

Classe en Singleton

By B0b.K !113r

1) Qu'est ce qu'une classe en singleton ?

Une classe en singleton permet de construire un seul objet de cette classe et que celui-ci soit accessible depuis tout le programme. Cela permet donc de créer un objet global facilement appelable de partout !

2) Exemple d'une classe en Singleton

Tout d'abord nous allons créer le (.h) header de la classe :

```
#ifndef _SINGLETON_H_
#define _SINGLETON_H_

// Classe en Singleton
class Csingleton
{
private:
    // variable x et y
    int x,y;

    // constructeur en private
    Csingleton();

public:
    ~Csingleton();

    // fonction static qui retourne un pointeur sur l instance
    static Csingleton * GetInstance();

    // Fonction inline pour Get/Set x et y
    inline void SetX(int _x){ x = _x; };
    inline void SetY(int _y){ y = _y; };

    inline int GetX(){ return x; };
    inline int GetY(){ return y; };
};

#endif
```

Les caractéristiques principales d'une classe en singleton sont :

- le constructeur en **private**
- une fonction **static** qui renvoi une instance de la classe singleton

Le constructeur est en **private** pour que l'on ne puisse pas créer d'instance de cette classe autre part que dans la classe elle-même.

La fonction **static** permet d'être appelé sans nécessiter une instance de la classe singleton.

Le (.cpp) source de la classe :

```
#include "singleton.h"

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

Csingleton::Csingleton():x(0),y(0){ }

Csingleton::~~Csingleton(){ }

////////////////////////////////////
// Fonctions Retournant la seule instance de la classe
////////////////////////////////////
Csingleton * Csingleton::GetInstance()
{
    // crée l'instance 1 seule fois
    static Csingleton Instance;

    // retourne un pointeur sur l'instance
    return &Instance;
}
```

Ici c'est la fonction déclarée en `static` qui est importante. En effet, elle génère une variable du type de la classe en `static` et renvoie une référence sur elle. Grâce à ceci, la variable sera créée au premier appel à cette fonction, mais par la suite cette déclaration sera sautée. Nous avons donc qu'une seule instance de la classe qui sera déclaré pour l'ensemble du programme.

3) Exemple d'un main

Afin de comprendre comment marche et comment utiliser un singleton, je vous fais voir comment jouer avec la classe créée au dessus.

```
#include <windows.h>
#include <iostream>
#include "singleton.h"

void EcritXetY()
{
    // affiche le contenu de x et y
    printf("X : %d, Y : %d",Csingleton::GetInstance()->GetX(),Csingleton::GetInstance()->GetY());
}
```

```
void main()
{
    // on écrit les valeurs de x et y
    EcrivXetY();

    // on init x et y
    Csingleton::GetInstance()->SetX(10);
    Csingleton::GetInstance()->SetY(-10);

    // on re-écrit le contenu de x et y
    EcrivXetY();
}
```

Ici nous avons initialisé notre instance de la classe singleton dans le main et nous pouvons y accéder dans la fonction EcrivXetY sans la passer en paramètre.