Patrones Creacionales

Singleton

Daza Corredor Alejandro Paolo

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [Christopher Alexander].

1. Introducción

Muchas veces nos encontramos con la necesidad de garantizar que solo se pueda instanciar un elemento de una clase en nuestras aplicaciones, ejemplos existen muchos, conexiones a bases de datos, controladores en aplicaciones web, etc; por un lado en ocasiones no sabemos como garantizar esta restricción, del otro lado se usa de forma incorrecta o en los casos no apropiados esta restricción. Veamos que nos dicen los patrones creacionales del patrón que nos puede ayudar en estos casos el patrón singleton.

3. Singleton

Nombre del patrón:
Singleton

Clasificación del patrón:
Creacional.

Intención:
Provee una interfaz para crear un objeto, pero deja a las subclases decidir cual clase instanciar, Factory Method permite a una clase aplazar la instanciación a las subclases.

También conocido como:
Singleton

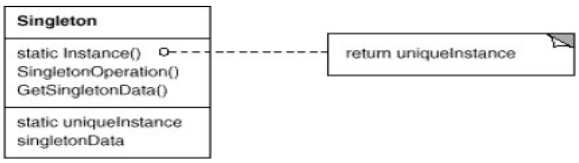
Motivación:
En ciertos sistemas es importante que una clase solo tenga una sola clase, por ejemplo en el manejo de impresoras, el control de ventanas, el control de archivos,etc.

Una buena solución es hacer que las clases por si mismas se responsabilicen por mantener su única instancia. La clase asegurara que ninguna otra instancia será creada y proveerá una forma de obtener dicha instancia. Este es el patrón Singleton.

Aplicabilidad:
El patrón Singleton debe ser usado cuando:

- Solo debe existir una única instancia de una clase y debe existir un acceso a esta.
- Cuando esta única instancia debe ser extendida por subclases y los clientes deben ser capaces de extender esta instancia sin la modificación de sus códigos.

Estructura:



Participantes:

- Singleton: define una única instancia de si misma y una operación para obtener esta instancia.

Colaboraciones:

- Los clientes solo pueden acceder a la instancia a través de la operación que la habilita.

Consecuencias:

- Controla el acceso a la única instancia. Debido a que encapsula la creación de una única instancia, puede mantener un control de como y cuando los clientes pueden acceder a ella.
- Reduce el name-space. El patrón singleton es una mejora sobre las variables globales. Se puede pulular el name-space con variables globales que almacenan instancias únicas.
- Permite el refinamiento de operaciones y representaciones. La clase singleton puede ser heredada, y es fácil configurar una aplicación con estas subclases.
- Permite la extensión a múltiples instancia controlando el número. Al igual que se puede controlar que sea una única instancia, se puede controlar un numero de estas.

Implementación:

- Asegurando una única instancia. La clase debe ser escrita de tal manera que solo una única instancia pueda ser creada. Una forma común de lograr esto es ocultar la operación que crea la instancia tras una operación de la clase que garantice que solo una instancia se podrá crear. Los clientes del singleton solo podrán acceder a la única instancia a través de un miembro de la clase encargada de esto.

Código de ejemplo:

```
public class Singleton {

    static private Singleton singleton = null;

    private Singleton() { }

    static public Singleton getSingleton() {

        if (singleton == null) {
            singleton = new Singleton();
        }
        return singleton;
    }

}
```

Usos conocidos:

- Muy usado en WidgetKits y en el manejo de Sesiones.

Patrones relacionados:

- Abstract Factory, Builder y Prototype.

Referencias bibliográficas

[1] Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four)

[2] Desing Patterns Java. James W Cooper. ebook – Addison Wesley.

[3] Patrones de Diseño, Diseño de Software Orientado a Objetos - Joaquin Garcia. http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

[4] Singletons en Java, el patrón instancia única – David Bernal - http://www.elrincondelprogramador.com/default.asp?pag=articulos/leer.asp&id=45

Daza Corredor, Alejandro Paolo
Ingeniero de Sistemas y Especialista de Ingeniería de Software, egresado de la Universidad Distrital Francisco José de Caldas, Docente TCO del proyecto curricular de Ingeniería de Sistemas en el área de programación de la Universidad Distrital y docente de medio tiempo de UNITEC en la Escuela de Ingeniería en el área de programación.

e-mail: apdaza@gmail.com