Pattern Singleton

Introduction

Le pattern Singleton est un patron de conception (design pattern) appartenant à la catégorie Créationnel. Il vise à garantir qu'une seule instance d'une classe soit créée et à fournir un point d'accès global à cette instance. Il est couramment utilisé pour contrôler des ressources partagées (ex: bases de données, logs, gestionnaires de configuration).
Objectif :empêcher la création multiple d'une classe sensible et offrir un accès centralisé et contrôlé à cette instance unique.

Composants

- Instance privée statique de la classe.
- Constructeur privé pour empêcher l'instanciation externe.
- Méthode publique statique (getInstance()) qui crée et/ou retourne l'instance unique.

Structure

```
class Singleton {
   private static instance: Singleton
   private constructor() { }

   static getInstance() {
      if (instance == null) {
         instance = new Singleton()
      }
      return instance
   }
}
```

Utilisation

- Contrôle strict sur l'instance : impossible de dupliquer accidentellement la ressource.
- Accès global facile à la ressource partagée.
- Économie de ressources pour des objets coûteux.
- Gestionnaire de configuration.
- Gestionnaire de logs (logger).
- Connexion à une base de données.
- Services d'impression, caches globaux.

Conclusion

Le pattern Singleton est idéal pour gérer des ressources uniques et assurer un accès contrôlé et centralisé à une instance partagée dans toute une application.
Cependant, mal utilisé, il peut introduire des problèmes d'architecture comme un couplage excessif ou des difficultés de testabilité.
Il est recommandé d'appliquer le Singleton avec discernement, particulièrement dans des environnements multithreadés, où l'accès concurrent doit être sécurisé pour éviter des anomalies.