C# Summery part-1 By Mahmoud Badawy

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

الملخص يحتوى على

- Design Patterns

المستخدمه فى الشرح   
<https://refactoring.guru/design-patterns/singleton>

MB

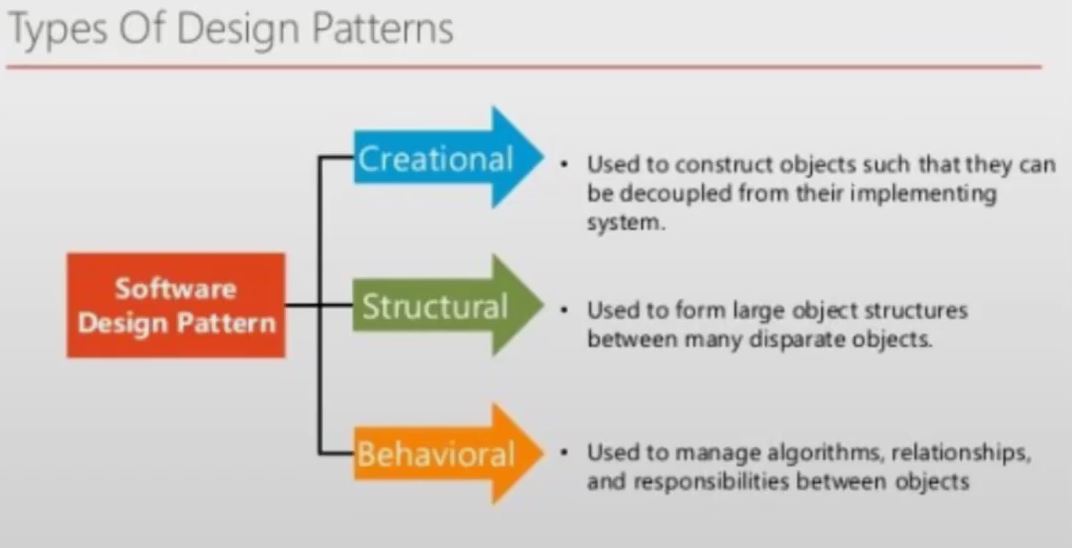
##################################################################

Design Pattern Importance

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

الناس لما عملت ال OOP و ابتدوا يشتغلوا بيها و يعملوا مشاريع كبيره ابتدت تظهر مشاكل سواء فى ال Creation لل instance او سلوكه او طريقه تعامله مع الانستانس المختلفه عنه و ده الى خلاهم فكروا يعملوا حلول باستاندرد محترم يخلى الكود نضيف و فى نفس الوقت يحل المشكله الى بيوجهوها .و هى دى اهميه ال Design Pattern

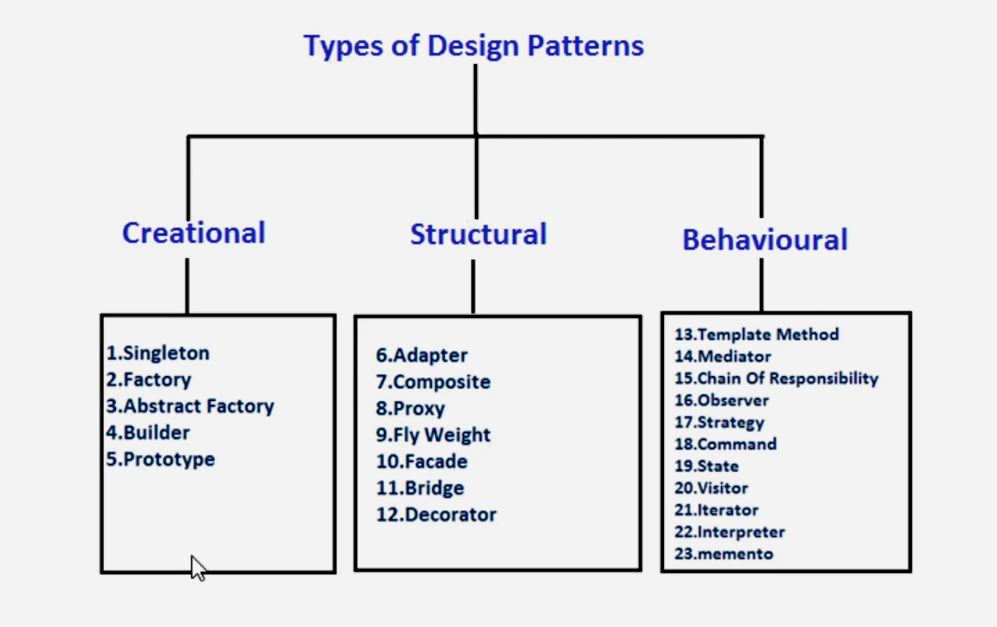
فى تلات انواع للديزاين باترن



ال creational : ده المسؤول عن انشاء ال objects .

