# Exercices LINQ

## Objectifs

* Gérer plusieurs projets dans une solution
* Etre capable d’écrire une requête(query) LINQ
* Pouvoir utiliser les opérateurs LINQ suivants :
  + Select
  + Where, OrderBy
  + Count, Min, Max, Sum, Average
  + GroupBy, Join, GroupJoin

## Exercice 1 – Ajouter un projet à une solution

Nous allons ajouter un projet à une solution existante contenant déjà un projet avec des données en mémoire. Ce projet est une bibliothèque de classe, il ne s’exécute pas seul. Nous utiliserons ce projet pour tester nos queries LINQ. Nous allons créer un autre projet de type Application Console et l’ajouter à cette solution. C’est dans ce projet que nous créerons nos queries (dans le Main). Cette découpe en projet est classique en .NET et dans les applications d’entreprises.

1. Télécharger sur MooVin la solution LINQ\_DataContext.
2. Ouvrer cette solution en double-cliquant sur le fichier .sln.
3. Ajouter un projet de type Application Console
   1. Clic droit sur la solution -> Ajouter -> Projet
4. Ajouter une référence vers le projet LINQ\_DataContext
   1. Dépendances -> clic droit -> ajouter référence au projet LINQ\_DataContext
5. Définissez le projet que vous avez créé comme projet de démarrage
   1. Clic droit sur le projet -> définir en tant que projet de démarrage
6. Dans Program.cs, commencez par instancier le modèle de données
   1. DataContext dc = new DataContext();
7. Essayer le query test

**Student? jdepp = (from student in dc.Students**

**where student.Login == "jdepp"**

**select student).SingleOrDefault();**

**if (jdepp != null)**

**{**

**Console.WriteLine(jdepp.Last\_Name + jdepp.First\_Name);**

**}**

## Exercice 2 – Opérateur Select

**Ne faites les exercices « SUPPLEMENTAIRE » et « COMPLEXE » que si vous avez tout terminé !**

Les exercices supplémentaires sont similaires à d’autres exercices faits en séance. Ils peuvent être faits pour s’exercer. Les exercices complexes ne vous seront pas demandés à l’examen. Nous allons privilégier une autre approche pour faciliter la lisibilité et réduire la complexité du code.

### Exercice 2.1 (SUPPLEMENTAIRE)

Ecrire une requête pour présenter, pour chaque étudiant, le nom de l’étudiant, la date de naissance, le login et le résultat pour l’année de l’étudiant.

### Exercice 2.2

Ecrire une requête pour présenter, pour chaque étudiant, son nom complet (nom et prénom séparés par un espace), son id et sa date de naissance. L ’objectif pour cet exercice est de réaliser un maximum dans le query et non dans l’affichage.

### Exercice 2.3 (SUPPLEMENTAIRE)

Ecrire une requête pour présenter, pour chaque étudiant, dans une seule colonne l’ensemble des données relatives séparées par un « | ».

## Exercice 3 – Opérateurs Where, OrderBy

### Exercice 3.1

Pour chaque étudiant né avant 1955, donner le nom, le résultat annuel et le statut. Le statut prend la valeur « OK » si l’étudiant a obtenu au moins 12 comme résultat annuel et « KO » dans le cas contraire.

Exemple :

|  |  |  |
| --- | --- | --- |
| **Last\_name** | **Year\_result** | **statut** |
| Lucas | 10 | KO |
| Eastwood | 4 | KO |
| Connery | 12 | OK |
| …. |  |  |

### Exercice 3.2 ( SUPPLEMENTAIRE)

Donner pour chaque étudiant né entre 1955 et 1965 le nom, le résultat annuel et la catégorie à laquelle il appartient. La catégorie est en fonction du résultat obtenu : < 10 : inférieure ; > 10 : supérieure ; = 10 : neutre .

### Exercice 3.3 ( SUPPLEMENTAIRE)

Ecrire une requête pour présenter le nom et l’id de section de tous les étudiants dont le nom de famille se termine par « r ».

### Exercice 3.4

Ecrire une requête pour présenter le nom et le résultat annuel classé par résultats annuels décroissants de tous les étudiants qui ont obtenu un résultat inférieur ou égal à 3.

### Exercice 3.5

Ecrire une requête pour présenter le nom complet (nom et prénom séparés par un espace) et le résultat annuel classé par ordre croissant sur le nom des étudiants appartenant à la section 1110.

### Exercice 3.6 ( SUPPLEMENTAIRE)

Ecrire une requête pour présenter le nom, l’id de section et le résultat annuel classé par ordre croissant sur la section de tous les étudiants appartenant aux sections 1010 et 1020 ayant un résultat annuel qui n’est pas compris entre 12 et 18.

### Exercice 3.7 ( SUPPLEMENTAIRE)

Ecrire une requête pour présenter le nom, l’id de section et le résultat annuel sur 100 ( nommer une colonne result\_100) classé par ordre décroissant du résultat de tous les étudiants appartenant aux sections commençant par 13 et ayant un résultat annuel sur 100 inférieur ou égal à 60.

