C# Interview Questions Summery By Mahmoud Badawy

-----------------------------------------------------

MB

##################################################################

اول حاجه كده ملخص هحط فيه كل الاساله و الاجوبه الخاصه بال C# و Dot net للانترفيوهات

و نبتدى معاكم اول سؤال

MB

##################################################################

In Debugging What is Different Between (Autos , Locals , Watches)

-----------------------------------------------------------------------------------

A screenshot of a computer

Description automatically generated  
Locals : دى و انت بتعمل ديباج بتجيبلك كل قيم الفاريبولزو البراميتارز حتى الى فى فانكشن بتناديها و الى عديت عليها و الى انت فى نفس الاسكوب بتعها و بتعرفك انت فى انهى اسكوب دلوقتى و اى تغيير فى اى فاريبول من الفاريبولز دى بتعرفها لان لون الفاريبول و قيمته بيتغيروا للون الاحمر و ده معناه ان حصل تغيير بعد الخطوه الى انت مشيتها دى فبتعرف التغييره حصلت امتى من متابعه ال locals دى لانها بتتعامل مع الاسكوب و بالتالى اى مشكله خاصه بقيمه ما تم تغييرها و انت مش فاهم اتغيرت فين و امتى بتعرفه عن طريق ال locals

Autos : نفس الى اللوكال بتعمله بس الفرق انها بتراقب الفاريبولز فقط و ملهاش دعوه باى براميتر مبتشوفهمش هى بتركز على الفاريبولز فقط و التغيرات الى بتطرأ عليها كل شويه سواء فى نفس الاسكوب او فى اسكوبس مختلفه

Watches : دى بقه يا سيدى بتشتغل بمزاجك انت يعنى تحطلها فاريبول معينه و تتبعها هى على وجه الخصوص علشان متوجعش دماغك بالباقى و عادتا احنا بنستخدم دى بدلا من اللوكالز علشان نعرف نركز على الفاريبول الى بيحصل فيها المشكله بسرعه

MB

##################################################################

Ternary Operator Vs If-Else

------------------------------------------

الفرق الواضح جدا ان if – else مش لازم يعمل ريتيرن لقيمه - ينفع نحط جواها Retern للقيم او ممكن نعمل كود مبيعملش Return لاى قيمه و يبقى مجرد قراءه للداتا او تغيير فى قيمه فاريبول بس مش هعمله ريتيرن عادى

اما ال Ternary Operator : ده لازم يعمل return لقيمه فلو معملتش ريترين لقيمه منه هيديك ايرور

A screen shot of a computer

Description automatically generated

MB

##################################################################

Do While Vs While

------------------------------------------

While : مش هيشتغل لو الى جواه مبيحققش الشرط

Do While : اول مره هيشغل الكود الى جوه ال do حتى لو كان مش محقق الشرط الى جوه while بعد كده يبتدى يطبق الشروط لما يخش بقه فى ال while

var x = 11;

while (x < 11 && x >=0)

{

Console.WriteLine("Hello from While");

--x;

}

var x2 = 11;

do

{

Console.WriteLine("Hello from Do While ");

--x2;

}

while (x2 < 11 && x2 >= 0);

MB

########################################################################

Value Type Vs Refrence Type  
---------------------------------

عندنا بقه انواع من ال Data type من نوع ال Value Type و نوع تانى من نوع Refrence Type

طيب ايه الفرق بينهم ؟؟

ال Value Type ذى مثلا ال ال

Bool , byte , char, decimal, double , enum , float , int , long , sbyte , short , struct , uint , ulong , ushort

و دى يا سيدى الميكانيزم بتعها ان اسمها و قيمتها بيتسجلوا فى ال Stack

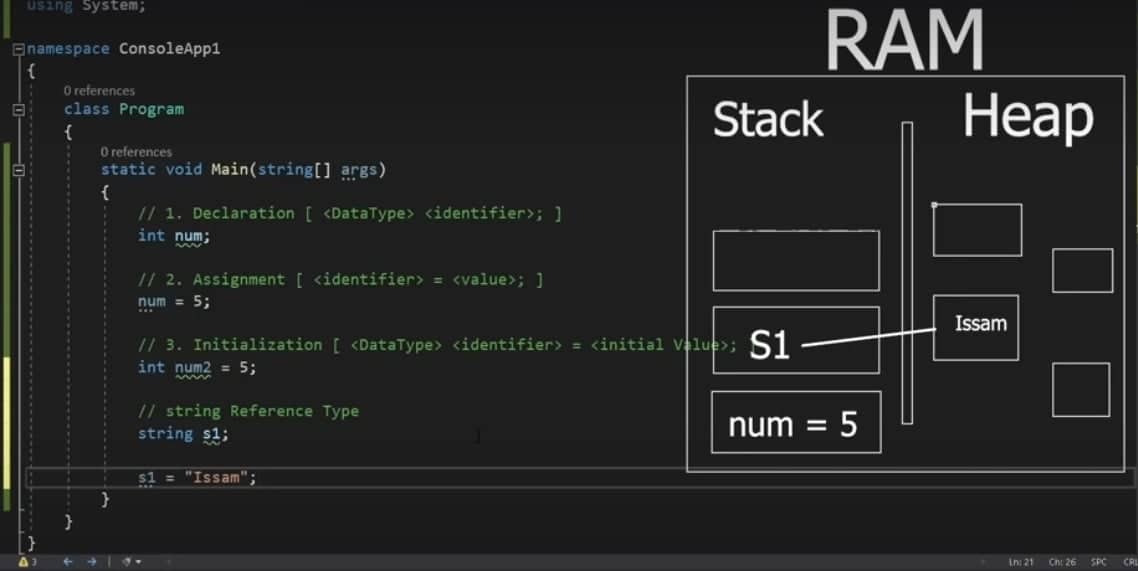
يعنى لما باجى اعرفها ذى ما فى الابلكيشن ده كده

