# Filter

#Filter #Controller #JavaEE

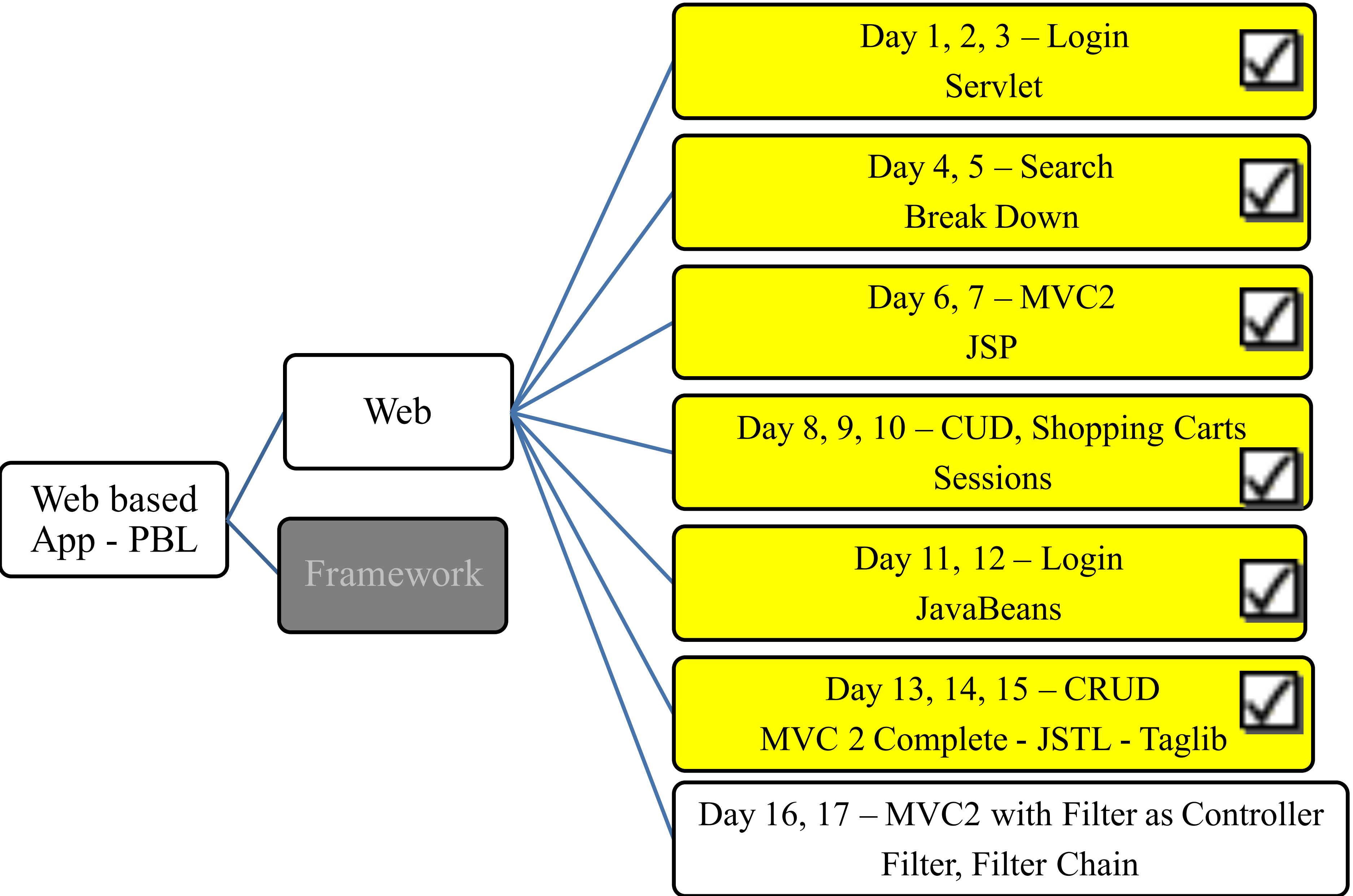
Review

- **How to remove all java code in JSP (View)? Complete the MVC 2 Design Pattern with View**
  - **JSTL**
- **How to build the data grid tag library using in JSP?**
  - **Tag Libraries**
    - Model
    - Classical, Simple, and Handles
    - How to implement the custom Tag Lib and use it in JSP

Objectives

- **How to build application using MVC2 Pattern using Filter as Controller?**
  - Filter
  - Filter Chain

Objectives


Web based App - PBL
Web
Framework
Day 1, 2, 3 – Login
Servlet
Day 4, 5 – Search
Break Down
Day 6, 7 – MVC2
JSP
Day 8, 9, 10 – CUD, Shopping Carts
Sessions
Day 11, 12 – Login
JavaBeans
Day 13, 14, 15 – CRUD
MVC 2 Complete - JSTL - Taglib
Day 16, 17 – MVC2 with Filter as Controller
Filter, Filter Chain

Filter
Requirements

- **Preprocess request** and **Postprocess response** in web application
- **Build** the Project CRUD, Shopping **applying MVC2 pattern with Filter as Controller**

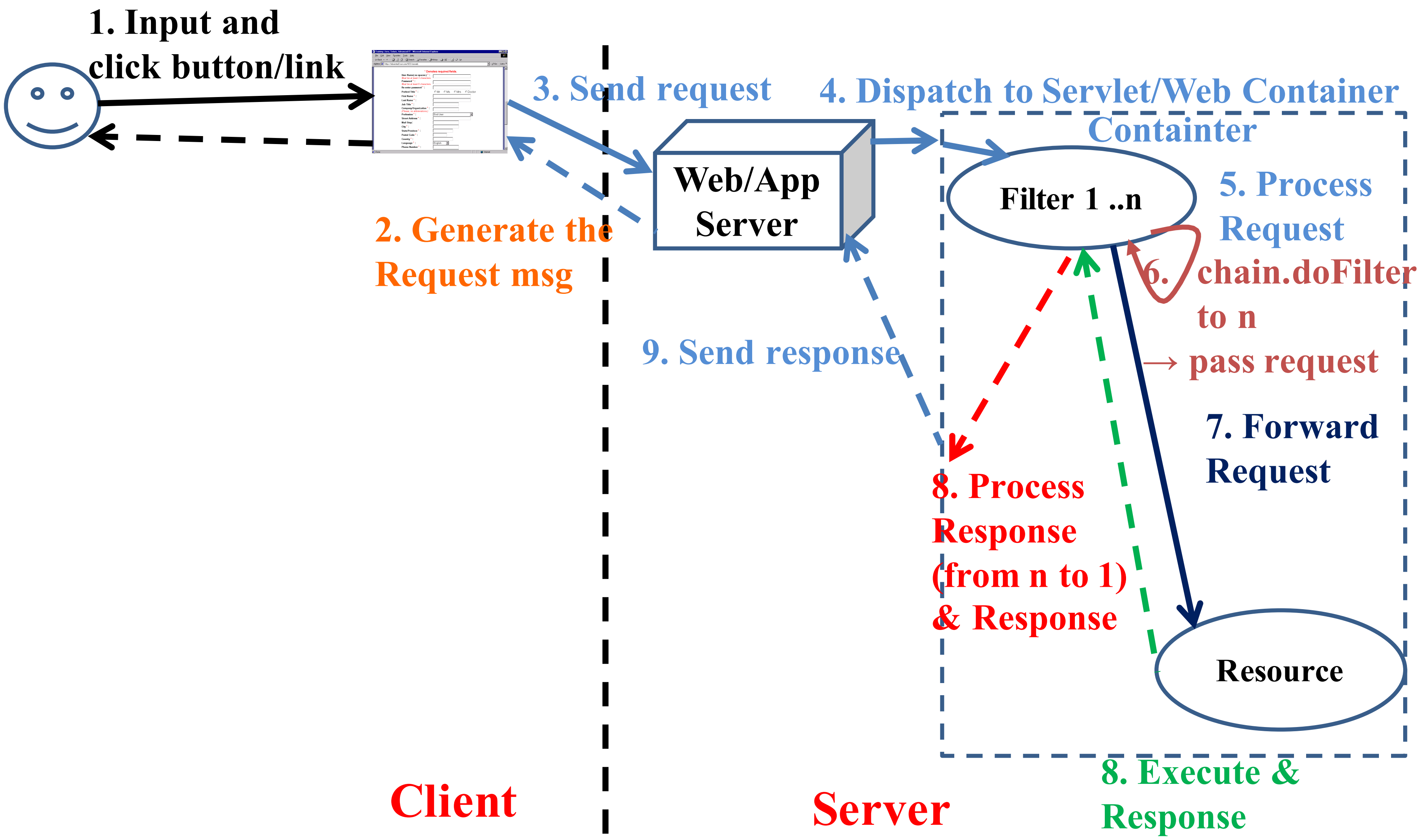
Filter

Overview

- **Are components that add functionality to the request and response processing of a Web Application**
  - **Intercept** the requests and response that flow between a client and a Servlet/JSP.
  - Supports **dynamic modification of requests and responses** between client and web applications.
  - Dynamically **access incoming requests** from the user before the servlet processes the request
  - **Access the outgoing response** from the web resources before it reaches the user
- **Categorized** according to the **services** they provide to the web applications
- **Resides** in the **web container** along with the web applications
- Was introduced as a Web component in Java servlet specification version 2.3

Filter

Interactive Filter Model


1. Input and click button/link
2. Generate the Request msg
3. Send request
4. Dispatch to Servlet/Web Container Containter
Web/App Server
Filter 1 ..n
5. Process Request
6. chain.doFilter to n → pass request
7. Forward Request
8. Process Response (from n to 1) & Response
Resource
8. Execute & Response
9. Send response
Client
Server

Filter

Usage

- Authorize request
- Altering request headers and modify data
- Modify response headers and data
- Authenticating the user
- Comprising files
- Encrypting data
- Converting images
- Logging and auditing filters
- Filters that trigger resource access events

Filter

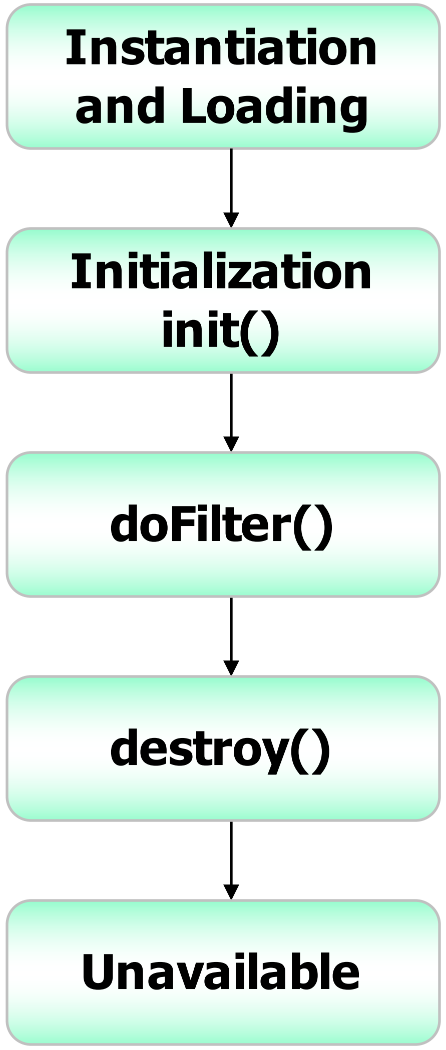
Benefits - Advantages

- **Optimization** of the **time** taken to send a response
- **Compression** of the **content** size before sending
- **Optimization** of the **bandwidth**
- **Security**
- **Identify** the **type of request** coming from the Web client, such as HTTP and FTP, and invoke the Servlet that needs to process the request.
- **Retrieve** the user information from the request parameters to authenticate the user.
- **Validate** a client using Servlet filters before the client accesses the Servlet.
- **Identify** the **information** about the MIME types and other header contents of the request.
- **Facilitate** a **Servlet** to **communicate** with the **external resources.**
- **Intercept** responses and compress it before sending the response to the client

Filter

Life Cycle

- Working of Filter
  - The filter intercepts the request from a user to the servlet
  - The filter then provides customized services
  - The filter sends the serviced response or request to the appropriate destination



Filter API

- **Creates and handles** the functionalities of a filter
- Contains **three interfaces**
  - Filter Interface, FilterConfig Interface, FilterChain Interface
- **Filter Interface**
  - Must be implemented to create a filter class **extends javax.servlet.Filter**
  - An object performs filtering tasks on the request and the response

Methods	Descriptions
init	- **public void init(FilterConfig fg);** - Called by the servlet container to initialize the filter - Called only once - Must complete successfully before the filter is asked to do any filtering work
doFilter	- **public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException** - Called by the container each time a request or response is processed - Then examines the request/response headers & customizes them as per the requirements - Passed the request/response through the FilterChain object to the next entity in the chain
destroy	- **public void destroy();** - Called by the servlet container to inform the filter that its service is no more required - Called only once.

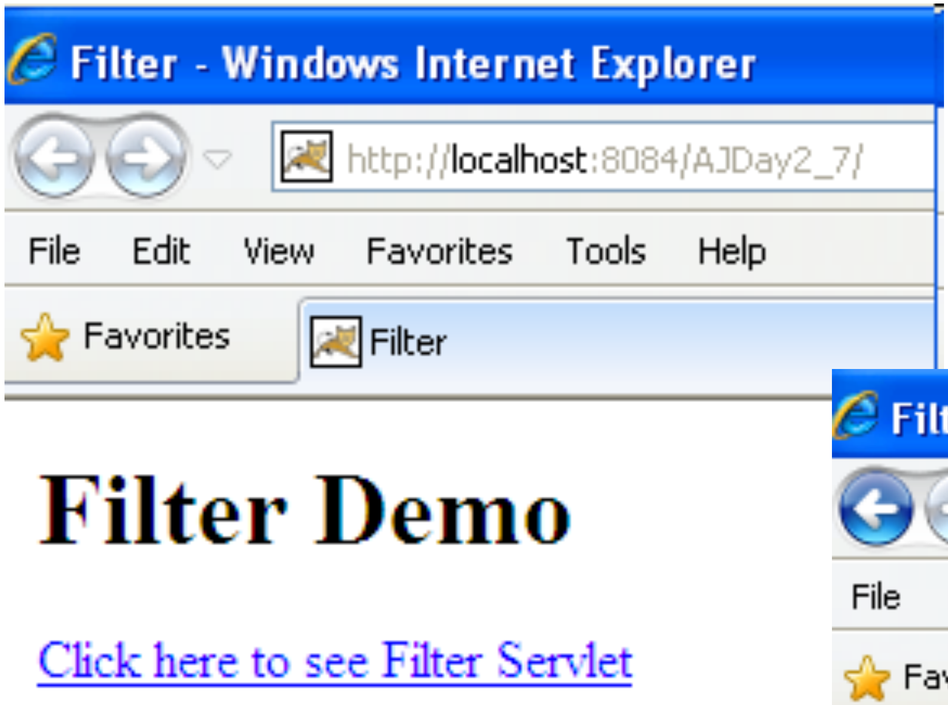
Filter Configuration

- **In Web Deployment Descriptor**

```
<web-app>
….
<filter>
      <filter-name>Name of Filters</filter-name>
      <filter-class>implemented Filter Class</filter-class>
       [<init-param>
              <param-name>parameter name</param-name>
               <param-value>value </param-value>
       </init-param>]
    </filter>
    <filter-mapping>
        <filter-name>FilterName</filter-name>
        <url-pattern>/context</url-pattern>
    </filter-mapping>
    ….
</web-app>
```

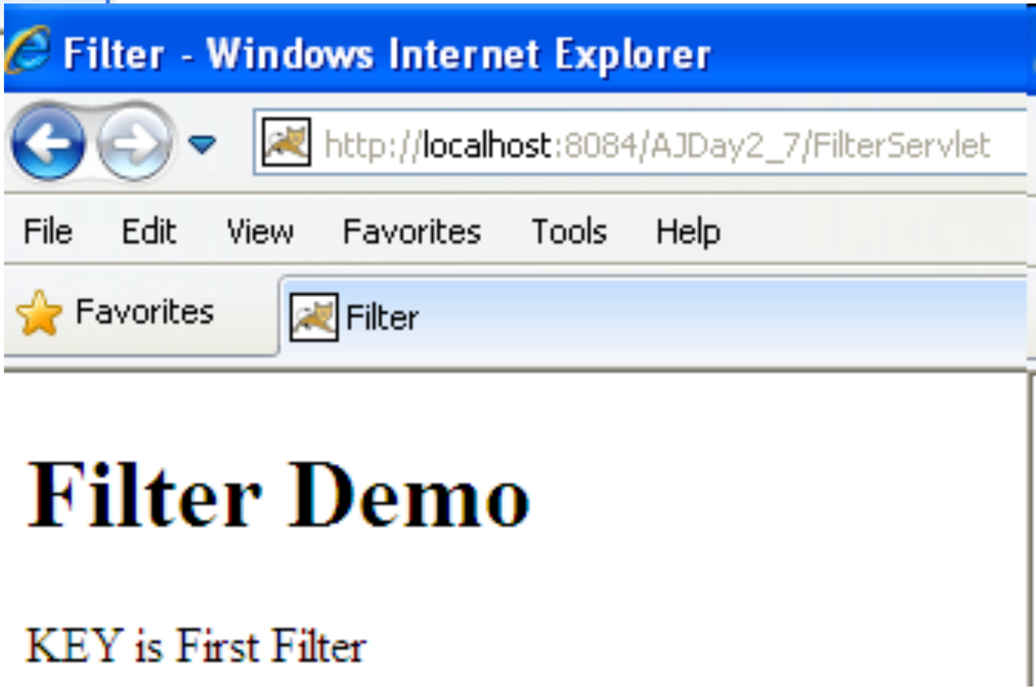
Filter
Example

- Building the web application shows as the following GUI in sequence


Filter - Windows Internet Explorer
http://localhost:8084/AJDay2_7/
File Edit View Favorites Tools Help
Favorites
Filter

Filter Demo

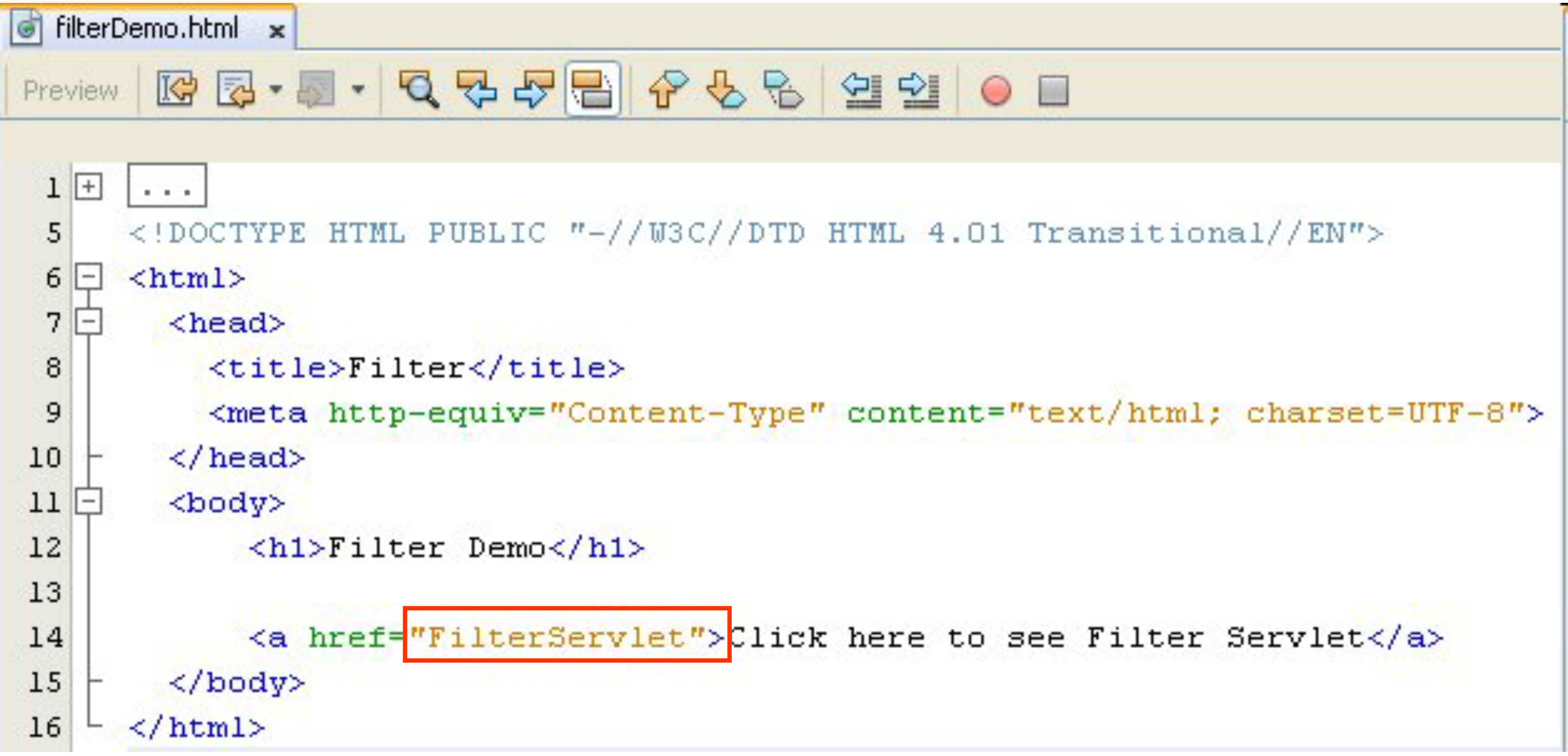
Click here to see Filter Servlet


Filter - Windows Internet Explorer
http://localhost:8084/AJDay2_7/FilterServlet
File Edit View Favorites Tools Help
Favorites
Filter

Filter Demo

KEY is First Filter

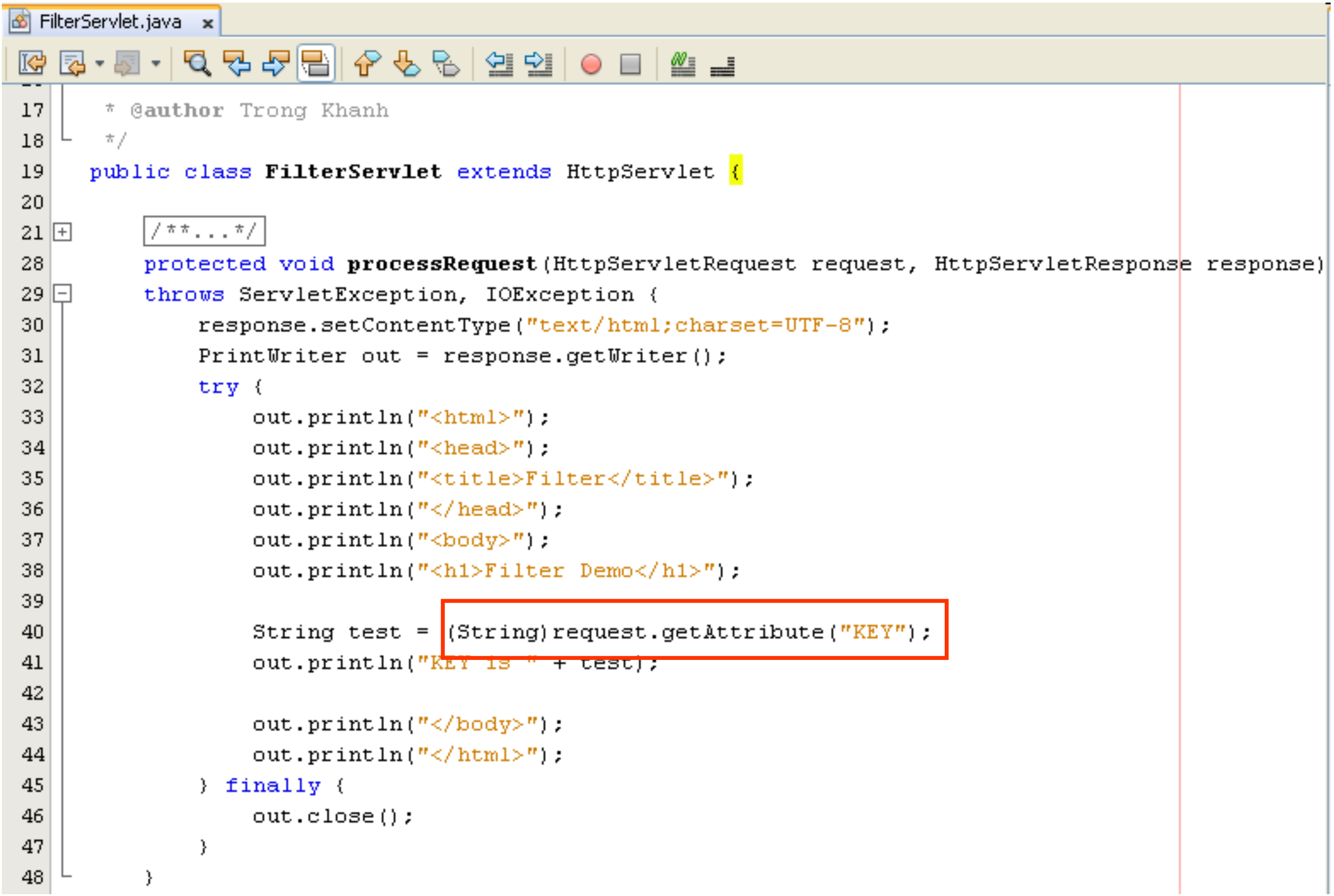
Filter
Example


filterDemo.html

Preview

```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Filter</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
      <h1>Filter Demo</h1>

      <a href="FilterServlet">Click here to see Filter Servlet</a>
  </body>
</html>
```

