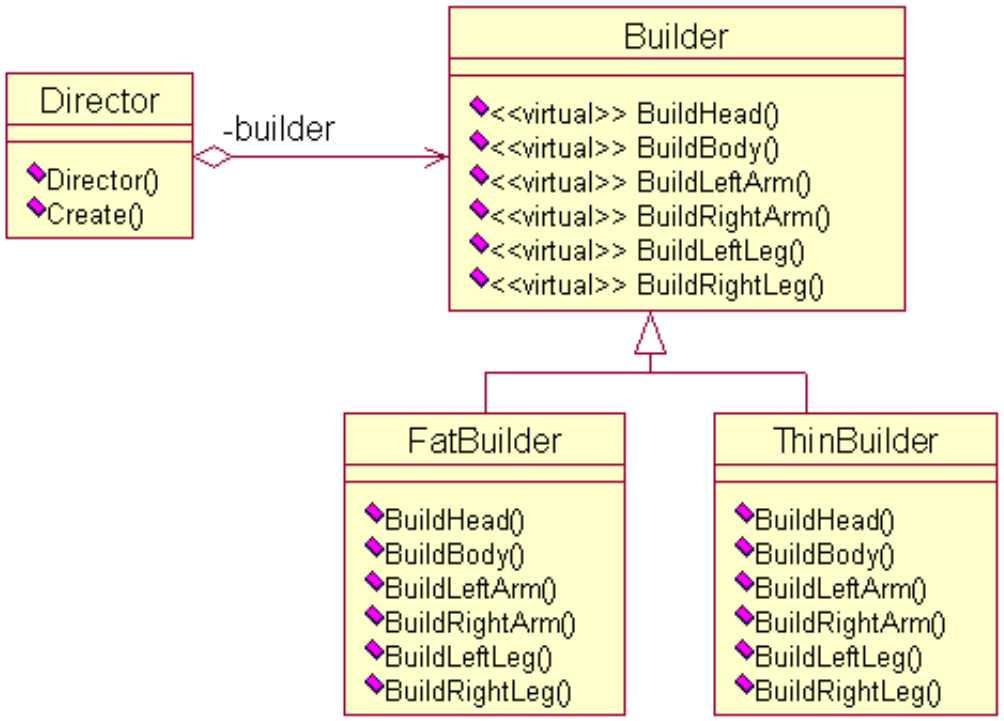
設計模式C++實現（6）——建造者模式

星期六, 2013 12月 14, 1:04 上午

軟件領域中的設計模式為開發人員提供了一種使用專家設計經驗的有效途徑。設計模式中運用了面向對象編程語言的重要特性：封裝、繼承、多態，真正領悟設計模式的精髓是可能一個漫長的過程，需要大量實踐經驗的積累。最近看設計模式的書，對於每個模式，用C++寫了個小例子，加深一下理解。主要參考《大話設計模式》和《設計模式:可複用面向對象軟件的基礎》（DP）兩本書。本文介紹建造者模式的實現。

建造者模式的定義將一個複雜對象的構建與它的表示分離，使得同樣的構建過程可以創建不同的表示（DP）。《大話設計模式》舉了一個很好的例子——建造小人，一共需建造6個部分，頭部、身體、左右手、左右腳。與工廠模式不同，建造者模式是在導向者的控制下一步一步構造產品的。建造小人就是在控制下一步步構造出來的。創建者模式可以能更精細的控制構建過程，從而能更精細的控制所得產品的內部結構。下面給出建造者模式的UML圖，以建造小人為實例。



對於客戶來說，只需知道導向者就可以了，通過導向者，客戶就能構造複雜的對象，而不需要知道具體的構造過程。下面給出小人例子的代碼實現。

```
class  Builder
{
public :
    virtual  void  BuildHead() {}
    virtual  void  BuildBody() {}
    virtual  void  BuildLeftArm(){}
    virtual  void  BuildRightArm() {}
    virtual  void  BuildLeftLeg() {}
    virtual  void  BuildRightLeg() {}
};
//構造瘦人
```

```
class ThinBuilder : public Builder
{
public :
    void BuildHead() { cout<< "build thin body" <<endl; }
    void BuildBody() { cout<< "build thin head" <<endl; }
    void BuildLeftArm() { cout<< "build thin leftarm" <<endl; }
    void BuildRightArm() { cout<< "build thin rightarm" <<endl; }
    void BuildLeftLeg() { cout<< "build thin leftleg" <<endl; }
    void BuildRightLeg() { cout<< "build thin rightleg" <<endl; }
};
//構造胖人
class FatBuilder : public Builder
{
public :
    void BuildHead() { cout<< "build fat body" <<endl; }
    void BuildBody() { cout<< "build fat head" <<endl; }
    void BuildLeftArm() { cout<< "build fat leftarm" <<endl; }
    void BuildRightArm() { cout<< "build fat rightarm" <<endl; }
    void BuildLeftLeg() { cout<< "build fat leftleg" <<endl; }
    void BuildRightLeg() { cout<< "build fat rightleg" <<endl; }
};
//構造的指揮官
class Director
{
private :
    Builder *m_pBuilder;
public :
    Director(Builder *builder) { m_pBuilder = builder; }
    void Create(){
        m_pBuilder->BuildHead();
        m_pBuilder->BuildBody();
        m_pBuilder->BuildLeftArm();
        m_pBuilder->BuildRightArm();
        m_pBuilder->BuildLeftLeg();
        m_pBuilder->BuildRightLeg();
    }
};
```

客戶的使用方式：

```
int main()
{
    FatBuilder thin;
    Director director(&thin);
    director.Create();
    return 0;
}
```

至此，《設計模式:可複用面向對象軟件的基礎》一書上的5種創建型模式介紹完了，分別是工廠方法模式、抽象工廠模式、原型模式、建造者模式、單例模式。

[設計模式C++實現（1）——工廠模式]

設計模式C++實現（４）——單例模式

設計模式C++實現（５）——原型模式、模板方法模式