

# Java Servlets 3.0

## Lesson 10: Servlet Filter

## Lesson Objectives

- In this lesson, we will learn:
  - What is a filter
  - Filtering Components
  - Programming a Filter
  - Filter Configuration
  - Filter Life Cycle



Copyright © Capgemini 2015. All Rights Reserved 2

### Lesson Objectives:

This lesson introduces Servlets Filter. The lesson contents are:

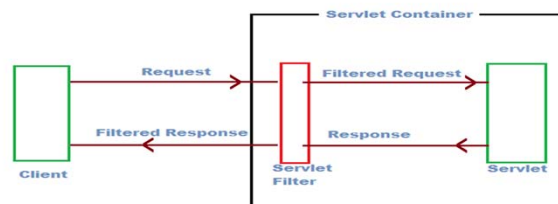
#### Lesson 10: Servlet Filter

- 10.1: Introduction to Servlet Filter
- 10.2: Filtering Components
- 10.3: Programming a Filter
- 10.4: Filter Configuration
- 10.5: Filter Life Cycle

## 10.1: Introduction to Servlet Filter

## What is a Filter?

- Filter is a component which dynamically intercepts requests and responses to transform or use the information contained in the requests or responses
- Filters typically do not themselves create responses but provide universal functions that can be “attached” to any type of servlet or JSP page
- A filter is a program that runs on the server before the servlet with which it is associated



### Introduction to Servlet Filter:

#### What is a Filter?

A filter is a reusable piece of code that transforms either the content of an HTTP request or response and can also modify header information. Filters typically do not themselves create responses, but instead provide universal functions that can be "attached" to any type of servlet or JSP page

A filter is a program that runs on the server before the servlet or JSP page with which it is associated. A filter can be attached to one or more servlets or JSP pages and can examine the request information going into these resources. After doing so, it can choose among the following options :

- Invoke the resource (i.e., the servlet or JSP page) in the normal manner

- Invoke the resource with modified request information

- Invoke the resource but modify the response before sending it to the client

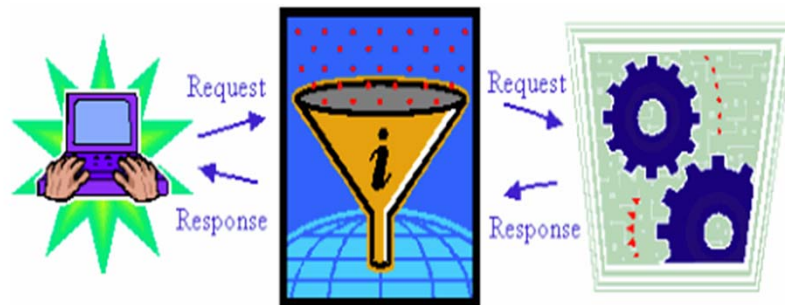
- Prevent the resource from being invoked and instead redirect to a different resource, return a particular status code, or generate replacement output

Filters are important because they provide the ability to encapsulate recurring tasks in reusable units

## 10.1: Introduction to Servlet Filter

## Benefits of Filters

- Filters encapsulate common behaviour in a modular and reusable manner
- Filters separate high-level access decisions from presentation code
- Filters apply wholesale changes to many different resources



Introduction to Servlet Filter:

Benefits of Filters:

Filters allow to encapsulate common behaviour in a modular and reusable manner.

Modular code is more manageable and documentable, is easier to debug, and if done well, can be reused in another setting.

If we have 30 different servlets that need to compress their content to decrease download time? No problem, make a single compression filter and apply it to all 30 resources.

Filters separate high-level access decisions from presentation code


Filters allow to apply wholesale changes to many different resources

For example: If there a bunch of existing resources that should remain unchanged except that the company name should be changed? No problem: make a string replacement filter and apply it wherever appropriate.

10.2: Filtering Components

## Types of Filter

- Based on the various functionalities filters carry out there are following Filter Components :
  - Authentication
  - Logging and auditing
  - Image conversion
  - Data Compression
  - Localization
  - XSL/T transformations of XML

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 5

Filtering Components:

Types of Filter:

Filters can perform many different types of functions. Based on their functionalities following are the filters :

Authentication-Blocking requests based on user identity

Logging and auditing-Tracking users of a web application

Image conversion-Scaling maps, and so on

Data compression-Making downloads smaller

Localization-Targeting the request and response to a particular locale

XSL/T transformations of XML content-Targeting web application responses to more than one type of client.

These are just a few of the applications of filters. There are many more, such as encryption, tokenizing, triggering resource access events, mime-type chaining, and caching.

