职责链模式

一、概述

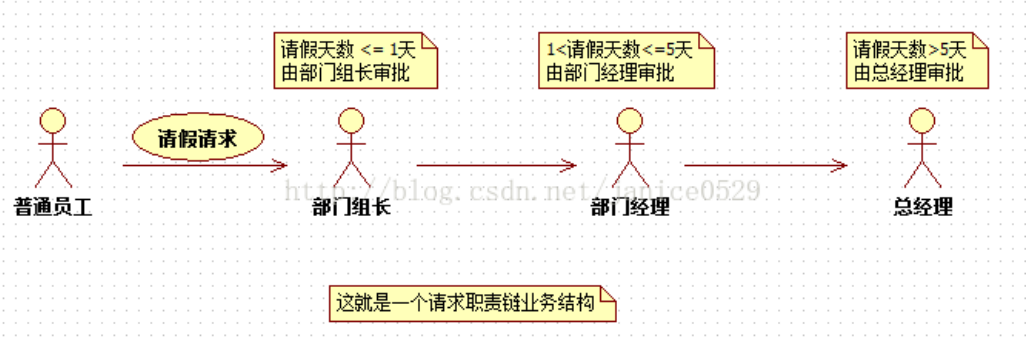
避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。职责链模式是一种对象行为型模式。

核心在于引入一个抽象处理者类

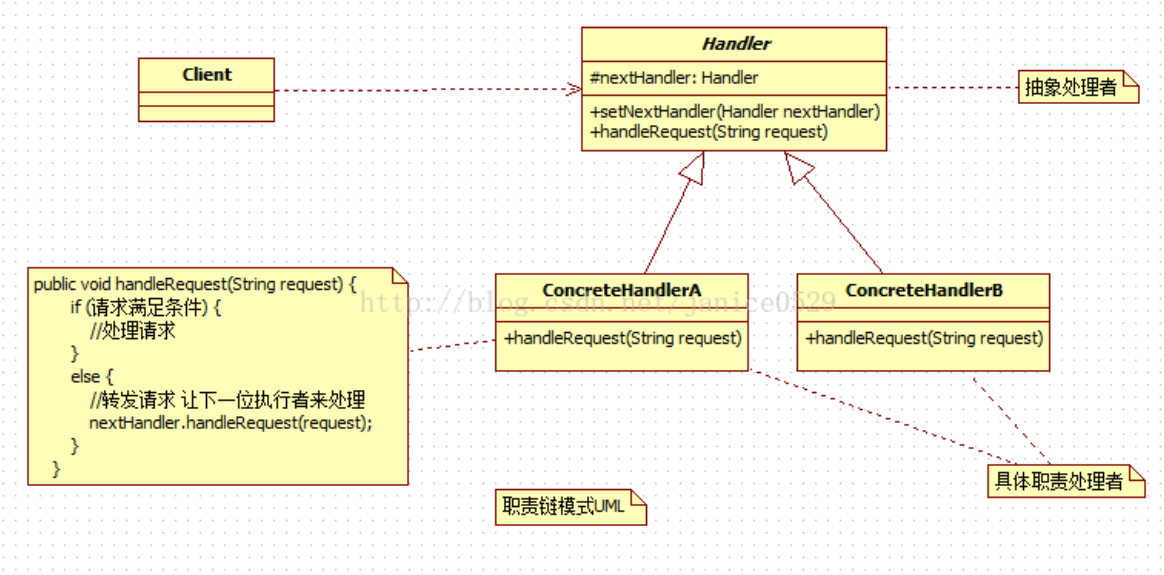
二、适用场景

请求的链式处理，多个对象可以处理同一请求、但是具体由哪个对象来处理由运行时系统根据条件判断确定。

如请假业务场景：



三、UML类图



四、参与者

1、Handler（抽象处理者）：它定义了一个处理请求的接口，一般设计为抽象类，由于不同的具体处理者处理请求的方式不同，因此在其中定义了抽象请求处理方法。因为每一个处理者的下家还是一个处理者，因此在抽象处理者中定义了一个抽象处理者类型的对象（如结构图中的nextHandler），作为其对下家的引用。通过该引用，处理者可以连成一条链。

