Rencontre R20

Performance

Bases de données et programmation Web

Aller chercher les bonnes infos

- C’est important d’aller chercher seulement les infos qu’on veut.

  - Ce sera super important quand vous serez en entreprise, surtout si l’entreprise utilise un cloud storage pour gérer ses BD.
    - Ce type de service charge selon le volume des requêtes exécutées.

  - C’est aussi super important quand vous développez des applications mobiles.

Performance

- Au niveau de la BD:
  - **Utiliser les vues** au lieu des requêtes Link complexes.

Lorsque vous créez une vue, la requête sous-jacente est optimisée par le serveur SQL. Son plan d'exécution est mis en cache dans le serveur pour une performance optimale.

Plus les requêtes Link sont complexes, moins elles sont performantes par rapport à une vue.

Performance

- Au niveau de la BD:
  - **Utiliser les index** pour optimiser l'exécution des requêtes.
    - Si vous avez une requête pour voir les données sur les ventes les plus récentes, vous voudrez faire un index sur le champ DateVente DESC.
    - Les index sur les clés étrangères peuvent améliorer l'efficacité des jointures.

Performance

- Au niveau du code:
  - **Utiliser des filtres** avant d'aller chercher les données avec .ToListAsync()

```
List<Course> premieresCourses = await _context.Courses.Take(30).ToListAsync();
```

Performance

- Au niveau du code:
  - Utiliser **.AsQueryable()** au lieu de .ToListAsync() pour obtenir un objet de type **IQueryable<>** puis appliquez les filtres sur cet objet.
  - Utiliser ensuite .ToListAsync() sur cet objet à la fin pour aller chercher seulement les données qui nous intéresse.

```
//Construction d'une requête sur toutes les courses
IQueryable<Course> toutesLesCourses =  _context.Courses.AsQueryable();

//Filtres pour garder les 30 courses les plus récentes
toutesLesCourses = toutesLesCourses.OrderByDescending(x => x.CourseId).Take(30);

//Effectivement aller chercher les données désirées dans la BD
List<Course> dernieresCourses = await toutesLesCourses.ToListAsync();
```

Performance

- Au niveau du code:
  - Aller chercher uniquement les colonnes qui vous intéressent, avec **.Select()**
    - Surtout si certaines colonnes de la table, que vous ne voulez pas voir, sont 'lourdes' comme nvarchar(max) par exemple.

C# Copier

```
foreach (var blog in context.Blogs)
{
    Console.WriteLine("Blog: " + blog.Url);
}
```

On veut seulement voir le Url du blog. Mais l'entité entière du Blog est extraite et les colonnes inutiles sont transférées à partir de la base de données.

Vous pouvez optimiser cela à l'aide de `Select` pour indiquer à EF les colonnes à sélectionner :

C# Copier

```
foreach (var blogName in context.Blogs.Select(b => b.Url))
{
    Console.WriteLine("Blog: " + blogName);
}
```