10.3: Programming a Filter

## Steps for Programming a Filter

- Filter API is defined by the Filter, FilterChain, and FilterConfig interfaces in the javax.servlet package.
- Five basic steps to create a filter are given below:
  - Create a class that implements the Filter interface
  - Put the filtering behaviour in the doFilter method
  - Call the doFilter method of the FilterChain object
  - Register the filter with the appropriate servlets(/jsp)
  - Disable the invoker servlet

### Programming a Filter:

Creating a filter involves five basic steps:

Create a class that implements the Filter interface.

The class will need three methods: doFilter, init, and destroy. The doFilter method contains the main filtering code (see Step 2), the init method performs setup operations, and the destroy method does cleanup.

Put the filtering behaviour in the doFilter method.

The first argument to the doFilter method is a ServletRequest object. This object gives the filter full access to the incoming information, including form data, cookies, and HTTP request headers. The second argument is a ServletResponse; it is mostly ignored in simple filters. The final argument is a FilterChain; it is used to invoke the servlet or JSP page or the next filter as described in the next step.

Call the doFilter method of the FilterChain object.

The doFilter method of the Filter interface takes a FilterChain object as one of its arguments. When doFilter method of that object is invoked, the next associated filter is invoked. If no other filter is associated with the servlet or JSP page, then the servlet or page itself is invoked.

Register the filter with the appropriate servlets and JSP pages.

Use the filter and filter-mapping annotations above the class.

Disable the invoker servlet. Prevent users from bypassing filter settings by using default servlet URLs.

The most important method in the Filter interface is doFilter, which is passed request, response, and filter chain objects. This method can perform the following actions:

Examine the request headers.

Customize the request object if the filter wishes to modify request headers or data.

Customize the response object if the filter wishes to modify response headers or data.

Invoke the next entity in the filter chain. If the current filter is the last filter in the chain that ends with the target web component or static resource, the next entity is the resource at the end of the chain; otherwise, it is the next filter that was configured in the WAR. The filter invokes the next entity by calling the doFilter method on the chain object, passing in the request and response it was called with or the wrapped versions it may have created.

Alternatively, the filter can choose to block the request by not making the call to invoke the next entity. In the latter case, the filter is responsible for filling out the response.

Examine response headers after invoking the next filter in the chain.

Throw an exception to indicate an error in processing.

In addition to doFilter, the init and destroy methods must be implemented. The init method is called by the container when the filter is instantiated. If need to pass initialization parameters to the filter, can be retrieved the FilterConfig object passed to init method

10.3: Programming a Filter

## @WebFilter – Configuring a Filter

- Servlets 3.0 uses an annotation to create a filter that is `@WebFilter`
- Annotation is specified on a class and contains metadata about the filter being declared
- The annotated filter must specify at least one URL pattern. This is done by using the `urlPatterns` or `value` attribute on the annotation
- All other attributes are optional, with default settings. Use the `value` attribute when the only attribute on the annotation is the URL pattern; use the `urlPatterns` attribute when other attributes are also used
- To add configuration data to the filter, specify the `initParams` attribute of the `@WebFilter` annotation. The `initParams` attribute contains a `@WebInitParam` annotation. The following code snippet defines a filter, specifying an initialization parameter



Copyright © Capgemini 2015. All Rights Reserved 8

Classes annotated with the `@WebFilter` annotation must implement the `javax.servlet.Filter` interface.

`@WebFilter` is used to create a filter in Servlet 3.0.

Configuring Filter with annotations:

Next is registering a filter with appropriate servlet. To map a filter to a servlet make note of following:

Declare the filter using the annotation `@WebFilter` above class name. This annotation will inform the container to create a filter. This filter would then be invoked before the actual web resource. Also initialization parameters could be used for the filter, by making use of annotation `@WebInitParam`.

Map the filter to a servlet by defining an attribute `urlPatterns`, this URL pattern could be of any web resource.

If all web resources need to be matched to particular filter then, use `/*` as the URL Pattern



## 10.4: Filter Configuration

## Initializing Filter

- Filters could be configured with initParams attribute of the @WebFilter annotation; Need to pass the attribute with name and value pairs
- These parameters could be retrieved in lifecycle method- init() and they are used for one time configuration of filters
- Following is the code snippet, which specifies same

```
@WebFilter(filterName = "TimeFilter",urlPatterns = {"/*"},
initParams = {
    @WebInitParam(name = "mood", value = "awake")})
public class AuthFilter implements Filter {
    ...
}
```



Copyright © Capgemini 2015. All Rights Reserved 9

In above code snippet, we have used attribute initParams to pass the configuration information to filters.


