
Doan Van Toan @doan.van.toan Theo dõi
4.9K 264 26

Đã đăng vào thg 12 22, 2017 8:18 CH - 6 phút đọc
60.1K 11 43

Học Singleton Pattern trong 5 phút.

Đặt vấn đề

Trong bài viết này mình sẽ giúp các bạn trả lời 4 câu hỏi về Single pattern trong vòng 5 phút.

1. Singleton Pattern là gì?
2. Tại sao cần dùng Singleton Pattern
3. Làm thế nào để implement Singleton Pattern
4. Có những cách nào để implement Singleton Pattern Liệu có đủ không nhỉ các bạn cùng theo dõi nhé

1. Single Pattern là gì?

Theo Gang of Four patterns một cuốn sách rất nổi tiếng về design pattern thì Single Pattern là một design pattern trong số 5 design pattern thuộc nhóm Creational Design Pattern

Creational	**Structure**	**Behavioral**
Abstract factory	Adapter	Chain of responsibility
Builder	Bridge	Command
Factory	Composite	Interpreter
Prototype	Decorator	Iterator
#Singleton	Facade	Mediator
Flyweight	Memento	Memento
	Proxy	Observer
		Strategy
		Template Method

Creational	Structure	Behavioral
		Visitor

Single Pattern là một design pattern mà

1. Đảm bảo rằng một class chỉ có duy nhất một instance (khởi tạo - mình xin phép để nguyên không dịch từ này)
2. Và cung cấp một cáchs toàn cầu để truy cấp tới instance đó.

Vậy tại sao cần phải sử dụng Single Pattern

2. Tại sao cần dùng Singleton Pattern?

Hầu hết các đối tượng trong một ứng dụng đều chịu trách nhiệm cho công việc của chúng và truy xuất dữ liệu tự lưu trữ (self-contained data) và các tham chiếu trong phạm vi được đưa ra của chúng. Tuy nhiên, có nhiều đối tượng có thêm những nhiệm vụ và có ảnh hưởng rộng hơn, chẳng hạn như quản lý các nguồn tài nguyên bị giới hạn hoặc theo dõi toàn bộ trạng thái của hệ thống. Ví dụ có thể có rất nhiều máy in trong hệ thống nhưng chỉ có thể tồn tại duy nhất một Sprinter Spooler (Phần quản lý máy in)

Hay

giả sử trong ứng dụng có chức năng bật tắt nhạc nền chẳng hạn, khi người dùng mở app thì ứng dụng sẽ tự động mở nhạc nền và nếu người dùng muốn tắt thì phải vào setting trong app để tắt nó, trong setting của app cho phép người dùng quản lí việc mở hay tắt nhạc, và trong trường hợp này bạn sẽ cần sử dụng singleton để quản lí việc này. Chắc chắn bạn phải cần duy nhất 1 instance để có thể ra lệnh bật hay tắt, tại sao ? vì đơn giản bạn không thể tạo 1 instance để mở nhạc rồi sau đó lại tạo 1 instance khác để tắt nhạc, lúc này sẽ có 2 instance được tạo ra, 2 instance này không liên quan đến nhau nên không thể thực hiện thực hiện việc cho nhau được, bạn phải hiểu rằng instance nào bật thì chỉ có instance đó mới được phép tắt nên dẫn đến phải cần 1 instance.

3.Làm thế nào để implement Singleton Pattern

1. Làm sao để 1 class chỉ có thể có duy nhất 1instance? **Trả lời**

- Private constructor của class đó để đảm bảo rằng class lớp khác không thể truy cập vào constructor và tạo ra instance mới
- Tạo một biến private static là thể hiện của class đó để đảm bảo rằng nó là duy nhất và chỉ được tạo ra trong class đó thôi.

2. Làm sao để có thể ccung cấp một cáchs toàn cầu để truy cấp tới instance đó. **Trả lời**

- Tạo một public static menthod trả về instance vừa khởi tạo bên trên, đây là cách duy nhất để các class khác có thể truy cập vào instance của class này

Vậy cụ thể có những cách nào để implement Singleton Pattern

4. Có những cách nào để implement Singleton Pattern

4.1 Eager initialization

```
public class EagerInitializedSingleton {

   private static final EagerInitializedSingleton instance = new EagerInitializedSingleton();

   //private constructor to avoid client applications to use constructor
   private EagerInitializedSingleton(){}

   public static EagerInitializedSingleton getInstance(){
       return instance;
   }
}
```

Đây là cách dễ nhất nhưng nó có một nhược điểm là mặc dù instance đã được khởi tạo nhưng có thể sẽ không dùng tới. vì vậy chúng ta có cách thứ 2.

4.2 Lazy initialization

```
public class LazyInitializedSingleton {

    private static LazyInitializedSingleton instance;

    private LazyInitializedSingleton(){}

    public static LazyInitializedSingleton getInstance(){
        if(instance == null){
            instance = new LazyInitializedSingleton();
        }
        return instance;
        }
}
```

Cách này đã khắc phục được nhược điểm của cách 1 Eager initialization, chỉ khi nào gelnstance được gọi thì instance mới được khởi tạo. Tuy nhiên cách này chỉ sử dụng tốt trong trường hợp đơn luồng, trường hợp nếu có 2

của instance. Vậy ta phải làm sao với trường hợp đa luồng. chúng ta đi tới cách tiếp theo

4.3 Thread Safe initialization

```
public class ThreadSafeSingleton {

    private static ThreadSafeSingleton instance;

    private ThreadSafeSingleton(){}

    public static synchronized ThreadSafeSingleton getInstance(){
        if(instance == null){
            instance = new ThreadSafeSingleton();
        }
        return instance;
     }
```

Cách đơn gản nhất là chúng ta gọi phương thức synchronized của hàm getInstance() và như vậy hệ thống đảm bảo rằng tại cùng một thời điểm chỉ có thể có 1 luồng có thể truy cập vào hàm getInstance(), và đảm bảo rằng chỉ có duy nhất 1 thể hiện của class Tuy nhiên một menthod synchronized sẽ chạy rất chậm và tốn hiệu năng vì vậy chúng ta cần cải tiến nó đi 1 chút.

4.4 Thread Safe Upgrade initialization

Mình tạm gọi nó là Thread Safe Upgrade initialization, thay vì chúng ta Thread Safe cả menthod getInstance() chúng ta chỉ Thread Safe một đoạn mã quan trong

```
ublic class ThreadSafeSingleton {
    private static ThreadSafeSingleton instance;
    private ThreadSafeSingleton(){}

    public static ThreadSafeSingleton getInstance(){
        if(instance == null){
            synchronized(ThreadSafeSingleton.class){
                if(instance == null){
                   instance = new ThreadSafeSingleton();
                }
            }
        }
        return instance;
     }
}
```

Có rất nhiều cách implement cho Singleton, mình thì hay sử dụng cách 2 cho những ứng dụng chỉ làm việc với 1 thread và cách thứ 4 cho trường hợp đa luồng. Các bạn hãy chọn cho mình cách implement phù hợp cho từng trường hợp nhé.👍 Trên đây là phần giới thiệu của mình về Singleton, các bạn có thể tham khảo slide của mình tại link sau https://goo.gl/KUtZsW. Chúc các bạn học tốt!

singleton Design Pattern