

Patron singleton et factory methode

exemples de bibliotheques/frameworks Java qui utilisent le patron Singleton en interne :

1. Spring Framework

Dans Spring, le **patron Singleton** est l'un des modes de gestion des beans. Par défaut, tous les beans dans Spring sont des singletons.

Exemple d'implémentation avec Spring :

```
java Copier le code

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public MySingletonService mySingletonService() {
        return new MySingletonService();
    }
}

public class MySingletonService {
    public void doSomething() {
        System.out.println("Service Singleton exécuté !");
    }
}

// Utilisation du service
```

```
// Utilisation du service
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(AppConf

        MySingletonService service1 = context.getBean(MySingletonService.class);
        MySingletonService service2 = context.getBean(MySingletonService.class);

        // Vérification que les deux instances sont identiques (Singleton)
        System.out.println(service1 == service2); // true
    }
}
```

Explication :


- Spring gère les beans en mode Singleton par défaut, ce qui signifie que chaque bean est instancié une seule fois dans le cycle de vie de l'application.
- Il utilise un conteneur IoC (Inversion of Control) pour gérer les dépendances et l'instanciation des classes.

3. Apache Commons Logging

Apache Commons Logging utilise le patron Factory Method pour créer des instances de logger en fonction de la configuration du système.

Exemple avec `LogFactory` :

java

 Copier le code

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class FactoryMethodExample {
    // Utilise la méthode Factory pour obtenir un Logger
    private static final Log logger = LogFactory.getLog(FactoryMethodExample.class)

    public static void main(String[] args) {
        // Utilisation du Logger
        logger.info("Message de journalisation via Factory Method");
    }
}
```

Explication :

- `LogFactory` utilise le patron Factory Method pour créer et renvoyer des instances de `Log` qui dépendent du moteur de journalisation sous-jacent (comme Log4j, JDK logging, etc.).
- Le programmeur n'a pas besoin de connaître la classe concrète du Logger, car c'est le rôle de la Factory de l'abstraire.