Tâches d'arrière-plan

Vous pouvez définir des tâches d'arrière-plan qui seront exécutées après avoir retourné une réponse.

Ceci est utile pour les opérations qui doivent avoir lieu après une requête, mais où le client n'a pas réellement besoin d'attendre que l'opération soit terminée pour recevoir une réponse.

Cela comprend, par exemple :

- Les notifications par email envoyées après l'exécution d'une action :
  - Étant donné que se connecter à un serveur et envoyer un email a tendance à être «lent» (plusieurs secondes), vous pouvez retourner la réponse directement et envoyer la notification en arrière-plan.
- Traiter des données :
  - Par exemple, si vous recevez un fichier qui doit passer par un traitement lent, vous pouvez retourner une réponse «Accepted» (HTTP 202) puis faire le traitement en arrière-plan.

Utiliser `BackgroundTasks`

Pour commencer, importez `BackgroundTasks` et définissez un paramètre dans votre *fonction de chemin* avec `BackgroundTasks` comme type déclaré.

```
{!../../../docs_src/background_tasks/tutorial001.py!}
```

FastAPI créera l'objet de type `BackgroundTasks` pour vous et le passera comme paramètre.

Créer une fonction de tâche

Une fonction à exécuter comme tâche d'arrière-plan est juste une fonction standard qui peut recevoir des paramètres.

Elle peut être une fonction asynchrone (`async def`) ou une fonction normale (`def`), **FastAPI** saura la gérer correctement.

Dans cet exemple, la fonction de tâche écrira dans un fichier (afin de simuler un envoi d'email).

L'opération d'écriture n'utilisant ni `async` ni `await` , on définit la fonction avec un `def` normal.

```
{!../../../docs_src/background_tasks/tutorial001.py!}
```

Ajouter une tâche d'arrière-plan

Dans votre *fonction de chemin*, passez votre fonction de tâche à l'objet de type `BackgroundTasks` (`background_tasks` ici) grâce à la méthode `.add_task()` :

```
{!../../../docs_src/background_tasks/tutorial001.py!}
```

`.add_task()` reçoit comme arguments :

- Une fonction de tâche à exécuter en arrière-plan (`write_notification`).
- Les arguments positionnels à passer à la fonction de tâche dans l'ordre (`email`).

- Les arguments nommés à passer à la fonction de tâche (`message="some notification"`).

Injection de dépendances

Utiliser `BackgroundTasks` fonctionne aussi avec le système d'injection de dépendances. Vous pouvez déclarer un paramètre de type `BackgroundTasks` à différents niveaux : dans une *fonction de chemin*, dans une dépendance, dans une sous-dépendance...

FastAPI sait quoi faire dans chaque cas et comment réutiliser le même objet, afin que tous les paramètres de type `BackgroundTasks` soient fusionnés et que les tâches soient exécutées en arrière-plan :

```
{!../../../docs_src/background_tasks/tutorial002.py!}
```

Dans cet exemple, les messages seront écrits dans le fichier `log.txt` après que la réponse soit envoyée.

S'il y avait une `query` (paramètre nommé `q`) dans la requête, alors elle sera écrite dans `log.txt` via une tâche d'arrière-plan.

Et ensuite une autre tâche d'arrière-plan (générée dans les paramètres de la *la fonction de chemin*) écrira un message dans `log.txt` comprenant le paramètre de chemin `email` .

Détails techniques

La classe `BackgroundTasks` provient directement de `starlette.background` .

Elle est importée/incluse directement dans **FastAPI** pour que vous puissiez l'importer depuis `fastapi` et éviter d'importer accidentellement `BackgroundTask` (sans `s` à la fin) depuis `starlette.background` .

En utilisant seulement `BackgroundTasks` (et non `BackgroundTask`), il est possible de l'utiliser en tant que paramètre de *fonction de chemin* et de laisser **FastAPI** gérer le reste pour vous, comme en utilisant l'objet `Request` directement.

Il est tout de même possible d'utiliser `BackgroundTask` seul dans **FastAPI**, mais dans ce cas il faut créer l'objet dans le code et renvoyer une `Response` Starlette l'incluant.

Plus de détails sont disponibles dans la documentation officielle de Starlette sur les tâches d'arrière-plan (via leurs classes `BackgroundTasks` et `BackgroundTask`).

Avertissement

Si vous avez besoin de réaliser des traitements lourds en tâche d'arrière-plan et que vous n'avez pas besoin que ces traitements aient lieu dans le même process (par exemple, pas besoin de partager la mémoire, les variables, etc.), il peut s'avérer profitable d'utiliser des outils plus importants tels que Celery.

Ces outils nécessitent généralement des configurations plus complexes ainsi qu'un gestionnaire de queue de message, comme RabbitMQ ou Redis, mais ils permettent d'exécuter des tâches d'arrière-plan dans différents process, et potentiellement, sur plusieurs serveurs.

Pour voir un exemple, allez voir les Générateurs de projets{.internal-link target=_blank}, ils incluent tous Celery déjà configuré.

Mais si vous avez besoin d'accéder aux variables et objets de la même application **FastAPI**, ou si vous avez besoin d'effectuer de petites tâches d'arrière-plan (comme envoyer des notifications par email), vous pouvez simplement

vous contenter d'utiliser `BackgroundTasks`.

Résumé

Importez et utilisez `BackgroundTasks` grâce aux paramètres de *fonction de chemin* et les dépendances pour ajouter des tâches d'arrière-plan.