UNIVERSIDAD POLITECNICA DE VALENCIA

Pràctiques

Butlletí Pràctica 4

Persistència

Enginyeria del Programari

ETS Enginyeria Informàtica
DSIC – UPV

Curs 2017-2018

1. Objectiu

IMPORTANT: En primer lloc llegir tot el document abans de començar a desenvolupar

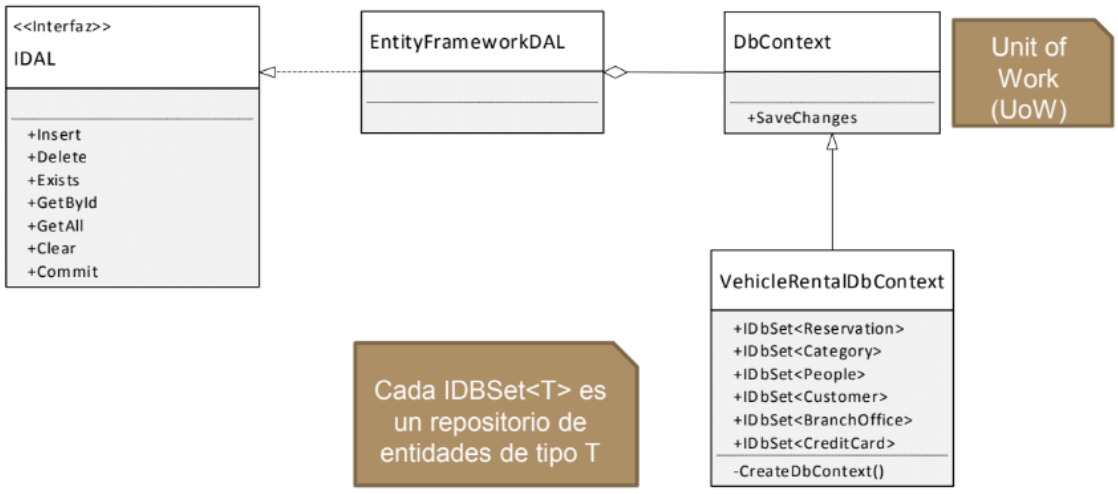
L'objectiu d'aquesta sessió és el desenvolupament d'una capa d'accés a les dades que ofereix la lògica dels serveis aplicació necessaris per a recuperar, modificar, suprimir o afegir objectes a la capa de persistència, sense que la capa lògica sàpiga quin mecanisme particular de persistència s'està utilitzant. Per a facilitar el treball, s' utilitzarà Entity Framework (EF), un marc de persistència desenvolupat per Microsoft per a la plataforma. NET, la qual offereix mapejat objecte-relacional i un disseny basat en el patró *Repositori + Unitat de Treball*.

Prèviament a la realització de la pràctica, els estudiants hauria d'haver hi repassat el tema 6 dedicat a la persistència" i els dos seminaris relacionats amd "Entity Framework " i "DAL". D'altra banda, el cas d'estudi de referència "VehicleRental", que es pot descarregar des de Poliformat, és una aplicació sencera que fa servir l'arquitectura explicada a classe i hauria de servir com a exemple a l'alumne en el desenvolupament del cas d'estudi que ens ocupa.

L'objectiu d'aquesta sessió és desenvolupar les classes que formen la capa de persistència del cas d'estudi GestDep d'una manera anàloga a l'exemple de referència i comprovar que la capa de persistència està funcionant correctament.

2. Disseny de la capa d'accés a dades

La gestió transparent de l'accés a dades (agnòstic pel que fa al mecanisme de persistència) la aconseguirem mitjançant una **interfície** entre la capa lògica i la capa de persistència real. Aquesta interfície, anomenada **`IDAL`** (DAL = Data Access Layer), s'encarrega d'abstraure els serveis de la capa d'accés a dades del mecanisme de persistència particular que s'haja emprat (SQL, XML, ets).



Ilustració 1. Arquitectura de la capa d'accés a dades

Per al projecte de laboratori treballarem amb Entity Framework com a mecanisme de persistència, i per tant necessitem una implementació de IDAL específica per a EF, que en el projecte d'exemple es denomina EntityFrameworkDAL.

