

## Lab 5 - Filtros (Filter) e Eventos (Listener)

O objetivo deste laboratório é praticar as funções dos Filtros em uma aplicação Web, especialmente para demonstrar para que serve a filtragem junto ao manuseio do seu ciclo de vida. No primeiro exercício, você irá adicionar um filtro para a aplicação Web e observar a forma como o método **doFilter()** e os métodos de filtros são invocados. No segundo exercício, você irá adicionar eventos manipuladores de ciclo de vida para a mesma aplicação e observar como os tratadores de eventos são invocados.

### Exercícios

**Exercício 1:** Criar e configurar filtros na aplicação.

**Exercício 2:** Criar e configurar captura de eventos na aplicação.

### Exercício 1 - Criar e configurar filtros na aplicação

1. Vamos criar um pacote **threeway.projeto.filter**, em **src** no projeto **Banco-WebView**.
2. Crie um classe java dentro do pacote criado no tópico 1, com o nome **SessionFilter** e adicione o código abaixo.

```
/**
 * Servlet Filter implementation class SessionFilter
 */
public class SessionFilter implements Filter {

    /**
     * Default constructor.
     */
    public SessionFilter() {}

    /**
     * @see Filter#destroy()
     */
    public void destroy() {}

    /**
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {

        System.out.println("Ola, o filtro de verificar sessão esta ativo!");

        HttpServletRequest req = (HttpServletRequest) request;

        HttpServletResponse resp = (HttpServletResponse) response;

        String usuarioSession = (String) req.getSession().getAttribute("usuario");

        if (usuarioSession == null || usuarioSession.isEmpty()) {

            System.out.println("Você não esta logado, não pode acessar a página: " + req.getRequestURI());

            resp.sendRedirect("login.jsp");
        } else {

            chain.doFilter(request, response);
        }
    }
}
```

```

    }
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
}

```

3. Filtros, listeners e Servlets tem a mesma forma de mapeamento na aplicação, ou seja, podem ser mapeados via anotação ou no **web.xml**, veja o exemplo de filtro abaixo.

### 3.1. Por anotação

```

@WebFilter(filterName="/SessionFilter", urlPatterns={"/index.jsp", "/manterCliente.jsp", "/operacoesBancarias.jsp" })
public class SessionFilter implements Filter {

```

### 3.2. Através do Deployment Descriptor( **web.xml**).

```

<filter>
  <filter-name>SessionFilter</filter-name>
  <filter-class>threeway.projeto.filter.SessionFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>SessionFilter</filter-name>
  <url-pattern>/index.jsp</url-pattern>
  <url-pattern>/manterCliente.jsp</url-pattern>
  <url-pattern>/operacoesBancarias.jsp</url-pattern>
</filter-mapping>

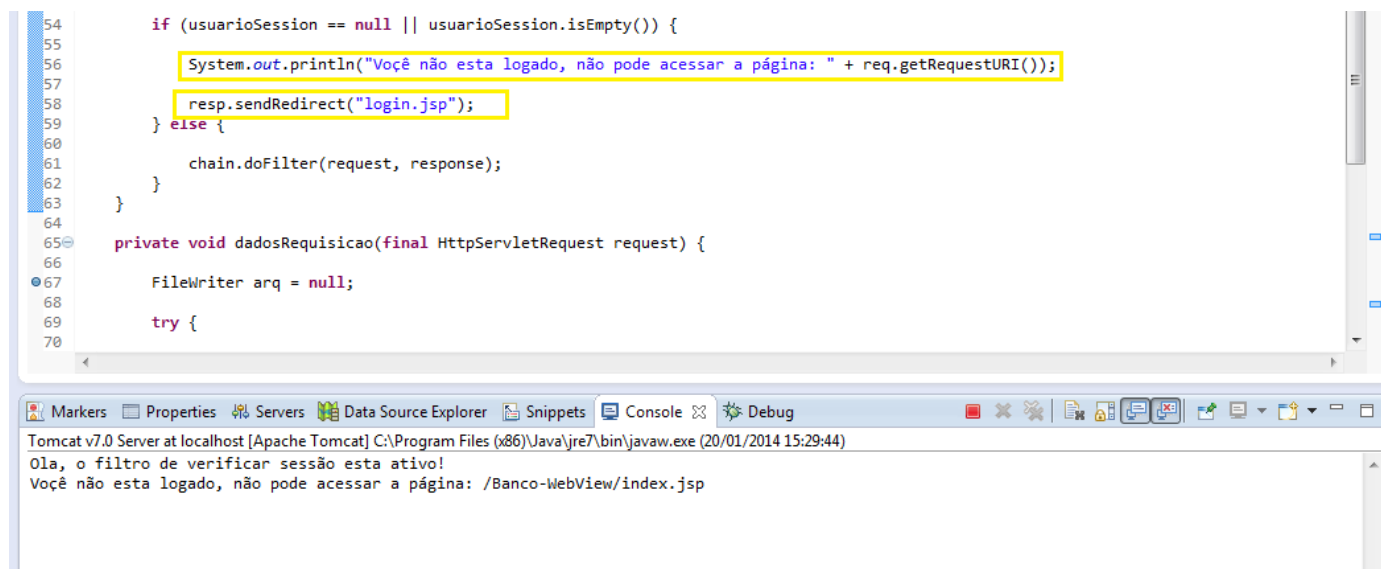
```