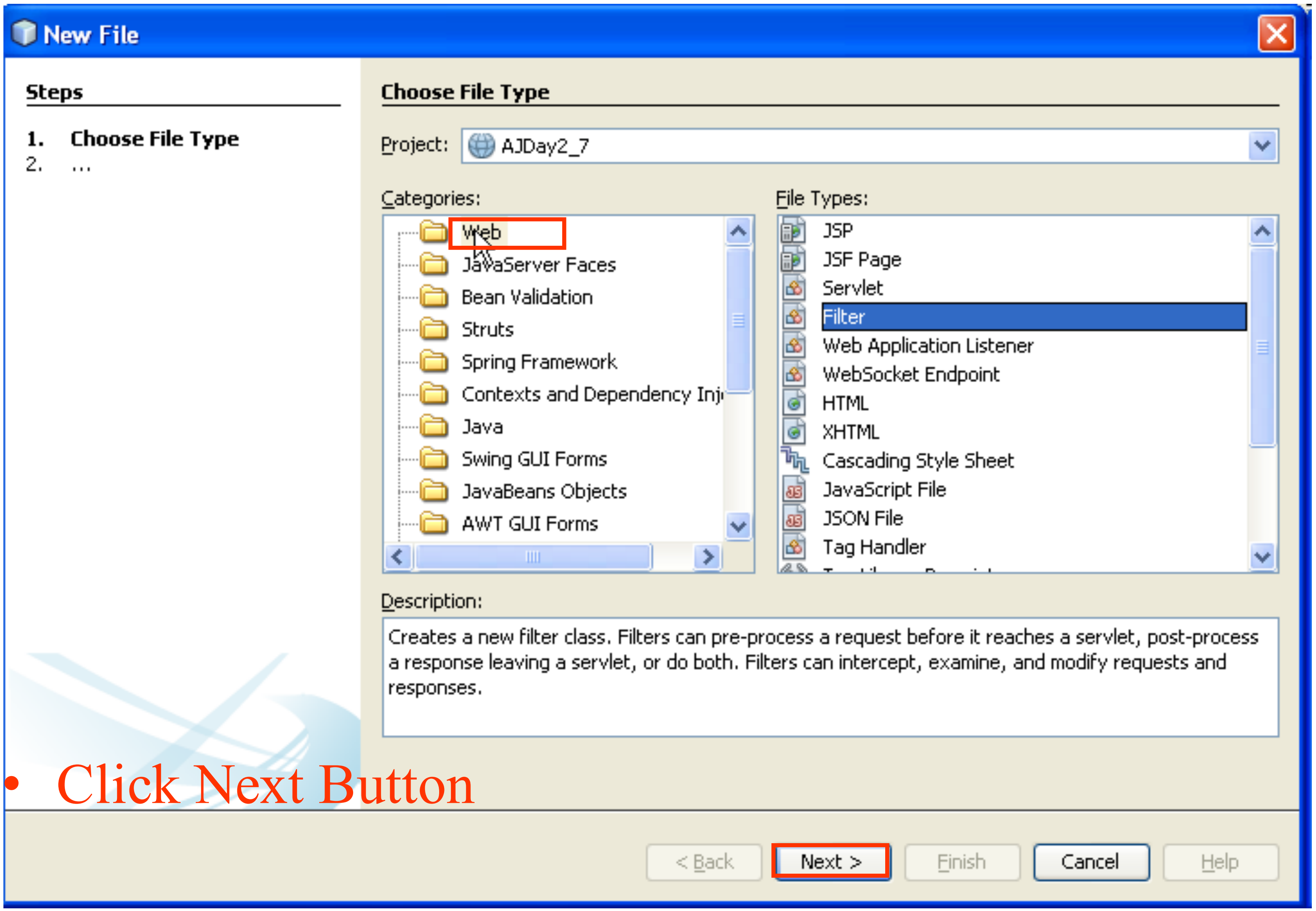
Filter
Example



FilterServlet.java

```
     * @author Trong Khanh
     */
public class FilterServlet extends HttpServlet {

    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Filter</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Filter Demo</h1>");

            String test = (String)request.getAttribute("KEY");
            out.println("KEY is " + test);

            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

Filter
Example

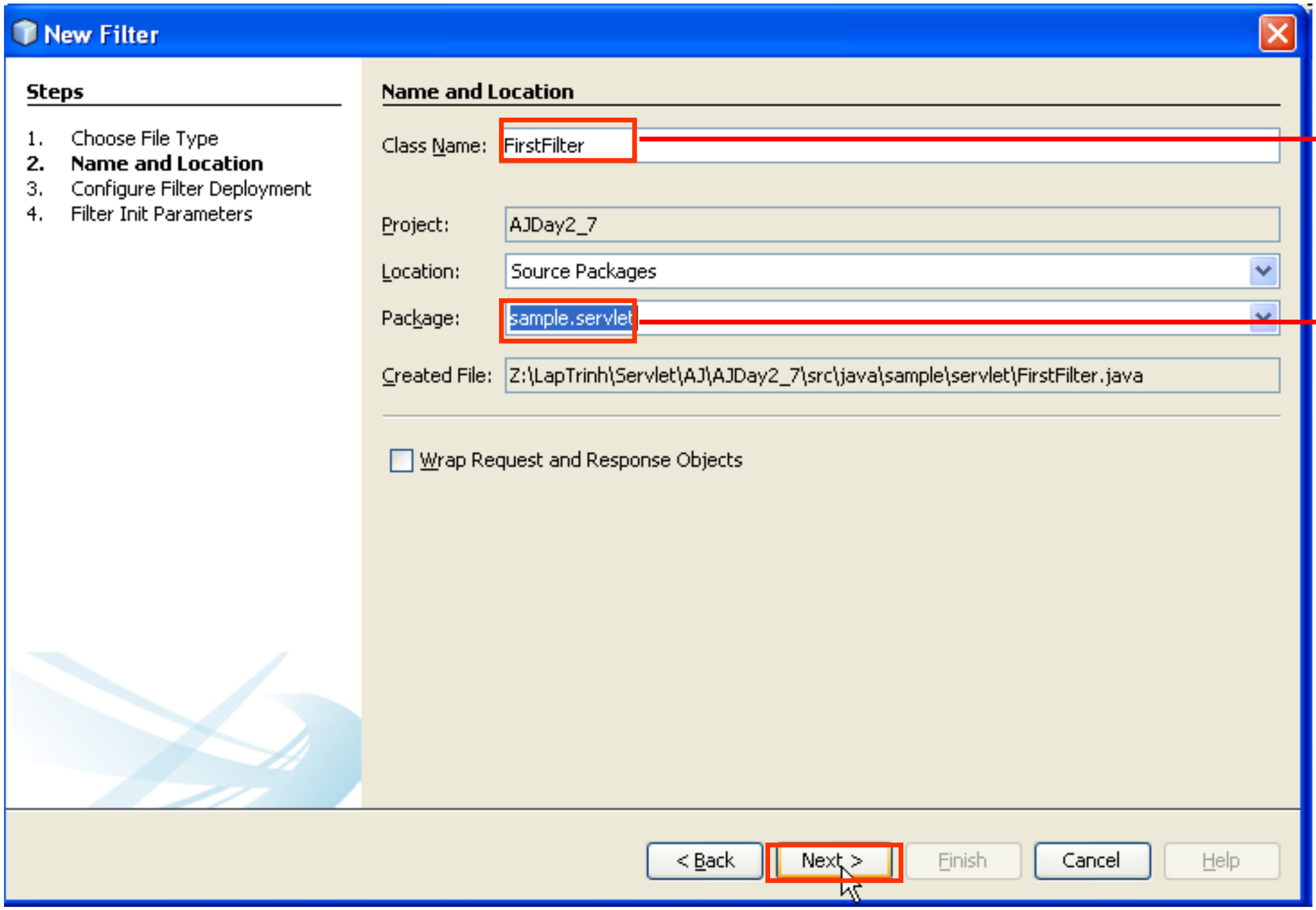


Description:

Creates a new filter class. Filters can pre-process a request before it reaches a servlet, post-process a response leaving a servlet, or do both. Filters can intercept, examine, and modify requests and responses.

- Click Next Button

Filter
Example


New Filter

Steps

1. Choose File Type
2. **Name and Location**
3. Configure Filter Deployment
4. Filter Init Parameters

Name and Location

Class Name: FirstFilter

Project: AJDay2_7

Location: Source Packages

Package: sample.servlet

Created File: Z:\LapTrinh\Servlet\AJ\AJDay2_7\src\java\sample\servlet\FirstFilter.java

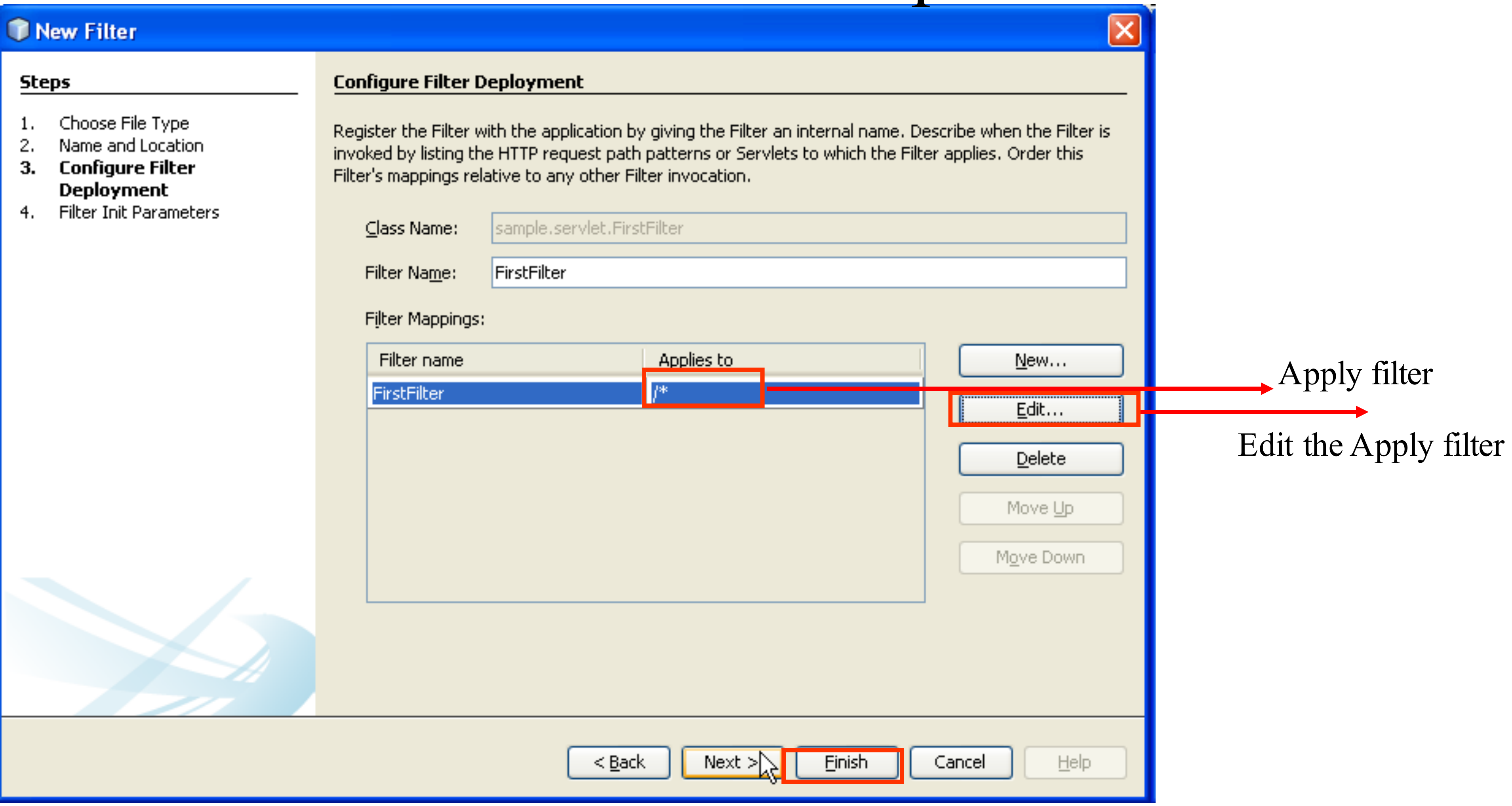
☐ Wrap Request and Response Objects

Fill your filter name

Fill/choose package name

- Click Next Button

Filter
Example


New Filter

Steps

1. Choose File Type
2. Name and Location
3. **Configure Filter Deployment**
4. Filter Init Parameters

Configure Filter Deployment

Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.

Class Name: sample.servlet.FirstFilter

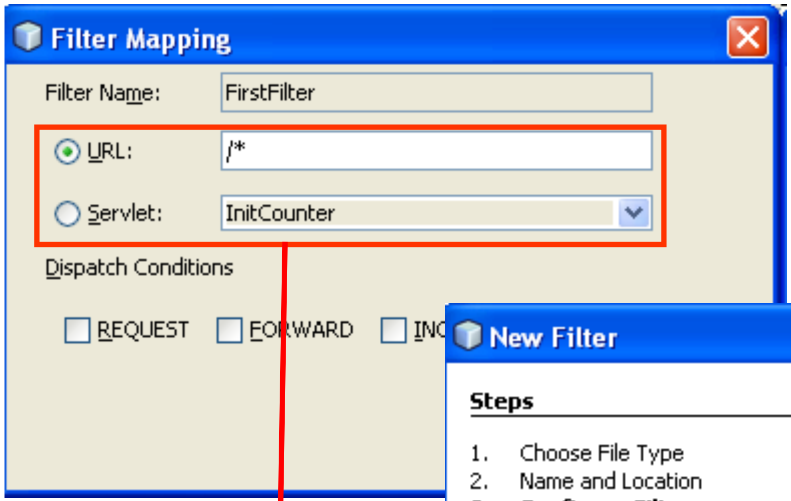
Filter Name: FirstFilter

Filter Mappings:

Filter name	Applies to
FirstFilter	/*

- Click Edit Button to apply Filter the selected Servlet
- Otherwise, click Finish Button

Filter
Example



Filter Mapping

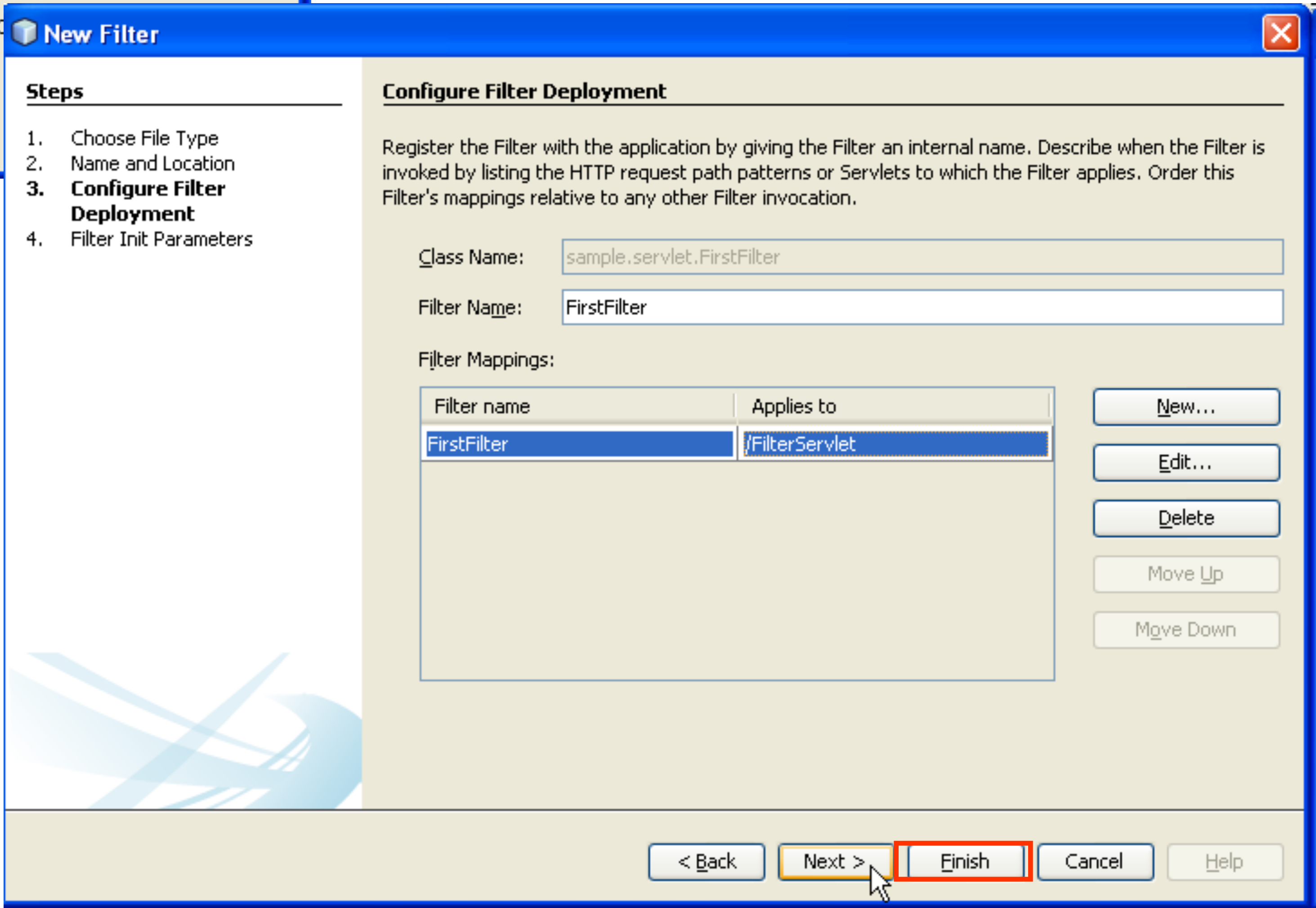
Filter Name: FirstFilter

(● URL:) /*

○ Servlet: InitCounter

Dispatch Conditions

☐ REQUEST ☐ FORWARD ☐ INC



New Filter

Steps

1. Choose File Type
2. Name and Location
3. **Configure Filter Deployment**
4. Filter Init Parameters

Configure Filter Deployment

Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.

Class Name: sample.servlet.FirstFilter

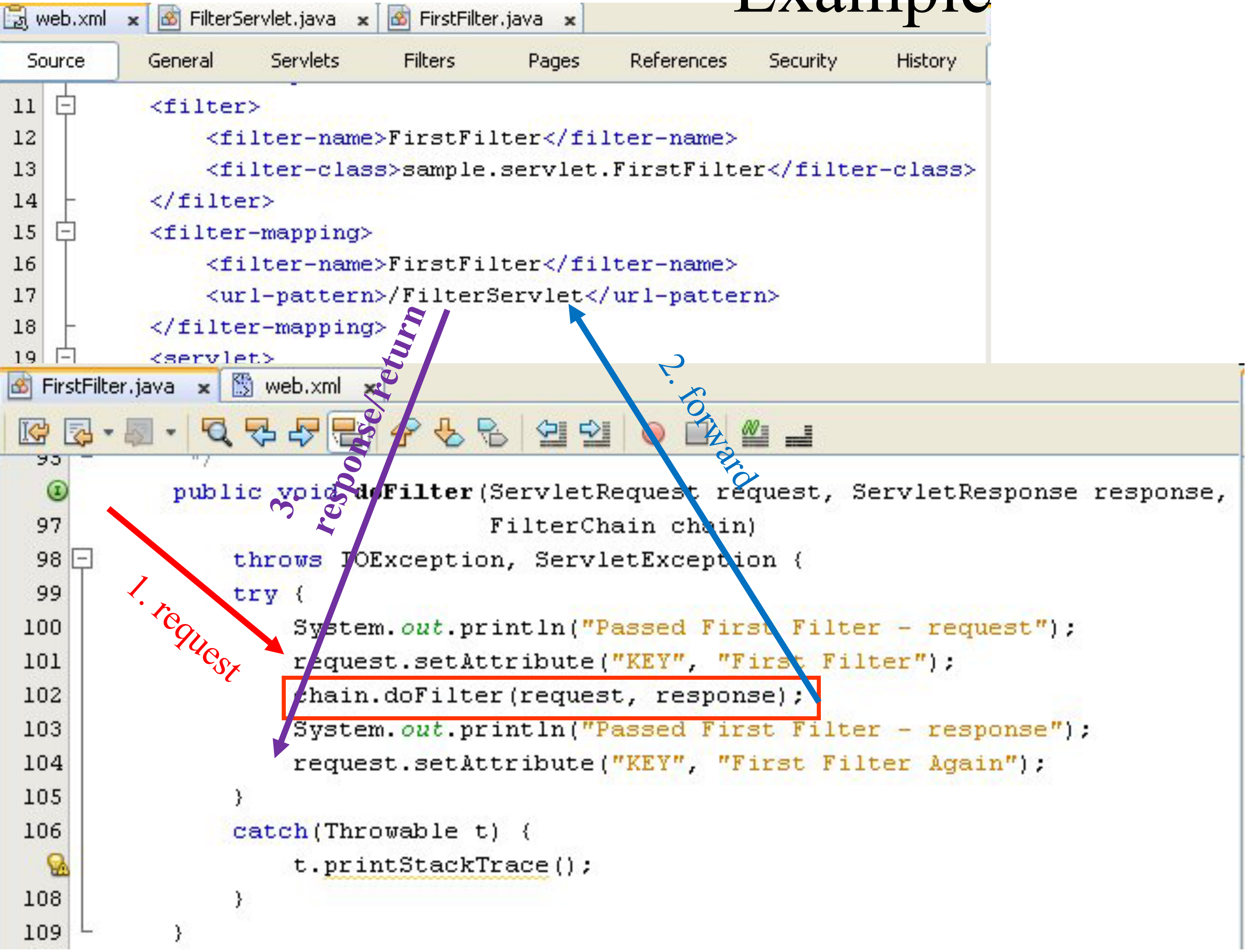
Filter Name: FirstFilter

Filter Mappings:

Filter name	Applies to
FirstFilter	/FilterServlet

Select the URL and typing the URL string, or Select the Servlet and choose the approximate Servlet in combo box

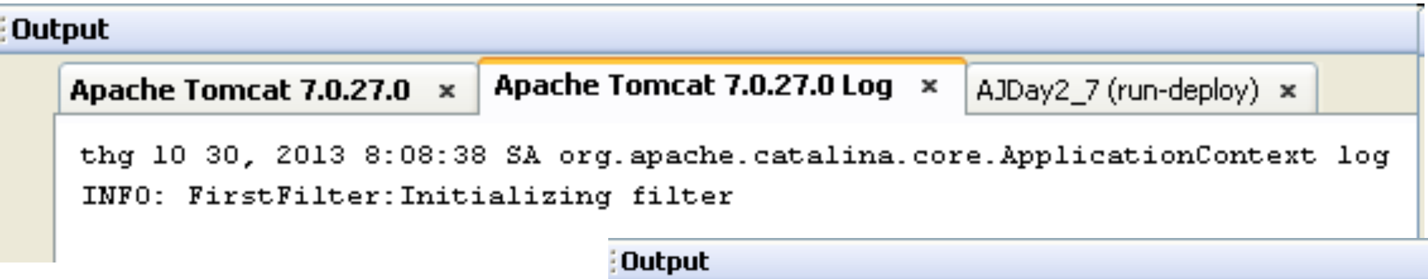
Filter
Example



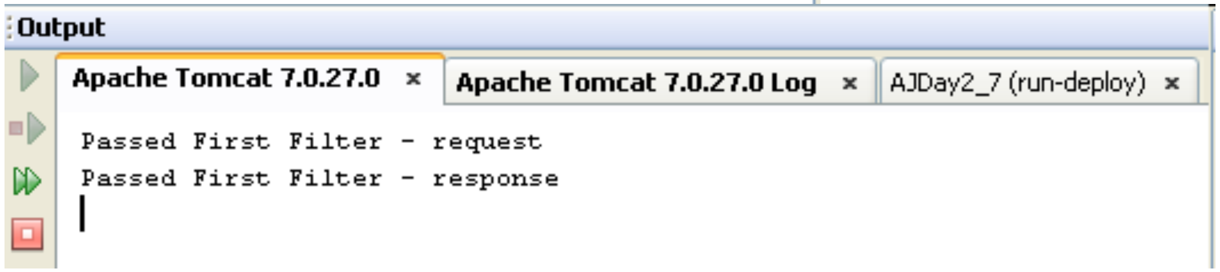
```
    <filter>
        <filter-name>FirstFilter</filter-name>
        <filter-class>sample.servlet.FirstFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>FirstFilter</filter-name>
        <url-pattern>/FilterServlet</url-pattern>
    </filter-mapping>
    <servlet>
```