ال Structural : ده المسؤول عن العلاقه بين ال objects المختلفه .

ال Behavioral : ده المسؤول عن السلوك الى بيحصل بين ال objects دى الى هو بيبقى سبب فى نقل البيانات بينهم ذى مثلا ال events , Algorithms , states



MB

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

Singleton DP

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

قبل ما يعملوا ال singleton كان الواحد ممكن يعمل كلاس ما و يبتدى يستخدمه فيعمل منه new instance ثم فى مكان تانى يعمل new instance تانى و هكذا لحد ما ممكن يكون بيعمل ملايين ال instances و المشكله ان كل دول محفوظين فى الميمورى و واخدين مساحه كبيره منها و مش بيتمسحم و بالتالى الريسوريس بتاعت الجهاز بيتم استهلاكها بدون داعى و من هنا طلع فكره ال Singleton علشان تحل المشكله دى



ال Singleton بقه فكرتها ان انا هعمل only One Object من الكلاس و بالتالى هيبقى استهلاك الميمورى قلبل جدا جدا و بالتالى الكود هيبقى سريع جدا .

تعالى بقه اوريك المشكله قبل ال SingleTon و الحل لما طبقناه عمل ايه

MB

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

Before Singleton DP

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

المشكله قبل ال singleton هو انى عندى كلاس و كل اما اعمل new instance منه بيروح يخزن ال new instance ده فى الميمورى و للاسف مبيكونش مسح القدام و بالتالى كده انا بعمل instances كتير جدا جدا من class واحد و ممكن مبقاش محتاج كل ده و بالتالى ده استهلاك لموارد الكمبيوتر بشكل كبير و هيبطء البرنامج الى انا بعمله جدا و ده الكود الى بيوضح الفكره

using System;

namespace Before

{

internal class Program

{

static void Main(string[] args)

{

// with no Singleton it will create New 5 Instance in Memory

Console.WriteLine($"########################");

Console.WriteLine($"------------------------");

Counter ObjA = new Counter();

Counter ObjB = new Counter();

Counter ObjC = new Counter();

Counter ObjD = new Counter();

Counter ObjE = new Counter();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

}

}

هنا انا بنادى على الكلاس 5 مرات و كل مره من ال 5 مرات دول بينشأ instance new من كلاس ال counter

using System;

using System.Collections.Generic;

using System.Text;

namespace Before

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public Counter()

{

Count++;

NumberOfInstance++;

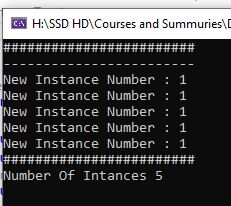
Console.WriteLine($"New Instance Number : " + Count);

}

}

}

و الدليل على انه بينشأ New instance كل مره بعمل منه object هى نتايج البرنامج و اهه علشان تبقى فاهم



**طيب انا بقه مش عاوز كده انا عاوز انى اعمل Creation لل instance مره واحده و فى اى وقت اناديله تانى يروح رايح واخد نفس ال instance و ميعملش غيرها و بالتالى انا كده هتعامل مع only One Instance و بالتالى استهلاك الميمورى هيكون اقل بكتير و بالتالى سرعه البرنامج هتبقى كبيره جدا خصوصا لما نستخدم الباترن ده مع الفانكشنز الى بتتعامل مع heavy Data**

MB

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

After Singleton DP

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

اول حاجه هنحل المشكله الى فوق دى بال Singleton DP و ده بقه بيتعمل اذاى ؟؟؟

1- اول حاجه نخلى ال Constractor برايفيت Private و ده علشان فى ال Client side محدش يعرف يعمل new instance of class

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

2- ثم بعد كده هعرف جوه الكلاس بتاعى ال instance من نوع public static Counter ذى كده و هو ده هيبقى ال instance الوحيد الى بتعامل معاه علشان كده عملته Static يا ريس

public static Counter Instance = null;

