

# Session Management

## Sessions and Listeners

## Technique: Error Handling in Servlets

#Session #UrlRewriting #HiddenForm

#SessionTracking #Cookie

#ShoppingCart #ErrorHandler

#JSP

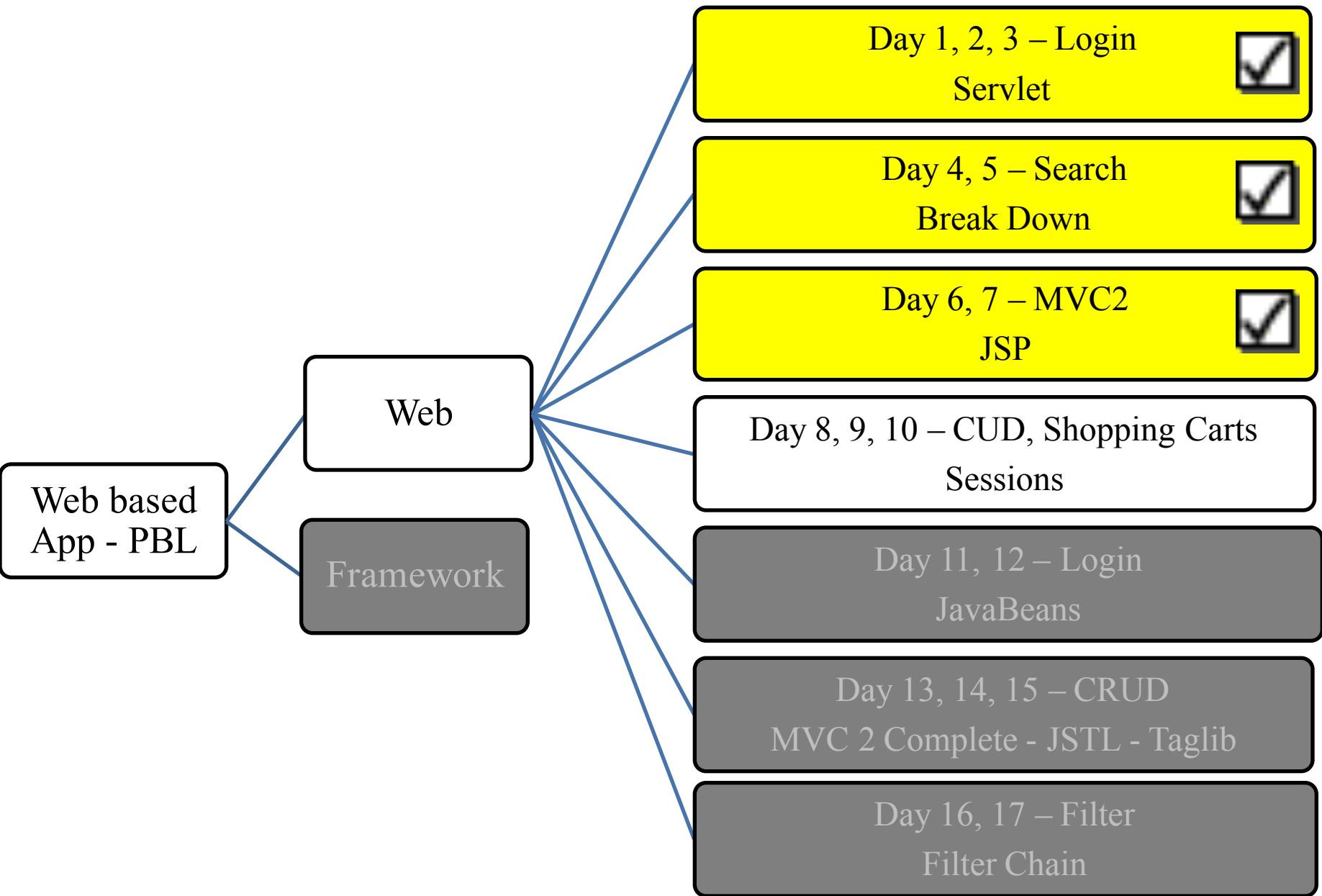
# Review

- **JSP**
  - JSP Syntax
    - Comment
    - Scripting Element: declaration, scriptlets, expression
    - Directives (page, include, taglib)
  - JSP Life Cycles
  - JSP Implicit Object
- **MVC Pattern**
  - No MVC
  - MVC 1
  - MVC 2

# Objectives

- **How to write CUD Web Application?**
  - Session Tracking Techniques
  - Manipulate DB Techniques in Web Application
  - Break down structure component in building web application
- **Techniques: Error Handling in Servlets**
  - Reporting Errors
  - Logging Errors
  - Users Errors vs. System Errors

# Objectives





# How to write CRUD Web Application

## Requirements

- After the web application had searched and shown the result, some following functions are required
  - The data grid allows the user **delete the selected row**. After **delete** action is completed, **the data grid is updated**
  - The data grid also allows the user **update the password and roles on the selected row**. After **update** action is completed, **the data grid is refreshed**
  - The application allows to **store the user's account** that the user can **access the resource without login in the second access**. **The username can be shown at the search result**
  - The application allows the user **shopping book and order them**
  - When the user login fail, the **register page** is shown. When **register is fail**, the **error page is shown**. Otherwise, the **login page is shown**
- The GUI of web application is present as following

# How to write CRUD Web Application

## Expectation



## Search Page

Search Value

No.	Username	Password	Last name	Role	Delete
1	IA1161	123456	Class IA1161	<input type="checkbox"/>	<a href="#">Delete</a>
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	<a href="#">Delete</a>
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	<a href="#">Delete</a>
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	<a href="#">Delete</a>
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	<a href="#">Delete</a>

http://localhost:8084/SE1162Servlet/SE1162Servlet?btAction=delete&pk=IA1161&lastSearchValue=a

# How to write CRUD Web Application

## Expectation



Welcome, khanh

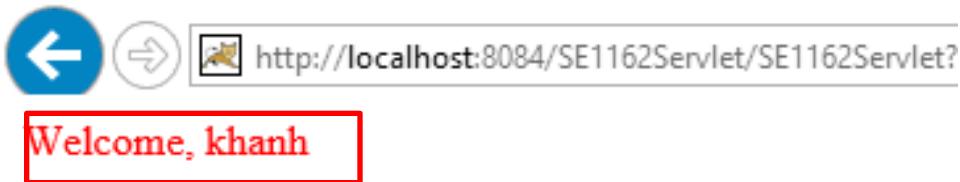
## Search Page

Search Value

No.	Username	Password	Last name	Role	Delete	Update
1	IA1161	123456	Class IA1161	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>

# How to write CRUD Web Application

## Expectation

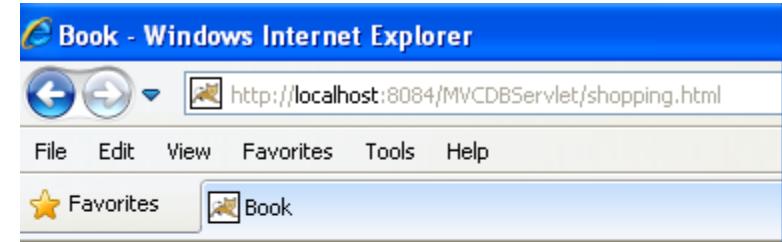
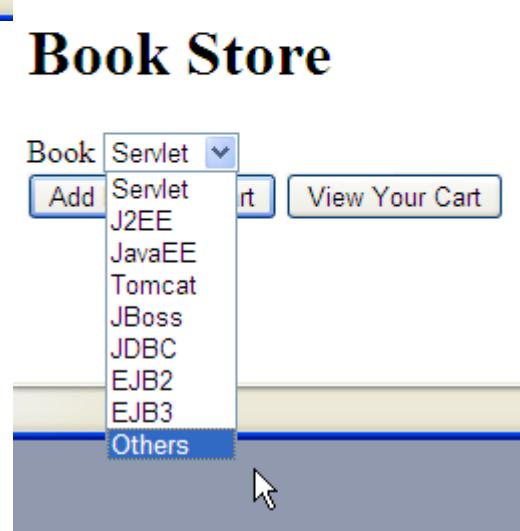
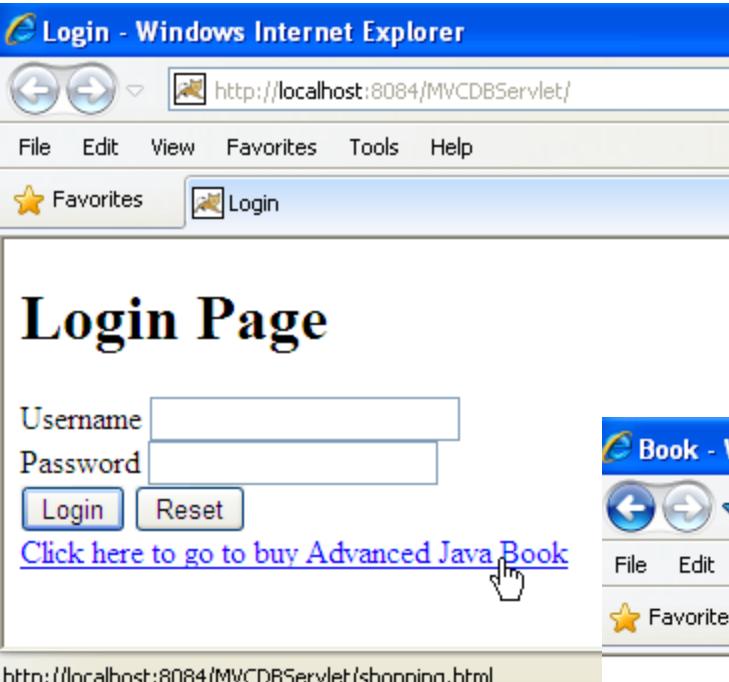


## Search Page

Search Value

# How to write CRUD Web Application

## Expectation



# How to write CRUD Web Application

## Expectation

A screenshot of a web browser window. The address bar shows the URL: <http://localhost:8084/MVCDBServlet/ProcessServlet?cboBook=Servlet&btAction=View+Your+Cart>. The menu bar includes File, Edit, View, Favorites, Tools, and Help. A toolbar with icons for back, forward, and search is visible. Below the toolbar, there are Favorites and Carts buttons. The main content area displays a table titled "Your Cart Items". The table has columns: No., Title, Quantity, and Action. It contains three items: JavaEE (1), Servlet (2), and Tomcat (2). At the bottom of the table are two buttons: "Add More Item to Cart" and "Remove".

No.	Title	Quantity	Action
1	JavaEE	1	<input type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input type="checkbox"/>

[Add More Item to Cart](#) [Remove](#)

A screenshot of a web browser window showing the same URL as the first screenshot. The main content area displays a table titled "Your Cart Items". The table has columns: No., Title, Quantity, and Action. It contains three items: JavaEE (1), Servlet (2), and Tomcat (2). The "Action" column for all three items now contains a checked checkbox (). At the bottom of the table are two buttons: "Add More Item to Cart" and "Remove".

No.	Title	Quantity	Action
1	JavaEE	1	<input checked="" type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input checked="" type="checkbox"/>

[Add More Item to Cart](#) [Remove](#)

A screenshot of a web browser window showing the same URL as the first screenshot. The main content area displays a table titled "Your Cart Items". The table has columns: No., Title, Quantity, and Action. It contains one item: Servlet (2). The "Action" column for this item contains an unchecked checkbox (). At the bottom of the table are two buttons: "Add More Item to Cart" and "Remove".

No.	Title	Quantity	Action
1	Servlet	2	<input type="checkbox"/>

[Add More Item to Cart](#) [Remove](#)

# How to write CRUD Web Application

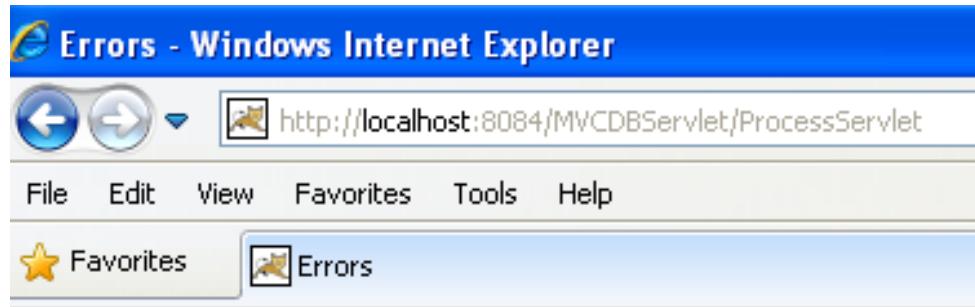
## Expectation



A screenshot of a web browser window. The address bar shows the URL <http://localhost:8084/MVCDBServlet/register.html>. The page title is **Register Page**. The form contains four input fields with validation requirements: **Username\*** (6 - 12 chars), **Password\*** (8 - 20 chars), **Confirm\***, and **Full name\*** (2 - 40 chars). Below the form are two buttons: **Register** and **Reset**. At the bottom of the page, there is a link labeled **Done**.

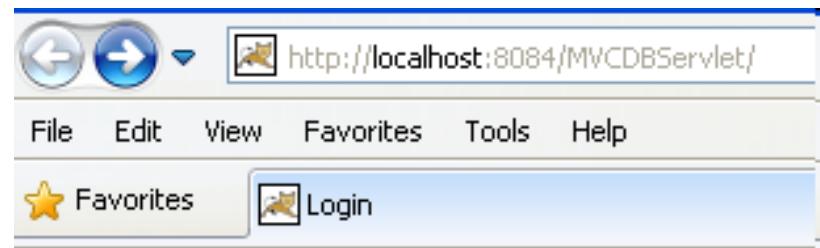
# How to write CRUD Web Application

## Expectation



### Errors occur

Username phai tu 6 - 15 ky tu  
Password phai tu 8 - 20 ky tu  
Full name phai tu 2 - 40 ky tu



### Login Page

Username

Password

# Sessions & Listeners

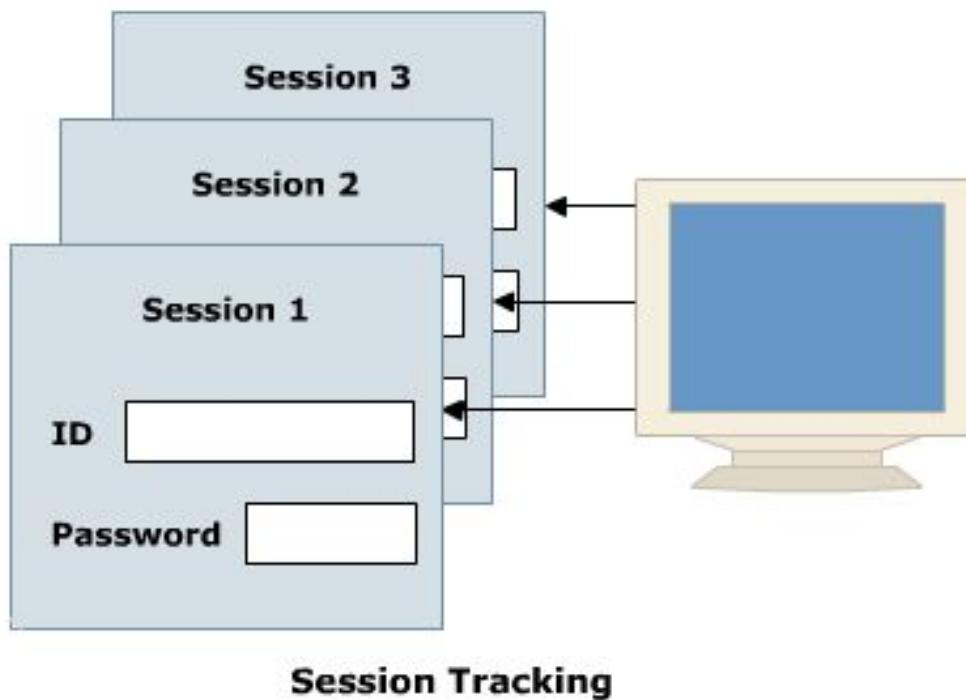
## Session

- Is the **period of connection** between client and server
- Is a group of activities that are performed by a user while accessing a particular web site
- HttpSession are **virtual connection** between client and server
- Web container reserves **an individual memory block for storing information about each session** → **Session objects.**
- **The session tracking** (mechanism)
  - Serves the purpose **tracking** the client identity and other state information required throughout the session
  - Allows the **server to keep a track of successive requests** made by same client
  - Allows **the customer to maintain the information with the server** as long as the customer does not log out from the website

# Sessions & Listeners

## Session Tracking Techniques

- URL Rewriting
- Hidden form field
- Cookies
- HttpSession interface



# Sessions & Listeners

## URL Rewriting

- **Maintains the state of end user by modifying the URL.**
- **Adds some extra data at the end of the URL**
- Is **used** when the **information to be transferred** is not critical.
- **Syntax: url?query\_string**
- **Ex**
  - <form action="<http://localhost:8080/UpdateProfile?uid=123>" method="get">  
-----</form>
- **Disadvantages:**
  - Server side **processing is tedious**.
  - Every URL that is **returned** to the **user** should have **additional information appended** to it.
  - If the user **leaves the session** and **opens** the **Web page using a link or bookmark** then the session information is lost.
  - The **query string is limited**

# How to write CRUD Web Application

## Delete Function



Welcome, khanh

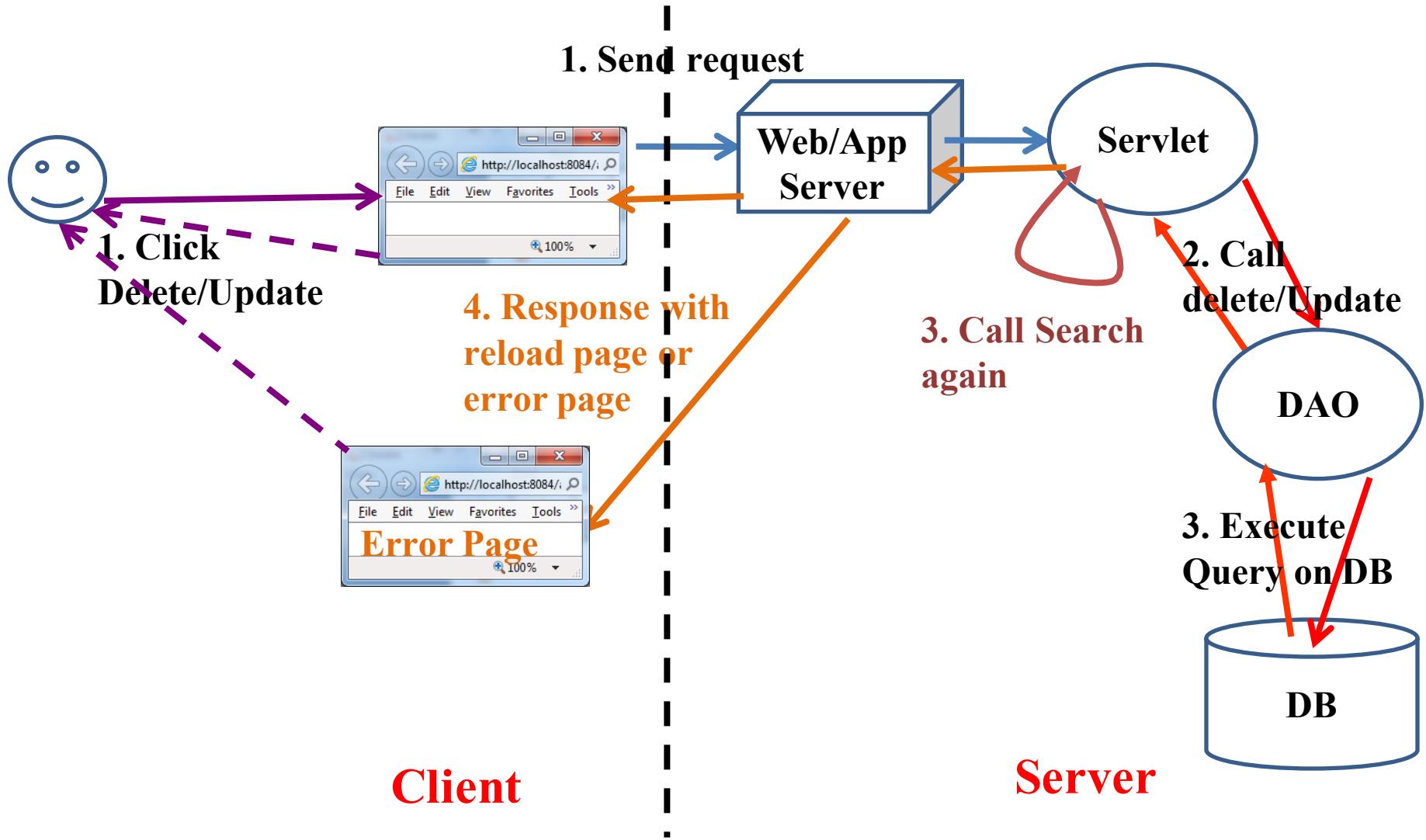
## Search Page

Search Value

No.	Username	Password	Last name	Role	Delete
1	IA1161	123456	Class IA1161	<input type="checkbox"/>	<a href="#">Delete</a>
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	<a href="#">Delete</a>
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	<a href="#">Delete</a>
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	<a href="#">Delete</a>
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	<a href="#">Delete</a>

# How to write CRUD Web Application

## Interactive Server Model



# How to write CRUD Web Application

## Delete Function



Welcome, khanh

## Search Page

Search Value

No.	Username	Password	Last name	Role	Delete
1	IA1161	123456	Class IA1161	<input type="checkbox"/>	<a href="#">Delete</a>
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	<a href="#">Delete</a>
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	<a href="#">Delete</a>
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	<a href="#">Delete</a>
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	<a href="#">Delete</a>

# Sessions & Listeners

## Hidden Form Fields

- Simplest technique to **maintain the state of an end user**.
- **Insert** the session identifier into the **hidden form field** in the HTML of each page
- **Embedded the hidden form field in an HTML form** and **not visible** when you view an HTML file in a browser window.
- The session information can be **extracted** by the application by **searching** for these fields. The servlets or JSP pages read the field **using request.getParameter()**.
- **Syntax**

```
<input type="hidden" name="..." value="...">
```

- **Ex**

```
<input type="hidden" name="productId" value="P01">
```

- **Advantages**
  - Simplest way to implement session tracking
  - Displays **nothing** on the HTML page but can be used to hold any kind of data
  - Helps to maintain a **connection between two pages**

- **Disadvantages:**
  - Work on the **dynamic pages**.
  - This method of session tracking **displays sensitive information to the user**.

