Java Servlet Filter Example Tutorial

This is the fourth article in the series of Web Applications Tutorial, you might want to check out earlier articles too.

1. Java Web Application
2. Java Servlet Tutorial
3. Servlet Session Management

In this article, we will lean about the Filter in servlet. We will look into various usage of filter, how can we create filter and learn it's usage with a simple web application.

1. Why do we have Servlet Filter?

In the last article, we learned how we can manage session in web application and if we want to make sure that a resource is accessible only when user session is valid, we can achieve this using servlet session attributes. The approach is simple but if we have a lot of servlets and jsps, then it will become hard to maintain because of redundant code. If we want to change the attribute name in future, we will have to change all the places where we have session authentication.

That's why we have servlet filter. Servlet Filters are **pluggable** java components that we can use to intercept and process requests *before* they are sent to servlets and response *after* servlet code is finished and before container sends the response back to the client.

Some common tasks that we can do with filters are:

- Logging request parameters to log files.
- Authentication and autherization of request for resources.
- Formatting of request body or header before sending it to servlet.
- Compressing the response data sent to the client.
- Alter response by adding some cookies, header information etc.

As I mentioned earlier, **servlet filters are pluggable** and configured in deployment descriptor

(web.xml) file. Servlets and filters both are unaware of each other and we can add or remove a filter just by editing web.xml.

We can have multiple filters for a single resource and we can create a chain of filters for a single resource in web.xml. We can create a Servlet Filter by implementing `javax.servlet.Filter` interface.

2. Filter interface

Filter interface is similar to Servlet interface and we need to implement it to create our own servlet filter. Filter interface contains lifecycle methods of a Filter and it's managed by servlet container.

Filter interface lifecycle methods are:

A. **void init(FilterConfig paramFilterConfig)** – When container initializes the Filter, this is the method that gets invoked. This method is called only once in the lifecycle of filter and we should initialize any resources in this method. **FilterConfig** is used by container to provide init parameters and servlet context object to the Filter. We can throw ServletException in this method.
B. **doFilter(ServletRequest paramServletRequest, ServletResponse paramServletResponse, FilterChain paramFilterChain)** – This is the method invoked every time by container when it has to apply filter to a resource. Container provides request and response object references to filter as argument. **FilterChain** is used to invoke the next filter in the chain. This is a great example of **Chain of Responsibility Pattern**.
C. **void destroy()** – When container offloads the Filter instance, it invokes the destroy() method. This is the method where we can close any resources opened by filter. This method is called only once in the lifetime of filter.

3. WebFilter annotation

`javax.servlet.annotation.WebFilter` was introduced in Servlet 3.0 and we can use this annotation to declare a servlet filter. We can use this annotation to define init parameters, filter name and description, servlets, url patterns and dispatcher types to apply the filter. If you make frequent changes to the filter configurations, its better to use web.xml because that will not require you to recompile the filter class.

Read: **Java Annotations Tutorial**

4. Filter configuration in web.xml

We can declare a filter in web.xml like below.

```
<filter>
  <filter-name>RequestLoggingFilter</filter-name> <!-- mandatory -->
  <filter-class>com.journaldev.servlet.filters.RequestLoggingFilter</filter-class> <!-- mandatory
  <init-param> <!-- optional -->
  <param-name>test</param-name>
  <param-value>testValue</param-value>
  </init-param>
</filter>
```