**urlPatterns:** referente a url que o filtro controlará, poderíamos ter criado uma pasta chamada *pages* e colocado todas nossas paginas jsp dentro desta pasta, depois poderíamos mapear todo o diretório *pages* como segue:  
`<url-pattern>/pages/*</url-pattern>`.

4. Adotaremos o mapeamento por **Annotations**, então altere a classe **SessionFilter** para que fique conforme o exemplo 3.1.
5. Depois de criar o filtro, reinicie o servidor e acesse a url abaixo sem realizar o login.

<http://localhost:8080/Banco-WebView/index.jsp>

Obs.: Antes de implementar o filtro o sistema permitia acessar qualquer página sem autenticar, veja no console do eclipse o resultado do filtro, veja que é impresso no console as mensagens note também no código marcado em amarelo a mensagem é impressa e logo após a aplicação é redirecionada para o login.



The screenshot shows the Eclipse IDE with a Java file open. The code is as follows:

```

54     if (usuarioSession == null || usuarioSession.isEmpty()) {
55         System.out.println("Você não esta logado, não pode acessar a página: " + req.getRequestURI());
56         resp.sendRedirect("login.jsp");
57     } else {
58         chain.doFilter(request, response);
59     }
60 }
61
62 private void dadosRequisicao(final HttpServletRequest request) {
63     FileWriter arq = null;
64     try {
65
66     }
67 }
68
69
70

```

The console output at the bottom shows:

```

Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20/01/2014 15:29:44)
Ola, o filtro de verificar sessão esta ativo!
Você não esta logado, não pode acessar a página: /Banco-WebView/index.jsp

```

6. Vamos adicionar mais uma funcionalidade no filtro, iremos criar um gerenciamento de log semelhante ao do servidor **tomcat** onde o container mantém logs de acesso a aplicação dentro de arquivos txt no diretório **apache-tomcat-7.0.47/logs**.

- 6.1. Vamos mapear o nome do diretório onde será salvo o arquivo. Crie uma pasta na unidade `C://` (para sistema operacional Windows) com o nome **banco-web-log**, depois acesse o **web.xml** e adicione o contexto abaixo.

```
<context-param>
  <param-name>dirLog</param-name>
  <param-value>c://banco-web-log/</param-value> <!-- caminho de onde foi criado a pasta -->
</context-param>
```

- 6.2. Abra o arquivo **SessionFilter** e altere o adicionando o código abaixo.

```
private void gerarLogTxt(final HttpServletRequest request) {
    try {
        FileWriter arquivo = null;

        //obtem o diretório mapeado no web.xml, esse diretório está no contexto da aplicação, pode ser recuperado
        //em qualquer lugar a através do getServletContext();
        String diretorio = (String) request.getServletContext().getInitParameter("dirLog");

        //Criar um arquivo com o nome access-log.txt dentro do diretorio mapeado no web.xml
        arquivo = new FileWriter(diretorio + "access-log.txt", true);

        //Criar o objeto para a escrita dentro do arquivo access-log
        PrintWriter escreverFile = new PrintWriter(arquivo);

        //escreve a string dentro do arquivo access-log
        escreverFile.printf("[Data|Url Requisitada|Ip solicitante]: " + new Date() + " | " + request.getRequestURI() + " | " +
            request.getRemoteAddr() + "\n");

        arquivo.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

- 6.3. Agora altere o método **doFilter()** adicionando a chamada do método **geraLogTxt()**.

```
gerarLogTxt(req);

chain.doFilter(request, response);
```

- 6.4. Faça o **import** das bibliotecas dentro da classe **SessionFilter** com o comando `ctrl+sift+8`, salve as alterações e reinicie o servidor. Acesse a aplicação, clique nas abas **Manter Cliente** e **Operações Bancárias**, depois veja o log que foi gerado no arquivo **access-log.txt** dentro do diretório **banco-web-log**, semelhante ao código abaixo.