# How to write CRUD Web Application

## Update Function



Welcome, khanh

## Search Page

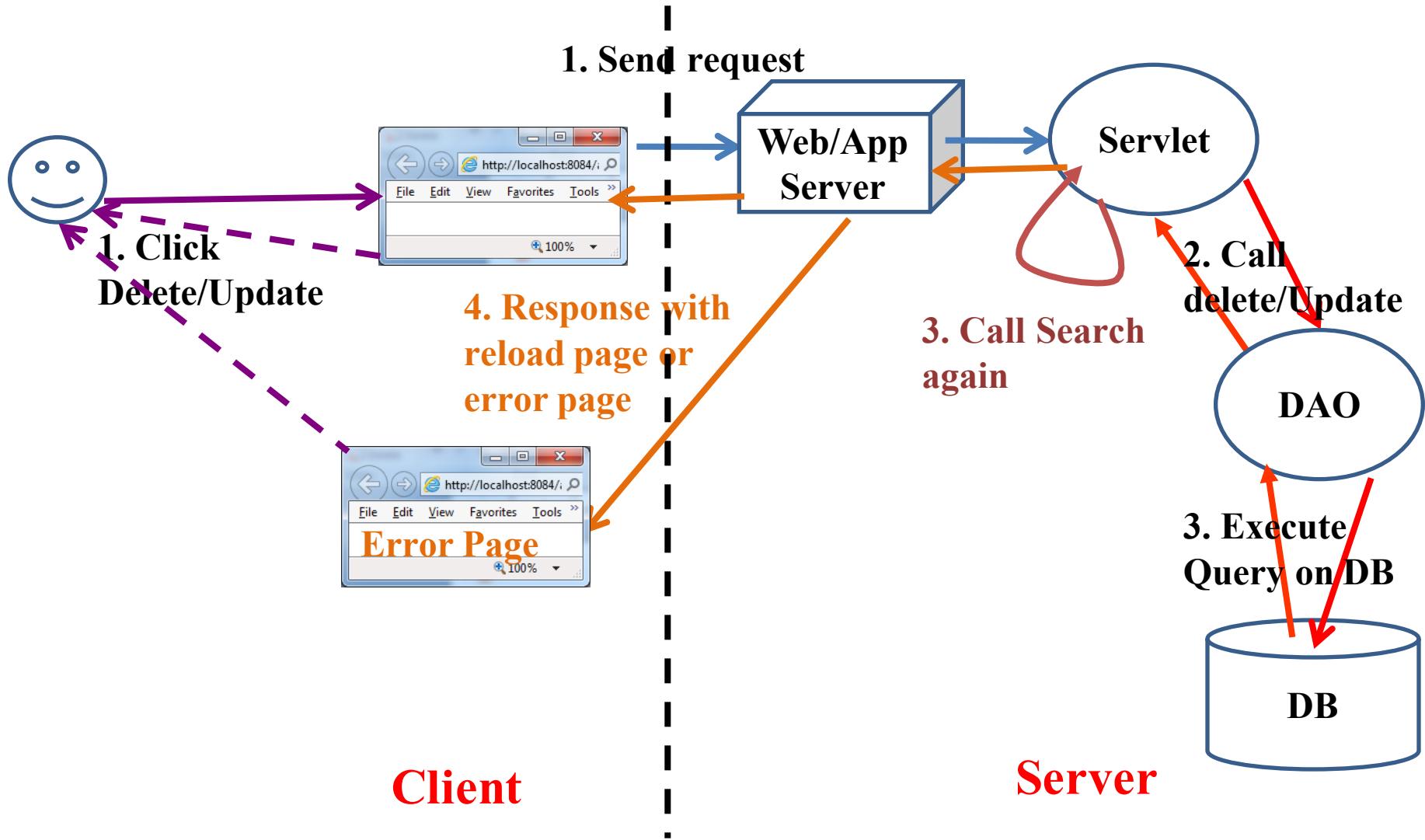
Search Value a

Search

No.	Username	Password	Last name	Role	Delete	Update
1	IA1161	123456	Class IA1161	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	<a href="#">Delete</a>	<a href="#">Update</a>

# How to write CRUD Web Application

## Interactive Server Model



# Sessions & Listeners

## Cookies

- Is a **small piece** of information **sent by the web server to the client** to keep track of users.
- Size of each cookie can be a maximum of 4 KB.
- Cookie has values in the form of **key-value pairs**
- When the **server sends** a cookie, the **client receives** the cookie, **saves and sends it back** to the **server** each time the **client accesses** a page on that server
- Can uniquely identify a client (In the case of J2EE web applications, the cookie returned has a standard name **JSESSIONID** and store in memory)
- A web browser is expected to support **20 Cookies** per host



Concept of Cookie

# Sessions & Listeners

## Cookies

- **Advantages**
  - Remember user IDs and password.(low security)
  - To track visitors on a Web site for better service and new features.
  - Cookies enable efficient ad processing.
  - Support e-advertisement on Internet.
  - Security (can not affect virus).
- **Disadvantages**
  - Personal information is exposed to the other users.  
(spam/junk mail, pop up ...)
  - Cookies fails to work if the security level is set too high in the Internet browser.
  - Most browsers enable the user at the client machine to deactivate (not to accept) cookies.
  - The size and number of cookies stored are limited.
- **Note**
  - Browser is accepted cookies
  - Cookies are stored at
    - C:\Documents and Settings\UserName\Cookies\UserName@ContextPath[n].txt
    - C:\Users\UserName\AppData\Local\Microsoft\Windows\Temporary Internet Files\UserName@host[n].txt
  - Cookies are existed following the **setMaxAge** and deleted automatically by OS

# Sessions & Listeners

## Cookies

- The servlet API provides **javax.servlet.http.Cookie** class for creating and working with cookies
- The **constructor** for the cookies class is: `Cookie(java.lang.String name, java.lang.String value)`
- Sending Cookie**

Methods	Descriptions
<b>addCookie</b>	<ul style="list-style-type: none"> <li>- <b>public void addCookie(cookie1);</b></li> <li>- Adds field to the HTTP response headers to send cookies to the browser, one at a time</li> <li>- Adds specified cookie to the response</li> <li>- Can be called multiple times to set more than one cookies</li> </ul>
<b>setValue</b>	<ul style="list-style-type: none"> <li>- <b>public void setValue(String newValue);</b></li> <li>- Assigns a new value to a cookie after the cookie is created. In case if binary value is used, base 64 can be used for encoding</li> </ul>
<b>setPath</b>	<ul style="list-style-type: none"> <li>- <b>public void setPath(String path);</b></li> <li>- Sets the path for the cookie. The cookie is available to all the pages specified in the directory and its subdirectories. A cookie's path must have the servlet which sets the cookie</li> </ul>
<b>setMaxAge</b>	<ul style="list-style-type: none"> <li>- <b>public void setMaxAge(int expiry);</b></li> <li>- The maximum age of the cookie in seconds. If the value is positive, then the cookie will expire after that many seconds which is specified by the expiry</li> </ul>

# Sessions & Listeners

## Cookies

```
// Create cookies for first and last names.  
Cookie firstName = new Cookie("first_name",  
                               request.getParameter("first_name"));  
Cookie lastName = new Cookie("last_name",  
                             request.getParameter("last_name"));  
  
// Set expiry date after 24 Hrs for both the cookies.  
firstName.setMaxAge(60 * 60 * 24);  
lastName.setMaxAge(60 * 60 * 24);  
  
// Add both the cookies in the response header.  
response.addCookie(firstName);  
response.addCookie(lastName);
```

# Sessions & Listeners

## Cookies

- **Reading Cookie**

Methods	Descriptions
<b>getCookies</b>	<ul style="list-style-type: none"> <li>- <b>Cookie [] cookies = request.getCookies();</b></li> <li>- Returns an array containing all of the Cookie objects the client sends with the request</li> </ul>
<b>getMaxAge</b>	<ul style="list-style-type: none"> <li>- <b>public int getMaxAge();</b></li> <li>- Returns the maximum age of the cookie.</li> <li>- Returns an integer which specify the maximum age of the cookies in seconds</li> </ul>
<b>getValue</b>	<ul style="list-style-type: none"> <li>- <b>public String getValue();</b></li> <li>- Returns the value of the cookie</li> </ul>
<b>getName</b>	<ul style="list-style-type: none"> <li>- <b>public String getName()</b></li> <li>- Returns the name of cookie. Once the cookie has been created its name cannot be changed</li> </ul>
<b>getPath</b>	<ul style="list-style-type: none"> <li>- <b>public void getPath()</b></li> <li>- Returns the path on the server to which the client return the cookie. The cookie is available to all sub paths on the server</li> </ul>

# Sessions & Listeners

## Cookies

- Reading Cookie

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

    Cookie cookie = null;
    Cookie[] cookies = null;

    // Get an array of Cookies associated with this domain
    cookies = request.getCookies();

    // Set response content type
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    String title = "Reading Cookies Example";
    String docType = "<!doctype html public '-//w3c//dtd html 4.0 ' +
                    "transitional//en\\>\\n";
    out.println(docType + "<html>\\n" + "<head><title>" + title +
                "</title></head>\\n" +
                "<body bgcolor = \"#f0f0f0\">\\n");

    if (cookies != null) {
        out.println("<h2> Found Cookies Name and Value</h2>");

        for (int i = 0; i < cookies.length; i++) {
            cookie = cookies[i];
            out.print("Name : " + cookie.getName() + ", ");
            out.print("Value: " + cookie.getValue() + " <br/>");
        }
    } else {
        out.println("<h2>No cookies founds</h2>");
    }
    out.println("</body>");
    out.println("</html>");

}
```

# Sessions & Listeners

## Cookies

- Delete Cookie

```

public class DeleteCookie extends HttpServlet {

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

        Cookie cookie = null;
        Cookie[] cookies = null;

        // Get an array of Cookies associated with this domain
        cookies = request.getCookies();

        // Set response content type
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        String title = "Delete Cookies Example";
        String docType = "<!doctype html public '-//w3c//dtd html 4.0 ' +
                        "transitional//en'\n";

        out.println(docType + "<html>\n" + "<head><title>" + title +
                   "</title></head>\n" +
                   "<body bgcolor = '#f0f0f0'>\n");

        if (cookies != null) {
            out.println("<h2> Cookies Name and Value</h2>");

            for (int i = 0; i < cookies.length; i++) {
                cookie = cookies[i];

                if ((cookie.getName()).compareTo("first name") == 0) {
                    // delete cookie
                    cookie.setMaxAge(0);
                    response.addCookie(cookie);
                    out.print("Deleted cookie : " + cookie.getName() + "<br/>");
                }
                out.print("Name : " + cookie.getName() + ", ");
                out.print("Value: " + cookie.getValue() + " <br/>");
            }
        } else {
            out.println("<h2>No cookies founds</h2>");
        }
        out.println("</body>");
        out.println("</html>");
    }
}

```

# Sessions & Listeners

## Cookies – Example

**Cookie - Windows Internet Explorer**

http://localhost:8084/AJDay3\_7/

File Edit View Favorites Tools Help

Favorites Cookie

### Cookie Demo

Name

**Info - Windows Internet Explorer**

http://localhost:8084/AJDay3\_7/PrintCookieServlet

File Edit View Favorites Tools Help

Favorites Info

### Cookie Information

Name: userName Value: khanhkt

**Add - Windows Internet Explorer**

http://localhost:8084/AJDay3\_7/AddCookieServlet?txtName=khanhkt

File Edit View Favorites Tools Help

Favorites Add

### Adding Cookie processing

[Print Cookie](#)

http://localhost:8084/AJDay3\_7/PrintCookieServlet

c:\Documents and Settings\Trong Khanh\Cookies\\*

Name	Ext	Size
[..]	<DIR>	
ZQ2VOROO	txt	84
index	dat	32.768

**Lister - [C:\Documents and Settings\Trong Khanh\Cookies\ZQ2VOROO.txt]**

File Edit Options Help

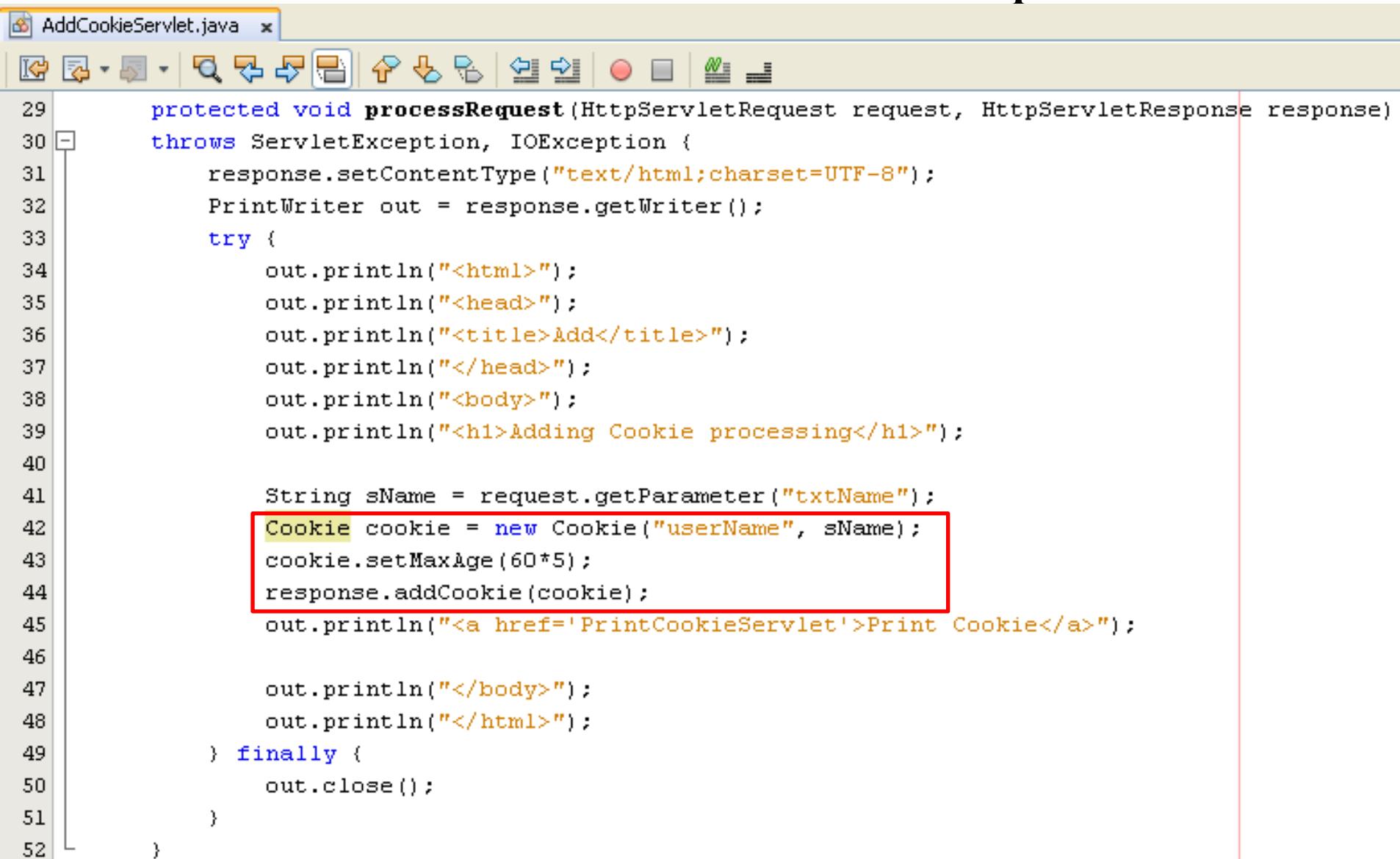
```

userName
khanhkt
localhost/AJDay3_7/
1024
1426045440
30324610
2727262736
30324609
*

```

# Sessions & Listeners

## Cookies – Example



The screenshot shows a Java code editor with the file `AddCookieServlet.java` open. The code implements a `HttpServlet` to handle cookie creation. A red box highlights the section where a cookie is created and added to the response.

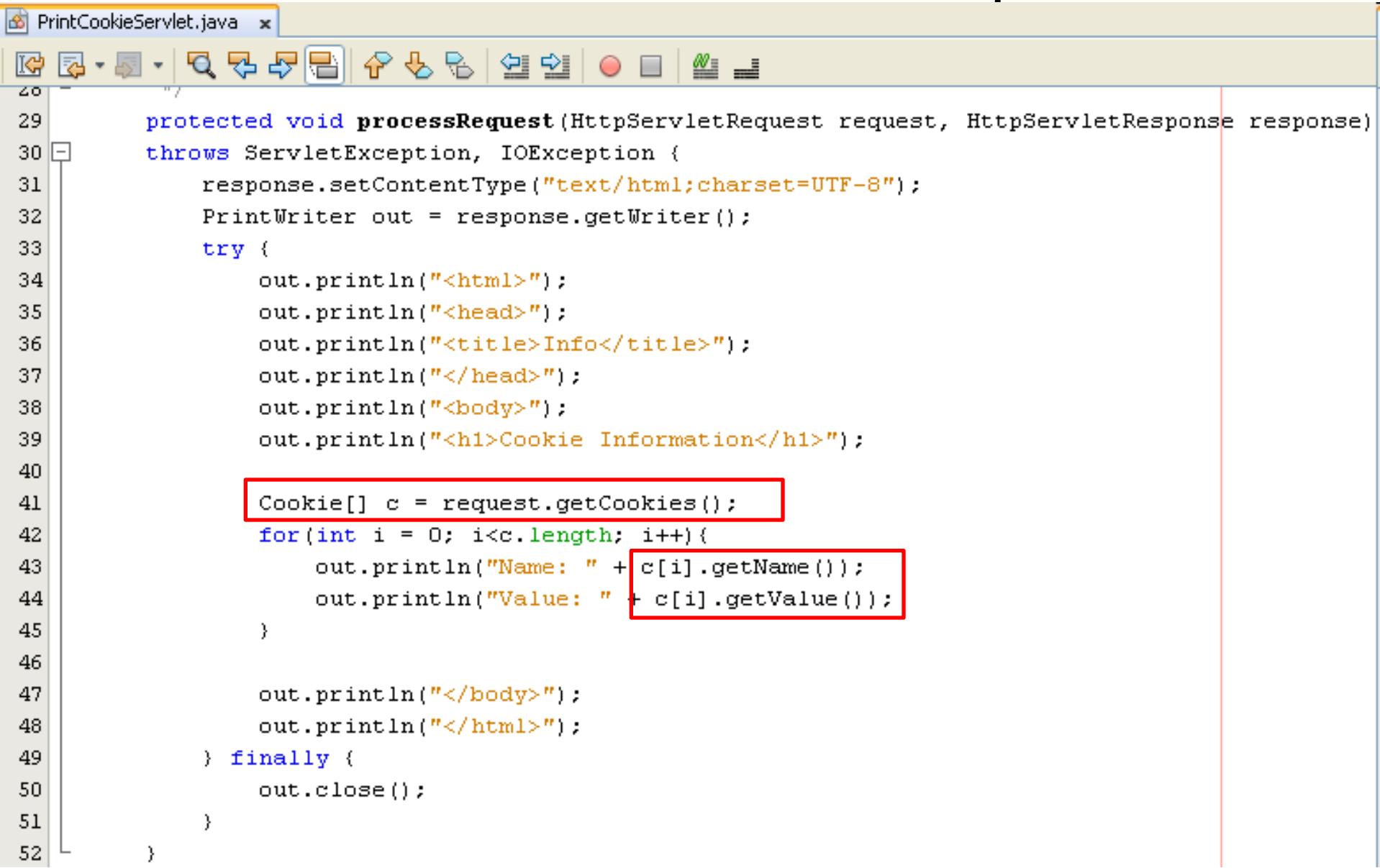
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");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

        String sName = request.getParameter("txtName");
        Cookie cookie = new Cookie("userName", sName);
        cookie.setMaxAge(60*5);
        response.addCookie(cookie);
        out.println("<a href='PrintCookieServlet'>Print Cookie</a>");

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

# Sessions & Listeners

## Cookies – Example



