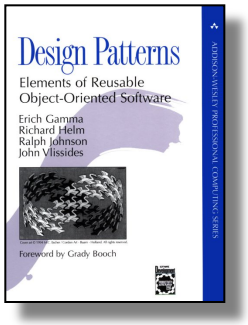
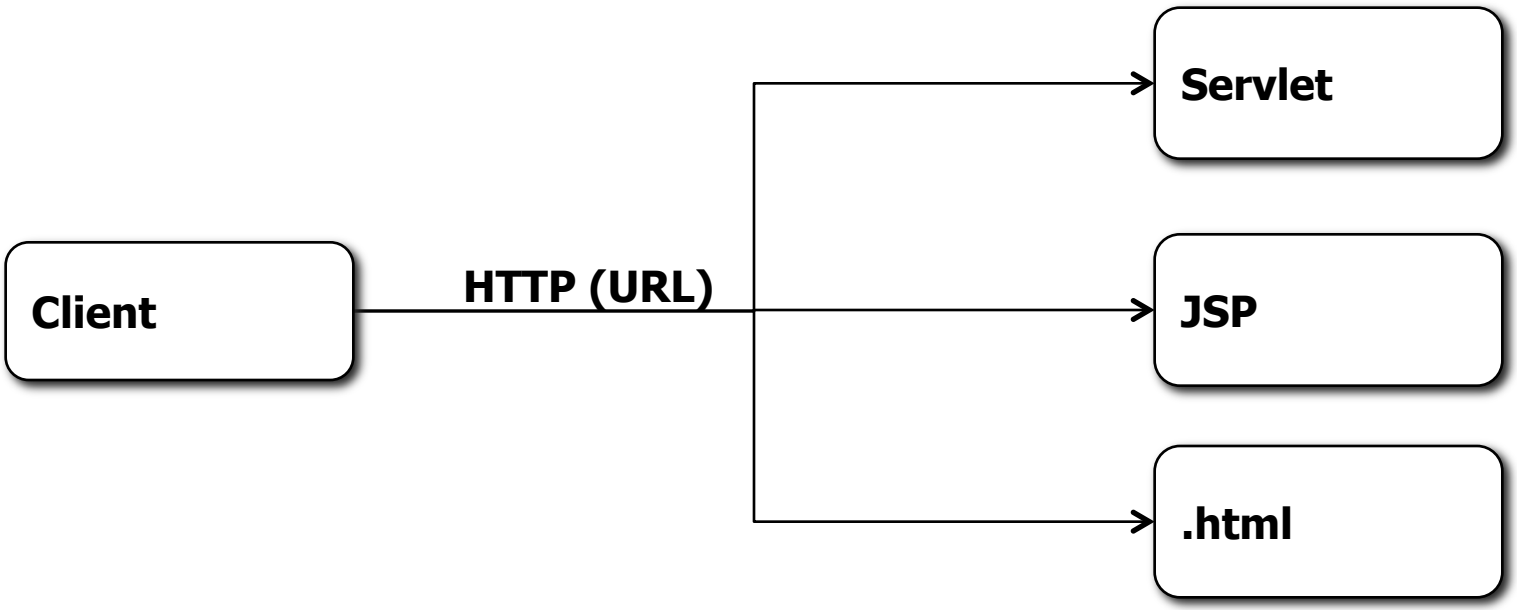


