1/3/2023

Abdel-Rahman Gamal Ahmed

no campany

Design pattern

Data access pattern



Table of Contents

[1- Topic: 2](#_Toc123856462)

[2-simple app for test: 2](#_Toc123856463)

[4- Repository pattern: 6](#_Toc123856464)

[3.1-problem: 6](#_Toc123856465)

[3.2- start in solution of problem: 6](#_Toc123856466)

[3.3-implement Repository layer: 7](#_Toc123856467)

[3.3.1-Why still create context in ui layer ? 10](#_Toc123856468)

[3.3.2-why create context in ui layer : 10](#_Toc123856469)

[3.4 continue in implement for Repository : 10](#_Toc123856470)

[3.5-business layer or business role : 14](#_Toc123856471)

[3.6- summarization: 17](#_Toc123856472)

# 1- Topic:

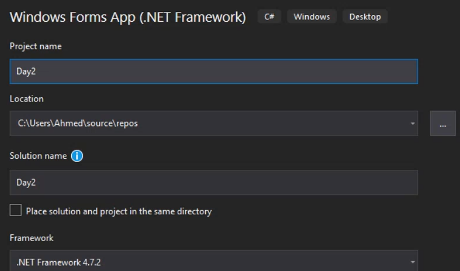
* هناخد النهاردة ال repository pattern ودة واحد من اهم ال pattern الى هنشفها وبنصنفه على انه data access pattern .
* لو انت هتشتغل backend من ضمن ال layer الى هتبنيها هى ال repository layer .
* وال المرة الجايه هناخد حاجة اسمها dependence injection principle
* والمرة الجايه برضو هناخد ال structure pattern كمان على السريع كدة .

# 2-simple app for test:

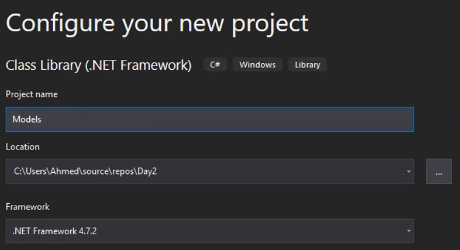
ف الاول كدة هنعمل على السريع مشروع بسيط وكنا علمناه قبل كدة لمجرد ان نطبق عليه مش اكتر .

* واحنا هنعمل اكتر من layer الى كنا بنعملهم زمان
* Business layer ودى الى هيكون فيها ال model بتاعتى وهيكون ايمها model
* وlayer هيكون هنعمل فيها map للmodel بتاعتنا ف ال database . يعنى هتكون layer فيها شغل ال EF
* UI layer .
* بعد الكام layer دول والى اصلا عملنها قبل كدةومفيش فيهم جديد هنعمل ال repository layer .

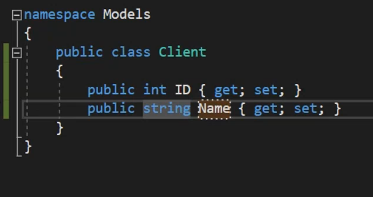
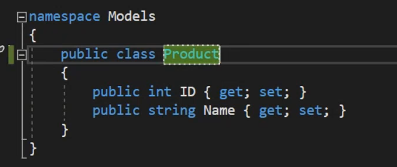
1. اول مشروع هنعمله هو بتاع ال UI : واسمه هيكون day2 وهنرج ليه بعد شويه



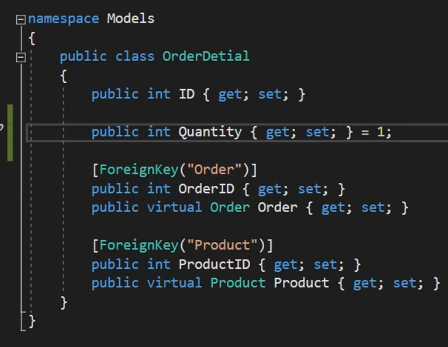
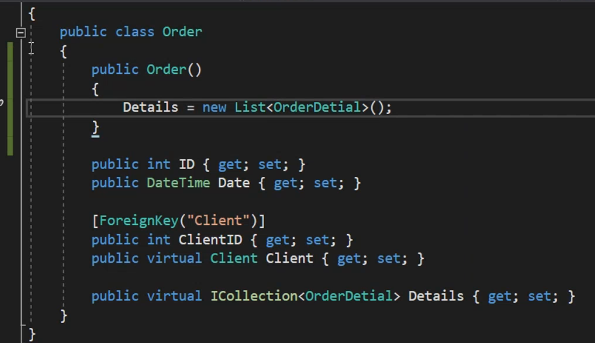
1. تانى حاجة هنعملها business layer : وطبعا دة من نوع class library . وهنعمل فيه order system صغير



* اول مودل هو ال product والتانى هو ال client .

