http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 #include <iostream>

2 #include <string>

3

4 using namespace std;

5

6 //管理者类

7 class Manager

8 {

9 private:

10 string name;

11 Manager \*nextManager;

12 public:

13 Manager(string n,Manager \*m = NULL) : name(n),nextManager(m){};

14 void set\_name(string name)

15 {

16 this->name = name;

17 }

18 string get\_name()

19 {

20 return this->name;

21 }

22 //设置下一个要处理请假信息的管理者

23 void set\_next\_manager(Manager \*m)

24 {

25 this->nextManager = m;

26 }

27 Manager\* get\_next\_manager()

28 {

29 return nextManager;

30 }

31 //处理请假

32 virtual void deal\_holiday(int day){};

33 };

34 //主管类

35 class ZhuGuan : public Manager

36 {

37 public:

38 ZhuGuan(string n,Manager \*m = NULL):Manager(n,m){};

39 virtual void deal\_holiday(int day)

40 {

41 if(day <= 0.5)

42 {

43 cout<<"主管："<<get\_name()<<"同意请假!!"<<endl;

44 }

45 else if(get\_next\_manager() != NULL)

46 {

47 get\_next\_manager()->deal\_holiday(day);

48 }

49 }

50 };

51 //人事部门类

52 class RenShi : public Manager

53 {

54 public:

55 RenShi(string n,Manager \*m = NULL):Manager(n,m){};

56 virtual void deal\_holiday(int day)

57 {

58 if(day < 2)

59 {

60 cout<<"人事部："<<get\_name()<<"同意请假!!"<<endl;

61 }

62 else if(get\_next\_manager() != NULL)

63 {

64 get\_next\_manager()->deal\_holiday(day);

65 }

66 }

67 };

68 class ZongJingLi : public Manager

69 {

70 public:

71 ZongJingLi(string n,Manager \*m = NULL):Manager(n,m){};

72 virtual void deal\_holiday(int day)

73 {

74 if(day < 30)

75 {

76 cout<<"总经理："<<get\_name()<<"同意请假!!"<<endl;

77 }

78 //总经理室最后一个进行处理的，所以一定要给出一个答复

79 else

80 {

81 cout<<"总经理："<<get\_name()<<"关于你说的请假的事情，由于时间太长，以后再说吧!!"<<endl;

82 }

83 }

84 };

85 int main()

86 {

87 //主管Mike

88 ZhuGuan Z\_G("Mike");

89 //人事部Tom

90 RenShi R\_S("Tom");

91 //总经理Vincent

92 ZongJingLi Z\_J\_L("Vincent");

93

94 Z\_G.set\_next\_manager(&R\_S);

95 R\_S.set\_next\_manager(&Z\_J\_L);

96

97 Z\_G.deal\_holiday(0.5);

98 Z\_G.deal\_holiday(1);

99 Z\_G.deal\_holiday(3);

100 Z\_G.deal\_holiday(31);

101 return 0;

102 }

[复制代码](javascript:void(0);)

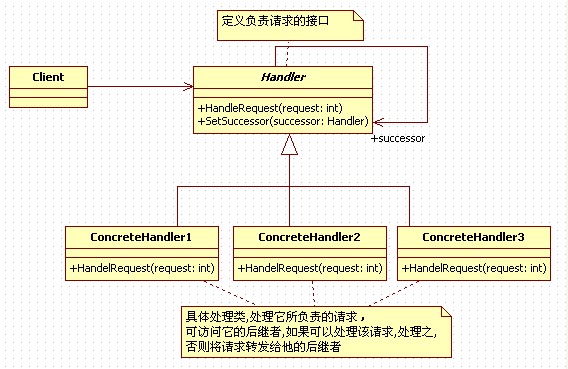
看上面的代码，我们不难发现，要完成系统的扩展时很方便，只需要在增加管理者类，并且这个"职责链”是在客户端动态确定的哦。当然了如果您现在已经是部门经理了，那么可能您可以直接向总经理请假，这也是可以的。

**二、再看职责链模式**

首先看看GoF怎么说：

**职责链模式（Chain Of Responsibility）：**使多个对象都有机会处理请求，从而避免请求的发送者和接受者之间的耦合关系。将这个对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理他为止。

说的通俗一点就是：职责链模式就是把让请求沿着链条一直传下去，知道有一个对象能够处理这个请求。在职责链上的对象都有机会处理到请求，但是只有前一个职责链不能完成请求的处理，才将请求传递给下一个对象。可以看出职责链上的对象到底谁能完成请求的处理是不确定的。并且需要注意的是，职责链是在客户端动态确定的哦，具体职责链上对象的顺序也是动态确定的。还是看看结构图：



**三、职责链模式和状态模式**

我的初中生活大概是这样的：上午七点起床，八点之前洗脸刷牙吃早饭，然后去上课到十一点四十，中午吃饭，然后午睡，下午一点半继续上课，五点十分下课，然后吃晚饭，从六点二十到九点是晚自习时间，九点之后放学回家，洗洗睡了。

初中的时候请假是这样的流程：首先是可以找班长请假，班长批准请半天，超过半天班长向老师申请，如果请假时间超过一周，老师要跟副年级主任请示，如果请假超出一个月，主任要跟年级正主任请示，然后被批准，或不被批准。

实际上第一种情况就是适合状态模式的，而第二种情况适合职责链模式。为什么呢？？ 状态模式中每个对象之间的变换是在程序编译的时候确定的，也就是编译的时候每个对象就已经知道自己的下一个状态了，这适合每个状态的后继状态基本是比较稳 定不会发生变化的情况，比如，初中生活就是那样，每天都是，而职责链模式呢？链上的每个对象是在客户端动态确定的，可能每次都不一样，为什么请假就适合职 责链模式呢？？就是因为职责是动态确定的，如果某天班主任家里有点事，没来，按照职责链模式可以把班长的在职责链上的下一对象设置为副年级主任。这样其他 学生还可以正常请假。而如果按照状态模式的话，如果一旦班主任没来，学生就不能请超过半天的假啦。。。这不太好吧~~~个人感觉状态模式和职责链模式的区 别就是这样的。

**四、使用职责链模式的场合和优缺点**

**使用职责链模式的场合：**

个人觉得，职责链模式主要用于当一个请求有多种处理方式的时候，并且具体处理方式不确定的情况。

**使用职责链模式的优点：**

1.增强了系统的可扩展性。

2.使用职责链模式可以避免众多的if或者if-else语句。

3.使用职责链模式可以减小对象之间的耦合性。使得对象之间的联系减小。

4.可以根据需要自由组合工作流程。如工作流程发生变化，可以通过重新分配对象链便可适应新的工作流程。

5.责任的分担。每个类只需要处理自己该处理的工作（不该处理的传递给下一个对象完成），明确各类的责任范围，符合类的最小封装原则。

**使用职责链模式的缺点：**

1.使用职责链模式的时候会有多个处理者对象，但是实际使用的处理者对象却只有一个，这在某种程度讲是资源的浪费。

2.同时职责链的建立的合理性要靠客户端来保证，增加了程序的复杂性，也有可能由于职责链导致出错。