شو يعني Entity Framework ؟

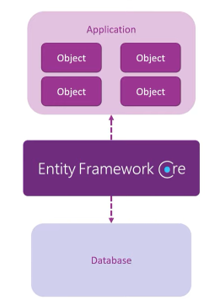
هي بمنتهى البساطة ORM: Object Relational Mapper بمعني أنها تسمح لل DotNet developers انهم يتعاملو مع الداتا بيز بإستخدام ال Donet Object

يعني بدل ماأتعامل مع الداتا بيز عن طريق ال Stored Procedures أو عن طريق ال Statements العادية بصير أتعامل معها عن طريق Classes ضمن البروجيكت تبعي

قديما كانو يتعاملو مع الداتابيز عن طريق ال Ado.Net : هي الطريقة بتخليك تتعامل مع الداتابيز بنفس التعليمات اللي بتكتبها على SQL نفسه اما عن طريق Stored Procedure وبتعملها call عن طريق الدوت نت بروجيكت أو أنك تكتب الselect statement جوا الكود ضمن البروجيكت نفسه (وطبعا هي الطريقة غير مفضلة نهائيا لعدة أسباب أشهرها لتجنب ال SQL Injection) طبعا كل مرة بتطر انك تفتح Connection مع الداتا بيز وبترجع بتسكره بس تخلص تعليماتك

طبعا هي الطريقة أصعب من ال EF لكن ال Ado.Net بستخدمها بالكويريات المعقدة

ال Ef هي عبارة عن Layer بين ال App والداتا بيز



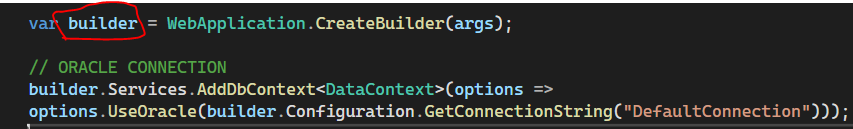
حتى استخدم ال EF

1. يجب تنزيل الباكيج تبعها Microsoft.EntityFrameworkCore
2. Microsoft.EntityFrameworkCore.Tools
3. الباكيج الأخيرة هي خاصة بالداتبيز اللي بدك تشتغل عليها مثلا:

Oracle: Oracle.EntityFrameworkCore

Sql server: Oracle.EntityFrameworkCore.SqlServer

* بعد ماعملت install للباكيجات اللي محتاجها, محتاج يكون عندي كلاس يكون حلقة الوصل مابين ال App والداتابيز (ApplicationDbContext) وخليه يعمل Inheritance من الكلاس DbContext الموجود ضمن ال EF نفسها (اللي بحط فيها ال DbSet)
* بعدين بضيف ال ConnectionString ضمن ال AppSetting.Json
* ضمن ال program.cs بضيف الإتصال مع الأوراكل (ال builder بالأساس بكون موجود)



مفهوم ال Up / Down في ال Migration

* ال Up بحط فيه التغييرات اللي عملتها على ال Domain Models (يعني الكلاسات تبعك)
* ال Down بحط فيها عكس ال UP ( بيمسح اللي بينعمل في ال UP ) مشان اذا بدي ارجع بالداتا بيز لورا (مثلا بدي ارجع من ال Migration رقم 10 لل Migration رقم 6) ببساطة هوبيعمل Run لل Down بين ال 6 و10
* Migration Rollback:

لكل Migration بعمله فيني احذفه قبل ما اعمل Update-Database عن طريق Remove-Migration

في حالات بتصادفني انه بدي الغي Migration بعد ماكون عامل Update-Database بهي الحالة ال Reomve-Migration ماعاد تنفع ورح يعطيني الايرور التالي:



الحل بسيط جدا فقط بكتب اسم ال Migration اللي بدي ارجعله وال EF لحالهل بتنفذ ال Down

يعني لوعندي 10 Migrations وبدي جع الداتابيز للوضع اللي كانت فيه بالرقم **6** بكتب:

Update-Database **6**

Update-Database **-migration:0 (في حال كنت واقف أول شي وبدي ارجع لل 0)**

طبعا بعد القيام بهذه العملية يتم حذف الاسطر الخاصة بال Migration من الجدول \_\_EFMigrationsHistory الموجود ضمن الداتابيز

* ملخص: لوبدي نفذ فقط 8 و9 -< رح يبدأ من بعد 7 (يعني 8) ويقف عند 9

Add-Migration (start-1) (end)

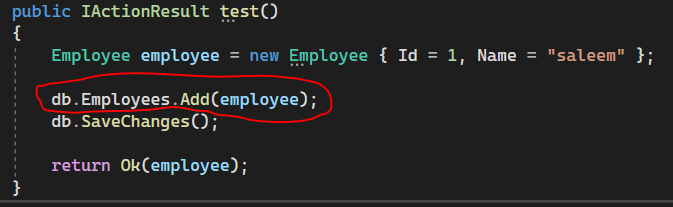
* Script-migration بولدلي ال script على شكل SQL statements