```
    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain)
        throws IOException, ServletException {
        try {
            System.out.println("Passed First Filter - request");
            request.setAttribute("KEY", "First Filter");
            chain.doFilter(request, response);
            System.out.println("Passed First Filter - response");
            request.setAttribute("KEY", "First Filter Again");
        }
        catch(Throwable t) {
            t.printStackTrace();
        }
    }
```

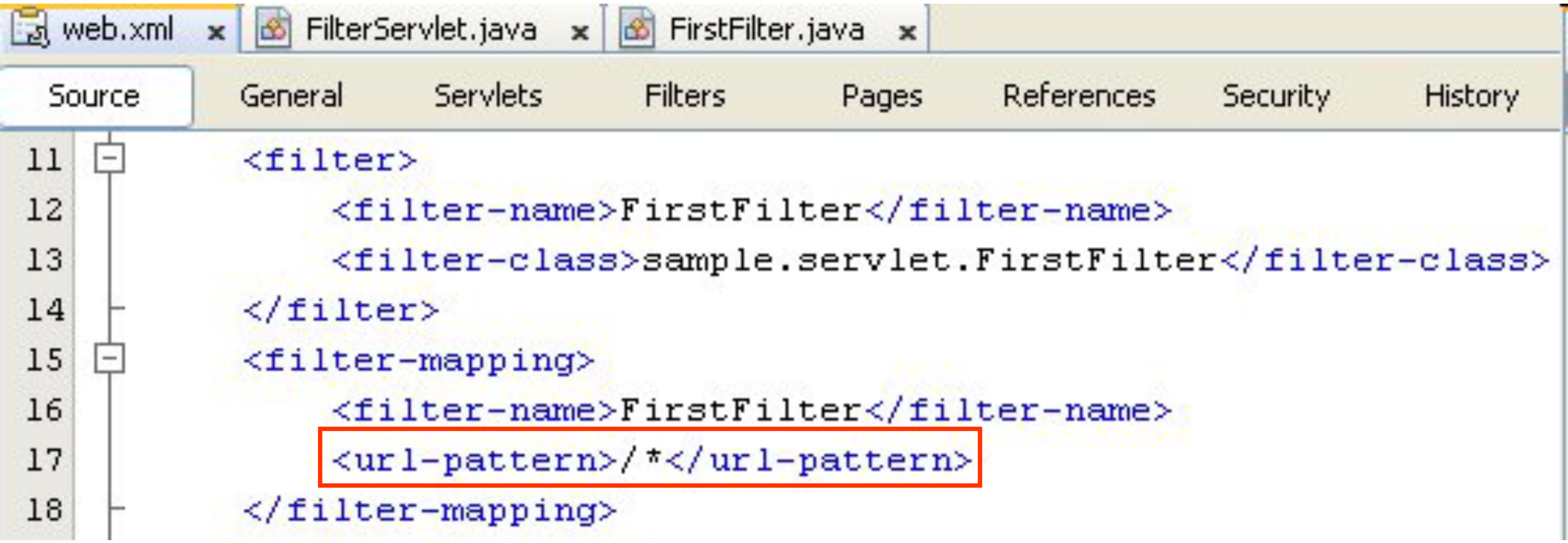
Filter
Example



```
thg 10 30, 2013 8:08:38 SA org.apache.catalina.core.ApplicationContext log
INFO: FirstFilter:Initializing filter
```

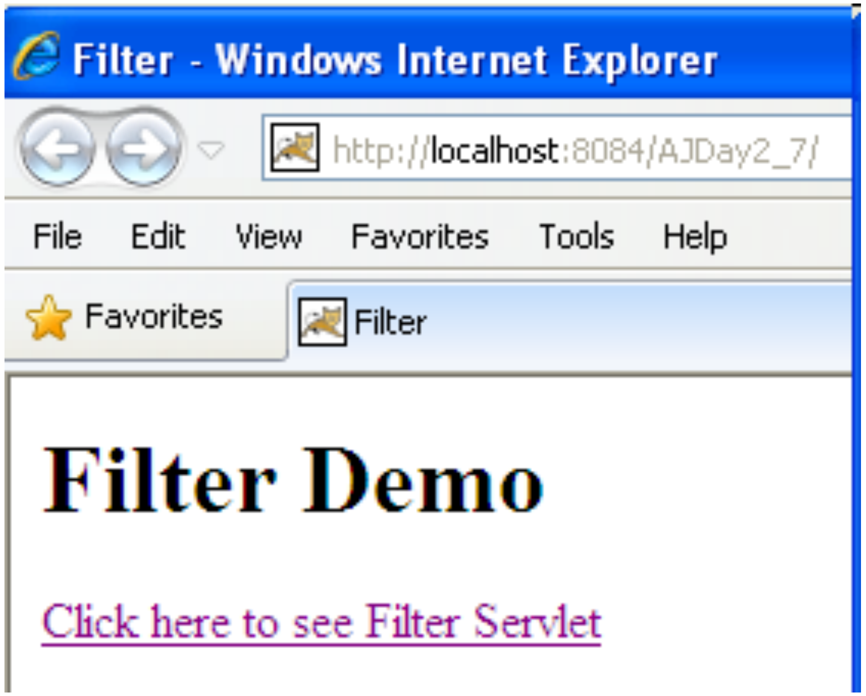


```
Passed First Filter - request
Passed First Filter - response
```



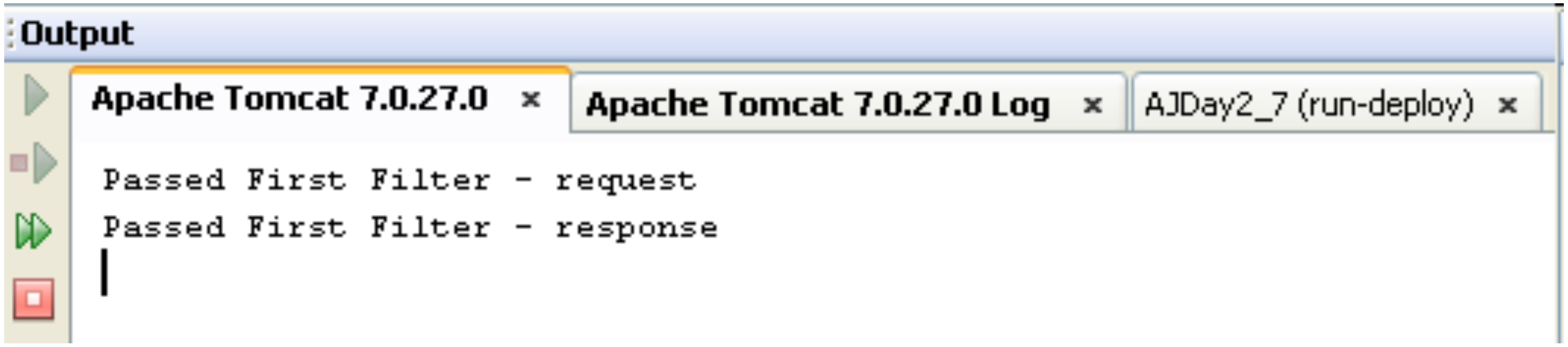
```
    <filter>
        <filter-name>FirstFilter</filter-name>
        <filter-class>sample.servlet.FirstFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>FirstFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Filter
Example


Filter - Windows Internet Explorer
http://localhost:8084/AJDay2_7/
File Edit View Favorites Tools Help
Favorites
Filter

Filter Demo

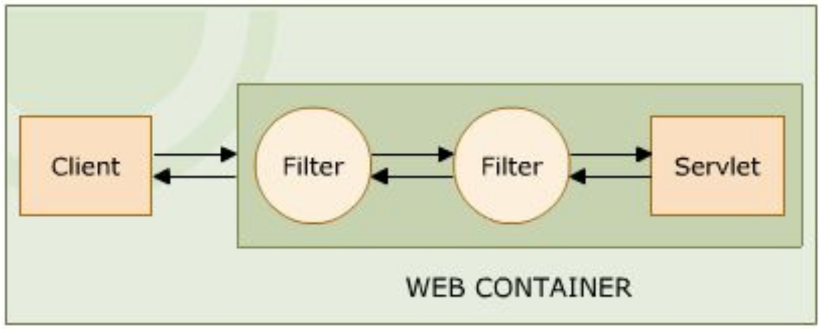
Click here to see Filter Servlet



```
Passed First Filter - request
Passed First Filter - response
```

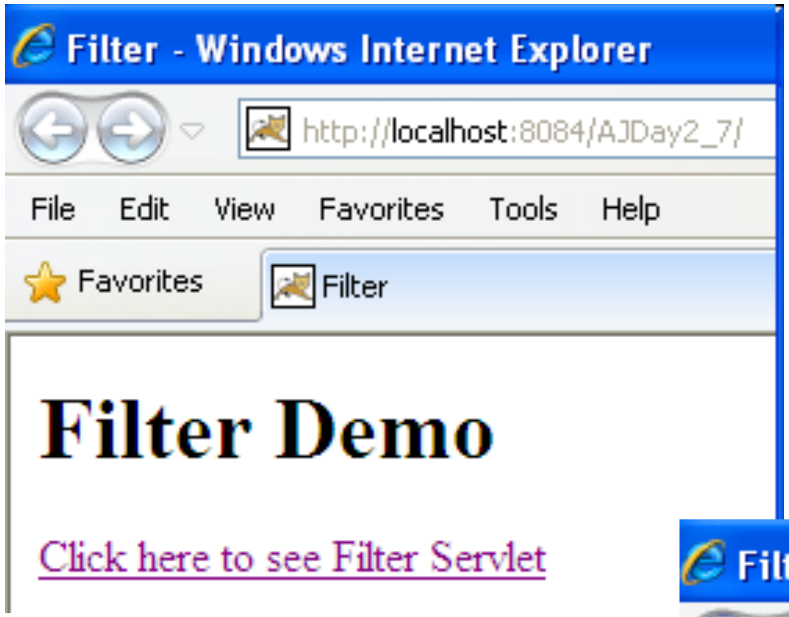
Filter Chain
Definition

- There can be **more than one filter** between the user and the endpoint – Invoke a **series of filters**
- A request or a response is **passed through one** filter to the **next** in the filter chain. So each request and response has to be serviced by each filter forming a filter chain
- If the Calling filter is last filter, will invoke web resource
- **FilterChain Interface**
  - Provides an object through the web container
  - The object invokes the next filter in a filter chain starting from the first filter from a particular end. If the calling filter is the last filter in the chain, it will invoke the web resource, such as JSP and servlet.
  - Only implement doFilter() method.
  - Forces the next filter in the chain to be invoked



Filter Chain

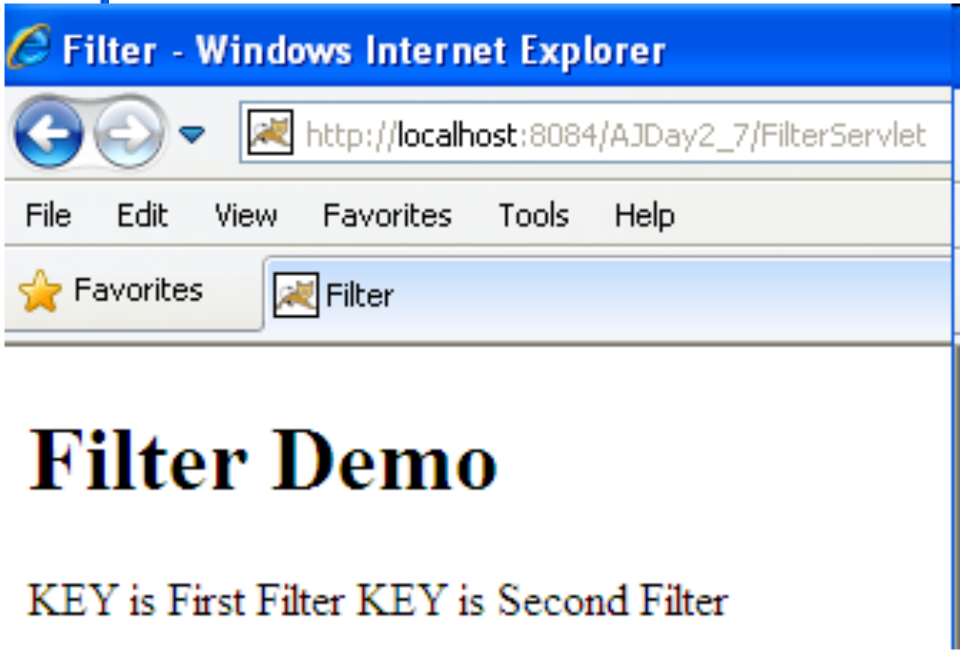
Filter Chain
Example


Filter - Windows Internet Explorer

http://localhost:8084/AJDay2_7/

Filter Demo

Click here to see Filter Servlet

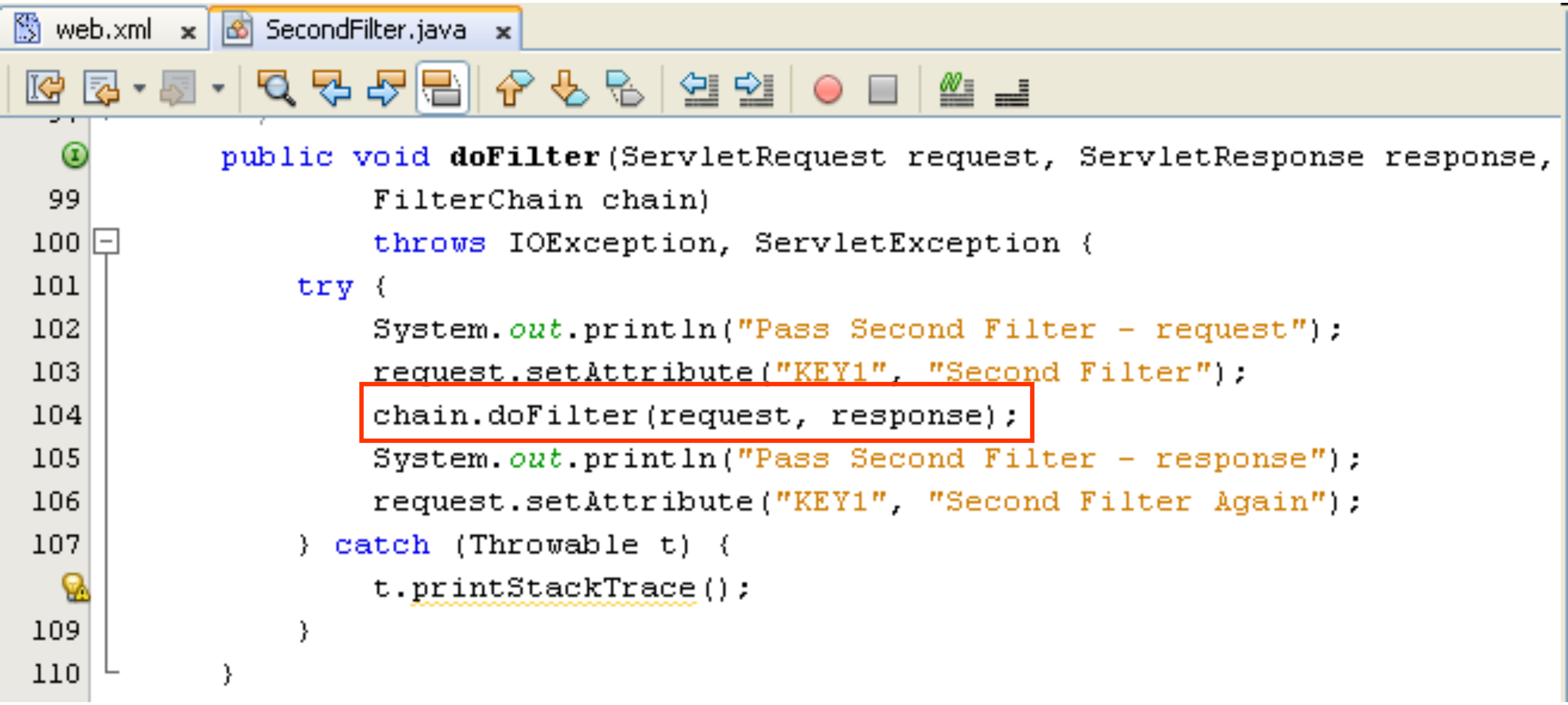

Filter - Windows Internet Explorer

http://localhost:8084/AJDay2_7/FilterServlet

Filter Demo

KEY is First Filter KEY is Second Filter

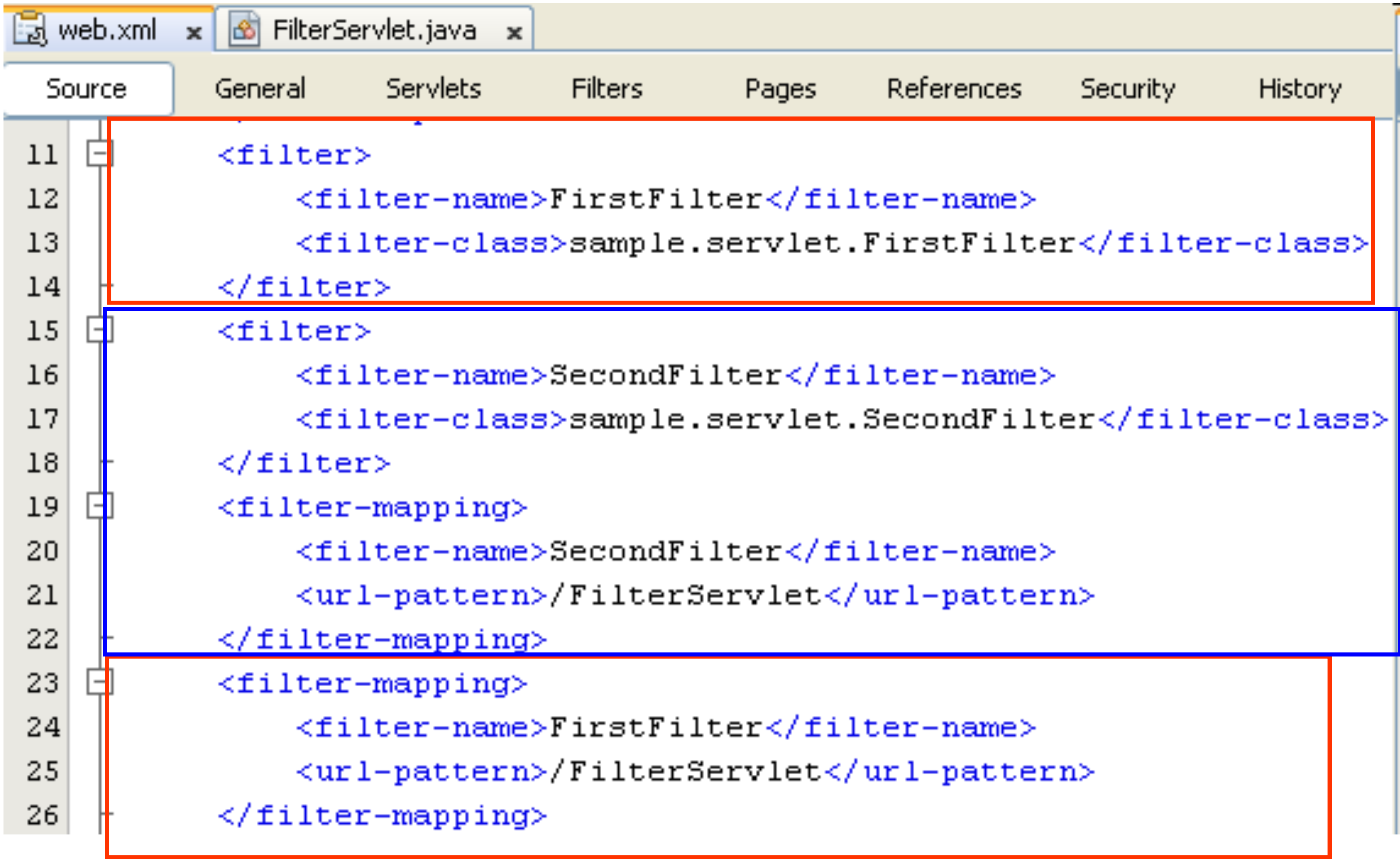
Filter Chain
Example



web.xml
SecondFilter.java

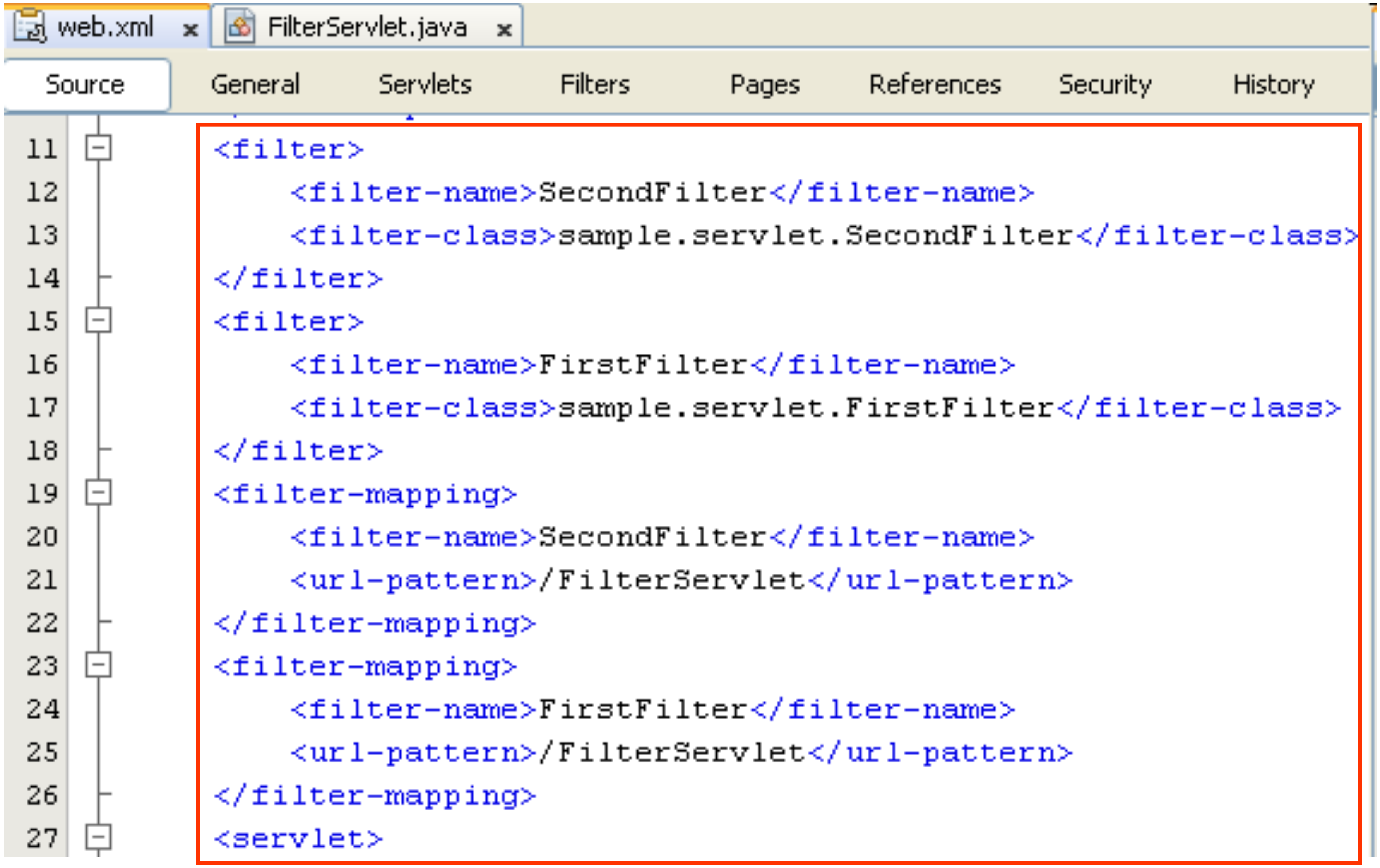
```java
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        try {
            System.out.println("Pass Second Filter - request");
            request.setAttribute("KEY1", "Second Filter");
            chain.doFilter(request, response);
            System.out.println("Pass Second Filter - response");
            request.setAttribute("KEY1", "Second Filter Again");
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
```

