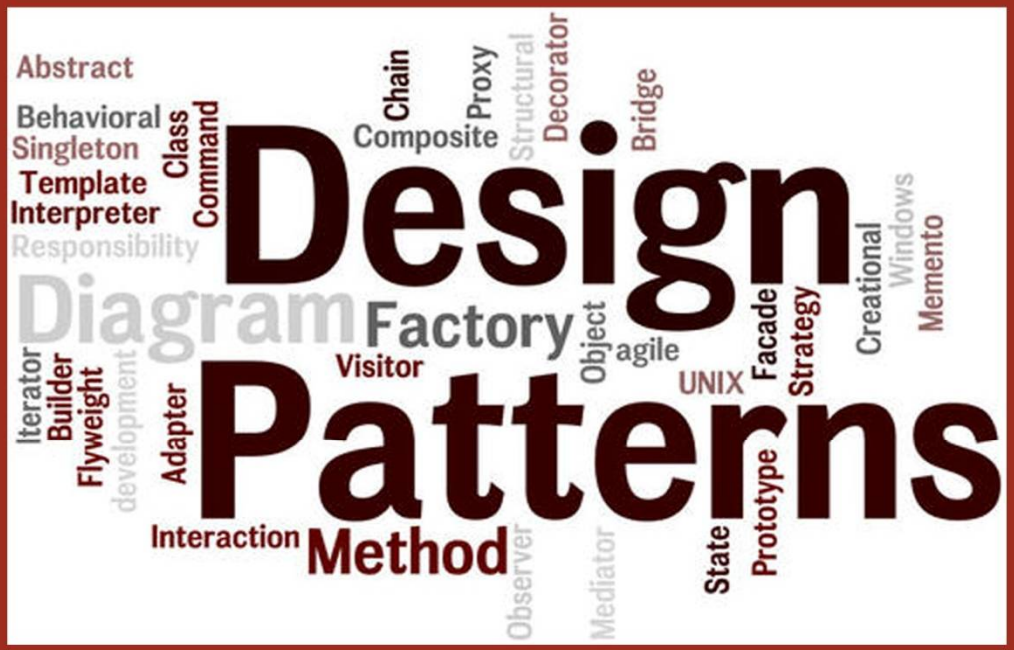
JAVA means DURGA SOFT

Design Patterns

Web Level Design Patterns

6. Intercepting Filter Design Pattern



Intercepting Filter

Problem: When a request enters a Web application, it often must pass several entrance tests prior to the main processing stage. For example,

- ✓ Has the client been authenticated?
- ✓ Does the client have a valid session?
- ✓ Is the client's IP address from a trusted network?
- ✓ Does the request path violate any constraints?
- ✓ What encoding does the client use to send the data?
- ✓ Do we support the browser type of the client?

Different type of processing is required for the different type of request received by the presentation tier request handler. Some requests are simply forwarded to the appropriate handler component, while other requests must be processed being further processed. **We require Preprocessing and post-processing of a client Web request and response.**

We require a common processing, such as checking the data-encoding scheme or logging information about each request, completes per request. Centralization of common logic is desired. Services should be easy to add or remove unobtrusively without affecting existing components, so that they can be used in a variety of combinations, such as Logging and authentication, Debugging and transformation of output for a specific client, Uncompressing and converting encoding scheme of input.

Most applications have some requirements, such as security and logging that are applicable across all application requests. To add such functionality separately to each application service would be time-consuming, error-prone, and difficult to maintain. Even implementing these services within a front controller would still require code changes to add and remove services. The sequence diagram in Figure 1 below shows how each Web resource is responsible for calling such services individually.