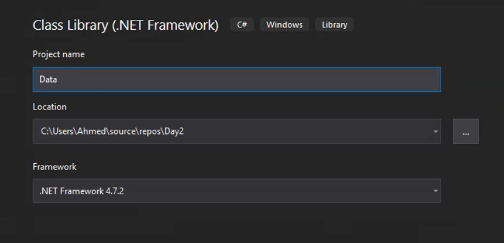


* تالت كلاس هو ال order وهيكون ف علاقه 1:m مع ال client وكمان علاقه m:m مع ال product فهنعمل bridge اسمهorderDetails ودة هيكون فيه كل سطر ف ال order عباره عن ايه او وصف كل سطر ف الorder .

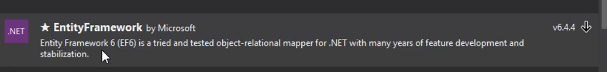


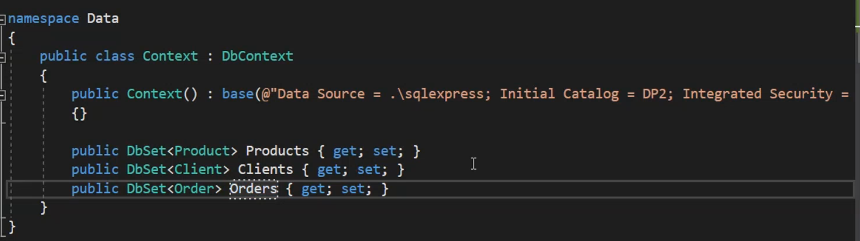
* مش محتاح اقلك ان الكلاس دى لمجرد الشرح .

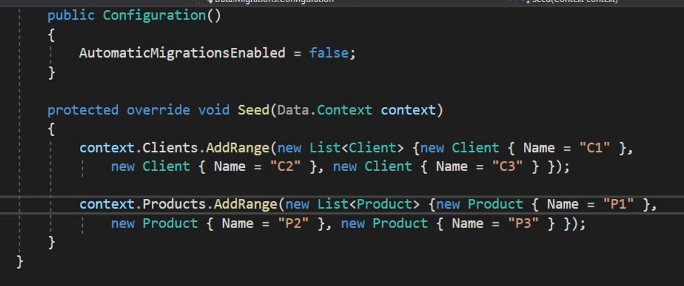
1. تالت layer الى هيكون فيها موضع الmap بين ال models والdatabase . وهو برضو class library :



1. وهنحتاج نحمل ال EF packages .

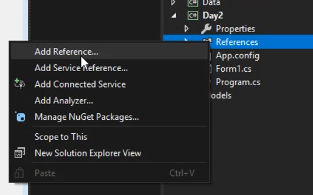


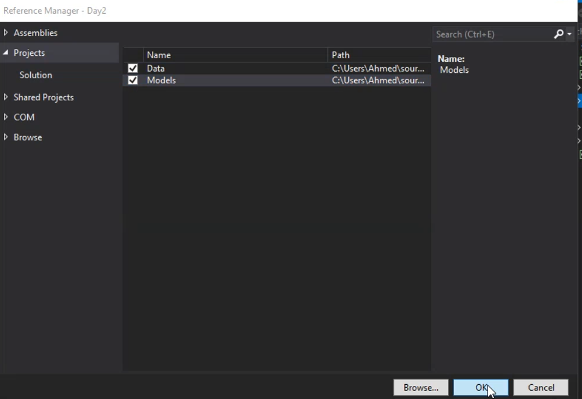
* ف دلوقتى هنعمل ال DB context class الى هيكون فيه ال connectionString وكمان ال class الى هعملها mapping
* دة ال connectionString وبنحطه ف كونستراكتور بتاع ال parent .
* ودول الكلاس بتوعنا ال هنعملهم map ف الداتا بيز
* وهنحط بس داتا ف seed method عشان واحنا بنعمل migration تتخزف ال DB . لو ناسى ال seed method هتكون ف migration class الى هيظهرك .