## Exercice 4– Opérateurs Count, Min, Max, Sum, Average

### Exercice 4.1

Donner le résultat annuel moyen pour l’ensemble des étudiants.

### Exercice 4.2 ( SUPPLEMENTAIRE)

Donner le plus haut résultat annuel obtenu par un étudiant.

### Exercice 4.3 ( SUPPLEMENTAIRE)

Donner la somme des résultats annuels.

### Exercice 4.4 ( SUPPLEMENTAIRE)

Donner le résultat annuel le plus faible.

### Exercice 4.5

Donner le nombre de lignes qui composent la « table » STUDENT.

## Exercice 5– Opérateurs GroupBy, Join, GroupJoin

### Exercice 5.1

Donner pour chaque section, le résultat maximum (Max\_Result) obtenu par les étudiants.

### Exercice 5.2 ( SUPPLEMENTAIRE)

Donner pour toutes les sections commençant par 10, le résultat annuel moyen (AVG\_Result) obtenu par les étudiants.

### Exercice 5.3

Donner le résultat moyen (AVG\_Result) et le mois en chiffre (BirtMonth) pour les étudiants né le même mois entre 1970 et 1985.

### Exercice 5.4 ( SUPPLEMENTAIRE)

Donner pour toutes les sections comptant plus de 3 étudiants, la moyenne des résultats annuels ( AVG\_Results).

### Exercice 5.5

Donner pour chaque cours le nom du professeur responsable ainsi que la section dont il fait partie.

|  |  |  |
| --- | --- | --- |
| **Course\_name** | **Section\_name** | **Professor\_name** |
| Derivatives | BA Sociology | giot |
| Marketing management | MSc Economics | decrop |
| Marketing engineering | MSc Management | zidda |
| Financial Management | BA Sociology | giot |
| Supply chain management … | MSc Management | scheppens |

### Exercice 5.6 ( SUPPLEMENTAIRE)

Donner pour chaque section, l’id, le nom et le nom de son délégué. Classer les sections dans l’ordre inverse des ids de section.

|  |  |  |
| --- | --- | --- |
| **Section\_id** | **Section\_name** | **Last\_name** |
| 1320 | MA Sociology | Basinger |
| 1310 | BA Sociology | Reeves |
| 1120 | MSc Economics | Basinger |
| 1110 | BSc Economics | Marceau |
| 1020 | MSc Management | Portman |
| 1010 | BSc Management | Willis |

### Exercice 5.7

Donner pour toutes les sections les professeurs qui en sont membres.

section\_id -> section\_name :

professor\_name1

professor\_name2

…

### Exercice 5.8 ( COMPLEXE)

Même objectif que le 5.7 mais seules les sections comportant au moins 1 professeur doivent être prises en compte.

### Exercice 5.9 ( COMPLEXE)

Donner à chaque étudiant ayant obtenu un résultat annuel supérieur ou égal à 12 son grade en fonction de son résultat annuel et sur base de la table grade. La liste doit être classée dans l’ordre alphabétique des grades attribués.

|  |  |  |
| --- | --- | --- |
| **Last\_name** | **Year\_result** | **Grade** |
| Berry | 18 | E |
| Garner | 18 | E |
| Garcia | 19 | E |
| Basinger | 19 | E |
| Connery | 12 | S |
| Roberts | 17 | TB |
| Bacon | 16 | TB |

### Exercice 5.10 ( COMPLEXE)

Donner la liste des professeurs et la section à laquelle ils se rapportent ainsi que le(s) cours ( nom du cours et crédits) dont le professeur est responsable. La liste est triée par ordre décroissant de crédits attribués à un cours.

|  |  |  |  |
| --- | --- | --- | --- |
| **Professor\_name** | **Section\_name** | **Course\_name** | **Course\_ects** |
| Giot | BA Sociology | Financial Management | 4.0 |
| Zidda | MSc Management | Marketing engineering | 4.0 |
| Decrop | MSc Economics | Marketing Management | 3.5 |
| Giot | BA Sociology | Derivatives | 3.0 |
| Scheppens | MSc Management | Supply chain …. | 2.5 |
| Lecourt | BA Sociology | NULL | NULL |
| Louveoux | BSc Economics | NULL | NULL |

### Exercice 5.11 (SUPPLEMENTAIRE)

Donner pour chaque professeur son id et le total des crédits ECTS (ECTSTOT) qui lui sont attribués. La liste est triée par ordre décroissant de la somme des crédits alloués.

|  |  |
| --- | --- |
| **Professor\_id** | **ECTSTOT** |
| 3 | 7 |
| 1 | 4 |
| 2 | 3.5 |
| 5 | 2.5 |
| 6 |  |
| 4 |  |

## Exercice 6 - Méthodes d’extensions

Pour cet exercice, créez une nouvelle solution et projet « Console Application ».

Créer une méthode d’extension « IsAPalindrome» pour la classe String.

Rappel : Un palindrome est un mot qui se lit de manière identique de gauche à droite et de droite à gauche.

Ex : été est un palindrome