3- المفروض بقه هعمل فانكشن كل شغلانتها انها بتشوف لو ال static instance ده ب null فتروح تملاه و تديله new instance اما لو مش ب null معنى كده انه فيه instance فكده ياباشا تروح قيلاله خد القديم استعمله يابا مش هعملك حاجه جديده

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

و الفانكشن دى اتعملت static علشان تبقى Shared و علشان اقدر اناديها باسم الكلاس و طبعا هى بترجع حاجه من نوع الكلاس Counter Class لان هى دى المسؤوله عن انشاء الاوبجيت اصلا او انها تديك الاوبجيكت القديم الى معمول اساسا بدلا من انها تعملك حاجه جديده فتهلك الميمورى

و بالتالى الكود بالكامل اهه

using System;

namespace After\_Simple\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########################");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

}

}

ثم الكلاس بقه

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Simple\_Singleton

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

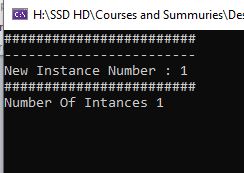
return Instance;

}

}

}

النتيجه اهه يا صديقى بقى مش بيعمل غير One Instance من الكلاس



MB

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

After Singleton DP But Using **Inhertance** (problem) solved By **Sealed** Key

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

بما انك عامل ال Constructor access modidier نوعها private فانت طول ما انت بره ال class و بتحاول تعمل inheritance مش هيرضى و ده المطلوب فعلا و لكن اول ما هتحط الكلاس دى جوه الكلاس بمعنى اول ما هتعمل علاقه وراثه بانك تحط كلاس ال child جوه كلاس ال parent ساعتها هيقبله عادى و هيعمل منه كمان مليون new instance عادى و ده بيضرب ال singleton pattern فى مقتل

و هنا هوريك بالكود ايه المقصود يا صديقى

using System;

using System.Collections.Generic;

using System.Text;

namespace Inhertance\_Problem\_Singleton

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

//ChildCls inherite from Counter Class that will Make easy to create Million of instance by Child Class

public class ChildCls : Counter

{

}

}

}

و هنا هنادى للكلاس العادى الى معمول فيه singleton و هنادى لل Child هتلاحظ انى طول ما بنادى للعادى مفيش اى instance جديد تم انشاءه و لكن اول ما ابتديت انادى للكلاس ال child بقى يعمل new instance كتير جدا بقه

using System;

namespace Inhertance\_Problem\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

// with no Singleton it will create New 5 Instance in Memory

Console.WriteLine($"########### Before using Child #############");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.WriteLine($"########### After usning Child #############");

Console.WriteLine($"------------------------");

Counter ChildObjA = new Counter.ChildCls();

Counter ChildObjB = new Counter.ChildCls();

Counter ChildObjC = new Counter.ChildCls();

Counter ChildObjD = new Counter.ChildCls();

Counter ChildObjE = new Counter.ChildCls();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

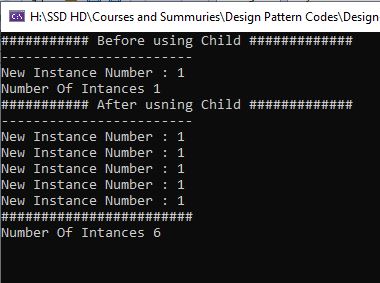
Console.ReadKey(true);

}

}

}

و ادى النتيجه علشان تفهم



**طب المشكله دى احلها اذاى ؟** احلها انى اخلى الكلاس الى بعمل عليه ال Singleton انه يبقى **Sealed** كلاس يعنى مبيورثش اى كلاس باى طريقه نهائى و بالتالى همنع الموضوع ده نهائيا و ابقى ضمنت انى هعمل only one instance تانى

using System;

using System.Collections.Generic;

using System.Text;

namespace Using\_Sealed\_ToSolve\_Inheretance\_Problem\_Singleton

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

//Sealed Will Stop any kind of Inhertance in Singleton Class

//public class ChildCls : Counter

//{

//}

}

}

كلاينت كلاس

using System;

namespace Using\_Sealed\_ToSolve\_Inheretance\_Problem\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Sealed class #############");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

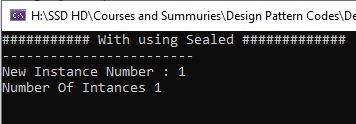
Console.ReadKey(true);