We can map a Filter to servlet classes or url-patterns like below.

```
<filter-mapping>
  <filter-name>RequestLoggingFilter</filter-name> <!-- mandatory -->
  <url-pattern>/*</url-pattern> <!-- either url-pattern or servlet-name is mandatory -->
  <servlet-name>LoginServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

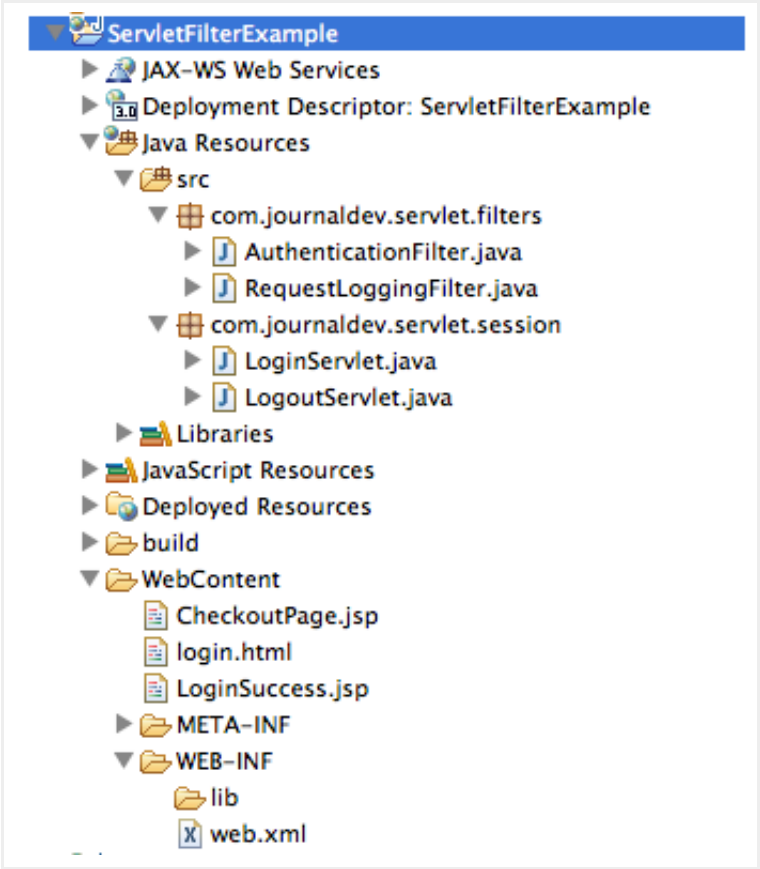
Note: While creating the filter chain for a servlet, container first processes the url-patterns and then servlet-names, so if you have to make sure that filters are getting executed in a particular order, give extra attention while defining the filter mapping.

Servlet Filters are generally used for client requests but sometimes we want to apply filters with **RequestDispatcher** also, we can use dispatcher element in this case, the possible values are REQUEST, FORWARD, INCLUDE, ERROR and ASYNC. If no dispatcher is defined then it's applied only to client requests.

5. Servlet Filter Example for Logging and session validation

In our **servlet filter example**, we will create filters to log request cookies and parameters and validate session to all the resources except static HTMLs and LoginServlet because it will not have a session.

We will create a dynamic web project **ServletFilterExample** whose project structure will look like below image.



login.html is the entry point of our application where user will provide login id and password for authentication.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="US-ASCII">
<title>Login Page</title>
</head>
<body>

<form action="LoginServlet" method="post">

Username: <input type="text" name="user">
<br>
Password: <input type="password" name="pwd">
<br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

LoginServlet is used to authenticate the request from client for login.

```
package com.journaldev.servlet.session;

import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private final String userID = "admin";
	private final String password = "password";

	protected void doPost(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {

		// get request parameters for userID and password
		String user = request.getParameter("user");
		String pwd = request.getParameter("pwd");

		if(userID.equals(user) && password.equals(pwd)){
			HttpSession session = request.getSession();
			session.setAttribute("user", "Pankaj");
			//setting session to expiry in 30 mins
			session.setMaxInactiveInterval(30*60);
			Cookie userName = new Cookie("user", user);
			userName.setMaxAge(30*60);
			response.addCookie(userName);
			response.sendRedirect("LoginSuccess.jsp");
		}else{
			RequestDispatcher rd = getServletContext().getRequestDispatcher("/login.ht
			PrintWriter out= response.getWriter();
			out.println("<font color=red>Either user name or password is wrong.</font>
			rd.include(request, response);
		}

	}

}
```

When client is authenticated, it's forwarded to LoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.d
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
//allow access only if session exists
String user = (String) session.getAttribute("user");
String userName = null;
String sessionID = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
        if(cookie.getName().equals("user")) userName = cookie.getValue();
        if(cookie.getName().equals("JSESSIONID")) sessionID = cookie.getValue();
}
}
%>
<h3>Hi <%=userName %>, Login successful. Your Session ID=<%=sessionID %></h3>
<br>
User=<%=user %>
<br>
<a href="CheckoutPage.jsp">Checkout Page</a>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

Notice that there is no session validation logic in the above JSP. It contains link to another JSP page, CheckoutPage.jsp.

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.d
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
String userName = null;
String sessionID = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
        if(cookie.getName().equals("user")) userName = cookie.getValue();
}
}
```

```
%>
<h3>Hi <%=userName %>, do the checkout.</h3>
<br>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

LogoutServlet is invoked when client clicks on Logout button in any of the JSP pages.