```
PrintCookieServlet.java x
29     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30         throws ServletException, IOException {
31         response.setContentType("text/html;charset=UTF-8");
32         PrintWriter out = response.getWriter();
33         try {
34             out.println("<html>");
35             out.println("<head>");
36             out.println("<title>Info</title>");
37             out.println("</head>");
38             out.println("<body>");
39             out.println("<h1>Cookie Information</h1>");
40
41             Cookie[] c = request.getCookies();
42             for(int i = 0; i<c.length; i++){
43                 out.println("Name: " + c[i].getName());
44                 out.println("Value: " + c[i].getValue());
45             }
46
47             out.println("</body>");
48             out.println("</html>");
49         } finally {
50             out.close();
51         }
52     }
```

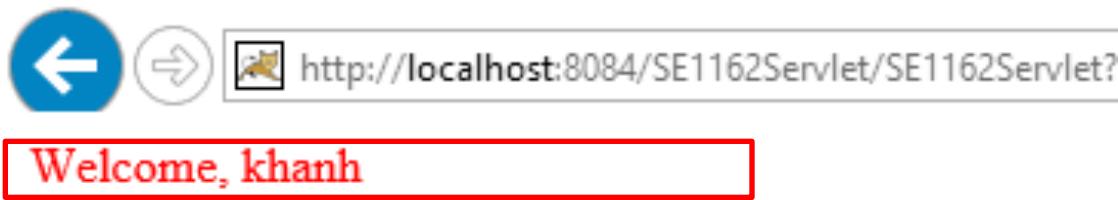


# How to write CRUD Web Application Requirements

- After the web application had searched and shown the result, some following functions are required
  - ...
  - The application allows to **store the user's account** that the **user can access the resource without login in the second access**.  
**The username can be shown at the search result**
  - ...
- The GUI of web application is present as following

# How to write CRUD Web Application

## Store Info



## Search Page

Search Value

# How to write CRUD Web Application



## Interactive Server Model

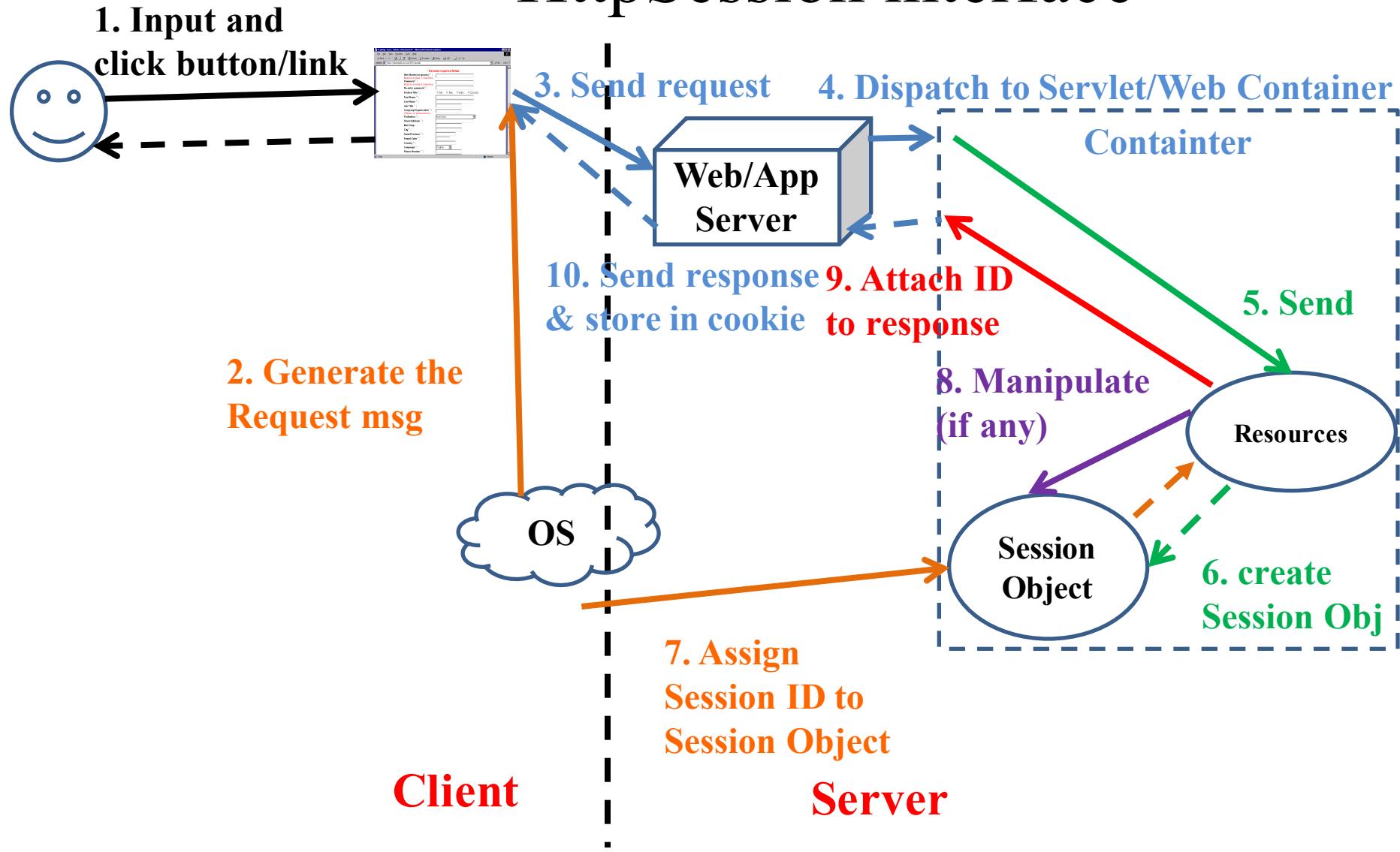
Draw your self

Client

Server

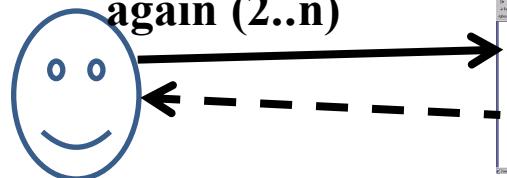
# Sessions & Listeners

## HttpSession interface



# Sessions & Listeners

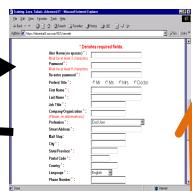
1. Input and click button/link again (2..n)



2. Generate the Request msg with id session



3. Send request



Web/App Server

4. Dispatch to Servlet/Web Container

Container

10. Send response

7. Response

5. Send

Resources

Session Object

6. Manipulate (if any)

Client

Server

# Sessions & Listeners

## Session Management: General Principles

- Each of these requests **needs to carry a unique ID**, which identifies the session to which it belongs.
- The web application will **allocate this unique ID** on the first request from the client.
- The **ID must be passed back to the client** so that the client **can pass it back again with its next request**. In this way, the web application will know to which session the request belongs. This implies that the client **must need to store the unique ID somewhere**—and that's where session management mechanisms come in
- The **default mechanism for session management is cookie**

# Sessions & Listeners

## HttpSession interface

- Identifying user in a multi-page request scenario and information about that user
- Is used to **create a session between the client and server** by servlet container
  - When **users make a request**, the **server signs it a session object** and a **unique session ID**
  - The session ID matches the user with the session object in subsequent requests
  - The **session ID and the session object** are **passed along with the request to the server**
- **Session Timeout**
  - Is necessary as session utilizes the memory locations
  - Prevent the number of session increasing infinitely.
  - Set either in the web.xml file or can be set by the method **setMaxInactiveInterval()**

# Sessions & Listeners

## HttpSession interface Methods

Methods	Descriptions
<b>getSession</b>	<ul style="list-style-type: none"> <li>- <b>request.getSession(boolean create);</b></li> <li>- Obtain a current session objects</li> <li>- The getSession() method with true parameter is used to create a new session (no current session)</li> </ul>
<b>getId</b>	<ul style="list-style-type: none"> <li>- <b>public String getId()</b></li> <li>- Returns a string containing the unique identifier assigned to this session. The servlet container assigns the identifier and it is implementation independent</li> </ul>
<b>getCreationTime</b>	<ul style="list-style-type: none"> <li>- <b>public long getCreationTime()</b></li> <li>- Returns the creation time of session.</li> </ul>
<b>getLastAccessedTime</b>	<ul style="list-style-type: none"> <li>- <b>public long getLastAccessedTime()</b></li> <li>- Returns the last accessed Time of session</li> </ul>
<b>getMaxInactiveInterval</b>	<ul style="list-style-type: none"> <li>- <b>public int getMaxInactiveInterval()</b></li> <li>- Returns the maximum time interval, in seconds, for which the servlet container will keep the session alive between the client accesses</li> </ul>
<b>setMaxInactiveInterval</b>	<ul style="list-style-type: none"> <li>- <b>public void setMaxInactiveInterval(int interval)</b></li> <li>- Specifies the time, in seconds, between the client requests before the servlet container invalidates the current session</li> </ul>

# Sessions & Listeners

## HttpSession interface Methods

Methods	Descriptions
<b>isNew</b>	<ul style="list-style-type: none"> <li>- <b>public boolean isNew()</b></li> <li>- Returns true if the client is unaware about the session or choose not to be part of the session</li> </ul>
<b>invalidate</b>	<ul style="list-style-type: none"> <li>- <b>public void invalidate()</b></li> <li>- Invalidates the session and the objects bound to the session are bounded. This method throws IllegalStateException if called on already invalidated session</li> <li>- To avoid the hacker from causing any harm</li> <li>- Destroys the data in a session that another servlet or JSP might require in future. Therefore, invalidating a session should be done cautiously as sessions are associated with client, not with individual servlets or JSP pages</li> </ul>

# Sessions & Listeners

## HttpSession interface – Example

Session - Windows Internet Explorer

http://localhost:8084/AJDay3\_7/SessionServlet

File Edit View Favorites Tools Help

Favorites Session

### HttpSession Interface Demo

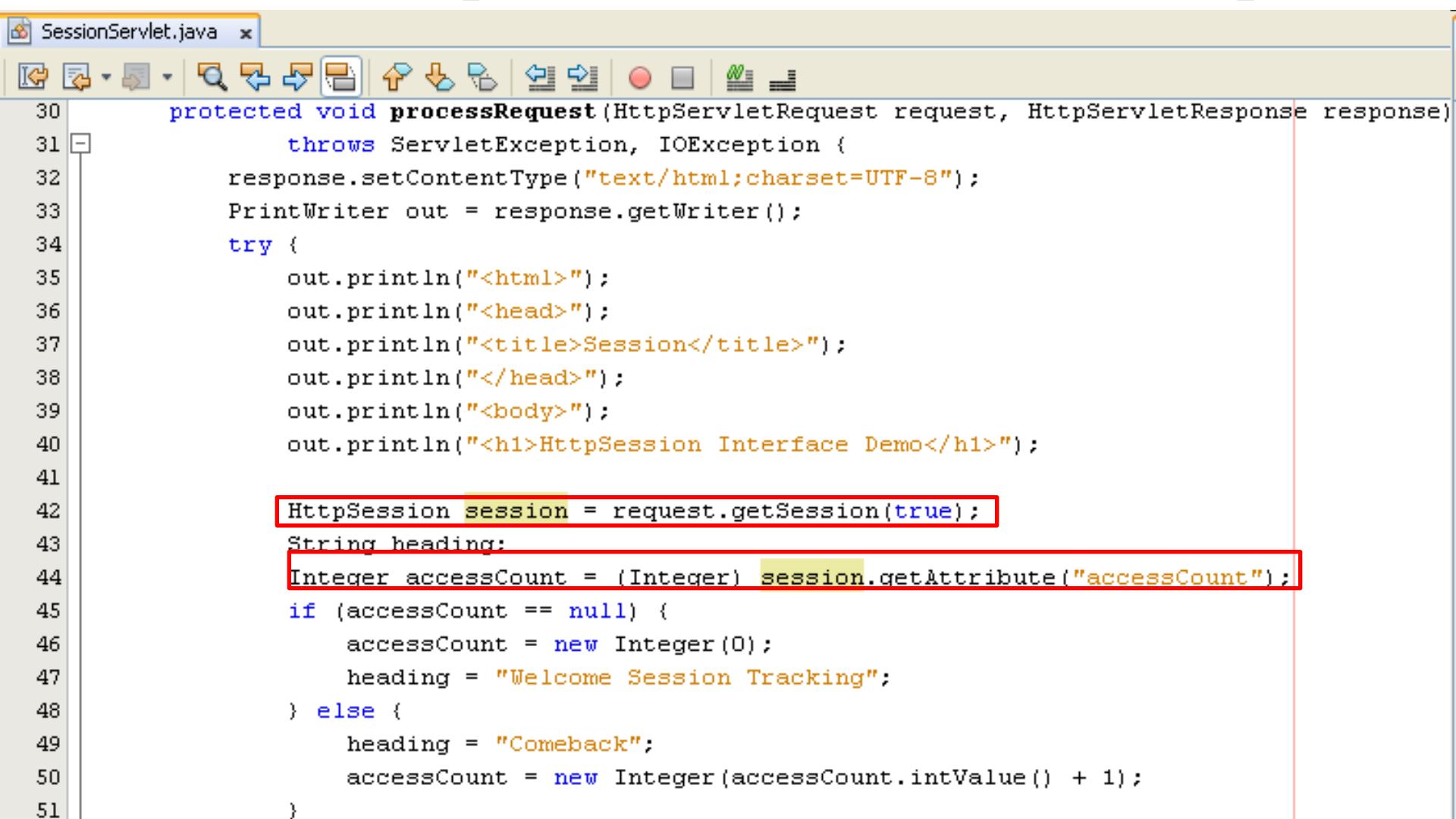
### Welcome Session Tracking

Information on Session:

Info Type	Value
ID	27C8C18AA205D9CBD95B53C6D62A783B
Create time	Sun Sep 22 18:02:59 ICT 2013
Time of Last Access	Sun Sep 22 18:02:59 ICT 2013
Number of Previous Accesses	0
Session Time out	1800

# Sessions & Listeners

## HttpSession interface – Example



The screenshot shows a Java code editor with the file `SessionServlet.java` open. The code implements the `processRequest` method of a servlet. It prints an HTML page and then retrieves the `HttpSession` from the request. The code uses `request.getSession(true)` to ensure a new session is created if none exists. This line is highlighted with a red rectangle. The code then checks if an attribute named `accessCount` exists in the session. If it does not, it creates a new `Integer` object set to 0 and sets the session attribute. If it does exist, it increments the value by 1. The heading of the page is then updated based on whether it's a new session or a comeback.

```
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>Session</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>HttpSession Interface Demo</h1>");

        HttpSession session = request.getSession(true);
        String heading;
        Integer accessCount = (Integer) session.getAttribute("accessCount");
        if (accessCount == null) {
            accessCount = new Integer(0);
            heading = "Welcome Session Tracking";
        } else {
            heading = "Comeback";
            accessCount = new Integer(accessCount.intValue() + 1);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

# Sessions & Listeners

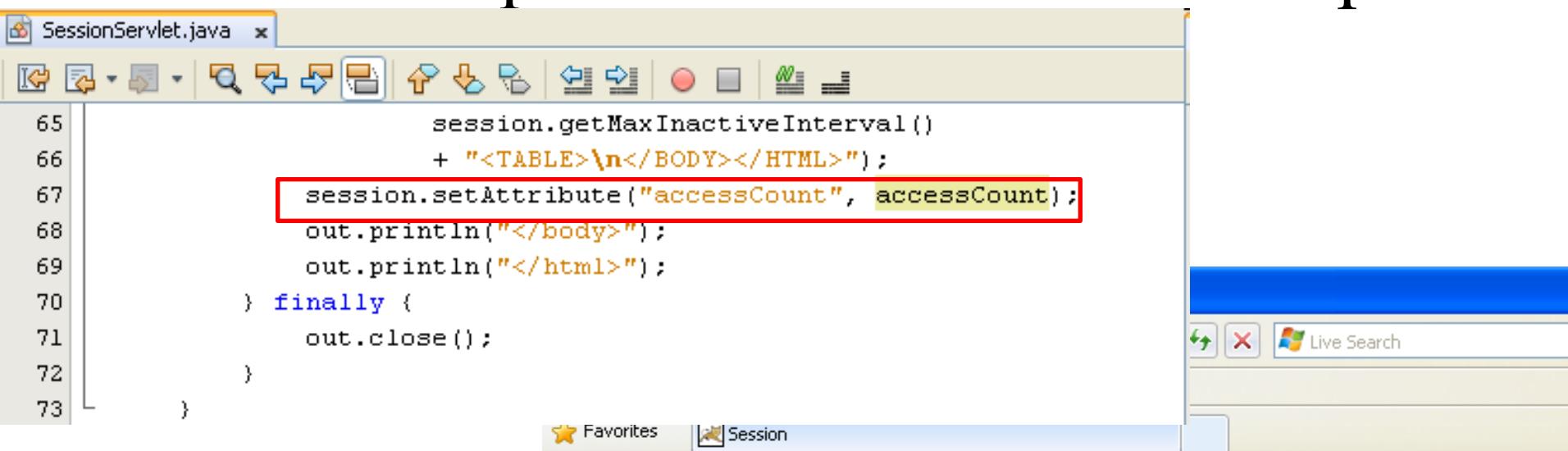
## HttpSession interface – Example

```
DateFormat formatter = DateFormat.getDateTimeInstance(
    DateFormat.MEDIUM, DateFormat.MEDIUM);
out.println("<H1 ALIGN=\"CENTER\">" + heading +
    "</H1>\n<H2>Information on Session:</H2>\n"
    + "<TABLE BORDER=1 ALIGN=\"CENTER\">\n<TR BGCOLOR="
    + "#FFAD00">\n<TH>Info Type<TH>Value\n"
    + "<TR>\n<TD>ID\n<TD>" + session.getId() +
    "\n<TR>\n<TD>Create time\n<TD>"
    + new Date(session.getCreationTime()) +
    "\n<TR>\n<TD>Time of Last Access\n<TD>"
    + new Date(session.getLastAccessedTime()) +
    "\n<TR>\n<TD>Number of Previous Accesses\n<TD>"
    + accessCount + "\n<TR>\n<TD>Session Time out\n<TD>" +
    session.getMaxInactiveInterval()
    + "<TABLE>\n</BODY></HTML>");
out.println("</body>");
out.println("</html>");

} finally {
    out.close();
}
```

# Sessions & Listeners

## HttpSession interface – Example



The screenshot shows an IDE interface with a code editor and a browser preview window.

**Code Editor (SessionServlet.java):**

```

65         session.getMaxInactiveInterval()
66         + "<TABLE>\n</BODY></HTML>");
67         session.setAttribute("accessCount", accessCount);
68         out.println("</body>");
69         out.println("</html>");
70     } finally {
71         out.close();
72     }
73 }
```

A red box highlights the line `session.setAttribute("accessCount", accessCount);`.

**Browser Preview:**

The title bar says "HttpSession Interface Demo". The page content includes a red-bordered box containing the word "Comeback".

### HttpSession Interface Demo

Comeback

#### Information on Session:

Info Type	Value
ID	198A528D0D5B47FD7BCBDA0FCEFA3229
Create time	Sun Sep 22 18:05:14 ICT 2013
Time of Last Access	Sun Sep 22 18:05:14 ICT 2013
Number of Previous Accesses	1
Session Time out	1800

# Sessions & Listeners

## HttpSession interface

- **Distributed Session**
  - A session is **available** to be **shared between web resources** in a single web application (*e.g. a session *cannot cross web application boundaries**)
- Session Death is **controlled** in one of 3 ways
  - Application Server Global Default
  - Web Application Default (minutes)
    - A negative value or zero value causes the session to never expire



The screenshot shows a code editor window titled "web.xml". The XML code defines a session timeout of 30 minutes:

```
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
```

- Individual Session Setting using **setMaxInactivateInterval()** method
  - A negative value supplied as an argument causes the session to never expire
- Other Session APIs
  - **HttpSession.getServletContext()** returns the SessionContext that the session is attached

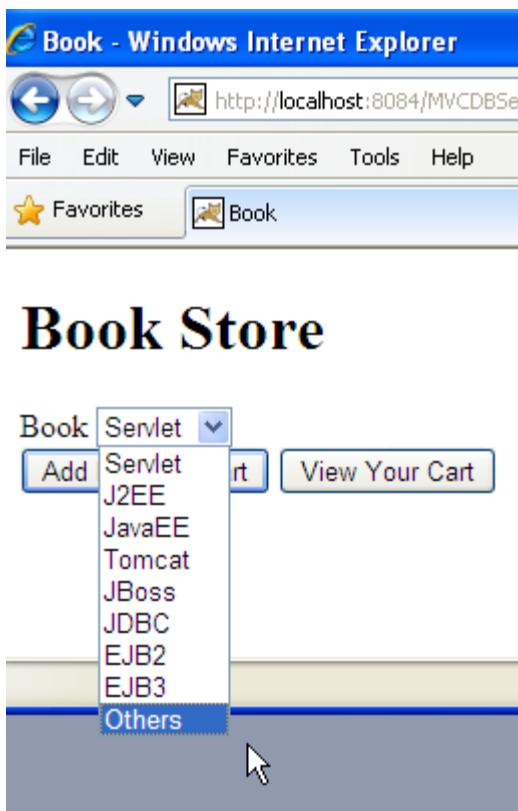
# How to write CRUD Web Application

## Shopping Cart



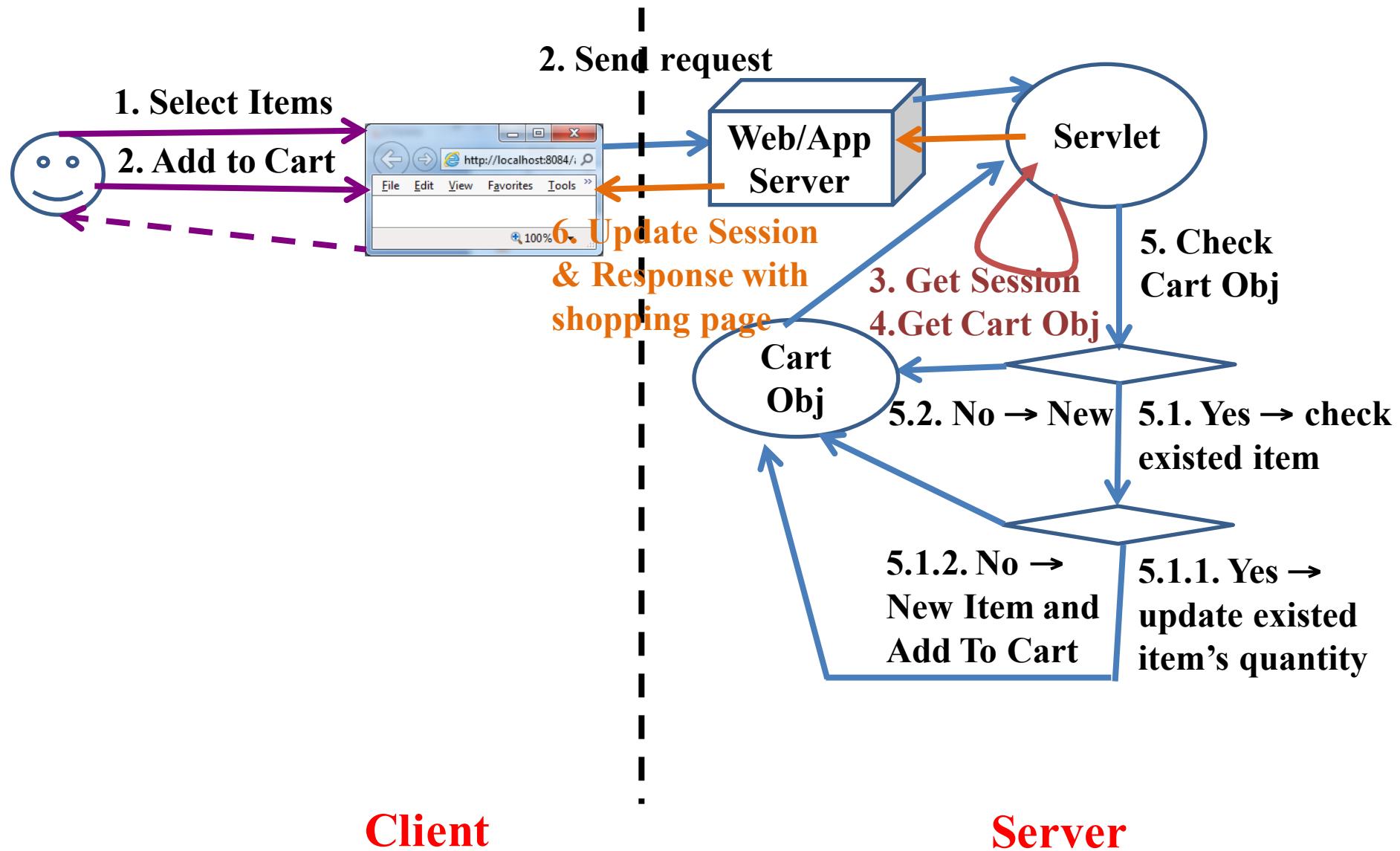
# How to write CRUD Web Application

## Shopping Cart – Add To Cart



# How to write CRUD Web Application

## Interactive Server Model – Add To Cart



# How to write CRUD Web Application

## Shopping Cart – View Cart

The screenshot shows a web browser window with the URL <http://localhost:8084/MVCDBServlet/ProcessServlet?cboBook=Servlet&btAction=View+Your+Cart>. The browser's title bar also displays this URL. The menu bar includes File, Edit, View, Favorites, Tools, and Help. A toolbar below the menu bar has a Favorites icon and a Carts icon. The main content area is titled "Your Cart Items". It contains a table with the following data:

No.	Title	Quantity	Action
1	JavaEE	1	<input type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input type="checkbox"/>

Below the table are two buttons: "Add More Item to Cart" and "Remove".

The screenshot shows a web browser window with the URL <http://localhost:8084/MVCDBServlet/ProcessServlet?btAction=View%20Your%20Cart>. The browser's title bar also displays this URL. The menu bar includes File, Edit, View, Favorites, Tools, and Help. A toolbar below the menu bar has a Favorites icon and a Carts icon. The main content area is titled "Your Cart Items". It contains a table with the following data:

No.	Title	Quantity	Action
1	Servlet	2	<input type="checkbox"/>

Below the table are two buttons: "Add More Item to Cart" and "Remove".

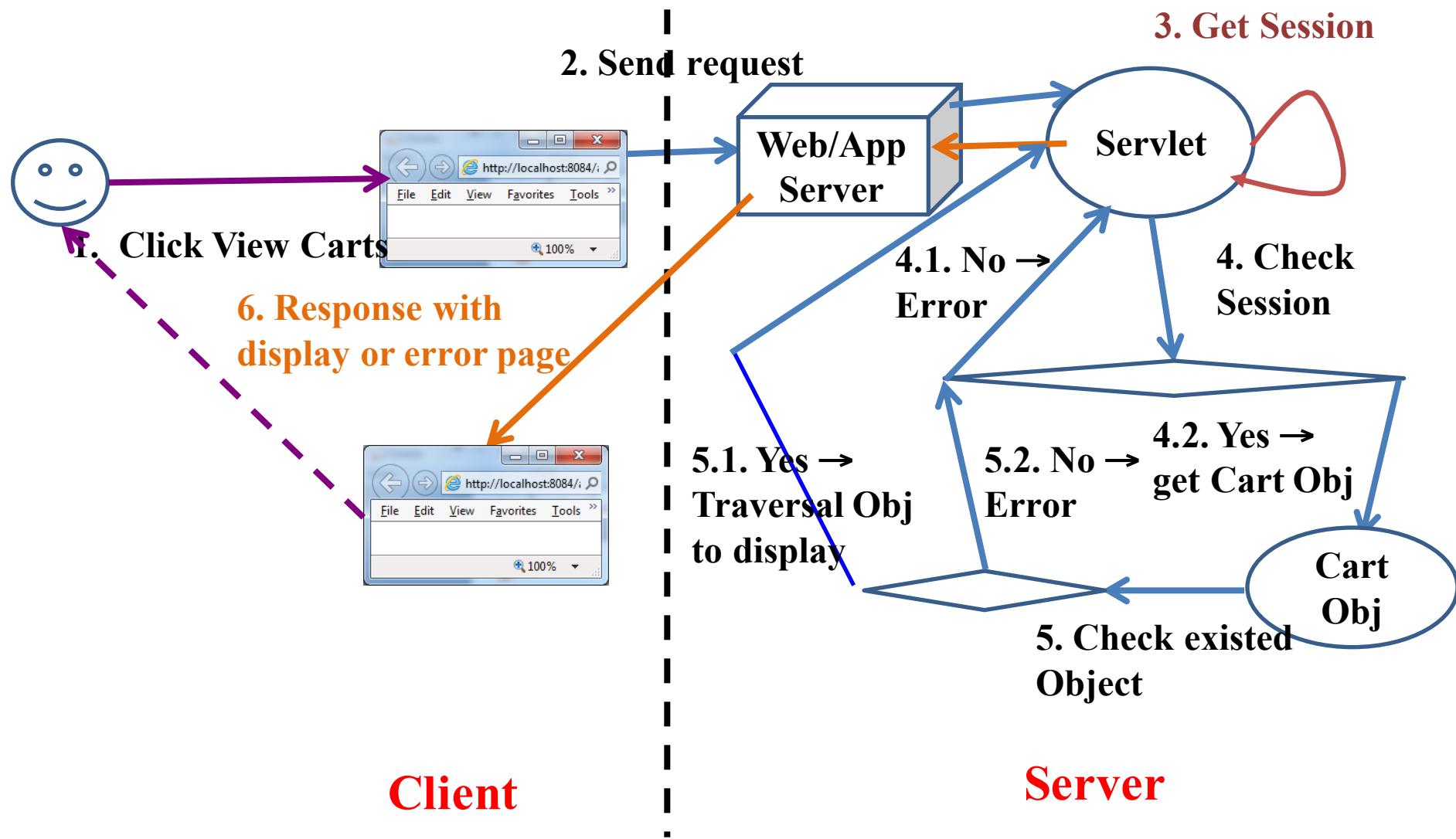
The screenshot shows a web browser window with the URL <http://localhost:8084/MVCDBServlet/ProcessServlet?btAction=View%20Your%20Cart>. The browser's title bar also displays this URL. The main content area is titled "Your Cart Items". It contains a table with the following data:

No.	Title	Quantity	Action
1	JavaEE	1	<input checked="" type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input checked="" type="checkbox"/>

Below the table are two buttons: "Add More Item to Cart" and "Remove".