}

}

}

و بالتالى اصبح غير مسموح الوراثه فى الكلاس الى عامل فيه سنجلتون و بالتالى مبقاش ينفع يعمل اى new instance تانى يعنى اتقفلت تانى بالضبه و المفتاح و النتيجه هتكون بالشكل ده تانى



MB

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

After Singleton DP But Using **Multi Threads** (problem) solved By **Lock**

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

المشكله الحاله بقه انى لو استخدمت Multi Thread ساعتها الكود هيخش بال thread الاولى يلاقى ال instance ب null فهيروح عامل new instance و ال thread التانيه هتبقى شغاله مع ال thread الاولى فى نفس الوقت فبرضوا هتخش معاها علشان تشوف ال instance فهتلاقيها هى كمان ب null فهتروح عامله new instance و بالتالى الشغل ال parallel بوظ فكره ال Singleton تانى بس حل المشكله ان لما thread تخش تروح عامله Lock على الكود بحيث ان ال Thread التانى ميقدرش يخش و هى شغاله و بالتالى هيخشوا بعد بعض مش مع بعض و بالتالى كده الى هتخش ورا التانيه هتلاقى ان ال instance مش فاضى فمش هتعمل new instance و بالتالى كده رجعنا حلينا المشكله بتاعت ال multi threads هى كمان

اوريك بقه المشكله على كود

using System;

using System.Threading.Tasks;

namespace SingletonWithOutThreadSafety

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Parallel Or Threads #############");

Console.WriteLine($"------------------------");

Parallel.Invoke(

() => Method\_1(),

() => Method\_2(),

() => Method\_3()

);

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

public static void Method\_1()

{

Counter ObjA = Counter.GetInstance();

}

public static void Method\_2()

{

Counter ObjB = Counter.GetInstance();

}

public static void Method\_3()

{

Counter ObjC = Counter.GetInstance();

}

}

}

و كلاس السينجلتون اهه مغيرناش فيه اى حاجه خالص

using System;

using System.Collections.Generic;

using System.Text;

namespace SingletonWithOutThreadSafety

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

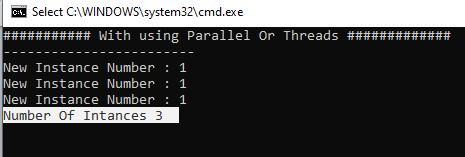
}

return Instance;

}

}

}



ولان ال parallel هيخلى الكود يشتغل على ال 3 فانكشن فى وقت واحد و بالتالى فى كل مره الكود هيروح يشوف هل فيه instance معمول ولا لأ فهيلاقيه فى الثلاثه لسه مش معمول فهيروح ينشأ واحد جديد لل 3 كلهم و النتيجه ان فى 3 new instance و بالتالى فكره ال singleton انضربنت تانى

حل المشكله اننا بقه جوه كلاس ال Singleton نعمل Lock على كود الى بيعمل فحص اذا كان فى instance و لا لأ بحيث مفيش Thread يخش المكان ده مع التانى و يخشوا بالدور الى يخش الاول يقفل الباب وراه لحد ما يخلص الى بعده يخش و يقفل وراه و هكذا و بالتالى لما التانى يجى بعد ما الاول خلص شغله هيلاقى ان فى instance و بالتالى مش هيعمل واحد جديد خلاص

العمليه دى يطلع عليها Thread Safty و كل الى بنعمله حاجتين

اول حاجه بنعرف حاجه من نوع Static ReadOnly Object

public static readonly object InstanceLock = new object();

الحاجه التانيه بنعمل lock على الكود الى هنعمل فيه ال Check عن اذا كان فى انستانس قديمه ولا لأ

public static Counter GetInstance()

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

}

return Instance;

}

الكود كامل اهه مفيش اى تغيير نهائى فى كلاس ال program كل الجديد فى كلاس ال Counter الى بعمله singleton

using System;

using System.Threading.Tasks;

namespace Singleton\_After\_ThreadSafety

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Parallel and ThreadSafety #############");

Console.WriteLine($"------------------------");

Parallel.Invoke(

() => Method\_1(),

() => Method\_2(),

() => Method\_3()

);

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

public static void Method\_1()

{

Counter ObjA = Counter.GetInstance();

}

public static void Method\_2()

{

Counter ObjB = Counter.GetInstance();

