

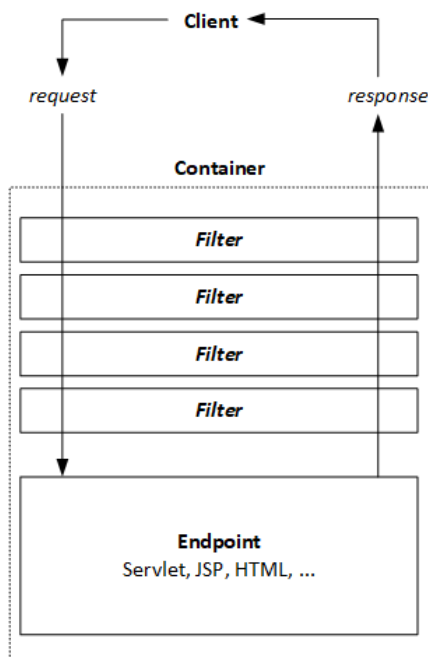
# Filter

## 1. Filter là gì?

Java Servlet 2.3 giới thiệu thành phần Filter. Filter sử dụng để “lọc”, hay thao-tác-trước đối với request và thao-tác-sau đối với response để chuyển đổi hoặc sử dụng thông tin được chứa trong request hoặc response đó.

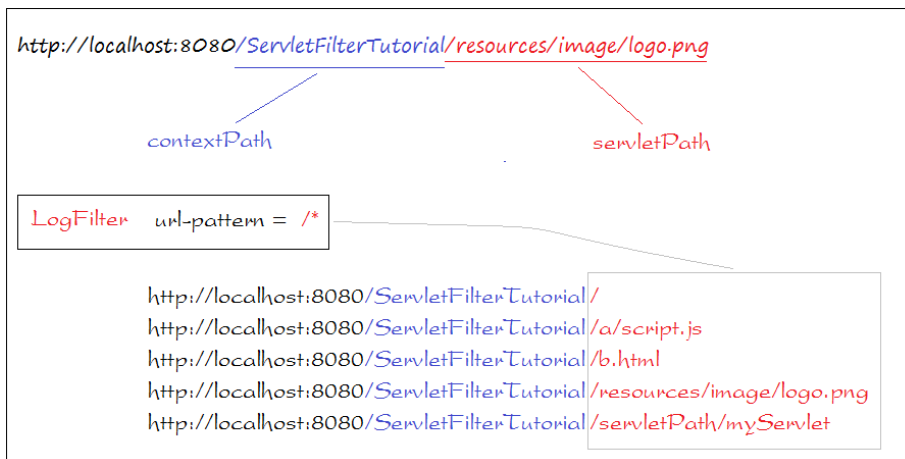
Hay dễ hiểu hơn là một đối tượng filter sẽ thực hiện các công việc tiền xử lí (request gửi đến) và hậu xử lí (response trả về).

## 2. Chức năng của Filter



Hình trên mô tả một số filter dùng để thao tác với tất cả request và response. Có thể thấy các filter được sắp xếp theo thứ tự, request phải đi qua lần lượt từng filter mới tới được endpoint; tương tự đối với response cũng phải đi qua lần lượt từng filter mới được chính thức trả về cho client. Tại mỗi filter, sẽ có code Java thao tác (can thiệp) cụ thể với request hoặc response.

```
HttpServletRequest request;
.....
String url=request.getServletPath();
String URI = ((HttpServletRequest)request).getRequestURI();
```



### 3. Khai báo sử dụng Filter

Tạo lớp Filter:

```
import jakarta.servlet.Filter;
import jakarta.servlet.FilterChain;
import jakarta.servlet.FilterConfig;
```

```
public class LogFilter implements Filter{}
```

Các thuộc tính:

```
private static final boolean debug = true;
private FilterConfig filterConfig = null;
```

Các Phương thức ghi đè:

1. `public void init(FilterConfig filterConfig)`
2. `public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException`
3. `public void destroy()`

Ngoài ra còn có các phương thức khác (nếu cần):