These parameters could be retrieved in init() - method of filter

10.3: Programming a Filter

## Logging Filter Code

```
@WebFilter(filterName = "TimeFilter", urlPatterns = { "/" })
public class LogFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse
        response,
        FilterChain chain) throws IOException, ServletException {
        String ipAddress = request.getRemoteAddr();
        //Log the IP address and current timestamp.
        System.out.println("IP "+ipAddress + ", Time "+ new
            Date().toString());

        chain.doFilter(request, response);
    }
    public void init(FilterConfig config) throws ServletException {
        //Code to initialize resources
        //Eg : String testParam = config.getInitParameter("test-param");
    }
    public void destroy() { //code to release any resource } }
```

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 10

### Programming a Filter:

#### Logging Filter Code:

The above filter creates a Servlet Filter that just prints the clients IP address and date time. (This is just to log users who are accessing the application). We have implemented the `javax.servlet.Filter` interface and implement its methods – `init()`, `doFilter()` and `destroy()`.

The `init()` method is used to initialize any code that is used by Filter. Note that `init()` method will get an object of `FilterConfig` which contains different Filter level information as well as init parameters which is passed from `web.xml`.

The `doFilter()` method is where the actual filtering process happens. User implementation would add code which can modify request / session / response, add any attribute in request etc.

The `destroy()` method is called by the container when it wants to garbage collect the filter.


The above code involves first three steps of programming a filter.


A set of initialization parameters can be associated with a filter using the `init-params` attribute of `@WebFilter` annotation. The names and values of these parameters are available to the Filter at runtime via the `getInitParameter()` and `getInitParameterNames()` methods on the filter's `FilterConfig`. Additionally, the `FilterConfig` affords access to the `ServletContext` of the web application for the loading of resources, for logging functionality or for storage of state in the `ServletContext`'s attribute list

See the next subsequent slides for examples

## Demo

- Create a filter which authenticates the user by passing user credentials via Filter – init params.
- Before this a filter should log the IP address and time of the day when request is given by client



 **Capgemini**  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 11

**Note:**

Refer the LogFilter and AuthFilter.java. AuthFilter authenticates the user by passing user credentials via init-params.

Also we need to log the IP address and time of day when request was raised.

Already a filter is created called LogFilter for this purpose.

Thus we use LogFilter and AuthFilter together before the request comes to web resource – HelloServlet

If the user credentials do not match, the filter will generate a response stating that user is invalid displaying the incorrect user name

Observe that we are mapping more than one filter with Servlet

10.4: Filter Configuration


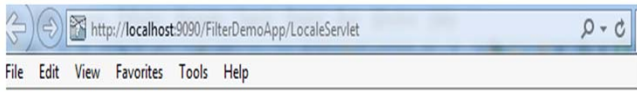
## Process of Registering Filter with Servlet

- One filter can be associated with more than one servlet
- More than one filter can be mapped to a servlet
- A filter can be mapped to all the servlet requests using an “/\*” as URL pattern of Filter
- All of above could be accomplished by using annotations


Refer:

## Demo

- Create a filter which displays locale.

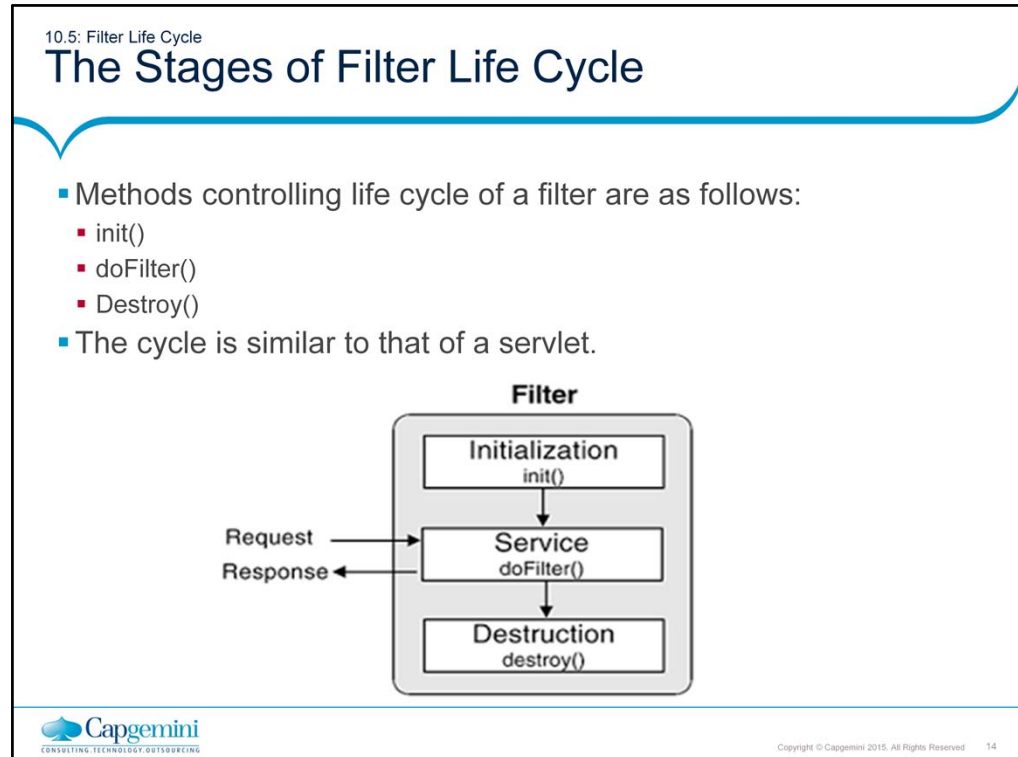


bonjour Reached here after , displaying locale....

