Experiment 8

Servlet with Filter

Name: Sai Harsha Vardhan AVN

Roll-no: BCSE1823

Batch: ‘A’

Aim: Write servlet with Filter program and implement it.

Theory:

Servlets are Java classes which service HTTP requests and implement the **javax.servlet.Servlet** interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Steps to create a servlet example

There are given 6 steps to create a **servlet example**. These steps are required for all the servers.

The servlet example can be created by three ways:

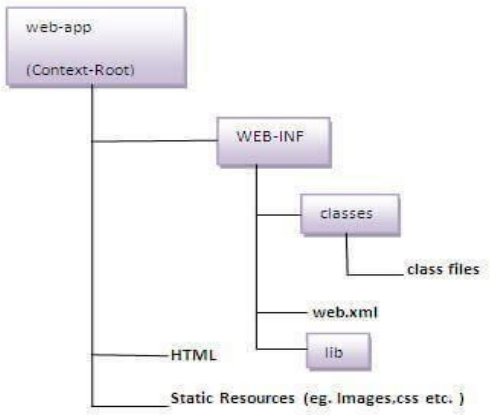
1. By implementing Servlet interface,
2. By inheriting GenericServlet class, (or)
3. By inheriting HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

There are few server's to run your application , you can use either of those for example we have **apache tomcat server, Glassfishserver etc**. The steps are as follows:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

Directory/File Structure for Servlet Programs to run is shown below:



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 the servlet.

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

1. Filter is pluggable.
2. One filter doesn't have dependency onto another resource.
3. Less Maintenance

Servlet Filter interface

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

Servlet Filter interface lifecycle methods are:

1. **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.
2. **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**.**
3. **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.

Syntax:

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

Code:

1) ***Web.xml Code:***

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd">
  <filter>
    <filter-name>NewFilter</filter-name>
    <filter-class>NewFilter</filter-class>
  </filter>
  <filter>
    <filter-name>MyFilter</filter-name>
    <filter-class>MyFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>MyFilter</filter-name>
    <servlet-name>AdminServlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>ERROR</dispatcher>
  </filter-mapping>
  <filter-mapping>
    <filter-name>NewFilter</filter-name>
    <servlet-name>AdminServlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>ERROR</dispatcher>
  </filter-mapping>
  <servlet>
    <servlet-name>AdminServlet</servlet-name>
    <servlet-class>AdminServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>AdminServlet</servlet-name>
    <url-pattern>/AdminServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

2) Index.html Code:

```
<html>
<head>
   <title>Authentication</title>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
   <form action="AdminServlet">
      Name:<input type="text" name="name" /><br/> Password:
      <input type="password" name="password" /><br/>
      <input type="submit" value="login">
   </form>
</body>
</html>
```

3) AdminServlet.java:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.*;

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();
   }
}
```

4) MyFilter.java:

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

public class MyFilter implements Filter {

   public void init(FilterConfig arg0) throws ServletException {
   }

   public void doFilter(ServletRequest req, ServletResponse resp,
         FilterChain chain) throws IOException, ServletException {
      PrintWriter out = resp.getWriter();
      String password = req.getParameter("password");
      if (password.equals("admin")) {
         chain.doFilter(req, resp);//sends request to next resource
      } else {
         out.print("username or password error!");
//RequestDispatcher rd=req.getRequestDispatcher("index.html");
//rd.include(req, resp);
      }
   }

   public void destroy() {
   }
}
```

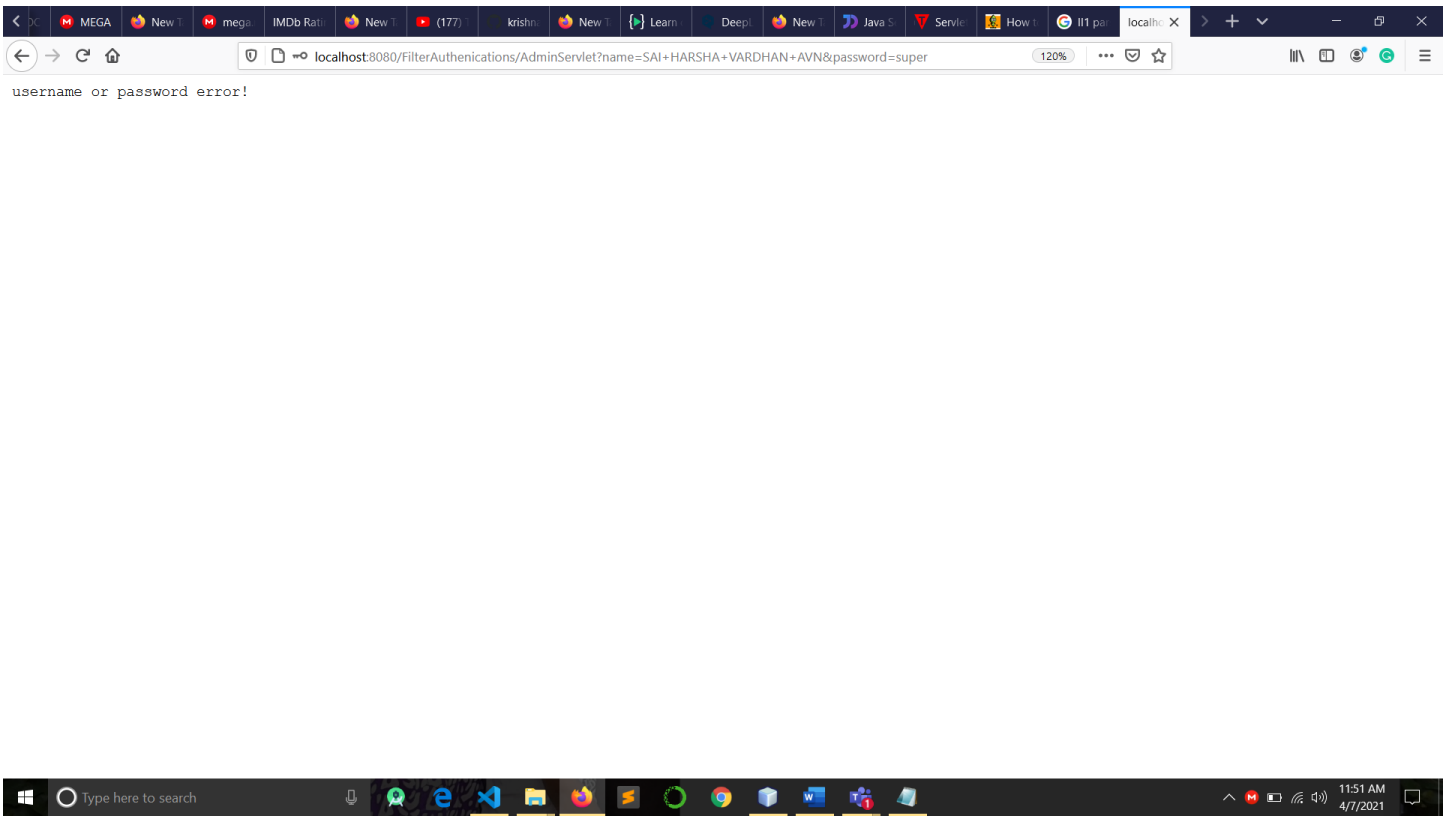
Output:

Name:

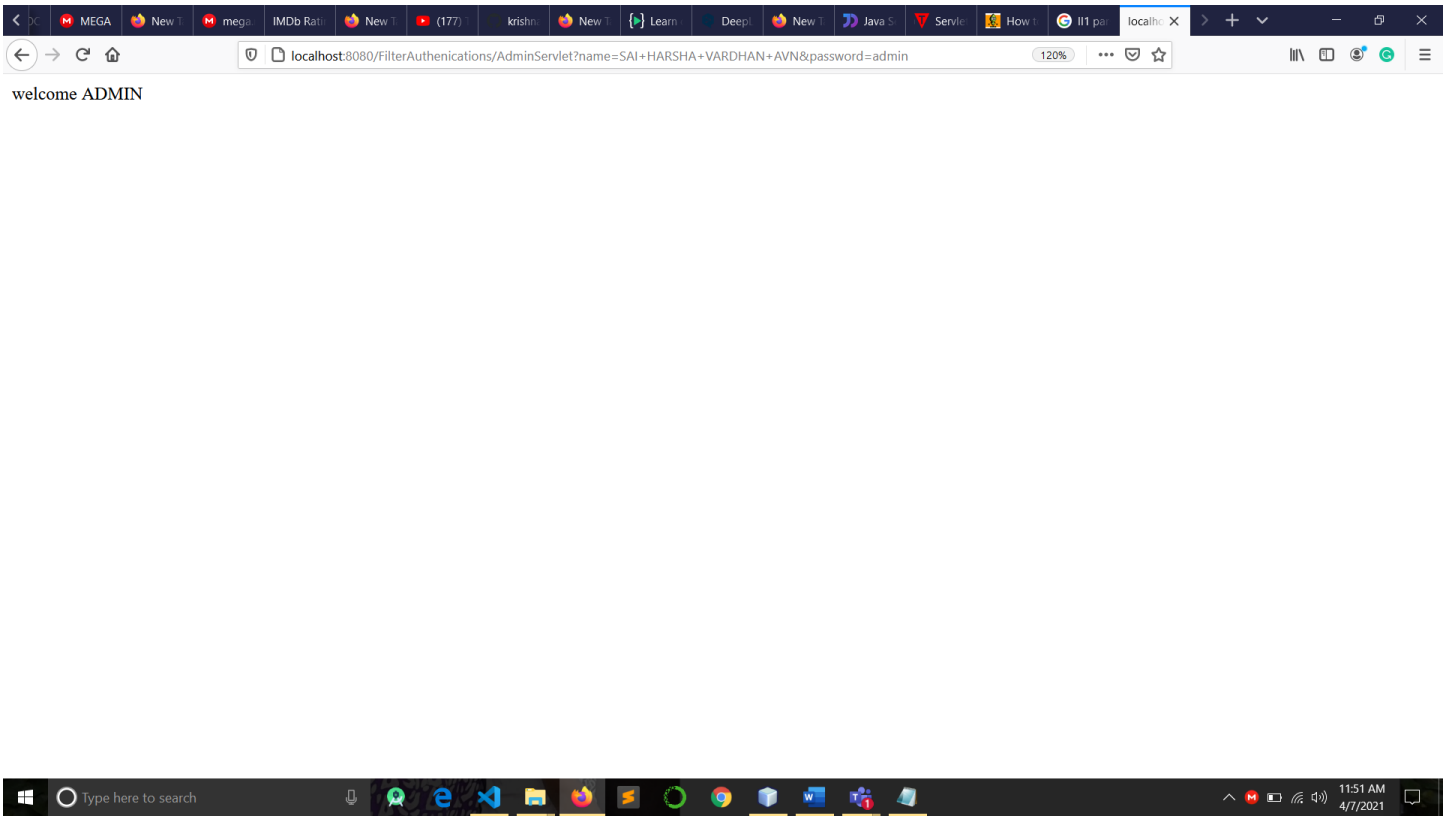
Password:

login

1) When the login user is an invalid



1) When the login user is an invalid



Conclusion: Servlet with Filter program implemented successfully.