1. `public void setFilterConfig(FilterConfig filterConfig)`
2. `private void doBeforeProcessing(ServletRequest request, ServletResponse response) throws IOException, ServletException`
3. `private void doAfterProcessing(ServletRequest request, ServletResponse response) throws IOException, ServletException`
4. `public FilterConfig getFilterConfig()`
5. `public void log(String msg) {  
 filterConfig.getServletContext().log(msg);  
}`

LogFilter là một class thực thi interface Filter, gồm 3 hàm:

- `init`: Được gọi bởi container, khi Filter được sử dụng.
- `doFilter`: Được gọi bởi container, mỗi khi có request/response phải đi qua Filter đó.
- `destroy`: Được gọi bởi container, khi Filter bị loại bỏ, không được sử dụng.

| @WebFilter annotation   | web.xml  |
|---|--|
| @WebFilter(filterName = "HomeFilter",<br>urlPatterns = {"//*"}) | <pre>&lt;filter&gt;   &lt;filter-name&gt;HomeFilter&lt;/filter-name&gt;   &lt;filter-class&gt;filter.HomeFilter&lt;/filter-class&gt; &lt;/filter&gt;</pre> |

|   |   |
|---|---|
|   | <pre> &lt;filter-mapping&gt;   &lt;filter-name&gt;HomeFilter&lt;/filter-name&gt;   &lt;url-pattern&gt;/*&lt;/url-pattern&gt; &lt;/filter-mapping&gt; </pre>   |
| <pre> @WebFilter(filterName = "HomeFilter", urlPatterns = {"/categories/*","/products/*"})  /list,/index.jsp </pre> | <pre> &lt;filter&gt;   &lt;filter-name&gt;MyFilter&lt;/filter-name&gt;   &lt;filter-class&gt;myservlet.filter.MyFilter&lt;/filter-class&gt; &lt;/filter&gt;  &lt;filter&gt;   &lt;filter-name&gt;AnotherFilter&lt;/filter-name&gt;   &lt;filter-class&gt;myservlet.filter.AnotherFilter&lt;/filter- class&gt; &lt;/filter&gt;  &lt;filter-mapping&gt;   &lt;filter-name&gt;MyFilter&lt;/filter-name&gt;   &lt;url-pattern&gt;/*&lt;/url-pattern&gt; &lt;/filter-mapping&gt;  &lt;filter-mapping&gt;   &lt;filter-name&gt;AnotherFilter&lt;/filter-name&gt;   &lt;url-pattern&gt;/categories/*&lt;/url-pattern&gt;   &lt;url-pattern&gt;/products/*&lt;/url-pattern&gt; &lt;/filter-mapping&gt; </pre> |

#### 4. Ví dụ

##### Bài 1:

Yêu cầu như sau: có trang home, nếu từ url chúng ta gõ vào bất kỳ trang jsp nào cũng sẽ chuyển về home??

Ta dùng filter như sau:

home.jsp và 1 số trang .jsp

HomeServlet.java (/home)

Trừ file này: log.jsp

HomeFilter.java

```

Viết vào HomeFilter có: urlPatterns={"/*"} – tất cả các trang đều đi qua filter này
HttpServletRequest req=(HttpServletRequest)request;
HttpServletResponse res=(HttpServletResponse)response;
String url=req.getServletPath();
if(url.endsWith(".jsp") && !url.contains("log.jsp")){
    res.sendRedirect("home");
}else {
    req.getRequestDispatcher("log.jsp").forward(req, res);
}

```

Tính số người truy cập:

Khai báo:

```
private int counter;  
Khởi tạo (trong phương thức init)  
counter=0;  
Viết vào doFilter  
HttpSession session=req.getSession();  
counter++;  
session.setAttribute("counter", counter);
```

Có thể lưu vào web.xml

```
<init-param>  
    <param-name>counter</param-name>  
    <param-value>0</param-value>  
</init-param>
```

Sau đó đọc ra:

```
counter=Integer.parseInt(filterConfig.getInitParameter("counter"));
```