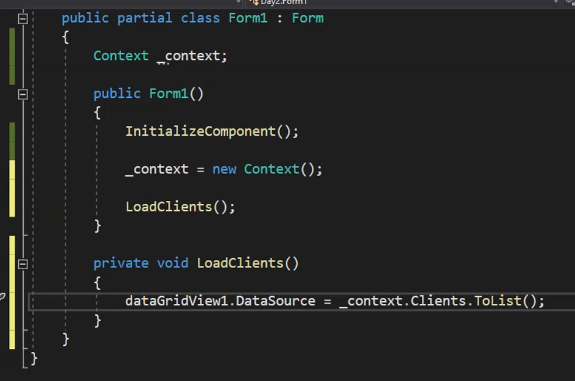


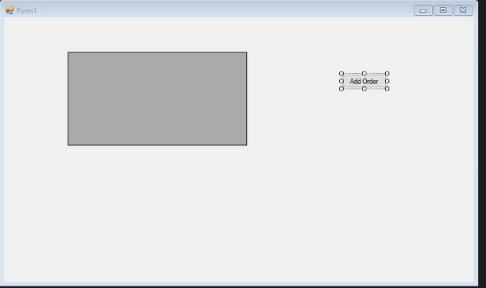
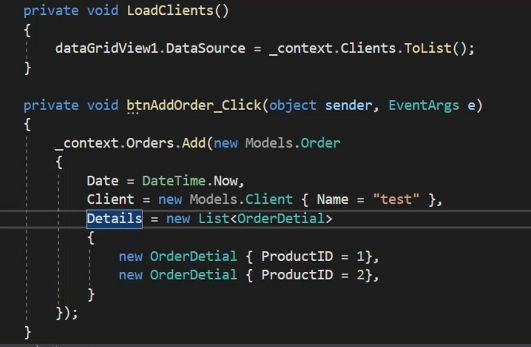
1. تانى حاجة هنعملها enable migration و add migration و update database .
2. كدة خلصنا الlayer بتاعه ال model وكمان ال layer بتاعه الmapping ف كدة جاهزين عشان نحط داتا ف الداتابيز .
3. هنرجع نعمل شويه شغل بسيط ف ال layer بتاعه ال UI الحجات الى هنعملها لمجرد التست الشغل ولفهم المشكله الى هتخلينا نعمل ال repository .

* وعشان نشتغل مع ال layer بتاعه الداتا هنعمل add reference ليها ف مشورع ال UI ال اسمه day2



* هحدد انا عاوز اعمل add reference لمين باضبط
* وهنعمل install للpackages بتاع EF طلام هنتعامل مع DB Context .
* بس كدة جاهزين نكتب كود جوة ال ui layer .

1. الكود الى هنعمله بسيط خالص : اول حاجة عملت grid هعرض فيها ال client . وطبعا عملت object form context .

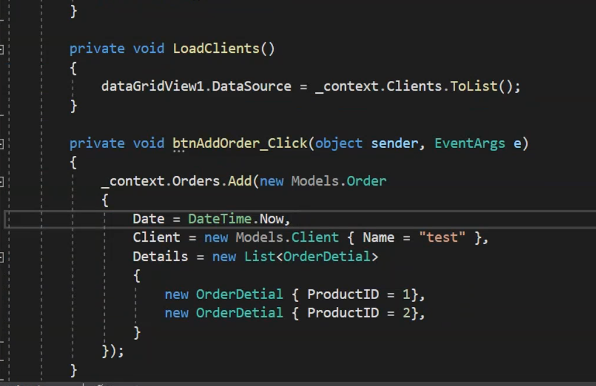
* ودة تانى كود بيخليك لما ادوس على زرار يضيف order والاسكرين الى على اليمين دى بتاعه ال ui .

# 4- Repository pattern:

الكود الى فات احنا اصلا عملناه قبل كدة وانا كتبته بس عشان هنكمل عليه ف يعتبر مفهوش ولا معلومه جديده:

تعاله بقا نتناقش ونشوف ليه هنزود ال access layer data :

* ايه المشكله ف الكود بتاع ال ui ودة اسكرين لجزء منه وقصدى بالمشكله يعنى ايه المبادئ بتاع ال solid الى متحققتش :



## 3.1-problem:

1. اول مشكله ممكن تلحظها ان ال ui بيستخدام ال database يعنى بيستخدام concrete object من ال context مباشرتا .

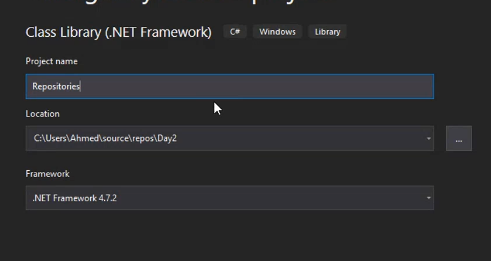
* يعنى ال ui layer كدة بقت tightly couple مع ال layer بتاعه ال data وال model layer .
* موضوع ال tightly couple مع ال model layer هناقشه ف ال mvc pattern .
* خلينا دلوقتى ف ال tightly couple بين data layer و الui layer. لو جيت وقلتلك حنا عاوزين نغير data provider بدال ماحنا شغلين sql server هنشتغل ado او mongoDB .او عاوز اتعامل مع اتنين data provider مش واحد .
* ف المشكله هنا ان خليت الlayer بتاع الui مرتبطه ب ال layer بتاعه ال data ودة هيعمل مشكله لما اغير الlayer دى باى data provider تانى .