Filter Chain
Example



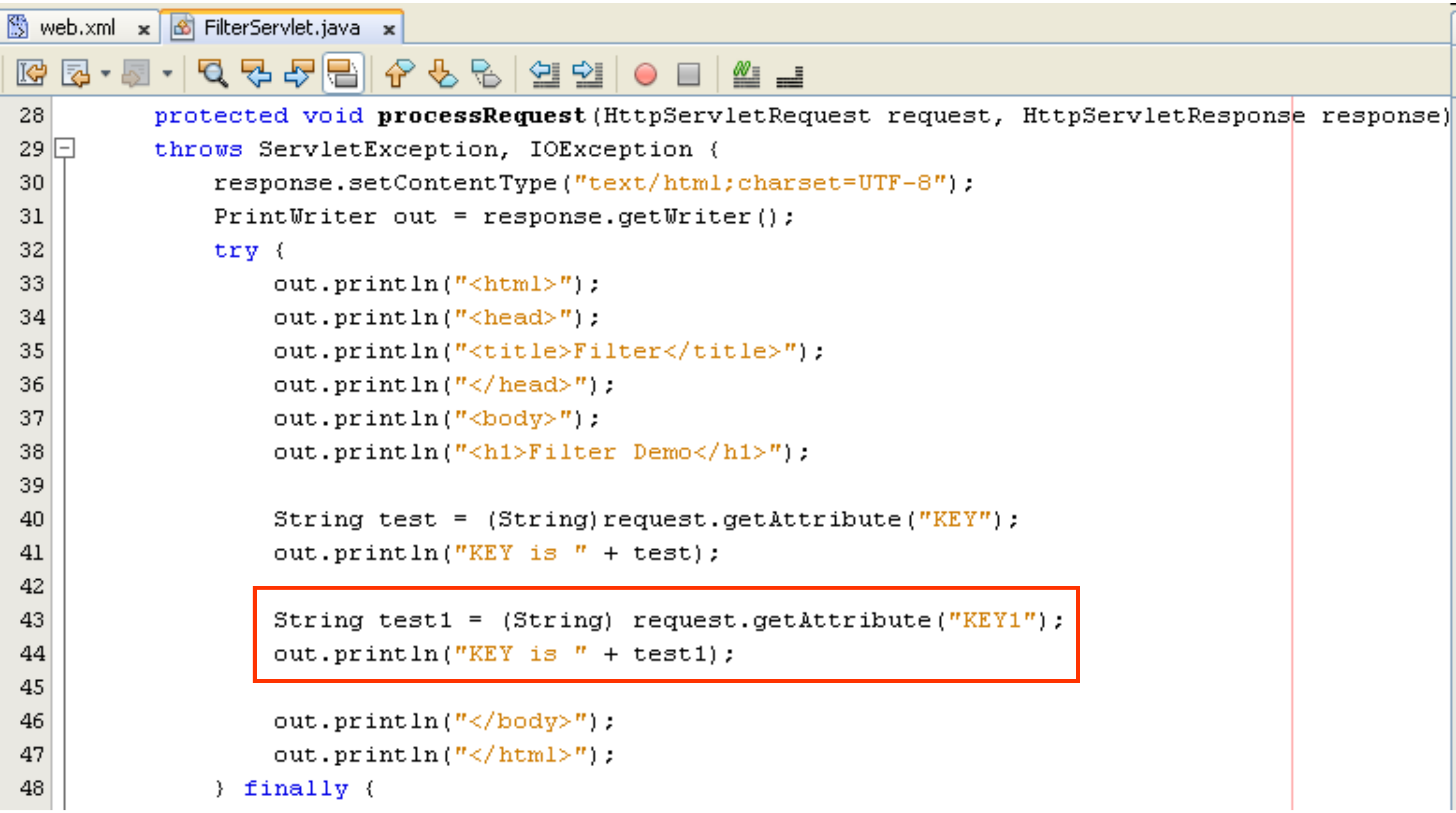
```
    <filter>
        <filter-name>FirstFilter</filter-name>
        <filter-class>sample.servlet.FirstFilter</filter-class>
    </filter>
    <filter>
        <filter-name>SecondFilter</filter-name>
        <filter-class>sample.servlet.SecondFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>SecondFilter</filter-name>
        <url-pattern>/FilterServlet</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>FirstFilter</filter-name>
        <url-pattern>/FilterServlet</url-pattern>
    </filter-mapping>
```

Filter Chain
Example



```
<filter>
    <filter-name>SecondFilter</filter-name>
    <filter-class>sample.servlet.SecondFilter</filter-class>
</filter>
<filter>
    <filter-name>FirstFilter</filter-name>
    <filter-class>sample.servlet.FirstFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SecondFilter</filter-name>
    <url-pattern>/FilterServlet</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>FirstFilter</filter-name>
    <url-pattern>/FilterServlet</url-pattern>
</filter-mapping>
<servlet>
```

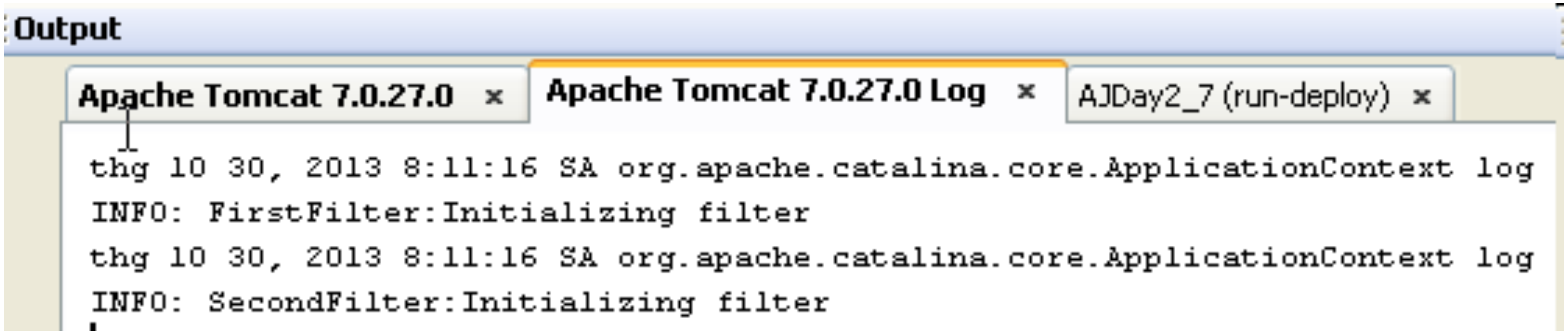
Filter Chain
Example



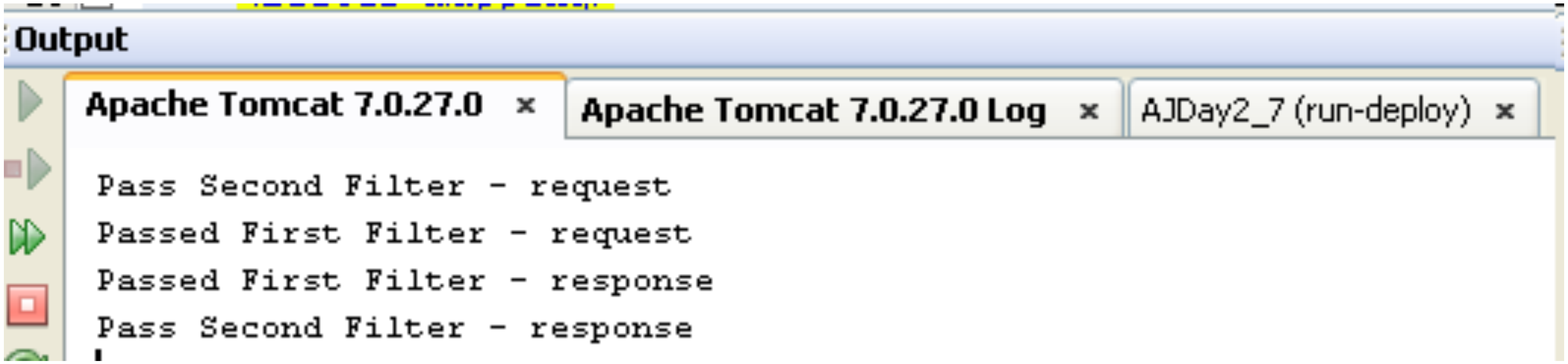
```java
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Filter</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Filter Demo</h1>");

            String test = (String)request.getAttribute("KEY");
            out.println("KEY is " + test);

            String test1 = (String) request.getAttribute("KEY1");
            out.println("KEY is " + test1);

            out.println("</body>");
            out.println("</html>");
        } finally {
```

Fpt University

Filter Chain
Example

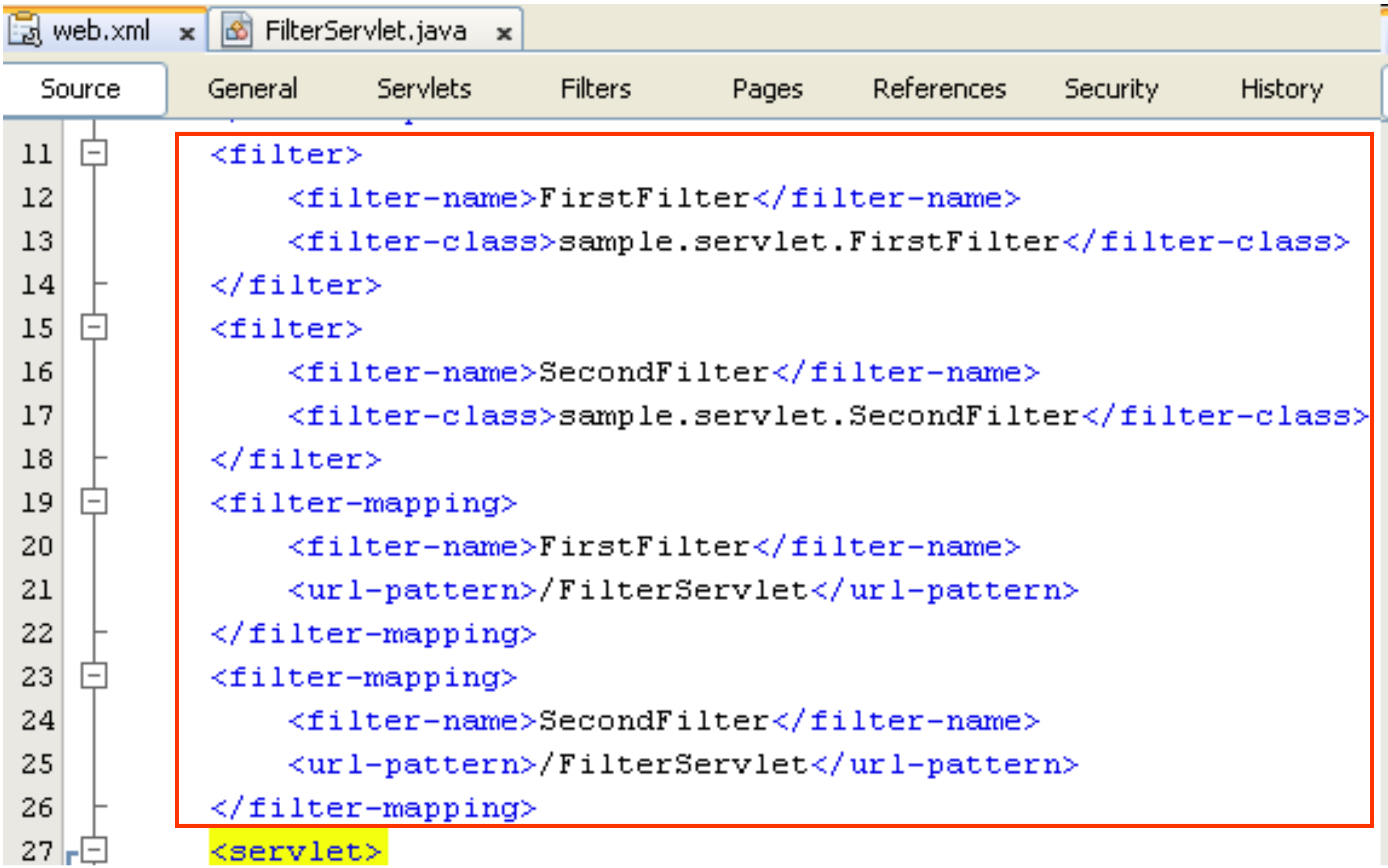


```
thg 10 30, 2013 8:11:16 SA org.apache.catalina.core.ApplicationContext log
INFO: FirstFilter:Initializing filter
thg 10 30, 2013 8:11:16 SA org.apache.catalina.core.ApplicationContext log
INFO: SecondFilter:Initializing filter
```



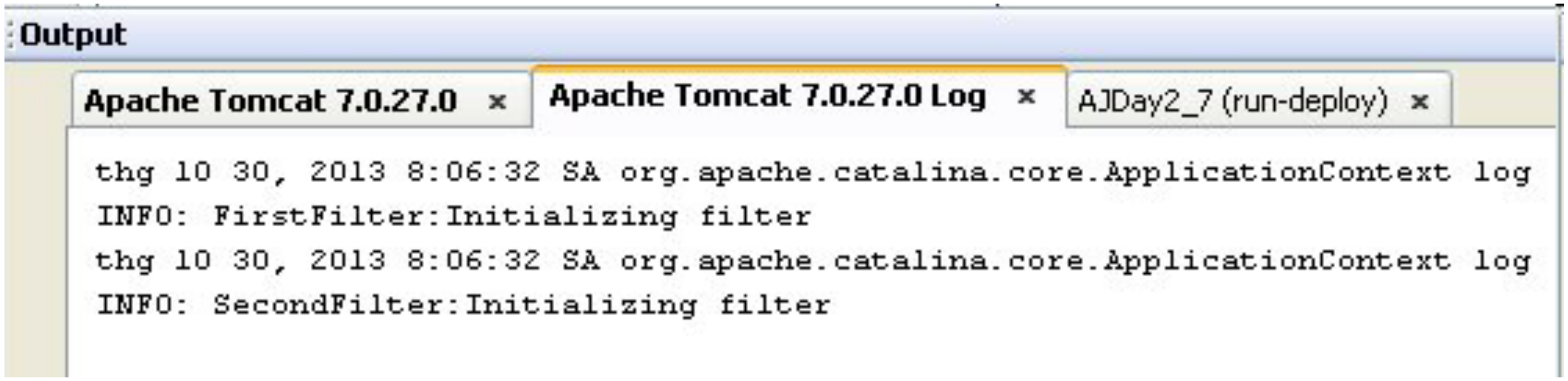
```
Pass Second Filter - request
Passed First Filter - request
Passed First Filter - response
Pass Second Filter - response
```

Filter Chain
Example – **Change position**

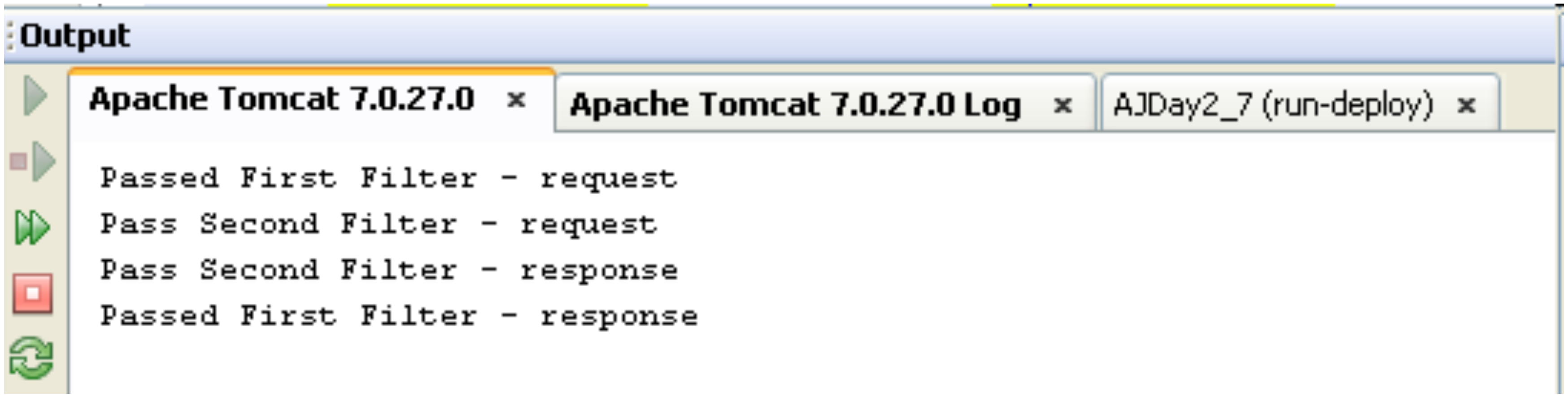


```
<filter>
    <filter-name>FirstFilter</filter-name>
    <filter-class>sample.servlet.FirstFilter</filter-class>
</filter>
<filter>
    <filter-name>SecondFilter</filter-name>
    <filter-class>sample.servlet.SecondFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>FirstFilter</filter-name>
    <url-pattern>/FilterServlet</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>SecondFilter</filter-name>
    <url-pattern>/FilterServlet</url-pattern>
</filter-mapping>
<servlet>
```

Filter Chain
Example



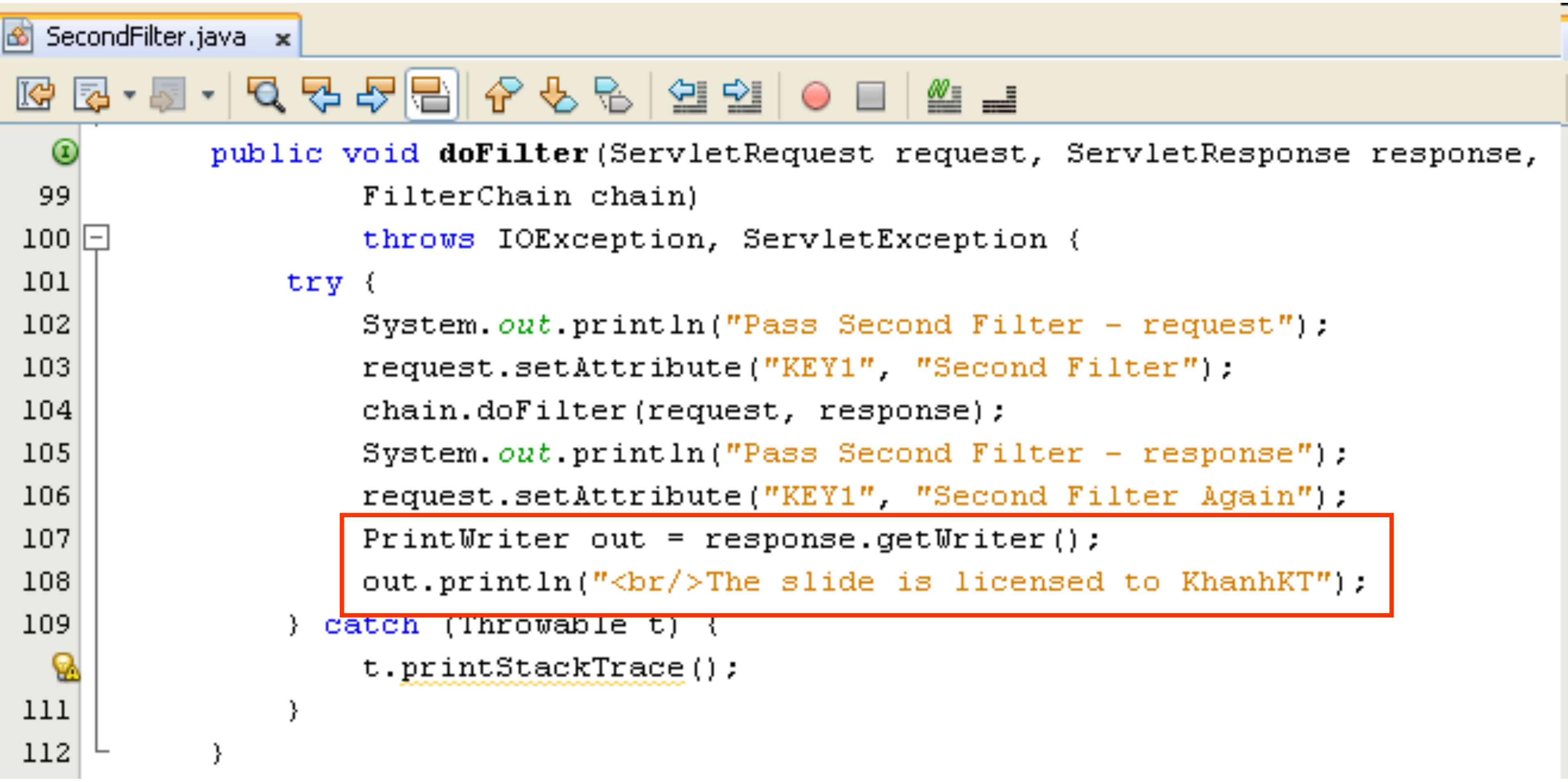
```
thg 10 30, 2013 8:06:32 SA org.apache.catalina.core.ApplicationContext log
INFO: FirstFilter:Initializing filter
thg 10 30, 2013 8:06:32 SA org.apache.catalina.core.ApplicationContext log
INFO: SecondFilter:Initializing filter
```



```
Passed First Filter - request
Pass Second Filter - request
Pass Second Filter - response
Passed First Filter - response
```

Filter Chain

Why need a Wrapper Class

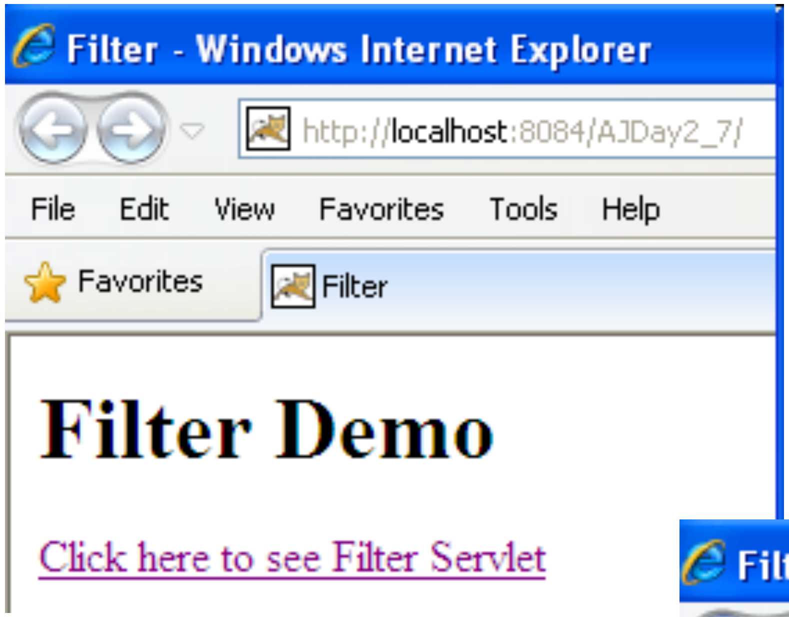


SecondFilter.java

```
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        try {
            System.out.println("Pass Second Filter - request");
            request.setAttribute("KEY1", "Second Filter");
            chain.doFilter(request, response);
            System.out.println("Pass Second Filter - response");
            request.setAttribute("KEY1", "Second Filter Again");
            PrintWriter out = response.getWriter();
            out.println("<br/>The slide is licensed to KhanhKT");
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
```

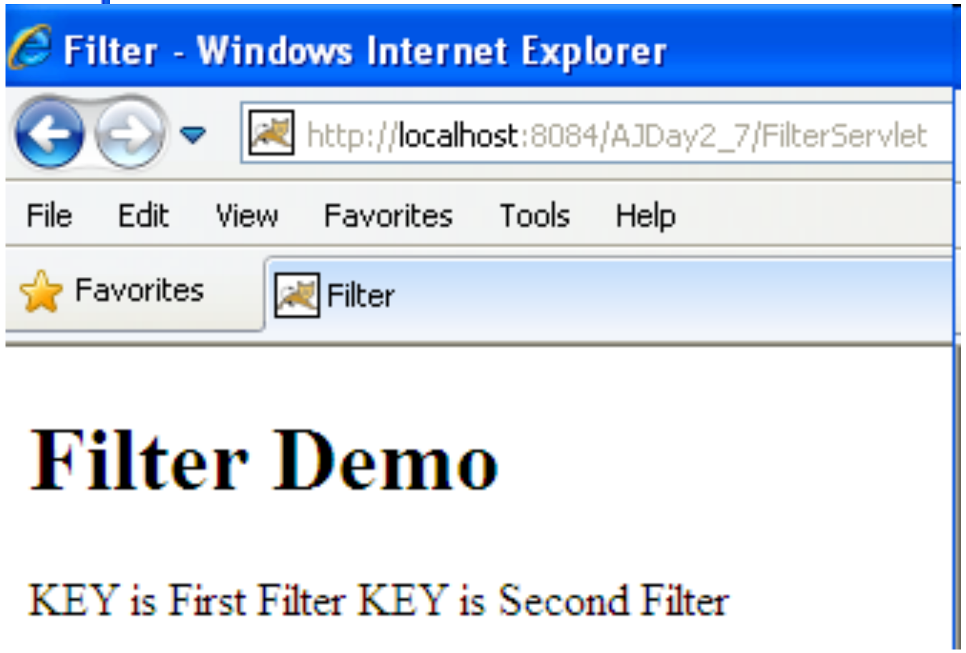
Filter Chain

Why need a Wrapper Class


Filter - Windows Internet Explorer
http://localhost:8084/AJDay2_7/
File Edit View Favorites Tools Help
Favorites
Filter