# How to write CRUD Web Application

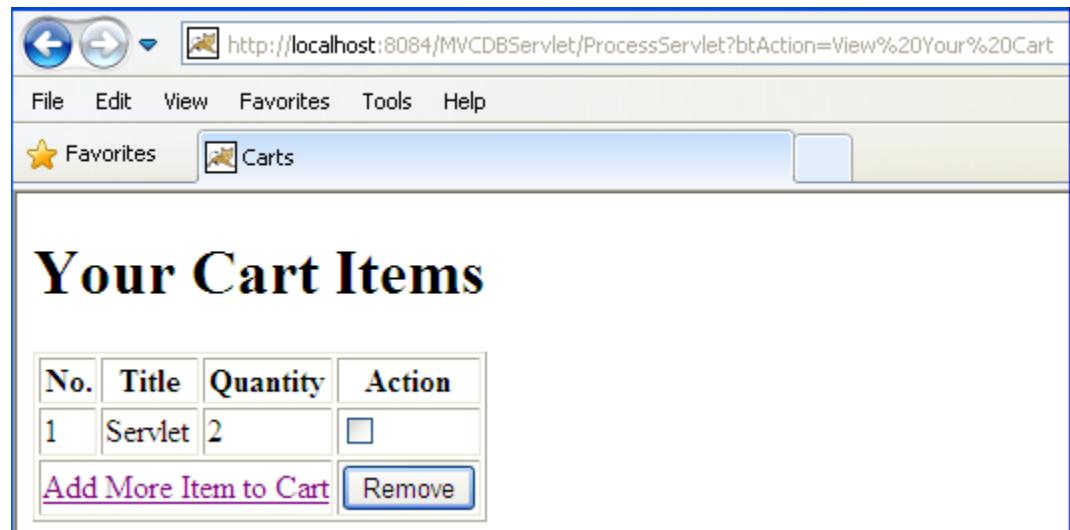
## Interactive Server Model – View Cart



# How to write CRUD Web Application

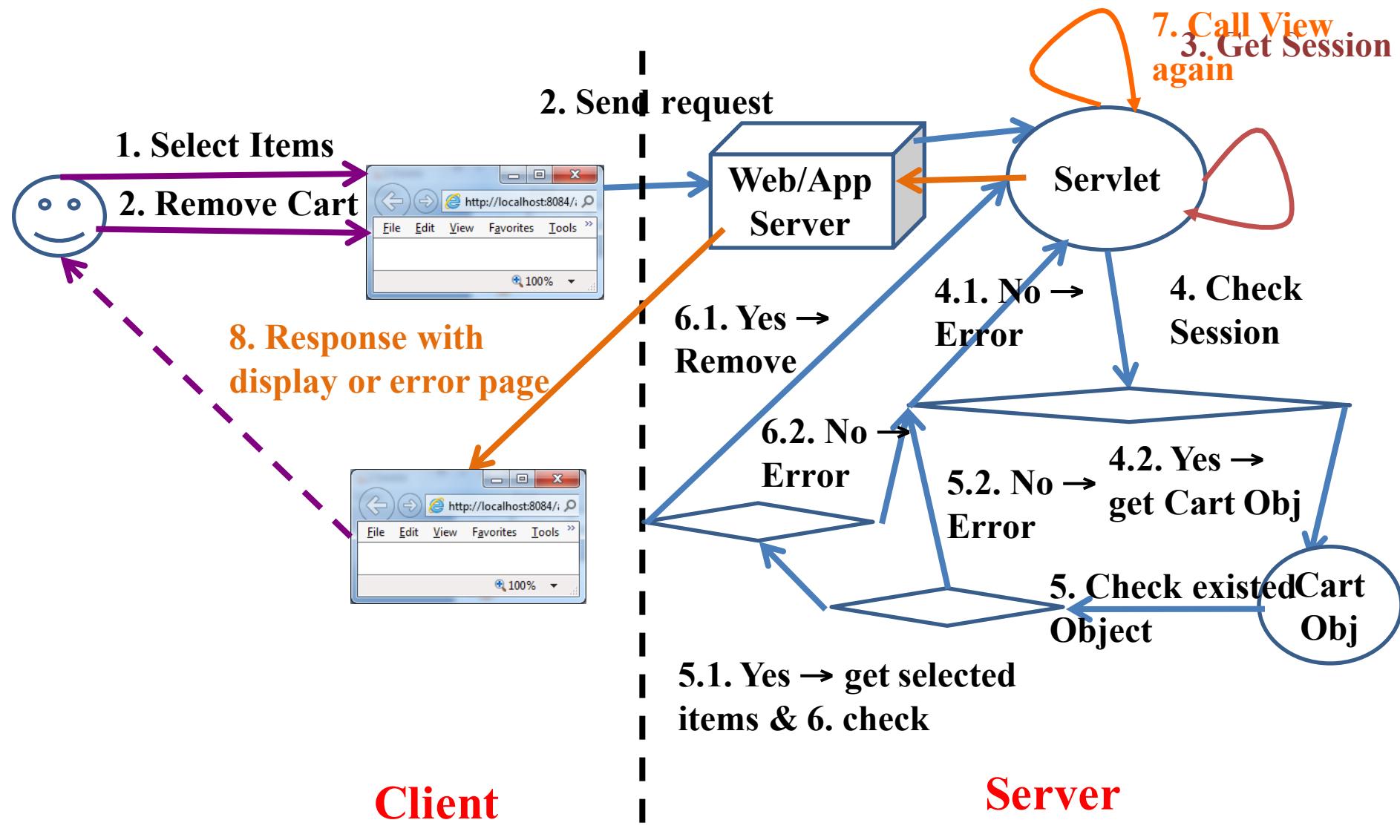
## Shopping Cart – Remove Cart

Your Cart Items			
No.	Title	Quantity	Action
1	JavaEE	1	<input checked="" type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input checked="" type="checkbox"/>
<a href="#">Add More Item to Cart</a>		<a href="#">Remove</a>	



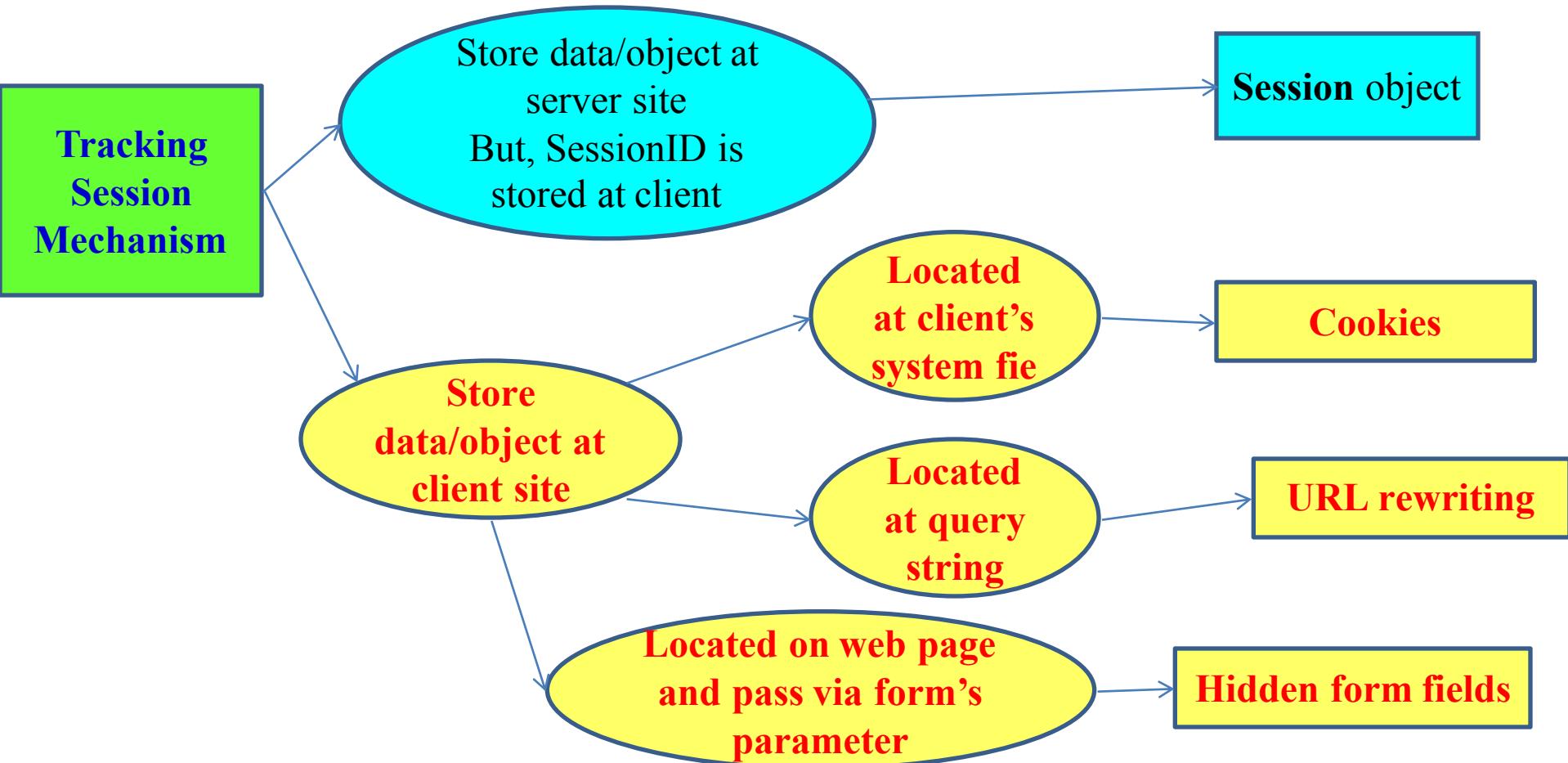
# How to write CRUD Web Application

## Interactive Server Model – Remove



# Sessions & Listeners

## Conclusion



# Error Handling in Servlet

## Reporting Error

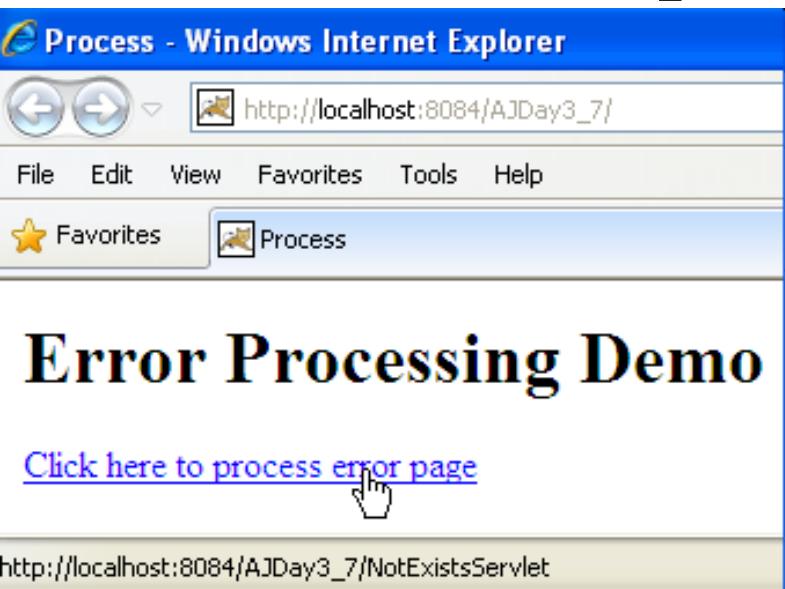
- There are many situations occur an error
  - A requested page may be moved from one location to another.
  - The address may be wrongly typed.
  - The requested page may be forbidden, may be temporarily deleted or correct HTTP version might not have found.
  - There are other situations where an error may generated.
- Error during the execution of a web application are reported

Methods	Descriptions
<b>sendError</b>	<ul style="list-style-type: none"> <li>- <b>public void sendError (int sc) throws IOException</b></li> <li>- Checks for the status code and sends to the user the specified response message</li> <li>- After sending the error message the buffer is cleared</li> <li>- <code>response.sendError(response.SC_NOT_FOUND);</code></li> </ul>
<b>setStatus</b>	<ul style="list-style-type: none"> <li>- <b>public void HttpServletResponse.setStatus (int sc)</b></li> <li>- This code is specified earlier so that on receiving the <code>setStatus()</code> method, the error message is throw. Or redirected to another default Web page</li> <li>- <code>response.setStatus(response.SC_NOT_MODIFIED);</code></li> </ul>

# Error Handling in Servlet

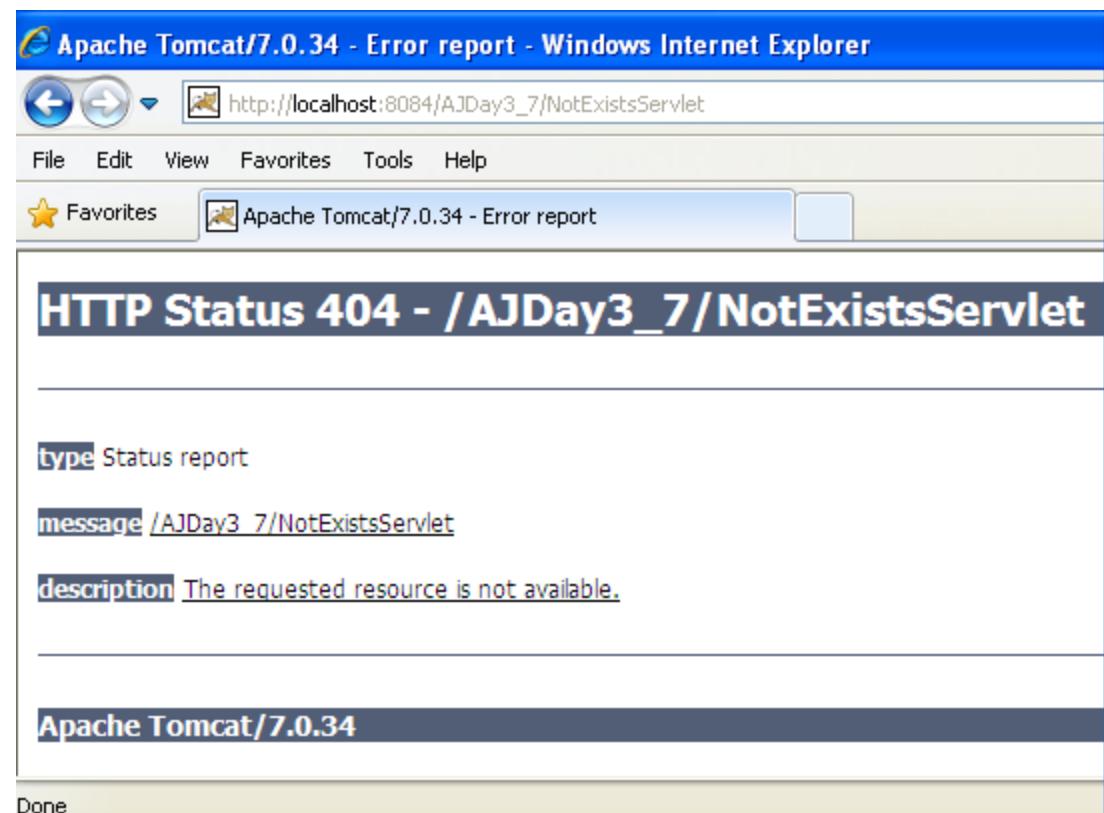
## Reporting Error – Example

Process - Windows Internet Explorer



The screenshot shows a Windows Internet Explorer window with the title "Process - Windows Internet Explorer". The address bar contains "http://localhost:8084/AJDay3\_7/". The menu bar includes File, Edit, View, Favorites, Tools, and Help. Below the menu is a toolbar with a star icon labeled "Favorites" and a process icon labeled "Process". The main content area displays the text "Error Processing Demo" in large bold letters. Below it is a link "Click here to process error page" with a cursor icon pointing at it. At the bottom of the page, the URL "http://localhost:8084/AJDay3\_7/NotExistsServlet" is visible.

Apache Tomcat/7.0.34 - Error report - Windows Internet Explorer

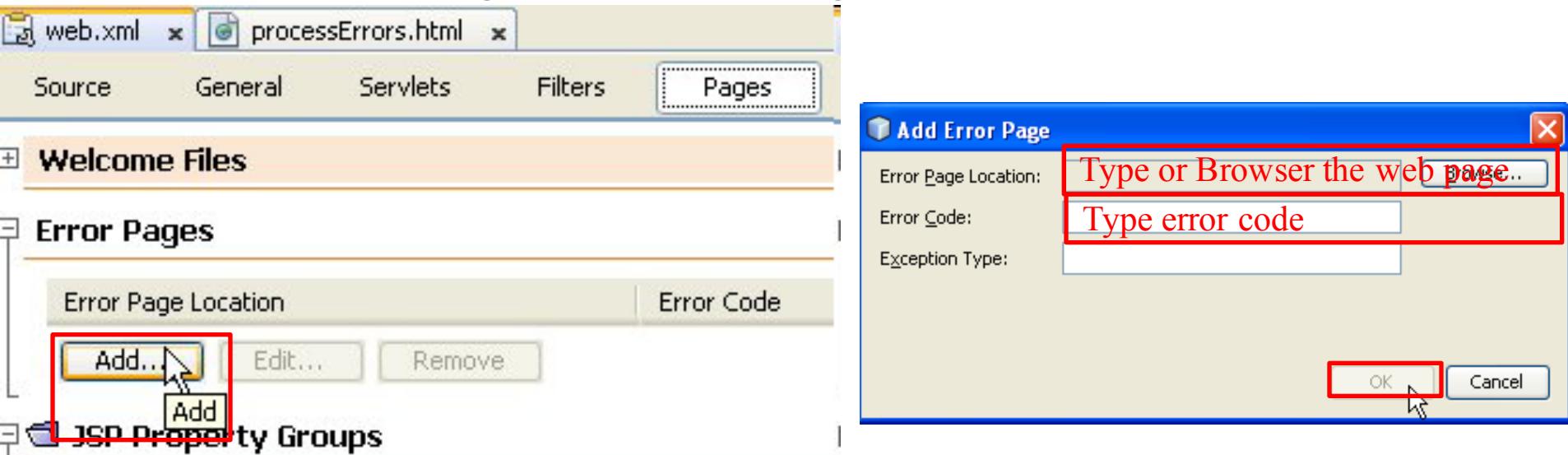


The screenshot shows an Apache Tomcat 7.0.34 error report page. The title bar says "Apache Tomcat/7.0.34 - Error report - Windows Internet Explorer". The address bar shows "http://localhost:8084/AJDay3\_7/NotExistsServlet". The menu bar has File, Edit, View, Favorites, Tools, and Help. A toolbar below the menu includes a star icon for "Favorites" and a process icon for "Apache Tomcat/7.0.34 - Error report". The main content is a large header "HTTP Status 404 - /AJDay3\_7/NotExistsServlet". Below it is a horizontal line. The page then lists three items: "type Status report", "message /AJDay3\_7/NotExistsServlet", and "description The requested resource is not available.". Another horizontal line follows. At the bottom, it says "Apache Tomcat/7.0.34" and "Done".

# Error Handling in Servlet

## Reporting Error – Example

- Addition the following contents to web.xml file
  - In web.xml, choose Page tab, choose Error Pages, click Add



Error Pages	
Error Page Location	Error Code
/FileNotFoundException.html	404
Add...	Edit... (highlighted with a blue box)

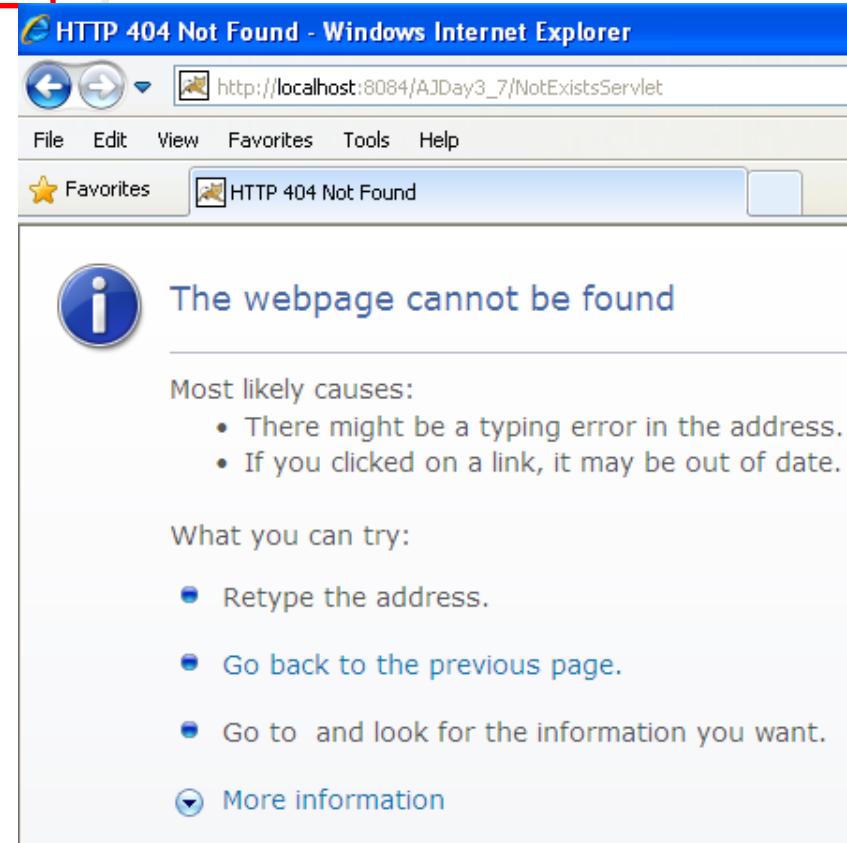
# Error Handling in Servlet

## Reporting Error – Example

web.xml

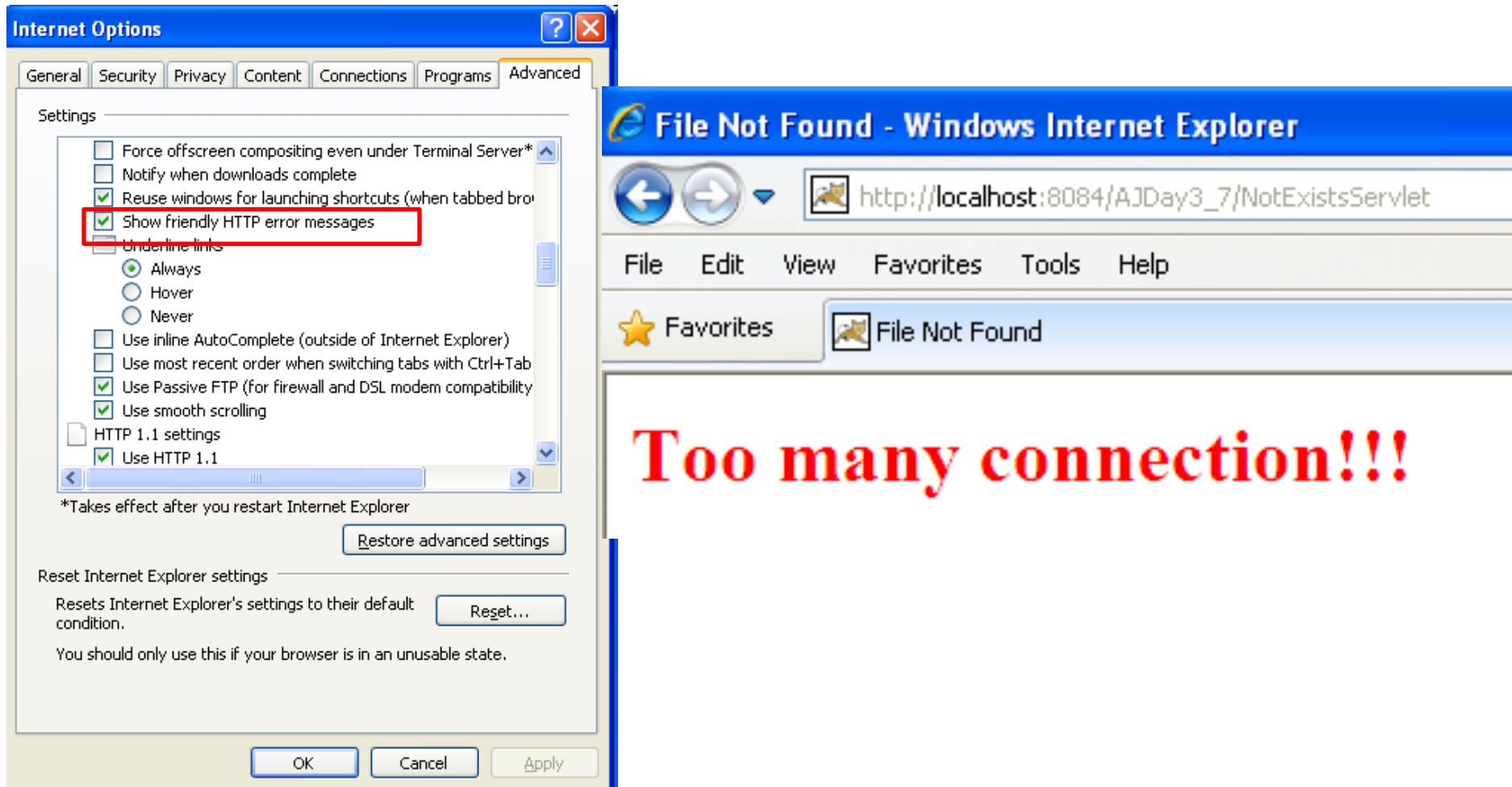
Source General Servlets Filters Pages References Security

```
172 <welcome-file-list>
173     <welcome-file>logine.html</welcome-file>
174 </welcome-file-list>
175 <error-page>
176     <error-code>404</error-code>
177     <location>/fileNotFound.html</location>
178 </error-page>
179 </web-app>
```