}

public static void Method\_3()

{

Counter ObjC = Counter.GetInstance();

}

}

}

**ده الكلاس الى بعمل عليه singleton و الى التغيير تم عليه**

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_After\_ThreadSafety

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

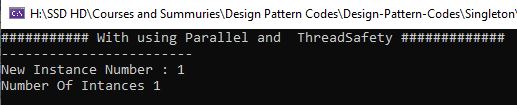
}

return Instance;

}

}

}



طيب دلوقتى فى مشكله كبيره هتواجهنا بقه ايه هى يا معلم ؟؟

كتر استخدام ال Lock بيستنذف الميمورى و بيبطء الابلكيشن و بالتالى انا كده لازم اعمل حاجه تخلى ان ال lock ميترنش الا لما اعوزه للشديد القوى فقط و بالتالى الابلكيشن يبقى سريع جدا جدا

طب حل المشكله ايه يا سيدى ؟؟

double check for thread safty

اعمل check فوق ال lock نفسه بال instance و اقوله فى حاله ان مفيش instance و ضرورى اعمل new instance خلاص ياعم خش على ال lock غير كده مش عاوز اشوف خلفتك اساسا

public static Counter GetInstance()

{

if (Instance == null) // Double thread Safty

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

}

}

return Instance;

}

دى كل التغييره الى هتتعمل فى الكود علشان يبقى double thread safty ياكبير فمش محتاجه كود كامل بقه

MB

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

Singleton DP Using Eager Loading Vs Lazy Loading

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

اول حاجه الفرق بين ال lazy loading و ال Eager loading

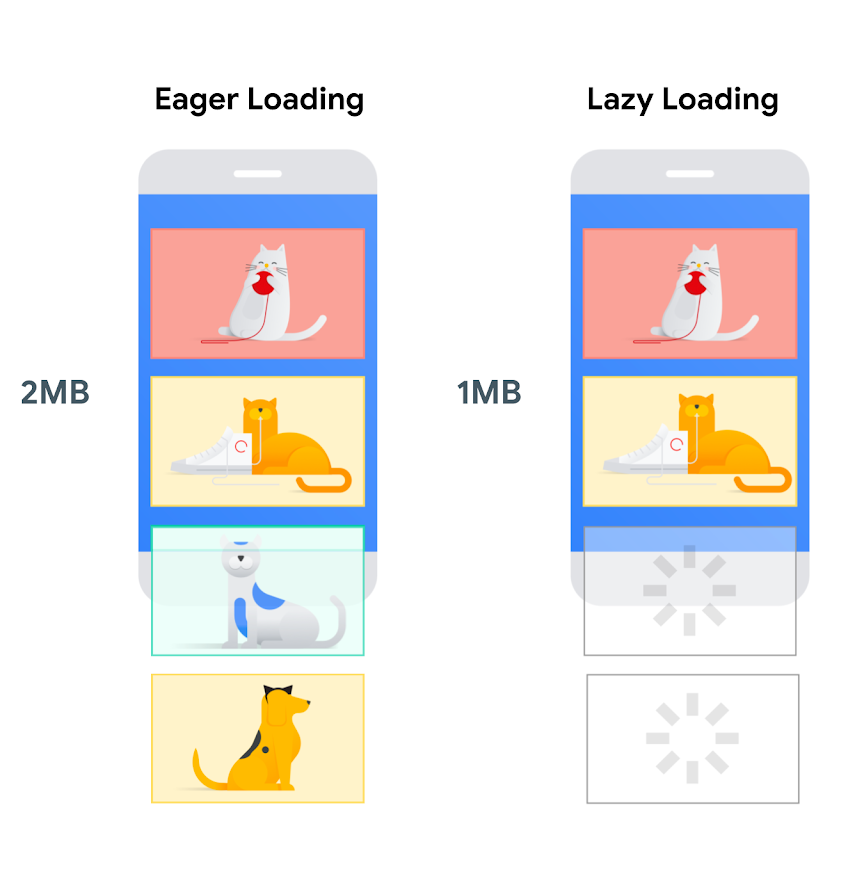
بص يا سيدى ال Lazy Loading ده بستخدمه لما اكون مش عاوز اجيب الداتا كلها و عاوز فقط اجيب الداتا الى محتاجها فقط بمعنى دلوقتى انا عندى كميه داتا كبيره جدا جدا فى البنك عن العملاء مثلا و انا دلوقتى عاوز اخش عليها اعرف بيانات اتنين او تلاته عملاء هيبقى ايه المفيد انى اجيب كل الداتا ؟؟ ولا اي حاجه بالعكس كده السيستم هيموت و يقولى حرام عليك انت كل مره تيجى تجيب بيانات عن واحد تروح رايح جايب كل البيانات الى فيا و تستهلك الميمورى بتعتى على الفاضى لا ياعم خد شوف الى انت محتاجه بس و متتعبنيش معاك

