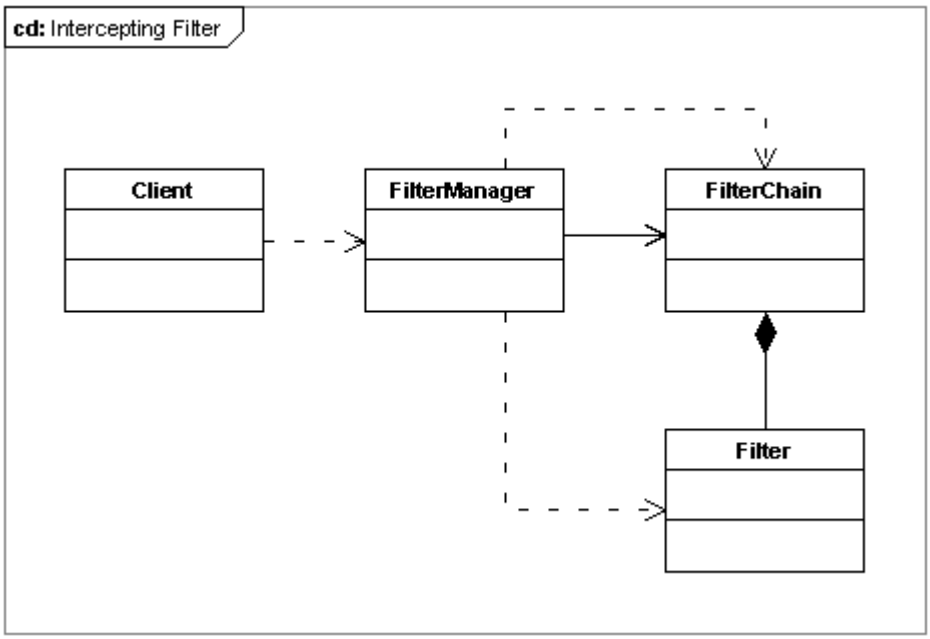
Intercepting Filter Tasarım Şablonu

KurumsalJava.com

Özcan Acar
Bilgisayar Mühendisi
http://www.ozcanacar.com

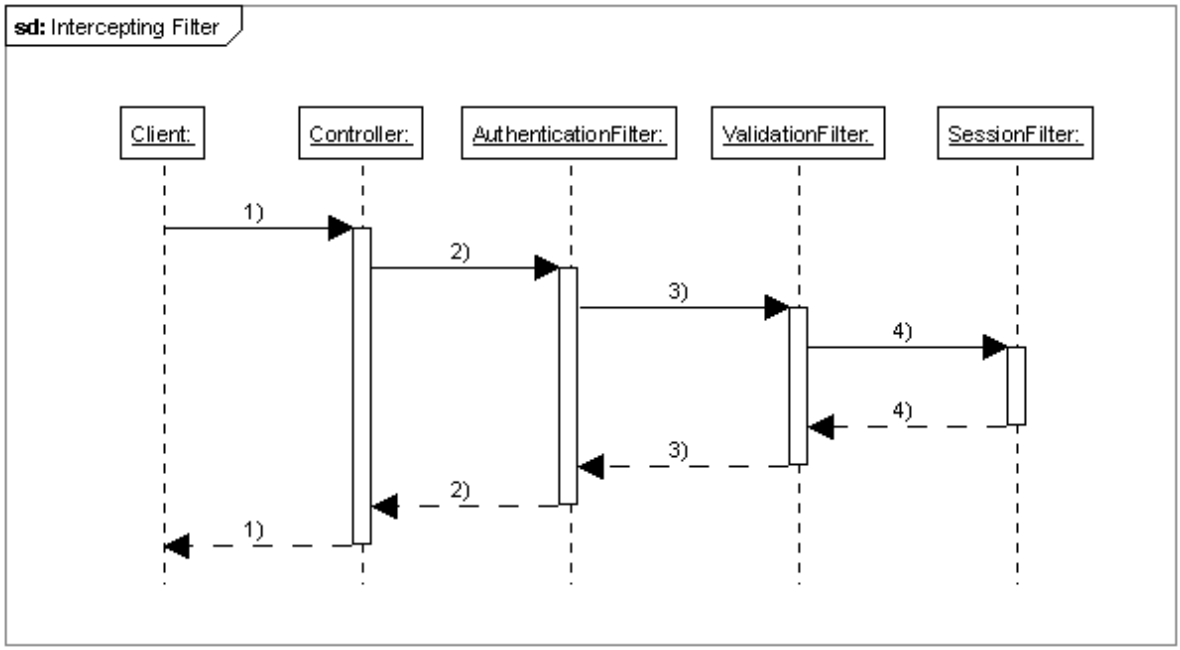
Front Controller[1] tasarım şablonunda, kullanıcıdan gelen isteklerin (request) merkezi bir yerde toplanarak, işlem yapıldığını daha önce görmüştük. Intercepting Filter tasarım şablonu ile, kullanıcının isteği (request) işleme alınmadan önce filtreler kullanılarak süzgeçten geçirilir. Örneğin bir filtre ile kullanıcının işlem öncesi login yaptığını konrol edebiliriz. Filtremiz, session (HttpServletSesion) içinde login bilgilerini bulamadığı taktirde, kullanıcıyı login sayfasına yönlendirebilir.

Aynı şekilde işlem tamamlandıktan sonra kullanıcıya gönderilecek cevap (response) filtreler yardımı ile modifike edilebilir.