Int16 Num = 12345; // Decleration and assignation Value is in Stack

بينما ال Refrence Type بيكون التعريف او الدكليريشن بتعها فى ال Stack و بيشاور على القيمه الى موجوده فى ال Heap و من اهم الانواع الى من نوع Refresnce Type هى string , Array , List , Objects

string Num = "Mebo"; // Decleration in stack and assignation Value is in Heap

و الصوره دى هتوضح اكتر للناس علشان تفعم طريقه التخزين



ذى ما احنا شايفين الانتيجر اتسجل الدكليريشن بتاعه الى هو ال num فى Stack و كذلك القيمه الخاصه بيه الى هى 5 فى ال Stack  
اما الاسترينج بقه راح اتسجل الدكليريشن بتاعه الى هو S1 فى Stack و راح رايح مخزن فيمته فى مكان تانى اوسع فى المساحه و اسرع اسمه ال Heap

الفرق التانى بقه يا معلم و المهم   
اولا ال Value Type انت لما بتديله قيمه بياخدها بشكل مباشر القيمه و ليس عنوان القيمه وبالتالى لو عمل للقيمه دى اسين على فاريبول جديد فهو بياخد القيمه من الفاريبول الاول برضوا بشكل مباشر و ليس عنوان القيمه و بالتالى لو فرضنا ان الفاريبول الاصلى غير القيمه بعد كده فالفاريبول الى اخد القيمه القديمه منه مش هيتاثر خالص و العكس صحيح

بالتالى طريقه التخزين اشبه ب ان   
واحد راح يتصور فى الاستوديو 2 صور و لما اخد ال 2 صور الورق دول راح مدى صوره لصحبه يعينهاله معاه و صوره لنفسه فبقى كل واحد منهم معاه الصوره دى الى هى ستايل كاجول هنقول قيمتها 5 مثلا فكده القيمه 5 دى مع الراجل و مع صاحبه برضوا

جه بعد ساعه لما صاحبه مشى و اخد الصوره بتعته معاه قعد يفكر و راح الراجل ده قالك الصوره الى معايا دى مش عجبانى و قطعها و راح اتصور صوره تانيه بلبس تانى خالص قيمته و ليكن 10 طيب اصبح دلوقتى الراجل معاه صوره ستايل جديد ب 10 و صاجبه معاه صوره ستايل قديم ب 5 فكده تغيير الصوره الى مع الراجل الاصلى و قيمتها ماثر على الصوره او القيمه الى مع صاحبه و ده الى بيتسمى ال Value type و سميت بالاسم ده نظرا لانها تعتمد على القيمه الفعليه لل Variable   
مثال

void ValueType()

{

int num1 = 5;

int num2 = num1;

num1 = 10;

Console.WriteLine(num1); // 10

Console.WriteLine(num2); // 5

}

ثانيا ال Reference Type ده يا سيدى انت لما بتديله قيمه بياخدها بشكل عنوان و ليس القيمه وبالتالى الفاريبول الاصلى بيعمل للعنوان ده اسين على فاريبول جديد فهو كمان بياخد العنوان ده من الفاريبول الاول و لما بيحب اى حد منهم يعرف القيمه بيروح يجيبها عن طريق العنوان الى اتسجل فى ال Heap ده و بالتالى اى تغيير فى القيمه سواء عن طريق الفاريبول الاصلى او النسخه الى منه الاتنين بيشوفوا القيمه الجديده و ده لان الاتنين مخزنين عناوين و ليس قيم و العنوان متمش تغييره لا ده القيمه المقابله للعنوان هى الى اتغيرت و بما ان الاتنين بصيين على نفس العنوان فالاتنين هيشوفوا نفس القيمه الجديده و لذلك طريقه التخزين اشبه ب ان

واحد راح عمل جروب واتس و عمل جروب مع صاحبه و الاتنين ادمنز جه حط صوره للجروب الى هنعتبرها بقيمه 5 فهو و صاحبه دلوقتى شايفين الصوره الى قيمتها 5 على الجروب ده الى هو هنعتبره العنوان الى هخزن فيه الصوره بتاعت الجروب و الى هى هنعتبرها

جه بعد يومين صاحبه مش عاجبه الصوره الى الاولانى حطها فراح مغيرها لصوره جديده قيمتها 10 فدلوقتى الاتنين صحاب شايفين الصوره الى قيمتها 10 و ادى مثال يا جماعه

void RefrenceType()

{

int[] num1 = { 5 };

int[] num2 = num1;

num1[0] = 10;

Console.WriteLine(num1[0]); // 10

Console.WriteLine(num2[0]); // 10

}

ال String بنعتبرها RefrenceValue علشان هى عباره عن Array of Char بس مع ذلك هى عمتا immutable Value يعنى غير قابله للتغيير و ده لان اى تغيير فى الفاريبول بتعتها هى مش بتغير فى الفاريبول لأ هى بتشيل الفاريبول خالص و تعمل فاريبول جديد بنفس الاسم و بترمى القديم فى الميمورى و بالتالى ده فى البرامج الكبيره لو ممسحناش الميمور اول باول بيحصل memory Leck و الميمور بتتملى و بيبقى حالها ما يعلم بيه الا ربنا

