

Home › Design Patterns › Creational patterns › Singleton

Singleton in C++: Before and after

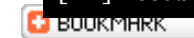


[Read full article](#)



[See the code](#)

[+] Feedback



Search



contents

Design Patterns

§ Creational patterns

- [Creational patterns](#)
- [Abstract Factory Design Pattern](#)
- [Builder Design Pattern](#)
- [Factory Method Design Pattern](#)
- [Object Pool Design Pattern](#)
- [Prototype Design Pattern](#)
- [Singleton Design Pattern](#)

§ Structural patterns

§ Behavioral patterns

Before

A global variable is default initialized - when it is declared - but it is not initialized in earnest until its first use. This requires that the initialization code be replicated throughout the application.

```
class GlobalClass
{
    int m_value;
public:
    GlobalClass(int v = 0)
    {
        m_value = v;
    }
}
```

C++

```

    }
    int get_value()
    {
        return m_value;
    }
    void set_value(int v)
    {
        m_value = v;
    }
};

// Default initialization
GlobalClass *global_ptr = 0;

void foo(void)
{
    // Initialization on first use
    if (!global_ptr)
        global_ptr = new GlobalClass;
    global_ptr->set_value(1);
    cout << "foo: global_ptr is " << global_ptr->get_value() << '\n';
}

void bar(void)
{
    if (!global_ptr)
        global_ptr = new GlobalClass;
    global_ptr->set_value(2);
    cout << "bar: global_ptr is " << global_ptr->get_value() << '\n';
}

int main()
{
    if (!global_ptr)
        global_ptr = new GlobalClass;
    cout << "main: global_ptr is " << global_ptr->get_value() << '\n';
}

```

```
foo();  
bar();  
}
```

```
main: global_ptr is 0  
foo: global_ptr is 1  
bar: global_ptr is 2
```

output

After

Make the class responsible for its own global pointer and "initialization on first use" (by using a private static pointer and a public static accessor method). The client uses only the public accessor method.

```
class GlobalClass  
{  
    int m_value;  
    static GlobalClass *s_instance;  
    GlobalClass(int v = 0)  
    {  
        m_value = v;  
    }  
public:  
    int get_value()  
    {  
        return m_value;  
    }  
    void set_value(int v)
```

C++

```

    {
        m_value = v;
    }
    static GlobalClass *instance()
    {
        if (!s_instance)
            s_instance = new GlobalClass;
        return s_instance;
    }
};

// Allocating and initializing GlobalClass's
// static data member. The pointer is being
// allocated - not the object itself.
GlobalClass *GlobalClass::s_instance = 0;

void foo(void)
{
    GlobalClass::instance()->set_value(1);
    cout << "foo: global_ptr is " << GlobalClass::instance()->get_value() << '\n';
}

void bar(void)
{
    GlobalClass::instance()->set_value(2);
    cout << "bar: global_ptr is " << GlobalClass::instance()->get_value() << '\n';
}

int main()
{
    cout << "main: global_ptr is " << GlobalClass::instance()->get_value() << '\n';
    foo();
    bar();
}

```

```
main: global_ptr is 0
foo: global_ptr is 1
bar: global_ptr is 2
```

output

List of Singleton examples

C# examples

- [Singleton in C#](#)

C++ examples

- [Singleton in C++: Before and after <=\[You are here\]](#)
- [Singleton in C++](#)

Delphi examples

- [Singleton in Delphi](#)

Java examples

- [Singleton in Java](#)

PHP examples

- [Singleton in PHP](#)

◀ Prototype Design
Pattern

↑ Singleton

Structural patterns ▶



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivative Works 3.0 Unported License](#)