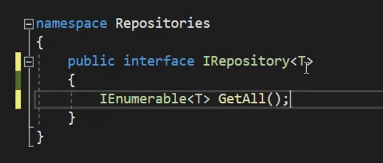
## 3.2- start in solution of problem:

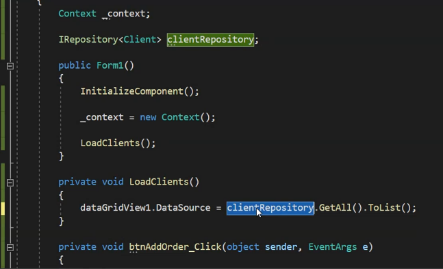
عوزين نفكر ف حل من ال pattern الى اخدنها او principle ؟

* لو فاكر لما اتكلمنا عن ال dependence inversion principle : ودة كان بيتكلم عن عندنا حجات low level وحجات high level والاتنين عوزين يتكلمه مع بعض ف كنا بنخلى ال high level معتمد على ال low level . والحل الى قلناه سعتها ان مفروض نخلى الاتنين يعتمدة على abstraction يعنى يكون ف layer بين الاتنين .
* الlayer الى هتكون بين ال low level و ال high level هى ال Repository layer .
* Repository layer هو عبايرة عن layer بتعمل wrapping للlayer بتاعه ال data . يعنى هتكون layer بين ال ui وبين ال data base .ودة عشان لو ال data provider اتغير ف اى وقت انا as ui متاثرش بحاجه عشان انا مش بتعامل مع ال data provider انا بتعامل مع ال Repository layer وهنشوف كذا فايدة تانى لل Repository.

## 3.3-implement Repository layer:

هنشوف ازاى هنعمل Repository layer . ومين هيتعامل معاه ف السكشن دة .

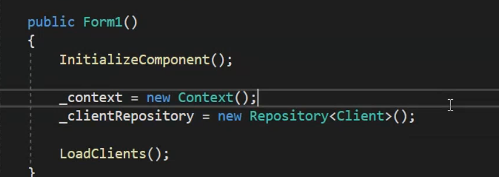
1. هعمل dll جديدة لل Repository .واسمها Repositories
2. Repository دة زى متفقنا احنا هنعمله abstraction وبعد كدة هنعمله هنعمل منه concrete class . عشان بعد كدة ال high level layer الى هى ال UI تستخدام ال concrete class .
3. وعلى فكرة مش شرط ف ال Repository ان نعمل abstract class وعبد كدة نعمل concrete class من abstract class. وممكن على طول نعمل concrete class بس زى مقلنا كتير قبل كدة الاحسن نعتمد على abstraction عشان نقد نعمل extend .
4. اول حاجة هعمله دلوقتى هعمل interface Repository وهبداء افكر ممكن ايه الى ممكن ال ui يحتاجه عشان اكتبه ف Repository دة وعشان ف الاول وفى الاخر انا عملت ال Repository عشان ال ui يستخدامه .
5. الui عاوز ال Repository عشان يوصل يعمل access للداتا عن طريق Repository بدال ال context ف هنبداء نعمل دة واحدة واحدة .
6. اول حاجة هنعملها ف ال Repository هنخليه generic عشان اتعامل مع اى DBset باحتى . وهعمل داله واحدة دلوقتى ال هى GetAll() . وتعاله نشوف هنستخدمها ازاى ف ال ui layer وبعد كدة نكمل الinterface .
7. ف ال ui او ال form دلوقتى مش هتتعامل مع ال context وهتتعامل مع ال repository فهنعمل reference من ال repoومش هنحوش ال context دلوقتى وهنعرف ليه كمان شويه وهنحوشها ازاى برضو كمان شويه .

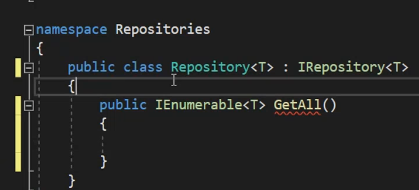
* دة ال repo الى عملناه واستخدمنا داله getall جوه ال loadClient .
* متنساش احنا لسه معملناس implement لداله getall
* والrepo الى عمناه من نوع client . ف كدة

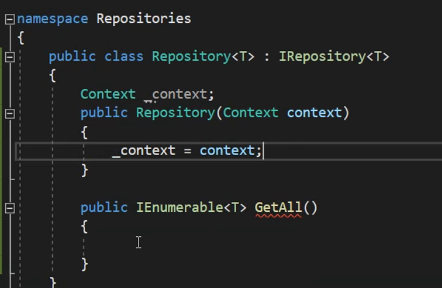
داله getAll مفروض ترجع clients .

1. هنعمل ال concrete class الى بيورث من ال repo عشان هو دة الى هيكون فيه الكود بتاه ال getall وباقى الدوال .

* بعد معمل ال concrete class هكون قادر اعمل object وابعته لل repo reference .
* وافتكر ان لحاد لدوقتى ممسحناش ال context وحالا تعرف ليه .