Filter Demo

Click here to see Filter Servlet

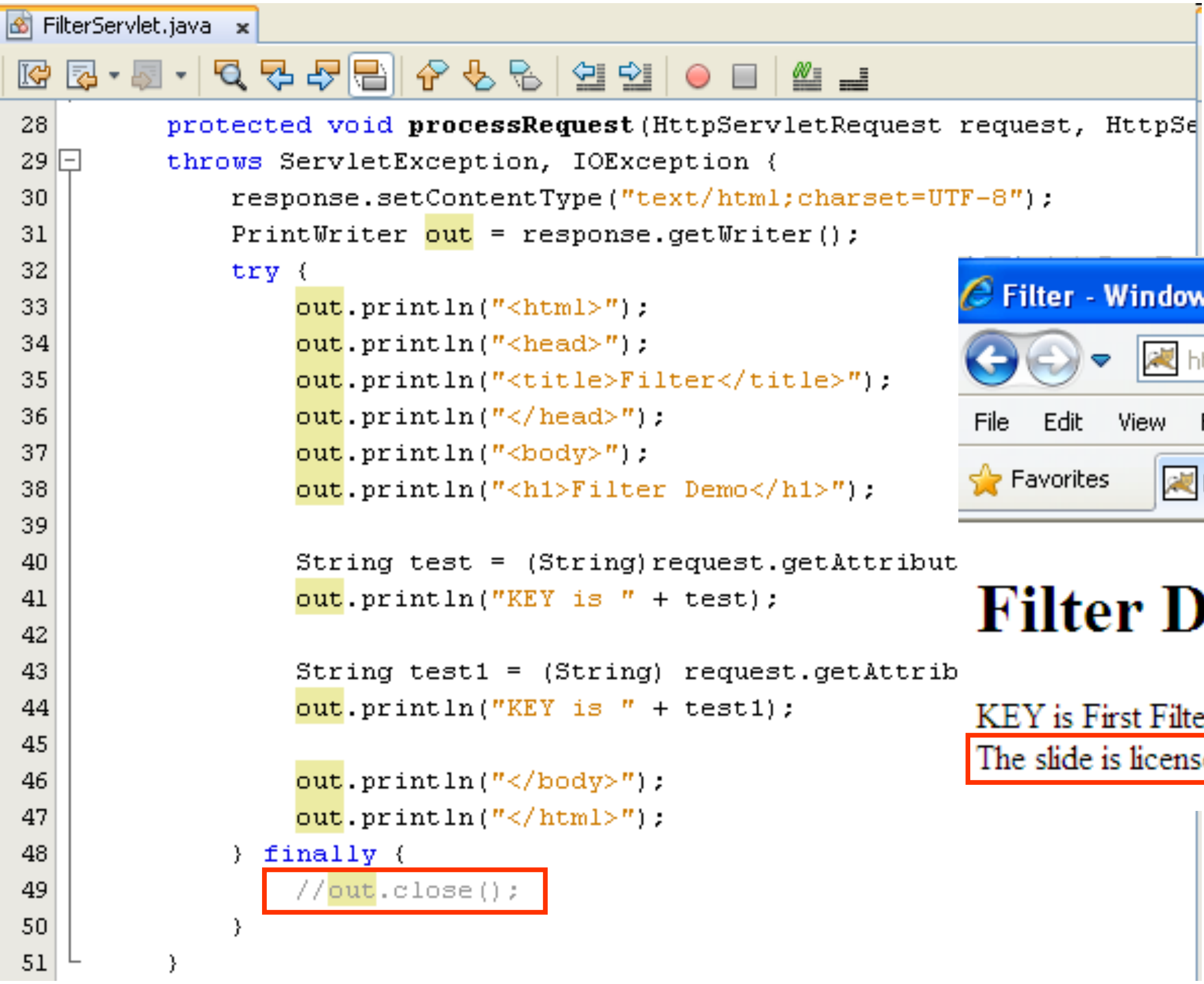

Filter - Windows Internet Explorer
http://localhost:8084/AJDay2_7/FilterServlet
File Edit View Favorites Tools Help
Favorites
Filter

Filter Demo

KEY is First Filter KEY is Second Filter

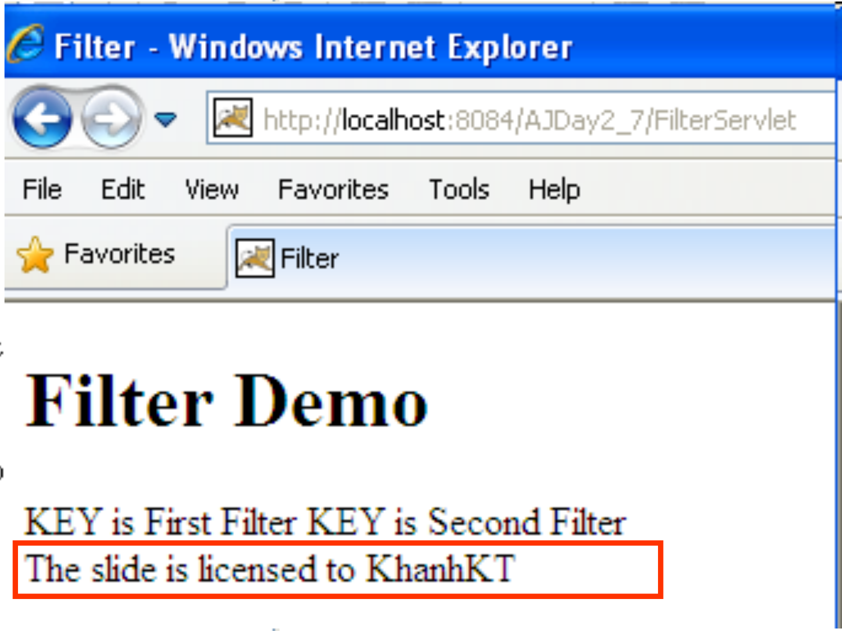
Filter Chain

Why need a Wrapper Class



FilterServlet.java

```
    protected void processRequest(HttpServletRequest request, HttpSe
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Filter</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Filter Demo</h1>");

            String test = (String)request.getAttribut
            out.println("KEY is " + test);

            String test1 = (String) request.getAttrib
            out.println("KEY is " + test1);

            out.println("</body>");
            out.println("</html>");
        } finally {
            //out.close();
        }
    }
```



Filter - Windows Internet Explorer

http://localhost:8084/AJDay2_7/FilterServlet

Filter Demo

KEY is First Filter KEY is Second Filter
The slide is licensed to KhanhKT

Filter Chain
Wrapper Class

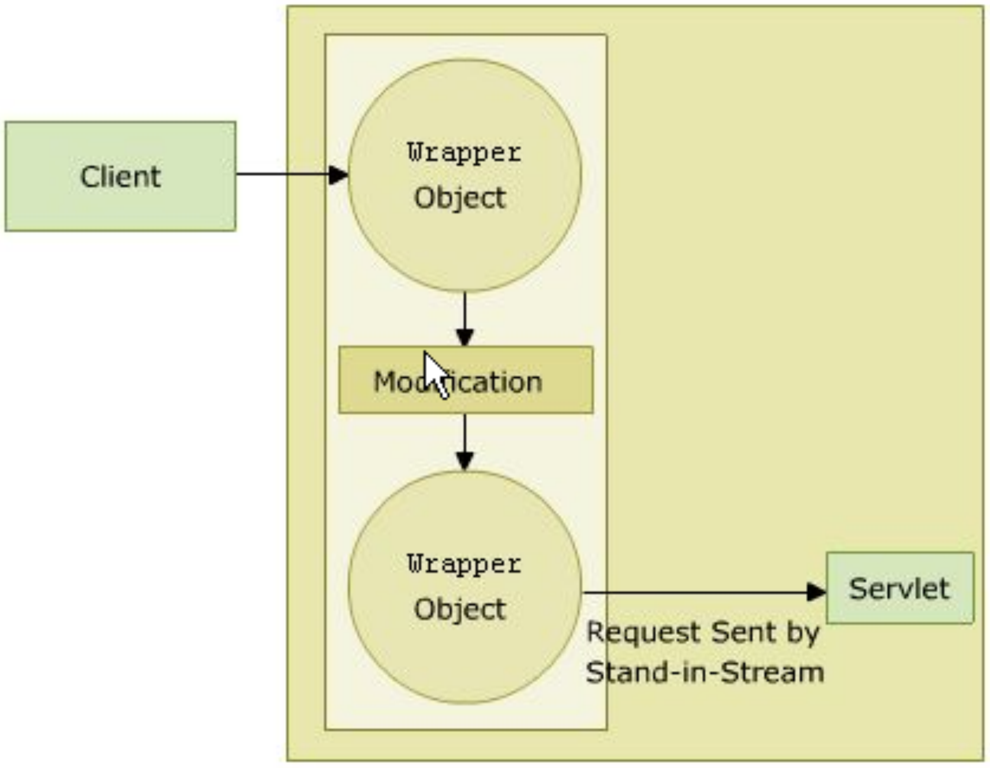
- To **modify or intercept** the request or response **before** they **can reach their logical destination**, the **required object can dynamically capture** the request or response
- Wrapper class
  - Creates the object to **capture the request and response** before they reach server and client respectively
  - The wrapper object generated by the filter **implements the getWriter() and getOutputStream(),** which returns a stand-in-stream. The stand-in-stream is passed to the servlet through the wrapper object
  - The wrapper object captures **the response through the stand-in-stream** and sends it back to the filter

Classes	Descriptions
ServletRequestWrapper	- Provides a convenient implementation of the ServletRequest interface - Can be sub-classed by developers wishing to send the request to a servlet - To override request methods, one should wrap the request in an object that extends ServletRequestWrapper or HttpServletRequestWrapper
ServletResponseWrapper	- Provides a convenient implementation of the ServletResponse interface - Can be sub classed by developers wishing to send the response from a servlet.

Filter Chain

Wrapper Class – Altering Request

- Create filter class extends to the **ServletRequestWrapper** or **HttpServletRequestWrapper** class.
- The object captures the HttpRequest object from the client and sends it to the filers
- Through the objects filter extends some services to the request.

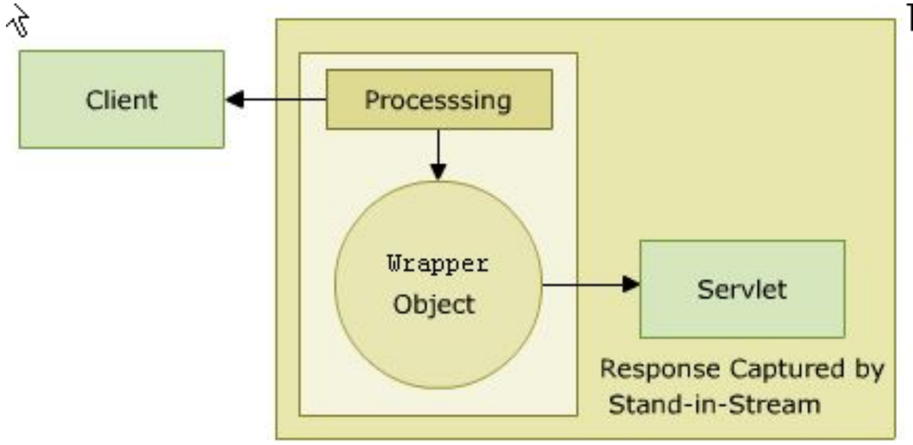


Modifying the Request

Filter Chain

Wrapper Class – Altering Response

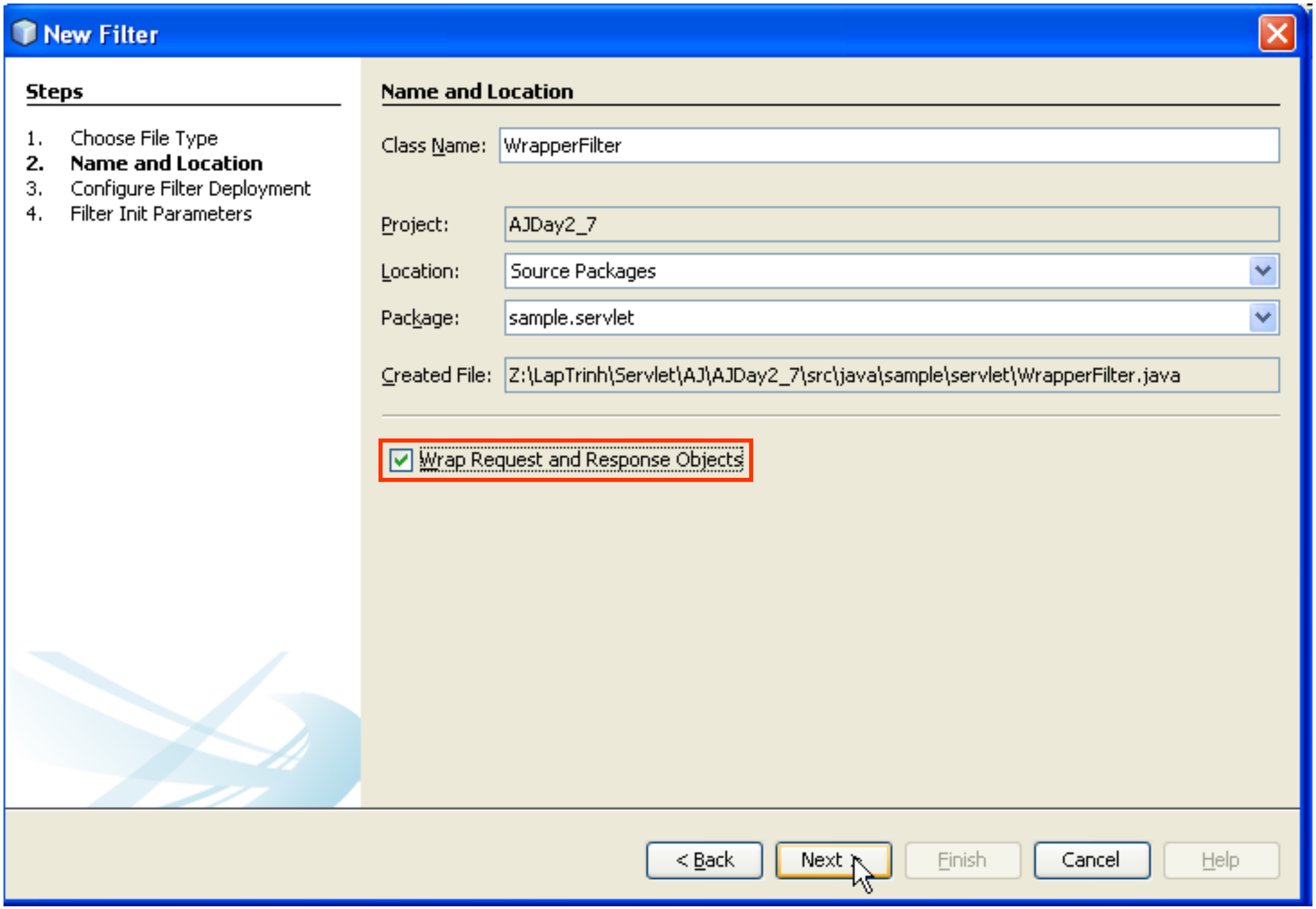
- Create filter class extends to the **ServletResponseWrapper** or **HttpServletResponseWrapper** class.
- The object **captures the httpRequest object** from the client and sends it to the filers
- Through the objects filter extends some services to the request.



Modifying the Response

Filter Chain

Wrapper Class – Example



Steps

1. Choose File Type
2. **Name and Location**
3. Configure Filter Deployment
4. Filter Init Parameters

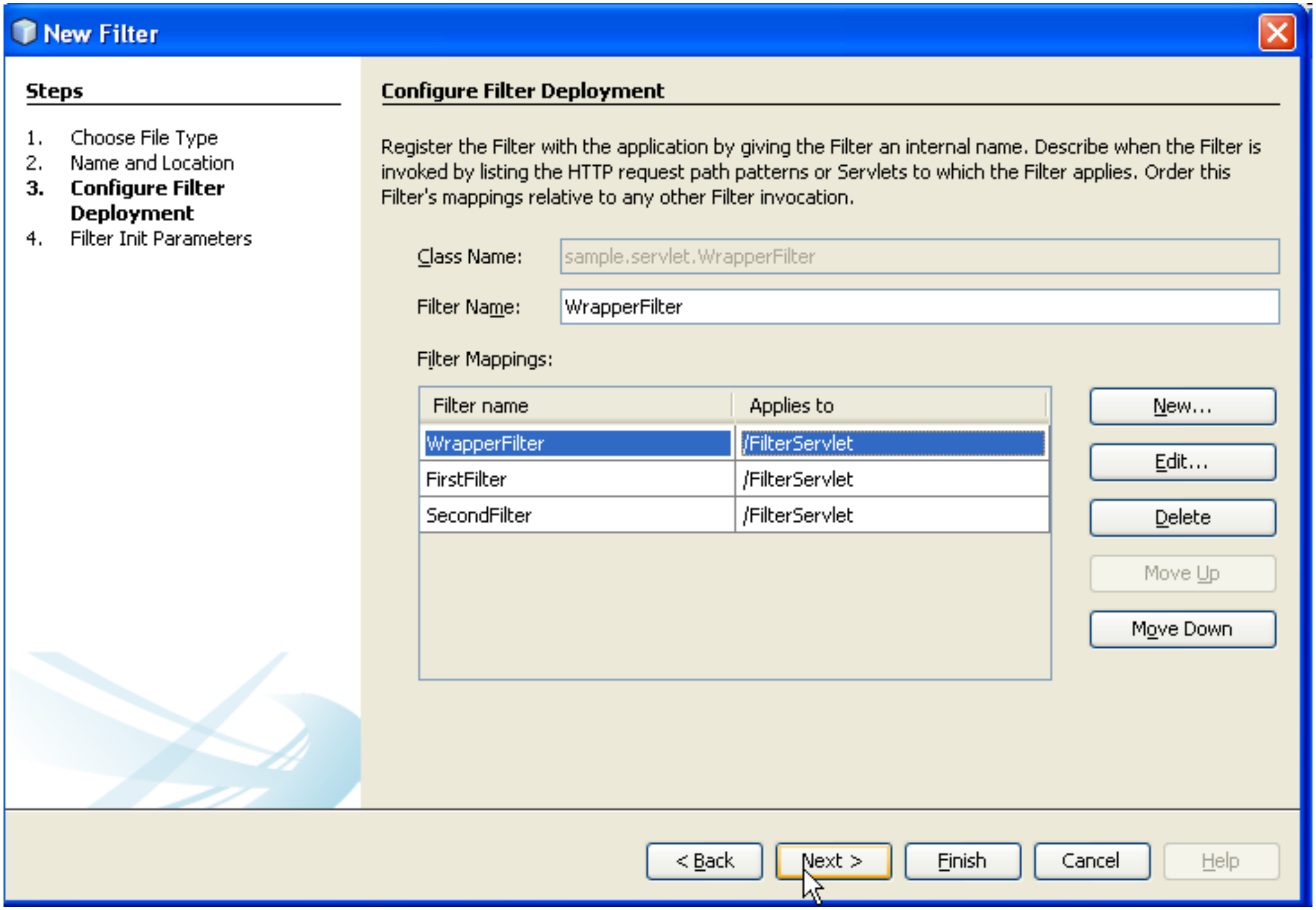
Name and Location

Class Name:	WrapperFilter
Project:	AJDay2_7
Location:	Source Packages
Package:	sample.servlet
Created File:	Z:\LapTrinh\Servlet\AJ\AJDay2_7\src\java\sample\servlet\WrapperFilter.java

[✓] Wrap Request and Response Objects

Filter Chain

Wrapper Class – Example



Steps

1. Choose File Type
2. Name and Location
3. **Configure Filter Deployment**
4. Filter Init Parameters

Configure Filter Deployment

Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.

Class Name: sample.servlet.WrapperFilter

Filter Name: WrapperFilter

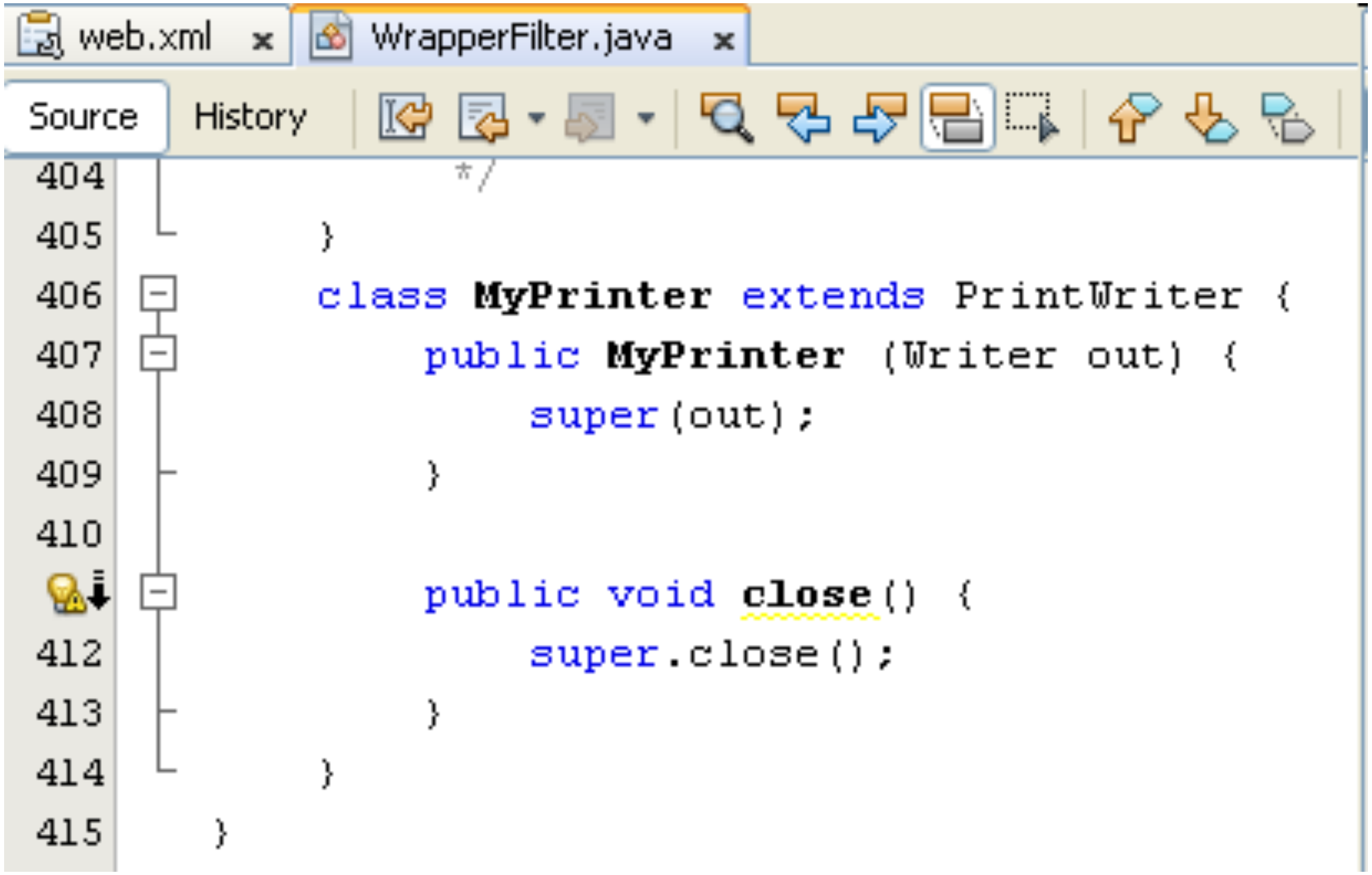
Filter Mappings:

Filter name	Applies to
WrapperFilter	/FilterServlet
FirstFilter	/FilterServlet
SecondFilter	/FilterServlet

Filter Chain

Wrapper Class – Example

•Adding the MyPrinter class extends PrintWriter in FilterWrapper class

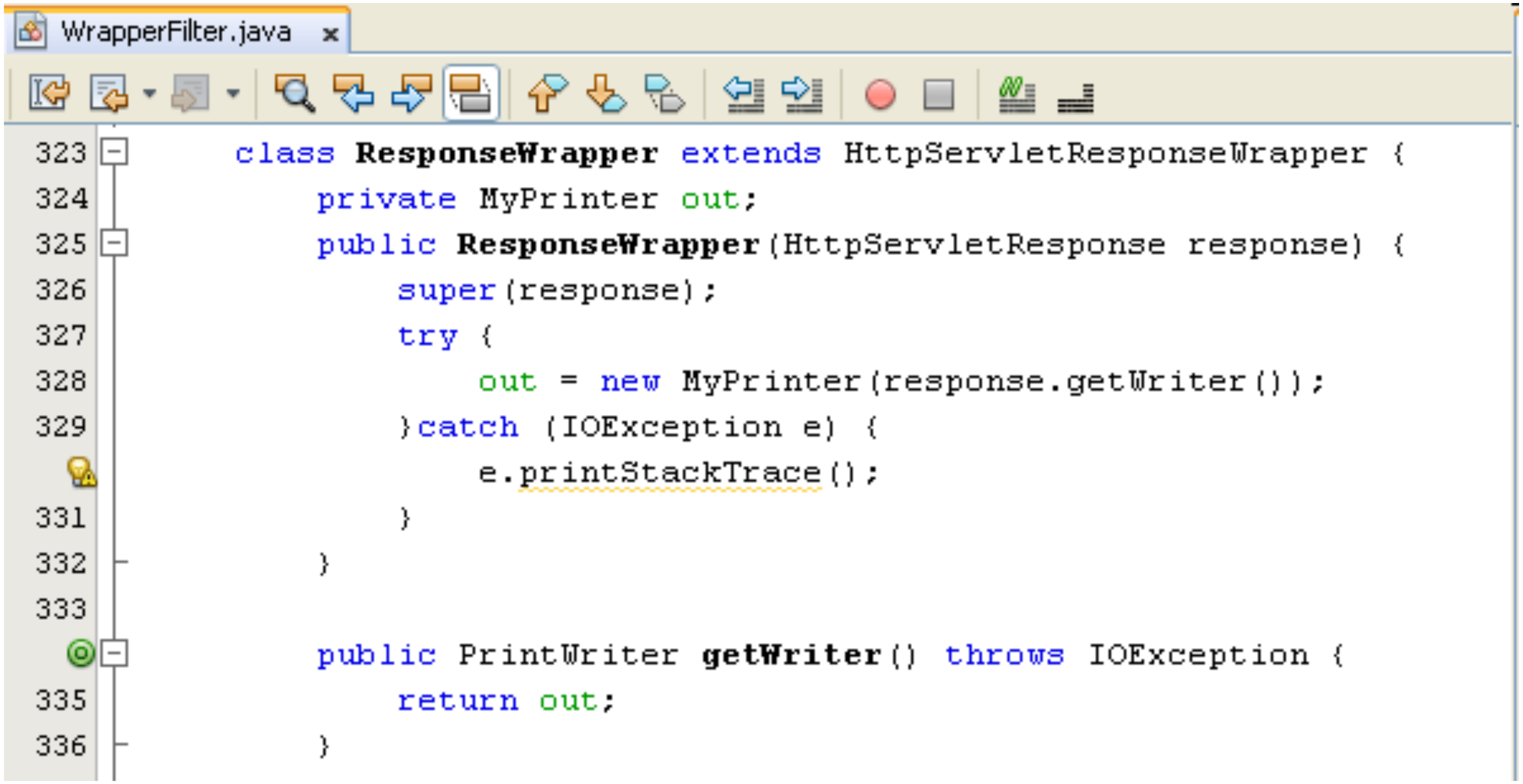


```
                    */
            }
            class MyPrinter extends PrintWriter {
                public MyPrinter (Writer out) {
                    super(out);
                }

                public void close() {
                    super.close();
                }
            }
        }
```