Entity Framework és un sistema de mapeig objecte-relacional que permetrà a la nostra aplicació treballar directament amb Entitats que són objectes de la lògica de negoci, i no objectes específics per a transferència

de dades. EF s'encarregarà, automàticament, de gestionar de forma transparent la persistència dels dits objectes en una base de dades relacional com SQL. L'accés a dades usant Entity Framework es basa en el patró *Repositori + Unitat de Treball*, tal com s'explica en el tema 6 dedicat a la persistència. La unitat de treball s'implementa estenent la classe `DbContext`. En el projecte d'exemple si observa la classe `VehicleRentalDbContext`, observarà les següents característiques:

- Hereta de `DbContext`
- Té propietats de tipus `IDbSet<Tipus>` on "`Tipus`" és cadascuna de les classes del model que han de ser persistents a la base de dades (cada IDbSet constituïx un repositori con los métodos típicos para acceder, añadir o eliminar objetos). Por ejemplo, en `VehicleRentalDbContext` tenemos los siguientes

```
public IDbSet<BranchOffice> BranchOffices { get; set; }
public IDbSet<Reservation> Reservations { get; set; }
public IDbSet<Category> Categories { get; set; }
public IDbSet<Person> People { get; set; }
public IDbSet<Customer> Customers { get; set; }
public IDbSet<CreditCard> CreditCards { get; set; }
```

- Té un constructor que proporciona al constructor base el nom de la cadena de connexió a la base de dades (`VehicleRentalDBConnection`), la qual està definida en l'arxiu `App.config`. Dins el constructor s'inclouen també alguns paràmetres de configuració

La Interfície `IDAL` és un adaptador que combina la funcionalitat dels repositoris (`Insert, Delete, GetXXX, Exists`) juntament amb la funcionalitat de la unitat de treball (`Commit`) . En efecte, si consultem el projecte d'exemple `VehicleRental`, veurem que `IDAL` té els següents mètodes.

```
void Insert<T>(T entity) where T : class;
void Delete<T>(T entity) where T : class;
IEnumerable<T> GetAll<T>() where T : class;
T GetById<T>(IComparable id) where T : class;
bool Exists<T>(IComparable id) where T : class;
void Clear<T>() where T : class;
void Commit();
```

Es tracta dels mètodes habituals ja comentats, més un mètode addicional (`Clear`) per a esborrar la base de dades. Observi que s'ha utilitzat **genericitat**, reduint notablement la quantitat de codi a implementar. És a dir, en compte d'implementar les següents operacions individuals.

```
void InsertOffice(BranchOffice o);
void InsertReservation(Reservation r);
void InsertCategory(Category c);
...
```

el mecanisme de genericitat ens permet definir un únic mètode

```
void Insert<T>(T entity) where T : class;
```

que s'instanciarà en execució en funció del tipus `T` (que serà una classe) corresponent.

Es important recordar que després de completar una transacció (una o més operacions que impliquen un cambi en les dades), s'ha d'invocar el mètode `Commit` per a persistir tots els cambis. Encara que no s'ofereix un mètode específic per actualitzar un objecte, si hem modificat objectes internament, igualment haurem d'executar el mètode `Commit`..

Tasca: L'equip haurà d'implementar la classe que hereta de `DbContext`, anàlogament a `VehicleRentalDbContext`, per al cas d'estudi que ens ocupa, definint una propietat IDbSet per a cadascuna de les classes persistents del model. Pot cridar a la classe `GestDepDbContext`, i com a espai de noms hauria de tindre `GestDepLib.Persistence`. Haurà d'afegir al projecte el codi de IDAL i `EntityFrameworkDAL`, que al

ser totalment genèrics es poden reutilitzar tal qual del projecte d'exemple. L'única cosa que caldrà fer és canviar l'espai de noms a `GestDepLib.Persistence`.

Les classes a implementar hauran d'estar ubicades en la carpeta `Persistence` del seu projecte de biblioteca de classes `GestDepLib`.

