Filters

A **filter** is an object that is invoked at the preprocessing and postprocessing of a request.
It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

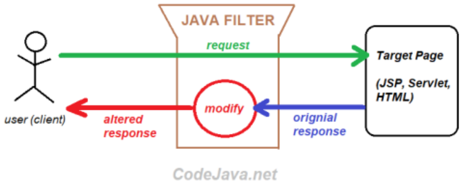
The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change and re-compile the servlet.
So maintenance cost will be less.

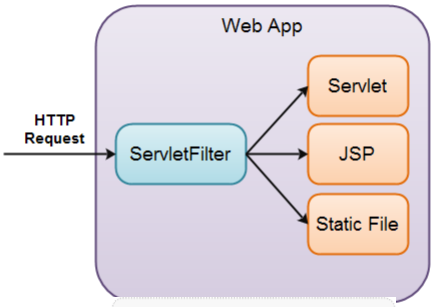
Usage of Filter

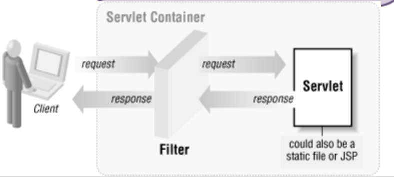
- recording all incoming requests
- logs the IP addresses of the computers from which the requests originate
- conversion
- data compression
- encryption and decryption
- input validation etc.

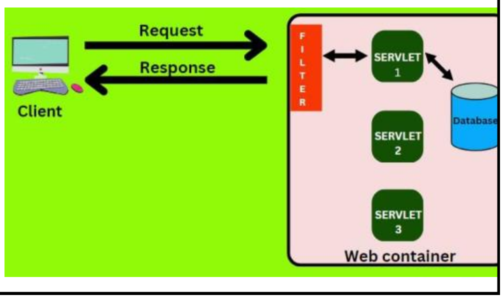
Advantage of Filter

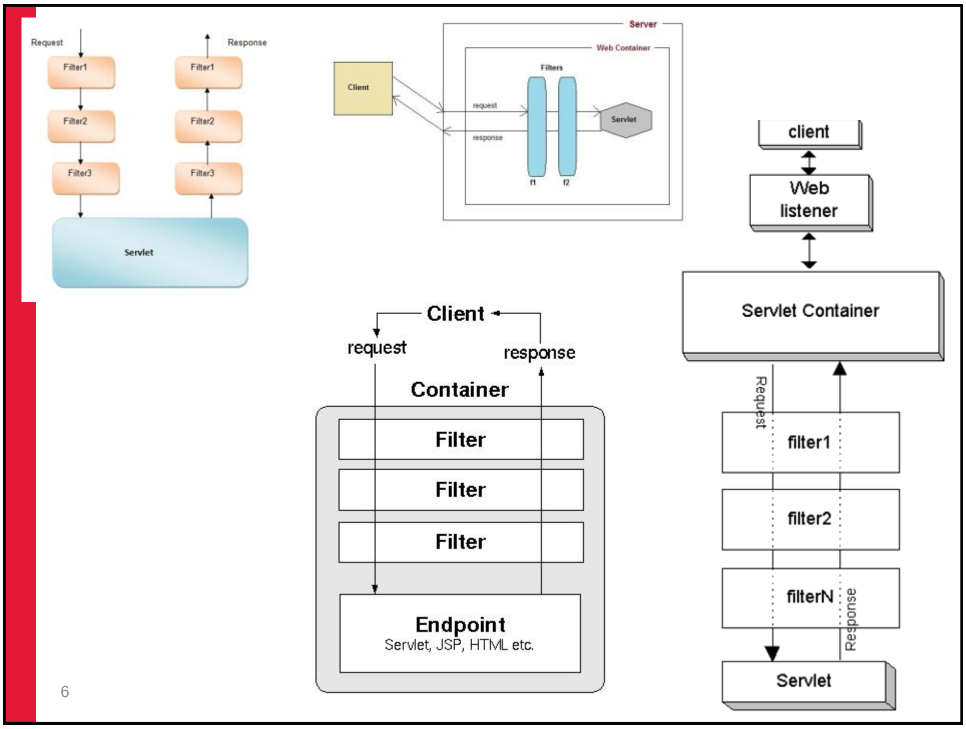
- Filter is pluggable.
- One filter don't have dependency onto another resource.
- Less Maintenance