Architecture des Systèmes d'Information

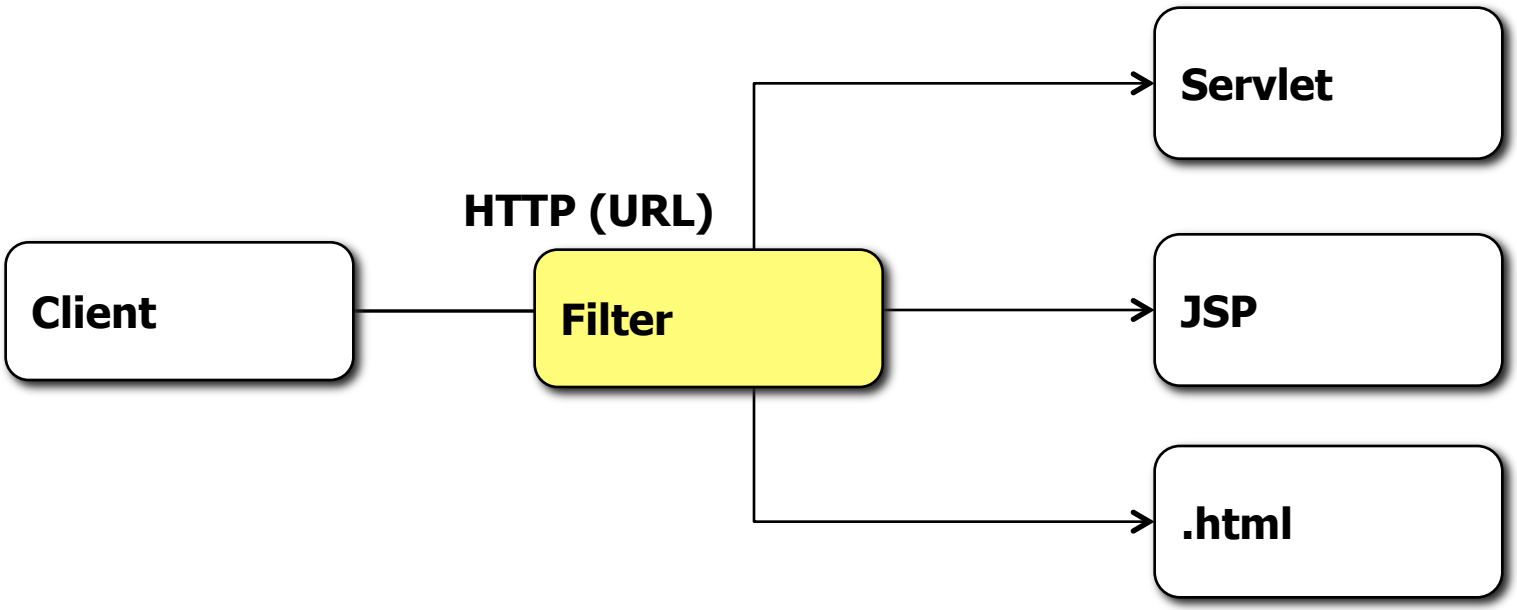
Servlet Filters

Traduction en cours

Servlet Filter principle



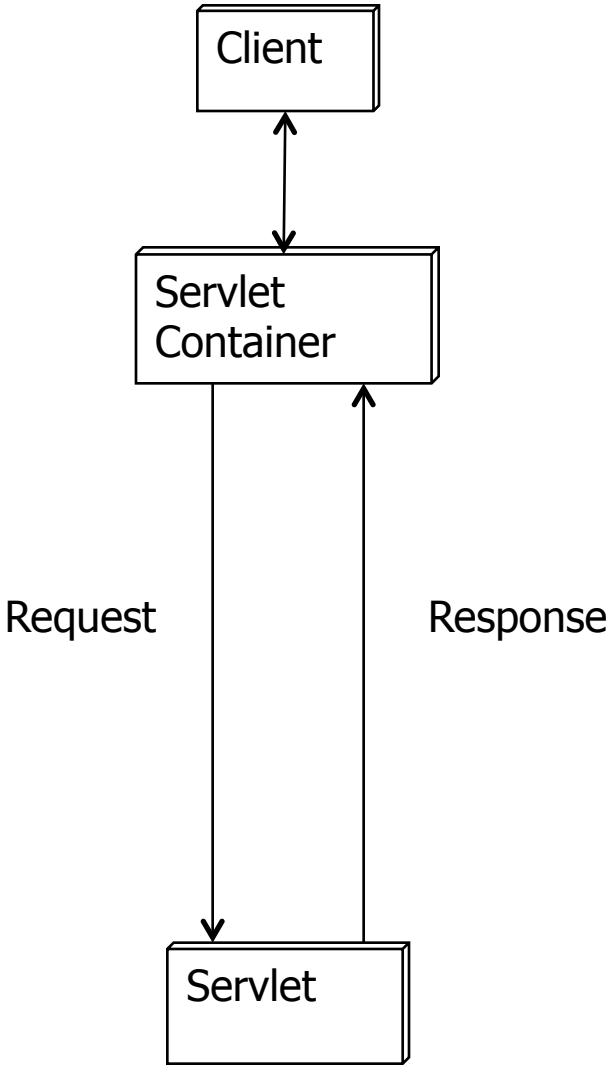
Servlet Filter principle

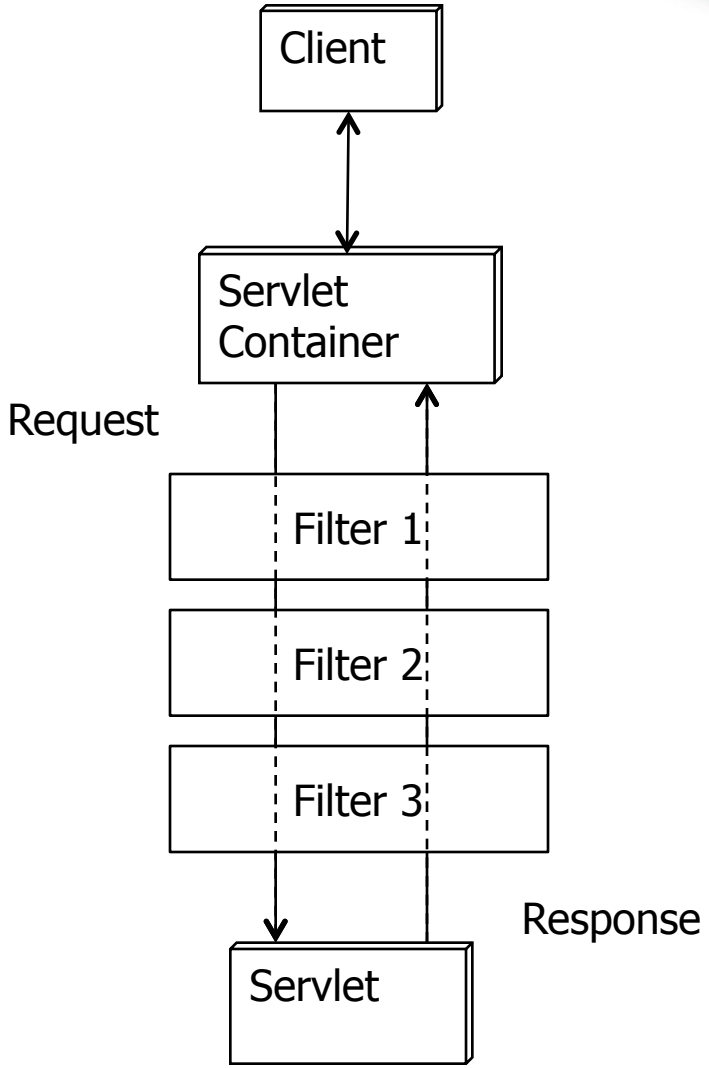


Servlet API version

Servlet API version	***Released***	***Platform***	***Important Changes***
Servlet 3.1	Mai 203	JavaEE 7,	
Servlet 3.0	December 2009	JavaEE 6, JavaSE 6	Pluggability, Ease of development, Async Servlet, Security, File Uploading
Servlet 2.5	September 2005	JavaEE 5, JavaSE 5	Requires JavaSE 5, supports annotations
Servlet 2.4	November 2003	J2EE 1.4, J2SE 1.3	web.xml uses XML Schema
Servlet 2.3	August 2001	J2EE 1.3, J2SE 1.2	Addition of Filter
Servlet 2.2	August 1999	J2EE 1.2, J2SE 1.2	Becomes part of J2EE, introduced independent web applications in .war files
Servlet 2.1	November 1998	Unspecified	First official specification, added RequestDispatcher, ServletContext
Servlet 2.0		JDK 1.1	Part of Java Servlet Development Kit 2.0
Servlet 1.0	June 1997		

Servlet Filter and container





Servlet Filters

- A servlet filter can intercept requests to servlets, JSP's, HTML files or other static content.
- A *filter* dynamically intercepts requests and responses to transform or use the information contained in the requests or responses.
- Filters typically do not themselves create responses, but instead provide universal functions that can be "attached" to any type of servlet or JSP page.

http://www.oracle.com/technetwork/java/filters-137243.html

Filter Usage

- Authentication-Blocking requests based on user identity.
- Logging and auditing-Tracking users of a web application.
- Image conversion-Scaling maps, and so on.
- Data compression-Making downloads smaller.
- Localization-Targeting the request and response to a particular locale.
- XSL/T transformations of XML content-Targeting web application responses to more that one type of client.

http://www.oracle.com/technetwork/java/filters-137243.html

Filter and Code Modularity

- Organized developers are constantly on the lookout for ways to modularize their code.
- Modular code is more manageable and documentable, is easier to debug, and if done well, can be reused in another setting.
- No need to recompile anything to change the input or output of your web application.
- Just edit WEB.xml file or use a tool to change the WEB app configuration.

