

There is code to use automatic locking for shared object among threads

```
#ifndef _shared_shared_hpp_
#define _shared_shared_hpp_

#include <Core/Core.h>

namespace Upp
{

template<typename T>
struct Shared
{
    struct Locker
    {
        Locker(T& value, Mutex& mutex) : m_value(value), m_lock(mutex) {}
        T* operator->()
        {
            return &m_value;
        }
        T& operator*()
        {
            return m_value;
        }
        T* operator&()
        {
            return &m_value;
        }
    }

private:
    T& m_value;
    Mutex::Lock m_lock;
};

static Locker get()
{
    return Locker(Single<T>(), mutex);
}

private:
    static Mutex mutex;
};

template<typename T>
```

```

Mutex Shared<T>::mutex;

template<typename T>
struct RWShared
{
    struct ReadLocker
    {
        ReadLocker(const T& value, RWMutex& mutex) : m_value(value), m_lock(mutex) {}
        const T* operator->()
        {
            return &m_value;
        }
        const T& operator*()
        {
            return m_value;
        }
        const T* operator&()
        {
            return &m_value;
        }
    }

private:
    const T& m_value;
    RWMutex::ReadLock m_lock;
};

struct WriteLocker
{
    WriteLocker(T& value, RWMutex& mutex) : m_value(value), m_lock(mutex) {}
    T* operator->()
    {
        return &m_value;
    }
    T& operator*()
    {
        return m_value;
    }
    T* operator&()
    {
        return &m_value;
    }
}

private:
    T& m_value;
    RWMutex::WriteLock m_lock;
};

static ReadLocker read()

```

```
{  
    return ReadLocker(Single<T>(), mutex);  
}
```

```
static WriteLocker write()  
{  
    return WriteLocker(Single<T>(), mutex);  
}
```

```
private:  
    static RWMutex mutex;  
};
```

```
template<typename T>  
RWMutex RWShared<T>::mutex;  
  
}
```

```
#endif
```

example usage:

```
struct A  
{  
    int a;  
    int b;  
  
    int getSum() const  
    {  
        return a + b;  
    }  
};  
  
...  
RWShared<A>::write()->a = 2;  
RWShared<A>::write()->b = 3;  
Cout() << RWShared<A>::read()->getSum() << "\n";
```

Subject: Re: Shared object with auto locking strategy
Posted by [captainc](#) on Sun, 10 Aug 2008 14:32:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

Very cool, definitely will use this. Thank you.
