📌 Step-by-Step Notes on Filters in Servlets (Java EE/Jakarta EE)

A **Filter** in Servlets is used to **intercept requests and responses** before they reach a servlet or after a response is sent. Filters are commonly used for **logging, authentication, compression, and request/response modifications**.

🔹 1️⃣ What is a Filter?

A **filter** is a Java class that:

- **Intercepts HTTP requests/responses.**
- **Modifies request/response objects if needed.**
- **Passes the request to the next filter or servlet.**

🔹 2️⃣ Steps to Create and Use a Filter in Servlets

📌 Step 1: Create a Filter Class

A filter must implement the `javax.servlet.Filter` interface.

```
import java.io.IOException;
import javax.servlet.*;

public class MyFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("Filter initialized");
    }

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

        // Pre-processing before request reaches the servlet
        System.out.println("Filter executed before servlet");
        // Pass the request to the next filter or servlet
        chain.doFilter(request, response);

        // Post-processing before response is sent
        System.out.println("Filter executed after servlet");
    }

    @Override
    public void destroy() {
        System.out.println("Filter destroyed");
    }
}
```

📌 Step 2: Configure the Filter in `web.xml` (For Servlet 2.5 and earlier)

If you're using older versions of Servlets, you need to configure the filter in `web.xml`.

```
<filter>
    <filter-name>MyFilter</filter-name>
    <filter-class>MyFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>MyFilter</filter-name>
    <url-pattern>/*</url-pattern>  <!-- Applies filter to all requests -->
</filter-mapping>
```

📌 Step 3: Use `@WebFilter` Annotation (For Servlet 3.0 and later)

If you're using **Servlet 3.0+**, you can use annotations instead of `web.xml`.

```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;

@WebFilter("/*")  // Applies filter to all requests
public class MyFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("Filter initialized");
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {

        System.out.println("Pre-processing filter logic");

        chain.doFilter(request, response); // Forward request

        System.out.println("Post-processing filter logic");
    }

    @Override
    public void destroy() {
        System.out.println("Filter destroyed");
    }
}
```

🔹 3️⃣ Types of Filters in Servlets

Filters can be used for various purposes, such as:

Filter Type	Purpose
Authentication Filter	Checks user authentication before accessing a servlet.
Logging Filter	Logs request and response details.
Compression Filter	Compresses response data using GZIP.
Character Encoding Filter	Ensures request encoding is set properly.
XSS/SQL Injection Filter	Removes malicious content from requests.

🔹 4️⃣ Example: Authentication Filter

A filter that restricts access based on a session attribute.

```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter("/secure/*") // Applies filter to URLs starting with /secure/
public class AuthFilter implements Filter {

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

        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
        HttpSession session = req.getSession(false);

        if (session == null || session.getAttribute("user") == null) {
            res.sendRedirect("/login.jsp"); // Redirect to login page if not logged in
        } else {
            chain.doFilter(request, response); // Forward request if authenticated
        }
    }
}
```

🔹 5️⃣ Filter Lifecycle

1️⃣ `init(FilterConfig filterConfig)`

- Called **once** when the filter is created.
- Used for initialization tasks (e.g., reading config parameters).

2️⃣ `doFilter(ServletRequest request, ServletResponse response, FilterChain chain)`

- Called for **each request**.
- Used to process requests and modify responses.

3️⃣ `destroy()`

- Called when the filter is removed.
- Used for cleanup tasks.

🔹 6️⃣ Multiple Filters & Order of Execution

When multiple filters are used, they execute in the order they are defined in `web.xml` or by annotation priority.

Example with Multiple Filters

```
<filter-mapping>
   <filter-name>FirstFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
   <filter-name>SecondFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Execution Order:1️⃣`FirstFilter` → **Pre-processing**
2️⃣`SecondFilter` → **Pre-processing**
3️⃣**Servlet execution**
4️⃣`SecondFilter` → **Post-processing**
5️⃣`FirstFilter` → **Post-processing**

🔹 7️⃣ Advantages of Using Filters

✔ **Reusability:** One filter can be used for multiple servlets.
✔ **Separation of Concerns:** Helps keep servlets clean by handling pre/post-processing separately.
✔ **Security:** Filters help in **authentication, authorization, and input validation**.
✔ **Performance Optimization:** Can be used for **response compression, caching, and request logging**.

📌 Summary

✔ **Filters** are used to intercept and process requests/responses before reaching a servlet.
✔ **Implements `Filter` interface** with `init()`, `doFilter()`, and `destroy()` methods.
✔ Can be **configured via `web.xml` (older versions) or `@WebFilter` annotation (Servlet 3.0+)**.
✔ Commonly used for **authentication, logging, compression, and request modification**.
✔ **Order of execution** matters when multiple filters are used.
