Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

I314A
.NET Outils et Concepts d'Application d'Entreprise

Performance et Scalabilité

Sommaire

- Scalabilité
- Processus, Thread et Tâches
- Exemple Tâche – async/await

Scalabilité & Performance

- Mise à l'échelle horizontale (horizontal scaling)
  - **Plusieurs serveurs/conteneurs sans état(stateless)**
  - API REST, RESTful
- Mise à l'échelle verticale (vertical scaling)
  - Augmenter les performances **d'un serveur (mémoire, nombre de requêtes, ...)**
  - async/await améliore la mise à l'échelle verticale des sites Webs
    - Du point de vue d'un client en particulier -> amélioration **légère** de l'expérience au niveau utilisateur
    - Du point de vue serveur -> on peut servir plus de clients en simultané **(Le gain se situe surtout ici !)**

Scalabilité & Performance

- Préparation petit déjeuner
  - Synchrone - 1 thread (1 personne) : la personne met le pain à griller, attend que le pain soit grillé, puis prépare le café à chauffer et ensuite mange.
  - Async/Await - 1 thread : la personne met le pain à griller, pendant ce temps prépare le café et ensuite mange
    - **Node.js par défaut**
  - Async/Await - Multi-thread (un couple) : l'homme s'occupe de mettre le pain à griller pendant que madame s'occupe de préparer le café et ensuite ils mangent
    - **Attention augmenter fortement le nombre de threads nécessite collaboration !**

Processus, Thread , Tâches

- Un processus est l'exécution d'un programme avec ses ressources allouées (mémoire, threads, …).
  - Le système d'exploitation isole chaque processus. Un processus ne peut donc pas modifier une variable d'un autre processus
  - Différents mécanismes de communication inter-processus existent (IPC)
- Un thread est l'exécution du code en tant que tel. Par défaut, chaque processus possède un thread.
  - Multithreading -> plusieurs thread pour un processus
- Une tâche est un moyen de gérer les thread à un haut niveau
  - Les systèmes d'exploitations utilise le multitâches préemptif -> chaque thread dispose d'un slot de temps d'exécution
  - Une tâche est donc asynchrone !

Tâche en .NET

- Classe Task
  - Classe introduite en . NET pour gérer les thread plus facilement
  - Il est en effet plus « ennuyant » de les gérer directement avec la classe Thread
  - Utilisation possible de async/await
  - Les méthodes peuvent renvoyer une Tâche typée

Types qui supportent le multitasking

- Dbcontext
- Dbset
- Streamreader
- Streawriter
- …

Tâche – async/await Exemple

```
[HttpGet]
public async Task<IList<CustomerDTO>> GetCustomers()
{
      IList<CustomerDTO> lst = await _repo.GetAllAsync();
      return lst.Select(e => CustomerToDTO(e)).ToList();
}
```

Exemple Web API Async/Await

- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio