Singleton Pattern

Singleton Pattern không quá khó để tìm hiểu mà lợi ích của nó đem lại cho ta còn nhiều hơn, dường như là nó đang bị xem nhẹ và hiểu nhầm.

Nếu đây là lần đầu bạn nghe về “Singleton Pattern” hoặc vừa mới làm quen thì bây giờ mình sẽ giới thiệu về nó.

Singleton là một trong năm design pattern của nhóm Creational Design Pattern, Singleton là một pattern khởi tạo mà:

* Nó đảm bảo một lớp chi có duy nhất một instance(khởi tạo).
* Nó cung cấp một cách toàn diện để truy cập tới instance đó.

Singleton không phải đơn độc như tên gọi của nó mà ở đây là nó chỉ có một instance duy nhất.

Chúng ta nên chọn Singleton Pattern bởi vì:

Đa phần các đối tượng trong một ứng dụng đều chịu trách nhiệm cho công việc của chúng, truy xuất dữ liệu tự lưu trữ (self-contained data) và các tham chiếu trong phạm vi của chúng.

Tuy nhiên, nhiều đối tượng có thêm những nhiệm vụ và có ảnh hưởng của nó rộng hơn. Chẳng hạn như quản lý các nguồn tài nguyên bị giới hạn hay là theo dõi toàn bộ trạng thái của hệ thống.

**Ví dụ về singleton:**

Về một ứng dụng có chức năng bật tắt nhạc nền.

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 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. Đơn giản vì 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.

Khi instance nào bật thì chỉ có instance đó mới được phép tắt nên dẫn đến phải cần 1 instance.

## Singleton dùng để làm gì?

Vấn đề mà singleton giải quyết là:

* Đảm bảo rằng 1 class chỉ có 1 instance duy nhất và class này luôn sẵn sàng để sử dụng ở bất kỳ thời điểm hoặc vị trí nào trong phần mềm ứng dụng của chúng ta.
* Việc quản lý việc truy cập tốt hơn vì chỉ có một thể hiện duy nhất.
* Có thể quản lý số lượng thể hiện của một lớp trong giớn hạn chỉ định.

## Mục đích của Singleton Pattern

Một class thuộc dạng Singleton có nghĩa là: nó chỉ có một instance duy nhất, bất kỳ ở đâu đều có thể truy cập tới instance của class singleton đó.

## Những lưu ý khi sử dụng Singleton

Bạn không nên hiểu máy móc là Singleton có nghĩa tồn tại chính xác đúng 1 thể hiện. Tùy vào trường hợp, có thể có những thể hiện khác nhau cho những mục đích khác nhau. Đó cũng là ưu điểm của Singleton so với việc sử dụng biến toàn cục (global variable).

Những design pattern có thể dùng cùng với Singleton. Ví dụ, Abstract Factory, Builder, Prototype.

Bạn cần thận trọng với những xử lý đa luồng (multithreading). Vì 2 luồng khác nhau có thể gọi phương thức khởi tạo ở cùng một thời điểm và sẽ sinh ra hai thể hiện. Nhưng nếu thực hiện đồng bộ (synchronized) phương thức khởi tạo lại gây ảnh hưởng tới hiệu suất.

Singleton là toàn cục. Chính vì vậy, nếu muốn truyền một đối tượng A cho đối tượng B xử lý, bạn hãy cân nhắc xem bạn có thật sự cần một đối tượng toàn cục hay không.

**Ưu điểm và khuyết điểm của Singleton Pattern**

Về mặt **ưu điểm** của pattern này đem lại quá rõ ràng:

* Ai cũng có thể truy cập vào instance của singleton class,  thực hiện gọi nó ở bất cứ đâu.
* Dữ liệu ứng dụng không thay đổi bởi chỉ có một instance duy nhất.
* Singleton class có hỗ trợ interface trong khi static class thì lại không. Đây là sự khác biệt giúp chúng ta nhận biết cần phải xác định để tạo static class hay singleton class.
* Hỗ trợ kế thừa, static class thì không hỗ trợ kế thừa.

Một số **nhược điểm** của pattern này:

* Cần phải sử dụng tới keyword trung gian là .instance (có thể cached khi sử dụng tần suất nhiều).
* Chỉ tạo một instance duy nhất.
* Làm tăng kết nối giữa các script và điều này thì không tốt, các scripts con phụ thuộc quá nhiều vào các singletons và khi singletons thay đổi có thể gây ra bug hoặc lỗi.
* Không sử dụng được đa hình.

## Những lưu ý khi sử dụng Singleton

Bạn không nên hiểu máy móc là Singleton có nghĩa tồn tại chính xác đúng 1 thể hiện. Tùy vào trường hợp, có thể có những thể hiện khác nhau cho những mục đích khác nhau. Đó cũng là ưu điểm của Singleton so với việc sử dụng biến toàn cục (global variable).

Những design pattern có thể dùng cùng với Singleton. Ví dụ, Abstract Factory, Builder, Prototype.

Bạn cần thận trọng với những xử lý đa luồng (multithreading). Vì 2 luồng khác nhau có thể gọi phương thức khởi tạo ở cùng một thời điểm và sẽ sinh ra hai thể hiện. Nhưng nếu thực hiện đồng bộ (synchronized) phương thức khởi tạo lại gây ảnh hưởng tới hiệu suất.

Singleton là toàn cục. Chính vì vậy, nếu muốn truyền một đối tượng A cho đối tượng B xử lý, bạn hãy cân nhắc xem bạn có thật sự cần một đối tượng toàn cục hay không.

### Sử dụng Singleton Pattern khi nào?

Một số trường hợp sử dụng của Singleton Pattern thường gặp là:

* Vì class dùng Singleton chỉ tồn tại 1 Instance nên nó thường được dùng cho các trường hợp giải quyết những bài toán cần truy cập vào các ứng dụng như: Shared resource, Logger, Configuration, Thread pool,..
* Sử dụng trong một số class của core java như: java.lang.Runtime, java.awt.Desktop.

Bài viết giới thiệu về singleton là gì và những đặc điểm của nó. Cảm ơn các bạn đã xem bào viết của mình.