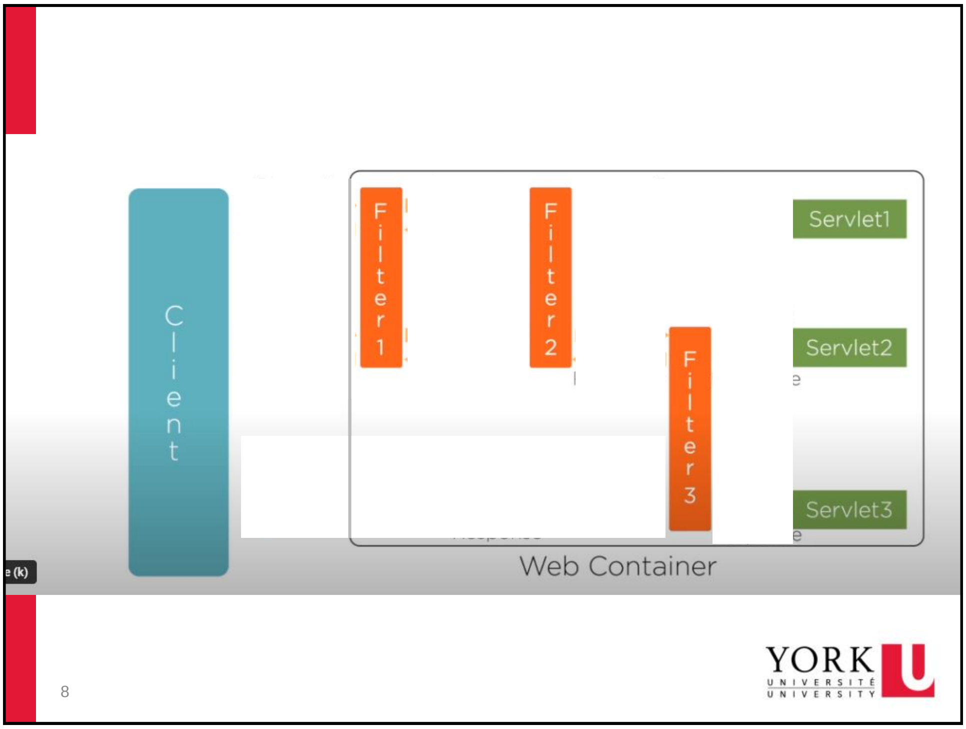




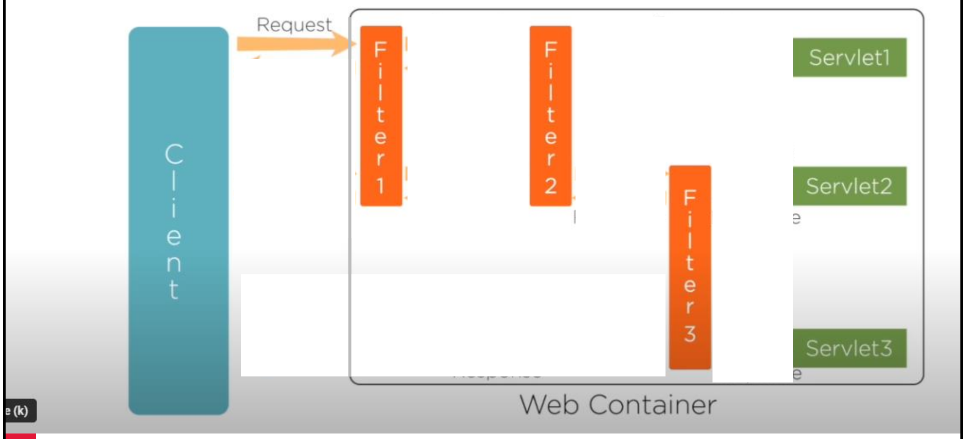



Request
Filter1
Filter2
Filter3
Servlet
Response
Filter1
Filter2
Filter3
Server
Web Container
Client
Filters
request
response
Servlet
f1
f2
client
Web listener
Servlet Container
Request
filter1
filter2
filterN
Response
Servlet
Client
request
response
Container
Filter
Filter
Filter
Endpoint
Servlet, JSP, HTML etc.
6

