


# Filters



Un filtro è una classe in grado di **intercettare una richiesta** e eseguire elaborazioni sulla richiesta stessa o sulla risposta generata da una servlet.

- Queste funzionalità possono essere aggiunte o eliminate in maniera indipendente l'una dall'altra e con una semplice configurazione esterna al codice
- I filtri vengono **configurati nel deployment descriptor**  per intercettare tutte le richieste del client ad una determinata servlet o a un gruppo di servlet individuate da uno specifico URL pattern
- L'elaborazione della richiesta da parte di un filtro avviene prima che questa arrivi alla servlet a cui era destinata la richiesta.

# I metodi

Una classe per essere un filtro deve implementare l'interfaccia standard **javax.servlet.Filter** che prevede tre metodi:

- public void **init()**

È il metodo invocato dal web container per inizializzare il filtro

- public void **doFilter**(ServletRequest request, ServletResponse response, FilterChain chain)

È il metodo invocato dal web container per eseguire l'elaborazione del filtro quando questo è invocato nella catena di filtri

- public void **destroy()**

È il metodo invocato dal web container per mettere fuori servizio il filtro



# Filter chain

La determinazione della sequenza dei filtri della catena è fatta nel deployment descriptor dell'applicazione.

Definire un filtro è molto simile alla definizione di una servlet •I filtri sono particolarmente adatti per operazioni quali:

- Autenticazione
- Logging e Auditing
- Conversione di immagini
- Compressione dati
- Encryption
- Tokenizing



# Esempio:

Vogliamo realizzare un'applicazione che contenga due tipi di pagine:

- **Pagine pubbliche**, visibili a tutti
- **Pagine riservate**, visibili solo agli utenti registrati
- Per accedere alle pagine riservate l'utente deve fornire login e password
- Non deve essere possibile accedere alle pagine riservate scavalcando la verifica di login e password, neanche conoscendo l'URL di una pagina riservata

Soluzione:

- Inseriamo tutte le pagine riservate in una sottocartella riservata
- Definiamo un filtro che controlla gli accessi al contenuto di riservate
- Il filtro userà informazioni di sessione per verificare se l'utente è stato autenticato
- La verifica di login e password viene effettuata da una servlet associata a un form



# Home page

Questa è una pagina pubblica.

[Clicca qui](#) per vedere una pagina riservata. [Clicca qui](#) per vedere un'altra pagina riservata.

## Inserimento login e password

Per accedere alla sezione riservata devi fornire login e password.

Login:

Password:

## Inserimento login e password

Per accedere alla sezione riservata devi fornire login e password.

Login:

Password:

## Autenticazione eseguita

Login e password verificati.

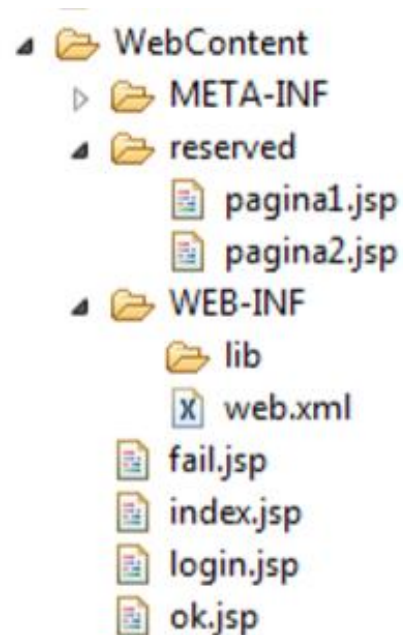
[Clicca qui](#) per tornare alla home page e accedere alle pagine riservate.

## Questa è una pagina riservata!

Questa è la pagina riservata 2.

[Clicca qui](#) per tornare alla home page.

[Clicca qui](#) per vedere la prima pagina riservata.



```

public class LoginFilter implements Filter {

    private ServletContext ctx;

    @WebFilter("/LoginFilter");
    public void init(FilterConfig fConfig) throws ServletException {
        ctx=ctg.getServletContext();
    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

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

        String auth = (String)session.getAttribute("authenticated");
        if("yes".equals(auth)) {
            chain.doFilter(req, res);
        }else {
            request.sendRedirect(ctx.getContextPath() + "/login.jsp");
        }
    }

    public void destroy() {
    }

}

```



```
public class CheckLogin extends HttpServlet {  
  
    ...  
  
    protected void doPost(ServletRequest request, ServletResponse response)  
        throws IOException, ServletException {  
  
        HttpSession session = request.getSession();  
  
        String trueLogin = getInitParameter("login");  
        String truePass = getInitParameter("pass");  
        RequestDispatcher disp;  
  
        if(trueLogin.equals(login) && truePass.equals(pass)){  
            session.setAttribute("authenticated", "yes");  
            disp = request.getRequestDispatcher("ok.jsp");  
            disp.forward(request, response);  
        }else{  
            session.setAttribute("authenticated", "no");  
            disp = request.getRequestDispatcher("fail.jsp");  
            disp.forward(request, response);  
        }  
    }  
}
```





```
<servlet>
  <servlet-name>CheckLogin</servlet-name>
  <servlet-class>CheckLogin</servlet-class>
  <init-param>
    <param-name>login</param-name>
    <param-value>pipipo</param-value>
  </init-param>
  <init-param>
    <param-name>pass</param-name>
    <param-value>baudo</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>CheckLogin</servlet-name>
  <url-pattern>/CheckLogin</url-pattern>
</servlet-mapping>
<filter>
  <filter-name>LoginFilter</filter-name>
  <filter-class>LoginFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>LoginFilter</filter-name>
  <url-pattern>/reserved/*</url-pattern>
</filter-mapping>
```