http://www.oracle.com/technetwork/java/filters-137243.html

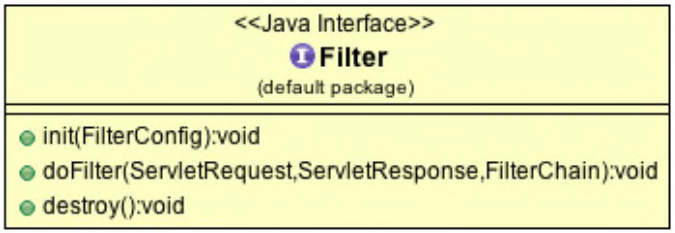
Filter WEB.xml

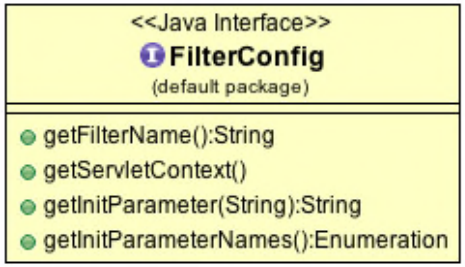
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
  <display-name>helloServlet</display-name>
  <servlet>
    <display-name>HelloServlet</display-name>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>hello.HelloServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/HelloServlet</url-pattern>
  </servlet-mapping>
  <filter>
    <display-name>HelloServletFilter</display-name>
    <filter-name>HelloServletFilter</filter-name>
    <filter-class>hello.HelloServletFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>HelloServletFilter</filter-name>
    <url-pattern>/HelloServletFilter</url-pattern>
  </filter-mapping>
</web-app>
```

Filters API

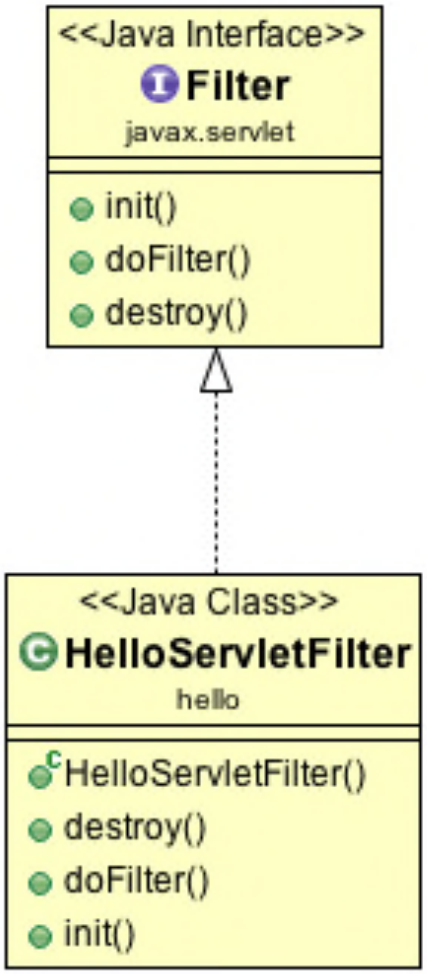