اضافة سطر جديد لجدول

ملاحظة مهمة :

يجب أن يكون بالمشروع جزء خاص بال Ef للتعامل مع الداتابيز مثل ال Repository Pattern (رح ندرسه بكورس لحاله)

بتعامل مع الكلاسات كانها List يعني بضيف وبحذف وبعدل على الجدول وكأني شغال على List بمنتهى البساطة كمايلي:

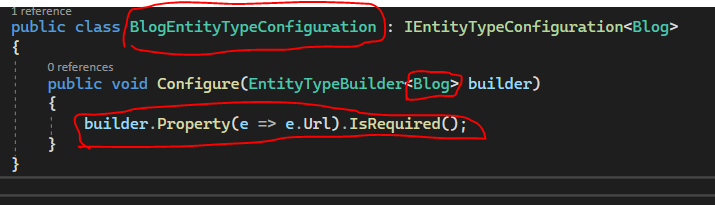


طول ماني عامل db.SaveChanges رح **يضل شغلي بالميموري** وبس اعمل db.SaveCanges بتنتقل التعديلات على الداتابيز

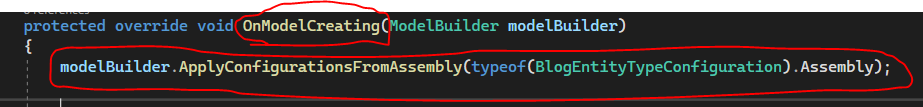
DataAnnotations vs FluentApi

* ممكن تكون ال DataAnnotation أسهل لكن استخدام ال FluentApi بخليلك شكل ال **Domain Models** أنضف ومافي عجقة (خلص المودل بمثلك شكل الجدول بالداتابيز وبس)
* **استخدام ال FluentApi بحققلك مفهوم ال Separation Of concerns مشان اكتب Clean Code**
* كمان بحط الكود الخاص بال fluentApi بفايل لحاله وبعمله Reference على ال DbContext
* أنا ممكن أعمل لكل Model عندي فايل لحاله يحتوي على ال FluentApi وبعمل عليه Reference كما موضح بالخطوات التالية:

1. أنشأ مجلد اسمه Configuration
2. بعمل ضمنه كلاس كالتالي:



1. استدعيه ضمن ال OnModelCreating في ملف ال DataContext :



On Delete

1. Restrict: بتحذف الأبناء بالأول بعدين بتحذف الأب (Recommended)
2. Cascade: بس حذفت الأب بتحذف أولاده

عندي ثلاث طرق بعرف من خلالها ال EF على ال Domain Models اللي عندي(يعني هنن رح يصيرو جداول بالداتابيز وليس مجرد Models عادية):

1. الطريقة العادية DbSet<Entity>
2. عن طريق العلاقة بين كلاس وكلاس (مثل ال Departments + Employees بينهم علاقة One-To-Many)
3. عن طريق ModelBuilder.Entity<TableName>(); ضمن ال DataContext

Column Property

* builder.Property(e => e.Url).HasColumnType("VARCHAR2(2000)");
* builder.Property(e => e.Url).HasMaxLength(50);
* builder.HasKey(e => e.Id).HasName("PK\_BLOGS"); **(لاحظ انه على مستوى الجدول وليس العمود)**
* builder.HasKey(e => new { e.Id, e.Url }); **(Composite PK)**
* builder.Property(e => e.Url).HasDefaultValue("test");
* builder.Property(e => e.Url).HasDefaultValueSql("getdate()");

**لما بدي ابعت Default value عبارة عن sql statement**

* builder.Property(e => e.Url).HasComputedColumnSql("[firstName +' '+ lastName]");

Computed Column:

لآزم أنشأ العمود full name بعدين عند الإضافة بعبي قيمته من ال first+last name

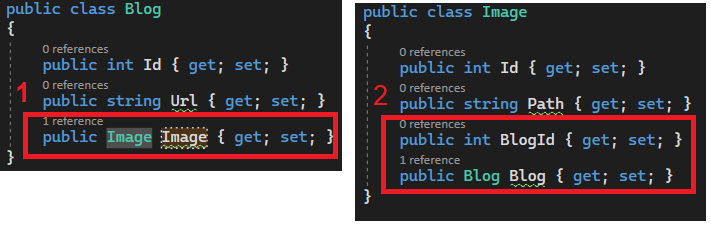
انتبه هون بيرجع بيعمل Update للقيم القديمة وبيرجع بعبيها بالقيم الجديدة

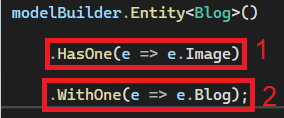
شو بتفهم من هالحكي؟

ال computed column هو عبارة عن virtual column في كل مرة بعمل استرجاع للداتا هو بيعمل Update نفسه من القيم الجديدة