# Error Handling in Servlet

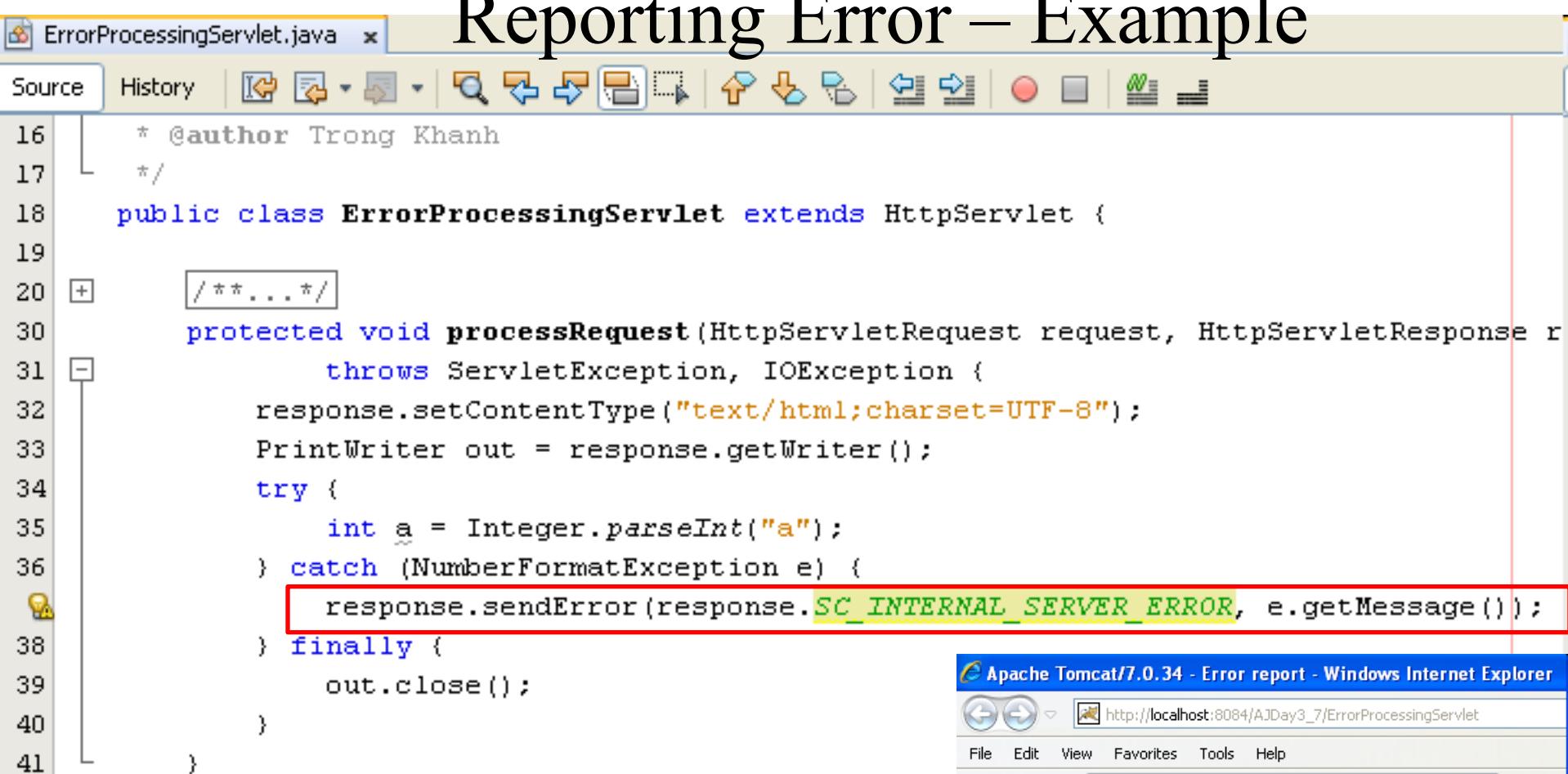
## Reporting Error



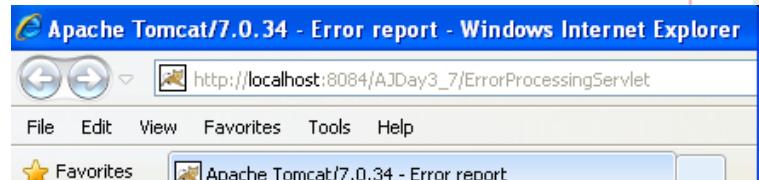
Uncheck the option “Show friendly HTTP error messages” from Tools/ “Internet Options” to set up the browser would be presented the user defined message

# Error Handling in Servlet

## Reporting Error – Example



```
16 * @author Trong Khanh
17 */
18 public class ErrorProcessingServlet extends HttpServlet {
19
20     /**
21      * ...
22      */
23
24     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         response.setContentType("text/html;charset=UTF-8");
27         PrintWriter out = response.getWriter();
28         try {
29             int a = Integer.parseInt("a");
30         } catch (NumberFormatException e) {
31             response.sendError(response.SC_INTERNAL_SERVER_ERROR, e.getMessage());
32         } finally {
33             out.close();
34         }
35     }
36 }
```



HTTP Status 500 - For input string: "a"

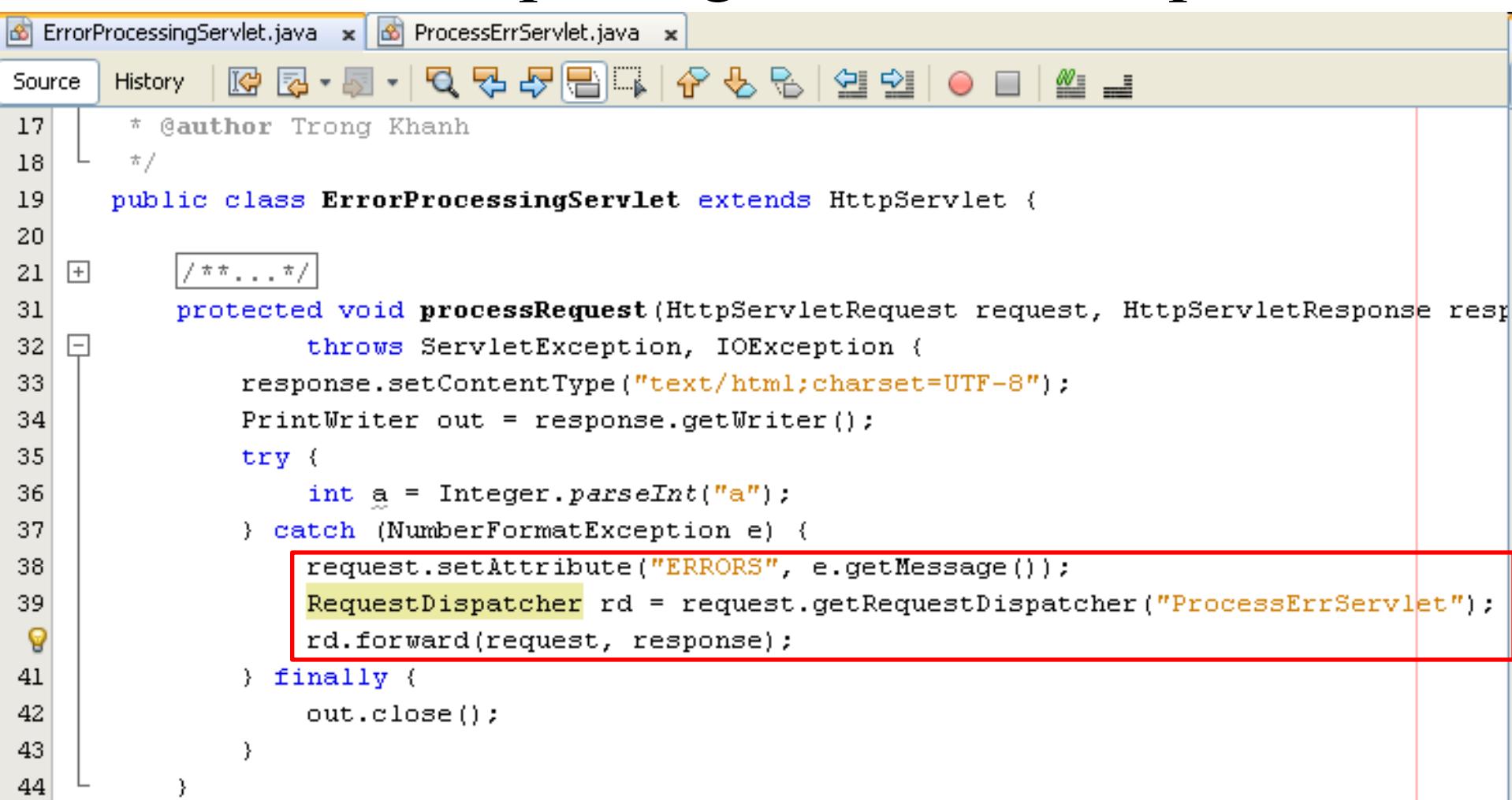
type Status report

message For input string: "a"

description The server encountered an internal error that prevented it from

# Error Handling in Servlet

## Reporting Error – Example



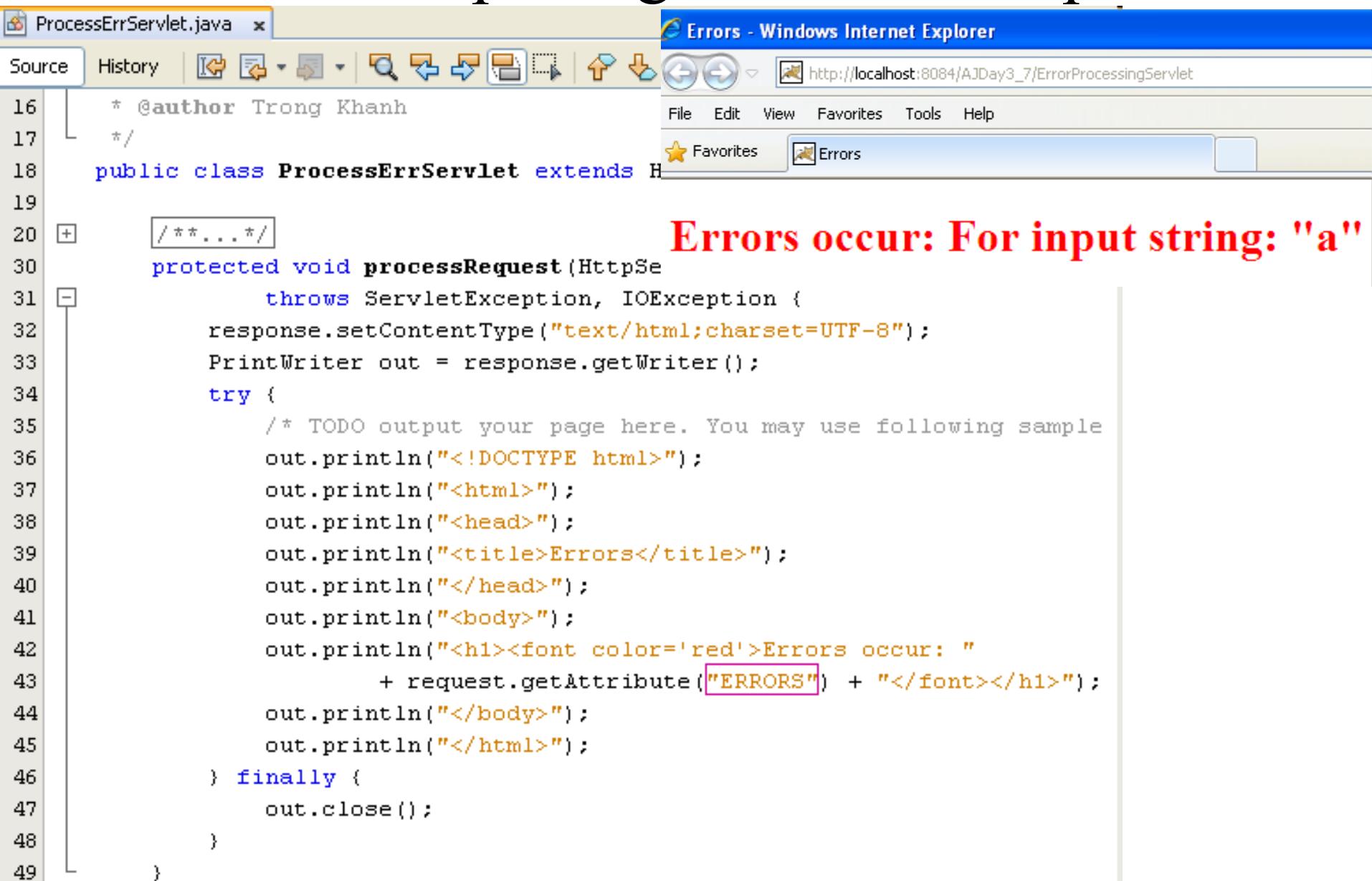
The screenshot shows a Java code editor with two tabs: "ErrorProcessingServlet.java" and "ProcessErrServlet.java". The "ErrorProcessingServlet.java" tab is active, displaying the following code:

```
17 * @author Trong Khanh
18 */
19 public class ErrorProcessingServlet extends HttpServlet {
20
21     /**
22      * ...
23      */
24
25     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
26             throws ServletException, IOException {
27         response.setContentType("text/html;charset=UTF-8");
28         PrintWriter out = response.getWriter();
29         try {
30             int a = Integer.parseInt("a");
31         } catch (NumberFormatException e) {
32             request.setAttribute("ERRORS", e.getMessage());
33             RequestDispatcher rd = request.getRequestDispatcher("ProcessErrServlet");
34             rd.forward(request, response);
35         } finally {
36             out.close();
37         }
38     }
39
40     // ...
41 }
```

The code demonstrates how to handle a NumberFormatException by setting an attribute on the request and then forwarding it to another servlet, "ProcessErrServlet". The line `rd.forward(request, response);` is highlighted with a red rectangle.

# Error Handling in Servlet

## Reporting Error – Example



The screenshot shows a Java IDE interface on the left and a web browser window on the right.

**Java IDE (Left):**

- File: ProcessErrServlet.java
- Toolbar: Source, History, etc.
- Code area:

```
16 * @author Trong Khanh
17 */
18 public class ProcessErrServlet extends H
19
20     /* ... */
21
22     protected void processRequest(HttpServletRequest
23                                     throws ServletException, IOException {
24         response.setContentType("text/html;charset=UTF-8");
25         PrintWriter out = response.getWriter();
26         try {
27             /* TODO output your page here. You may use following sample
28             out.println("<!DOCTYPE html>");
29             out.println("<html>");
30             out.println("<head>");
31             out.println("<title>Errors</title>");
32             out.println("</head>");
33             out.println("<body>");
34             out.println("<h1><font color='red'>Errors occur: "
35                         + request.getAttribute("ERRORS") + "</font></h1>");
36             out.println("</body>");
37             out.println("</html>");
38         } finally {
39             out.close();
40         }
41     }
42 }
```

**Web Browser (Right):**

- Title: Errors - Windows Internet Explorer
- Address bar: http://localhost:8084/AJDay3\_7/ErrorProcessingServlet
- Toolbar: File, Edit, View, Favorites, Tools, Help
- Buttons: Favorites, Errors
- Content: Errors occur: For input string: "a"

# Error Handling in Servlet

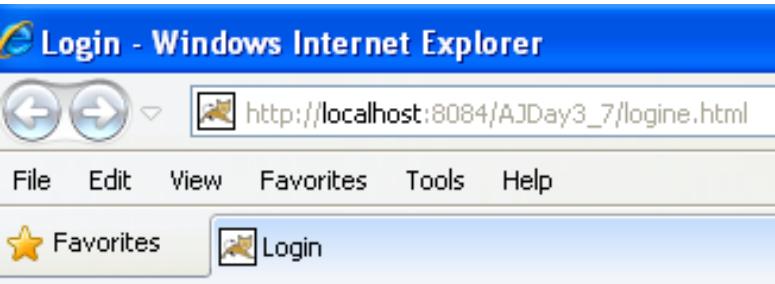
## Reporting Error – Example

The screenshot shows a Java IDE interface with the title "Reporting Error – Example". The current file is "DisplayErrorServlet.java". The code implements a servlet that handles requests and performs a temporary redirect if the parameter "action" is null. The code is annotated with comments and icons.

```
15
16     * @author Trong Khanh
17     */
18
19     public class DisplayErrorServlet extends HttpServlet {
20
21         /**
22
23         * @param request the servlet request
24         * @param response the servlet response
25         * @throws ServletException if an error occurs
26         * @throws IOException if an error occurs
27         */
28
29         protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30             throws ServletException, IOException {
31
32             response.setContentType("text/html;charset=UTF-8");
33             PrintWriter out = response.getWriter();
34
35             try {
36                 String action = request.getParameter("action");
37                 if (action == null) {
38                     response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);
39                     String url = "http://" + request.getLocalName() + ":"
40                         + request.getLocalPort() + request.getContextPath() +
41                         "/logine.html";
42                     response.setHeader("Location", url);
43                 }
44             } finally {
45                 out.close();
46             }
47         }
48
49         // ...
50     }
```

# Error Handling in Servlet

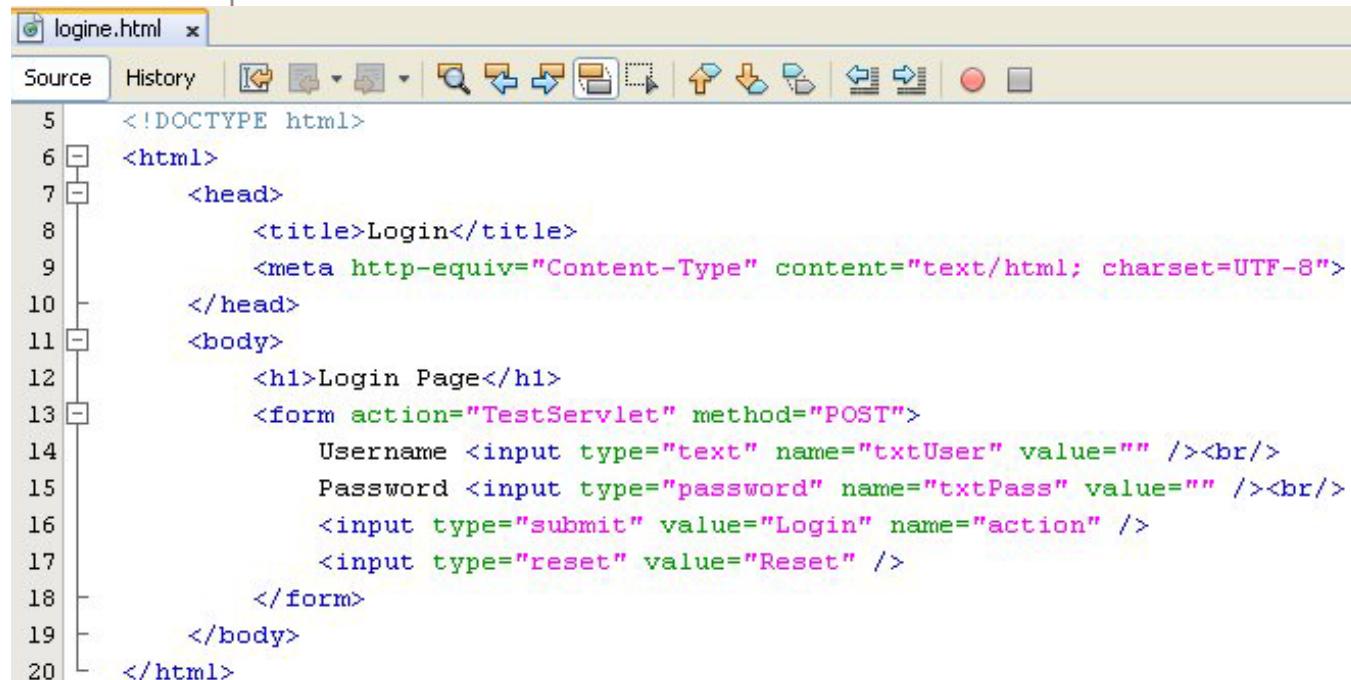
## Reporting Error – Example



## Login Page

Username

Password



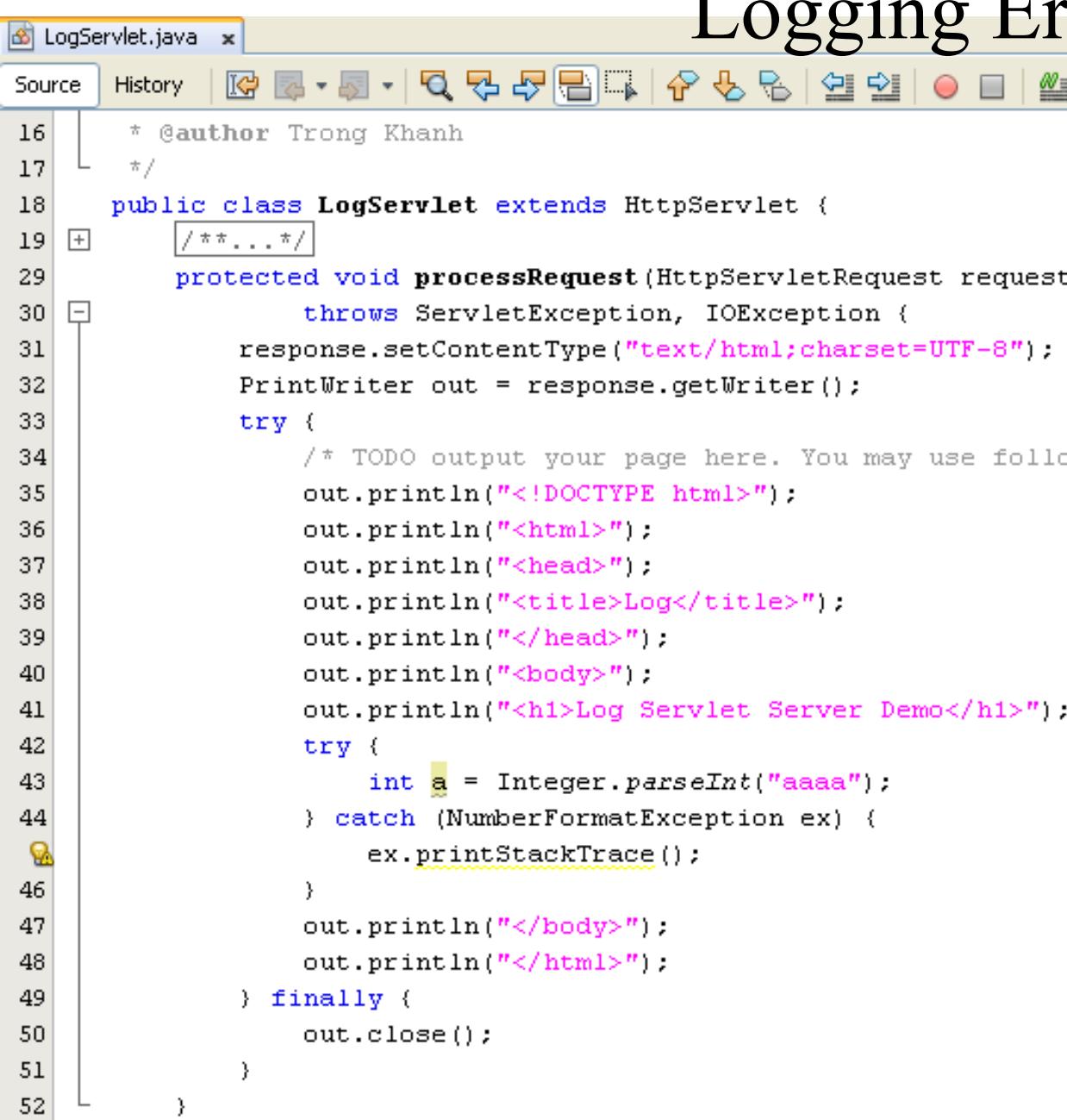
```
5 <!DOCTYPE html>
6 <html>
7   <head>
8     <title>Login</title>
9     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10    </head>
11    <body>
12      <h1>Login Page</h1>
13      <form action="TestServlet" method="POST">
14        Username <input type="text" name="txtUser" value="" /><br/>
15        Password <input type="password" name="txtPass" value="" /><br/>
16        <input type="submit" value="Login" name="action" />
17        <input type="reset" value="Reset" />
18      </form>
19    </body>
20  </html>
```

# Error Handling in Servlet

## Reporting Error – Example – Others

# Error Handling in Servlet

## Logging Error



The screenshot shows a Java code editor with the file `LogServlet.java` open. The code defines a servlet named `LogServlet` that extends `HttpServlet`. It overrides the `processRequest` method to handle HTTP requests. The code includes a try-catch block where an attempt is made to parse a string "aaaa" into an integer. If a `NumberFormatException` occurs, the stack trace is printed to the output stream. The rest of the code outputs a simple HTML page with a title and body.

```
16     * @author Trong Khanh
17     */
18     public class LogServlet extends HttpServlet {
19         /**
20          */
21         protected void processRequest(HttpServletRequest request
22             throws ServletException, IOException {
23             response.setContentType("text/html;charset=UTF-8");
24             PrintWriter out = response.getWriter();
25             try {
26                 /* TODO output your page here. You may use following
27                  out.println("<!DOCTYPE html>");
28                  out.println("<html>");
29                  out.println("<head>");
30                  out.println("<title>Log</title>");
31                  out.println("</head>");
32                  out.println("<body>");
33                  out.println("<h1>Log Servlet Server Demo</h1>");
34                  try {
35                      int a = Integer.parseInt("aaaa");
36                  } catch (NumberFormatException ex) {
37                      ex.printStackTrace();
38                  }
39                  out.println("</body>");
40                  out.println("</html>");
41             } finally {
42                 out.close();
43             }
44         }
45     }
```

# Error Handling in Servlet

## Logging Error

The screenshot shows a Java development environment with a code editor and a browser window.