MB

########################################################################

Var Vs dynamic   
---------------------------------

Var احنا بنحطها و نعرف المتغير و بعد كده نحط اى قيمه للمتغير و هى بتفهم هو من نوع ايه و على اساسه بتتعامل على انها من النوع ده فلو ادناها انتيجر هتبقى انتيجر اديناها دابل هتبقى دابل ادينها استرينج هتبقى استرينج

و مينفعش انى اجى بعد ما ادتها قيمه من نوع معين و اروح مغيره يعنى مينفعش اديلها مثلا قيمه استرينج و اجى بعديها بسطر اديها انتيجر لا ساعتها هتعمل ايرور

اما ال Dynamic لا مختلف عن ال Var فى انه ينفع تديله فى السطر الاول انتيجر و فى السطر التانى استرينج و فى التالت بتنجان مقلى عادى خالص

و دى مثال للكلام ده

using System;

class Program

{

static void Main(string[] args)

{

var IntNum = 5;

Console.WriteLine(IntNum); // 5

IntNum = "5"; // syntax Error

}

}

هيعمل مشكله اول ما اكتب جوه الفاريبول استرينج بعد ما اديته اول قيمه انتيجر

بينما فى ال dynamic هيشتغل عادى خالص

using System;

class Program

{

static void Main(string[] args)

{

dynamic IntNum = 5;

Console.WriteLine(IntNum); // 5

IntNum = "5";

Console.WriteLine(IntNum); // 5

}

}

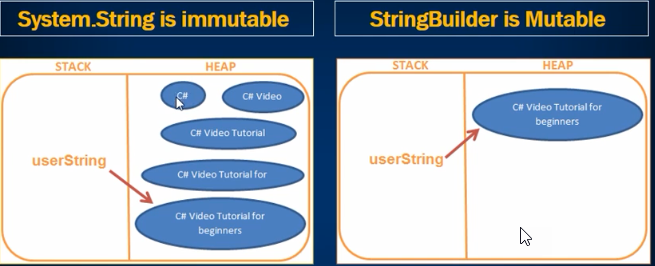
MB

##################################################################

String VS StringBuilder

---------------------

String : بص يا سيدى الاسترينج العادى لو جيت تعمل كونكاتيونيشن او تعديل فهو بيفضل يخزن فى اماكن كتير جدا فى الميمورى يعنى لو هتعمل تعمل كونكاتيونيشن او تعديل بين مليون كلمه سترينج فكده كل تعديله او كونكتينسشن بيحجز مكان فى الميمورى و بالتالى هو حجز مليون مكان فى الميمورى ففرتك الميمور

StringBuilder : ده بقه لو عملت 100 مليون كونكاتيونيشن هو بيجمعهم كلمهم فى مكان واحد و يخزنهم فيه و بعد كده يشاور عليه يخده و بالتالى هو احسن مليون مره فى البيرفورمانس 

اذاى اكتب الاسترينج و الاسترينج بيلدر بقه فى الكود ؟؟

using System;

using System.Text;

namespace CsharpLearning

{

class Program

{

public static void Main(string[] args)

{

string FullName = "My ";

FullName += "Name is Mostafa ";

FullName += "Badawy";

Console.WriteLine(FullName);

StringBuilder MyStr = new StringBuilder("My ");

MyStr.Append("Name ").Append("is ").Append("Mahmoud ");

MyStr.Append("Badawy");

Console.WriteLine(MyStr);

}

}

}

MB

##################################################################

Implicit Convert VS Explicit Convert

-----------------------------------------------

Implicit Convert : بص يا سيدى ده التحويل الى الكومباير بيعمله Automaticly بدون ما يحتاج اى كود زياده منك و ده لان اوقات بينفع انى اخزن مثلا ال int جوه ال Double لان الانتيجر اقل فى المساحه من الدوبل فبينفع اخزنه فيه بس هو الكومبايلر بيخزن ال int ده جوه ال double بعد ما بيحوله فى الخفاء كده الى double و لكن لو جينا نقلب الايه و نحط ال Double جوه ال int فالكومبيلر هيرفض و ده لانه مش قادر يحول ال double الى int علشان يخزنه جوه ال int و ده راجع لان ال double مساحته اكبر فممكن اكون مخزن فيه رقم كبير و علشان احوله ل int فوارد يحصل Date lose و بالتالى يحصل مشكله كبيره و علشان كده الكومبايلر بيطلع ايرور فى الحاله دى و يقولك لأ مش نافع

Explicit convert : ده بقه يا سيدى تحويل Manual و ده بقه يا سيدى كانك كده بتقوله انا مسؤول على الى انا بعمله فحاولى ال Double الى int عادى و مش مشكله لو حصل Data lose بس علشان النوعيه دى تشتغل اكيد محتاج تكتب كود يفعل التحويل ده علشان تبقى انت كتبتها بأيدك و مسؤول عن اى داتا هيتم فقدها يا صديقى

static void Main(string[] args)

{

int IntNum = 10;

double Num = IntNum; // implicitly Convert

Console.WriteLine(Num);

IntNum = Num; // Syntax Error (Can not implicitly Convert From double to int)

IntNum = (int)Num; // Explicit Convert

Console.WriteLine(IntNum);