← → ↺ ⓘ localhost:9999/ex/home

# Day la Home page

## số người dùng truy cập:3

### Bài 2: Thứ tự filter

Thứ tự của các tag <filter-mapping> chính là thứ tự mà container sẽ áp dụng cho endpoint. Như ví dụ trên, container sẽ áp dụng Filter1 trước, sau đó mới đến cho Filter2.

```
<filter>  
    <filter-name>Filter1</filter-name>  
    <filter-class>filter.Filter1</filter-class>  
    <init-param>  
        <param-name>name</param-name>  
        <param-value>toi la filter 1</param-value>  
    </init-param>  
</filter>  
<filter>  
    <filter-name>Filter2</filter-name>  
    <filter-class>filter.Filter2</filter-class>  
    <init-param>  
        <param-name>name</param-name>  
        <param-value>toi la filter 2</param-value>  
    </init-param>  
</filter>  
<filter-mapping>  
    <filter-name>Filter1</filter-name>  
    <url-pattern>/home</url-pattern>  
</filter-mapping>  
<filter-mapping>  
    <filter-name>Filter2</filter-name>  
    <url-pattern>/home</url-pattern>
```

```
String name  
=filterConfig.getInitParameter("name");  
    System.out.println(name);
```

|                   |  |
|-------------------|--|
| </filter-mapping> |  |
|-------------------|--|

### Bài 3: Authentication dùng filter

|  |  |
|--|--|
| <p>LoginFilter có</p> <pre>&lt;filter-mapping&gt;   &lt;filter-name&gt;LoginFilter&lt;/filter-name&gt;   &lt;url-pattern&gt;/home&lt;/url-pattern&gt;   &lt;url-pattern&gt;/home.jsp&lt;/url-pattern&gt; &lt;/filter-mapping&gt;</pre> | <pre>HttpServletRequest req=(HttpServletRequest)request; HttpServletResponse res=(HttpServletResponse)response; HttpSession session=req.getSession(); if(session.getAttribute("account")==null){ req.getRequestDispatcher("login.jsp").forward(request, response); }</pre> |
|--|--|

### Bài 4: Tạo một ứng dụng Login đơn giản và bảo mật với Java Servlet Filter

Bảo mật (Security) là một khía cạnh quan trọng của một ứng dụng có sự vận chuyển các dữ liệu quan trọng trên Internet.

Authentication (Xác thực)

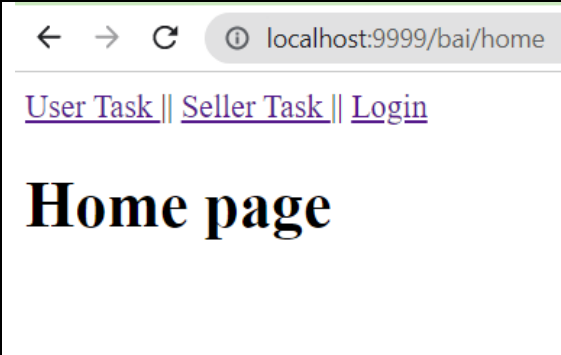
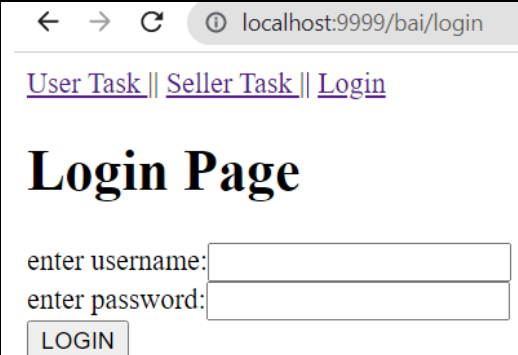
Xác thực là quá trình mà quyền truy cập (access privileges) của người dùng được xác minh trước khi họ vào khu vực được bảo vệ của Website. Có hai cách tiếp cận xác thực chính: xác thực cơ bản và xác thực dựa trên biểu mẫu (Form-based authentication).

Basic Authentication (Xác thực cơ bản).