Relationships

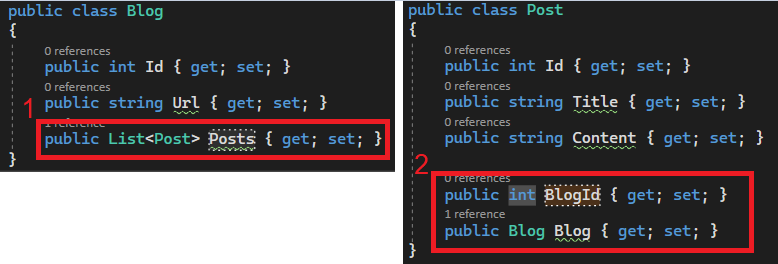
1. One-To-One:

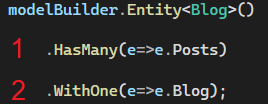




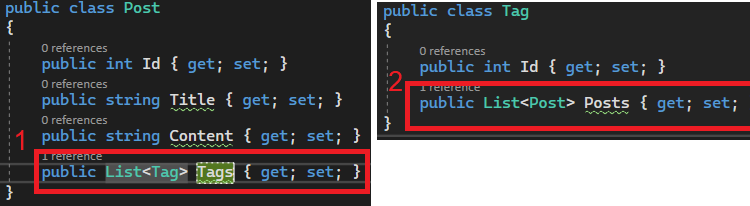
* **HasOne** من اليسار لليمين وكمان بدي حدد نم اليمين لليسار **WithOne**

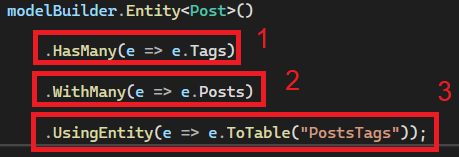
1. One-To-Many:





1. Many-to-Many:





يفضل كتابة العلاقة بشكل كامل مع تحديد اسم جدول كسر العلاقة(سهلة بس طويلة شوي)

ملاحظة:

* كل شي بين الأقواس اللي بستخدمهن مع ال **Has and With** اسمهن **Navigation Property**وبحط ضمن كل قوس ال navigation property الموجودة ضمن الكلاس نفسه وليس الوجهة

Index

طبعا لازم تكون فهمان كل مفاهيم ال Indexes

* modelBuilder.Entity<Blog>().HasIndex(e => e.Url).IsUnique();

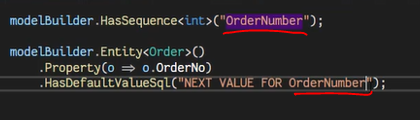
هون بنزل طول الأحرف من ال max الى 450 بشكل تلقائي

* modelBuilder.Entity<Blog>().HasIndex(e => e.Url)

.HasFilter("[Url] IS NOT NULL");

Sequences

* ال PK بزود على العامود 1,2,3,.. وبجدول تاني كمان ببلش من 1,2,3… لكن ال sequence بكون **shared بين أكثر من جدول (يعني بالجدول الأول بس يوصل لل3 واستخدمع مع الجدول التاني ببلش من ال4 وهكذا...)**



Query Section

* بما أنه رح نتعامل مع الداتا بيز فبالتالي رح نحتاج ل object من **ال DbContext اللي بمثل حلقة الوصل بين ال App وال Database**
* OrDefault :

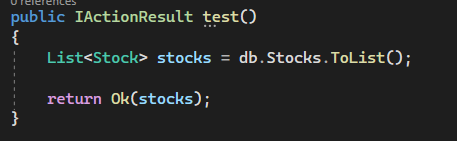
اذا راح على الداتابيز ومالقى ولا سطر بحقق الشرط برجع Null بدل مايضرب

**Exception: contains no elements**

فبالتالي بشوف اذا هي Null برجع مثلا Not Found أو حسب ال Business تبعي

* طبعا ضمن الأقواس فيني مرر الشرط اللي بدي ياه فبالتالي بالأول بيعمل filter للداتا بعدين بطبق الفنكشن على **الداتا المفلترة وليس الكلية**

1. Select All

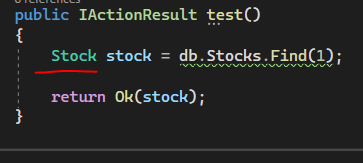


1. Select One Row:

عندي أكثر من طريقة:

* Find (Pk Number)

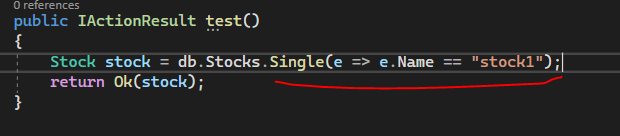
اذا معي ال Pk تبع العنصر



* Single ()

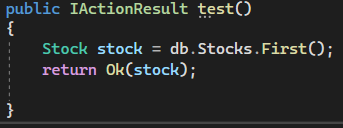
بستخدمها فقط لما كون متأكد أنه عندي row واحد فقط رح يحقق الشرط المبعوت معها (اذا مانك متأكد أنه رح يرجعلك بس سطر لاتستخدمها)

يعني ممكن استخدمها مع العمود اللي بكون Unique



* First ()