Filter Chain

Wrapper Class – Example

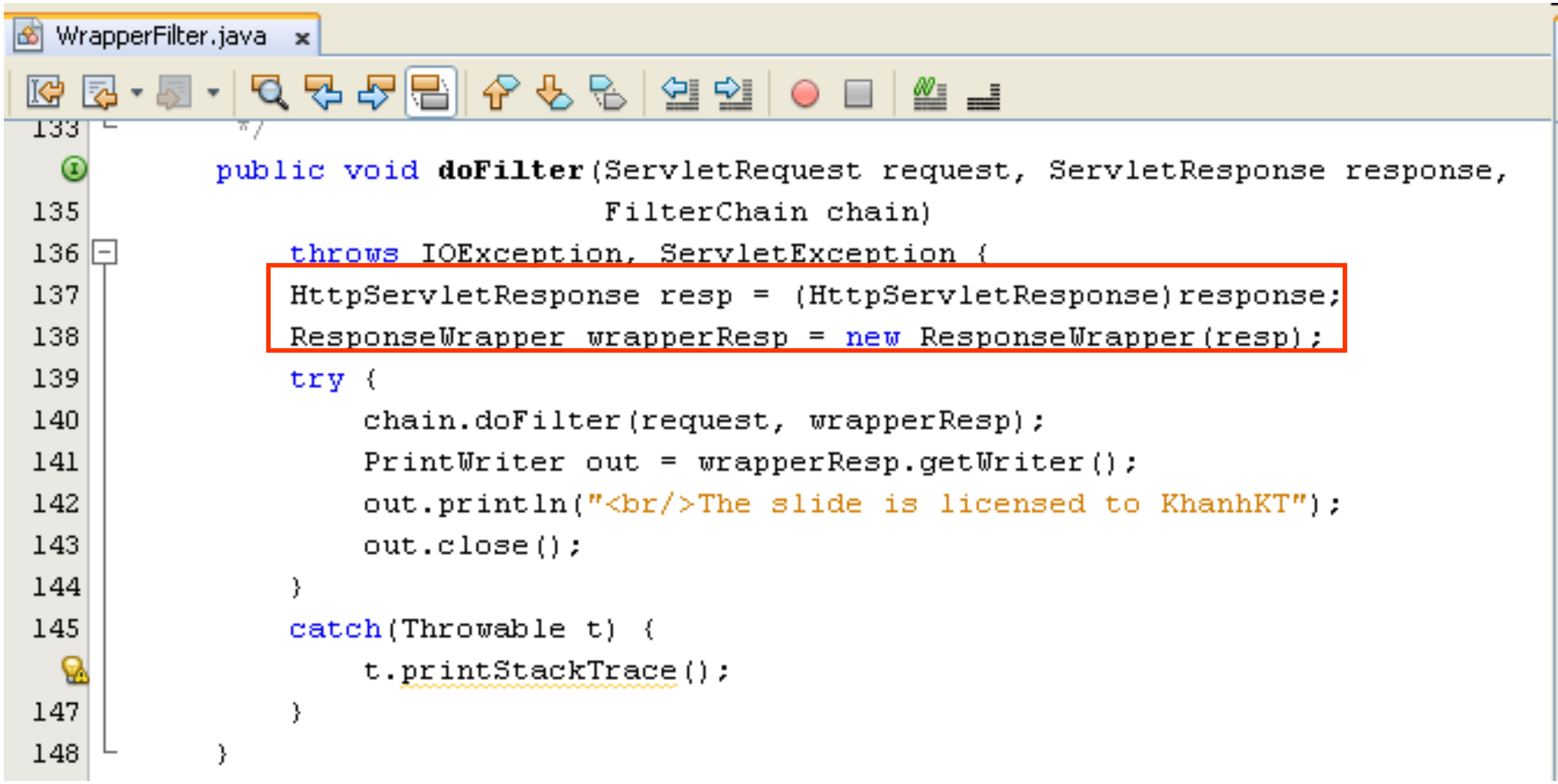
•Modifying the ResponseWrapper class uses MyPrinter to output stream



```
    class ResponseWrapper extends HttpServletResponseWrapper {
        private MyPrinter out;
        public ResponseWrapper(HttpServletResponse response) {
            super(response);
            try {
                out = new MyPrinter(response.getWriter());
            }catch (IOException e) {
                e.printStackTrace();
            }
        }

        public PrintWriter getWriter() throws IOException {
            return out;
        }
```

Filter Chain

Wrapper Class – Example



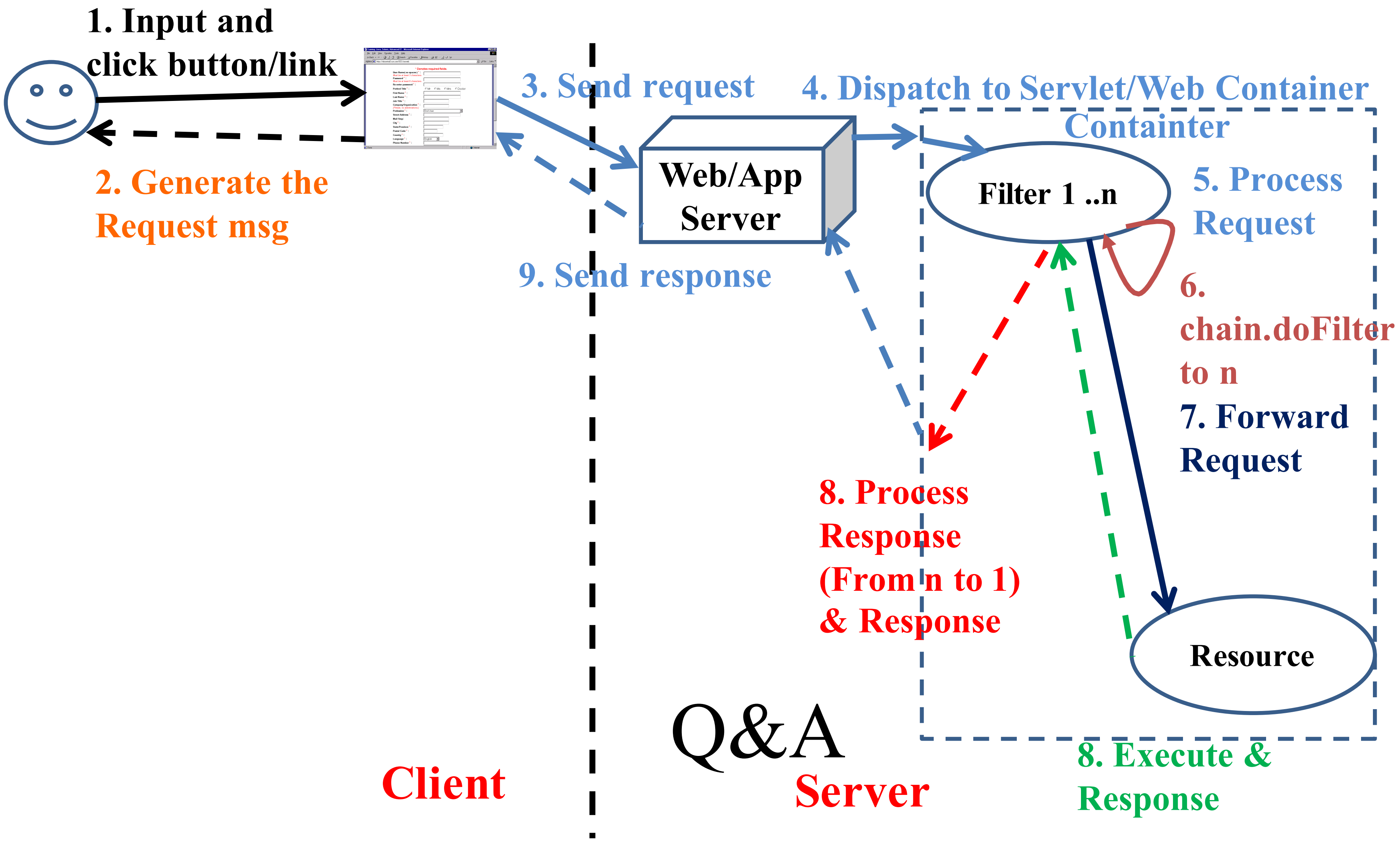
```
        */
    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain)
        throws IOException, ServletException {
        HttpServletResponse resp = (HttpServletResponse)response;
        ResponseWrapper wrapperResp = new ResponseWrapper(resp);
        try {
            chain.doFilter(request, wrapperResp);
            PrintWriter out = wrapperResp.getWriter();
            out.println("<br/>The slide is licensed to KhanhKT");
            out.close();
        }
        catch(Throwable t) {
            t.printStackTrace();
        }
    }
```

MVC2

Filter Acts as Controller

- While a **servlet** is the **most common controller**, a **filter** can **act as a controller** too
- While a **servlet** only **handles access** to the **dynamic** part of the application, a **filter** can **serve all** the **resources** in application, **including static ones**
- A filter as the controller **allows** to **block all requests** to the application, including request for static contents

Summary



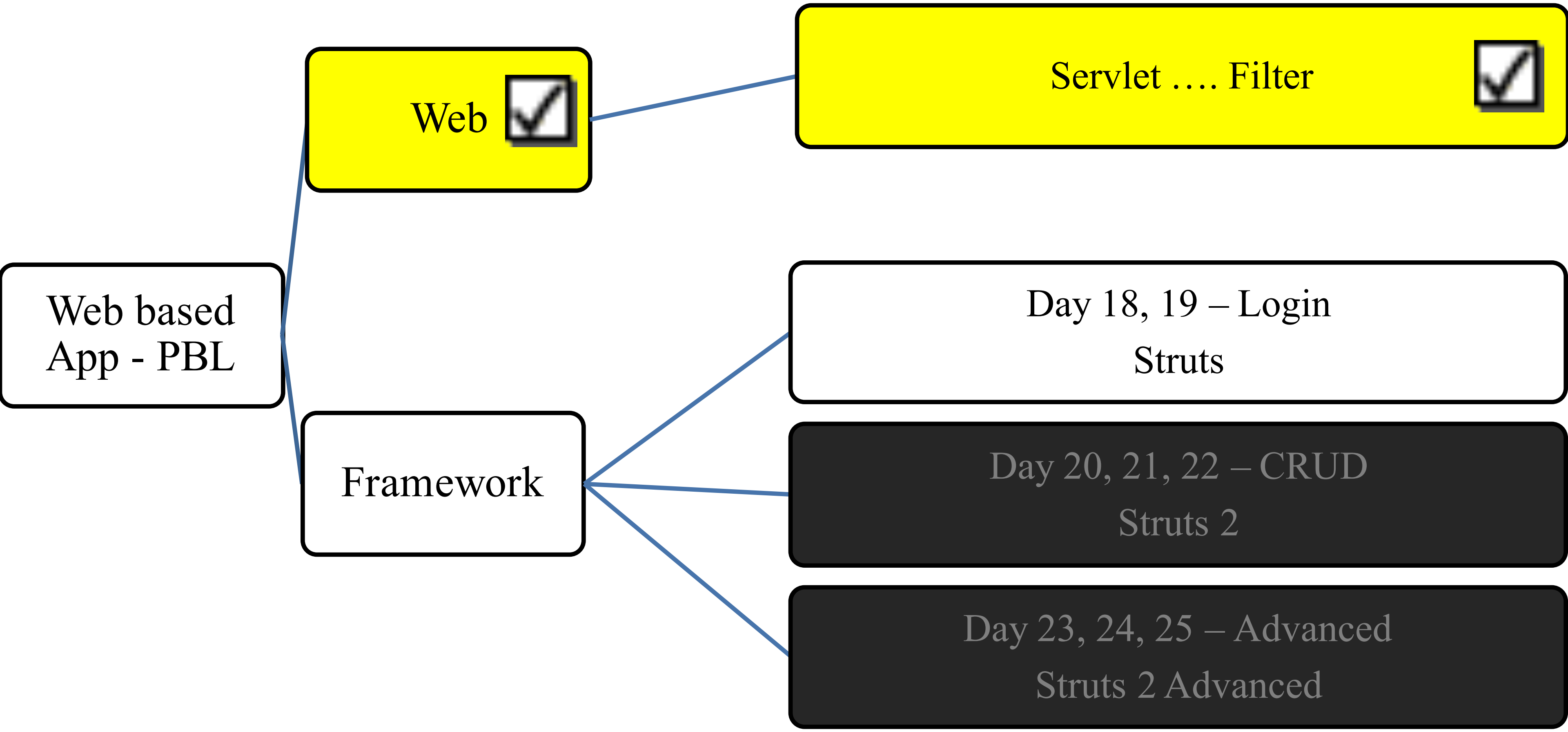
1. Input and click button/link
2. Generate the Request msg
3. Send request
4. Dispatch to Servlet/Web Container
Containter
Web/App Server
Filter 1 ..n
5. Process Request
6. chain.doFilter to n
7. Forward Request
Resource
8. Process Response (From n to 1) & Response
8. Execute & Response
9. Send response
Client
Server

Q&A

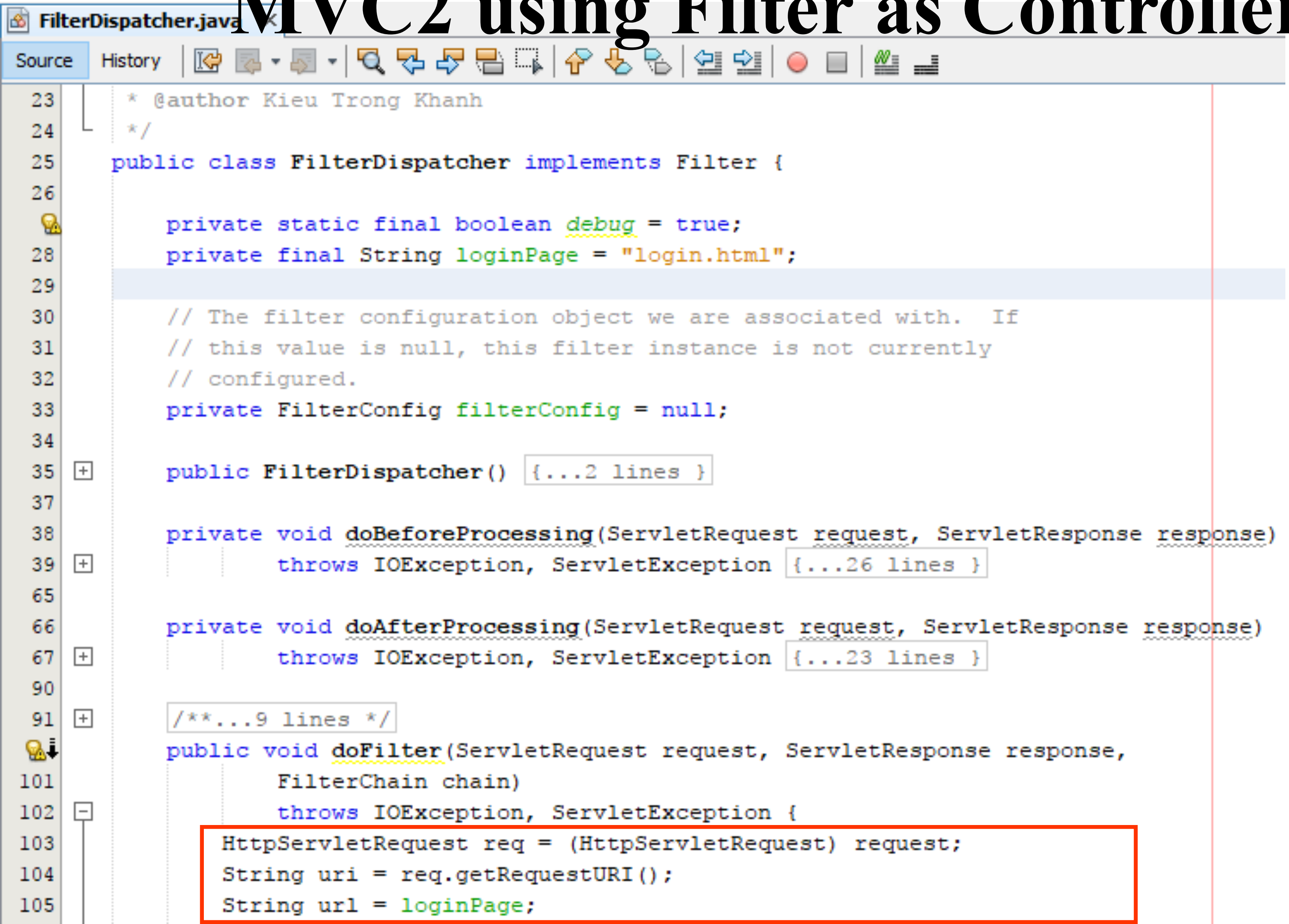
Next Lecture

- **How to build simple project MVC2 Web using Struts Framework?**
  - Struts
  - Struts Components
  - Struts Tag Lib (self-study)

Next Lecture


Web based App - PBL
Web ✓
Servlet …. Filter ✓
Framework
Day 18, 19 – Login Struts
Day 20, 21, 22 – CRUD Struts 2
Day 23, 24, 25 – Advanced Struts 2 Advanced

Appendix
MVC2 using Filter as Controller

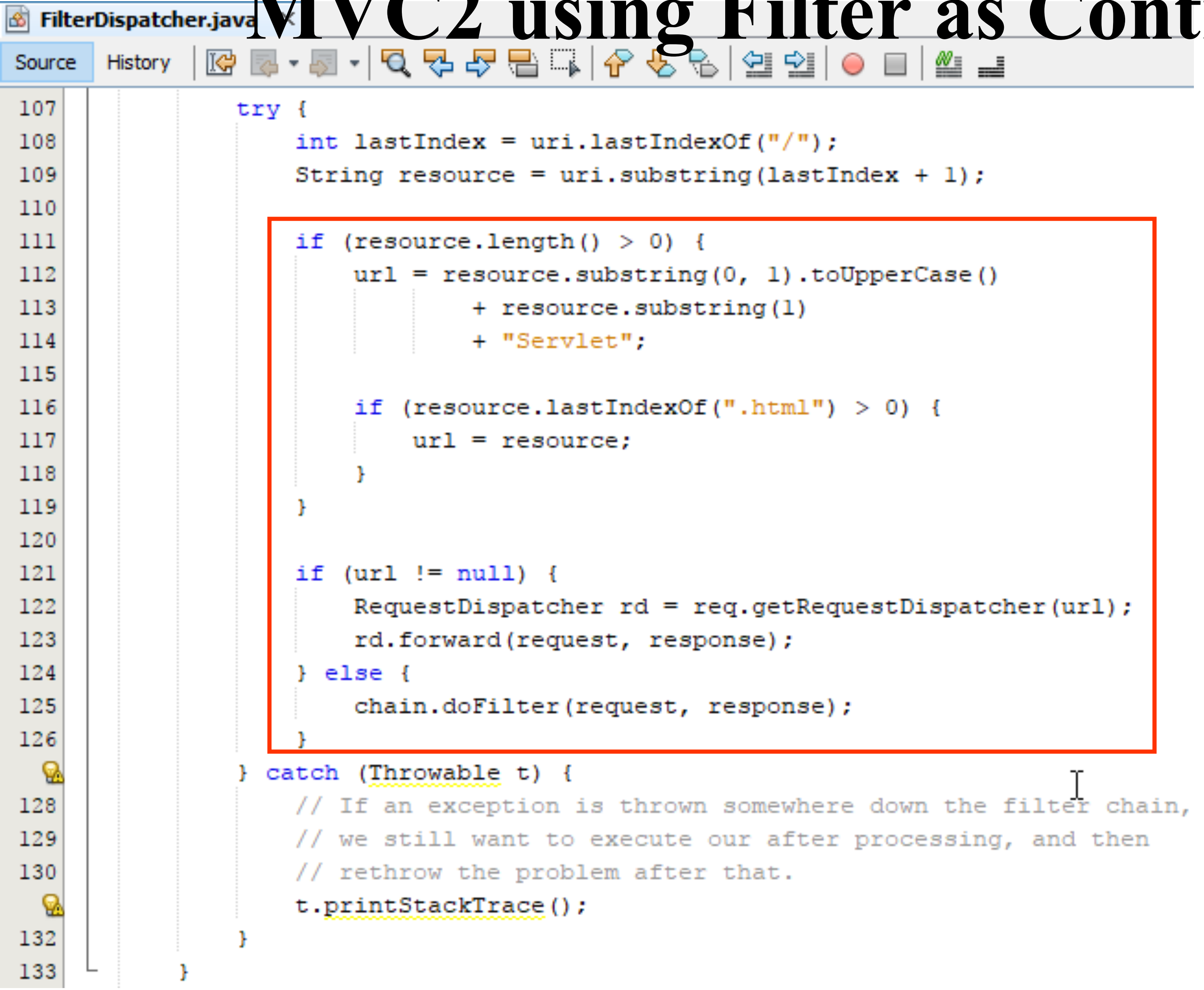


FilterDispatcher.java

Source History

```
 * @author Kieu Trong Khanh
 */
public class FilterDispatcher implements Filter {

    private static final boolean debug = true;
    private final String loginPage = "login.html";

    // The filter configuration object we are associated with.  If
    // this value is null, this filter instance is not currently
    // configured.
    private FilterConfig filterConfig = null;

    public FilterDispatcher() {...2 lines }

    private void doBeforeProcessing(ServletRequest request, ServletResponse response)
            throws IOException, ServletException {...26 lines }

    private void doAfterProcessing(ServletRequest request, ServletResponse response)
            throws IOException, ServletException {...23 lines }

    /**...9 lines */
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        String uri = req.getRequestURI();
        String url = loginPage;
```

Appendix
MVC2 using Filter as Controller

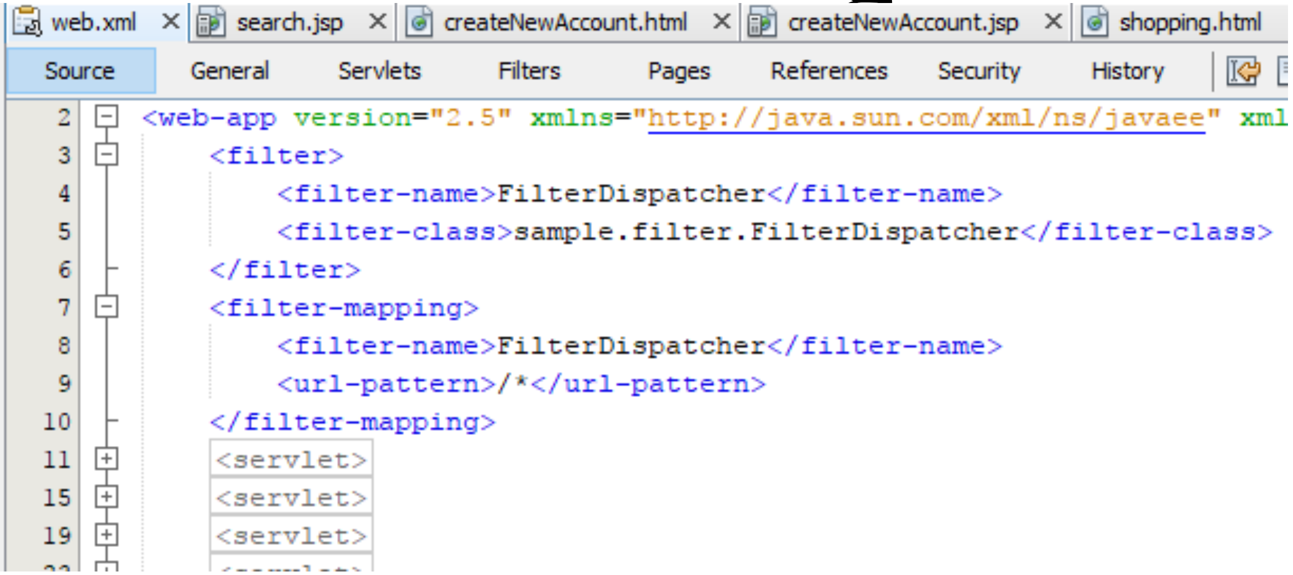


FilterDispatcher.java

Source History

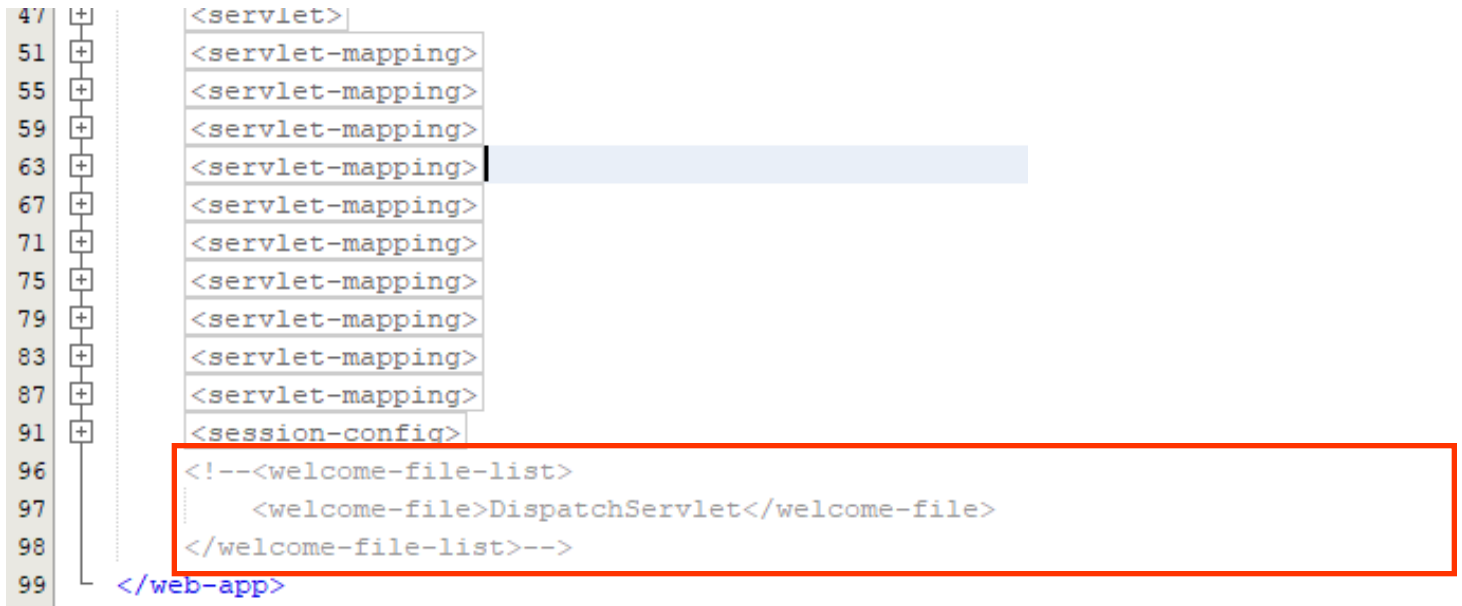
```
        try {
            int lastIndex = uri.lastIndexOf("/");
            String resource = uri.substring(lastIndex + 1);

            if (resource.length() > 0) {
                url = resource.substring(0, 1).toUpperCase()
                        + resource.substring(1)
                        + "Servlet";

                if (resource.lastIndexOf(".html") > 0) {
                    url = resource;
                }
            }

            if (url != null) {
                RequestDispatcher rd = req.getRequestDispatcher(url);
                rd.forward(request, response);
            } else {
                chain.doFilter(request, response);
            }
        } catch (Throwable t) {
            // If an exception is thrown somewhere down the filter chain,
            // we still want to execute our after processing, and then
            // rethrow the problem after that.
            t.printStackTrace();
        }
    }
```

