链滴

过滤器模式

作者：614756773

原文链接：https://ld246.com/article/1583834577563

来源网站：链滴

解释

过滤器模式有两大特点：

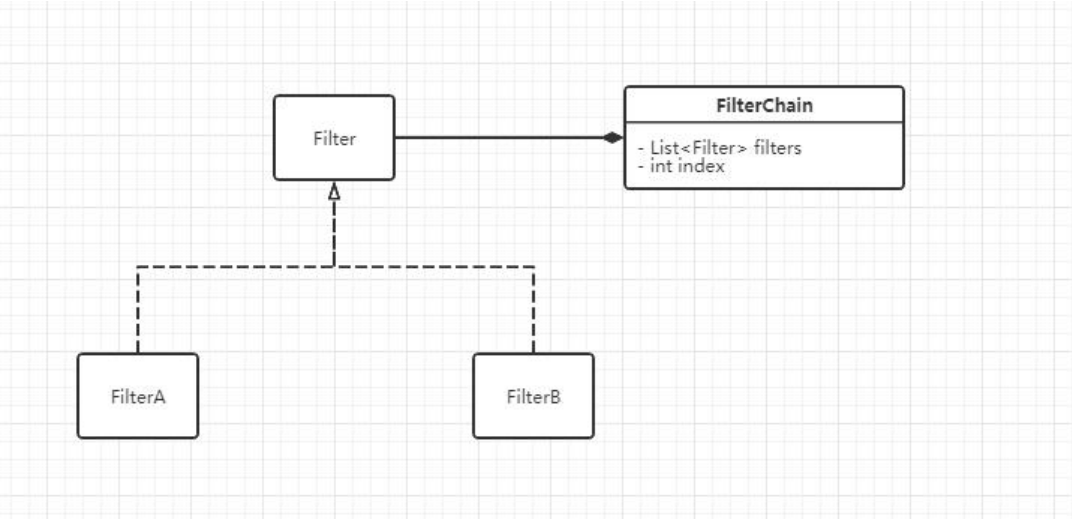
- 自由组合过滤器
- 递归

过滤器模式的的最常见的一个例子就是servlet中的过滤器，我在实现自己的aop功能时，发现环绕织入太好解决，使用过滤器模式的递归特点刚好能够解决这个问题

使用要点

- 定义Filter
- 定义FilterChain
  - 持有Filter列表
  - 持有一个计数器，用于标志在递归的过程中什么时候该返回

类图



Code

FilterChain

```
public class FilterChain {
    private List<Filter> filterList;

    private int index;

    public FilterChain(List<Filter> filterList) {
        this.filterList = filterList;
    }
```

```
    public Object execute() {
        if (index >= filterList.size()) {
            System.out.println("处理完毕");
            return null;
        }
        return filterList.get(index++).doFilter(this);
    }

    public static void main(String[] args) {
        List<Filter> filters = new ArrayList<>(2);
        filters.add(new FilterA());
        filters.add(new FilterB());

        Object execute = new FilterChain(filters).execute();
    }
}
```

Filter

```
public interface Filter {
    Object doFilter(FilterChain filterChain);
}
```

Filter实现类

```
public class FilterA implements Filter {
    @Override
    public Object doFilter(FilterChain filterChain) {
        System.out.println("A处理前");
        Object result = filterChain.execute();
        System.out.println("A处理后");
        return result;
    }
}
public class FilterB implements Filter {
    @Override
    public Object doFilter(FilterChain filterChain) {
        System.out.println("B处理前");
        Object result = filterChain.execute();
        System.out.println("B处理后");
        return result;
    }
}
```