من اسمه برجع أول سطر



**ملاحظة الفرق بين ال First وال Single:**

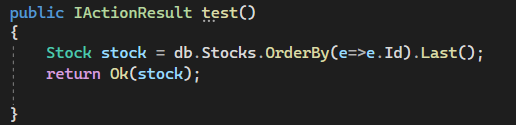
ال Single بستخدمها فقط لما كون متأكد أنه رح يرجعلي سطر واحد(يعني اذا بطبقها على المثال تبع ال First رح تضرب

Exceptions: Contains many Elements

بينما ال First بتعالج هي الحالة وبتجبلك أول سطر فقط من الداتا

* Last ()

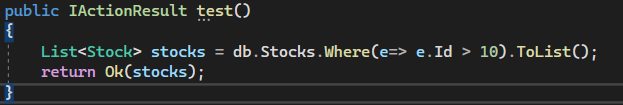
من اسمها بترجع اخر قيمة بس بدها معها OrderBy



1. Filtering Data Using Where

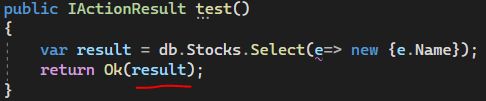
لما استخدم ال where بنفذ الجملة على الداتا بيز (server side) وبرجعلي القيم مفلترة (يعني لوعندي 1000 سطر وقلتله where id>500 ) يعني رح يرجعلي بس 500 سطر على التطبيق

لكن اذا ما استخدمتها وجبت كل الداتا باستخدام ToList وبعدين صرت أعمل فلتر عليهن بالتطبيق (Client Side) هي كارثة وبتبطأ الشغل كثير ومانها best practice



ToList() دغري بتبعت التعليمة على الداتا بيز حتى تتنفذ وترجع بالنتيجة

أما ال ()Select )واللي معها بنفس التصنيف( بتضل عندي ومابتنفذ إلا لما قلها .ToList() أو return() كما في الشكل:

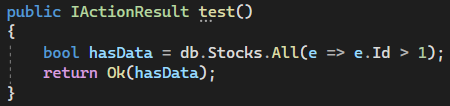


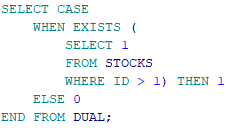
1. Any vs All

* Any

معناتها اذا لقيت **أي** سطر رجع True **ولاتكمل بحث عن اللي بعده**  وإلا رجع False

**ممكن تشتغل من دون lambda expression**

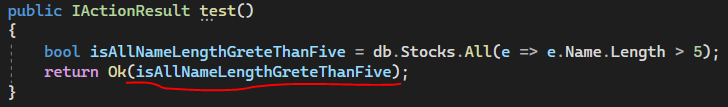


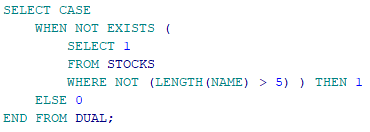


* All

عكس Any لازم كل الداتا تحقق الشرط اللي مرقتلها ياه **ولازم تمر على كامل الأسطر لتتحقق من الشرط**

**إجباري بدها lambda expression**

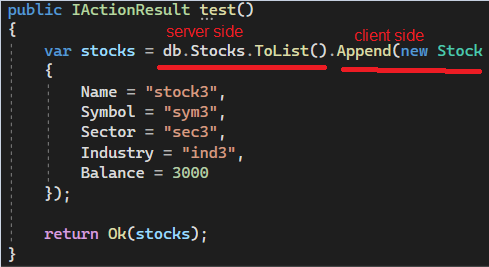




1. Append vs Prepend

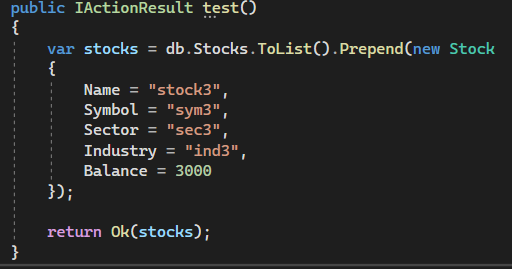
* Append

بدي ضيف قيم اضافية **باخر** الداتا اللي رجعتلي (الإضافة ضمن الميموري Client Side)



* PrePend

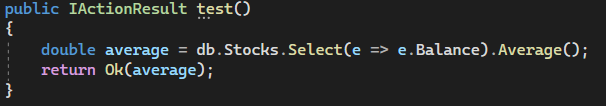
بدي ضيف قيم اضافية **بأول** الداتا اللي رجعتلي (الإضافة ضمن الميموريClient Side )