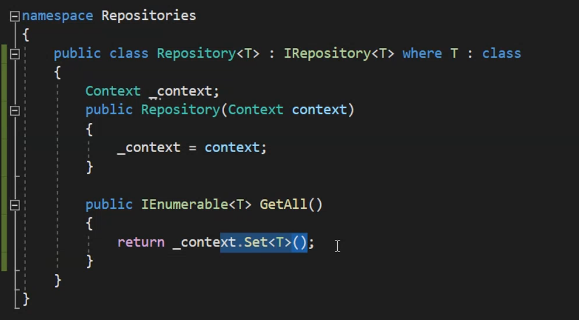
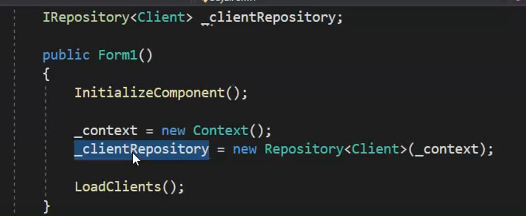


* احنا قلنا ان ال Repository دى هتعمل wrapping لل data layer يعنى مفروض يهكون جوها object من الcontext ف مفروض ال object دة هتجيبه من ايه ؟
* وخلى بالك ان عملت ال concrete class كمان generic class .
* نرجع لموضع من اين هتجيب ال context ؟ انا as Repository هقول الى هيعمل منى object يبعتلى ال object .

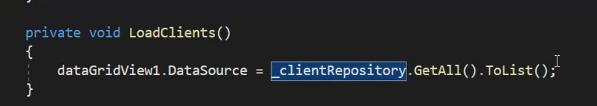


* ف حاجة مهم خد بالك منها انا دلوقتى ال class بتاعى هو generic طيب انا دقوتى انا هعرف ازا ايه ال object الى هتعامل معاه من الcontext يعنى هتعامل مع client ف هعمل context.client ولا هتعامل مع product ف هعمل context.product ؟

الحل اخدناه زمان لما قلنا عاوز ارجع داتا بتاعه DBset مش موجودة جوه الcontext .

* كنا استخدمنا حاجة اسمها Set<> ودى هتخلينا نحمل ال EF ف ال layer دى وكمان نحدد ان ال T دى لازم تبقا من reference type. الset دى بنديها data type وهى بترجعلىDBSet من ال type الى اخته . ودع بقى شكل الكلاس ونقدر نستخدام ال getAll جوة ال UI layer .
* وف ال ui كل الى هعمله ان هبعت object من context لل conceate class الى هعمله بروض وكدة داله ال getall اشتغلت الف مبروك .
* وطبعا ال concrete repo هو generic ف عشان كدة كتبت client عشان احدد هو هيشتغل على انهى type .
* عرفت انا انا لسه عامل createللcontext جوة Ui ليه ؟

الاجابه ف السكشن الجاى لو لسه معرفتش

*  وكدة انا ستخدامت ال get all حوة ال gui .

### 3.3.1-Why still create context in ui layer ?

* عرفت انا انا لسه عامل createللcontext جوة Ui ليه ؟ اكيد عشان هبعتها repo contactor .واخر اسكين ف سكشن 3.3 موضحه الموضع .

### 3.3.2-why create context in ui layer :

ممكن تقلى مفروض اعمل create للcontext جوة ال repo layer عشان احنا عوزين نخلى ال ui layer مش معتمدة علىcontext . ف هقلك كدة انت بقا يحلو لو عملت اكتر من object من ال repo انا كده هيكون عندى اكتر من object من ال context . واكيد احتاج اعمل اكتر من object من repo اول سبب ومش الوحيد ان ال repo class هو generic ف هحتاج اعمل object منه لل client وواحد تانى لل product وهكذا وطبعا هحتاج اعمل object ف اماكن تانيه .

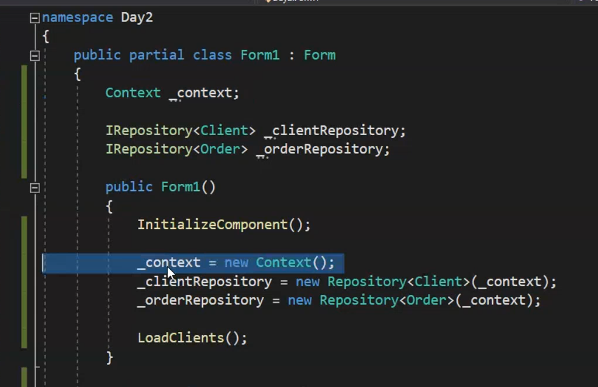
* طيب لو انا عاوز اعمل create لل context جوة ال repo وفى نفس الوقت مش عاوز يكون فيه اكتر من object من ال context هنعمل ايه ؟ بيسطه هنستخدام ال design pattern بتاع ال singleton pattern .
* ليه مستخدمناش singleton عشان احنا هنستخدام الشكل دة ف حاجة تانى هنخدها قدام . يعنى هستتخدام موضوع ان ببعض ال object ف ال contractor ف dependence injection قدام ف تابع معايه .

## 3.4 continue in implement for Repository :

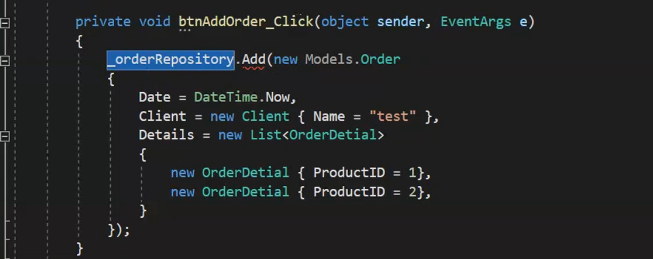
الى عملناه ف سكشن 3.3 ان شفنا صورة كامله لازاى هعملimplement لل Repository وازاى هنخلى ال Ui يكلم الdata layer من خلال ال Repository .