}

MB

##################################################################

Compilation Process

------------------------------------

A screen shot of a computer

Description automatically generated

أول حاجه احنا بنكتب الكود فده الى اسمه ال Source Code ده الكود بتعنا الى بنفهمه ده اول ما بنعمل Build الى بيحصل ان ال Compiler بيبدأ يشتغل علشان يحول ال Source Code ل Machine Code   
  
طيب خطوات حدوث الكلام ده ايه بقه ؟؟

اول حاجه عندى Souce Code فعمل بيلد للبروجيكت او Run الى هيحصل ان الكومبايلر هيششتغل و ياخد الكود يحوله الى لغه وسيطه IL او MSIL او CIL و اللغه الوسيطه دى هى كوبرى بين ال Machine language , C# language و لذلك بعد تحويل السورس كود الى لغه وسيطه عن طريق الكومبايلر بيتم تحويل اللغه الوسيطه الى machine Code عن طريق ال CLR & JIT

طيب ايه هو ال Source code ؟

ده الكود الى احنا بنكتبه و بنفهمه انا و انت و اى ديفلولوبر

طب ايه هو ال IL ؟

ده ال ddls & exe files الى هو مكتوب عن طريق ال intermediate language لسه متحولش ل machine language

طب ايه هو ال CLR ؟

ده بقه يا باشا لما بتدوس على Exe file علشان يشغل البرنامج : الى بيحصل ساعتها ان ال exe file بيتعمله Compile عن طريق ال Just in time Compiler ليتم تحويله ال Machine Code و الى بيتم التعامل معاه عن طريق ال CLR الى هى هنا بنعتبرها ال Virtual Machine بديله قبل مرحله البروسيسور

يعنى مجرد ما ندوس على ال exe file بيشغله

ال CLR , JIT = Common language Runtime – just in time Compilation و الفايل الى مكتوب ب IL ده بيروح متحول فى الران تايم الى Machine Code البروسيسور يقدر يفهمه بقه بدون التركيز فى موضوع المعماريه يعنى يشتغل على اى بروسيسور عادى لان اعتماده على virtual CPU

MB

##################################################################

Manage code VS Un manage Code

------------------------------------------------

ال C# هى Manage Code و ده لان هى مش بتطلع ال machine Code بشكل مباشر و ده لانها بتستهدف Virtual Machine كبديل لل Processor علشان لو المعماريه بتاعت البروسيسور مختلفه عن المكتوب فى الكود ميحصلش مشكله لان البروسيسور و البرنامج معماريتها مختلفه فاحينانا هتلاقى حاجه بتحتاج برامج 64 و تانيه عاوزه 32 و لذلك ال manage Code بيحل المشكله دى بانه يعمل intermediate language وسيطه تخلى الماشينج كود تنفع لاى حاجه

على عكس ال unmanage Code : دى و انا بعمل البرنامج لازم ابقى محدد البروسيسور الى هتتعامل معاه بالظبط و اكتبله هو الكود مخصوص لان لواتكتب و اتبنى بمعماريه مختلفه عن البروسيسور مش هيشتغل على البروسيسور ده ابدا و ده لان الكود بيتعامل بيتحول بشكل مباشر الى machine Code فمفيش وسيط يظبط الماشين كود قبل ما تتعمال مع البروسيسور

MB

##################################################################

Class VS Object

----------------------------------

Class : هو الكتالوج الى بحط فيه الخصائص الخاصه بالشئ الى ببنيه مثلا انا عندى كلاس اسمه Person ليه خصائص properties ذى الاسم و العمر و الوظيفه ال behavior ( Methods) و شويه وظائف الى هى يقدر يعملها

Object : دى النسخه الى بنقدر ناخدها من الكلاس لما نعوز نستعملها فى انشاء شخص جديد

MB

##################################################################

ايه هى فائده ال Constructor Method

------------------------------------------------

تستخدم فى انشاء الاوبجيكت و نقدر نعمل فاليديشن فيها اثناء انشاء الكونستراكتور ذى ان مثلا احبره انه لازم يبعت اسم فى الكونستراكتور علشان الاوبجيكت يتعمله كريشن و ده معناه انى اقدر اخلى جواه كل ال Required Prop بحيث مينفعش انشأ اى اوبجيكت من الكلاس الا بعد ما يبعتوهالى فى الكنستراكتور و كمان اقدر ابعت فيه ال Default Values لكل ال prop او انى اشغل فيه فانكشنز معينه محتاجها تتعمل اول ما هنبتدى نكريت الاوبجيكت .

MB

##################################################################

Auto Implemented Property VS custom Property

-----------------------------------------------------------------

Auto Implemented Property : ده بروبرتى مجرد انا عاوز اكتب عليه فاليو و اقرا الفاليو دى و خلاص

// Auto Implemented Prosperity

public string firstName { get; set; }

custom Property : دى يا صديقى بروبرتى بس انت عاوز تعمل جواه فاليديشن و انت بتكتب عليه او و انت بتقرأ منه او تزود عليه شغل اثناء القراءه و الكتابه

// Custom Implemented Prosperity

private string \_lastName; // Backing Field

public string lastName

{

get { return \_lastName == null ? "Badawy" : \_lastName; }

set { \_lastName = value + " The King"; }

}

MB

##################################################################

**Read-Only Property** Vs **Init Property** Vs **Constant** Vs **Static Read-Only Property**

---------------------------------------------------------------------------------------------------------------------