Aggregation Query Section

* Average

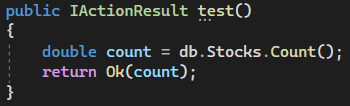
المتوسط الحسابي لمجموعة الأرقام



* Count

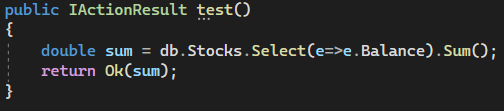
عدد القيم المرجعة

استخدم ال LongCount مع الأرقام الأكبر



* Sum

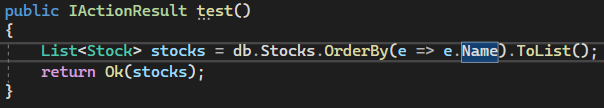
مجموع القيم المرجعة



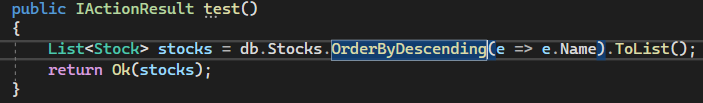
* Max
* Min

Data Sorting Using OrderBy

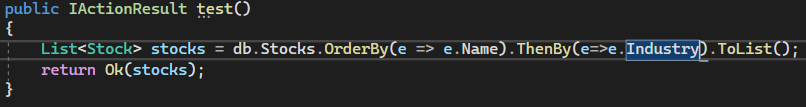
* Asc (Default)



* Desc

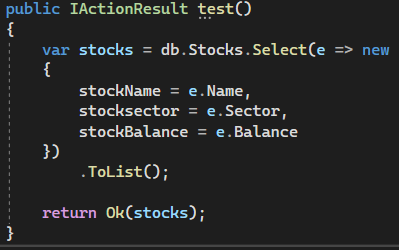


* ThenBy

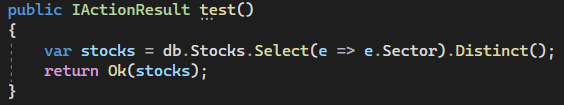


Select

* بحدد شو الأعمدة اللي بدي اعملها select (وهاد الشي كتير مهم لل performance موضروي جيب كل الأعمدة وأنا بدي عمود واحد)
* الشغلة التانية بتخليني غير شكل الداتا (يعني بغير أسماء الأعمدة اللي رح ترجع)



Distinct



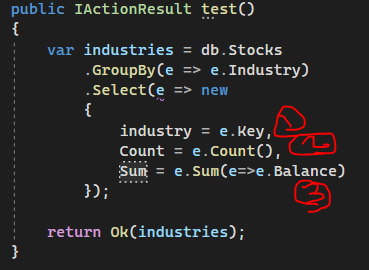
Skip Take

* تستخدمهن مع ال pagination
* بتاخد من بعد ال skip بواحد الى طول ال take



Group by

* بدها معها **Select** بحط ضمنها ال key بعدين تابع التجميع (لازم يكون اله اسم) متل الصورة (1+2+3) يعني فيني حط اكتر من تابع تجميع مع بعض



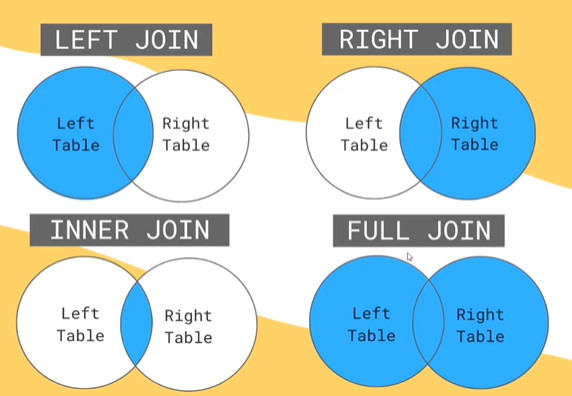
Join

بدها مراجعه

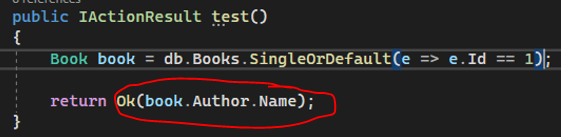
<https://www.youtube.com/watch?v=_dQ8qIskm8Y&list=PL62tSREI9C-cHV28v-EqWinveTTAos8Pp&index=53>

<https://www.youtube.com/watch?v=rCdGtt3bqMo&list=PL62tSREI9C-cHV28v-EqWinveTTAos8Pp&index=59>

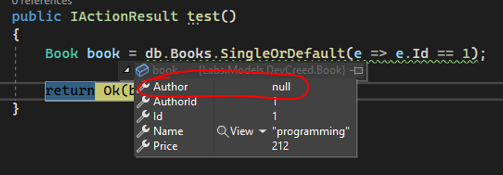
* IQueryrable : يعني مافي شي راح ع الداتا بيز وحتى نفذها ببساطة بحطها ضمن قوسين بعدين بعملها ToList



Eager Loading

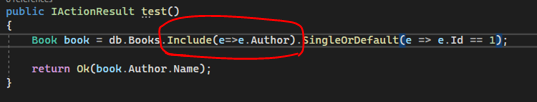


* بالمثال السابق رح يضرب Error لأني أنا حملت معلومات الكتاب فقط وعم اطلب معلومات ال Author ومن ال navigation property (فقط قمنا بتحميل جدول الكتب وليس جدول المؤلفين)
* شوف لما عملت debugging بينت معلومة ال Author = null

