Instituto Politécnico Nacional

Escuela Superior de Cómputo

Web App Development.

Tarea 5 : Patrón de diseo Singleton

Profesor: M. en C. José Asunción Enríquez Zárate

Alumno: Mauro Sampayo Hernández

mauro_luigi@hotmail.com

3CM18

4 de enero de 2022

Índice

1. Introducción

Los patrones de diseño son herramientas muy útiles en la programación orientada a objetos, pues proporcionan de plantillas probadas y comprobadas para resolver tareas de programación específicas.

Uno de estos patrones es el patrón Singleton, el cuál a pesar de ser un patrón de diseño viejo, resulta bastante útil y su uso cuenta con diversas ventajas.

En este documento se mostrará en que cosiste el patrón de diseño Singleton, sus ventajas y desventajas y su implementación en Hibernate.

2. Desarrollo

El patrón Singleton es un patrón de diseño cuyo propósito es evitar que sea creado más de un objeto por clase. Esto se logra creando el objeto deseado en una clase y recuperándolo como una instancia estática.

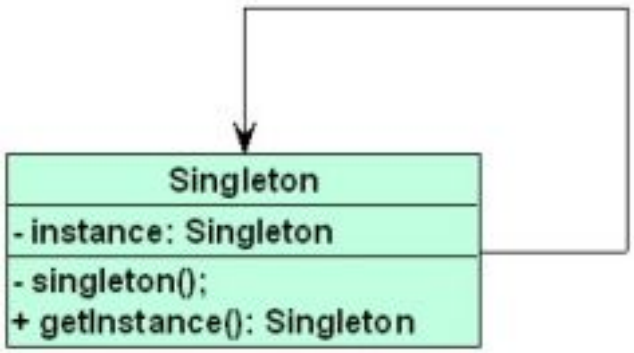
2.1. Uso del patrón Singleton

Al momento de usar el patrón Singleton para crear una instancia de una clase, el patrón se asegura de que realmente sólo permanezca con esta instancia única. Singleton hace que esta clase de software sea accesible globalmente. Para asegurarse de que permanezca con una sola instancia única, se debe impedir que los usuarios creen nuevas instancias; esto se logra mediante el constructor, declarando el patrón como "privado", lo que significa que sólo el código en el Singleton puede instanciar el Singleton en sí mismo y, por lo tanto, esto garantiza que sólo un mismo objeto puede llegar al usuario. Si esta instancia ya existe, no se crea ninguna nueva instancia.

A continuación, se muestra un ejemplo sencillo de un patrón Singleton:

```
1         public class Singleton {
2             private static Singleton instance;
3             private Singleton() {}
4             public static  getInstance() {
5                 if (instance == null) {
6                     instance = new Singleton();
7                 }
8                 return instance;
9             }
10        }
11
```

La representación del patrón Singleton en el diagrama UML es el siguiente:



2.2. Ventajas del patrón Singleton

- Un patrón Singleton puede escribirse de forma rápida y sencilla al no estar poblado de innumerables variables (globales).
- El patrón encapsula su creación, lo que significa que se puede ejercer un control preciso sobre cuándo y cómo se accede a él.
- Un patrón Singleton existente puede derivarse mediante subclases para cumplir nuevas funcionalidades.
- Un Singleton se crea exactamente cuándo se necesita (lazy loading).

2.3. Desventajas del patrón Singleton

- El uso desinhibido de patrón Singleton puede conducir a un estado similar al de la programación procedimental (no orientada a objetos), y a ensuciar el código fuente.
- La disponibilidad global de patrones Singleton plantea riesgos si se manejan datos sensibles. Esto pasa debido a que, si se hacen cambios en el Singleton, no se podrá rastrear qué partes del programa están afectadas, lo que dificulta el mantenimiento de software, al ser que los fallos de funcionamiento son difíciles de rastrear.
- En aplicaciones con muchos usuarios (aplicaciones multiusuario), un patrón Singleton puede reducir el rendimiento del programa.

2.4. Implementación del patrón Singleton en Hibernate

Para implementar el patrón Singleton en Hibernate se debe de crear una clase aparte la cual se encargará de la implementación de un objeto Session Factory de Hibernate como Singleton.

Debido a que el "utility class" es responsable únicamente de crear el objeto Session Factory de Hibernate como patrón Singleton y retornar dicho objeto, no es necesario la creación de un objeto adicional para el "utility class".

El constructor del "utility class" debe ser privado, para de esta manera asegurar que ninguna otra clase pueda crear objetos de la sesión de Hibernate.

A continuación, se presenta un ejemplo de implementación del patrón Singleton en Hibernate:

2.4.1. HibernateUtility.java

```
1   package com.onlinetutorialspoint.config;
2
3   import org.hibernate.SessionFactory;
4   import org.hibernate.cfg.Configuration;
5
6   public class HibernateUtility {
7
8       public static SessionFactory factory;
9       //Para no permitir la creacion de objetos a otras clases
10
11      private HibernateUtility() {
12      }
13      //Creando el objeto Hibernate SessionFactory como Singleton
14
15      public static synchronized SessionFactory getSessionFactory() {
16
17          if (factory == null) {
18              factory = new Configuration().configure("hibernate.cfg.xml").
19                      buildSessionFactory();
20          }
21          return factory;
22      }
23  }
24
```

2.4.2. Main.java

```
1   package com.onlinetutorialspoint.service;
2
3   import com.onlinetutorialspoint.config.HibernateUtility;
4   import org.hibernate.SessionFactory;
5
6   public class Main {
7     public static void main(String[] args) {
8       SessionFactory sessionFactory = HibernateUtility.getSessionFactory();
9       System.out.println("Session Factory : " + sessionFactory.hashCode());
10      SessionFactory sessionFactory2 = HibernateUtility.getSessionFactory();
11      System.out.println("Session Factory 2 : " + sessionFactory2.hashCode());
12      SessionFactory sessionFactory3 = HibernateUtility.getSessionFactory();
13      System.out.println("Session Factory 3 : " + sessionFactory3.hashCode());
14    }
15  }
16
```

3. Conclusión

El patrón Singleton es una herramienta bastante poderosa en la programación orientada objetos, al permitir el ahorro de memoria y recursos durante la ejecución de una aplicación cuya programación este orientada a objetos, al ser que solo permite la creación de un solo objeto por clase. Adicionalmente la implementación de aplicaciones con este patrón de diseño resulta ser mucho más sencilla y rápida.

Mauro Sampayo Hernández

4. Referencias Bibliográficas

Referencias

[1] *Patrones de Diseño Software* **Informática PC** [accesed 2021 Jan 02] *https://informaticapc.com/patrones-de-diseno/singleton.php*

[2] *Patron singleton: una clase propia* **Digital Guides IONOS, 2021** [accesed 2021 Jan 02] *https://www.baeldung.com/java-request-getsession*

[3] *Singleton Hibernate SessionFactory Example* **Onlinetutorialspoint, 2015** [accesed 2021 Jan 02] *https://www.onlinetutorialspoint.com/hibernate/singleton-hibernate-sessionfactory-example.html*