**Constant Field** : بص يا سيدى دى بنحطلها القيمه اثناء و انا بعملها **declaration** فقط و مينفعش نعدل عليها فى اى مكان تانى فى الكلاس خالص خالص خالص not in Constructor & not In Object Initializer & not After Creating Object  
لذلك ال **Constant** بتاخد الفاليو اثناء ال Compilation Time

static void Main(string[] args)

{

Person person = new Person { Name = "Yassin"}; // Syntax Error in Object Initializer

person.Name = "Yassin" ; // Syntax Error after Creation

}

class Person

{

public **const** string Name = "Mahmoud Badawy"; // Ok in Declaration

public Person()

{

Name = "Mostafa"; // Syntax Error in Construcor

}

}

--------------------------------------------------------------------------

**Read-Only Property** : بص يا سيدى دى بنحطلها القيمه اثناء و انا بعملها **declaration** و ممكن نعدل عليها فى **constructor** و لكن لا يمكن ابدا التعديل عليها اثناء ال الاوبجكيت انشياليزر & not After Creating Object not In Object Initializer

لذلك ال **Read-Only Property** بتاخد الفاليو اثناء ال Run Time

static void Main(string[] args)

{

// Syntax Error in Object Initializer

Person person = new Person { Name = "Yassin" , FName="Lolo" };

person.Name = "Yassin" ; // Syntax Error after Creation

}

class Person

{

public string Name **{ get; }** = "Mahmoud Badawy"; // Ok in Declaration

public readonly string FName = "Mostafa Badawy"; // Ok in Declaration

public Person()

{

Name = "Mostafa"; // Ok in Constructor

FName = "Mostafa"; // Ok in Constructor

}

}

--------------------------------------------------------------------------

**Init Property :**  بص يا سيدى دى بنحطلها القيمه اثناء و انا بعملها **declaration** و ممكن نعدل عليها فى **Contructor** وكمان اثناء **Object Initializer** لكن لا يمكن ابدا التعديل عليها اثناء ال الاوبجكيت انشياليزر not After Creating

لذلك ال **Init Property** بتاخد الفاليو اثناء ال Run Time

static void Main(string[] args)

{

Person person = new Person { Name = "Yassin" }; // Ok in Object Initializer

person.Name = "Yassin"; // Syntax Error after Creation

}

class Person

{

public string Name { get; init; } = "Mahmoud Badawy"; // Ok in Declaration

public Person()

{

Name = "Mostafa"; // Ok in Constructor

}

}

--------------------------------------------------------------------------

**Static Read-Only Property:**  بص يا سيدى دى بنحطلها القيمه اثناء و انا بعملها **declaration** و ممكن نعدل عليها فى **Static Contructor** وليس ال Constructor not العادى و ينفع نعدل عليها كمان اثناء **Object Initializer** لكن لا يمكن ابدا التعديل عليها اثناء --- not After Creating و ال constructor not العادى

لذلك ال **Static Read-Only Property** بتاخد الفاليو اثناء ال Run Time & Compilation Time

static void Main(string[] args)

{

Person person = new Person { Name = "Yassin" , FName="Lolo" }; // Syntax Error in Initialized Object

Person.FName = "Lolo"; // Syntax Error in After Creation

Console.WriteLine( Person.FName);

}

class Person

{

// Ok in Declaration Only

public static string Name { get; } = "Mahmoud Badawy";

public static readonly string FName = "Mostafa Badawy";

public Person() // Normal Constructor

{

Name = "Mostafa"; // Syntax Error in Constructor

FName = "Mostafa"; // Syntax Error in Constructor

}

static Person() // static Constructor

{

Name = "Mostafa"; // Ok in Constructor

FName = "Mostafa"; // Ok in Constructor

}

}

فروقات جوهريه بقه بين ال Const & ReadOnly و الى متكلمناش عنها فوق ان ال Const مبيخدش Static

انما ال Read-Only بياخدها عادى ذى ما انت شايف فوق اهه

ألكونستانت لازم تديله فيمه اثناء ال Declaration و مينفعش تسيبه فاضى انما Read-only لأ ممكن فى الدكليريشن تطنشه خالص و متعرفوش فى ال Declaration و تحطله القيمه بتعته فى ال Constructor و لذلك

ال Const بيتم استخدام قيمته فى البرنامج كله اثناء ال Compilation Time و ده لانه خلاص مش هيتغير فالكومبايلر بيروح واخد قيمته و مبدلها فى كل الاماكن الى هو تم استخدامه فيها بالقيمه دى على عكس ال Read Only هو ممكن ميعملوش Assignation و هو بيعمله Declaration و بالتالى مش هيعرف يخده و هو بيكومبايل هيستنى لما يخش فى ال run Time و يروح مستعمل قيمته الى ممكن يكون هو ابتدى يحطها فى ال Constructor و محطهاش وقت ال Declaration

Several blue and orange rectangular boxes with text

Description automatically generated

MB

##################################################################

Private Class

-----------------------------------------------------------------

بص يا حج ال private كاسيس موديفير مينفعش احطها قبل ال Class الا فى حاله واحده بس : اذا كان الكلاس داخل كلاس اخر اما لو مش داخل كلاس اخر و هو كلاس تحت namespace فمش مسموح باستخدام ال private معاه ال Class كده

و كذلك ذيه ال Struct & Enum & interface و اى حاجه شبهه كده

MB

########################################################################

Class Vs Static Class  
---------------------------------

Static Class : هو عباره كلاس يحتوى على One Instance فقط و يكون Shared فى السيستم كله بمعنى ان ال Static كلاس ده مش بينفع نعمل منه new instance كل شويه لأ هو بيتكريت مره واحده فقط و تفضل شغال بيه بقه طول ما انت شغال فى السيستم بقه