* بس ف حجات لسه عاوزين نحسنها زى ان لسه ال object من context موجود ف ال ui layer .
* وهنكمل شويه دوال مفروض ال ui layer تستخدمها ونحل المشاكل الى هتظهر فيها ف تابع معايه ي باشا .

1. هعمل object من ال Repository بس المرة دى هخليه يشتغل على order كلاس مش client .

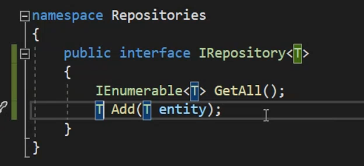
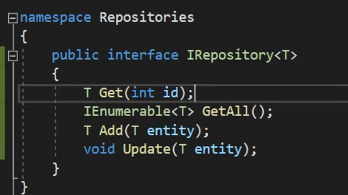


1. انا هعمل object من ال Repository بس المرة دى هخليه يشتغل على order ودة عشان احوش الاعتماد على ال context من الداله الى بضيف order وخليها تعتمد على Repository .

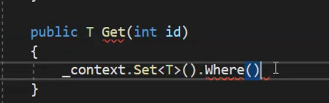


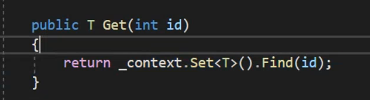
1. فاضل كدة اعمل implement لداله الadd جوة كلاس repo .

* وطبعا لازم نعمل method add جوه ال interface repo : وهتلاحظ انها بترجع object بعد مضيفه ودة عشان ال object الى راجع هيكون واخد identity id .
* وتعاله بالمرة نعمل باقى الدوال

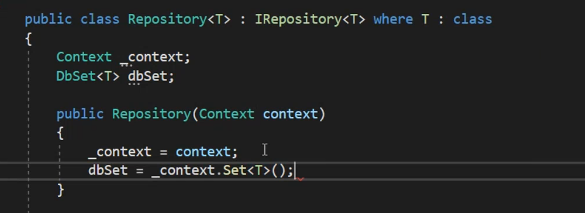


* كدة عمنا الدوال جوة ال interface فاضل نعملها جوه الكلاس repo .
* اول داله هى get وهنا هتلاحظ ان مينفعش اعمل where عشان انا اصلا مش عارف هشتغل على انهى type عشان الكلاس generic لو فاكر ف ممكن اشتغل على type مفهوش property اسمها id اصلا . ف الحله دة هنستخدام حبيبتنا ال find ودى بنديها ال primary key وهى بتعمل search على الentity

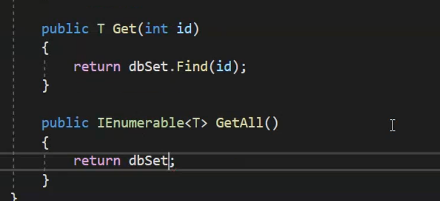


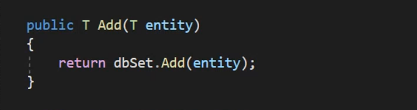


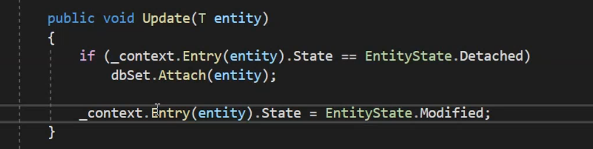
* وقبل مروح لfunction تانى انا بس هعمل اختصار بدال كل مرة افضل اكتب \_context.set <T> () هكتب \_dbset



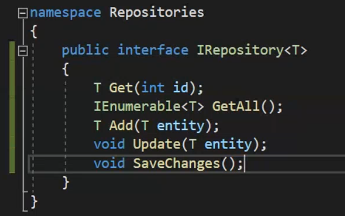
* ودة هيكون شكل الكود بعد مضفت . \_dbset ونرجع تكمل بقاى الدوال يغالى .