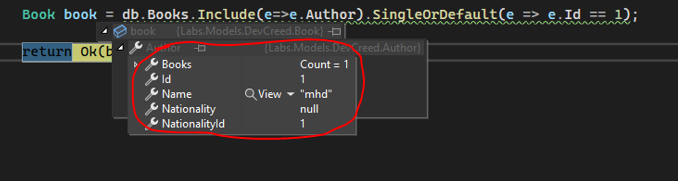


* حتى حل المشكلة السابقة بستخدم ال Eager Loading بإستخدام ال Include (من الأخير هي Join)

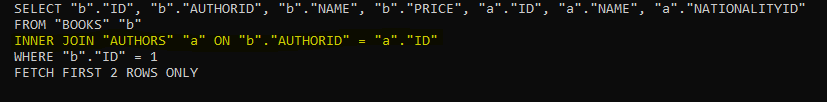
using Microsoft.EntityFramworkCore



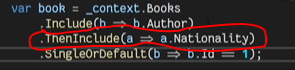
* لاحظ هون رجع معلومات ال Author



* هو من الاخير استخدم ال Join



* فيني اربط مع جدول تالت عن طريق ThenInclude



* ملاحظات مهمة:
* فيني حط الشروط اللي بدي ياها ضمن ال Include

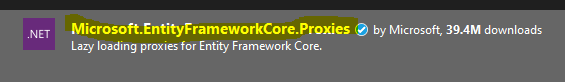
Include(e=>e.Author**.where(e=>e.firstName=”Saleem”)**)

* ال Eager Laoding بأثر جدا جدا على أداء التطبيق

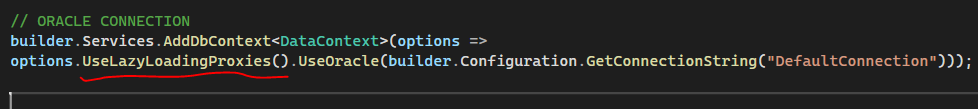
 Lazy Loading

* عكس ال Eager Loading بيعمل load لل related tables لما access ال navigation property اللي عندي
* في عندي كذا طريقة حتى طبقه, رح نستخدم الطريقة التالية بدها ثلاث خطوات:

1. لازم نزل الباكيج التالية:

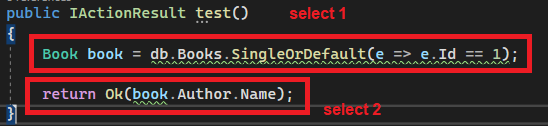


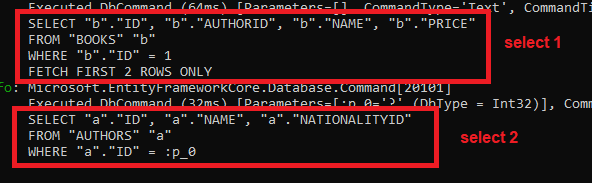
1. بحط UseLazyLoadingProxies()قبل ال UseOracle



1. لازم كل ال navigation properties يكونو virtual

بكتب التعليمة متل مابدي وبدون ال Include ولحاله بجيب الداتا على دفعتين بإرسال 2 selcets للداتابيز





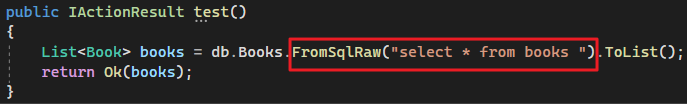
ملخص الفرق بين النوعين

* Eager Loading: load all data and related date one time using 1 query
* Lazy Loading:

بيعمل Load لل **Related data** لما استخدم ال navigation property بإستخدام 2 quires

Select Data Using SQL Statement or Stored Procedure

1. Using Normal Sql Statement



1. بنفس الطريقة اذا بدي استدعي Procedure or Function

* بحط الإسم مع البارميترات لكن بدك تنتبه من البارمترات المبعوتة وتعملها check مشان مايصر معك Sql Injection (يعني ممكن يبعتلك تعليمات تعمل Drop database)
* اذا عندي الداتا الراجعة عبارة عن join من أكثر من Table بعملها dto (بحط فيه كل الأعمدة الراجعة من الداتابيز وبضيفه ك DbSet وبالتأكيد لازم فهم ال EF أنه هاد مانه Table)

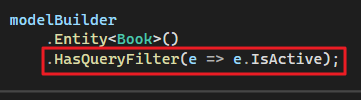


* Exclude bookDto From Migration:



Global Query Filters

بدي حط filter بستخدمه مع كل شي لهاد الجدول مثلا بدي جيب بس الكتب ال IsActive=1



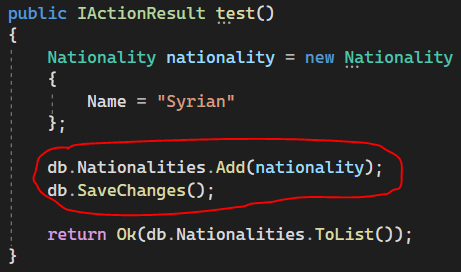
CRUD Operations

* ملاحظة كتير مهمة:

طبعا مالازم تنسى أنك بتتعامل مع الجدول وكأنك بتتعامل مع List

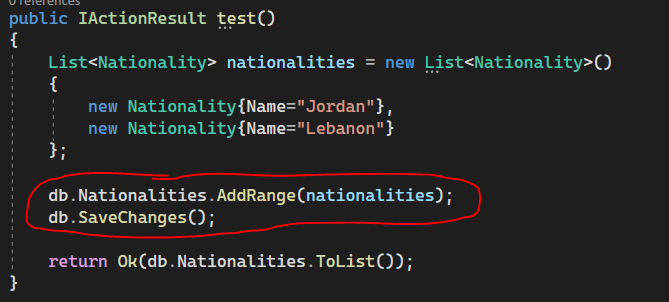
1 - Create

1. Add New Record (Add one):



1. Add Multi Records (Add Range):

* طبعا إذا عندي أكتر من Transaction ورا بعضها لازم أعمل save changes بس خلصهن كلهن



1. Add on Two Tables or More:

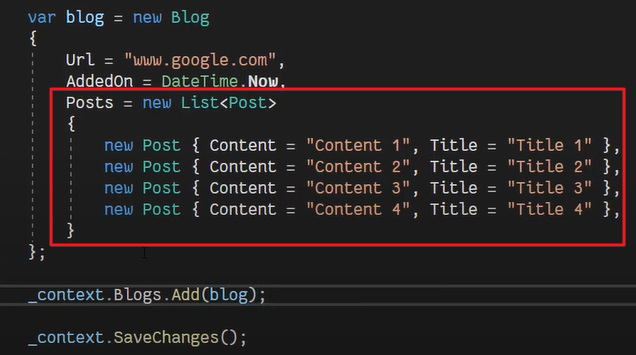
ملاحظة مهمة:

اذا بدي ضيف Author جديد مع nationality اله جديدة كمان (معظم العالم بهي الحالة بضيفو ال nationality بالأول بعدين بياخدو ال Id تبعها وبضيفوها مع سطر ال Author بعدين بخزنو ال Author record)

ال EF بتقدملك Option أسهل بحيث أنك تعمل Save للسطرين مرة واحدة على الجدولين كما موضح بالشكل التالي:



* هي نفس الفكرة بس اذا بدك تضيف لأكثر من سطر في جدول الإبن (يعني العلاقة بينهم One To Many):

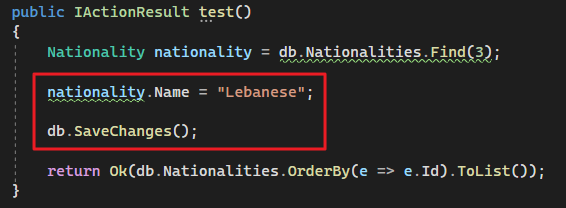


2 - Update

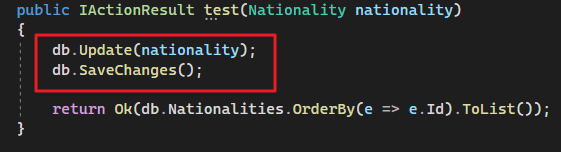
1. Update Record:

عندي عدة طرق حتى أعمل Update :

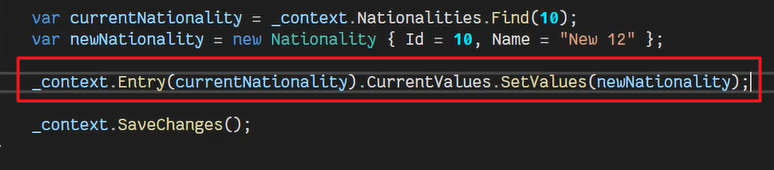
1. لاحظ الشكل بمجرد اني جبت السطر اللي بدي عدله وعملت عليه Update لل properties اللي بدي دغري بستخدم بعدها SaveChanges() (طبعا لأن الداتا بيز بتكون شغالة ب Tracking Mode)



1. الحالة الطبيعية بكون اليوزر باعتلي ال object بعمل Update مباشرة وال EF لحالها بتحدد نوع الجدول:



1. كمان هي طريقة أخرى لل Update :

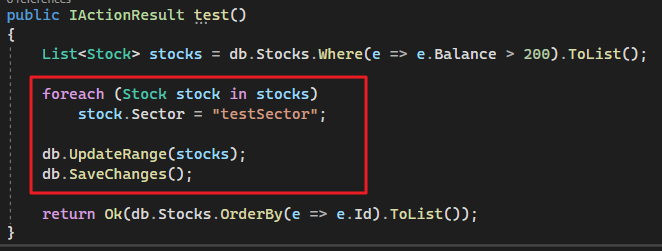


* ملاحظة اختار طريقة ال Update اللي بتريحك لكن انتبه في حال كان باعتلك object فيه فقط 5 قيم من أصل 10 فبالتالي ال 5 قيم الاخرى سوف يتم استبدالها ب Null (طبعا لو القيم Allow Null)

