组合模式

定义：将对象组合成树形结构以表示"部分整体”的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性。

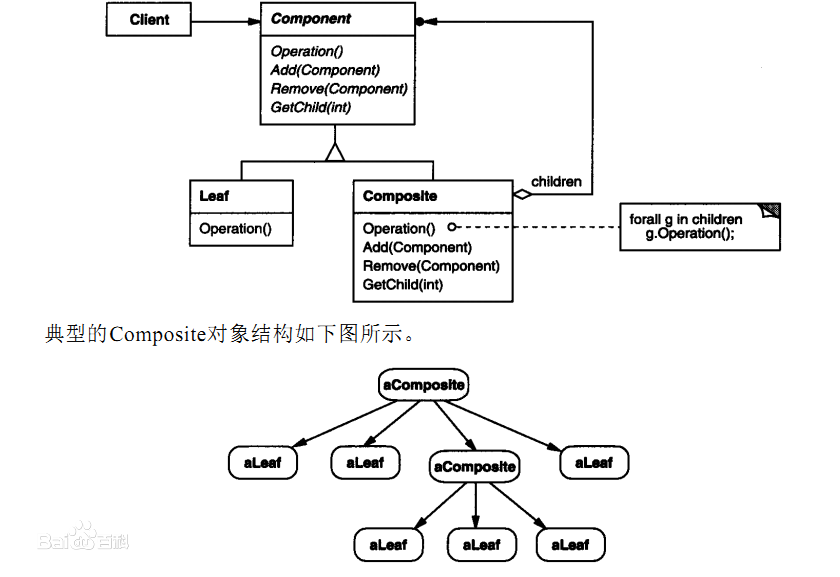
组合模式使得用户对单个对象和组合对象的使用具有一致性。有时候又叫做部分-整体模式，它使我们树型结构的问题中，模糊了简单元素和复杂元素的概念，客户程序可以像处理简单元素一样来处理复杂元素,从而使得客户程序与复杂元素的内部结构解耦。

涉及角色：

1. Component 是组合中的对象声明接口，在适当的情况下，实现所有类共有接口的默认行为。声明一个接口用于访问和管理Component子部件。
2. Leaf 在组合中表示叶子结点对象，叶子结点没有子结点。
3. Composite定义有枝节点行为，用来存储子部件，在Component接口中实现与子部件有关操作，如增加(add)和删除(remove)等。

适用性：

1. 你想表示对象的部分-整体层次结构。
2. 你希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。



实现代码（C++）：

#include <string>

class Menu

{

public:

virtual ~Menu();

virtual void Add(Menu\*);

virtual void Remove(Menu\*);

virtual Menu\* GetChild(int);

virtual void Display() = 0;

protected:

Menu();

Menu(std::string);

std::string m\_strName;

};

//Menu.cpp

#include "stdafx.h"

#include "Menu.h"

Menu::Menu()

{

}

Menu::Menu(std::string strName) : m\_strName(strName)

{

}

Menu::~Menu()

{

}

void Menu::Add(Menu\* pMenu)

{}

void Menu::Remove(Menu\* pMenu)

{}

Menu\* Menu::GetChild(int index)

{

return NULL;

}

//SubMenu.h

#include "Menu.h"

class SubMenu : public Menu

{

public:

SubMenu();

SubMenu(std::string);

virtual ~SubMenu();

void Display();

};

//SubMenu.cpp

#include "stdafx.h"

#include "SubMenu.h"

#include <iostream>

using namespace std;

SubMenu::SubMenu()

{

}

SubMenu::SubMenu(string strName) : Menu(strName)

{

}

SubMenu::~SubMenu()

{

}

void SubMenu::Display()

{

cout << m\_strName << endl;

}

//CompositMenu.h

#include "Menu.h"

#include <vector>

class CompositMenu : public Menu

{

public:

CompositMenu();

CompositMenu(std::string);

virtual ~CompositMenu();

void Add(Menu\*);

void Remove(Menu\*);

Menu\* GetChild(int);

void Display();

private:

std::vector<Menu\*> m\_vMenu;

};

//CompositMenu.cpp

#include "stdafx.h"

#include "CompositMenu.h"

#include <iostream>

using namespace std;

CompositMenu::CompositMenu()

{

}

CompositMenu::CompositMenu(string strName) : Menu(strName)

{

}

CompositMenu::~CompositMenu()

{

}

void CompositMenu::Add(Menu\* pMenu)

{

m\_vMenu.push\_back(pMenu);

}

void CompositMenu::Remove(Menu\* pMenu)

{

m\_vMenu.erase(&pMenu);

}

Menu\* CompositMenu::GetChild(int index)

{

return m\_vMenu[index];

}

void CompositMenu::Display()

{

cout << "+" << m\_strName << endl;

vector<Menu\*>::iterator it = m\_vMenu.begin();

for (; it != m\_vMenu.end(); ++it)

{

cout << "|-";

(\*it)->Display();

}

}

#include "stdafx.h"

#include "Menu.h"

#include "SubMenu.h"

#include "CompositMenu.h"

int main(int argc, char\* argv[])

{

Menu\* pMenu = new CompositMenu("国内新闻");

pMenu->Add(new SubMenu("时事新闻"));

pMenu->Add(new SubMenu("社会新闻"));

pMenu->Display();

pMenu = new CompositMenu("国际新闻");

pMenu->Add(new SubMenu("国际要闻"));

pMenu->Add(new SubMenu("环球视野"));

pMenu->Display();

return 0;

}