Intercepting Filter

Context

The presentation-tier request handling mechanism receives many different types of requests, which require varied types of processing. Some requests are simply forwarded to the appropriate handler component, while other requests must be modified, audited, or uncompressed before being further processed.

Problem

Preprocessing and post-processing of a client Web request and response are required.

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?*

Some of these checks are tests, resulting in a yes or no answer that determines whether processing will continue. Other checks manipulate the incoming data stream into a form suitable for processing.

The classic solution consists of a series of conditional checks, with any failed check aborting the request. Nested if/else statements are a standard strategy, but this solution leads to code fragility and a copy-and-paste style of programming, because the flow of the filtering and the action of the filters is compiled into the application.

The key to solving this problem in a flexible and unobtrusive manner is to have a simple mechanism for adding and removing processing components, in which each component completes a specific filtering action.

Forces

- 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

Solution

Create pluggable filters to process common services in a standard manner without requiring changes to core request processing code. The filters intercept 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.

We are able, in effect, to decorate our main processing with a variety of common services, such as security, logging, debugging, and so forth. These filters are components that are independent of the main application code, and they may be added or removed declaratively. For example, a deployment configuration file may be modified to set up a chain of filters. The same configuration file might include a mapping of specific URLs to this filter chain. When a client requests a resource that matches this con

d URL mapping, the filters in the chain are each processed in order before the requested target resource is invoked.

Structure

Figure 1.1represents the Intercepting Filter pattern.

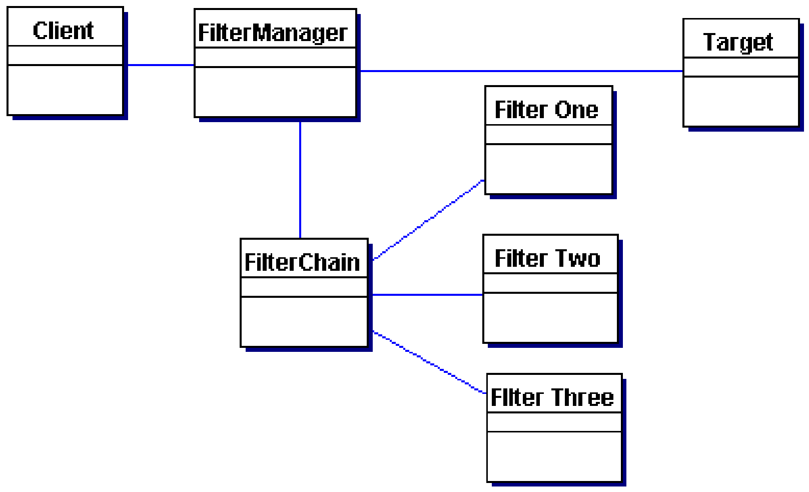

Figure 1.1 Intercepting Filter pattern class diagram.

Participants and Responsibilities

Figure 1.2 represents the Intercepting Filter pattern.

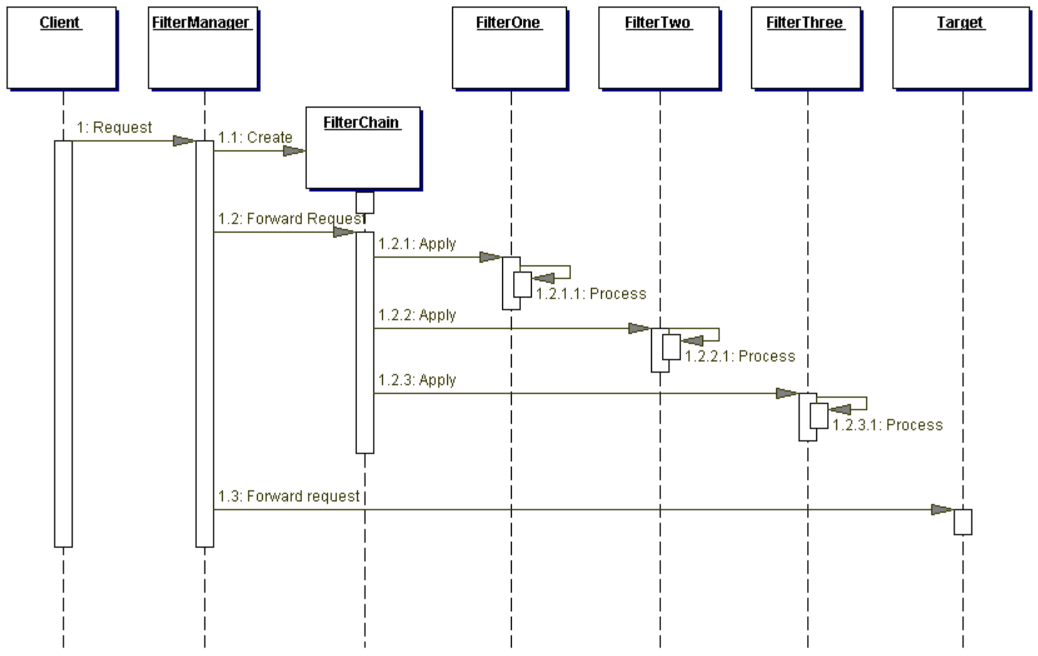

Figure 1.2 Intercepting Filter sequence diagram.

FilterManager

The FilterManager manages filter processing. It creates the FilterChain with the appropriate filters, in the correct order, and initiates processing.

FilterChain

The FilterChain is an ordered collection of independent filters.

FilterOne, FilterTwo, FilterThree

These are the individual filters that are mapped to a target. The FilterChain coordinates their processing.

Target

The Target is the resource requested by the client.