Với xác thực cơ bản, người dùng có thể truy cập mọi trang (page) hoàn toàn bình thường, với các trang yêu cầu bảo mật, một cửa sổ sẽ hiển thị để người dùng nhập vào username/password của họ. Thông tin username/password sẽ được gói lại gửi kèm theo yêu cầu (request) đến Server.

Form-based Authentication (Xác thực dựa trên biểu mẫu)

Hầu hết các Website sử dụng hình thức xác thực dựa trên biểu mẫu (Form-based Authentication). Website cho phép người dùng truy cập mọi trang thông thường mà không yêu cầu mật khẩu. Tuy nhiên nếu người dùng truy cập vào một trang được bảo vệ, nó sẽ chuyển hướng tới một trang đăng nhập.

|   |  |
|---|--|
|  |  |
|---|--|

Trong bài học này, cô sẽ sử dụng Servlet Filter để bảo mật ứng dụng Java Web.

Trong bảo mật, có 2 khái niệm quan trọng là Principal (Quyền hạn) và Role (vai trò).

Role (Vai trò) là một tập hợp các quyền (permission) đối với một ứng dụng.

Để đơn giản chúng ta có ví dụ, ứng dụng có 2 vai trò "USER" (người dùng) và "SELLER" (Người bán hàng).

Vai trò "USER" được phép mua hàng, và xem thông tin của mình.

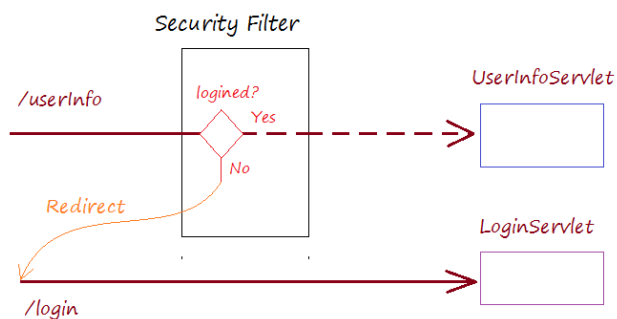
Vai trò "SELLER" được phép bán hàng, và xem thông tin của mình.

Principal (quyền hạn) có thể tạm hiểu là một "Chủ thể" sau khi đã đăng nhập vào một hệ thống, họ có quyền làm điều gì đó trong hệ thống. Một "Chủ thể" có thể có một hoặc nhiều vai trò. Điều này phụ thuộc vào sự phân quyền của ứng dụng cho mỗi tài khoản người dùng khác nhau.

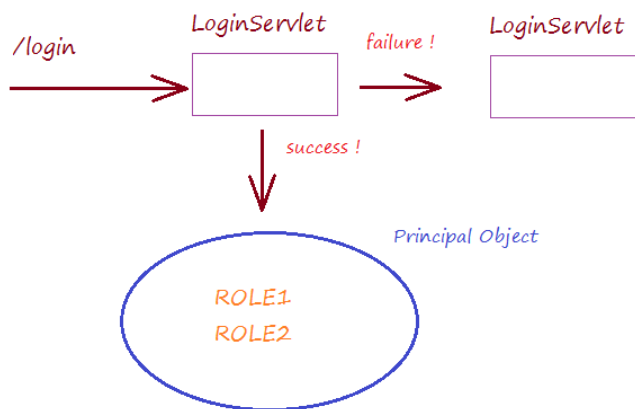
Trong ứng dụng Java Servlet, một Servlet Filter đặc biệt được sử dụng để xử lý bảo mật, nó thường được gọi là Security Filter.

Dưới đây là nguyên tắc hoạt động của Security Filter.

Khi người dùng truy cập vào một trang (page) được bảo vệ, Security Filter sẽ kiểm tra, nếu người dùng chưa đăng nhập, yêu cầu của người dùng sẽ bị chuyển hướng (redirect) sang trang đăng nhập.



Nếu người dùng đã đăng nhập thành công, một đối tượng Principal được tạo ra, nó mang các thông tin của người dùng, bao gồm cả các vai trò.