Intercepting Filter tasarım şablonunun merkezinde filtreler bulunur. Bunlar belirli özelliklere sahip normal Java sınıflarıdır. JEE dünyasında Servlet Filter[2] olarak bilinen bu sınıflar, filter chaining mekanizması ile bir kolyenin üzerinde bulunan boncuklar gibi, arka arkaya dizilerek, işleme tabi tutulabilirler.

[1] http://www.kurumsaljava.com/2009/10/09/front-controller-tasarim-sablonu/
[2] Bakınız: http://java.sun.com/products/servlet/Filters.html



Uml diagramında görüldüğü gibi birden fazla filtre arka arkaya dizilerek işlem yapılabilir. Sıradaki her filtre kendi görevini yerine getirdikten sonra kontrolü FilterManager yardımı ile kendinden sonraki filtreye bırakır. Eğer işlem esnasında filtrelerden birisi hatalı bir durum tespit ederse (örneğin session içinde login bilgilerinin bulunamaması) kendinden sonra gelen filtreler devreye girmeden işlemi durdurur ve filtre içinde tanımlanmış aksiyonu gerçekleştirir (örneğin login sayfasına yönlendirme).

Tomcat [3] gibi application serverlerde filtreler web.xml dosyasında tanımlanır. FilterManager Tomcat içinde implemente edilen bir sınıf olduğundan, filtrelerin oluşturulmasını ve işleme alınması Tomcat tarafından otomatik olarak gerçekleşir.

Üyelerin login yaptığını kontrol etmek için *AuthenticationFilter* isminde bir filtre örneğini inceliyelim:

```
package org.javatasarim.pattern.interceptingfilter;

import java.io.IOException;
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;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;


/**
 * Login yapmis üye bilgilerini session
 * içinde kontrol eder. Login yapmamis bir
 * kullaniciyi login.jsp sayfasina yönlendirir.
```

[3] Bakınız: http://tomcat.apache.org/

```java
 *
 * @author Oezcan Acar
 *
 */
public final class AuthenticationFilter implements Filter
{

    /**
     * Bir kullanici login yaptiktan sonra, session içine
     * USER etiketi altinda login ismi yerlestirilir.
     *
     *  Eger session içinde USER isminde bir etiket yoksa,
     *  o zaman kullanici login yapmamis demektir.
     */
      private static final String USER = "user";


    /**
     * FilterManager (Tomcat) doFilter() metodunu kullanarak,
     * filtreyi işleme alir.
     */
      public void doFilter(final ServletRequest req,
                                    final ServletResponse res,
                                    final FilterChain chain)
                               throws IOException, ServletException
      {

        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        HttpSession session = request.getSession();

        if(session.getAttribute(USER) != null)
        {
            /**
             * Kontrolü bir sonraki filtreye vermek için
             * chain.doFilter() metodu kullanılır.
             */
            chain.doFilter(req, res);
        }
        else
        {
            response.sendRedirect("login.jsp");
        }
    }


    /**
     * Filter init()
     */
      public void init(final FilterConfig arg0)
                               throws ServletException
    {
      }


      /**
       * Filter destroy()
       */
      public void destroy()
      {
      }
}
```

Tomcat altında kullanmak istediğimiz filtrelerin *javax.servlet.Filter* interface sınıfını implemente etmeleri gerekmektedir. *Filter* interface sınıfında, kendi filtre sınıfımızın implemente etmesi gereken üç metod bulunmaktadır: *init()*, *doFilter()* ve *destroy()*.

init() metodu ile filtre tarafından kullanılacak kaynaklar oluşturulur. *init()* metodunu bir sınıf konstruktörü olarak düşünebilirsiniz. Filtre görevine başlamadan önce *FilterManager* (Tomcat) tarafından önce *init()* metodu işleme alınır.

Filtrenin asıl görevi *doFilter()* metodu bünyesinde implemente edilir. İşlem tamamlandıktan sonra *chain.doFilter()* ile kontrol bir sonraki filtreye verilir. Eğer sırada başka bir filtre yoksa, *FilterManager* gelen isteği (request) gerçek sahibine iletir (örneğin *FrontController*). Örneğimizde görüldüğü gibi session içinde gerekli bilgi bulunamadığı taktirde, üye *login.jsp* sayfasına yönlendilir. Bu noktadan itibaren sıradaki filtreler işlem görmez ve kullanıcı login yapabilmesi için *login.jsp* sayfasına yönlendirilir. Böylece sadece login yapmış üyelerin kullanabileceği fonksiyonların login yapmamış bir üye tarafından kullanımı engellenmiş olur.

Filtre görevini yerine getirdikten sonra *destroy()* metodu ile yok edilir. Bunu FilterManager otomatik olarak gerçekleştirir.

Filtremizin Tomcat altında çalışabilmek için aşağıdaki şekilde web.xml dosyasına kayıtlanması gerekmektedir:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.4"
      xmlns="http://java.sun.com/xml/ns/JEE"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/JEE/web-
app_2_4.xsd">

      <filter>
            <filter-name>AuthenticationFilter</filter-name>
            <filter-class>
                  org.javatasarim.pattern.
                      interceptingfilter.AuthenticationFilter
            </filter-class>
      </filter>

</web-app>
```

Intercepting Filter tasarım şablonu ne zaman kullanılır?

- Kullanıcı isteği (request) işleme alınmadan önce ve işlem tamamlandıktan sonra filtreler aracılığı ile logging, üyelik ve veri kontrolü gibi işlemler için Intercepting Filter tasarım şablonu kullanılır.

İlişkili tasarım şanlonları:

- Front Controller tasarım şablonu da Intercepting Filter tasarım şablonu gibi verilerin merkezi bir yerde işlem görmesini kolaylaştırır.

EOF (End Of Fun)
Özcan Acar