6

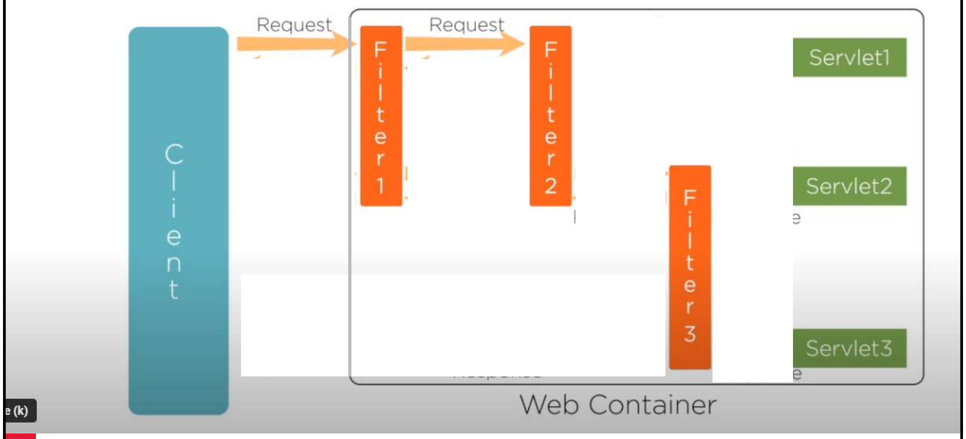

Client
Filter1
Filter2
Filter3
Servlet1
Servlet2
Servlet3
Web Container
e (k)
8

8


Request
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

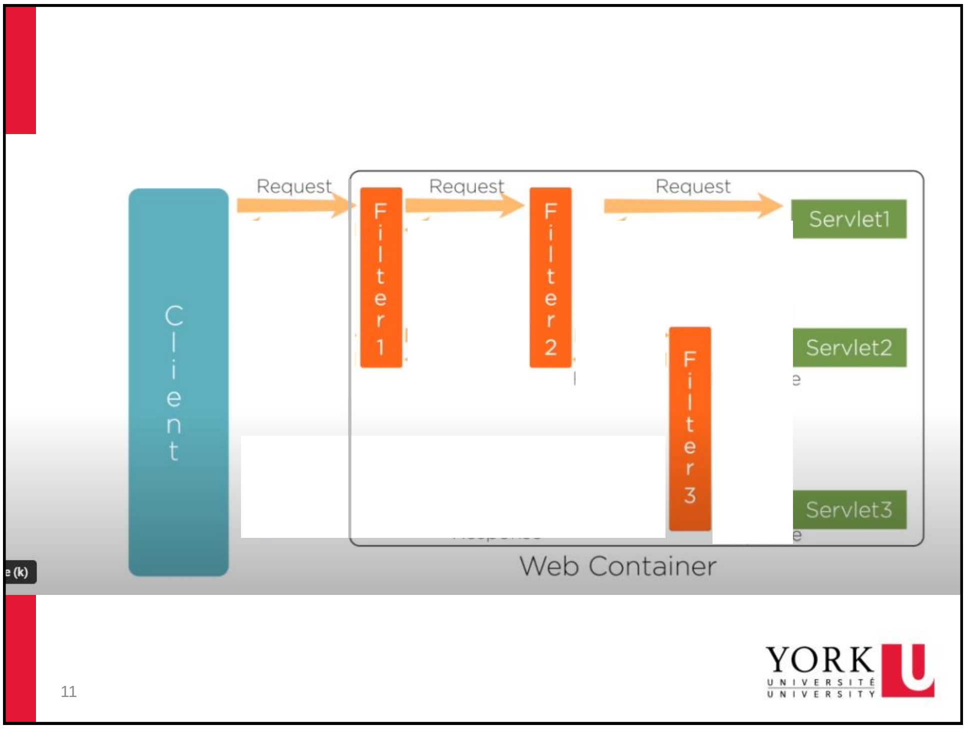
9

9


Request
Request
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

10

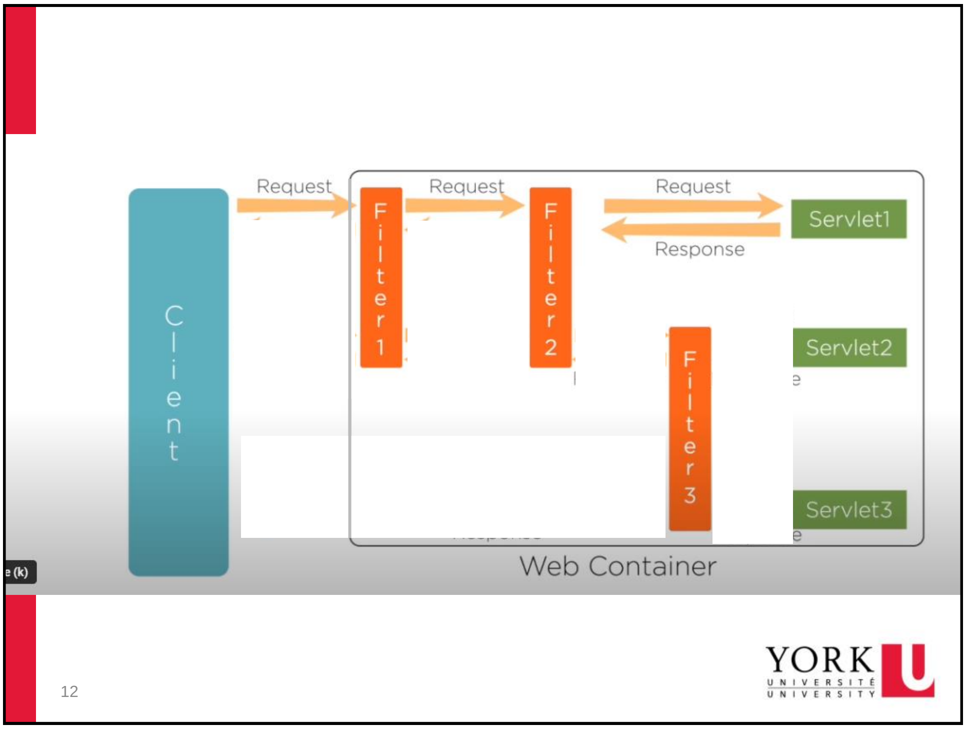
10



Request
Request
Request
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

11

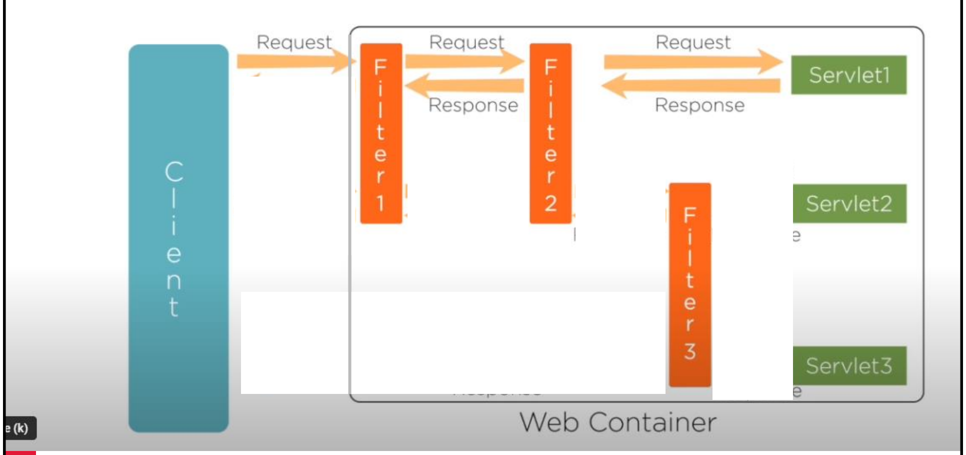
11



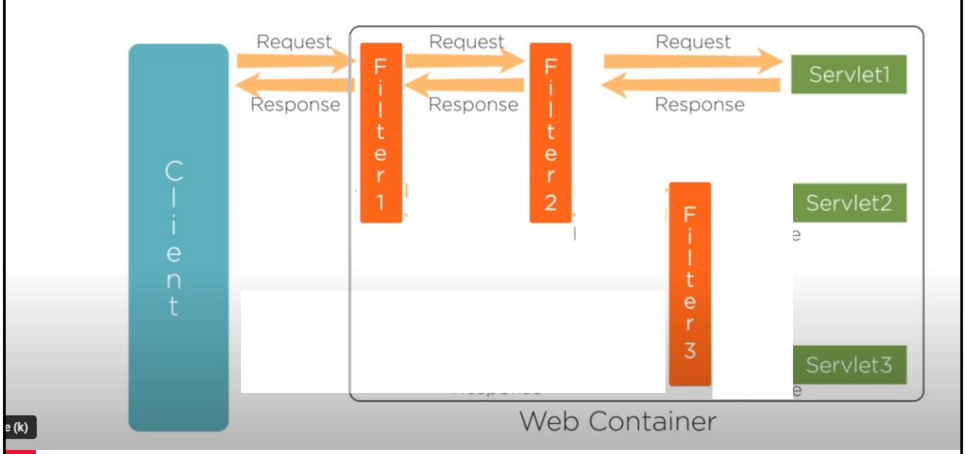
Request
Request
Request
Response
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

12

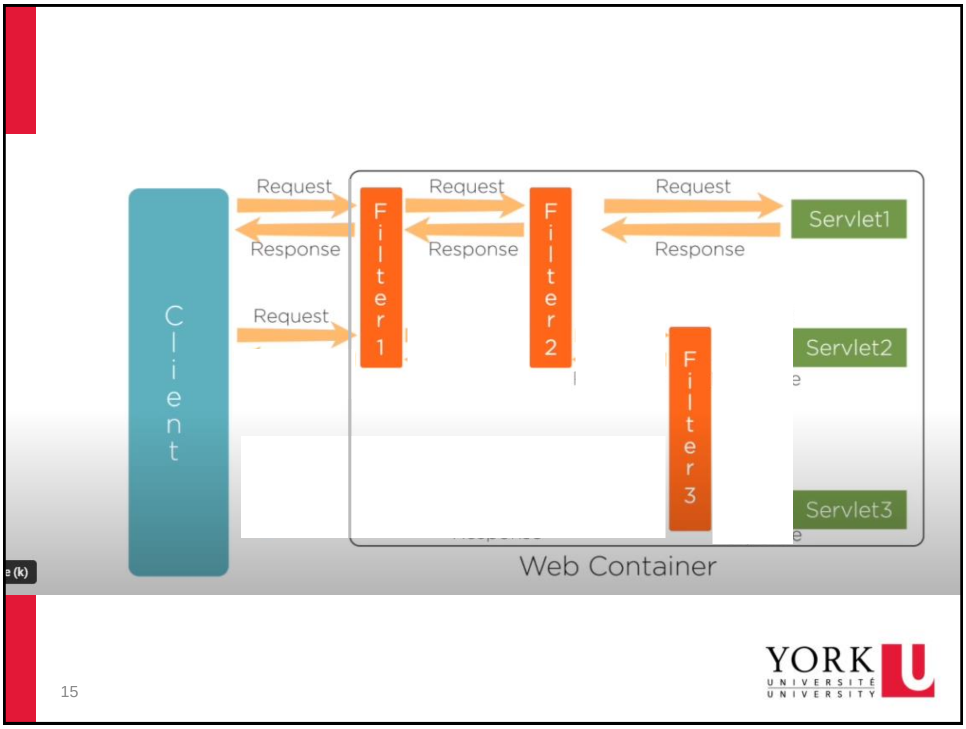
12


Request
Request
Response
Request
Response
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

13


Request
Response
Request
Response
Request
Response
Client
Filter 1
Filter 2
Filter 3
Servlet1
Servlet2
Servlet3
Web Container