Nếu người dùng đã đăng nhập thành công trước đó, và truy cập vào một trang (page) được bảo vệ. Security Filter sẽ kiểm tra các vai trò của người dùng có phù hợp để truy cập vào trang này hay không. Nếu không hợp lệ, nó sẽ hiển thị cho người dùng một trang thông báo truy cập bị cấm (Access Denied).



|  |
|--|
| <pre>private String username,password; private int role;</pre>   |
| <p>DAO.java</p> <pre>public Admin check(String username, String password) {     String sql = "SELECT [Username]\n"         + "      ,[Password]\n"         + "      ,[role]\n"         + " FROM [dbo].[Admin]\n"         + " where [Username]=? and [Password]=?";     try{         PreparedStatement st=connection.prepareStatement(sql);         st.setString(1, username);         st.setString(2, password);         ResultSet rs=st.executeQuery();         if(rs.next()){             return new Admin(rs.getString("Username"),                 rs.getString("Password"),                 rs.getInt("role"));         }     }catch(SQLException e){      }     return null; }</pre> |

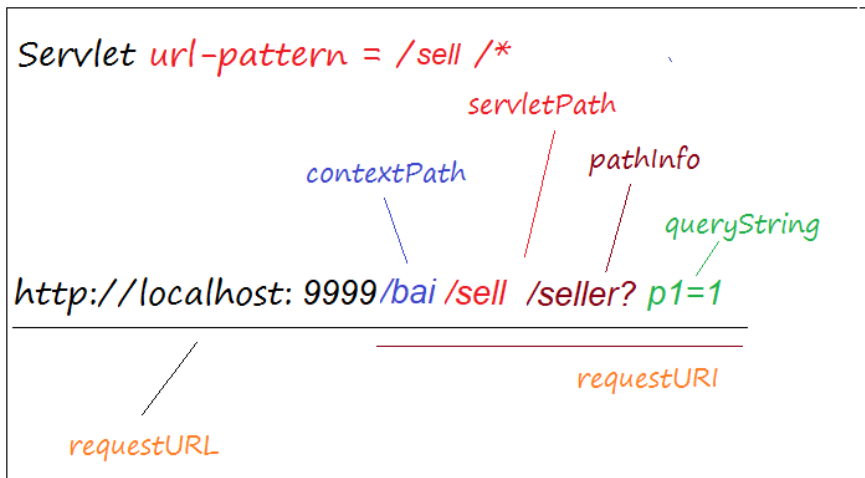
Có form:

|  |
|--|
| <p>login.jsp</p> <pre>&lt;jsp:include page="menu.jsp"/&gt; &lt;h1&gt;Login Page&lt;/h1&gt; &lt;h3 style="color: red"&gt;\${requestScope.ms}&lt;/h3&gt; &lt;form action="login" method="post"&gt;     enter username:&lt;input type="text" name="user"/&gt;&lt;br/&gt;     enter password:&lt;input type="password" name="pass"/&gt;&lt;br/&gt;     &lt;input type="submit" value="LOGIN"/&gt; &lt;/form&gt;</pre>  |
| <p>Servlet: LoginServlet.java và url /login</p>  |
| <p>doGet</p> <pre>request.getRequestDispatcher("login.jsp").forward(request, response);</pre>  |
| <p>doPost</p> <pre>String u=request.getParameter("user"); String p=request.getParameter("pass"); DAO d=new DAO(); Admin a=d.check(u, p); if(a==null){     request.setAttribute("ms", "username or password invalid!!! ");     request.getRequestDispatcher("login.jsp").forward(request, response); }else{     //co roi     HttpSession session=request.getSession();     session.setAttribute("account", a);     response.sendRedirect("home"); }</pre> |