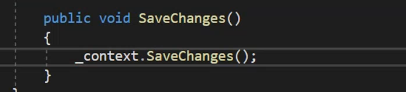


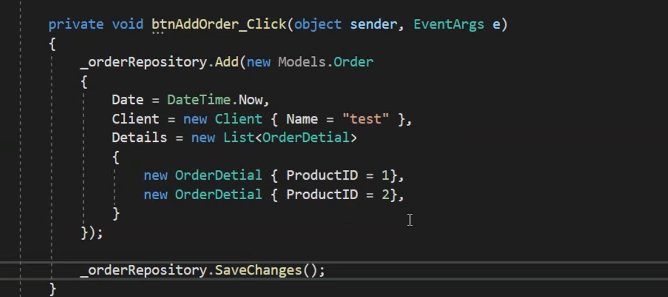
* تالت فنكشن
* رابع فانكشن وفى معلومه كويسه هنا : احنا كنا متفقين زمان ان مفيش داله اسمها update جوة ال context وان ال update بيحصل automatic لما بعم save change هو بيعمل detect وupdate لوحدة ف ايه لزمه ان اعمل function اسمها update طلام هو هيعمل update لوحده ؟ عشان ال automatic update بيحصل لما يكون ال object الى اتبعت معمله track من ال context غير كدة مش هيحصل ال automatic update ف احنا عملين داله ال update عشان نعمل بايدنا track للobject . ولو انت ناس راجع الEF :
* ف طريقه عمل track هى : ف الاول لازم اتشوف هو معموله track ولا لا ولو كان لا هخليك track وتانى حاجة هعلم على ال object ان هو change
* يعنى اول حاجة انا بقول لل context خلى بالك ال object دة تبعك تانى حاجة انا بقول ان كمان ال object الى تبعك دة هوchange ف بعد الحاجتين دول هو هيفهم ويعمل automatic update .
* الكود بتاع ال if دة الى بيخلى الcontext تفهم ان object دة تبعها والكود الى تحته دة بيعمله انه هو modify



* ف حلات تانى بنحتاج نعمل check على موضوع الstate زى الdelete وكمان داله ال update الى عملنها اى كلام المفروض ان اكتب شويه كود يخلينى اعرف انهى prosperity اتغيرت واعلم على الstate بتاعه ال prosperity عشان لما يحصل ال update يحصل على الحجات الى اتعدلت او اتغير فيها حاجة . بس مش هنعمل الكلام دة دلوقتى عشان الوقت
* ف داله بس نسيتها تعاله نعملها وهى ال saveChanges





* وكدة خلصنا كل الدوال يغالى ولو فاكر ☹ احنا كنا بنعمل كل دة عشان نشغل داله بتاعه اضافهorder الى بتشتغل لما ادوس على زرار add order الى ف ال form .

1. خلى بالك ان كدة ال repo layer متعرفش اصلا ال ui موجود وال repo layer بتكلم ال database وال ui هو الى بيكلم ال repo layer وال repo layer يعرف بس ان ال context موجود . دى حجات خليها ف دماغك
2. الجمال والحلاوه لسه جاين وكل الى عملناه ان احنا عملنا ال repository layer وخلينا كل تفاصيل ال context مستخبيه وره ال repo layer .
3. اطيب هى ال prosperity layer بديله للbusiness layer الى عملنها قبل كدة لحاد دلوقتى لا .عشان لحاد دقوتى احنا يدوب عملنا wrapper بينا ال ui وال data provider عشان لو الداتا بيز اتغيرت منغيرش ف الكود بتاعنا .
4. **واحنا لسه لحاد دقوتى معملناش ال business layer يعتبر مش عشان ان ال context لسه موجودة معانا ف UI layer لا الفكرة ان احنا لحاد دوقتى محطناش اى business layer يعنى محطناش اى logic ليه زى مثلا همنع اى اتنين client يكونه بنفس رقم التلفون او غيرة وهنعمل ال business layer بعد شويه .**
5. **ال repo layer مش موجودة عشان تبقا business layer هى موجودة ف الاصل عشان تبقا wrapper للdata provider**

## 3.5-business layer or business role :

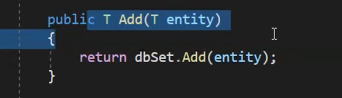
**لو انا مش شغال repository**  : وعاوز اعمل business logic زى ان مينفعش اضيف اتنين product بنفس ال code ومينفعش اضيف اتنين مثلا client بنفس رقم البطاقه ف الكود الى هيعمل كدة هيتكتب فين ؟وافتكر ان بقلك لو **مش** شغال ب repo pattern

اكيد هنضيف ف الجزء بتاع ال ui كدة انا وقعت ف كام غلطه والى هى

1. مفيش single responsibility عشان خليت ال ui هو الى يعمل check على ال business logic ودة هيسبب ان لو بعد شويه بقى فيه شاشه تانيه بضيف فيها client او product ف كدة هكرر الكود تانى ف الشاشه الجديدة

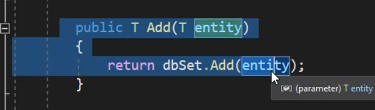
**ف حاله ان شغال repository :** ف اكيد الكلام الى فات هيتغير

* ممكن تقلى هحط الlogic جوة كلاس ال repository ف داله بتاه add هقلك مينفعش عشان الكلاس دة هو generic يعنى داله add دة هنادم عليها ف حاله ان بضيف client او بضيف product او حتى order ف مينفعش اعمل logic بتاعى هنا وخلينا نقول ال logic الى عوزين نعمله تانى ونحدده اكتر احنا عوزين نخلى مينفعش اضيف اتنين client ليهم نفس الاسم .