2、ConcreteHandler（具体处理者）：它是抽象处理者的子类，可以处理用户请求，在具体

处理者类中实现了抽象处理者中定义的抽象请求处理方法，在处理请求之前需要进行判断，看是否有相应的处理权限，如果可以处理请求就处理它，否则将请求转发给后继者；在具体处理者中可以访问链中下一个对象，以便请求的转发。

注意：职责链模式并不创建职责链，职责链的创建工作必须由系统的其他部分来完成，一般是在使用该职责链的客户端中创建职责链。

五、用例学习

1、抽象处理者类：Handler.Java

[java] view plain copy

```java
/**
 * 抽象处理者类
 * @author  lvzb.software@qq.com
 *
 */
public abstract class Handler {

    protected Handler nextHandler;

    public void setNextHandler(Handler nextHandler){
        this.nextHandler = nextHandler;
    }

    public abstract void handleRequest(int request);

}
```

2、具体处理者类A：部门组长 ConcreteHandlerA.java

[java] view plain copy

```java
/**
 * 具体职责处理者A：案例中的 部门组长角色
 * @author  lvzb.software@qq.com
 *
 */
public class ConcreteHandlerA extends Handler {

    @Override
    public void handleRequest(int leaveDay) {
        if(leaveDay <= 1){
            // 满足处理条件  处理请求
            System.out.println("请假天数小于2天  由部门组长审批");
        } else {
            nextHandler.handleRequest(leaveDay);
        }

    }

}
```

3、具体处理者类B：部门经理 ConcreteHandlerB.java

[java] view plain copy

```java
/**
 * 具体职责处理者B：案例中的 部门经理角色
 * @author  lvzb.software@qq.com
 *
 */
public class ConcreteHandlerB extends Handler {

    @Override
    public void handleRequest(int leaveDay) {
        if(1 < leaveDay && leaveDay <= 5){
            // 满足处理条件  处理请求
            System.out.println("请假天数大于1天且小于等于5天  由部门经理审批");
        } else {
            nextHandler.handleRequest(leaveDay);
        }

    }

}
```

4、具体处理者类C：总经理 ConcreteHandlerC.java

[java] view plain copy

```java
/**
 * 具体职责处理者C：案例中的 总经理角色
 * @author  lvzb.software@qq.com
 *
 */
public class ConcreteHandlerC extends Handler {

    @Override
    public void handleRequest(int leaveDay) {
        if(leaveDay > 5){
            System.out.println("当请假天数大于5天的情况下 由总经理亲自操刀审批。总经理的职责已经是最大的啦，还有他没有权限处理的事吗？");
        }
    }

}
```

5、客户端类：Client.java

[java] view plain copy

```java
/**
 * 客户端类<br/>
 * 负责创建职责链和发送请求<br/>
 * 当职责链（职责的传递顺序/请求的处理顺序）建好之后，客户端只负责将请求发送出去，
 * 而具体请求在职责链上的传递和最终由链上的哪个对象进行处理不由客户端关心
 * @author  lvzb.software@qq.com
 *
 */
public class Client {

    public static void main(String[] args) {
        Handler handlerA,handlerB,handlerC;
        handlerA = new ConcreteHandlerA();
        handlerB = new ConcreteHandlerB();
```

```
15.         handlerC = new ConcreteHandlerC();
16.
17.         // 创建职责链  handlerA ——> handlerB ——> handlerC
18.         handlerA.setNextHandler(handlerB);
19.         handlerB.setNextHandler(handlerC);
20.
21.         // 发送请假请求一
22.         handlerA.handleRequest(1);
23.
24.         // 发送请假请求二
25.         handlerA.handleRequest(7);
26.
27.         // 发送请假请求二
28.         handlerA.handleRequest(3);
29.     }
30.
31. }
```

6、运行结果：

[plain] view plain copy

```
1. 请假天数小于2天   由部门组长审批
2. 当请假天数大于5天的情况下 由总经理亲自操刀审批。总经理的职责已经是最大的啦，还有他没有权限处理的事吗？
3. 请假天数大于1天且小于等于5天   由部门经理审批
```