**Code Editor (LogServlet.java):**

```
16 * @author Trong Khanh
17 */
18 public class LogServlet extends HttpServlet {
19     /**
20      */
21     protected void processRequest(HttpServletRequest request
22             throws ServletException, IOException {
23         response.setContentType("text/html;charset=UTF-8");
24         PrintWriter out = response.getWriter();
25         try {
26             /* TODO output your page here. You may use following
27              out.println("<!DOCTYPE html>");
28              out.println("<html>");
29              out.println("<head>");
30              out.println("<title>Log</title>");
31              out.println("</head>");
32              out.println("<body>");
33              out.println("<h1>Log Servlet Server Demo</h1>");
34              try {
35                  int a = Integer.parseInt("aaaa");
36              } catch (NumberFormatException ex) {
37                  ex.printStackTrace();
38              }
39          } finally {
40              out.close();
41      }
42  }
```

**Browser Window (Log - Windows Internet Explorer):**

http://localhost:8084/AJDay3\_7/LogServlet

**Log Servlet Server Demo**

**Java Output:**

```
INFO: Reloading Context with name [/AJDay3_7] is completed
java.lang.NumberFormatException: For input string: "aaaa"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:492)
        at java.lang.Integer.parseInt(Integer.java:527)
        at sample.servlet.LogServlet.processRequest(LogServlet.java:43)
        at sample.servlet.LogServlet.doGet(LogServlet.java:67)
```

# Error Handling in Servlet

## Logging Error

- Servlet can store the actions and errors through the log() method of the **GenericServlet** class.
- The log() method also assists in debugging and can viewed record in a server
- **Syntax:** **public void log (String msg [, Throwable t])**
- **Ex:**

...

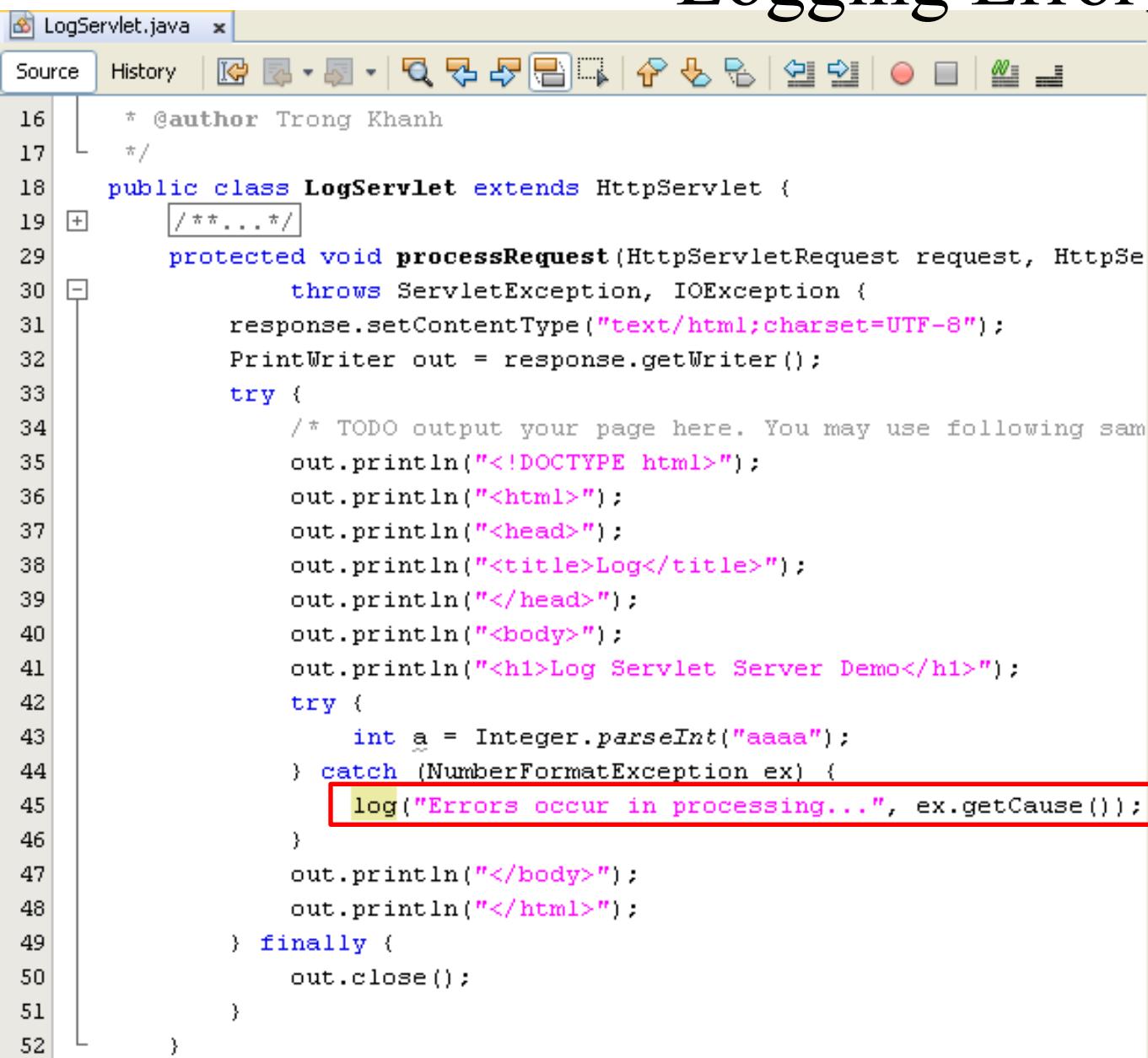
```
log("Servlet is not found ");
response.sendError(response.SC_INTERNAL_SERVER_ERROR, "The requested
page ["+ page + "] not found.");
```

...

- **A log file locate at**
  - C:\Documents and Settings\LoggedUser\Application Data\NetBeans\7.4\apache-tomcat-7.0.41.0\_base\logs\localhost.yyyy-mm-dd.log
  - C:\Users\LoggedUser\AppData\Roaming\NetBeans\7.4\apache-tomcat-7.0.41.0\_base\work\Catalina\logs\localhost.yyyy-mm-dd.log

# Error Handling in Servlet

## Logging Error



The screenshot shows a Java code editor with the file `LogServlet.java` open. The code implements a `HttpServlet` to log processing errors. A red box highlights the line `log("Errors occur in processing...", ex.getCause());`.

```
16 * @author Trong Khanh
17 */
18 public class LogServlet extends HttpServlet {
19     /**
20      */
21     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
22         throws ServletException, IOException {
23         response.setContentType("text/html;charset=UTF-8");
24         PrintWriter out = response.getWriter();
25         try {
26             /* TODO output your page here. You may use following sample code */
27             out.println("<!DOCTYPE html>");
28             out.println("<html>");
29             out.println("<head>");
30             out.println("<title>Log</title>");
31             out.println("</head>");
32             out.println("<body>");
33             out.println("<h1>Log Servlet Server Demo</h1>");
34             try {
35                 int a = Integer.parseInt("aaaa");
36             } catch (NumberFormatException ex) {
37                 log("Errors occur in processing...", ex.getCause());
38             }
39             out.println("</body>");
40             out.println("</html>");
41         } finally {
42             out.close();
43         }
44     }
45 }
```

# Error Handling in Servlet

## Logging Error

### Output

```
Apache Tomcat 7.0.27.0 x Apache Tomcat 7.0.27.0 Log x AJDay3_7 (run-deploy) x
thg 10 30, 2013 8:22:17 SA org.apache.catalina.core.ApplicationContext log
SEVERE: LogServlet: Errors occur in processing...
```

Lister - [c:\Documents and Settings\Trong Khanh\Application Data\NetBeans\7.2.1\apache-tomcat-7.0.27.0\_base\logs\localhost.2013-10-30.log]

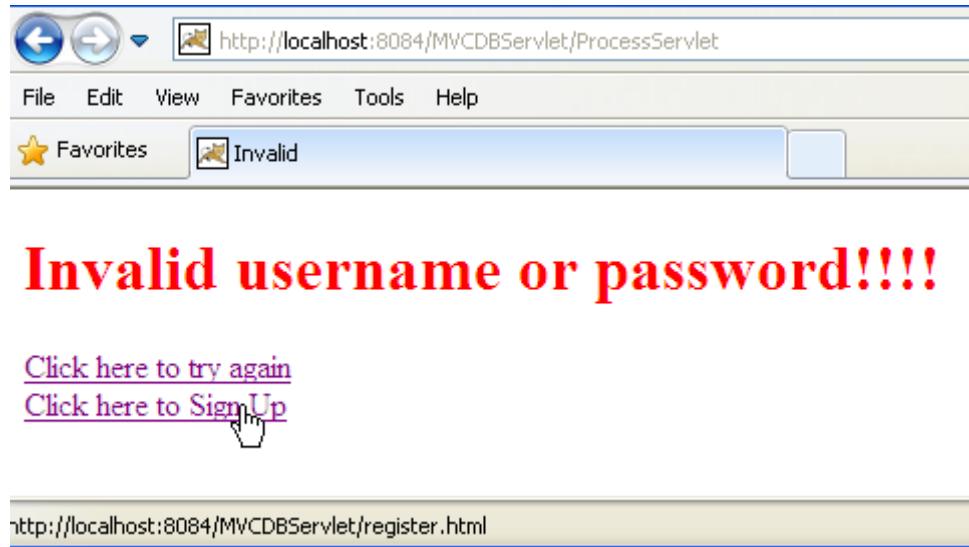
File Edit Options Help

```
thg 10 30, 2013 8:22:17 SA org.apache.catalina.core.ApplicationContext log
SEVERE: LogServlet: Errors occur in processing...
```

Name	Ext	Size	Date
[..]	<DIR>		30/10/2013 08:23
localhost_access_log.2013-10-30	txt	127	30/10/2013 08:23
localhost.2013-10-30	log	1.166	30/10/2013 08:22
manager.2013-10-30	log	10.655	30/10/2013 08:22
catalina.2013-10-30	log	6.253	30/10/2013 08:21
host-manager.2013-10-30	log	0	30/10/2013 07:54

# How to write CRUD Web Application

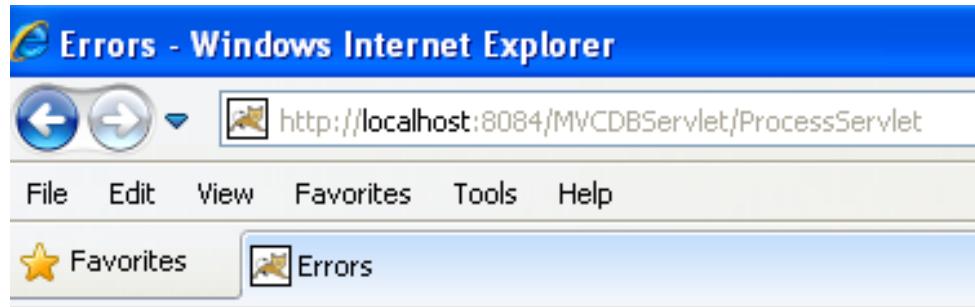
## Register Functions



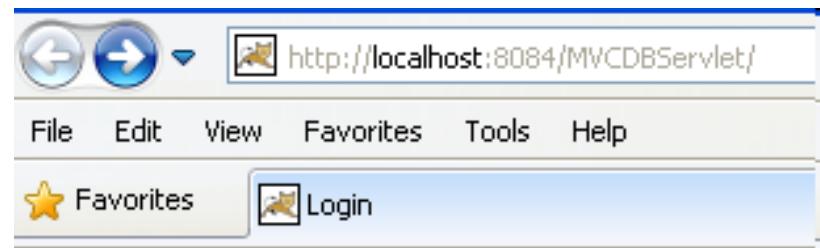
A screenshot of a web browser window showing the "Register Page". The address bar shows the URL <http://localhost:8084/MVCDBServlet/register.html>. The page title is **Register Page**. It contains four input fields with validation requirements: **Username\*** (6 - 12 chars), **Password\*** (8 - 20 chars), **Confirm\***, and **Full name\*** (2 - 40 chars). Below the input fields are two buttons: **Register** and **Reset**. At the bottom of the page, there is a link labeled **Done**.

# How to write CRUD Web Application

## Validation



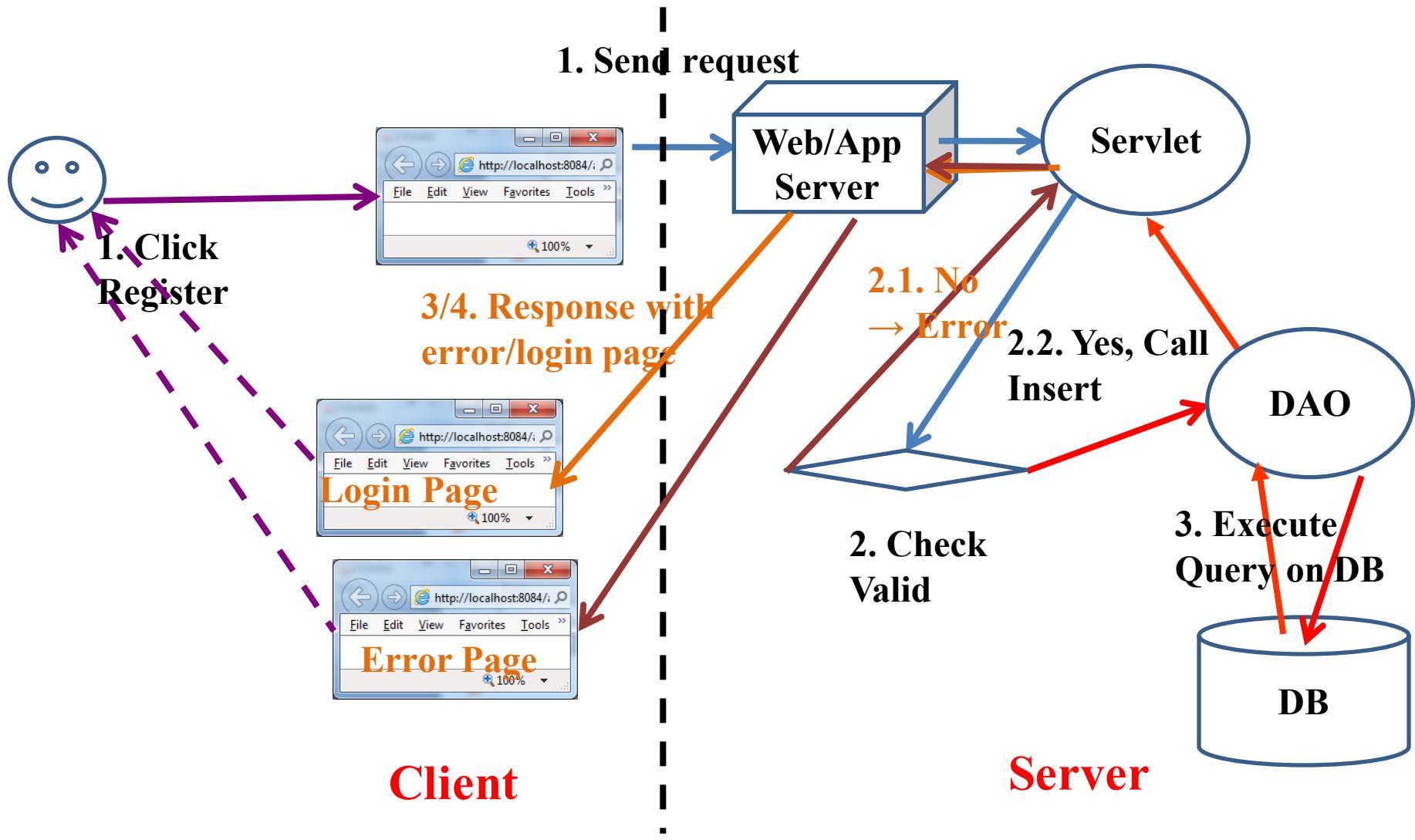
### Errors occur



### Login Page

# How to write CRUD Web Application

## Interactive Server Model



# Summary

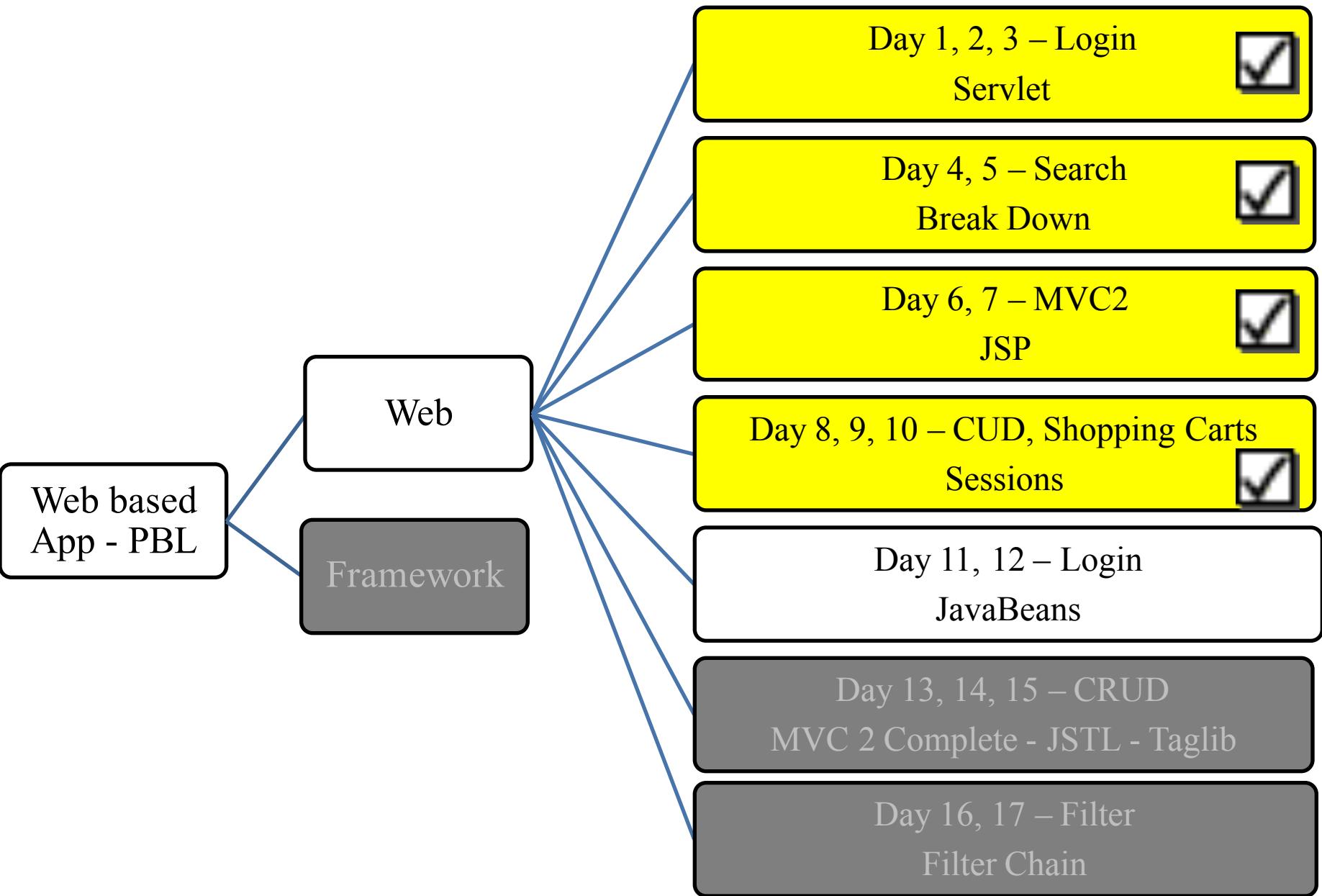
- **How to write CRUD Web Application**
  - Session Tracking Techniques
  - Manipulate DB Techniques in Web Application
  - Break down structure component in building web application
- **Techniques: Error Handling in Servlets**
  - Reporting Errors
  - Logging Errors
  - Users Errors vs. System Errors

Q&A

# Next Lecture

- **JSP Standard Actions**
  - JavaBeans
  - Standard Actions
- **Dispatcher Mechanism**
  - Including, Forwarding, and Parameters
  - Vs. Dispatcher in Servlets
- **EL – Expression Languages**
  - What is EL?
  - How to use EL in JSP?