```
package com.journaldev.servlet.session;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Servlet
        response.setContentType("text/html");
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
        for(Cookie cookie : cookies){
                if(cookie.getName().equals("JSESSIONID")){
                        System.out.println("JSESSIONID="+cookie.getValue());
                        break;
                }
        }
        }
        //invalidate the session if exists
        HttpSession session = request.getSession(false);
        System.out.println("User="+session.getAttribute("user"));
        if(session != null){
                session.invalidate();
        }
        response.sendRedirect("login.html");
    }

}
```

Now we will create logging and authentication filter classes.

```
package com.journaldev.servlet.filters;

import java.io.IOException;
import java.util.Enumeration;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;

/**
 * Servlet Filter implementation class RequestLoggingFilter
 */
@WebFilter("/RequestLoggingFilter")
public class RequestLoggingFilter implements Filter {

	private ServletContext context;

	public void init(FilterConfig fConfig) throws ServletException {
		this.context = fConfig.getServletContext();
		this.context.log("RequestLoggingFilter initialized");
	}

	public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) 
		HttpServletRequest req = (HttpServletRequest) request;
		Enumeration<String> params = req.getParameterNames();
		while(params.hasMoreElements()){
			String name = params.nextElement();
			String value = request.getParameter(name);
			this.context.log(req.getRemoteAddr() + "::Request Params::{"+name+"="+valu
		}

		Cookie[] cookies = req.getCookies();
		if(cookies != null){
			for(Cookie cookie : cookies){
				this.context.log(req.getRemoteAddr() + "::Cookie::{"+cookie.getNam
			}
		}
		// pass the request along the filter chain
		chain.doFilter(request, response);
	}

	public void destroy() {
```

```
                //we can close resources here
        }

}
```

```
package com.journaldev.servlet.filters;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter("/AuthenticationFilter")
public class AuthenticationFilter implements Filter {

        private ServletContext context;

        public void init(FilterConfig fConfig) throws ServletException {
                this.context = fConfig.getServletContext();
                this.context.log("AuthenticationFilter initialized");
        }

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

                HttpServletRequest req = (HttpServletRequest) request;
                HttpServletResponse res = (HttpServletResponse) response;

                String uri = req.getRequestURI();
                this.context.log("Requested Resource::"+uri);

                HttpSession session = req.getSession(false);

                if(session == null && !(uri.endsWith("html") || uri.endsWith("LoginServlet"))){
                        this.context.log("Unauthorized access request");
                        res.sendRedirect("login.html");
                }else{
                        // pass the request along the filter chain
                        chain.doFilter(request, response);
                }


