# Servlet Filters and Servlet Listeners

# Additional Useful Servlet Functionality

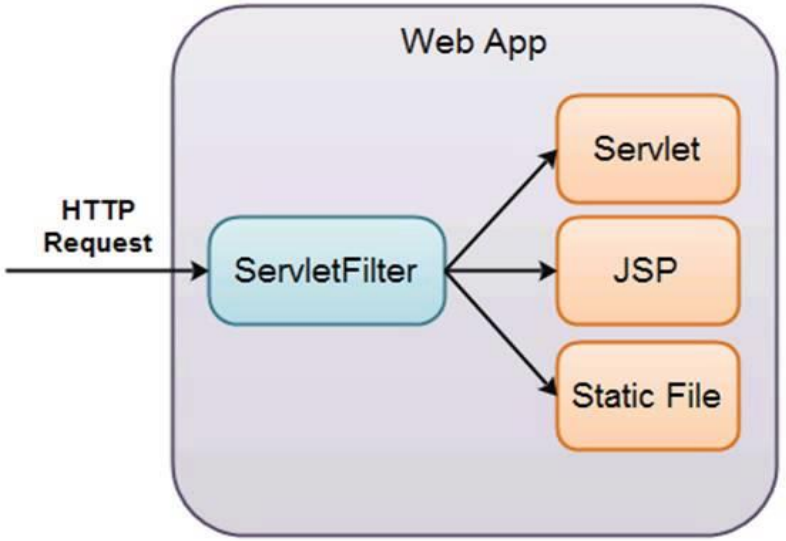
- How should I implement processing that is common to all servlets, such as compressing or transforming the response, logging servlet access, authentication?
  - Servlet filters
  - http://www.oracle.com/technetwork/java/filters-137243.html
  - http://stackoverflow.com/questions/13274279/authentication-filter-and-servlet-for-login

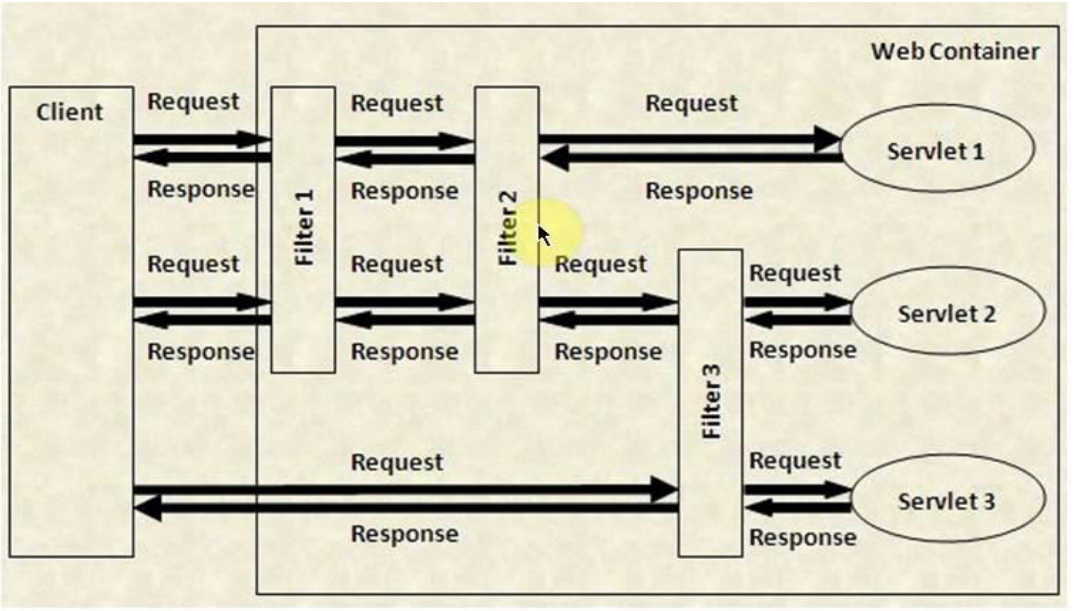
# Filters

- Servlet Filters are Java classes that can be used in Servlet Programming for the following purposes:
  - To intercept requests from a client before access a resource at back end.
  - To manipulate responses from server before they are sent back to the client.

# Filters Architecture

Servlet Filter



# How filters work

- Filters are deployed in the deployment descriptor file web.xml and then map to either servlet names or URL patterns in your application's deployment descriptor. Or annotation

  @WebFilter(filterName = "TimeOfDayFilter", urlPatterns = {"/*"}, initParams = { @WebInitParam(name = "mood", value = "awake")})

- When the web container starts up your web application, it creates an instance of each filter that you have declared in the DD. The filters execute in the order that they are declared in the DD.
- While sending back the response they would be executed in the REVERSE order

# Servlet Filter Methods

- A filter is simply a Java class that implements the javax.servlet.Filter interface.

- The javax.servlet.Filter interface defines three methods:

  - public void init(FilterConfig fConfig) throws ServletException{}

  - public void destroy(){}

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

# Filter configuration in web.xml

```
<filter>
    <filter-name>RequestLoggingFilter</filter-name>
    <filter-class>com.wap.filter.RequestLoggingFilter</filter-class>
</filter>
<filter>
    <filter-name>AuthenticationFilter</filter-name>
    <filter-class>com.wap.filter.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>
```

# Filter

```
@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)
            throws IOException, ServletException {

        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("jsp") || uri.endsWith("LoginServlet"))) {
            this.context.log("Unauthorized access request");
            res.sendRedirect("login.jsp");
        } else {
            // pass the request along the filter chain
            chain.doFilter(request, response);
        }

    }
```

# Filter vs Listener

- Servlet Filter is used for monitoring request and response from client to the servlet, or to modify the request and response, or to audit and log.
- Servlet Listener is used for listening to events in a web containers, such as when you deploy or destroy a web app, create a session, or place an attribute in an session or if you passivate and activate in another container, to subscribe to these events you can configure a listener in web.xml, for example HttpSessionListener or ServletContextListener

# Types of Listener

- ServletContextListener
- ServletContextAttributeListener
- HttpSessionListener
- HttpSessionAttributeListener
- ServletRequestListener
- ServletRequestAttributeListener
- HttpSessionActivationListener
- HttpSessionBindingListener

- http://www.wideskills.com/servlets/servlet-listeners

# ServletContextListener

- Sometimes you are in need of invoking some code during startup and shutdown of your web application.

```
@WebListener
public class MyServletContextListener implements ServletContextListener {


    public void contextDestroyed(ServletContextEvent arg0) {
        System.out.println("Context Destroyed");
    }

    public void contextInitialized(ServletContextEvent servletContextEvent) {
        System.out.println("Context Initialized");
        // get servlet context
        ServletContext context = servletContextEvent.getServletContext();
        // set attribute in context
        String attributeValue = "Context Param Value";
        String attributeName = "ContextParam";
        context.setAttribute(attributeName, attributeValue);

    }

}
```

# HttpSessionListener

- HttpSessionListener listens to HttpSessionEvent event which gives a notification when session is created or destroyed.

```
@WebListener
public class MySessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent httpSessionEvent) {
        HttpSession session = httpSessionEvent.getSession();
        System.out.println("Session Created at  " +
                session.getCreationTime() + " with id  " + session.getId());

    }

    public void sessionDestroyed(HttpSessionEvent httpSessionEvent) {
        HttpSession session = httpSessionEvent.getSession();
        System.out.println("Session destroyed  with id ::  " + session.getId());
    }

}
```