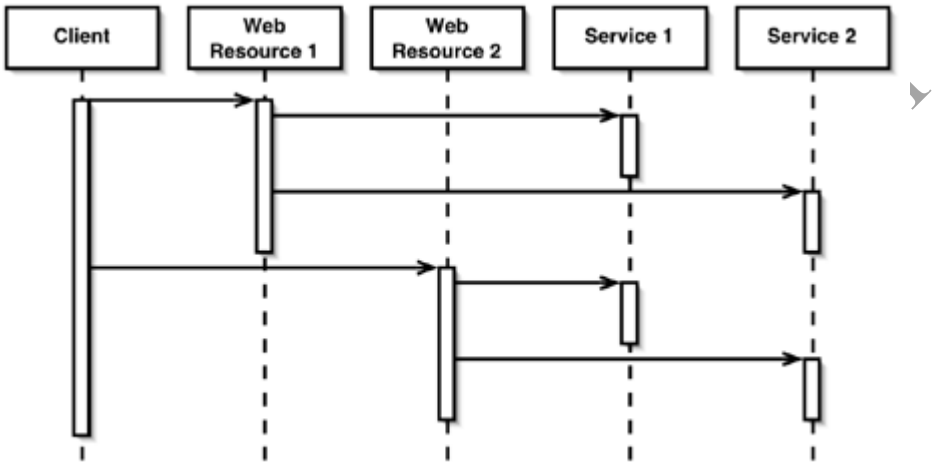


Figure1: Before Intercepting Filter

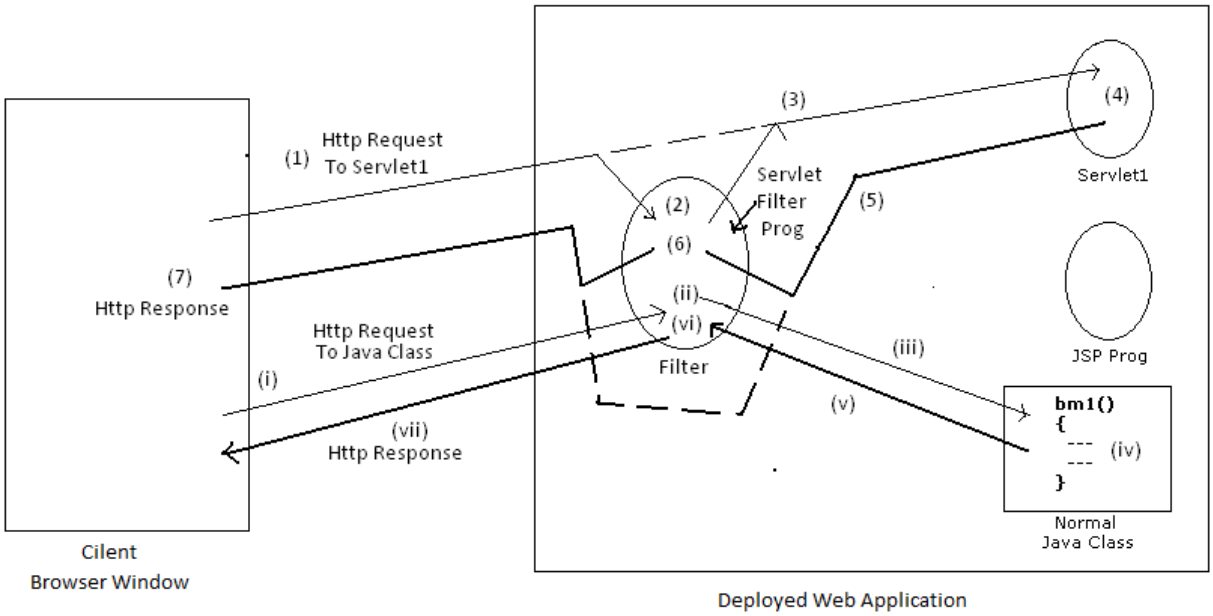
Solution: Create pluggable filters (Intercepting Filter) to process common services in a standard manner without requiring changes to core request processing code. The filters intercept (traps) incoming requests and outgoing responses, allowing preprocessing and post-processing. We are able to add and remove these filters unobtrusively, without requiring changes to our existing code.

Def: A pluggable component design to intercept incoming requests and outgoing responses, provide common services in a standard manner (independently) without changing core processing code.

The **Intercepting Filter**, as the name implies, intercepts the requests and helps separate all the tasks that need to be done on the server into reusable components. The actions/filters

can be assigned to specific requests, chained in any order, and added or removed within the application configuration. After the filters complete, they pass the request to the intended recipient, in this case Servlet or JSP. After the server is finished, a response is sent to the client.

In other words, Intercepting Filter is a special web resource program of a web application that is capable of trapping both request & response of other web resource programs by having pre-request processing and post response generation logic.



In the above diagram, Filter traps both request and response; steps (1-7) talks about how Servlet Filter trapping the request and response of Servlet1 and steps (i-vii) talks about how Servlet Filter takes Http request from client and process that request to normal java class.