        }
```

```
	public void destroy() {
		//close any resources here
	}

}
```

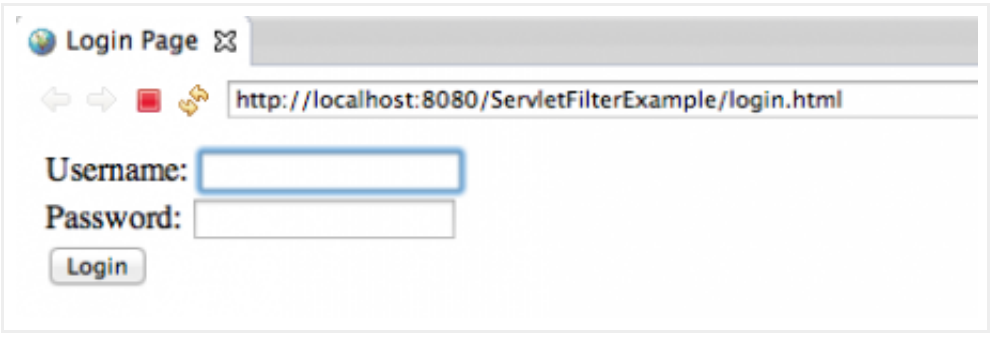
Notice that we are not authenticating any HTML page or LoginServlet. Now we will configure these filters mapping in the web.xml file.

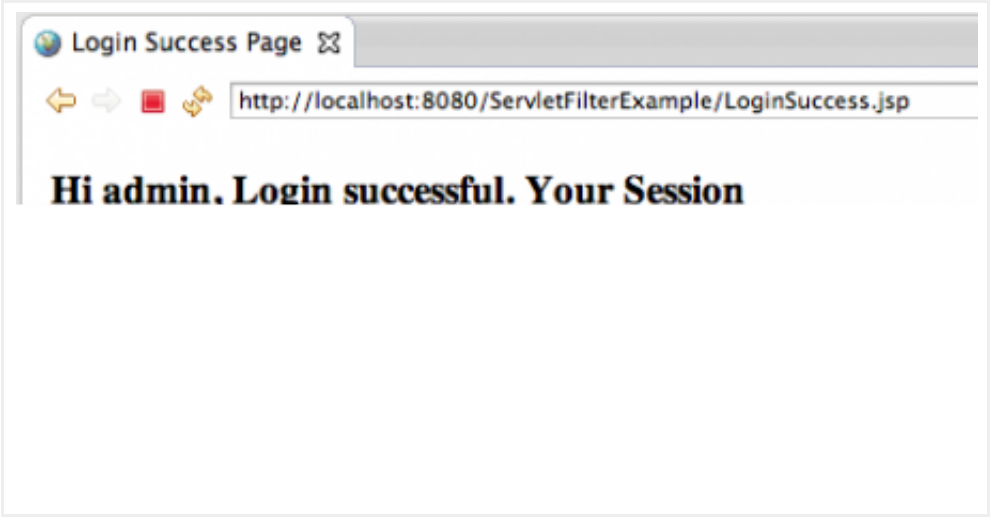
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/j
  <display-name>ServletFilterExample</display-name>
  <welcome-file-list>
    <welcome-file>login.html</welcome-file>
  </welcome-file-list>

  <filter>
    <filter-name>RequestLoggingFilter</filter-name>
    <filter-class>com.journaldev.servlet.filters.RequestLoggingFilter</filter-class>
  </filter>
  <filter>
    <filter-name>AuthenticationFilter</filter-name>
    <filter-class>com.journaldev.servlet.filters.AuthenticationFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>RequestLoggingFilter</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <filter-mapping>
    <filter-name>AuthenticationFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Now when we will run our application, we will get response pages like below images.





If you are not logged in and try to access any JSP page, you will be forwarded to the login page.

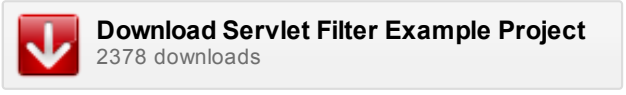
In server log file, you can see the logs written by filters as well as servlets.

```
Aug 13, 2013 1:06:07 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,B7275762B8D23121152B1270D6EB240A}
Aug 13, 2013 1:06:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/
Aug 13, 2013 1:06:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Unauthorized access request
Aug 13, 2013 1:06:07 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,B7275762B8D23121152B1270D6EB240A}
Aug 13, 2013 1:06:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/login.html
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Request Params::{pwd=password}
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Request Params::{user=admin}
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,B7275762B8D23121152B1270D6EB240A}
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/LoginServlet
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:06:43 AM org.apache.catalina.core.ApplicationContext log
```

```
INFO: Requested Resource::/ServletFilterExample/LoginSuccess.jsp
Aug 13, 2013 1:06:52 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:06:52 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:06:52 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/CheckoutPage.jsp
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/LogoutServlet
JSESSIONID=8BDF777933194EDCAC1D8F1B73633C56
User=Pankaj
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:07:00 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/login.html
Aug 13, 2013 1:07:06 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/LoginSuccess.jsp
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Unauthorized access request
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{JSESSIONID,8BDF777933194EDCAC1D8F1B73633C56}
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: 0:0:0:0:0:0:0:1%0::Cookie::{user,admin}
Aug 13, 2013 1:07:07 AM org.apache.catalina.core.ApplicationContext log
INFO: Requested Resource::/ServletFilterExample/login.html
```

Thats all for Filter in servlet, it's one of the important feature of J2EE web application and we should use it for common tasks performed by various servlets. In future posts, we will look into servlet listeners and cookies.

Update: After getting a lot of requests for the downloadable project, I have attached it to the post, download it from link below.



Check out next article in the series about Servlet Listener.

Update

Struts 2 uses Servlet Filter to intercept the client requests and forward them to appropriate action

classes, these are called Struts 2 Interceptors. Check out **Struts 2 Beginners Tutorial**.