}

Các page lần lượt



|  |
|--|
| home.jsp   |
| <code>&lt;jsp:include page="menu.jsp"/&gt;</code><br><code>&lt;h1&gt;Home page&lt;/h1&gt;</code>   |
| user.jsp   |
| <code>&lt;jsp:include page="menu.jsp"/&gt;</code><br><code>&lt;h1&gt;Trang người dùng&lt;/h1&gt;</code>  |
| Info.jsp   |
| <code>&lt;jsp:include page="menu.jsp"/&gt;</code><br><code>&lt;h1&gt;Profile page&lt;/h1&gt;</code><br><code>&lt;h3&gt;</code><br>Username: <code>\${sessionScope.account.username}</code> <code>&lt;br/&gt;</code><br><code>&lt;c:if test="\${sessionScope.account.role==1}"&gt;</code><br>CHÀO MỪNG BẠN ĐẾN VỚI KÊNH BÁN HÀNG!<br><code>&lt;/c:if&gt;</code><br><code>&lt;c:if test="\${sessionScope.account.role==2}"&gt;</code><br>CHÀO MỪNG BẠN KHÁCH HÀNG!<br><code>&lt;/c:if&gt;</code><br><code>&lt;/h3&gt;</code> |
| seller.jsp (folder sell)   |
| <code>&lt;jsp:include page="../menu.jsp"/&gt;</code><br><code>&lt;h1&gt;Trang bán hàng&lt;/h1&gt;</code>   |
| dined.jsp  |
| <code>&lt;jsp:include page="menu.jsp"/&gt;</code><br><code>&lt;h1&gt;Access Denied!&lt;/h1&gt;</code>  |

Các servlet:

|   |
|---|
| HomeServlet.java url /home  |
| <code>request.getRequestDispatcher("home.jsp").forward(request, response);</code> |
| InfoServlet.java url /info  |
| <code>request.getRequestDispatcher("info.jsp").forward(request, response);</code> |
| LogoutServlet.java url /logout  |
| <code>HttpSession session=request.getSession();</code>                            |

|  |
|--|
| <pre> if(session.getAttribute("account")!=null){     session.removeAttribute("account"); } response.sendRedirect("home"); </pre> |
| <pre> SellerServlet.java url / sellerTask request.getRequestDispatcher("sell/seller.jsp").forward(request, response); </pre>     |
| <pre> UserServlet.java url /userTask request.getRequestDispatcher("user.jsp").forward(request, response); </pre>                 |
| <pre> DinedServlet.java url /dined request.getRequestDispatcher("dined.jsp").forward(request, response); </pre>                  |

Các Filter :

|  |
|--|
| <pre> HomeFilter.java &lt;filter-mapping&gt;     &lt;filter-name&gt;HomeFilter&lt;/filter-name&gt;     &lt;url-pattern&gt;/*&lt;/url-pattern&gt; &lt;/filter-mapping&gt; </pre>  |
| <pre> HttpServletRequest req=(HttpServletRequest)request; HttpServletResponse res=(HttpServletResponse)response; String url=req.getServletPath(); if(url.endsWith(".jsp")){     res.sendRedirect("home"); } </pre>   |
| <pre> LoginFilter.java &lt;filter-mapping&gt;     &lt;filter-name&gt;LoginFilter&lt;/filter-name&gt;     &lt;url-pattern&gt;/userTask&lt;/url-pattern&gt;     &lt;url-pattern&gt;/sellerTask&lt;/url-pattern&gt;     &lt;url-pattern&gt;/info&lt;/url-pattern&gt; &lt;/filter-mapping&gt; </pre>                           |
| <pre> HttpServletRequest req=(HttpServletRequest)request; HttpServletResponse res=(HttpServletResponse)response; HttpSession session=req.getSession(); if(session.getAttribute("account")==null){     res.sendRedirect("login"); } </pre>  |
| <pre> RoleFilter.java &lt;filter-mapping&gt;     &lt;filter-name&gt;RoleFilter&lt;/filter-name&gt;     &lt;url-pattern&gt;/sellerTask&lt;/url-pattern&gt; &lt;/filter-mapping&gt; </pre>   |
| <pre> HttpServletRequest req=(HttpServletRequest)request; HttpServletResponse res=(HttpServletResponse)response; HttpSession session=req.getSession(); if(session.getAttribute("account")==null){     res.sendRedirect("login"); }else{     Admin a=(Admin)session.getAttribute("account");     if(a.getRole()==1){ </pre> |

```
        req.getRequestDispatcher("sell/seller.jsp").forward(request, response);  
    }else{  
        res.sendRedirect("dined");  
    }  
}
```