14



Client

Request

Response

Request

Filter 1

Request

Response

Filter 2

Request

Response

Servlet1

Servlet2

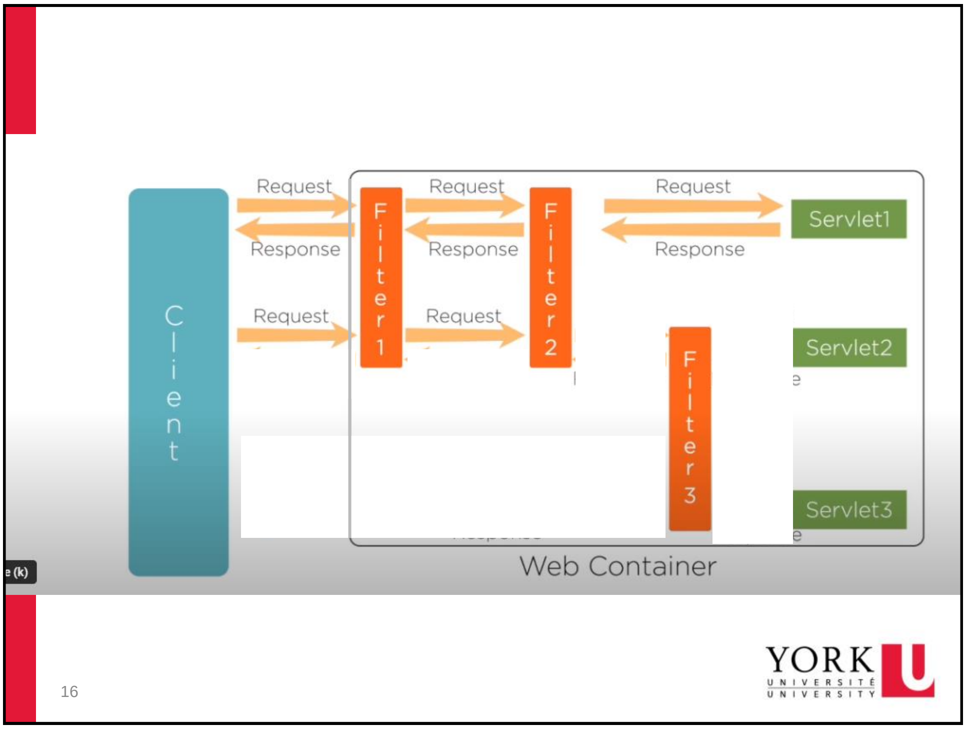
Filter 3

Servlet3

Web Container

e (k)

15



Client

Request

Response

Request

Filter 1

Request

Response

Request

Filter 2

Request

Response

Servlet1

Servlet2

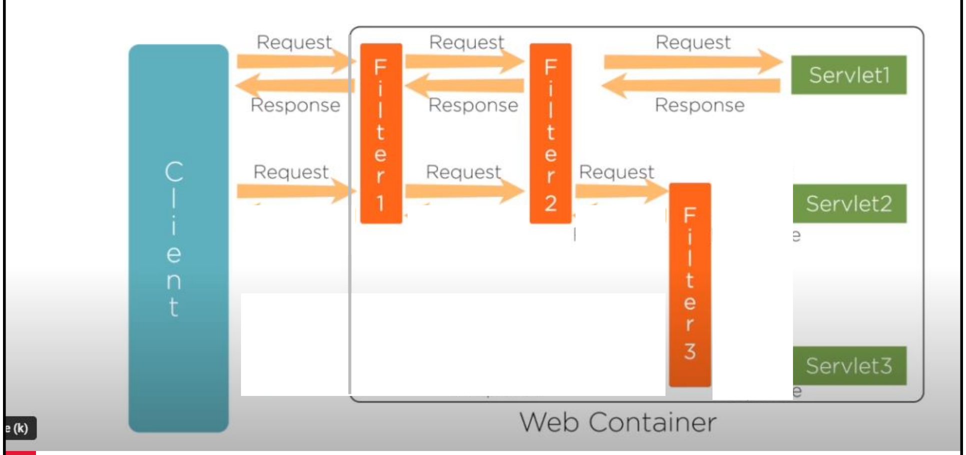
Filter 3

Servlet3

Web Container

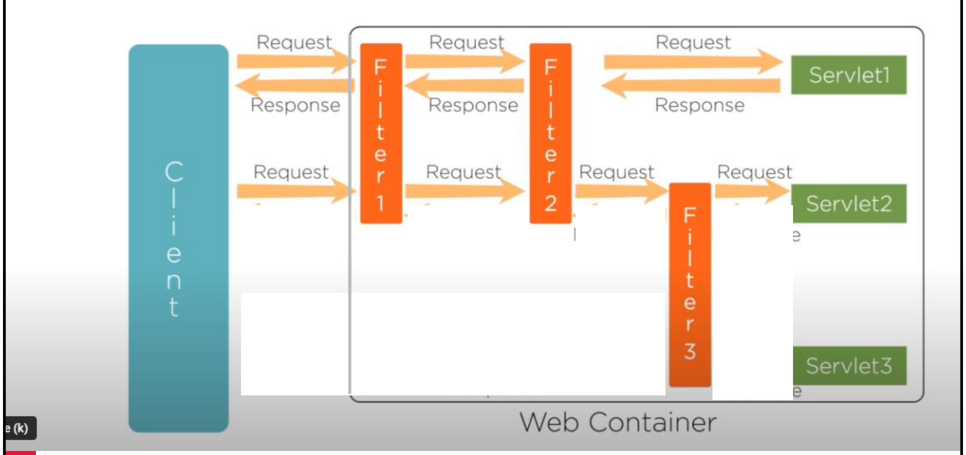
e (k)

16


Request
Response
Request
Response
Request
Response
Client
Filter 1
Filter 2
Servlet1
Request
Request
Request
Filter 3
Servlet2
Servlet3
Web Container
e (k)

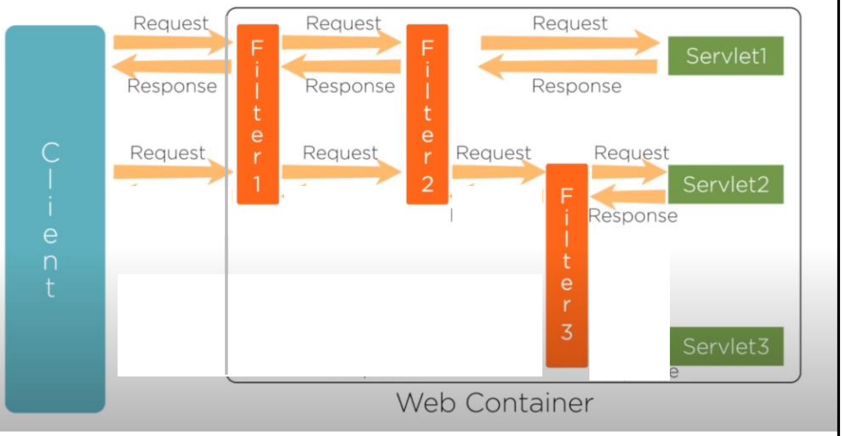
17

17


Request
Response
Request
Response
Request
Response
Client
Filter 1
Filter 2
Servlet1
Request
Request
Request
Request
Filter 3
Servlet2
Servlet3
Web Container
e (k)

18

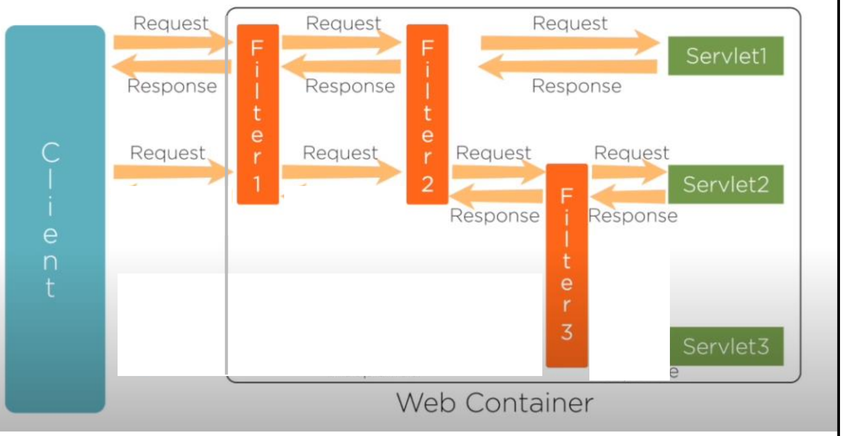
18



Request
Response
Request
Response
Request
Response
Servlet1
C l i e n t
F i l t e r 1
F i l t e r 2
Request
Request
Request
Request
Servlet2
Response
F i l t e r 3
Servlet3
e
Web Container