# Next Lecture



# Appendix

## Shopping Cart using Cookies

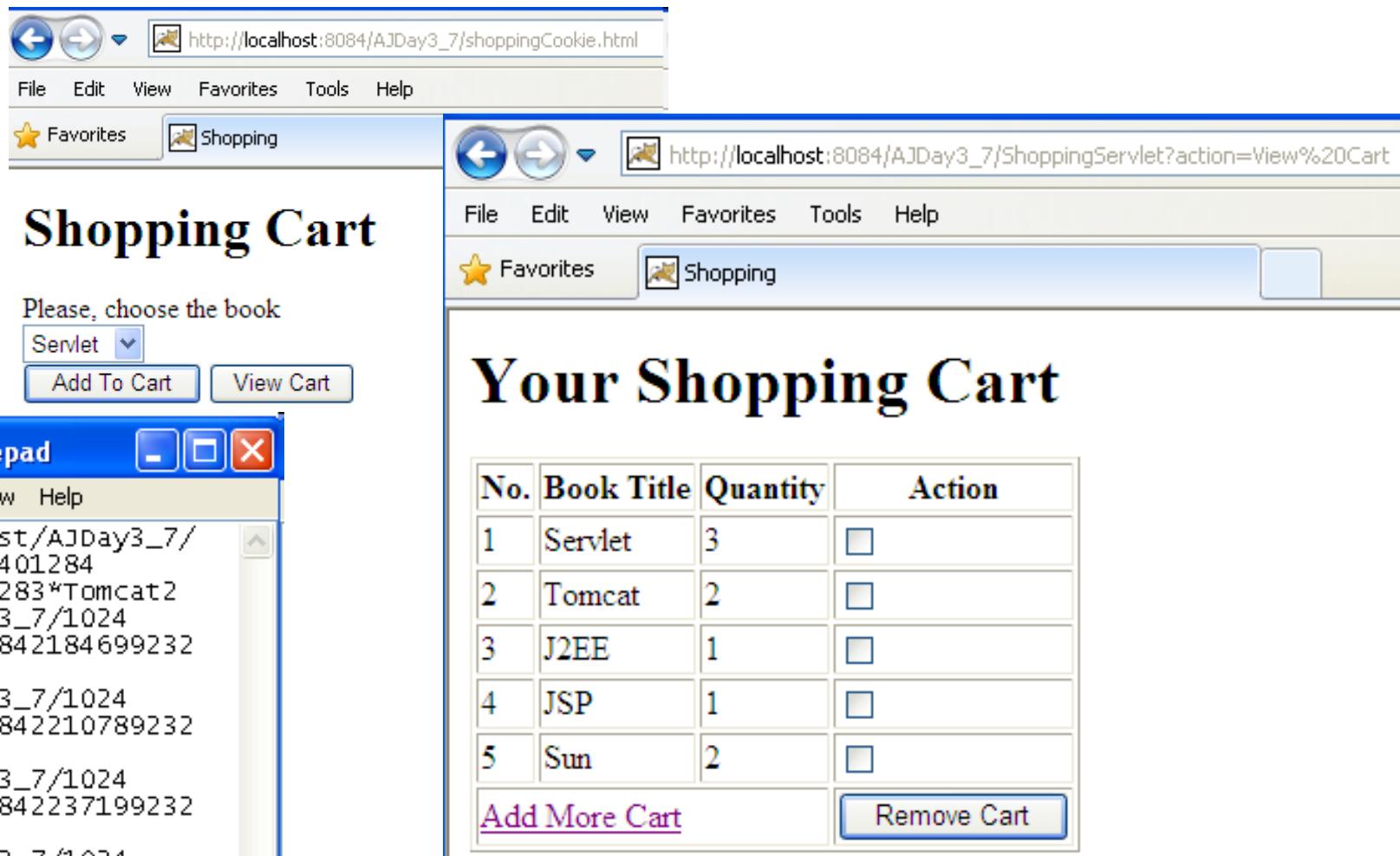
**Shopping Cart**

Please, choose the book  
 Servlet

**Your Shopping Cart**

No.	Book Title	Quantity	Action
1	Servlet	3	<input type="button" value=""/>
2	Tomcat	2	<input type="button" value=""/>
3	J2EE	1	<input type="button" value=""/>
4	JSP	1	<input type="button" value=""/>
5	Sun	2	<input type="button" value=""/>

[Add More Cart](#) [Remove Cart](#)



K7D2ZFKD - Notepad

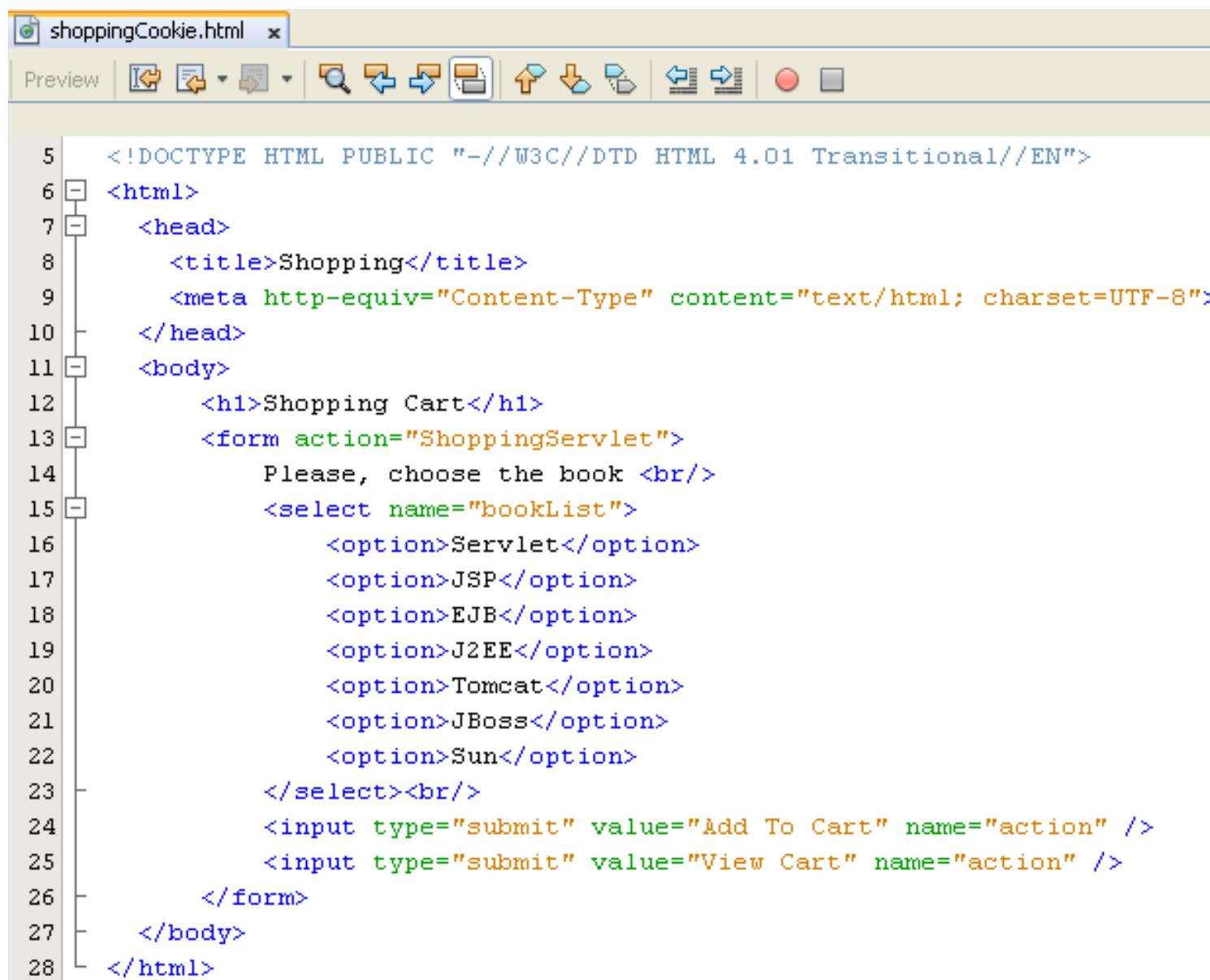
```

File Edit Format View Help
Servlet3localhost/AJDay3_7/
102480359193630401284
210703923230401283*Tomcat2
localhost/AJDay3_7/1024
883591936304012842184699232
30401283*J2EE1
localhost/AJDay3_7/1024
913591936304012842210789232
30401283*JSP1
localhost/AJDay3_7/1024
933591936304012842237199232
30401283*Sun2
localhost/AJDay3_7/1024
1003591936304012842303289232
30401283*JBoss0
localhost/AJDay3_7/1024
1083591936304012842379539232
30401283*

```

# Appendix

## Shopping Cart using Cookies

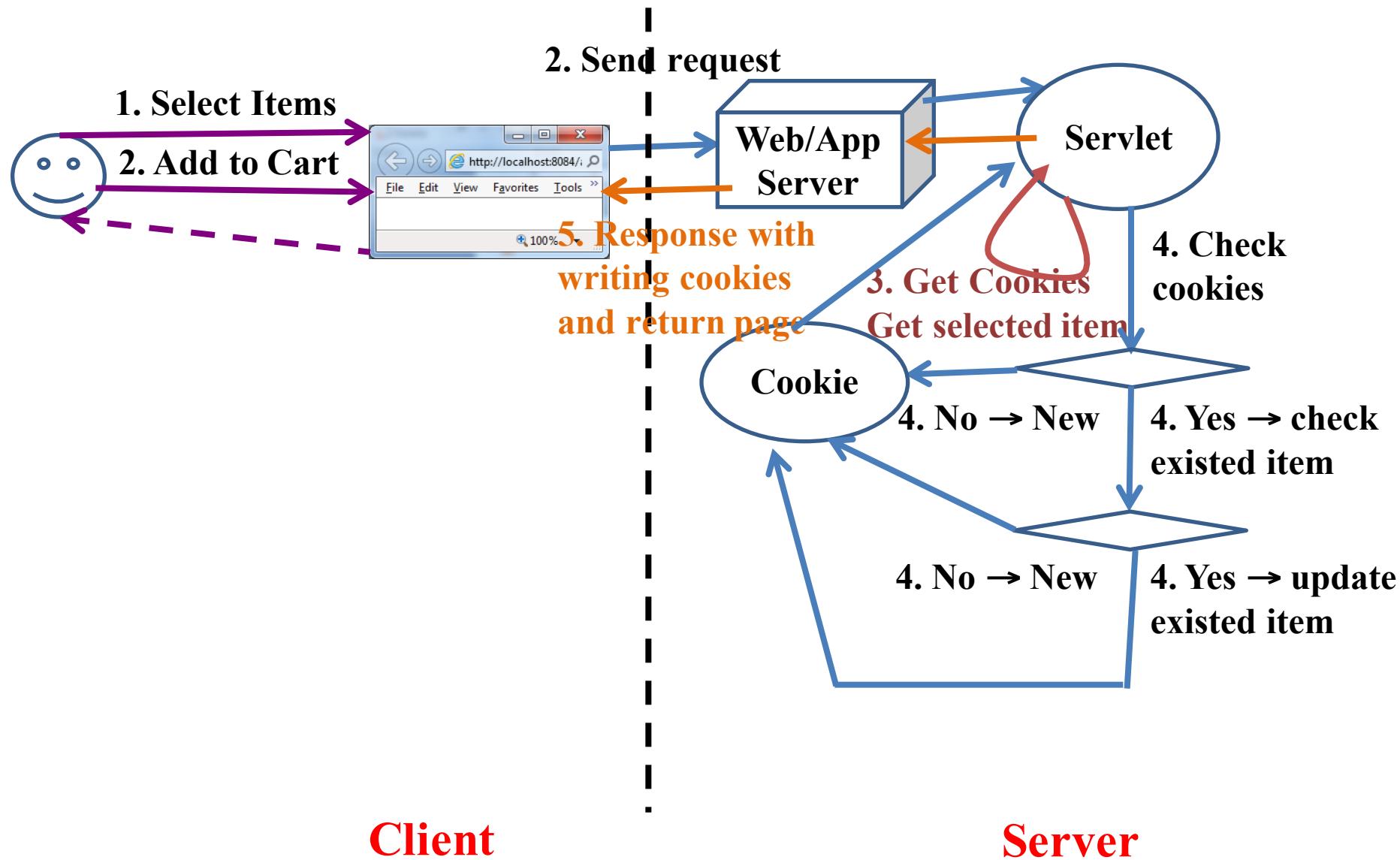


The screenshot shows a code editor window titled "shoppingCookie.html". The window has a toolbar with various icons for file operations like Open, Save, and Print. The main area displays an HTML document with the following code:

```
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
6  <html>
7      <head>
8          <title>Shopping</title>
9          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10     </head>
11     <body>
12         <h1>Shopping Cart</h1>
13         <form action="ShoppingServlet">
14             Please, choose the book <br/>
15             <select name="bookList">
16                 <option>Servlet</option>
17                 <option>JSP</option>
18                 <option>EJB</option>
19                 <option>J2EE</option>
20                 <option>Tomcat</option>
21                 <option>JBoss</option>
22                 <option>Sun</option>
23             </select><br/>
24             <input type="submit" value="Add To Cart" name="action" />
25             <input type="submit" value="View Cart" name="action" />
26         </form>
27     </body>
28 </html>
```

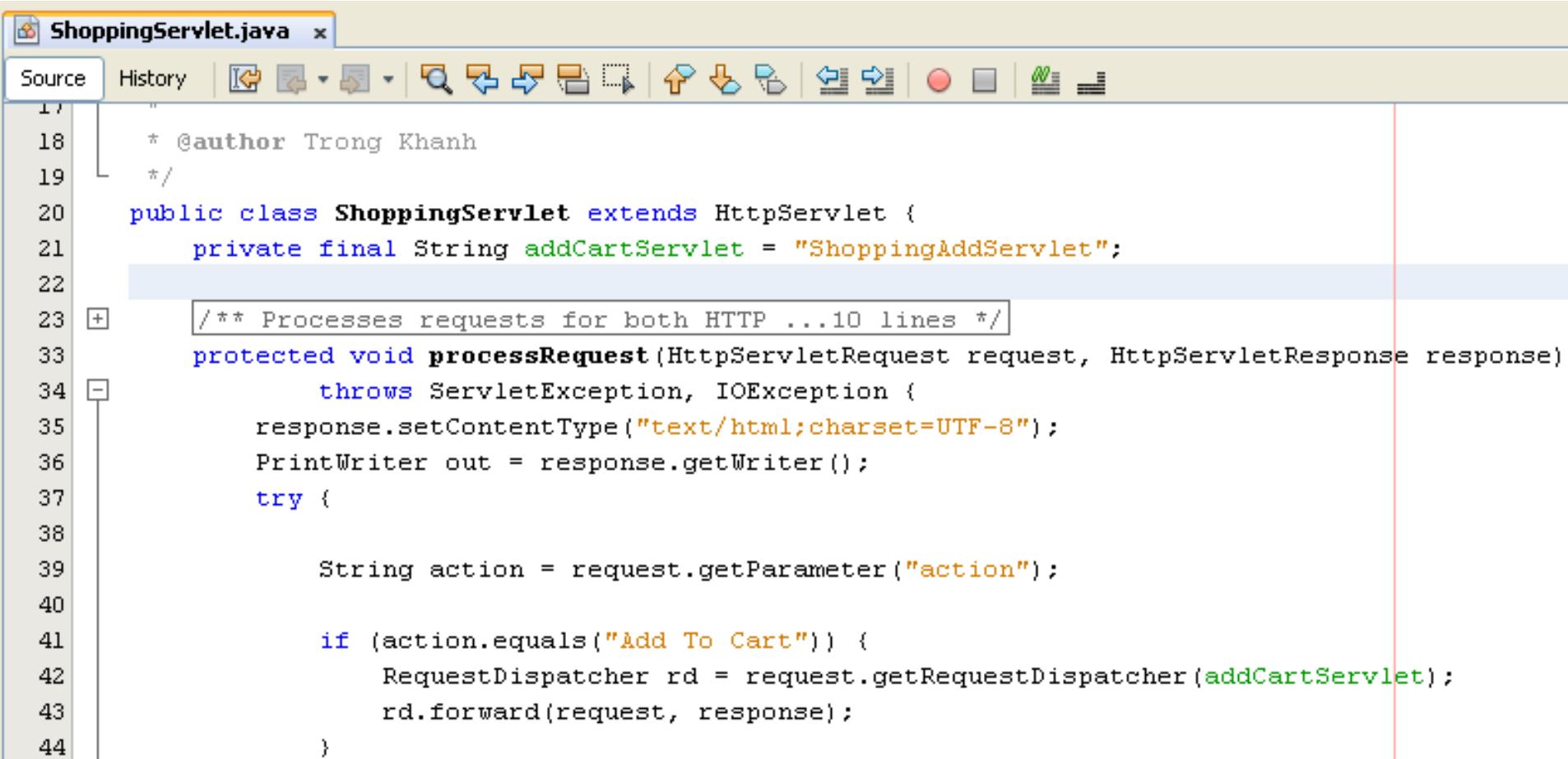
# Appendix

## Interactive Server Model



# Appendix

## Shopping Cart using Cookies

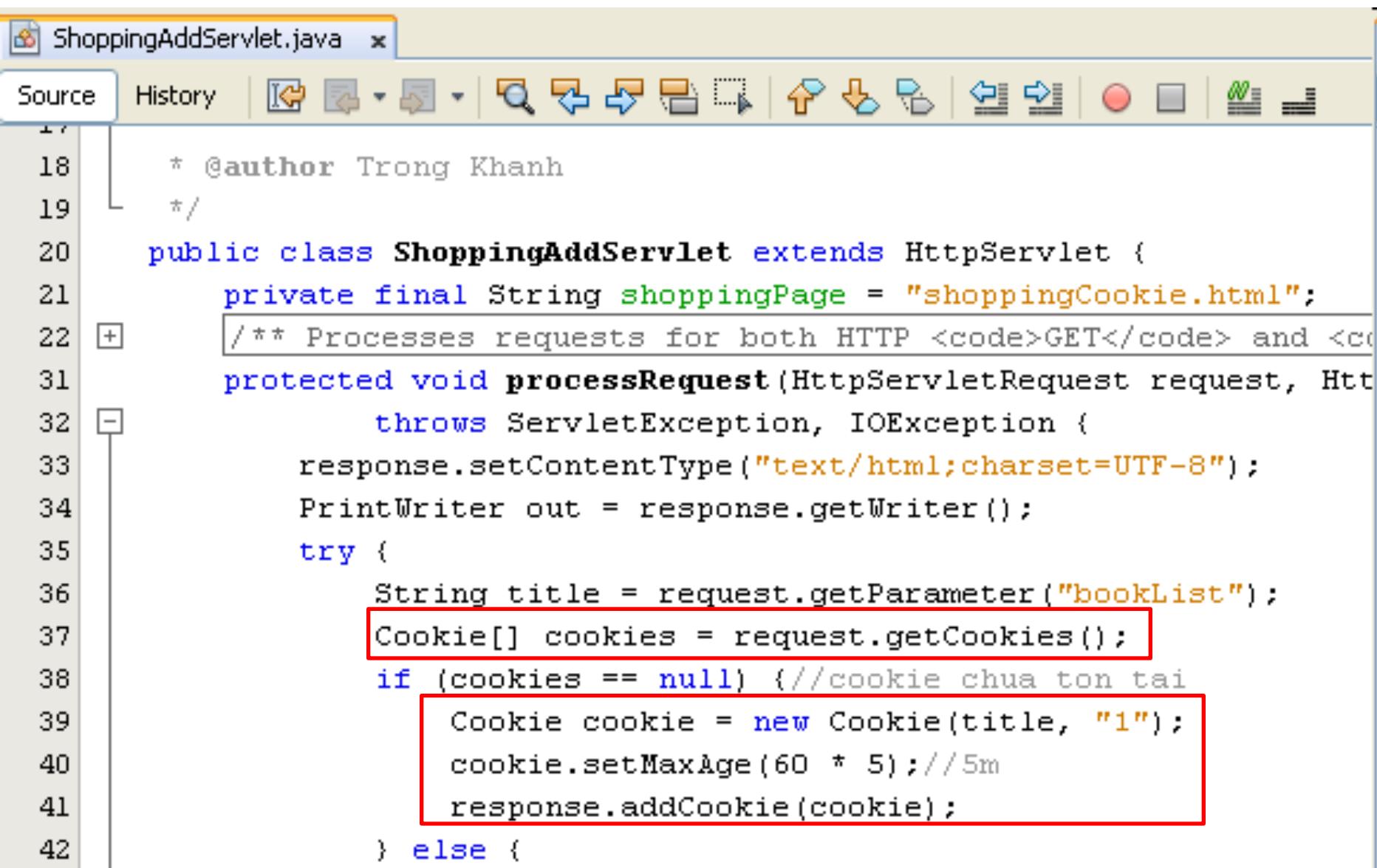


The screenshot shows a Java code editor window titled "ShoppingServlet.java". The code implements a HttpServlet for a shopping cart application. It includes a constructor with a postamble, a protected processRequest method, and a main try-catch block. The processRequest method checks the "action" parameter and forwards the request to the "ShoppingAddServlet" if the action is "Add To Cart".

```
17
18     * @author Trong Khanh
19     */
20
21     public class ShoppingServlet extends HttpServlet {
22         private final String addCartServlet = "ShoppingAddServlet";
23
24         /**
25          Processes requests for both HTTP ...
26          ...10 lines */
27         protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28             throws ServletException, IOException {
29             response.setContentType("text/html;charset=UTF-8");
30             PrintWriter out = response.getWriter();
31             try {
32
33                 String action = request.getParameter("action");
34
35                 if (action.equals("Add To Cart")) {
36                     RequestDispatcher rd = request.getRequestDispatcher(addCartServlet);
37                     rd.forward(request, response);
38                 }
39             } catch (Exception e) {
40                 e.printStackTrace();
41             }
42         }
43
44     }
```

# Appendix

## Shopping Cart using Cookies



The screenshot shows a Java code editor with the file `ShoppingAddServlet.java` open. The code implements a shopping cart using cookies. It defines a class `ShoppingAddServlet` that extends `HttpServlet`. The `processRequest` method retrieves a parameter `bookList` from the request, gets the cookies, creates a new cookie if it's null, and adds it to the response. The code is annotated with comments and imports.

```
18 * @author Trong Khanh
19 */
20 public class ShoppingAddServlet extends HttpServlet {
21     private final String shoppingPage = "shoppingCookie.html";
22     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
23      * methods.
24      *
25      * @param request the <code>HttpServletRequest</code> object
26      * @param response the <code>HttpServletResponse</code> object
27      * @throws ServletException if a servlet-specific error occurs
28      * @throws IOException if an I/O error occurs
29      */
30     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31             throws ServletException, IOException {
32         response.setContentType("text/html;charset=UTF-8");
33         PrintWriter out = response.getWriter();
34         try {
35             String title = request.getParameter("bookList");
36             Cookie[] cookies = request.getCookies();
37             if (cookies == null) { //cookie chua ton tai
38                 Cookie cookie = new Cookie(title, "1");
39                 cookie.setMaxAge(60 * 5); //5m
40                 response.addCookie(cookie);
41             } else {
42             }
43         } finally {
44             out.close();
45         }
46     }
47     // ...
48 }
```

