晚上：19:30进行测试 20:00正式开始

**网络环境**

建议先把音量调到最大

调整屏幕布局

客户端菜单 – 布局

调成比1024\*768更大的分辨率

**定期讲课**

<http://www.bjsxt.com>

ChainOfResponsibility

**假设**

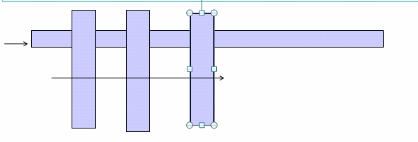
1. 初步具备面向对象的设计思维
   1. 网络课程第二课
   2. 视频第三章
2. 了解多态的概念
   1. 网络课程第二课
   2. 视频第三章

**真实案例**

在网上发表信息

* 需要检查信息

责任传递：



//接口定义

package com.test.designpattern.\_007;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public interface Filter {  
 void doFilter(String msg);  
}

//筛选规则实现接口

package com.test.designpattern.\_007;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class HtmlFilter implements Filter {  
 public void doFilter(String msg){  
  
 }  
}

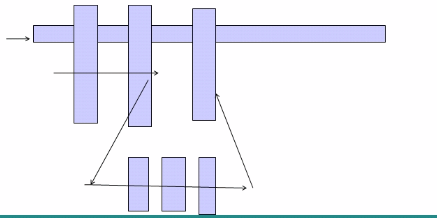
package com.test.designpattern.\_007;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class SentiveFilter implements Filter {  
 public void doFilter(String msg){  
  
 }  
}

//处理信息

package com.test.designpattern.\_007;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class Message {  
 private String msg;  
  
 private Filter[] filters={new HtmlFilter(),new SentiveFilter()};  
  
 public Message(){}  
  
 public void process(){  
 for(Filter filter:filters){  
 filter.doFilter(this.msg);  
 }  
 }  
  
 public String getMsg() {  
 return msg;  
 }  
  
 public void setMsg(String msg) {  
 this.msg = msg;  
 }  
}

这样，当我们增加过滤处理规则的时候，只需要定义一个新类并实现过滤接口的方法，然后添加到过滤数组中就可以了，非常方便，这样还可以排序过滤规则的处理顺序。

Java web的doFilter中就利用了这些



如果插入新的处理序列？

package com.test.designpattern.\_007;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class FilterChain {  
 private List<Filter> filters=new ArrayList<Filter>();  
  
 public FilterChain addFilter(Filter filter){

this.filters.add(filter);

return this; //这里使用了链条式

}  
 public void doFilter(String msg){  
 for(Filter filter:filters){  
 filter.doFilter(msg);  
 }  
 }  
}

package com.test.designpattern.\_007;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class Message2 {  
 private String msg;  
  
 private FilterChain filterChain;  
  
 public Message2() {  
 }  
  
 public void process() {  
 this.filterChain.doFilter(this.msg);  
 }  
  
 public String getMsg() {  
 return msg;  
 }  
  
 public void setMsg(String msg) {  
 this.msg = msg;  
 }  
  
 public FilterChain getFilterChain() {  
 return filterChain;  
 }  
  
 public void setFilterChain(FilterChain filterChain) {  
 this.filterChain = filterChain;  
 }  
}

在调用的时候可以使用链式添加过滤来初始化msg的filterchain，将所有的filter都交给filterchain来处理，之前是由msg来处理

另外FilterChain也可以实现Filter接口，将FilterChain看做一个大的Filter

模拟tomcat的filter

涉及的类/接口 Request Response Filter HtmlFilter SentiveFilter FilterChain

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class Request {  
 private String reqStr;  
  
 public String getReqStr() {  
 return reqStr;  
 }  
  
 public void setReqStr(String reqStr) {  
 this.reqStr = reqStr;  
 }  
}

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class Response {  
 private String respStr;  
  
 public String getRespStr() {  
 return respStr;  
 }  
  
 public void setRespStr(String respStr) {  
 this.respStr = respStr;  
 }  
}

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public interface Filter { #Filter接口  
 void doFilter(Request req, Response resp, FilterChain filterChain);  
}

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class HtmlFilter implements Filter{ #实现接口  
 public void doFilter(Request req, Response resp, FilterChain filterChain){  
 req.setReqStr(req.getReqStr().replace("<","[").replace(">","]")+"---Html Filter"); #处理请求  
 filterChain.doFilter(req,resp,filterChain); #链式处理  
 resp.setRespStr(resp.getRespStr()+"--Html Filter"); #处理响应  
 }  
}

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class SentiveFilter implements Filter { #实现接口  
 public void doFilter(Request req, Response resp, FilterChain filterChain){  
 req.setReqStr(req.getReqStr().replace("敏感","(mingan)").replace("特殊","(teshu)")+"---Sentive Filter");  
 filterChain.doFilter(req,resp,filterChain); #实现链式处理  
 resp.setRespStr(resp.getRespStr()+"--Sentive Filter");  
 }  
}

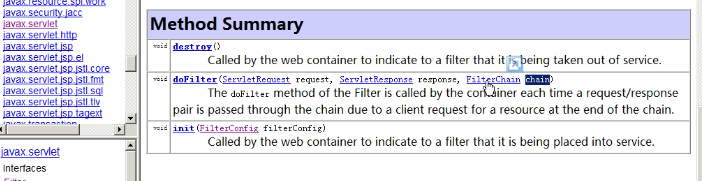
package com.test.designpattern.\_007.webfilter;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class FilterChain implements Filter { #实现接口并作为链式主要处理链  
 private List<Filter> filters;  
 private int index; #用来控制当前由哪个filter来执行  
  
 public void addFilter(Filter filter){  
 if(this.filters==null){  
 this.filters=new ArrayList<Filter>();  
 }  
  
 this.filters.add(filter);  
 }  
  
 public void doFilter(Request req, Response resp, FilterChain filterChain){  
 if(filters==null || index==filters.size()){  
 return;  
 }  
  
 Filter filter=filters.get(index++);  
 filter.doFilter(req,resp,filterChain);  
 }  
}

调用：

package com.test.designpattern.\_007.webfilter;  
  
/\*\*  
 \* Created by DaiYan on 2017/9/14.  
 \*/  
public class Main {  
 public static void main(String[] args){  
 Request request=new Request();  
 request.setReqStr("这<script>并非敏感，也并非特殊，仅仅是个测试");  
 Response response=new Response();  
  
 response.setRespStr("这<script>并非敏感，也并非特殊，仅仅是个测试");  
 FilterChain filterChain=new FilterChain();  
 filterChain.addFilter(new HtmlFilter());  
 filterChain.addFilter(new SentiveFilter());  
  
 filterChain.doFilter(request,response,filterChain);  
 System.out.println(request.getReqStr());  
 System.out.println(response.getRespStr());  
 }  
}

对request的处理顺序和对response的处理顺序正好是相反的

Java web中的Filter实现：



拦截器 责任链条 过滤器