    // Thực thi việc đọc file
    using (StreamReader reader = new StreamReader(file))
    {
        string v = await reader.ReadToEndAsync();

        count += v.Length;

        for (int i = 0; i < 10000; i++)
        {
            int x = v.GetHashCode();
            if (x == 0)
            {
                count--;
            }
        }
    }
    Console.WriteLine("HandleFile exit");
    return count;
}
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

GENERIC TYPE

GENERIC TYPE

- **Generic Type** là một tính năng đặc biệt của C#.
- Được sử dụng trong khai báo và có thể là bất kỳ kiểu dữ liệu gì mà bạn cần khi sử dụng bạn sẽ phải thay thế.
- Ta dùng ký tự T để khai báo cho kiểu **generic**

GENERIC CLASS

- *Ví dụ:*

Khai báo lớp ***generic***

```
class Test<T>
{
    T _value;

    public Test(T t)
    {
        this._value = t;
    }

    public void Write()
    {
        Console.WriteLine(this._value);
    }
}

class Program
{
    static void Main()
    {
        Test<int> test1 = new Test<int>(5);
        test1.Write();

        Test<string> test2 = new Test<string>("cat");
        test2.Write();
    }
}
```

GENERIC TYPE

- *<u>Ví dụ:</u> Khai báo phương thức dùng kiểu **generic***

```
static List<T> GetInitializedList<T>(T value, int count)
{
    List<T> list = new List<T>();
    for (int i = 0; i < count; i++)
    {
        list.Add(value);
    }
    return list;
}

static void Main()
{
    List<bool> list1 = GetInitializedList(true, 5);
    List<string> list2 = GetInitializedList<string>("Perls", 3);
    foreach (bool value in list1)
    {
        Console.WriteLine(value);
    }
    foreach (string value in list2)
    {
        Console.WriteLine(value);
    }
}
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

OBJECT

OBJECT

- **Object** là lớp cơ sở của mọi lớp.
- Ta có thể sử dụng kiểu object để lưu trữ tổng quát.

Ví dụ:

```
object o = new { field1 = "Abc", field2 = true, field3 = 123 };

var o = new { f1 = "Test", f2 = false, f3 = 20 };
```

OBJECT

```
public static T Prop<T>(this object src, string propName)
{
    try
    {
        var value = src.GetType().GetProperty(propName).GetValue(src, null);
        return (T)Convert.ChangeType(value, typeof(T));
    }
    catch (NullReferenceException)
    {
        return default(T);
    }
}

public static bool HasProp(this object src, string propName)
{
    return src.GetType().GetProperty(propName) != null;
}
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

DICTIONARY

DICTIONARY

- **Dictionary** là một kiểu rất tiện lợi trong C#
- Mỗi biến kiểu **Dictionary** bao gồm 2 trường: **key** và **value**

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
    Dictionary<string, int> dictionary =
        new Dictionary<string, int>();
    dictionary.Add("cat", 2);
    dictionary.Add("dog", 1);
    dictionary.Add("llama", 0);
    dictionary.Add("iguana", -1);
    }
}
```

DICTIONARY

```
class Program
{
    static void Main()
    {
        Dictionary<string, int> dictionary = new Dictionary<string, int>();
        dictionary.Add("apple", 1);
        dictionary.Add("windows", 5);

        if (dictionary.ContainsKey("apple"))
        {
            int value = dictionary["apple"];
            Console.WriteLine(value);
        }

        if (!dictionary.ContainsKey("acorn"))
        {
            Console.WriteLine(false);
        }
    }
}
```

DICTIONARY

```
class Program
{
    static void Main()
    {
        // Khởi tạo nhanh một Dictionary
        Dictionary<string, int> d = new Dictionary<string, int>()
        {
            {"cat", 2},
            {"dog", 1},
            {"llama", 0},
            {"iguana", -1}
        };
        // Lặp trên từng cặp key và value
        foreach (KeyValuePair<string, int> pair in d)
        {
            Console.WriteLine("{0}, {1}",
            pair.Key,
            pair.Value);
        }
        // Sử dụng từ khóa var cho từng cặp giá trị
        foreach (var pair in d)
        {
            Console.WriteLine("{0}, {1}",
            pair.Key,
            pair.Value);
        }
    }
}
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

LINQ

LINQ

- **LINQ** (*Language Integrated Query*) là một dạng truy vấn trực tiếp trên các kiểu dữ liệu.
- Thường sử dụng cho **List** và **ArrayList**.