Observe amb l'explorador de solucions l'estructura del cas de referència si té alguna dubte

3. Configuración inicial de la solución (Team Leader)

Per a treballar amb EF és necessari afegir al seu projecte el paquet necessari. Per a això, un membre de l'equip farà el següent: en Visual Studio vaja a `Herramientas > Administrador de paquetes NuGet > Administrar paquetes NuGet para la solución` i en el quadro de text `Examinar` escriba Entity Framework. Apareixerà en primer lloc eixe paquet (veure Figura 1) i haurà d'afegir-ho al projecte de biblioteca de la seua solució (projecte que va crear en la pràctica 3 i que conté les subcarpetes BusinessLogic i Persistence) . Comprove en l'Explorador de Solucions que en les Referències del seu projecte s'inclou Entity Framework. Una vegada afegit este paquet recomanen protegir la solució.

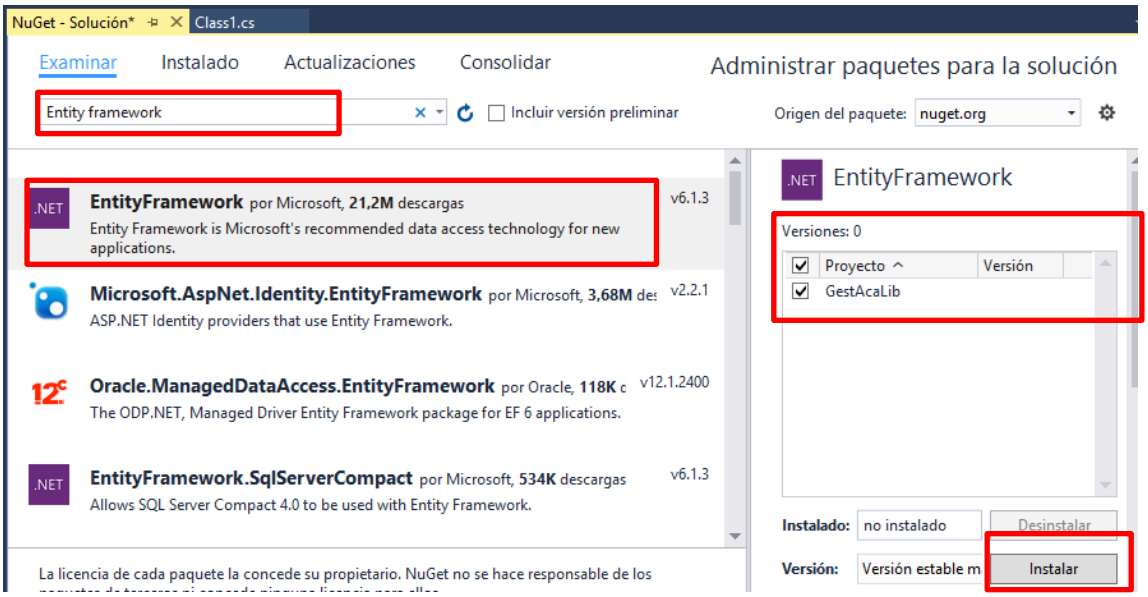


Figura 1. Afegint el paquet Entity Framework al projecte de biblioteca de classes

4. Prova de la capa de persistència

Un cop finalitzat el desenvolupament de la capa de persistència protegisca la seua solució amb un comentari com “Capa Persistència Finalitzada (versió beta)”. És el moment de provar la capa de persistència. Per a això podria crear-se un projecte de consola tal com es va veure en la primera sessió de pràctiques, i en eixe projecte caldria crear una instància de la classe `EntityFrameworkDAL` passant-li una instància de `GestDepDbContext`. Al crear la instància de `GestDepDbContext` per primera vegada es crearia la base de dades, i només quedaria crear alguns objectes de la lògica de negoci i persistir-los usant els servicis oferits pel DAL. Per a crear eixe projecte de consola hauria de complir els requisits següents:

- Incloure una referència a la seua biblioteca de classes (GestDepLib) . Per a això, en el seu projecte d'aplicació de consola: Botó dret del ratolí sobre `Referències > Agregar referència` ... i seleccionar dins de Projectes el seu projecte de biblioteca de classes.
- Incloure el paquet Entity Framework emprant el gestor de paquets NuGet
- Afegir una cadena de connexió a base de dades dins l’arxiu de configuració App.config. Si fem servir SQLServer com a motor de base de dades i anomenem “GestDepDB” a la base de dades a crear, la cadena de connexió quedaria com seguix

```
    <connectionStrings>
        <clear />
        <add name="GestDepDBConnection"
connectionString="Server=(localdb)\mssqllocaldb;Database=GestDepDB;Trusted_Connection=True;
MultipleActiveResultSets=true" providerName="System.Data.SqlClient" />
      </connectionStrings>
    </configuration>
```