19

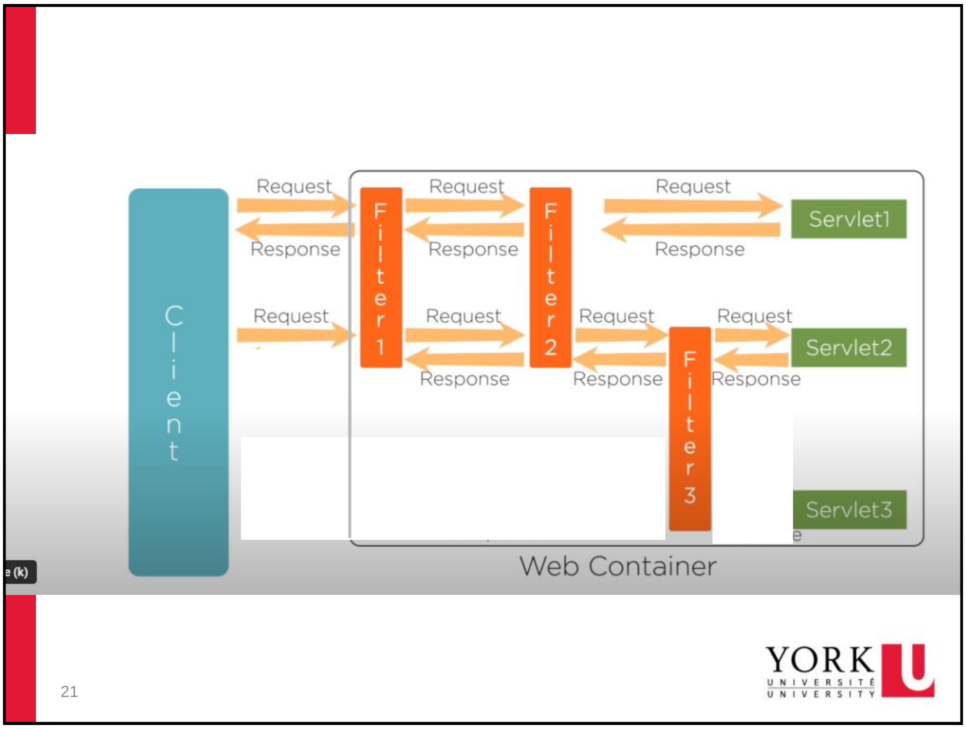
19



Request
Response
Request
Response
Request
Response
Servlet1
C l i e n t
F i l t e r 1
F i l t e r 2
Request
Request
Request
Request
Servlet2
Response
Response
F i l t e r 3
Servlet3
e
Web Container

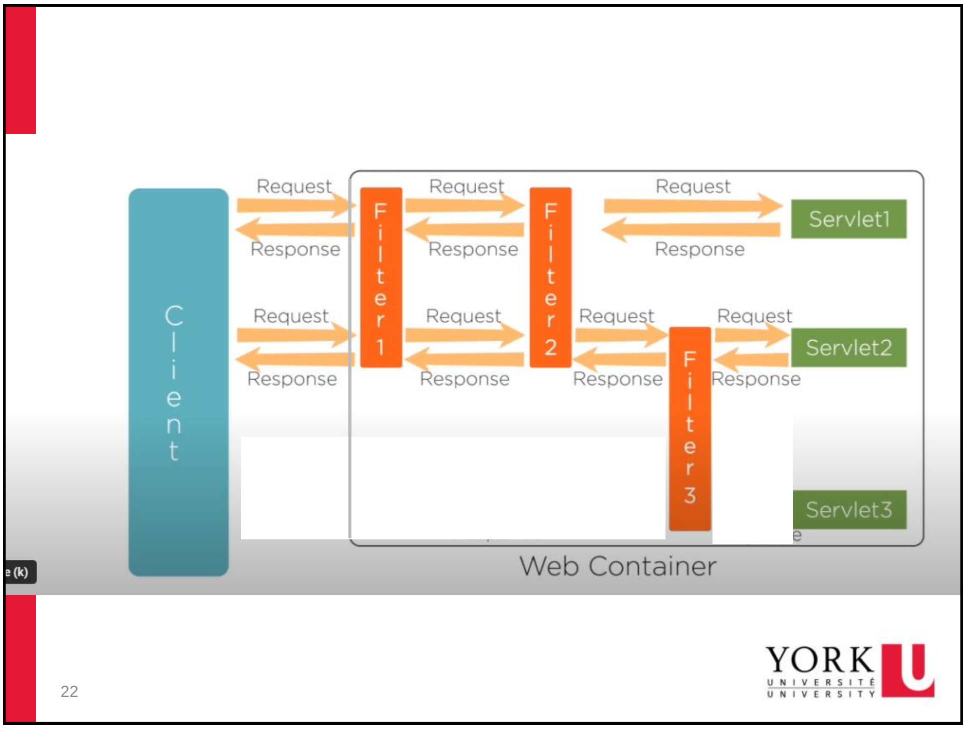
20

20


Client
Request
Response
Request
Filter 1
Request
Response
Request
Response
Filter 2
Request
Response
Request
Response
Servlet1
Request
Response
Filter 3
Request
Response
Servlet2
Servlet3
Web Container
e (k)
YORK UNIVERSITÉ UNIVERSITY

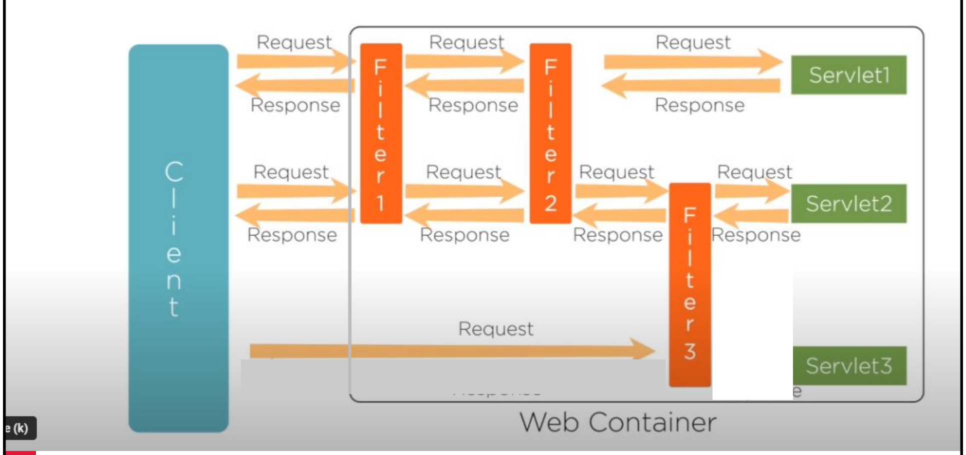
21

21


Client
Request
Response
Request
Response
Filter 1
Request
Response
Request
Response
Filter 2
Request
Response
Request
Response
Servlet1
Request
Response
Filter 3
Request
Response
Servlet2
Servlet3
Web Container
e (k)
YORK UNIVERSITÉ UNIVERSITY