- Filter API of the javax.servlet package is defined by :
  - Filter interface,
  - FilterChain interface,
  - FilterConfig interface.
- A Filter is defined by implementing the Filter interface.
- A filter chain, passed to a filter by the container, provides a mechanism for invoking a series of filters.
- The filter config contains initialization data.

Filter Interface





Filter Interface



Filter interface methods

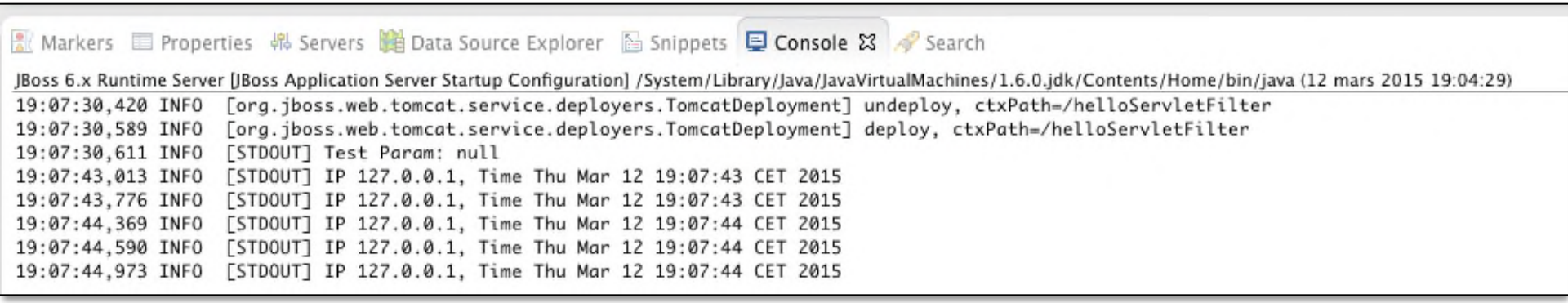
- The init() and destroy() methods must be implemented.
- The init() method is called by the container when the filter is instantiated.
- Filter initialization parameters are passed to init() in the FilterConfig object.
- doFilter() is where the user code is located.

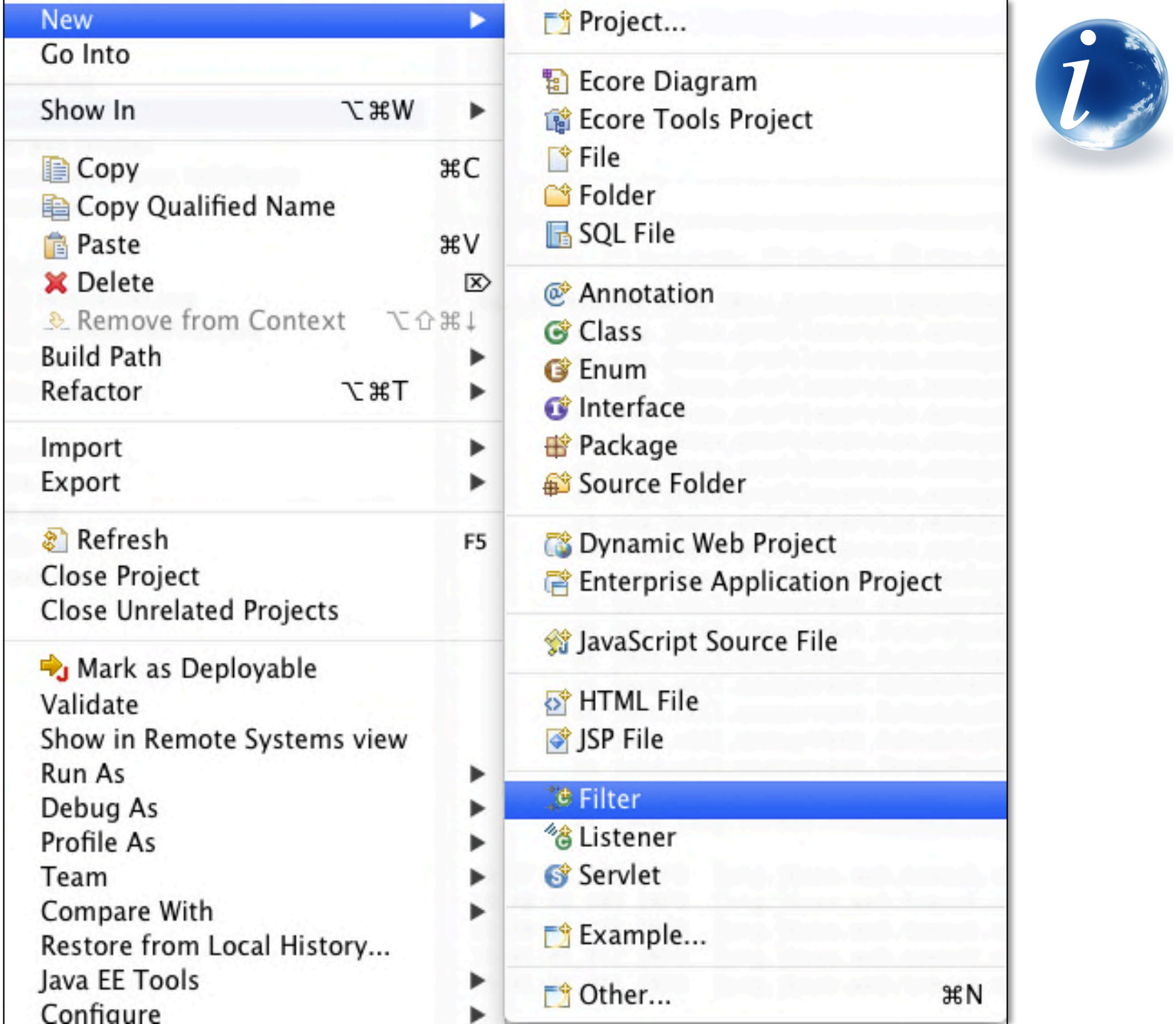
http://www.oracle.com/technetwork/java/filters-137243.html

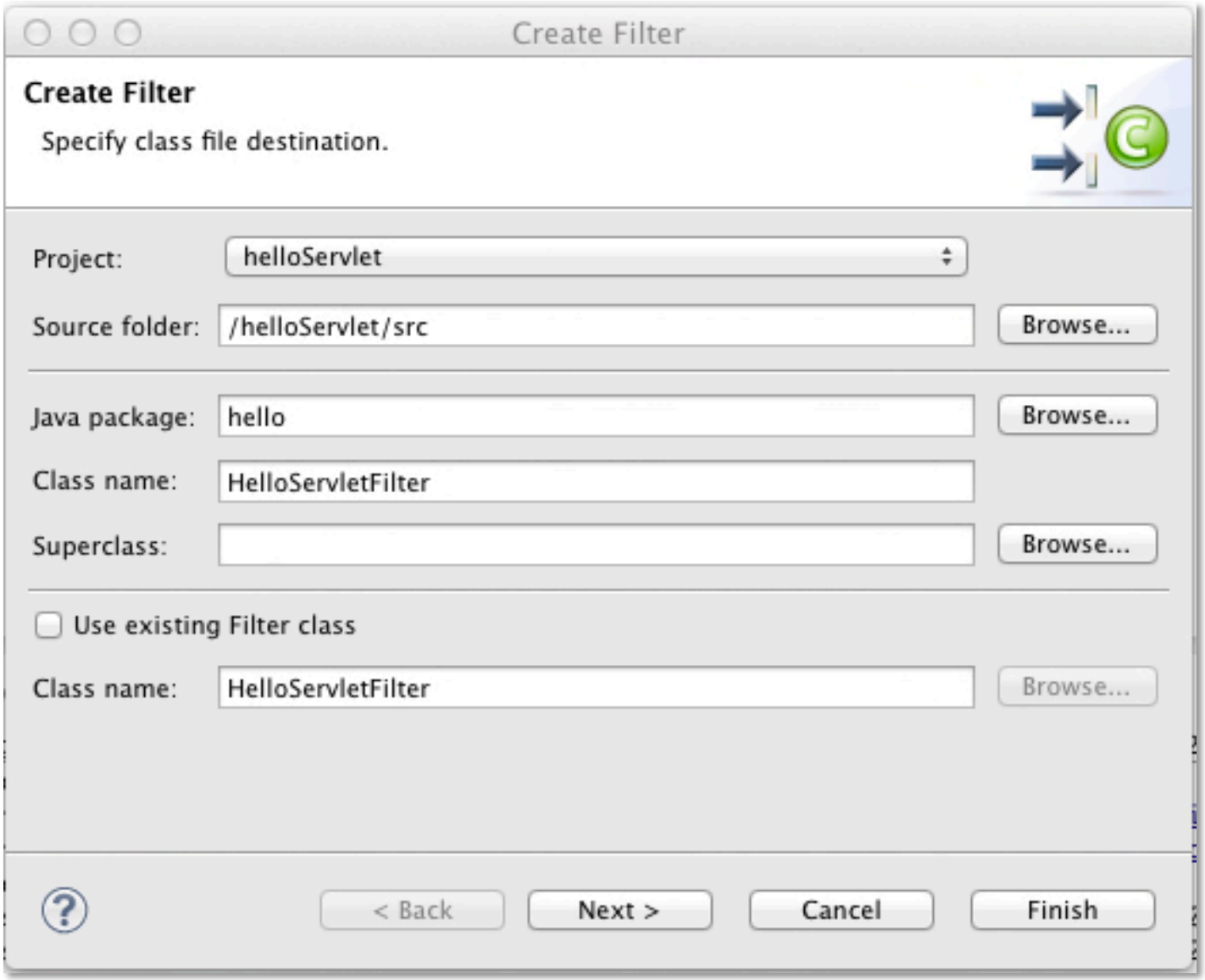
```java
package hello;