الحاجه التانيه انه الكلاس ال Static يحتوى فقط على Static Methods & Static Constructor & Static Properties يعنى اى حاجه جواه لازم تبقى واخده Static غير كده هتدى ايرور و ادى مثال

static void Main(string[] args)

{

Person.SayHelloWorld();

Person person = new Person(); // Syntax Error in Initialized Object

}

static class Person

{

public string Name { get; set; } = "Mahmoud Badawy"; // Syntax Error prop not Static

public static string FName = "Mostafa Badawy"; // Ok because Static prop

static Person() // Ok because static Constructor

{

Console.WriteLine("Hello From Static Constructor");

}

public Person() // Syntax Error because not static Constructor

{

Console.WriteLine("Hello From Normal Constructor");

}

public static void SayHelloWorld()

{

// Ok because static Method

}

public void SayHelloWorldNotStatic()

{

// Syntax Error because not static Method

}

}

الحاجه التانيه ان اى حاجه Static بيبقى عباره عن نسخه واحده فقط لا غير و لذلك التغيير فى اى حاجه فيها بيغير فى كله و كذلك الكونستراكتور بتعها هو بيتنده مره واحده فقط و دى مع اول كول لاى حاجه جوه ال static class

تعالى اوريك مثال

Person.SayHelloWorld(); // Ok Initialized Object call static constructor

Person.SayHelloWorld(); // Ok Initialized Object not call static constructor

Person.SayHelloWorld(); // Ok Initialized Object not call static constructor

static class Person

{

public static string FName = "Mostafa Badawy"; // Ok Static prop

static Person() // Ok static Constructor

{

Console.WriteLine("Hello From Static Constructor");

}

public static void SayHelloWorld()

{

// Ok static Method

}

}

Class : هو عباره كلاس عادى نفدر نعمل منه ملايين النسخ يعنى بينفع نعمل منه new instance كل شويه

الحاجه التانيه انه الكلاس العادى يحتوى على Normal Methods & Normal Constructor & Normal Properties & Static Methods & Static Constructor & Static Properties و ادى مثال

Person.SayHelloWorld(); // Ok Initialized Object call static constructor

Person.SayHelloWorld(); // Ok Initialized Object not call static constructor

Person.SayHelloWorld(); // Ok Initialized Object not call static constructor

Person person = new Person(); // Ok Initialized Object

person.SayHelloWorldNotStatic();

class Person

{

public string Name { get; set; } = "Mahmoud Badawy"; // Ok prop not Static

public static string FName = "Mostafa Badawy"; // Ok Static prop

static Person() // Ok static Constructor

{

Console.WriteLine("Hello From Static Constructor");

}

public Person() // Ok not static Constructor

{

Console.WriteLine("Hello From Normal Constructor");

}

public static void SayHelloWorld()

{

// Ok static Method

}

public void SayHelloWorldNotStatic()

{

// Ok not static Method

}

}

MB

########################################################################

Call by ref vs Call by value   
---------------------------------

Call by value : هتلاحظ ان اى تغيير في ال num لا يأثر على ال MyNum و ده لانهم أصلا بيشاوروا على مكان مختلف تماما في الميمورى

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

int MyNum = 10;

Myfunc(MyNum);

Console.WriteLine(MyNum); // 10

}

public static void Myfunc(int num)

{

num = 15;

Console.WriteLine(num); // 15

}

}

}

Call by ref : هتلاحظ بقه ان اى تغيير في ال num بيغير ال MyNum و العكس صحيح برضوا يعنى الى بيتغير بيغير التانى و ده لان الاتنين بيبصوا على عنوان القيمه الى هو الريفرنس و بالتالى اى تغيير فى القيمه الى متسجله على العنوان الى بيبصوا عليه هما الاتنين فهيشوفوها هما الاتنين بالقيمه الجديده و الكود عشان يفهم ان الكول باى ريفرنس بنكتبه بطريقتين

الطريقه الاولى بنستخدم ال Ref Key word

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

int MyNum = 10;

Myfunc(ref MyNum);

Console.WriteLine(MyNum); // 15

}

public static void Myfunc(ref int num)

{

num = 15;

Console.WriteLine(num); // 15

}

}

}

الطريقه التانيه بنستخدم ال Out Key word

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

int MyNum = 10;

Myfunc(out MyNum);

Console.WriteLine(MyNum); // 15

}

public static void Myfunc(out int num)

{

num = 15;

Console.WriteLine(num); // 15

}

}

}

هيعمل هنا اوت بوت 15 ثم 15 و ده لان هيعمل كول و يبعت براميتر ب 10 و اول ما يخش الفانكشن النمبر الى كان بعشره ده بقى بيشاور على 15 فهيطلع في كونسول الفانكشن 15 و بما ان ال MyNum بيشاور على نفس المكان فكل الى هيحصل بقه انه هيتغير هو كمان ل 15 ليصبح الاتنين بيشاوروا على نفس المكان و الاتنين قيمتهم ب 15

و ذي ما انت شايف الفرق بين العادى و الى بيشتغل بالريفرنس انى بحط كلمه or out ref في تعريف البراميتر جوه الفانكشن و في الكول برضوا

طيب احنا كده عرفنا الكول باى ريفرنس بيحصل عن طريق ال out و ref و الاتنين شبه بعض جدا طب ايه بقه الفرق بينهم يا رجاله ؟؟ هنقوله الفقره الى جايه