```
[Data|Url Requisitada|Ip solicitante]: Mon Jan 20 16:30:04 BRST 2014 | /Banco-WebView/index.jsp | 10.62.0.134
[Data|Url Requisitada|Ip solicitante]: Mon Jan 20 16:30:05 BRST 2014 | /Banco-WebView/manterCliente.jsp | 10.62.0.134
[Data|Url Requisitada|Ip solicitante]: Mon Jan 20 16:32:28 BRST 2014 | /Banco-WebView/operacoesBancarias.jsp | 10.62.0.134
[Data|Url Requisitada|Ip solicitante]: Mon Jan 20 16:34:15 BRST 2014 | /Banco-WebView/index.jsp | 10.62.0.136
[Data|Url Requisitada|Ip solicitante]: Mon Jan 20 16:34:19 BRST 2014 | /Banco-WebView/operacoesBancarias.jsp | 10.62.0.136
```

Fizemos a simulação do **access-log** do **apache tomcat** agora você pode manter o historio de quem acessou a aplicação e o que fez através do ip e url solicitada.

## Exercício 2 - Criar e configurar captura de eventos na aplicação.

1. Crie um novo pacote com o nome **threeway.projeto.listeners**.
2. Crie uma nova classe JAVA com o nome **LifeCycleListener** e adicione o código para que fique conforme abaixo.

```
package threeway.projeto.listeners;

@WebListener
public class LifeCycleListener implements ServletContextListener, ServletContextAttributeListener,
HttpSessionListener {
```

3. Note que o filtro foi mapeado por anotação e que implementa as **interfaces ServletContextListener, ServletContextAttributeListener e HttpSessionListener**, altere o listener adicionando os métodos conforme as especificações abaixo.

3.1. **ServletContextAttributeListener**, contem o métodos abaixo que devem ser implementados.

```
public void contextInitialized(ServletContextEvent sce) {  
    System.out.println("O container foi inicializado!");  
    System.out.println("Nome do Servidor: " + sce.getServletContext().getServerInfo());  
}  
  
public void contextDestroyed(ServletContextEvent sce) {  
    System.out.println("O container foi paralizado!");  
}
```

3.2. **ServletContextAttributeListener**, métodos que gerenciam os atributos do contexto.

```
public void attributeAdded(ServletContextAttributeEvent sca) {  
    System.out.println("Atributo Criado: " + sca.getServletContext().getInitParameter("dirLog"));  
}  
  
public void attributeReplaced(ServletContextAttributeEvent sca) {}  
  
public void attributeRemoved(ServletContextAttributeEvent sca) {}
```

3.3. **HttpSessionListener**, métodos que gerenciam o ciclo de vida da sessão.

```
@Override  
public void sessionCreated(HttpSessionEvent hse) {  
    System.out.println("Sessão criada!");  
}  
  
@Override  
public void sessionDestroyed(HttpSessionEvent hse) {  
    System.out.println("Sessão Destruida!");  
}
```

4. Agora reinicie o servidor e acompanhe os logs gerados no console do eclipse.

Stop Apache Tomcat:

```
O container foi paralizado!  
Jan 20, 2014 5:07:05 PM org.apache.coyote.AbstractProtocol stop  
Informações: Stopping ProtocolHandler ["http-bio-8080"]
```

Start Apache Tomcat:

```
Informações: Starting Servlet Engine: Apache Tomcat/7.0.47  
O container foi inicializado!  
Nome do Servidor: Apache Tomcat/7.0.47  
Atributo Criado: c://banco-web-log/  
Jan 20, 2014 5:09:50 PM org.apache.coyote.AbstractProtocol start  
Informações: Starting ProtocolHandler ["http-bio-8080"]  
Jan 20, 2014 5:09:50 PM org.apache.coyote.AbstractProtocol start  
Informações: Starting ProtocolHandler ["ajp-bio-8009"]  
Jan 20, 2014 5:09:50 PM org.apache.catalina.startup.Catalina start  
Informações: Server startup in 610 ms
```

Criar Sessão: acesse a aplicação de um navegador diferente ou limpe os dados de sessão do navegador aberto, não é necessário fazer o login veja o log no console do eclipse.

```
Informações: Starting Servlet Engine: Apache Tomcat/7.0.47  
O container foi inicializado!  
Nome do Servidor: Apache Tomcat/7.0.47
```

```
Atributo Criado c://banco-web-log/  
Jan 20, 2014 5:18:21 PM org.apache.coyote.AbstractProtocol start  
Informações: Starting ProtocolHandler ["http-bio-8080"]  
Jan 20, 2014 5:18:21 PM org.apache.coyote.AbstractProtocol start  
Informações: Starting ProtocolHandler ["ajp-bio-8009"]  
Jan 20, 2014 5:18:21 PM org.apache.catalina.startup.Catalina start  
Informações: Server startup in 610 ms  
Atributo Criado c://banco-web-log/  
Sessão criada!
```