Proxy 패턴

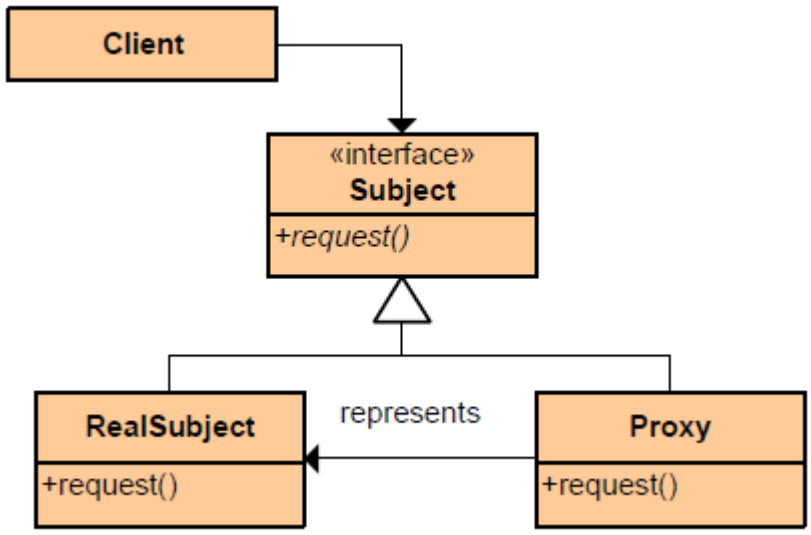
타입

Structural Pattern

문제

해결

클래스 다이어그램



예제

proxy.cpp

```
#include <iostream>

using namespace std;

class RealImage
{
    int m_id;
public:
    RealImage(int i)
    {
        m_id = i;
```

```
        cout << "   $$ ctor: " << m_id << '\n';
    }
    ~RealImage()
    {
        cout << "   dtor: " << m_id << '\n';
    }
    void draw()
    {
        cout << "   drawing image " << m_id << '\n';
    }
};

// 1. Design an "extra level of indirection" wrapper class
class Image  // Proxy
{
    // 2. The wrapper class holds a pointer to the real class
    RealImage *m_the_real_thing;
    int m_id;
    static int s_next;
public:
    Image()
    {
        m_id = s_next++;
        // 3. Initialized to null
        m_the_real_thing = 0;
    }
    ~Image()
    {
        delete m_the_real_thing;
    }
    void draw()
    {
        // 4. When a request comes in, the real object is
        //    created "on first use"
        if (!m_the_real_thing)
            m_the_real_thing = new RealImage(m_id);
        // 5. The request is always delegated
        m_the_real_thing->draw();
    }
};
int Image::s_next = 1;

int main()
{
    Image images[5];

    for (int i; true;)
    {
        cout << "Exit[0], Image[1-5]: ";
        cin >> i;
        if (i == 0)
```

```
                break;
            images[i - 1].draw();
        }
    }
```

참고

http://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns#Proxy