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 13

Refer LocaleFilter that displays locale information depending upon the language selected in browser



### Filter Life Cycle:

Three methods of Filter interface makes up the life cycle of a filter:

`public void init(FilterConfig config) throws ServletException`

The `init` method is executed only once, when the filter is first initialized. It is not executed each time the filter is invoked.

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

The `doFilter` method is executed each time a filter is invoked (i.e., once for each request for a servlet or JSP page with which the filter is associated). It is this method that contains the bulk of the filtering logic.

`public void destroy()`

This method is called when a server is permanently finished with a given filter object (e.g., when the server is being shut down). Most filters simply provide an empty body for this method, but it can be used for cleanup tasks like closing files or database connection pools that are used by the filter.

### Detailed Lifecycle of Filter:

After the time when the web application containing filters is deployed, and before an incoming request for a resource in the web application causes a the container to access the resource and serve it back, the container must look through the list of filter mappings to locate the list of filters that must be applied to the resource. How this list is built, is described next.

**Filter Life Cycle:**

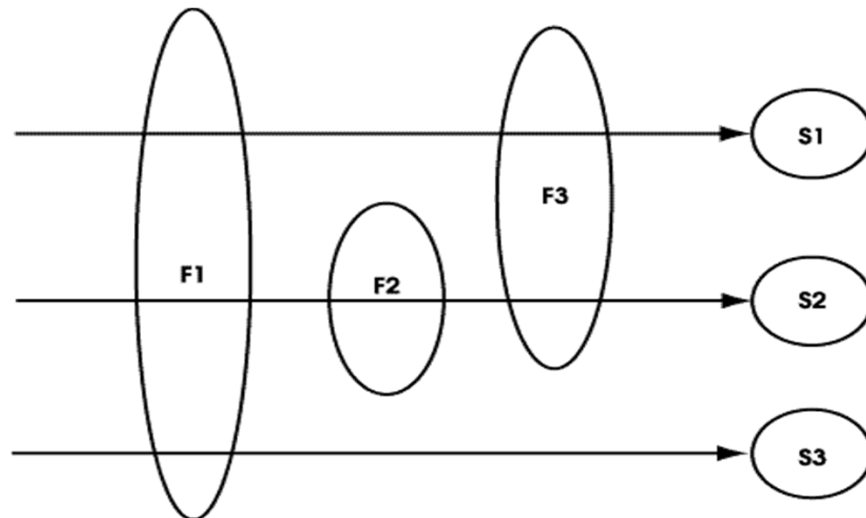
- The container must ensure at some point in this time that, for each filter instance that is to be applied, it has instantiated a filter of the appropriate class, and called `setFilterConfig(FilterConfig config)` on each filter instance in the list.
- The container ensures that the `javax.servlet.FilterConfig` instance that is passed in to this call has been initialized with the filter name as declared in the deployment descriptor for that filter, with the reference to the `ServletContext` for this web application and with the set of initialization parameters declared for the filter in the deployment descriptor.
- When the container receives the incoming request, it takes the first filter instance in the list and calls its `doFilter()` method, passing in the `ServletRequest` and `ServletResponse`, and a reference to the `FilterChain` object it will use.

**FilterConfig Interface:**

- A filter configuration object used by a servlet container to pass information to a filter during initialization.

**FilterChain Interface :**

A `FilterChain` is an object provided by the servlet container giving a view into the invocation chain of a filtered request for a resource. Filters use the `FilterChain` to invoke the next filter in the chain, or if the calling filter is the last filter in the chain, to invoke the resource at the end of the chain. For eg, the below figure shows how Filters F1 and F3 intercept invocations to Servlet S1.



## Summary

- In this lesson, we have learnt:
  - Concept of a Filter
  - Filtering Components
  - Programming a Filter
  - Filter Configuration
  - Filter Life Cycle



Add the notes here.



## Review Question

- Question 1: Which method of Filter interface contains all the logic ?
  - Option 1: init()
  - Option 2: doFilter()
  - Option 3: destroy()
- Question 2: We can use an asterisk in a url Pattern which will represent all servlets as well as JSP referring to same filter
  - True/False
- Question 3: We can not have multiple url Pattern for a filter?
  - True/False



Add the notes here.