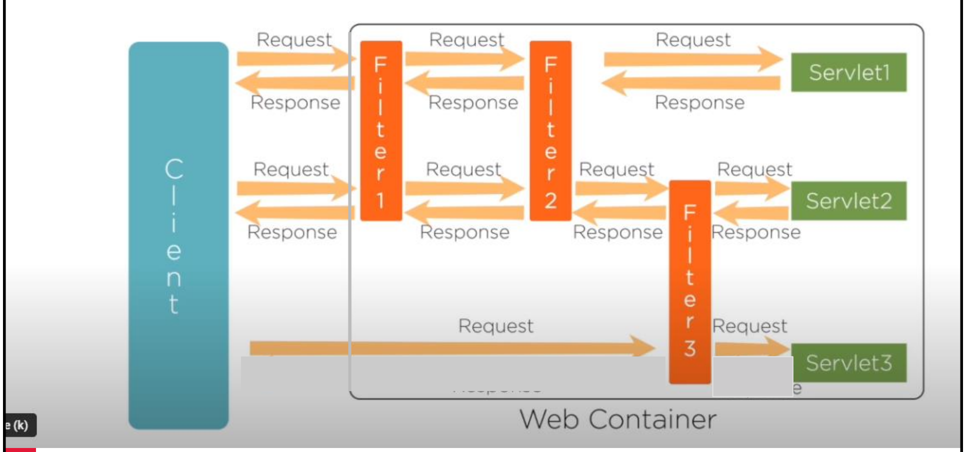
22

22


Client
Request
Response
Filter 1
Request
Response
Filter 2
Request
Response
Servlet1
Request
Response
Request
Response
Request
Response
Filter 3
Request
Response
Servlet2
Request
Servlet3
Web Container

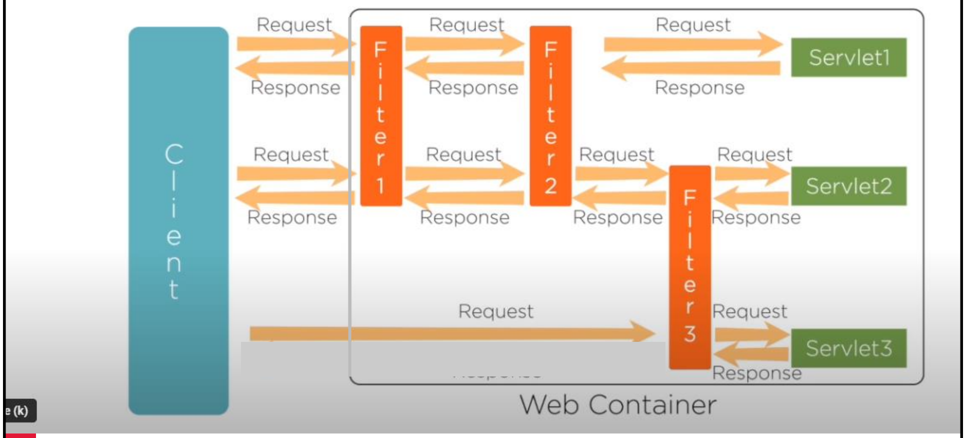
23

23

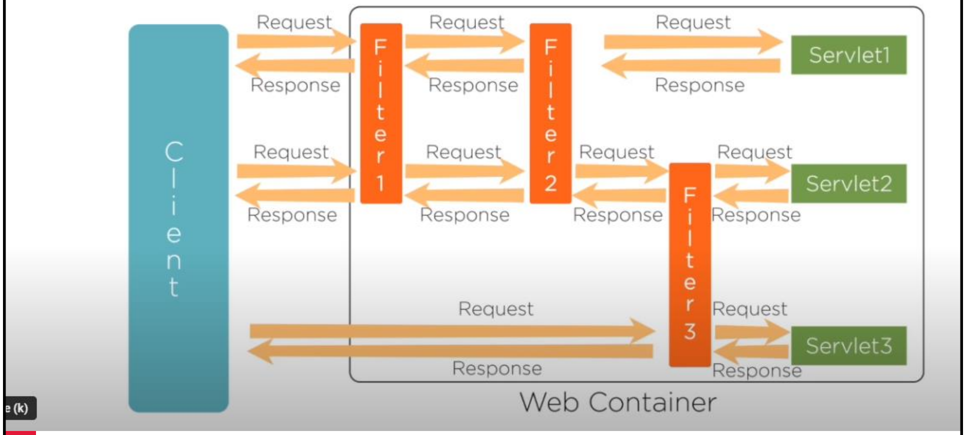

Client
Request
Response
Filter 1
Request
Response
Filter 2
Request
Response
Servlet1
Request
Response
Request
Response
Request
Response
Filter 3
Request
Response
Servlet2
Request
Request
Servlet3
Web Container

24

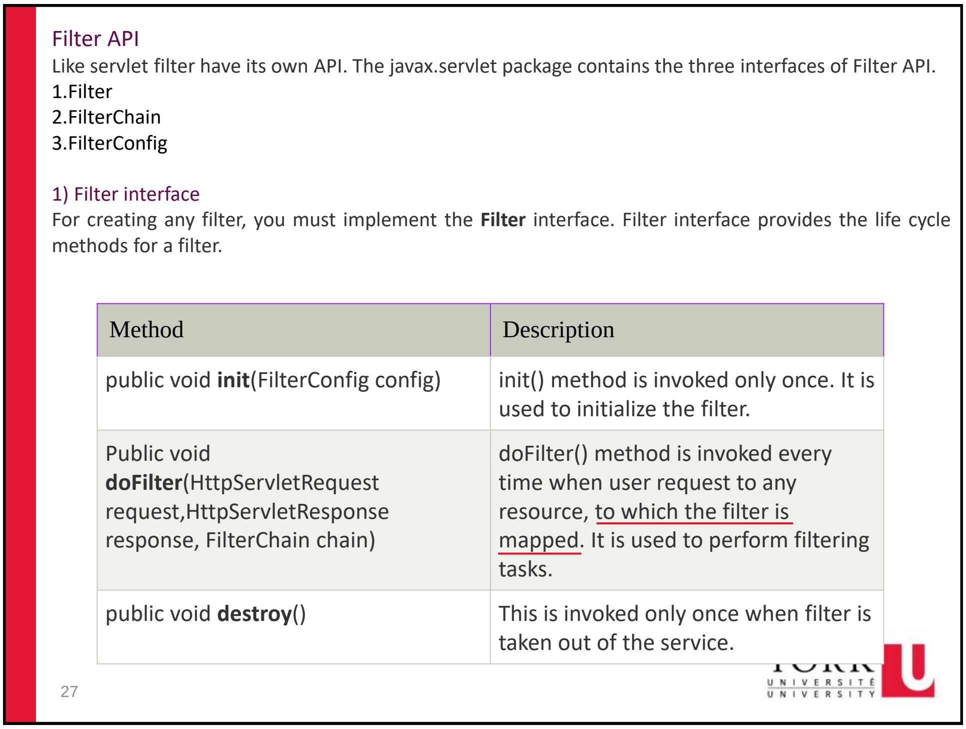
24


Client
Request
Response
Filter 1
Request
Response
Filter 2
Request
Response
Servlet1
Request
Response
Request
Response
Request
Response
Filter 3
Request
Response
Servlet2
Request
Request
Response
Servlet3
Response
Web Container
e (k)