MB

########################################################################

Call by ref (Ref Vs Out Key)  
---------------------------------

ال Ref اثناء الدكليريشن لازم و لابد و اجبار نحطلها قيمه اما ال out لا مش ضرورى

مثال اهه

دى ال ref لازم تاخد فيمه عند التعريف

int MyNum = 0;

Myfunc(ref MyNum);

دى ال out مش لازم تاخد فيمه عند التعريف

int MyNum ;

Myfunc(out MyNum);

الفرق التانى ان فى الفانكشن الى بعمل فيها الكول و ببعت فيه ال Ref او out   
فى حاله ال ref انا مش لازم اغير فى البراميتر اصلا و ده معناه ان مش لازم استخدمها فى فانشكن الكول اصلا

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

int MyNum ;

Myfunc(out MyNum);

Console.WriteLine(MyNum); // 15

}

public static void Myfunc(out int num)

{

num = 15; // Required

Console.WriteLine(num); // 15

}

}

}

اما فى حاله ال out اجبار نستخدمها داخل فانكشن الكول

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

int MyNum = 10;

Myfunc(ref MyNum);

Console.WriteLine(MyNum); // 10

}

public static void Myfunc(ref int num)

{

// using num is not required in ref call function

Console.WriteLine(num); // 10

}

}

}

**ملحوظه هامه جدا بقه**

**ال Ref & Out دول من الاخر كده بنستخدمهم فى تحويل الفاريبول الى هى فى الاساس Value type ذى ال int , float , double , bool , decimal, char, long الخ الخ الى فاريبول من نوع Reference type ذى الكلاس و ما يشبهه**

MB

##################################################################

Simple Enum

------------------------------------------

Simple Enum : بص يا سيدى انت عندك حاجات ذي مثلا ال Gender و ليكن انا عملها لوك اب بحيث ان   
رقم 0 معناه Male  
رقم 1 معناه Female

رقم 2 معناه Unknow  
فانا دلوقتى المفروض هخزنه في الداتا بيز على انه انتيجر مع ان انا هحتاج اظهره كسترينج فى الكود فهضطر ساعتها اعمله Switch cases بالارقام دى و لو غيرنا رقم معين منهم فهحتاج اجرى اغير فى كل مكان فى المشروع فهل يعقل انى كل مره اجى اعمل حاجه مشابهه اجرى اعمل سويتش كيس و اعد اكتب سويتش و اقرف نفسى لأ طبعا لان فى حل احسن بكتير و هو انى استخدم ال Enum عشان الكود يبقى سهل التعديل فيه لان الاينم بيتكتب فى مكان واحد و بفضل استخدمه مكان ما احب فى المشروع كله و لو حبيت اعدل فيه بعدل فى ال Enum بس وبيروح معدل فى كل الاكواد مره واحده و كمان سهل قرايته و لان انا هستخدم الكلام ده على مستوى كبير جدا في المشاريع الكبيره فمش عاوز لغبطه بدل ما اعد اكتب سويتشات ارقام و بتاع لا بكتب الحاجه بالاينم و هو الاينم بيخزن الداتا بتعته على انها int في الداتا بيز عادى خالص اما فى الكود بكتب اسم ال enum متبوع ب الحاجه الى جوه ال Enum ذى   
Gender.Male مثلا و ده مثال قبل استخدام ال Enum و بعد ما استخدمناها شوف الفرق

namespace LearningCSharp

{

class Program

{

static void Main(string[] args)

{

Person[] PerObj = new Person[3];

PerObj[0] = new Person

{

Name="Mahmoud Badawy",

Gender = 0

};

PerObj[1] = new Person

{

Name = "Soso",

Gender = 1

};

PerObj[2] = new Person

{

Name = "Niazy",

Gender = 2

};

foreach (Person Per in PerObj)

{

Console.WriteLine("Name is {0} - Gender is : {1}", Per.Name, GetGender(Per.Gender));

}

}

public static string GetGender(int GenderID)

{

// Gender 0 Male , 1 Female , 2 Unknown

switch (GenderID)

{

case 0 :

return "Male";

break;

case 1:

return "Female";

break;

case 2:

return "Unknown";

break;

default:

return "Nothing";

}

}

}

class Person

{

public String Name { get; set; }

public int Gender { get; set; }

}

}

طيب انا عاوز الكود ده بقه يتكتب بال Enum عشان اوريك حلاوه ال Enum وفيدته

الكود بال Enum هتبقى بتستخدمه ذي ما انت شايف من غير تحويل لانتيجر اما لما تيجى هتخزن في الداتا بيز يا كبير ساعتها حول الى راجعلك من ال enum ل Int

namespace LearningCSharp

{

public enum GenderEnum : int

{

Male = 0,

Female = 1,

Unknow = 2

}

class Program

{

static void Main(string[] args)

{

Person[] PerObj = new Person[3];

PerObj[0] = new Person

{

Name="Mahmoud Badawy",

Gender = GenderEnum.Male

};

PerObj[1] = new Person

{

Name = "Soso",

Gender = GenderEnum.Female

};

PerObj[2] = new Person

{

Name = "Niazy",

Gender = GenderEnum.Unknow

};

foreach (Person Per in PerObj)

{

// In Save in SQL use Convert Enum to Int

Console.WriteLine("Name is {0} - Gender is : {1}", Per.Name, Convert.ToInt32(Per.Gender));

