Singleton

Links

- Stack Overflow: Shared pointers to a singleton do not recognize each other
- Stack Overflow: C++ Singleton design pattern
- Rainer Grimm: Thread-Safe Initialization of a Singleton
- RIP Tutorial: Lazy Initialization

Examples

Static Instance

A thread-safe method (guaranteed since C++11, see here) using a static instance of class CFoo (Meyers singleton):

```
class CFoo {
public:
  static CFoo& getInstance()
  {
    static CFoo instance;
    return instance;
  }
};
```

Dynamic, Using Smart Pointers

A thread-safe method using *C++11* smart pointers for creation of an singleton instance of class CFoo:

```
#include <mutex>  // std::mutex
#include <memory> // std::weak_ptr, std::shared_ptr

using std;

static weak_ptr<CFoo> s_pFoo; // singleton instance pointer
static mutex s_lock;          // singleton instance lock

shared_ptr<CFoo> CFoo::getInstance()
{
  lock_guard<mutex> lock(s_lock);
  shared_ptr<CFoo> pInstance = s_pFoo.lock();

  if (!pInstance)
  {
```

```
    pInstance = make_shared<CFoo>(); // create shared smart pointer
    s_pFoo = pInstance;              // replace (set) managed object
  } // if

  return pInstance;
} // getInstance()
```

Dynamic, Using DCLP

See Jeff Preshing: Double-Checked Locking is Fixed In C++11.

From:
https://wiki.rho62.de/ - **rho62 Wiki**

Permanent link:
https://wiki.rho62.de/doku.php?id=programming:cpp:singleton

Last update: **2023/10/13 07:14**