25


Client
Request
Response
Filter 1
Request
Response
Filter 2
Request
Response
Servlet1
Request
Response
Request
Response
Request
Response
Filter 3
Request
Response
Servlet2
Request
Response
Request
Response
Servlet3
Web Container
e (k)

26



Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1.Filter
2.FilterChain
3.FilterConfig

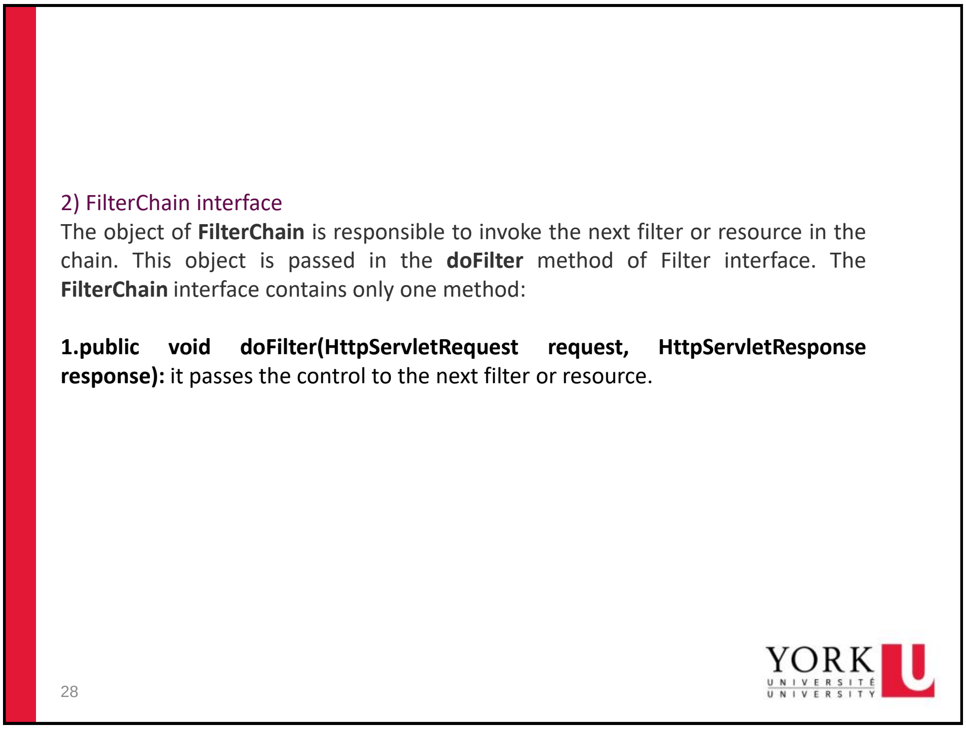
1) Filter interface

For creating any filter, you must implement the **Filter** interface. Filter interface provides the life cycle methods for a filter.

Method	Description
public void **init**(FilterConfig config)	init() method is invoked only once. It is used to initialize the filter.
Public void **doFilter**(HttpServletRequest request,HttpServletResponse response, FilterChain chain)	doFilter() method is invoked every time when user request to any resource, to which the filter is mapped. It is used to perform filtering tasks.
public void **destroy**()	This is invoked only once when filter is taken out of the service.

27

27



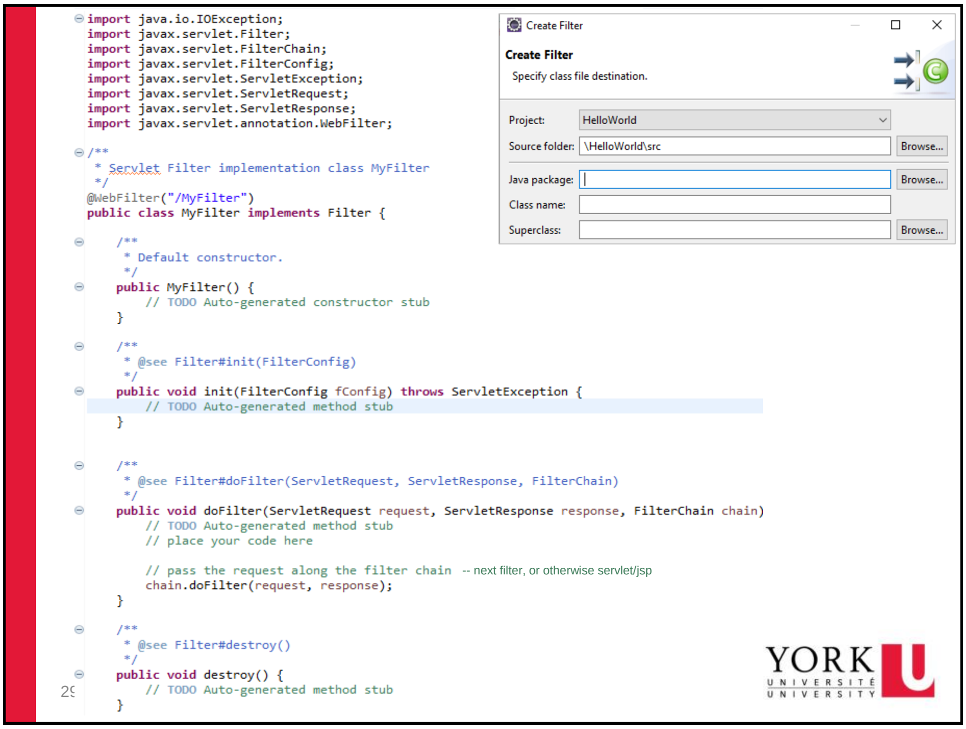
2) FilterChain interface

The object of **FilterChain** is responsible to invoke the next filter or resource in the chain. This object is passed in the **doFilter** method of Filter interface. The **FilterChain** interface contains only one method:

1.public void doFilter(HttpServletRequest request, HttpServletResponse response): it passes the control to the next filter or resource.

28

28



Create Filter

Specify class file destination.