// In Showing it Normaly Do not Convert

Console.WriteLine("Name is {0} - Gender is : {1}", Per.Name, Per.Gender);

}

}

}

class Person

{

public String Name { get; set; }

public GenderEnum Gender { get; set; } // We change Gender From int to Enum

}

}

لاحظ برضوا ان انا ممكن ادي لل Enum ارقام انتيجر و ممكن اسيبه و مديلوش ارقام و ساعتها هو هياخد ارقام الانديكس باى ديفولت يعنى يبدأ أول عنصر يكون ب 0 و يزيد بمقدار واحد بقه كل اما اعمل عنصر داخله

namespace LearningCSharp

{

public enum GenderEnum : int

{

Male = 0,

Female = 1,

Unknow = 2

}

class Program

{

static void Main(string[] args)

{

int[] GenderEnumInt = (int[])Enum.GetValues(typeof(GenderEnum));

string[] GenderEnumString = Enum.GetNames(typeof(GenderEnum));

foreach (int Per in GenderEnumInt)

{

Console.WriteLine("Enum Value : {0} ", Per);

}

foreach (string Per in GenderEnumString)

{

Console.WriteLine("Enum String : {0} ", Per);

}

}

}

}

في الكود الى فات ده انا بعمل كونفيرت للاينم لانتيجر واخزنه في الاراى اوف انتيجر و مره بخزنه في الاراى اوف سترينح لو برجع الاسترينج بتاعه و بعد كده بظهره بص على الكود و هتفهم بقه

MB

##################################################################

Simple Enum Vs Falg Enum

------------------------------------------

بص يا سيدى الفرق الواضح بين الاتنين

ال Simple Enum المتغيرفيه بيشيل قيمه واحده مثال

public enum Gender

{

Male = 0,

Female = 1,

Unknow = 2

}

static void Main(string[] args)

{

Gender gender = Gender.Male;

Console.WriteLine(gender); // Male

}

ال Flag enum : المتغير فيه بيشيل اكتر من قيمه و بيعمل عليهم Bitwise Operations ذى & | ~^ و ال Falg Enum لازم يحتوى على اتربيوت فوقه اسمه [Flags] و لازم يحتوى على ترقيم لكل قيمه فى ال Enum و يفضل يكون ترقيمهم بال Binary لان الفلاجز عمتا لو اديتها 1,2,3,4 هتلاقيه لما تقوله 3 هيجيبلك القيم الى مجموع ترفيمها 3 بمعنى هيجيبلك الى ترقيهم 1 و 2 و لذلك يتم ترقيمها بالنظبم الثنائى بتاع البينارى علشان ميبقاش فى اى عناصر مجموعها يساوى عنصر معين بعديهم فى الترتيب لذلك هتجد الترقيم كما يلى 1,2,4,8,16,32,64,128,256,512 و هكذا بقه الاوبريتورز الى شغاله عليها   
| دى Or بتجيب كل القيم الى بتدهالها كانك بتقولها زائد او ضيف

& دى بتجيبلهم المتشابهات بين القيم   
~ دى اعتبرها ! كانك بتقوله هات القيم المتاحه فيما عدا القيمه الفلانيه

^ دى toggle يعنى لو لقيت الفلاج امسحه لو ملقتوش ضيفه

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

WeekDays saturDay = WeekDays.SaturDay;

Console.WriteLine(saturDay); // SaturDay

WeekDays WeeKEnd = WeekDays.FriDay | WeekDays.SaturDay;

// | Or بيجيب كل المتاح

WeekDays SatMonDay = WeekDays.SaturDay | WeekDays.MonDay;

Console.WriteLine(SatMonDay); // SaturDay , MonDay

// & and بيجيب المتشابه بين الاتنين اينم فاريبول Most Common

WeekDays MostCommonDays = WeeKEnd & WeekDays.FriDay;

Console.WriteLine(MostCommonDays); // FriDay المشترك بين الاجازه و يوم الجمعه هو يوم الجمعه

// ~ Except = ! فيما عدا يعنى لو قولناله هات الويك اند ما عدا يوم الجمعه فهيجيب السيت

WeekDays WeekEndExceptFriDay = WeeKEnd & ~WeekDays.FriDay;

Console.WriteLine(WeekEndExceptFriDay); // SaturDay

WeekDays NewWeekEnd = WeekDays.FriDay | WeekDays.SaturDay;

// ^ toggle يعنى لو لقى الفلاج بيشيله لو ملقهوش بيحطه

WeekDays WeekEndtoggleFriDay = NewWeekEnd ^ WeekDays.FriDay;

Console.WriteLine(WeekEndtoggleFriDay); // SaturDay الجمعه محطوطه فهتتمسح

WeekEndtoggleFriDay = WeekEndtoggleFriDay ^ WeekDays.FriDay;

Console.WriteLine(WeekEndtoggleFriDay); // SaturDay , Friday = WeekEnd الجمعه ممسوحه فهتتحط

}

[Flags]

enum WeekDays

{

None = 0b\_0000\_0000, // 0

SaturDay = 0b\_0000\_0001, // 1

SunDay = 0b\_0000\_0010, // 2

MonDay = 0b\_0000\_0100, // 4

TuesDay = 0b\_0000\_1000, // 8

WedensDay = 0b\_0001\_0000, // 16

ThursDay = 0b\_0010\_0000, // 32

FriDay = 0b\_0100\_0000, // 64

//WeeKEnd = FriDay | SaturDay

}

}

}

MB

##################################################################

Vs

------------------------------------------