Appendix
MVC2 using Filter as Controller


web.xml search.jsp createNewAccount.html createNewAccount.jsp shopping.html

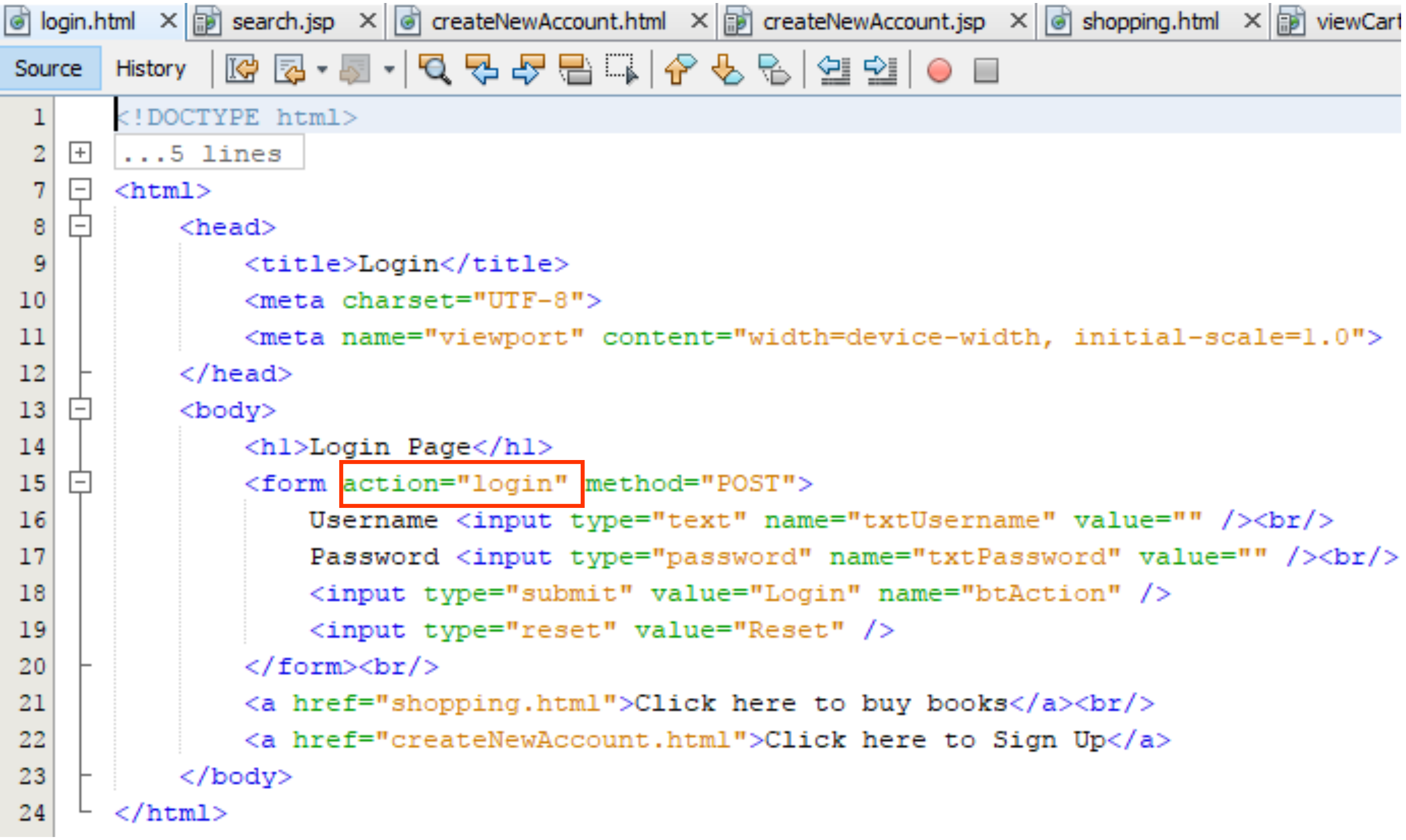
Source General Servlets Filters Pages References Security History

```
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xml
    <filter>
        <filter-name>FilterDispatcher</filter-name>
        <filter-class>sample.filter.FilterDispatcher</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>FilterDispatcher</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <servlet>
    <servlet>
    <servlet>
```



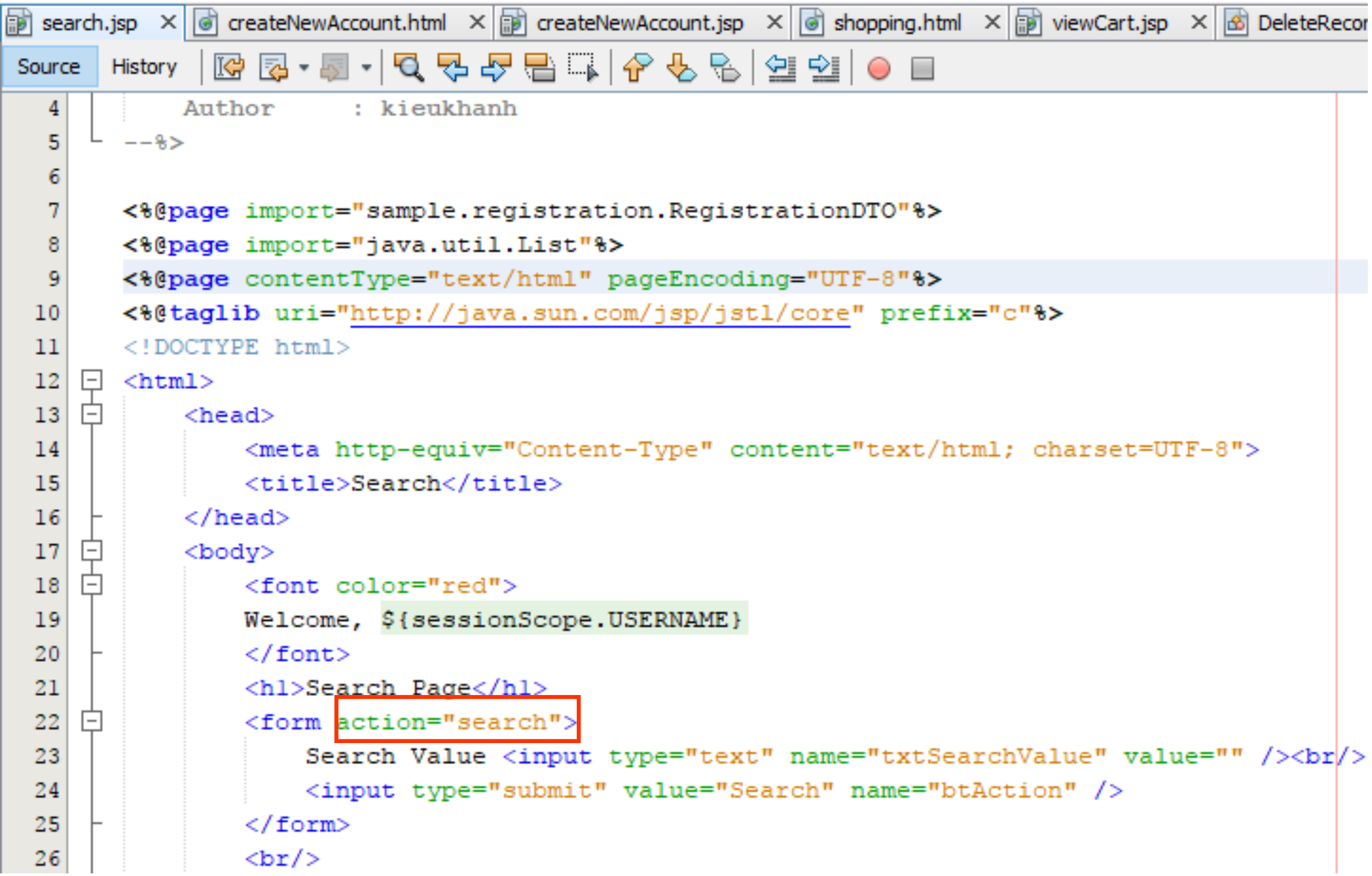
```
    <servlet>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <servlet-mapping>
    <session-config>
    <!--<welcome-file-list>
        <welcome-file>DispatchServlet</welcome-file>
    </welcome-file-list>-->
</web-app>
```

Appendix
MVC2 using Filter as Controller



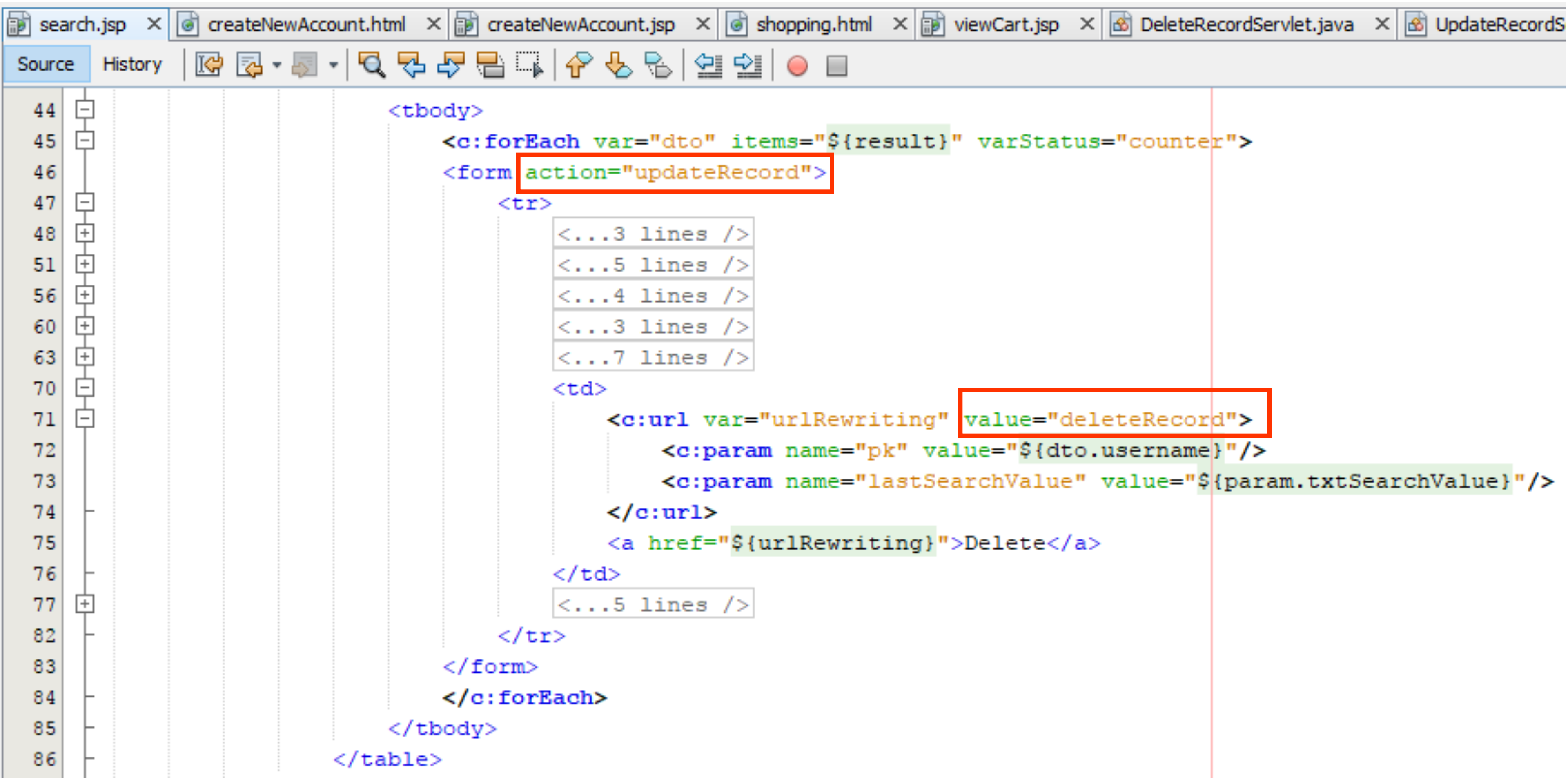
```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Login</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Login Page</h1>
        <form action="login" method="POST">
            Username <input type="text" name="txtUsername" value="" /><br/>
            Password <input type="password" name="txtPassword" value="" /><br/>
            <input type="submit" value="Login" name="btAction" />
            <input type="reset" value="Reset" />
        </form><br/>
        <a href="shopping.html">Click here to buy books</a><br/>
        <a href="createNewAccount.html">Click here to Sign Up</a>
    </body>
</html>
```

Appendix
MVC2 using Filter as Controller



```
    Author      : kieukhanh
--%>

<%@page import="sample.registration.RegistrationDTO"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Search</title>
    </head>
    <body>
        <font color="red">
        Welcome, ${sessionScope.USERNAME}
        </font>
        <h1>Search Page</h1>
        <form action="search">
            Search Value <input type="text" name="txtSearchValue" value="" /><br/>
            <input type="submit" value="Search" name="btAction" />
        </form>
        <br/>
```

Appendix
MVC2 using Filter as Controller

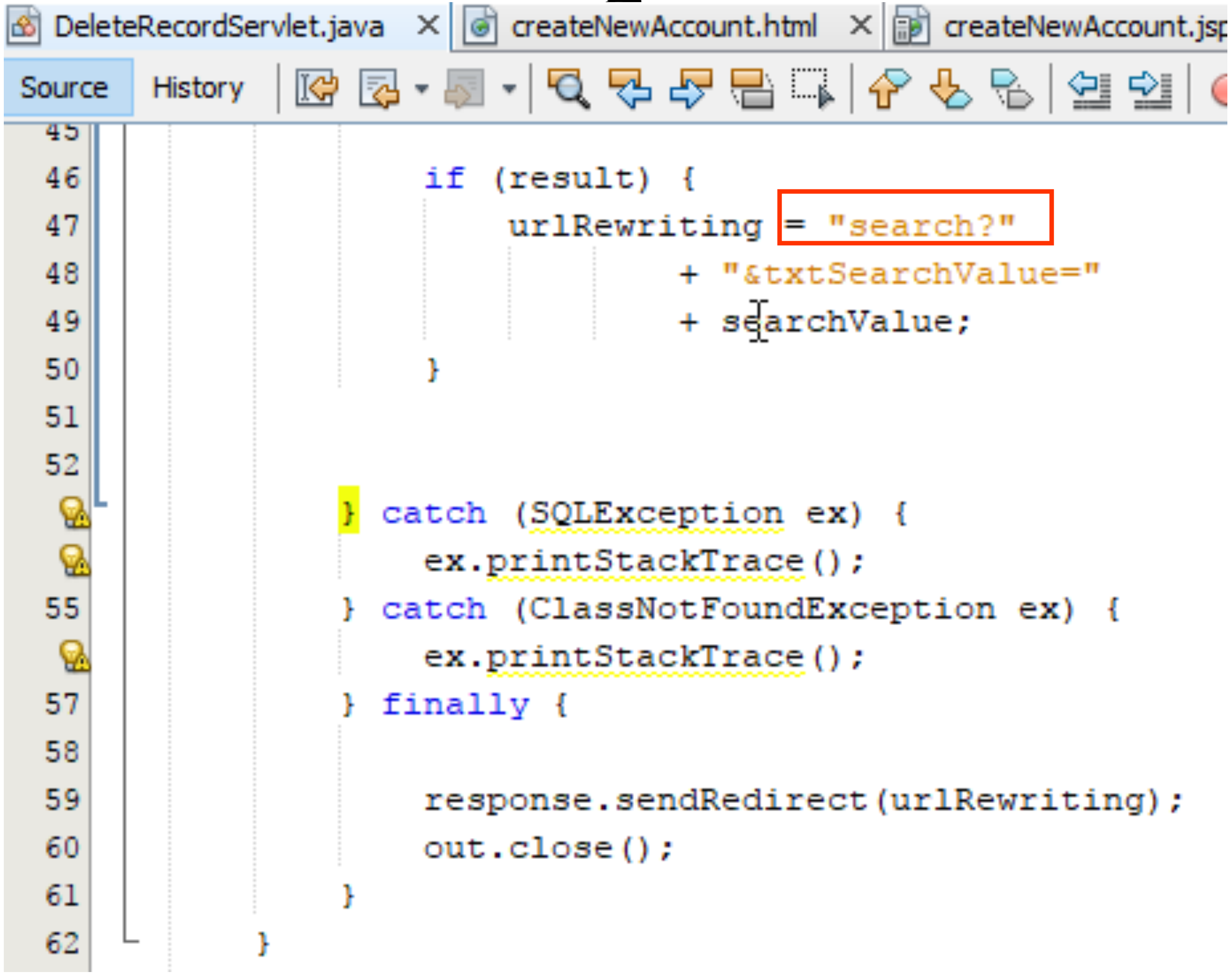


search.jsp | createNewAccount.html | createNewAccount.jsp | shopping.html | viewCart.jsp | DeleteRecordServlet.java | UpdateRecordS

Source | History

```
                <tbody>
                    <c:forEach var="dto" items="${result}" varStatus="counter">
                    <form action="updateRecord">
                        <tr>
                            <...3 lines />
                            <...5 lines />
                            <...4 lines />
                            <...3 lines />
                            <...7 lines />
                            <td>
                                <c:url var="urlRewriting" value="deleteRecord">
                                    <c:param name="pk" value="${dto.username}"/>
                                    <c:param name="lastSearchValue" value="${param.txtSearchValue}"/>
                                </c:url>
                                <a href="${urlRewriting}">Delete</a>
                            </td>
                            <...5 lines />
                        </tr>
                    </form>
                    </c:forEach>
                </tbody>
            </table>
```

Appendix
MVC2 using Filter as Controller

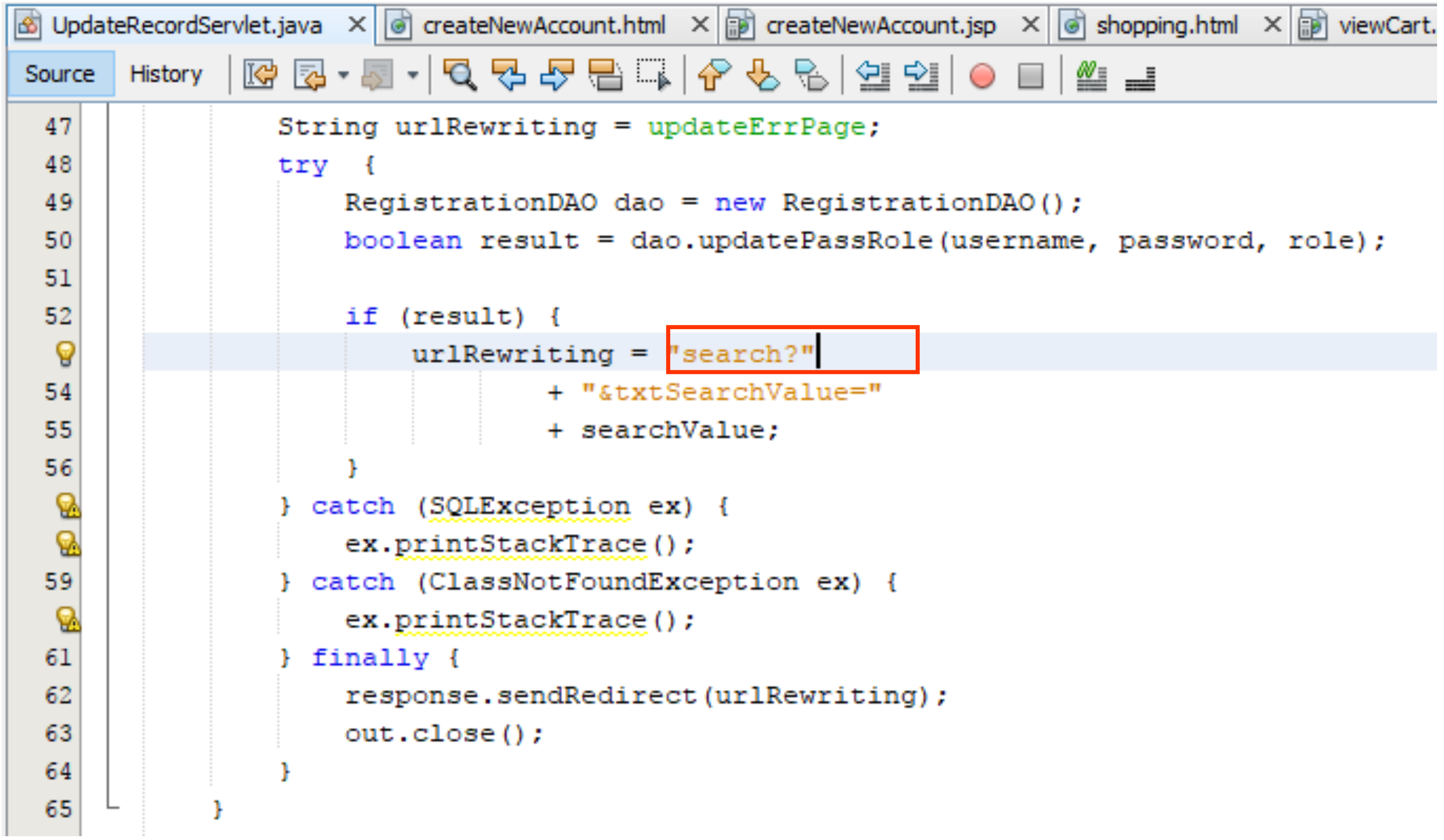


DeleteRecordServlet.java | createNewAccount.html | createNewAccount.jsp

Source | History

```
                if (result) {
                    urlRewriting = "search?"
                            + "&txtSearchValue="
                            + searchValue;
                }


            } catch (SQLException ex) {
                ex.printStackTrace();
            } catch (ClassNotFoundException ex) {
                ex.printStackTrace();
            } finally {

                response.sendRedirect(urlRewriting);
                out.close();
            }
        }
```

Appendix
MVC2 using Filter as Controller

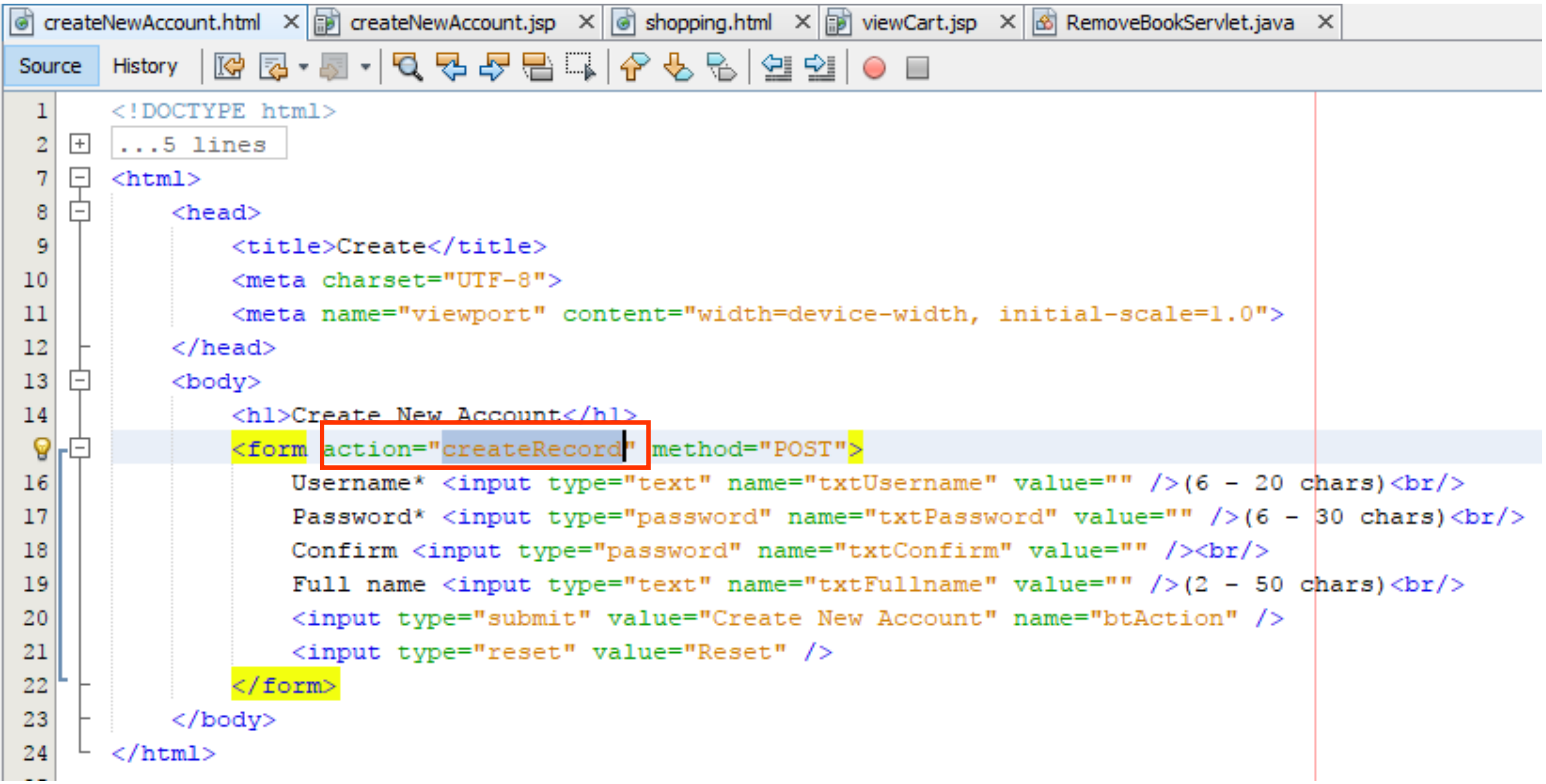


UpdateRecordServlet.java × createNewAccount.html × createNewAccount.jsp × shopping.html × viewCart.

