Quelques exemples de requêtes Linq.

Linq est un outil très puissant mais qui est complexe à appréhender. Il serait fastidieux de passer en revu toutes les méthodes possibles de requêtes. Donc, **je vous propose quelques exemples avec les méthodes les plus utilisées.**

Si vous ne comprenez pas tout, **ce n'est pas grave**. Essayez simplement de comprendre comment Linq fonctionne et avec l'expérience vous serez capable de créer des requêtes complexes.

Soit une liste de nombre.

```
var listOfNumber = new List<int>() { 0,1,2,3,4,5,6,7,8,9,10};
```

Filtrer une liste avec .Where()

`var query = listOfNumber.Where(a => a > 5);`

Le prédicat `a => a > 5` indique qu'on cherche à **filtrer** dans la liste les éléments supérieur strictement à 5.

Vérifier la présence d'un élément dans une liste avec .Any()

`var query = listOfNumber.Any(x => x == 5 );`

Le prédicat `a => x == 5` indique qu'on veut savoir si le chiffre 5 **est présent dans la liste**. `.Any()` va retourner un `bool` (true ou false).

Compter le nombre d'élément avec .Count()

`var query = listOfNumber.Count();`

.Count va **compter** le nombre d'élément dans la liste.

`var query = listOfNumber.Count(x => x > 5);`

Le prédicat `x => x > 5` va compter le nombre d'élément supérieur à 5 strictement.

Classer les éléments d'une liste avec .OrderBy() ou .OrderByDescending()

`var query = listOfNumber.OrderBy(x => x);` classe par ordre ascendant.

`var query = listOfNumber.OrderByDescending(x => x);` classe par ordre descendant.

Appliquer une projection avec .Select();

`var query = listOfNumber.Select(x => x * 5);`

La méthode .Select va **"projeter" les éléments de la liste au travers du prédicat.** En faisant l'instruction ci-dessus, je vais avoir chaque élément de ma liste multiplié par 5. Donc query sera une liste de nombre de 0 5 10 15 20 25 30 35 40 45 50.

Chainer les méthodes.

Il est possible de chainer les méthodes. Par exemple, que fait le bout de code suivant ?

```
var query = listOfNumber .Where(x => x % 2 == 0) .Select(x => x * 5).OrderByDescending(x => x) .Take(2);
```

Réponse :

1. On va filtrer la liste pour récupérer uniquement les nombres pairs. ⇒ (donc la liste sera 0,2,4,6,8,10)
2. Puis on va multiplier ces nombres (donc les nombres pairs) par 5. ⇒ (0 10 20 30 40 50)
3. Puis on va les trier par ordre décroissant. ⇒ (50 40 30 20 10 0)
4. Puis on va prendre seulement les 2 premiers. ⇒ (50 40)

Donc query est une liste de deux nombres 50 et 40.