Note A Front controller can trap only request of other web resource programs like servlet, JSP programs and it cannot trap response of them. But an Intecepting filter can trap both request & response of other web resource programs like servlet, JSP programs.

- The Servlet Filter interface in the Java 2 Platform, Enterprise Edition (J2EE) platform is a direct implementation of the Intercepting Filter pattern.
- In Struts 2.x environment a predefined servlet filter program i.e. **org.apache.struts2.dispatcher.FilterDispatcher** is given as controller.

Advantages:

- ❖ Logging and authentication.
- ❖ Enhance security.
- ❖ Add additional function to existing web application.
- ❖ Debug.

- Pre-processing or post-processing for specific clients.
- Uncompress incoming request.
- Convert input encoding schema.
- Being added or removed transparently or declaratively and triggered automatically
- Improve reusability
- Each filter is loosely coupled

Intecepting filters centralizes control with Loosely Coupled Handlers, Filters provide a central place for handling processing across multiple requests, as does a controller. Filters are better suited to massaging requests and responses for ultimate handling by a target resource, such as a controller. Additionally, a controller often ties together the management of numerous unrelated common services, such as authentication, logging, encryption, and so forth, while filtering allows for much more loosely coupled handlers, which can be combined in various combinations.

Intercepting filters improves reusability; Filters promote cleaner application partitioning and encourages reuse. These pluggable interceptors are transparently added or removed from existing code, and due to their standard interface, they work in any combination and are reusable for varying presentations.

Intercepting filters provide Declarative and Flexible Configuration; numerous services are combined in varying permutations without a single recompile of the core code base.

But sharing information between filters can be inefficient, since by definition each filter is loosely coupled. If large amounts of information must be shared between filters, then this approach may prove to be costly.

The Intercepting Filter pattern wraps existing application resources with a filter that intercepts the reception of a request and the transmission of a response. An intercepting filter can pre-process or redirect application requests, and can post-process or replace the content of application responses. Intercepting filters can also be stacked one on top of the other to add a chain of separate, declaratively-deployable services to existing Web resources with no changes to source code. Figure 2 below shows a chain of two intercepting filters intercepting requests to two Web resources that they wrap.

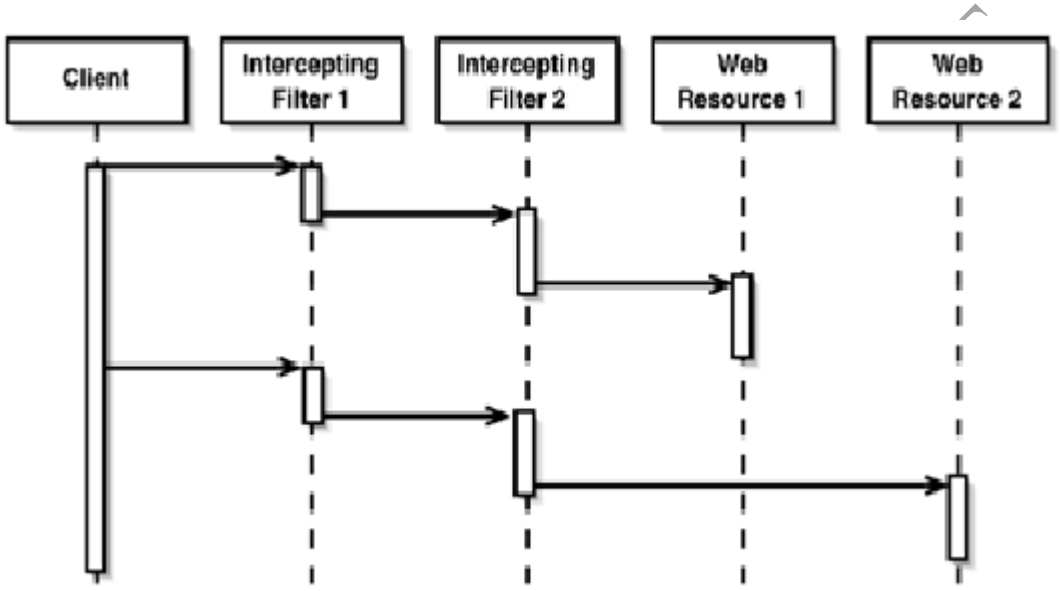


Figure2: After Intercepting Filter