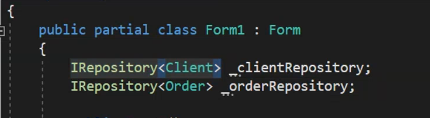


* ف عشان نطبق ال logic الى قوله فوق هحتاج اعمل

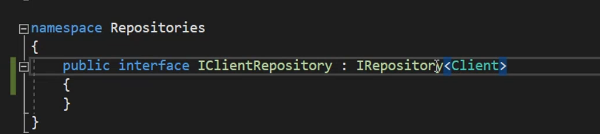
1. هسيب ال generic repo الى موجود عندى ومش همسحه عشان انا محتاج الدوال الى فيه عشان ممكن يكون ف model مش هنطبق عليه business logic
2. كدة ف حاله ال client انا هحتاج اعمل extent لل generic repo يعنى انا محتاج ف حاله ال client ان اعمل check على business logic معين قبل منادم داله ال add



* ف كدة من الى قلته ف النقطه رقم 2 هحتاج اعمل repository جديدة خاص ب ال client فقط عشان ف logic معين عاوز اطمن ان هو موجود . وكدة ف ال ui مش هعمل object form repo بيشتغل على client لا اناهعمل object و reference من ال layer الجديدة الى هعملها .



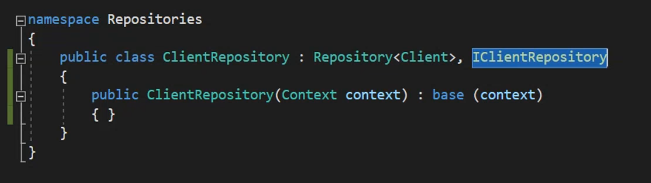
* تعاله نعمل الى قلته اول حاجة هعمل interface جديد وخليه يعمل implement لل generic interface عشان انا عاوز نفس الدوال الى فيه وكمان لاحظ ان حددت ال generic interface دة هيكون شغال على client . ف كدة اى حد هيعمل implement للinterface الى اسمه IclientRepo هيعمل كل الدوال الى ف IRepository .



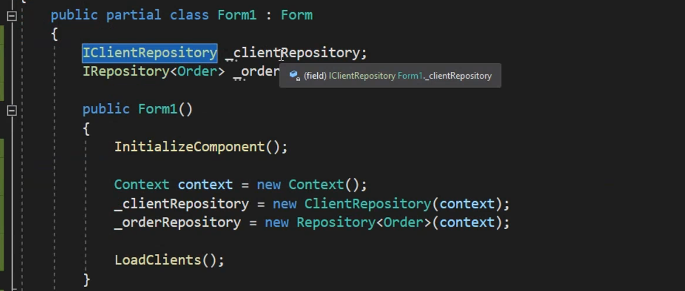
* خلصنا كدة ال abstract class تعاله نعمل ال concrete class الى هنحط فيه ال logic بتاعنا واسمه clientRepo

وهخلى ال clientRepo يورث من ال repo ويعمل implemt للiClientRepo وانا هنا خليته يورث من ال repo القديم الى كنا عملينه عشان مفضلش اكتب كل الدوال انا هعمل بس override لداله واحدة من الدوال القديمه الى منت عملها فكلاس repo .

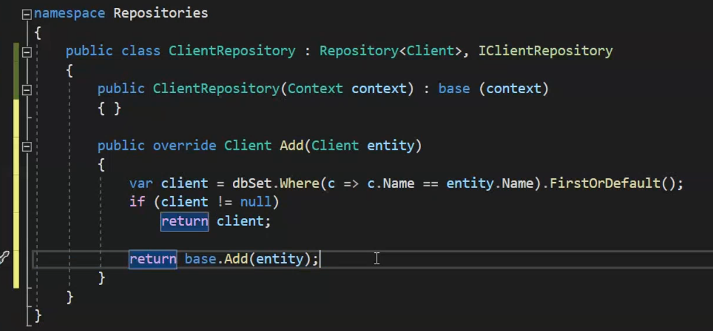
وطبعا هخلى ال contractor بتاع الكلاس الجديد ياخد object from context عشان يبعته لل parent class عشان زى مقلت اكتر من مرة ف الكلاس ال child بيتعملك object من كلاسparent ف لازم تشوف لو كلاس ال parent هياخد object تدهوله بستخدام كلمه parent .



* وكدة خلاص انا هحوش الreference بتاع ال repo القديم وحط ال reference بتاع ال ClientRepo الجديد الى هيكون فيه ال logic بتاعى . ف دة هيكون شكل كود الui .



* ف زى مقلنا انا هعمل override لداله add ف كدة هحتاج اخلى داله add ف الكلاس القديم بتاعى تكون virtual
* ف كدة الoverride لدله add وكدة الف الف مبروك خلصنا .



## 3.6- summarization:

احنا كدة خلصنا جزء كبير من عمل ال repository بس مش كل حاجة .

* احنا كدة طبقنا جزء كبير من ال repository pattern وكمان عملنا business logic .
* هنروح نكمل شويه ف ال dependence injection وبعد كدة هنرجع لل repository pattern نضيف كام معلومه هتفدنا جامد .