شو الحل لهي المشكلة؟



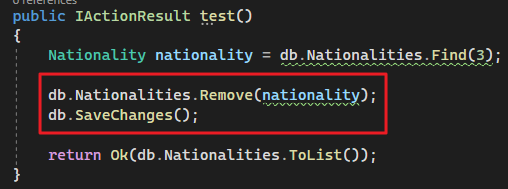
1. Update Range





3- Delete

1. الطريق العادية بجيب السطر من الداتابيز عن طريق رقمه وبعمله Delete كما موضح بالشكل التالي:



* طبعا فيني اعمل Remove Range بنفس الطريقة
* لازم تنتبه اذا أب واله اطفال شو لازم أعمل(راجع فقرة On Delete Set What???)

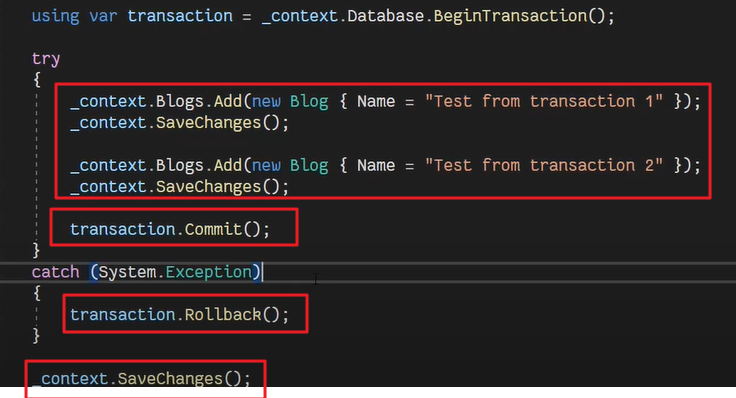
Cascade (Default) Drop Parent with its child’s

Restrict: throw Exception بيمنعك وبيعطيك ايرور

Set Null طبعا هون لازم الداتابيز تقبل قيمة Null

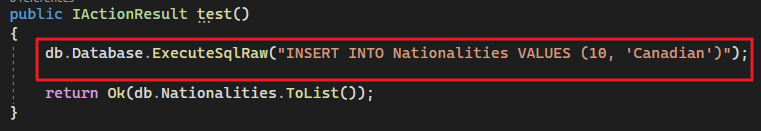
Transactions

* في أحيان بدي اعمل كذا قصة على كذا جدول وبدي ينفذ الكل أو يلغي الكل بهي الحلة استخدم ال Transactions
* حتى استخدمها بدها Try Catch ومع كل تعليمة بدي حط Save Changes وبالاخير بعمل commit متل الشكل التالي:



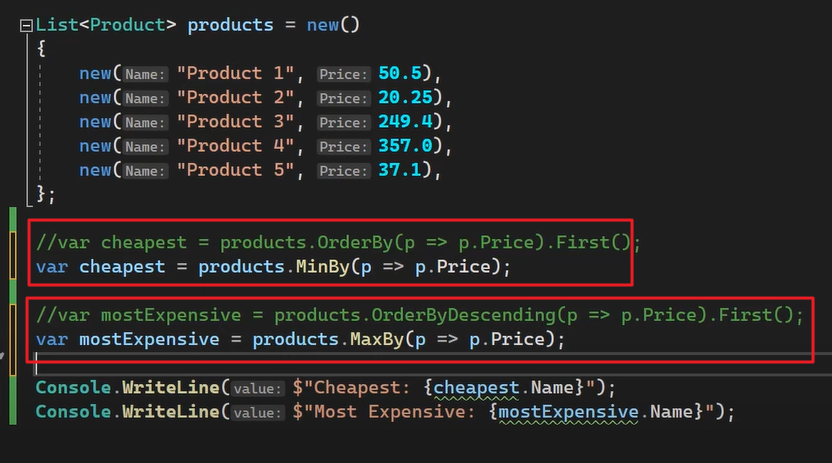
Save Data with Sql Statment and Stored Procedures

* سابقا تعلمنا شلون نجيب الداتا عن طريق SqlRow
* أما الان أنا بدي نفذ تعليمات SQL بدون ماترجع داتا
* الموضوع أبسط مماتتخيل كالتالي:



* طبعا بنفس الطريقة فيك تستدعي Function Or Procedure فقط بتحط اسمها ضمن الأقواس

MinBy MaxBy



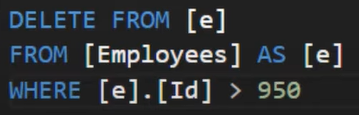
EF7 Updates - Bulk Delete/Bulk Update

* ميزة جديدة تم اضافتها بال EF7

1. Bulk Delete:



* اذا بتشوفها بال Profiler بتلاقي أنه ترجمها الى SQL بشكل طبيعي:



1. Bulk Update:

بدك تحدد العمود اللي بدك تعمله Update وشو القيمة اللي بدو ياخدها (كمان بيبعتها للداتابيز على شكل SQL)