Tot i això, per a facilitar-li el treball i assegurar que la base de dades es crega correctament, s'adjunta un projecte de consola amb algunes proves ja implementades. Així mateix, s'indiquen les dades a incloure en la base de dades perquè les proves resulten satisfactòries. Encara que estes proves es podrien realitzar millor per mitjà d'un motor de proves unitàries com MSTest o NUnit, de moment ho anem a provar amb una aplicació de consola (les proves seran tractades amb posterioritat) . Afegisca aquest projecte (GestDepDBTest) a la solució i òbriga l'arxiu `DBTest.cs`. Observarà un mètode buit anomenat `populateDB` que caldrà omplir amb el codi necessari per a crear els objectes de negoci i persistir-los en la base de dades. Si establiu este projecte com a projecte d'Inici podreu provar si funciona correctament.

```
private static void populateDB(IDAL dal)
        {
            // Remove all data from DB


            // Populate the database with the data described in lab 4 bulletin


        }
```

Dades d'example

Nota: els Ids numèrics (tots menys els referits a Person i les seues classes derivades) han de ser autogenerats, per la qual cosa eixos Ids poden canviar; així és que no donem els valors concrets d'eixos Ids

Pools

OpeningHour	ClosingHour	ZipCode	DiscountRetired	DiscountLocal	FreeSwimPrice
20/10/2017 8:00	20/10/2017 21:00	46122	20	15	3

Lanes

La única piscina creada debe tener 6 calles, numeradas de 1 a 6

People

Id	Address	Name	ZipCode	IBAN	BirthDate	Retired	Ssn	Discriminator
1234567890	Ona Carbonell's address	Ona Carbonell	46002	ES8912341212345678 90	05/06/1990	False	NULL	User
2345678901	Gemma Mengual's address	Gemma Mengual	46002	ES8912341212345678 90	12/04/1977	False	NULL	User
3456789012	Mireia Belmonte's address	Mireia Belmonte	46003	ES8912341212345678 90	10/11/1990	False	NULL	User
4567890123	Rigoberto's address	Rigoberto	46122	ES8912341212345678 90	28/02/1995	False	NULL	User
5678901234	Lázaro's address	Lázaro	46122	ES8912341212345678 90	01/01/1972	True	NULL	User
X-00000001	Michael Phelps'address	Michael Phelps	46001	ES8912341212345678 90	NULL	NULL	SSN01010101	Monitor

Courses

Start Date	Finish Date	StartHour	Duration	Minim. Enrollm	Max. Enrollm	Cancelled	Price	Description	Course Days	Monitor_Id

06/11/2017	29/06/2018	9:30	0:45:00	6	20	False	100	Learning with M. Phelps	L,X,V	X-00000001
07/11/2017	29/06/2018	19:00	1:00:00	8	16	True	75	Swimming for Dummys	M,J	NULL

Nota: El primer curs ocuparà els carrers 1 i 2, i el segon curs el carrer 3

Enrollments

EnrollmentDate	CancellationDate	ReturnedFirstCuotaIfCancelledCourse	Course_Id	User_Id
16/08/2017	NULL	NULL	1	1234567890
26/07/2017	NULL	NULL	1	2345678901
28/08/2017	NULL	NULL	1	3456789012
28/08/2017	20/10/2017	NULL	2	4567890123
04/09/2017	20/10/2017	NULL	2	5678901234

Nota: Indicaquem el valor de la clau aliena `Course_Id` perquè es veja que els tres primers Enrollment són en el primer curs, i els 2 últims en el segon curs, independentment dels seus Ids reals (al ser autogenerats, després de diverses execucions ja no coincidiran amb estos)

Payments

Description	Quantity	Date	Enrollment_Id
Free Swim	3	10/08/2017	NULL
Free Swim	3	20/08/2017	NULL
Free Swim	3	20/08/2017	NULL
First monthly quota - Learning with M. Phelps	100	16/08/2017	1
First monthly quota - Learning with M. Phelps	100	26/08/2017	2
First monthly quota - Learning with M. Phelps	100	28/08/2017	3
First monthly quota - Swimming for Dummys	75	28/08/2017	4
First monthly quota - Swimming for Dummys	75	04/09/2017	5

Nota: Indiquem el valor de la clau aliena `Course_Id` perquè es veja que els tres primers Payment no estan associats a cap Enrollment (la descripció indica “Free Swim”) , i els 3 últims estan associats al primer Enrollment, el d'Ona Carbonell, encara que eixe Id podria canviar en successives execucions donat que es autogenerat.