<u>Ví dụ:</u> Tìm giá trị trung bình trong một mảng số nguyên.

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
    int[] array = { 1, 3, 5, 7 };
    Console.WriteLine(array.Average());
    }
}
```

LINQ

- **LINQ** (*Language Integrated Query*) là một dang truy vấn trực tiếp trên các kiểu dữ liệu.
- Thường sử dụng cho **List** và **ArrayList**.

<u>Ví dụ:</u> Tìm giá trị trung bình trong một mảng số nguyên.

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
    int[] array = { 1, 3, 5, 7 };
    Console.WriteLine(array.Average());
    }
}
```

LINQ

- **Convert:**
  - *ToArray*
  - *ToDictionary*
  - *ToList*
  - *ToLookup*

LINQ

- **Mutate:**
  - *AsEnumerable*
  - *AsParallel*
  - *Cast*
  - *Concat*
  - *Contains*
  - *DefaultIfEmpty*
  - *Distinct*
  - *ElementAt*
  - *ElementAtOrDefault*
  - *Except*
  - *First*
  - *FirstOrDefault*
  - *GroupBy*
  - *GroupJoin*
  - *Intersect*
  - *Join*
  - *Last*
  - *LastOrDefault*
  - *OfType*
  - *OrderBy*
  - *OrderByDescending*
  - *Reverse*
  - *Select*
  - *SelectMany*
  - *Single*
  - *SingleOrDefault*
  - *Union*
  - *Where*
  - *Zip*

LINQ

- **Skip and take:**
  - *Skip, SkipWhile*
  - *Take, TakeWhile*
- **Computation**
  - *Aggregate*
  - *All*
  - *Any*
  - *Average*
  - *Count*
  - *SequenceEqual*
  - *Sum*

LINQ

- **Others:**
  - *Min, max*
  - *Empty*
  - *Range*
  - *Repeat*

LINQ

- **Truy vấn:**

```
class Program
{
    static void Main()
    {
    int[] array = { 1, 2, 3, 6, 7, 8 };
    // Biểu thức truy vấn.
    var elements = from element in array
                   orderby element descending
                   where element > 2
                   select element;
    // Liệt kê.
    foreach (var element in elements)
    {
        Console.Write(element);
        Console.Write(' ');
    }
    Console.WriteLine();
    }
}
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

LAMBDA EXPRESSION

LAMBDA EXPRESSION

- Là một tính năng mới trong C#, giúp chúng ta truy vấn nhanh như **LINQ**.
- Dùng toán tử lambda =>
- Biểu thức bên trái => là các tham số
- Biểu thức bên phải => là kết quả

LAMBDA EXPRESSION

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
    List<int> elements = new List<int>() { 10, 20, 31, 40 };
    // ... Tìm số lẻ đầu tiên trong List.
    int oddIndex = elements.FindIndex(x => x % 2 != 0);
    Console.WriteLine(oddIndex);
    }
}

Kết quả
2

Chi tiết Lambda
x           x là tên tham số.
=>          Dấu phân cách biểu thức lambda.
x % 2 !=0   trả lại true nếu x không phải là số chẵn.
```

NHỮNG VẤN ĐỀ TRONG LẬP TRÌNH WINFORMS

REGULAR EXPRESSION

REGULAR EXPRESSION

- Là biểu thức chính quy
- Thường để để so khớp một định dạng nào đó thông qua biểu thức cho sẵn.
- Dùng lớp **Regex** để so khớp với mẫu (dùng phương thức **Match**)

REGULAR EXPRESSION

```
public static int chkPassStr(string password)
{
        int score = 1;
        if (password.Length < 1)
            return 0;
        if (password.Length < 4)
            return 1;
        if (password.Length >= 8) score++;
        if (password.Length >= 12) score++;
        if (System.Text.RegularExpressions.Regex.IsMatch(password, @"[0-9]+(\.[0-9][0-9]?)?"))
            //number only //"^\d+$" if you need to match more than one digit.
            score++;
        if (System.Text.RegularExpressions.Regex.IsMatch(password, @"^(?=.*[a-z])(?=.*[A-Z]).+$"))
            //both, lower and upper case
            score++;
        if (System.Text.RegularExpressions.Regex.IsMatch(password, @"[!,@,#,$,%,^,&,*,?,_,~,-,£,(,)]"))
            //^[A-Z]+$
            score++;
        return score;
}
```

CÂU HỎI?