The C++ ScopedArray Template

`IceUtil::ScopedArray` is a smart pointer class similar to `Handle`. However, instead of managing the memory for class instances, `ScopedArray` manages memory for an array. This class is provided mainly for use with the stream API. However, you can use it with arrays for other purposes.

Here is the definition of the template in full:

C++

```
template<typename T>
class ScopedArray : private IceUtil::noncopyable
{
public:
    explicit ScopedArray(T* ptr = 0)
        : _ptr(ptr) { }

    ScopedArray(const ScopedArray& other) {
        _ptr = other._ptr;
        const_cast<ScopedArray&>(other)._ptr = 0;
    }

    ~ScopedArray() {
        if (_ptr != 0)
            delete[] _ptr;
    }

    void reset(T* ptr = 0) {
        assert(ptr == 0 || ptr != _ptr);
        if (_ptr != 0)
            delete[] _ptr;
        _ptr = ptr;
    }

    T& operator[](size_t i) const {
        assert(_ptr != 0);
        assert(i >= 0);
        return _ptr[i];
    }

    T* get() const {
        return _ptr;
    }

    void swap(ScopedArray& a) {
        T* tmp = a._ptr;
        a._ptr = _ptr;
        _ptr = tmp;
    }

private:
    T* _ptr;
};
```

The class allows you to allocate an array on the heap and assign its pointer to a `ScopedArray` instance. When the instance goes out of scope, it calls `delete[]` on the array, so you do not need to deallocate the array explicitly yourself. This greatly reduces the risk of a memory leak due to an early return or uncaught exception.

See Also

- C++ Streaming Interfaces

Copyright © 2017, ZeroC, Inc.