طيب ال Eager دى يا سيدى لما بنستخدمها لما تكون عاوز اجيب الداتا كلها مره واحده و طبعا بنستخدمها ففط مع الداتا القليله يعنى و ليكن انا عاوز اجيب بيانات عن محمود الى هى الاسم و السن و الصوره و الميلاد فهل يعقل انى اجيب كل واحد لما احتاجه بس و لا طالما الداتا صغيره كده اروح اجيبها كلها عادى ما هى تكلفه انى اجيبها كلها اخزنها فى الميمورى اقل بكتير من انى كل شويه اكلم السيرفر انى عاوز اسم محمود عاوز صوره محمود عاوز مقاس شورت محمود

و بالتالى الفرق بين الاتنين ان ال lazy بتجيب الداتا لما اطلبها فقط و بنستخدمها مع الداتا الضخمه

اما ال Eager بتجيب الداتا كلها الى انا عاوزها مره واحده و بنستخدمها مع الداتا الصغيره

و ادى صوره معبره توضح الفرق بين الاتنين



طيب احنا فى الاكواد السابقه دى كنا شغالين ايه ؟؟

Lazy loading و ده لان و انت بعمل Intailization لل instance قولتله ياباشا انت ب null و مش هملاك الا لما الكلاينت ينادينى و يعمل Check عليا لو انا لسه معملتش ال new instance هخليك تعمل غير كده مش هتعمل و بالتالى كده الكود ده LazyLoading

طب احوله اذاى الى Eager Loading ؟

تعديله هبله صغيره خالص

بدل من انى اقوله ال instance = null فى ال Defination هقوله بيساوى new instance و بكده حصل Creation لل instance بمجرد ما البرنامج اتفتح و راح حجز مساحه فى الميمورى حتى لو انا مش عاوز اكلم الكود من اى كلاينت سايد

و طبعا هعمله ReadOnly علشان محدش يقدر يكتب عليه تانى خلاص

و ادى الكود بالكامل بعد تحويله الى Eager Loading

مفيش تغيير فى الكلاينت سايد نهائى كل التغيير هيبقى فى كلاس ال Singleton

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_Eager\_Loading

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

// Eager loading add ReadOnly + New intance

//public static Counter Instance = null;

**public static readonly Counter Instance = new Counter();**

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

//if (Instance == null) // Double thread Safty

//{

// lock (InstanceLock)

// {

// if (Instance == null)

// {

// Instance = new Counter();

// }

// }

//}

return Instance;

}

}

}

كده بقى الكود Eager loading بمجرد ما خليت الديفينيشن فيه ReadOnly و اديته ال new instance فوق مع الديفينيشن و مسحت ال Double Safty Check لان خلاص الانستنس بعد كلمه ReadOnly اصبح مبيعملش غير انستانس واحد بس

طيب لو انا حابب احوله ل LazyLoading بس بشكل اسهل من الاكواد السابقه اعمل ايه ؟

بص يا سيدى سى شارب عملت حاجه اسمها Lazy و دى Key word جاهزه تقدر تستخدمها تسهل عليك الدنيا خالص و تخلى الكود اسهل بكتير و متلخص على الاخر و شكل الكود ساعتها هيبقى كده

كل الى هتعمله انك هتحط lazy فى ال Defination و هتنزل تقول لل instance فى الفانكشن انا عاوز ال Value بتعتك ذى ما انت شايف الدنيا سهله اذاى و بسطر واحد عمل كل شغل ال thread safty , double check و كل حاجه يا صديقى

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_Lazy\_Loading

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

// Eager loading add ReadOnly + New intance

//public static Counter Instance = null;

//public static readonly Counter Instance = new Counter();

public static readonly Lazy<Counter> Instance = new Lazy<Counter>(()=> new Counter());

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

return Instance.Value;

}

}

}