import java.io.IOException;
import java.util.Date;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

public class HelloServletFilter implements Filter {

    public void doFilter(ServletRequest req, ServletResponse res,
            FilterChain chain) throws IOException, ServletException {

        HttpServletRequest request = (HttpServletRequest) req;

        //Get the IP address of client machine.
        String ipAddress = request.getRemoteAddr();

        //Log the IP address and current timestamp.
        System.out.println("IP "+ipAddress + ", Time "
                            + new Date().toString());

        chain.doFilter(req, res);
    }
    public void init(FilterConfig config) throws ServletException {

        //Get init parameter
        String testParam = config.getInitParameter("test-param");

        //Print the init parameter
        System.out.println("Test Param: " + testParam);
    }
    public void destroy() {
        //add code to release any resource
    }
}
```

```
package hello;

public class HelloServletFilter implements Filter {

   public void doFilter(ServletRequest req, ServletResponse res,
         FilterChain chain) throws IOException, ServletException {

      HttpServletRequest request = (HttpServletRequest) req;

      //Get the IP address of client machine.
      String ipAddress = request.getRemoteAddr();

      //Log the IP address and current timestamp.
      System.out.println("IP "+ipAddress + ", Time "
                  + new Date().toString());

      chain.doFilter(req, res);
   }
   public void init(FilterConfig config) throws ServletException {

      //Get init parameter
      String testParam = config.getInitParameter("test-param");

      //Print the init parameter
      System.out.println("Test Param: " + testParam);
   }
   public void destroy() {
      //add code to release any resource
   }
}
```

Filter Output




New
Go Into
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Refactor ⌥⌘T
Import
Export
Refresh F5
Close Project
Close Unrelated Projects
Mark as Deployable
Validate
Show in Remote Systems view
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Java EE Tools
Configure
Project...
Ecore Diagram
Ecore Tools Project
File
Folder
SQL File
Annotation
Class
Enum
Interface
Package
Source Folder
Dynamic Web Project
Enterprise Application Project
JavaScript Source File
HTML File
JSP File
Filter
Listener
Servlet
Example...
Other... ⌘N


Create Filter
Create Filter
Specify class file destination.
Project:
helloServlet
Source folder:
/helloServlet/src
Browse...
Java package:
hello
Browse...
Class name:
HelloServletFilter
Superclass:
Browse...
Use existing Filter class
Class name:
HelloServletFilter
Browse...
< Back
Next >
Cancel
Finish

Specifying Filter Configuration

- A filter is mapped to a servlet.
- A filter class is defined as a servlet class.
- A filter is mapped to a servlet by defining a <filter-mapping> element in the deployment descriptor.
- This element maps a filter name to a servlet by name or by URL pattern.

http://www.oracle.com/technetwork/java/filters-137243.html

Filter config

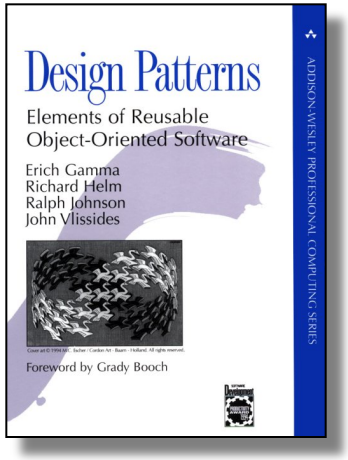
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">

  <display-name>helloServlet</display-name>
  <servlet>
    <display-name>HelloServlet</display-name>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>hello.HelloServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/HelloServlet</url-pattern>
  </servlet-mapping>
  <filter>
    <display-name>HelloServletFilter</display-name>
    <filter-name>HelloServletFilter</filter-name>
    <filter-class>hello.HelloServletFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>HelloServletFilter</filter-name>
    <url-pattern>/HelloServlet</url-pattern>
  </filter-mapping>
</web-app>
```