Source History

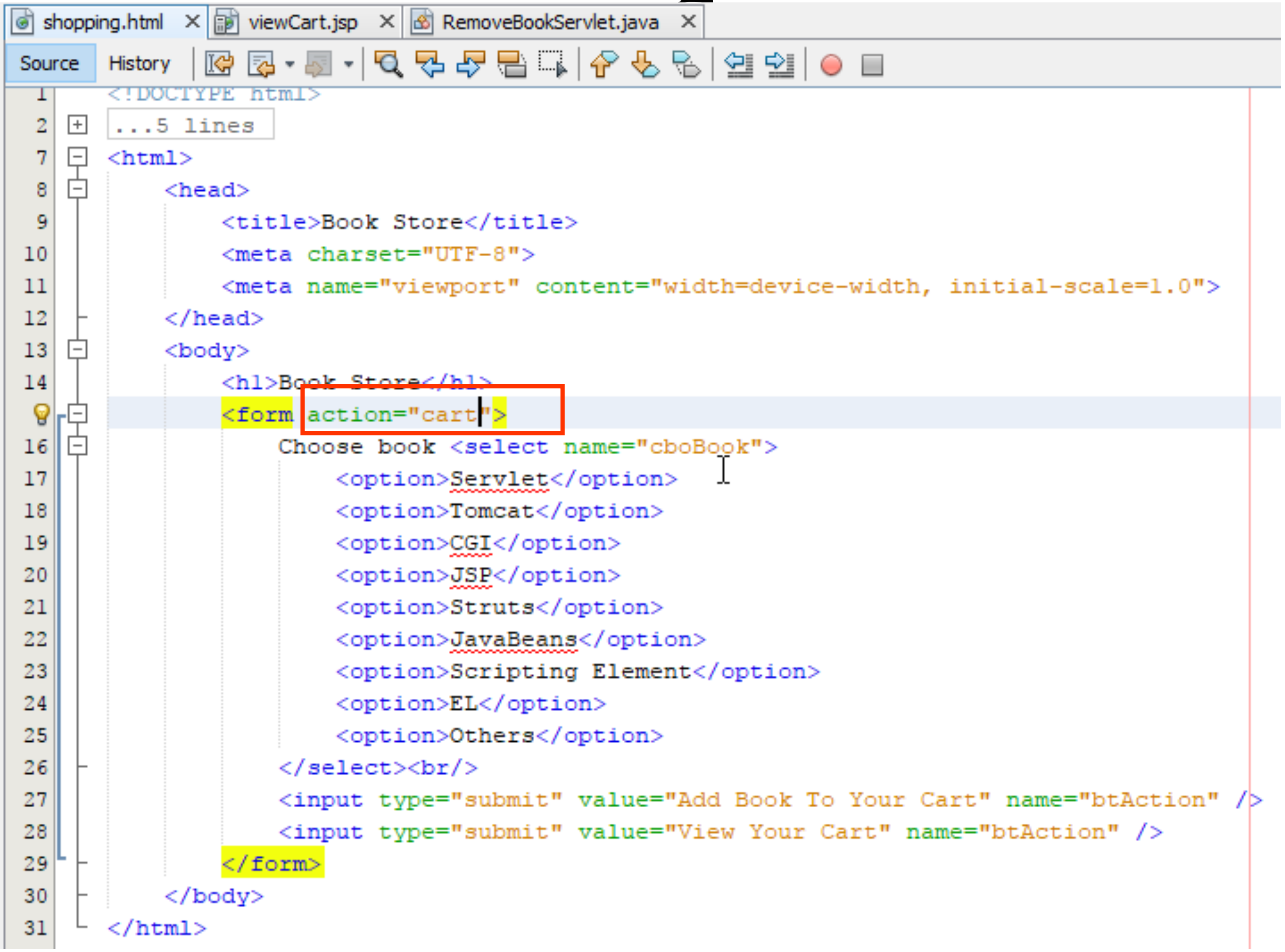
```
            String urlRewriting = updateErrPage;
            try  {
                RegistrationDAO dao = new RegistrationDAO();
                boolean result = dao.updatePassRole(username, password, role);

                if (result) {
                    urlRewriting = "search?"
                            + "&txtSearchValue="
                            + searchValue;
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            } catch (ClassNotFoundException ex) {
                ex.printStackTrace();
            } finally {
                response.sendRedirect(urlRewriting);
                out.close();
            }
        }
```

Appendix
MVC2 using Filter as Controller



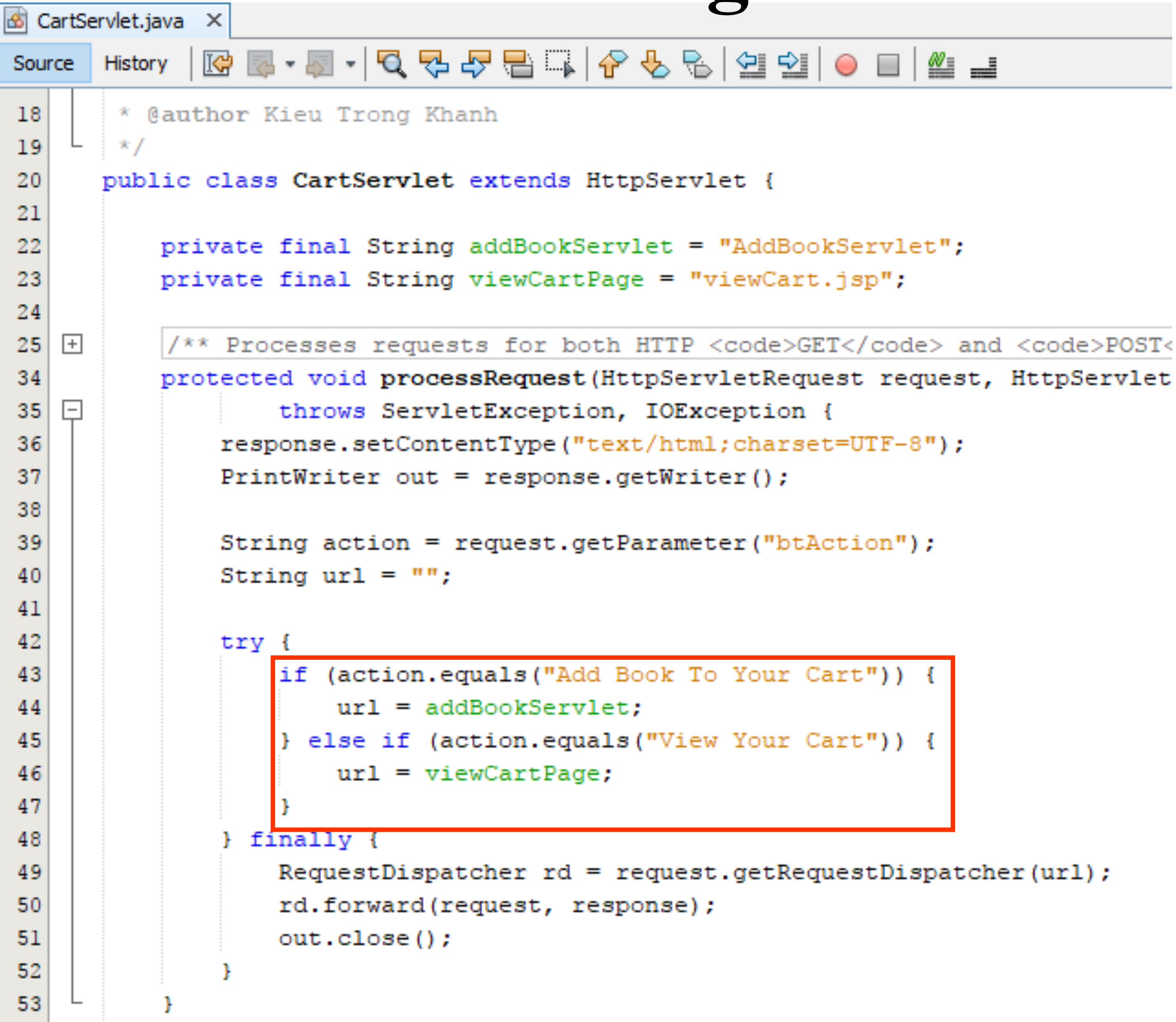
```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Create</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Create New Account</h1>
        <form action="createRecord" method="POST">
            Username* <input type="text" name="txtUsername" value="" />(6 - 20 chars)<br/>
            Password* <input type="password" name="txtPassword" value="" />(6 - 30 chars)<br/>
            Confirm <input type="password" name="txtConfirm" value="" /><br/>
            Full name <input type="text" name="txtFullname" value="" />(2 - 50 chars)<br/>
            <input type="submit" value="Create New Account" name="btAction" />
            <input type="reset" value="Reset" />
        </form>
    </body>
</html>
```

Appendix
MVC2 using Filter as Controller



```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Book Store</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Book Store</h1>
        <form action="cart">
            Choose book <select name="cboBook">
                <option>Servlet</option>
                <option>Tomcat</option>
                <option>CGI</option>
                <option>JSP</option>
                <option>Struts</option>
                <option>JavaBeans</option>
                <option>Scripting Element</option>
                <option>EL</option>
                <option>Others</option>
            </select><br/>
            <input type="submit" value="Add Book To Your Cart" name="btAction" />
            <input type="submit" value="View Your Cart" name="btAction" />
        </form>
    </body>
</html>
```

Appendix
MVC2 using Filter as Controller

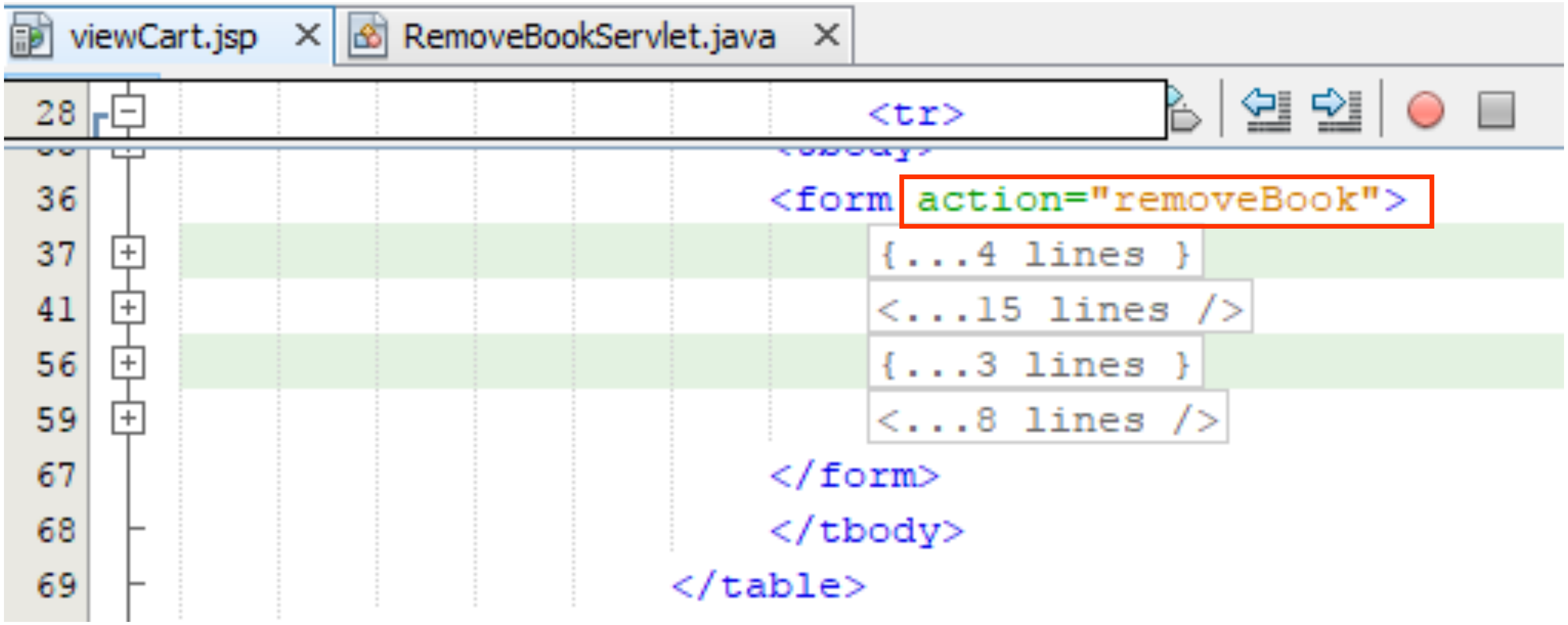


CartServlet.java

Source History

```
     * @author Kieu Trong Khanh
     */
public class CartServlet extends HttpServlet {

    private final String addBookServlet = "AddBookServlet";
    private final String viewCartPage = "viewCart.jsp";

    /** Processes requests for both HTTP <code>GET</code> and <code>POST<
    protected void processRequest(HttpServletRequest request, HttpServlet
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String action = request.getParameter("btAction");
        String url = "";

        try {
            if (action.equals("Add Book To Your Cart")) {
                url = addBookServlet;
            } else if (action.equals("View Your Cart")) {
                url = viewCartPage;
            }
        } finally {
            RequestDispatcher rd = request.getRequestDispatcher(url);
            rd.forward(request, response);
            out.close();
        }
    }
```

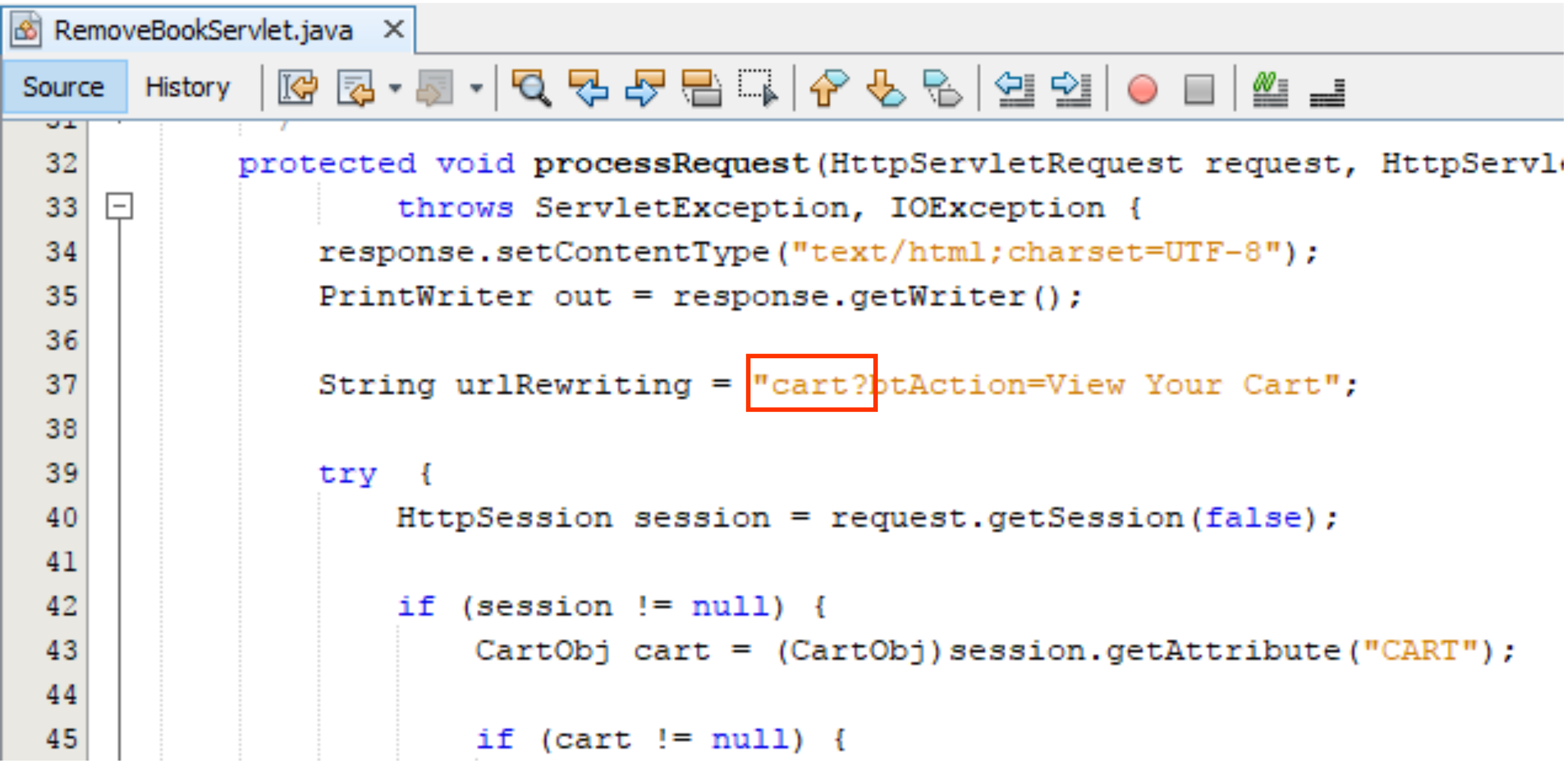
Appendix
MVC2 using Filter as Controller



viewCart.jsp RemoveBookServlet.java

```
        <tr>
    <form action="removeBook">
        {...4 lines }
        <...15 lines />
        {...3 lines }
        <...8 lines />
    </form>
    </tbody>
</table>
```

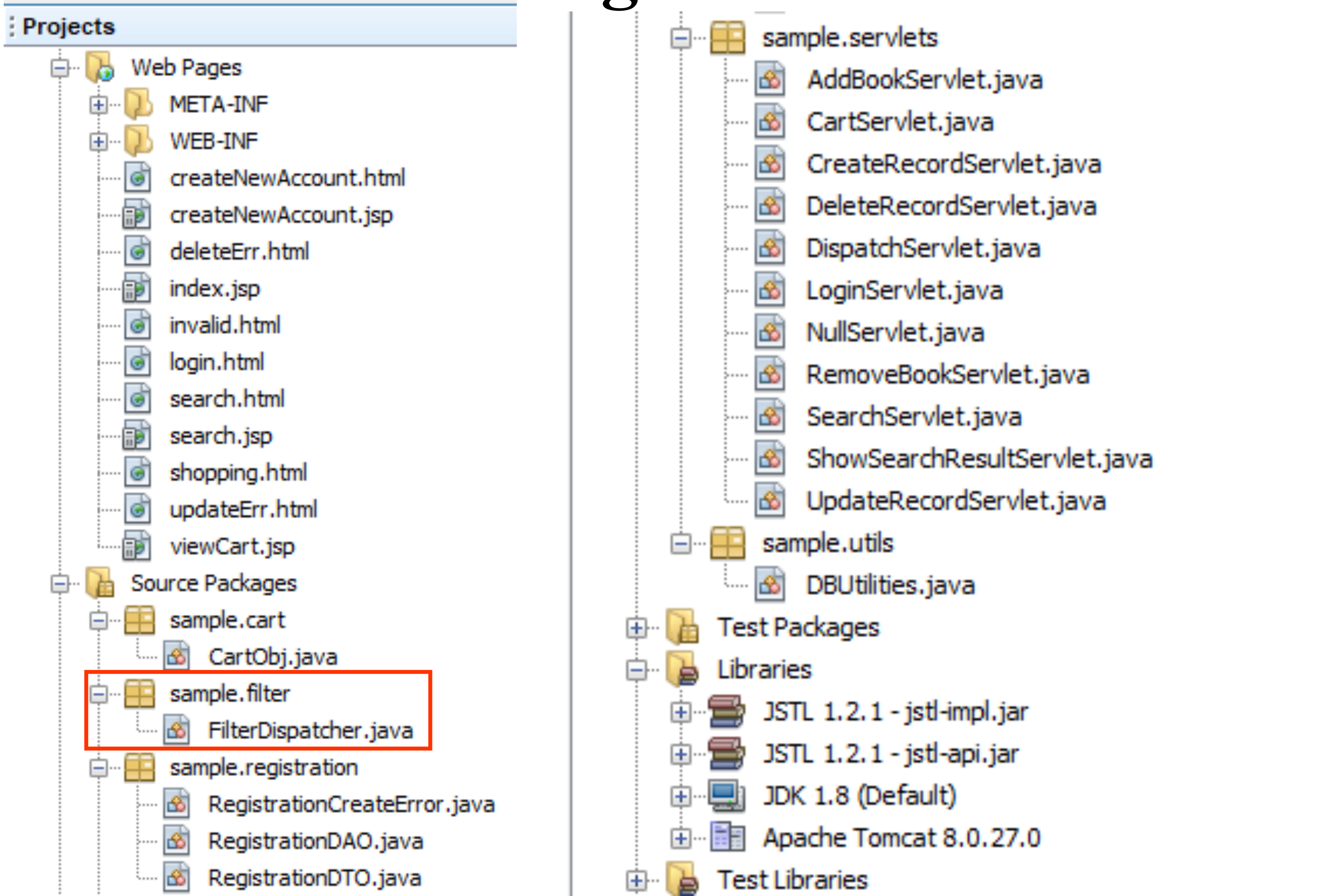
Appendix
MVC2 using Filter as Controller



```
    protected void processRequest(HttpServletRequest request, HttpServl
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String urlRewriting = "cart?btAction=View Your Cart";

        try  {
            HttpSession session = request.getSession(false);

            if (session != null) {
                CartObj cart = (CartObj)session.getAttribute("CART");

                if (cart != null) {
```

Fpt University

Appendix MVC2 using Filter as Controller



Projects
- Web Pages
  - META-INF
  - WEB-INF
  - createNewAccount.html
  - createNewAccount.jsp
  - deleteErr.html
  - index.jsp
  - invalid.html
  - login.html
  - search.html
  - search.jsp
  - shopping.html
  - updateErr.html
  - viewCart.jsp
- Source Packages
  - sample.cart
    - CartObj.java
  - sample.filter
    - FilterDispatcher.java
  - sample.registration
    - RegistrationCreateError.java
    - RegistrationDAO.java
    - RegistrationDTO.java
  - sample.servlets
    - AddBookServlet.java
    - CartServlet.java
    - CreateRecordServlet.java
    - DeleteRecordServlet.java
    - DispatchServlet.java
    - LoginServlet.java
    - NullServlet.java
    - RemoveBookServlet.java
    - SearchServlet.java
    - ShowSearchResultServlet.java
    - UpdateRecordServlet.java
  - sample.utils
    - DBUtilities.java
- Test Packages
- Libraries
  - JSTL 1.2.1 - jstl-impl.jar
  - JSTL 1.2.1 - jstl-api.jar
  - JDK 1.8 (Default)
  - Apache Tomcat 8.0.27.0
- Test Libraries