Programa de prova

Si has creat els objectes indicats i la capa de persistència és correcta, l'execució del projecte de consola `GestDepDBTest` hauria de produir la següent eixida (l'orde en què apareixen els cursos podria canviar)

```
==================================
         Course details
==================================
StartDate: 07/11/2017 0:00:00
FinishDate: 29/06/2018 0:00:00
Days : Tuesday, Thursday
Price: 75
Lanes assigned:
 Lane 3

Users enrolled in course Swimming for Dummys, with no monitor yet
 Rigoberto (4567890123) enrolled on 28/08/2017 0:00:00
 Lázaro (5678901234) enrolled on 04/09/2017 0:00:00

==================================
         Course details
==================================
StartDate: 06/11/2017 0:00:00
FinishDate: 29/06/2018 0:00:00
Days : Monday, Wednesday, Friday
Price: 100
Lanes assigned:
 Lane 1
 Lane 2
```

```
Users enrolled in course Learning with M. Phelps, with monitor Michael Phelps (X-00000001)
 Ona Carbonell (1234567890) enrolled on 16/08/2017 0:00:00
 Gemma Mengual (2345678901) enrolled on 26/07/2017 0:00:00
 Mireia Belmonte (3456789012) enrolled on 28/08/2017 0:00:00

Payments:
 10/08/2017 0:00:00 -> Free Swim: 3
 20/08/2017 0:00:00 -> Free Swim: 3
 20/08/2017 0:00:00 -> Free Swim: 3
 28/08/2017 0:00:00 -> First monthly quota - Swimming for Dummys: 75
 04/09/2017 0:00:00 -> First monthly quota - Swimming for Dummys: 75
 16/08/2017 0:00:00 -> First monthly quota - Learning with M. Phelps: 100
 26/08/2017 0:00:00 -> First monthly quota - Learning with M. Phelps: 100
 28/08/2017 0:00:00 -> First monthly quota - Learning with M. Phelps: 100
```

5. Herramientas de Inspección de la BD

Desde el propio entorno de desarrollo en Visual Studio es posible conectarse a cualquier base de datos para ver sus tablas y su contenido. Para conectarse a una base de datos existente local (creada como un fichero local) tendrá que realizar los siguientes pasos: `Herramientas > Conectar con la Base de Datos` y seleccionar como origen de datos "Archivo de base de datos de Microsoft SQL Server (SqlClient)" y como archivo de datos seleccionar el fichero con extensión ".mdf" creado. Dada la configuración realizada en el fichero App.config, el fichero de base de datos ".mdf" se crea en `C:\Users\nombreUsuario\GestDepDB.mdf`.

Una vez creada la conexión es posible explorar las tablas de la base de datos en el explorador de servidores que aparecerá en el entorno de desarrollo. Haciendo doble-click sobre una tabla se puede ver su estructura. Para ver los datos de una tabla se hará click sobre el `botón derecho de ratón sobre la tabla > Mostrar datos tabla`. De forma similar, podemos elegir la opción `Nueva consulta` que nos permitirá escribir sentencias SQL que trabajen sobre nuestra base de datos. Por ejemplo, fácilmente podremos visualizar todas las personas que hay en nuestra base de datos, más concretamente en nuestra tabla denominada People, escribiendo "`Select * From People`" y ejecutando la consulta (botón triángulo verde "play" del SQLQuery creado).

IMPORTANTE: Durante la realización de estas actividades, proteja su solución cuando lo considera conveniente y añada comentarios informativos de los cambios introducidos. Adicionalmente, por doble seguridad, haga una copia de la solución desde su workspace local (C:\Users\...) a otro directorio permanente. Nota: el contenido de C:\ de los laboratorios no es permanente.