Filter config map all servlet

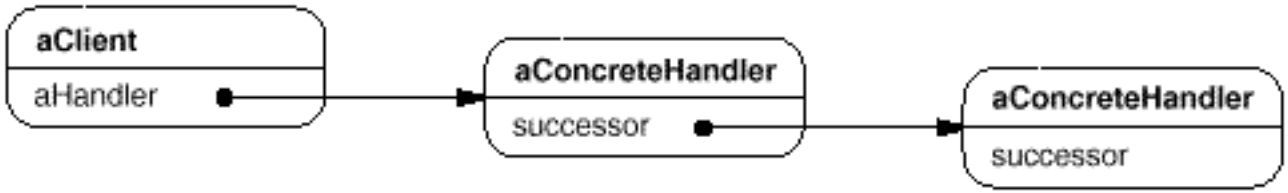
```
<filter-mapping>
  <filter-name>HelloServletFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

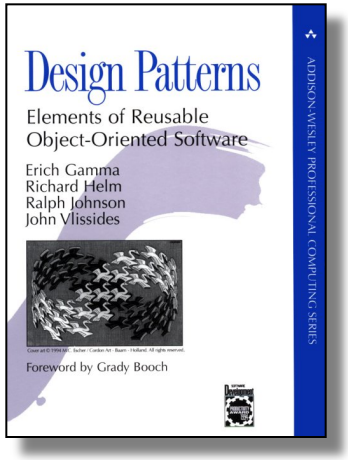
GOF Chain of Responsibility

- The Chain of Responsibility design pattern allows an object to send a command without knowing what object will receive and handle it.

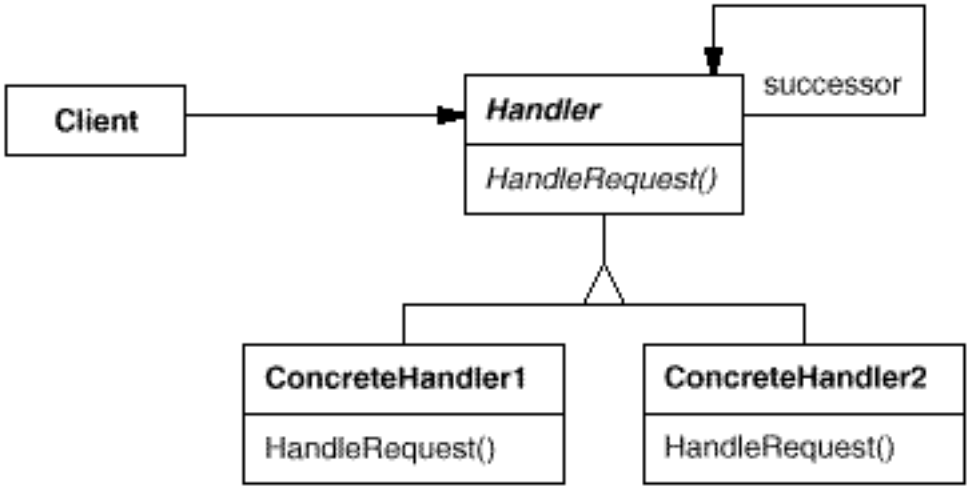