Project: HelloWorld

Source folder: \HelloWorld\src

Java package:

Class name:

Superclass:

```
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;

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

    /**
     * Default constructor.
     */
    public MyFilter() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }


    /**
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        // TODO Auto-generated method stub
        // place your code here

        // pass the request along the filter chain  -- next filter, or otherwise servlet/jsp
        chain.doFilter(request, response);
    }

    /**
     * @see Filter#destroy()
     */
    public void destroy() {
        // TODO Auto-generated method stub
    }
```

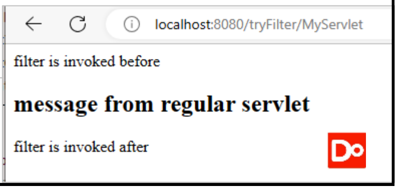
29

29

old way

```
<web-app>

<filter>
<filter-name>myF...</filter-name>
<filter-class>.../MyFilter</filter-class
</filter>

<filter-mapping>
<filter-name>myF</filter-name>
<url-pattern>MyServlet</url-pattern
</filter-mapping>
```

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

import javax.servlet.*;
@WebFilter ("/MyServler")
public class MyFilter implements Filter{

    public void init(FilterConfig arg0) throws ServletException {}

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

        PrintWriter out = response.getWriter();
        out.print("filter is invoked before");

        chain.doFilter(request, response);  sends request to next resource (in chain)

        out.print("filter is invoked after");
        }

     public void destroy() {}
}
```



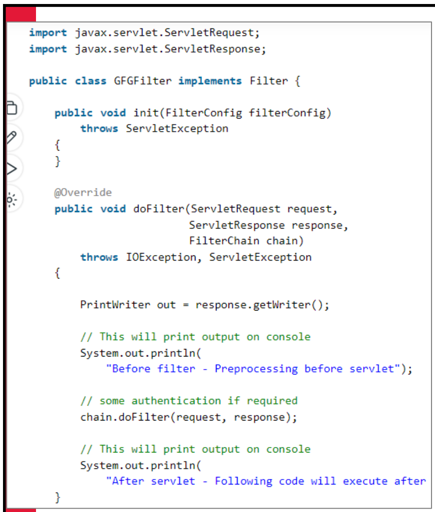
filter is invoked before

message from regular servlet

filter is invoked after

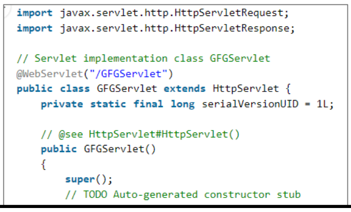
30

30

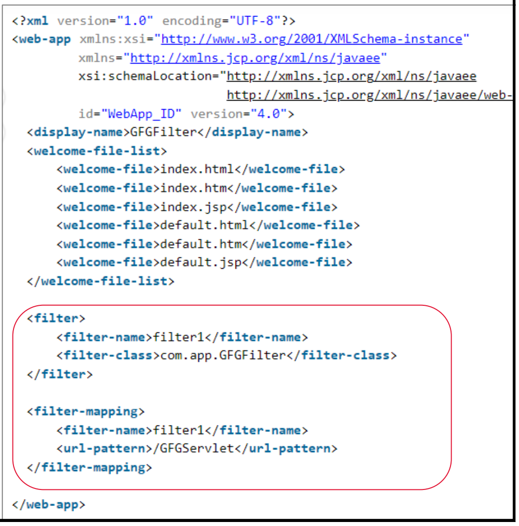


```
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class GFGFilter implements Filter {

    public void init(FilterConfig filterConfig)
        throws ServletException
    {
    }

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

        PrintWriter out = response.getWriter();

        // This will print output on console
        System.out.println(
            "Before filter - Preprocessing before servlet");

        // some authentication if required
        chain.doFilter(request, response);

        // This will print output on console
        System.out.println(
            "After servlet - Following code will execute after
    }
```

old way   FYI



```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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

    // @see HttpServlet#HttpServlet()
    public GFGServlet()
    {
        super();
        // TODO Auto-generated constructor stub
```



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-
         id="WebApp_ID" version="4.0">
  <display-name>GFGFilter</display-name>
  <welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.htm</welcome-file>
      <welcome-file>index.jsp</welcome-file>
      <welcome-file>default.html</welcome-file>
      <welcome-file>default.htm</welcome-file>
      <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <filter>
      <filter-name>filter1</filter-name>
      <filter-class>com.app.GFGFilter</filter-class>
  </filter>

  <filter-mapping>
      <filter-name>filter1</filter-name>
      <url-pattern>/GFGServlet</url-pattern>
  </filter-mapping>

</web-app>
```

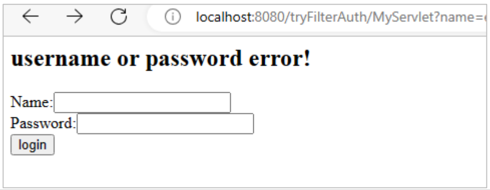
Authentication Filter

```
<form action="MyServlet">
    Name:<input type="text" name="name"/><br/>
    Password:<input type="password" name="password"/><br/>

    <input type="submit" value="login">

</form>
```

```
@WebServlet ("MySevlet")
public class AdminServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      out.print("welcome ADMIN");
      out.close();
  }
}
```

Authentication Filter

```
@WebServlet ("MyServlet")
public class MyFilter implements Filter{

    public void init(FilterConfig arg0) throws ServletException {}

    public void doFilter(ServletRequest request, ServletResponse response,
                              FilterChain chain) throws IOException, ServletException
    {
          String name=request.getParameter("name");
          String password=request.getParameter("password");         Same logic as servlet

          if( name.equals("admin") && password.equals("yu123")){
             chain.doFilter(request, response);   //sends request to next resource
          }
          else{
            response.setContentType("text/html");
            PrintWriter out=response.getWriter();
            out.print("username or password error!");
            RequestDispatcher rd=request.getRequestDispatcher("index.html");
            rd.include(request, response);
          }
    }
    public void destroy() {}
```


localhost:8080/tryFilterAuth/MyServlet?name=

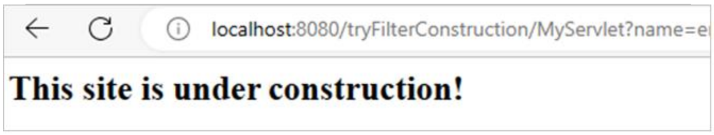
username or password error!

Name:

Password:

Example of sending response by filter only

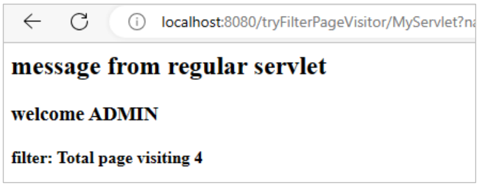
```
@WebServlet ("MyServlet")
public class MyFilter implements Filter{
    public void init(FilterConfig arg0) throws ServletException {}

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.print("<br/>This site is under construction!");


    }

    public void destroy() {}
}
```


localhost:8080/tryFilterConstruction/MyServlet?name=e

This site is under construction!

Example of counting number of visitors for a single page

```
@WebServlet ("MyServlet")
public class MyFilter implements Filter{
    static int count=0;
    public void init(FilterConfig fConfig) throws ServletException {}

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

        PrintWriter out=response.getWriter();
        chain.doFilter(request,response);

        count++;
        out.print("<br/>Total visitors "+ count );


    }
    public void destroy() {}
  }
```



localhost:8080/tryFilterPageVisitor/MyServlet?na

message from regular servlet

welcome ADMIN

filter: Total page visiting 4