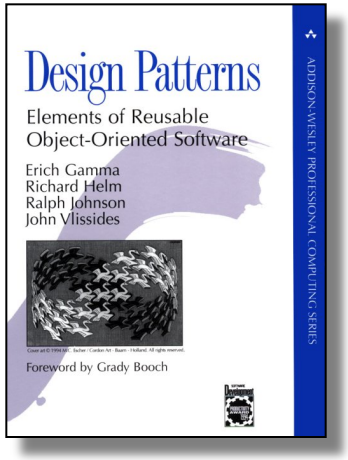
Chain of responsability



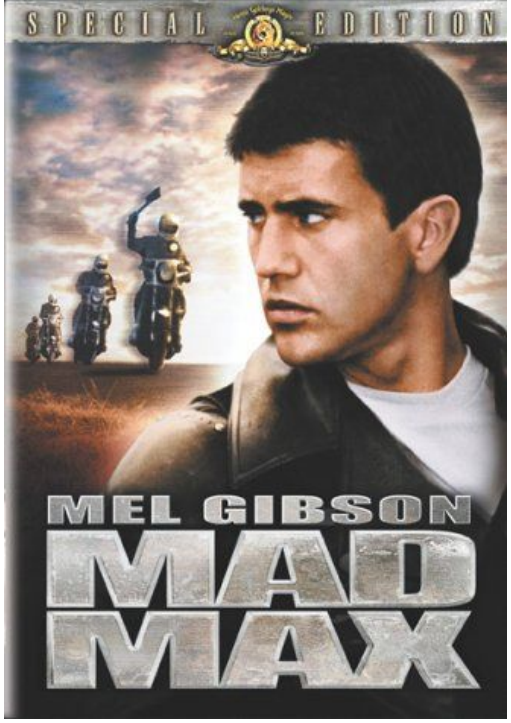


Chain of responsability



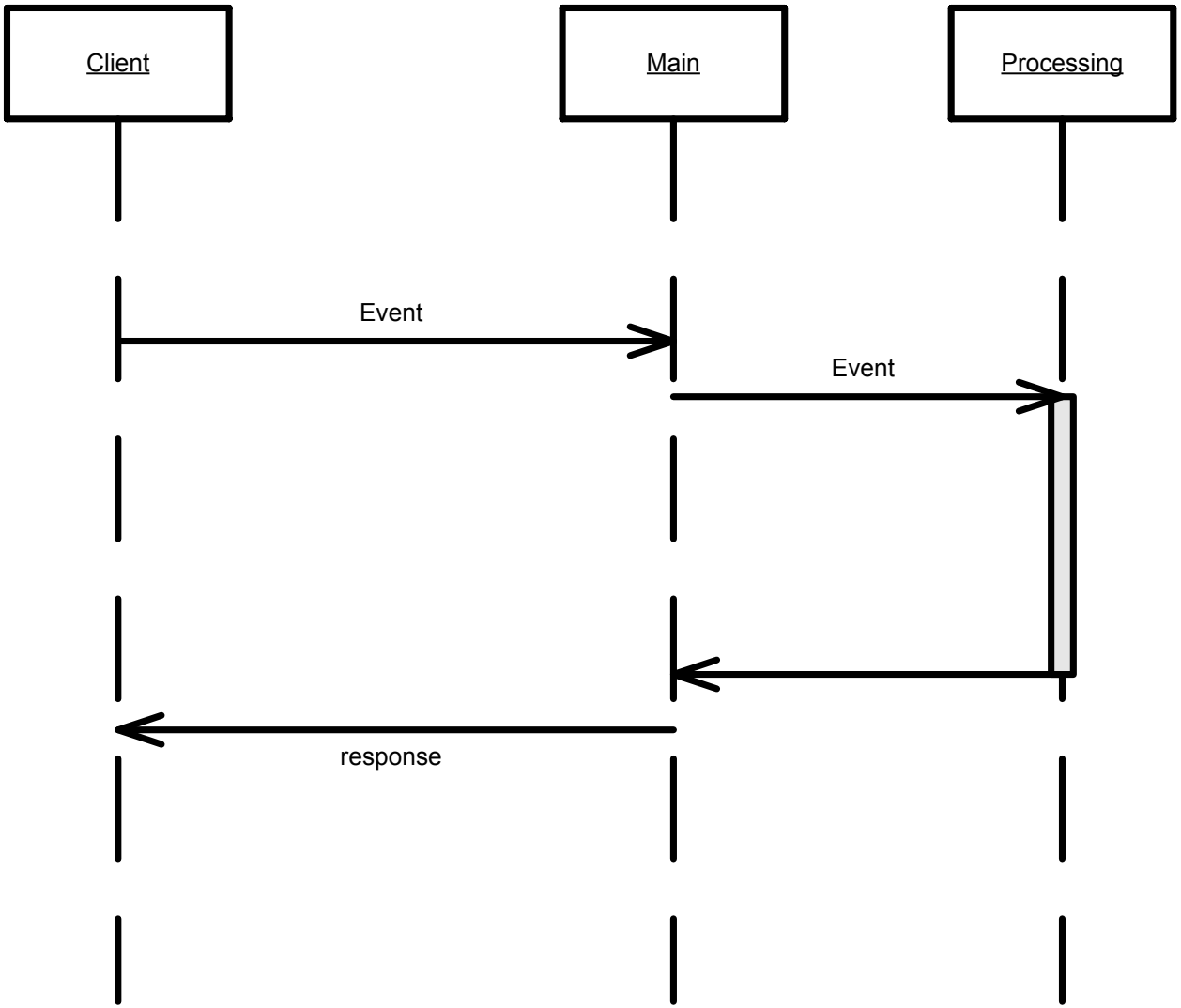


Interceptor

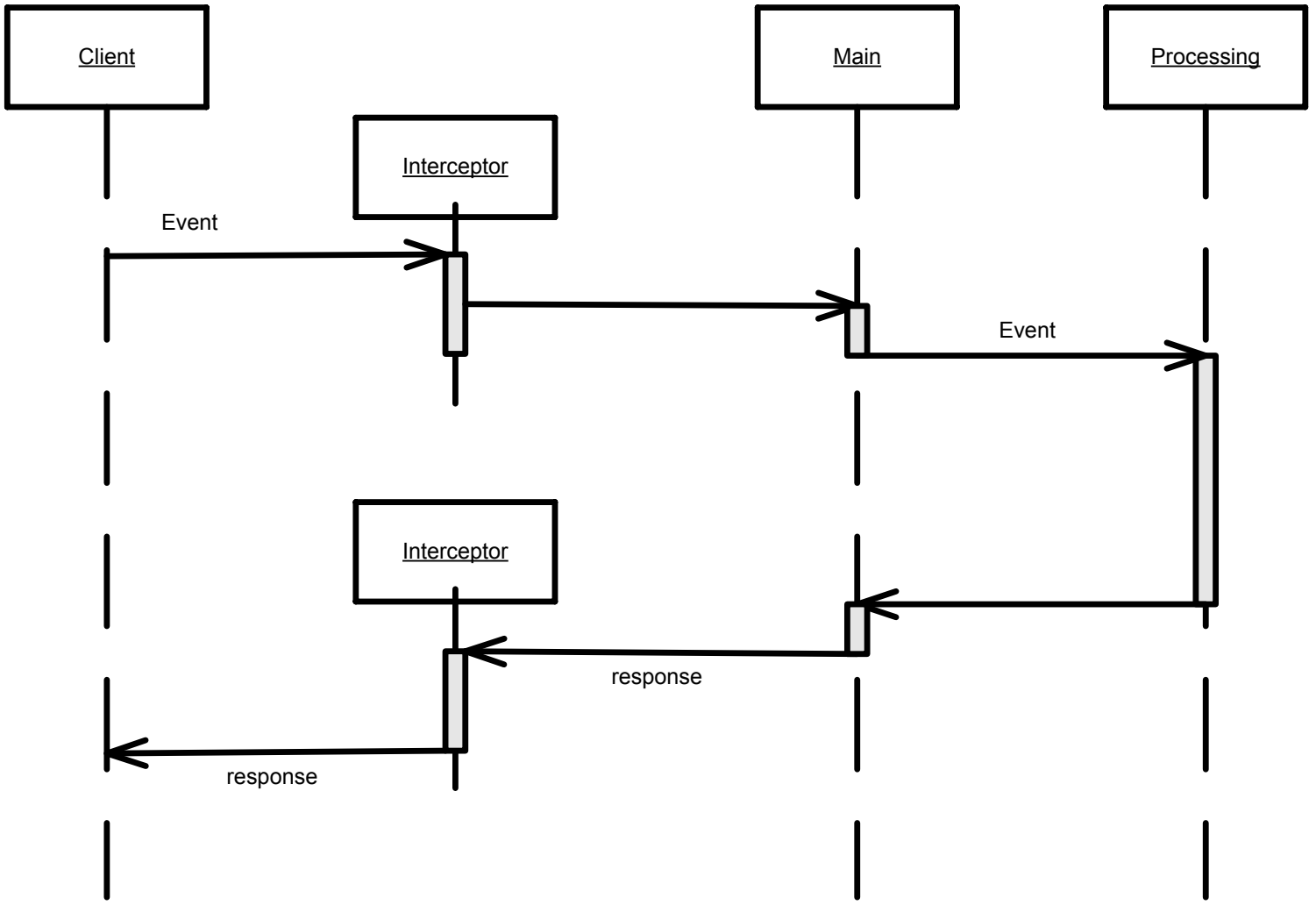


http://www.deviantart.com/morelikethis/artists/130272390?view_mode=2

Client to server interceptor



Client to server interceptor



Framework interceptor