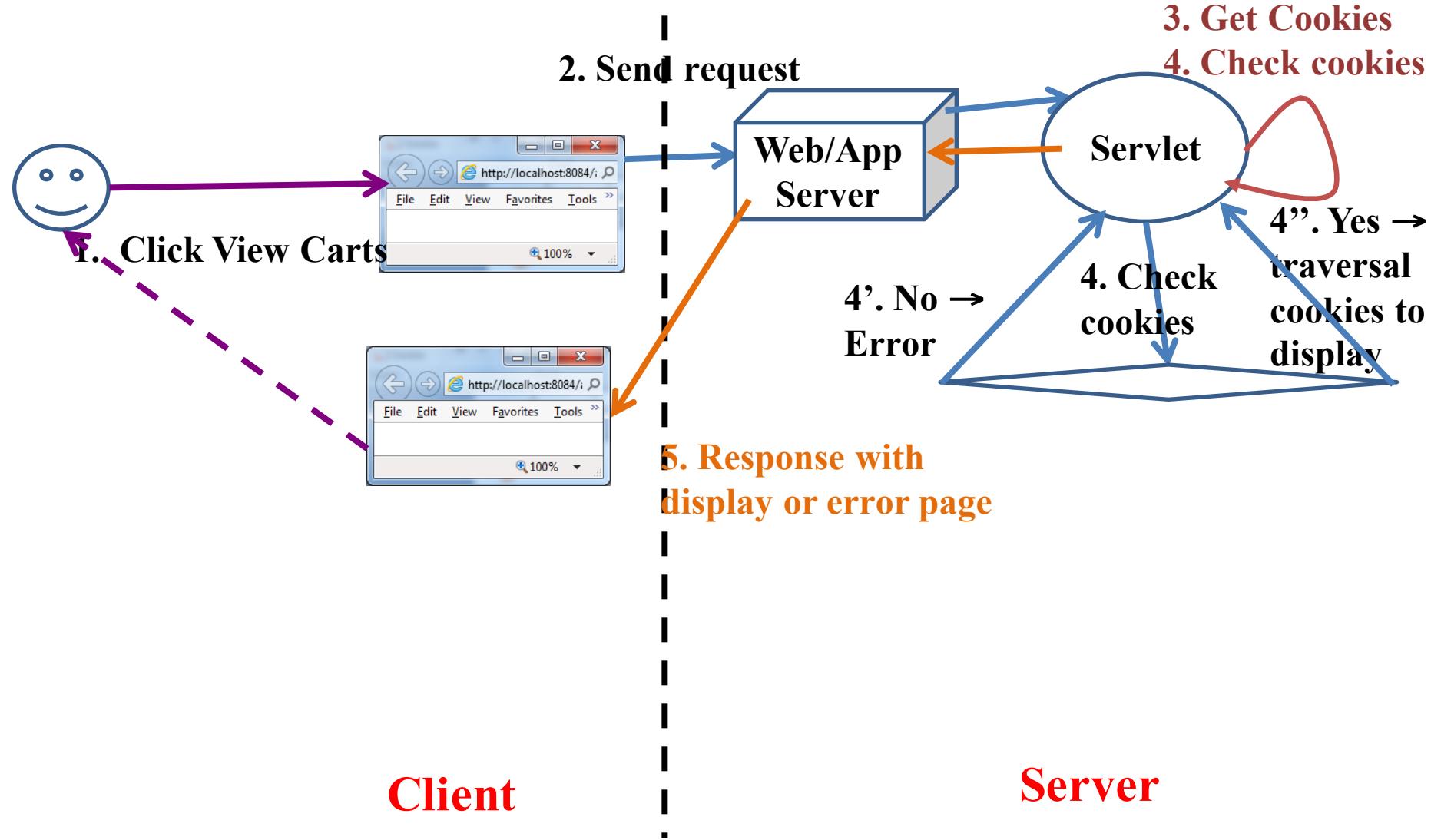
# Appendix

## Shopping Cart using Cookies

```
42 } else {
43     boolean bFound = false;
44     //find the exist title in the cart
45     for (int i = 0; i < cookies.length; i++) {
46         if (cookies[i].getName().equals(title)) {
47             bFound = true;
48             String value = cookies[i].getValue();
49             int quantity = Integer.parseInt(value) + 1;
50             Cookie cookie = new Cookie(title, String.valueOf(quantity));
51             cookie.setMaxAge(60 * 5); //5m
52             response.addCookie(cookie); //override
53             break;
54         }
55     }
56     if (!bFound) {
57         Cookie cookie = new Cookie(title, "1");
58         cookie.setMaxAge(60 * 5); //5m
59         response.addCookie(cookie);
60     }
61 }
62
63     response.sendRedirect(shoppingPage);
64 } finally {
65     out.close();
66 }
67 }
```

# Appendix

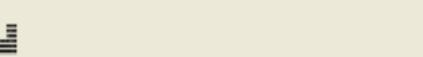
## Interactive Server Model



# Appendix

## Shopping Cart using Cookies

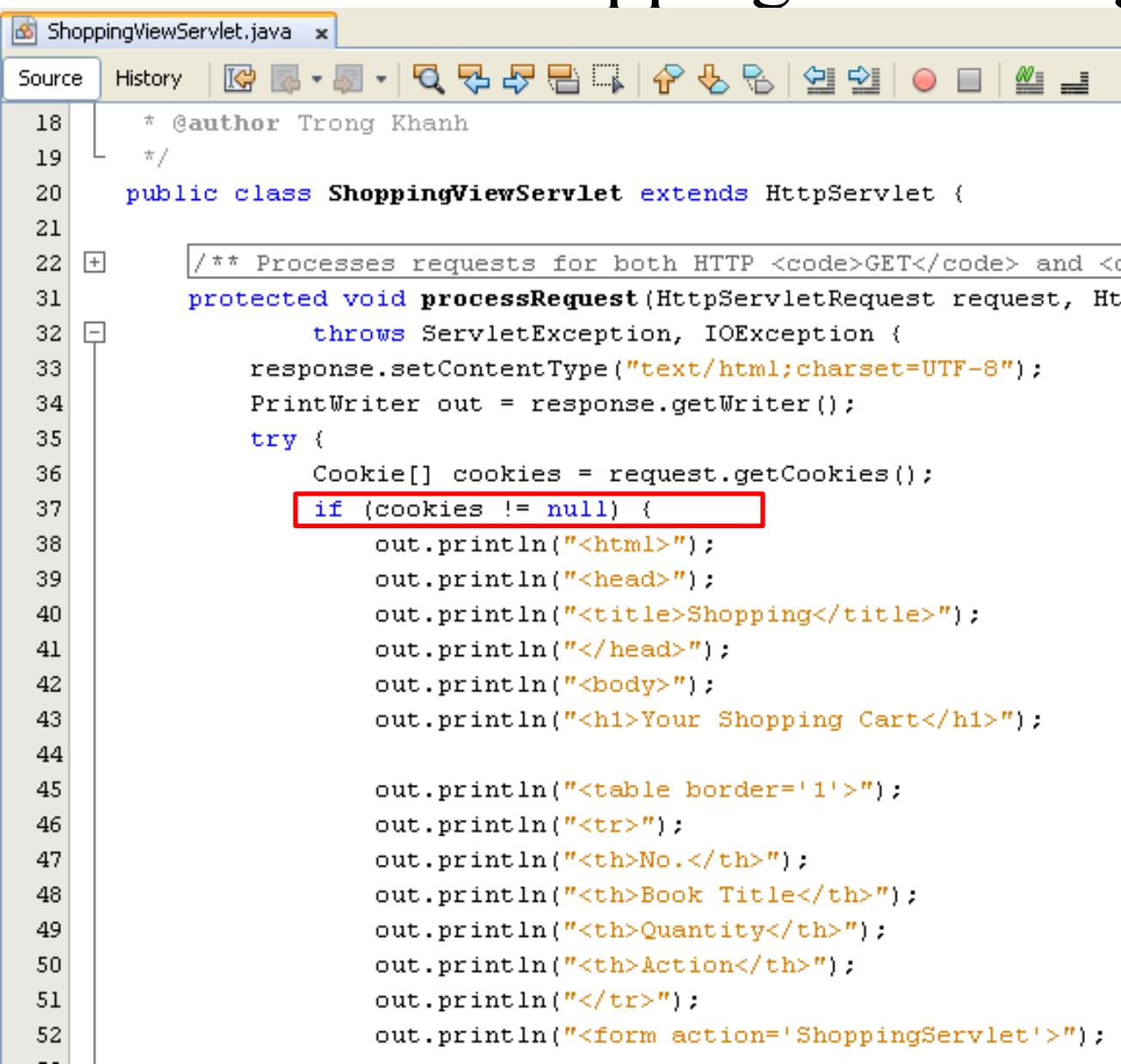
ShoppingServlet.java

```
Source History |  |  | 
```

```
18 * @author Trong Khanh
19 */
20 public class ShoppingServlet extends HttpServlet {
21     private final String addCartServlet = "ShoppingAddServlet";
22     private final String viewCartServlet = "ShoppingViewServlet";
23
24     /** Processes requests for both HTTP ...10 lines */
25     protected void processRequest(HttpServletRequest request, HttpServletResponse resp
26         throws ServletException, IOException {
27         response.setContentType("text/html;charset=UTF-8");
28         PrintWriter out = response.getWriter();
29         try {
30
31             String action = request.getParameter("action");
32
33             if (action.equals("Add To Cart")) {
34                 RequestDispatcher rd = request.getRequestDispatcher(addCartServlet);
35                 rd.forward(request, response);
36             } else if (action.equals("View Cart")) {
37                 RequestDispatcher rd = request.getRequestDispatcher(viewCartServlet);
38                 rd.forward(request, response);
39             }
40         }
41     }
42 }
```

# Appendix

## Shopping Cart using Cookies



The screenshot shows a Java code editor with the file `ShoppingViewServlet.java` open. The code implements a `HttpServlet` to handle shopping cart requests using cookies. The `processRequest` method retrieves the cookies from the request and prints an HTML page with a table and a form.

```
18 * @author Trong Khanh
19 */
20 public class ShoppingViewServlet extends HttpServlet {
21
22     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
23      * methods.
24      *
25      * @param request the <code>HttpServletRequest</code> object
26      * @param response the <code>HttpServletResponse</code> object
27      * @throws ServletException if a servlet-specific error occurs
28      * @throws IOException if an I/O error occurs
29      */
30     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31             throws ServletException, IOException {
32         response.setContentType("text/html;charset=UTF-8");
33         PrintWriter out = response.getWriter();
34         try {
35             Cookie[] cookies = request.getCookies();
36             if (cookies != null) {
37                 out.println("<html>");
38                 out.println("<head>");
39                 out.println("<title>Shopping</title>");
40                 out.println("</head>");
41                 out.println("<body>");
42                 out.println("<h1>Your Shopping Cart</h1>");
43
44                 out.println("<table border='1'>");
45                 out.println("<tr>");
46                 out.println("<th>No.</th>");
47                 out.println("<th>Book Title</th>");
48                 out.println("<th>Quantity</th>");
49                 out.println("<th>Action</th>");
50                 out.println("</tr>");
51                 out.println("<form action='ShoppingServlet'>");
```

# Appendix

## Shopping Cart using Cookies

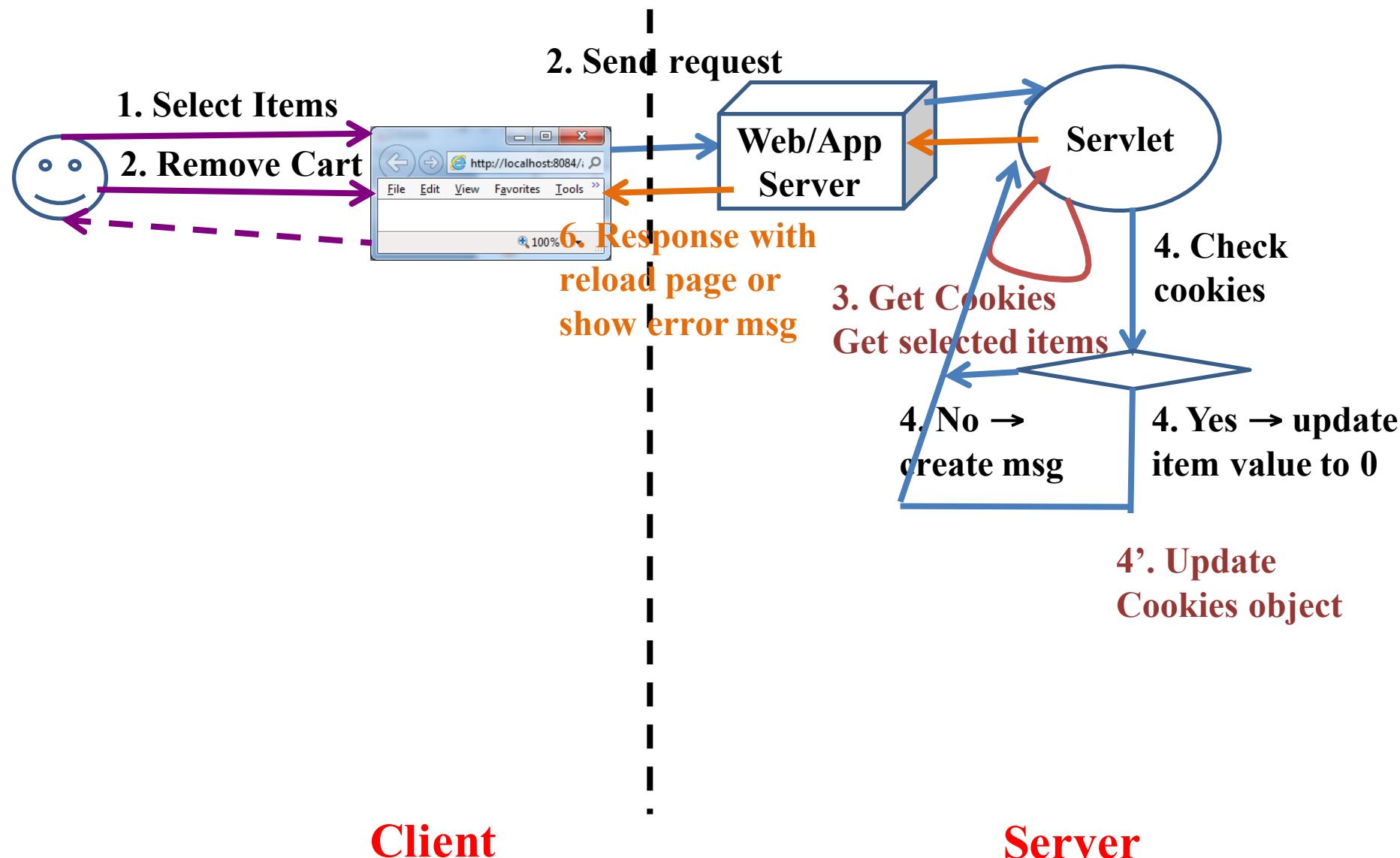
```
53
54     int count = 1;
55
56     for (int i = 0; i < cookies.length; i++) {
57         int tmp = Integer.parseInt(cookies[i].getValue());
58
59         if (tmp > 0) {
60             out.println("<tr>");
61             out.println("<td>" + count++ + "</td>");
62             out.println("<td>" + cookies[i].getName() + "</td>");
63             out.println("<td>" + cookies[i].getValue() + "</td>");
64             out.println("<td><input type='checkbox' name='rmv' value='"
65                         + cookies[i].getName() + "' /></td>");
66             out.println("</tr>");
67         }
68     }
69
70     out.println("<tr>");
71     out.println("<td colspan='3'><a href='shoppingCookie.html'>Add More Cart</a></td>");
72     out.println("<td><input type='submit' value='Remove Cart' name='action' /></td>");
73     out.println("</tr>");
74
75     out.println("</form>");
76     out.println("</table>");
77
78     out.println("</body>");
79     out.println("</html>");
80     return;
81 }
82
83 } finally {
84     out.close();
85 }
```

The code is a Java servlet for a shopping cart application using cookies. It prints an HTML table of items in the cart and provides links to add more items or remove them. The code is annotated with line numbers from 53 to 83.

- Line 53: Starts the main method.
- Line 54: initializes a counter for the table rows.
- Line 55: begins a for loop to iterate through the cookies array.
- Line 56: initializes a temporary variable tmp to store the integer value of the cookie.
- Line 57: checks if tmp is greater than 0.
- Line 58: prints the opening tag for a table row.
- Line 59: prints the first three columns of the table row.
- Line 60: prints the item name in the fourth column.
- Line 61: prints the item value in the fifth column.
- Line 62: prints a checkbox input field in the sixth column, linking to the current page with the item name as the value.
- Line 63: prints the closing tag for the table row.
- Line 64: ends the if block.
- Line 65: ends the for loop.
- Line 66: prints the closing tag for the table.
- Line 67: prints the closing tag for the tr element.
- Line 68: prints the opening tag for a new tr element.
- Line 69: prints the first three columns of the new tr element, containing a link to add more items.
- Line 70: prints the fourth column of the new tr element, containing a submit button to remove the item.
- Line 71: prints the closing tag for the new tr element.
- Line 72: prints the closing tag for the form element.
- Line 73: prints the closing tag for the table element.
- Line 74: prints the closing tag for the body element.
- Line 75: prints the closing tag for the html element.
- Line 76: returns from the method.
- Line 77: ends the main method.
- Line 78: prints a message indicating the cart is removed or empty.
- Line 79: begins a finally block.
- Line 80: closes the output stream.
- Line 81: ends the finally block.
- Line 82: ends the main method.
- Line 83: ends the class definition.

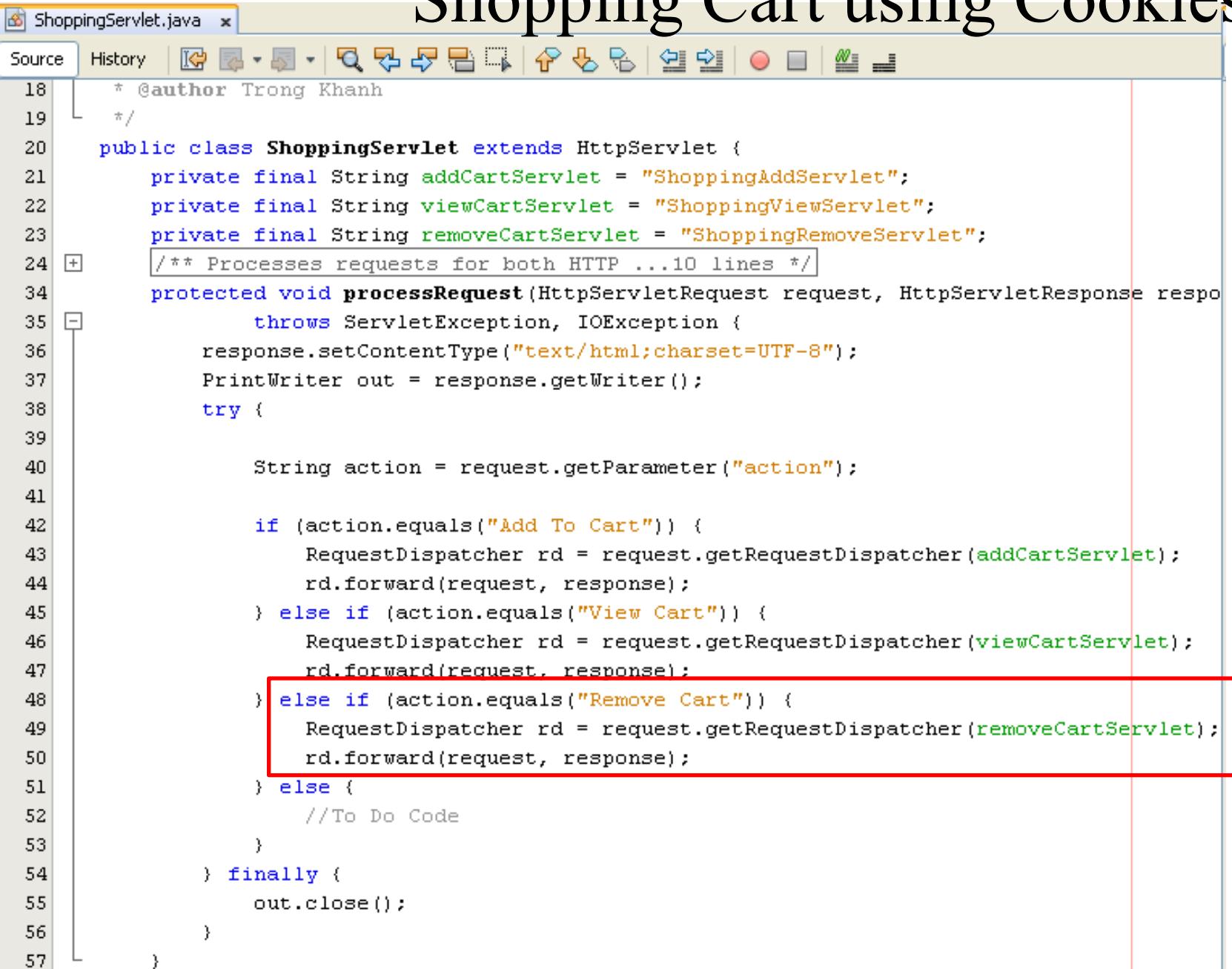
# Appendix

## Interactive Server Model



# Appendix

## Shopping Cart using Cookies



The screenshot shows a Java code editor with the file `ShoppingServlet.java` open. The code implements a `HttpServlet` that handles requests for adding items to a shopping cart, viewing the cart, and removing items from it. The code uses `RequestDispatcher` to forward requests to three other servlets: `ShoppingAddServlet`, `ShoppingViewServlet`, and `ShoppingRemoveServlet`. A red box highlights the `Remove Cart` logic.

```
18 * @author Trong Khanh
19 */
20 public class ShoppingServlet extends HttpServlet {
21     private final String addCartServlet = "ShoppingAddServlet";
22     private final String viewCartServlet = "ShoppingViewServlet";
23     private final String removeCartServlet = "ShoppingRemoveServlet";
24     /** Processes requests for both HTTP ...10 lines */
25
26     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27             throws ServletException, IOException {
28         response.setContentType("text/html;charset=UTF-8");
29         PrintWriter out = response.getWriter();
30         try {
31
32             String action = request.getParameter("action");
33
34             if (action.equals("Add To Cart")) {
35                 RequestDispatcher rd = request.getRequestDispatcher(addCartServlet);
36                 rd.forward(request, response);
37             } else if (action.equals("View Cart")) {
38                 RequestDispatcher rd = request.getRequestDispatcher(viewCartServlet);
39                 rd.forward(request, response);
40             } else if (action.equals("Remove Cart")) {
41                 RequestDispatcher rd = request.getRequestDispatcher(removeCartServlet);
42                 rd.forward(request, response);
43             } else {
44                 //To Do Code
45             }
46         } finally {
47             out.close();
48         }
49     }
50 }
```

# Appendix

## Shopping Cart using Cookies

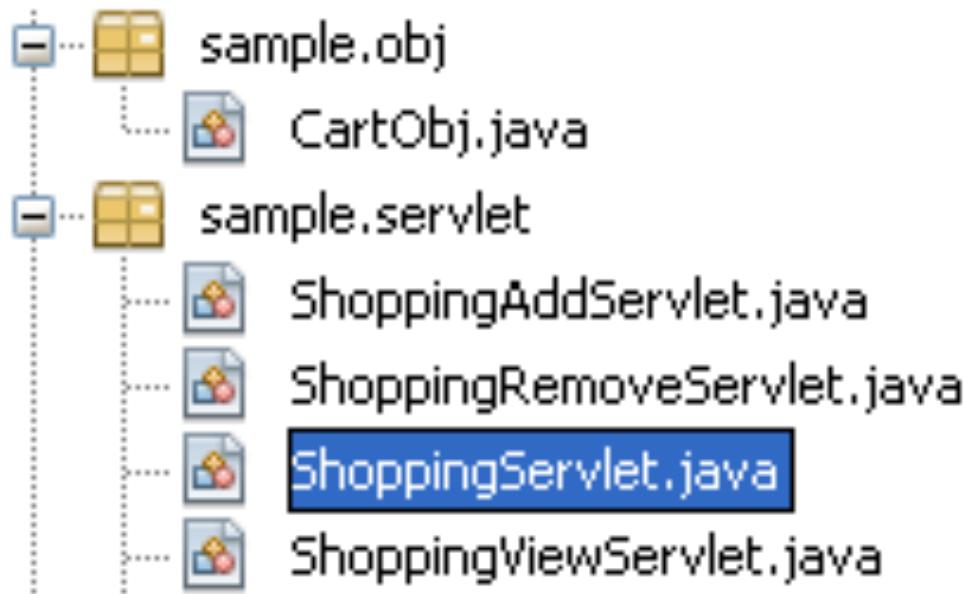
ShoppingRemoveServlet.java

```
Source History | 
```

```
18 * @author Trong Khanh
19 */
20 public class ShoppingRemoveServlet extends HttpServlet {
21
22     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
23      * @param request the servlet request
24      * @param response the servlet response
25      * @throws ServletException if a servlet-specific error occurs
26      * @throws IOException if an I/O error occurs
27      */
28     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30         response.setContentType("text/html;charset=UTF-8");
31         PrintWriter out = response.getWriter();
32         try {
33             Cookie[] cookies = request.getCookies();
34             if (cookies != null) {
35                 String[] list = request.getParameterValues("rmv");
36                 if (list != null) {
37                     for (int i = 0; i < list.length; i++) {
38                         String tmp = list[i];
39                         for (int j = 0; j < cookies.length; j++) {
40                             if (cookies[j].getName().equals(tmp)) {
41                                 cookies[j].setValue("0");
42                                 cookies[j].setMaxAge(60 * 5);
43                                 response.addCookie(cookies[j]);
44                                 break;
45                             }
46                         }
47                     }
48                 }
49             }
50         } catch (Exception e) {
51             e.printStackTrace();
52         }
53         String urlRewriting = "ShoppingServlet?action=View Cart";
54         response.sendRedirect(urlRewriting);
55     } else {
56         out.println("<h2>Cart is removed!!!!</h2>");
57     }
58 }
```

# Sessions & Listeners

## Shopping Cart using Cookies – Example



# Appendix

## Request and Context Listeners

Listener Interface Name	Applies to	Function
ServletRequestListener	Request objects	Responds to the life and death of each request.
ServletContextListener	The context object	Responds to the life and death of the context for a web application.
ServletRequestAttributeListener	Request objects	Responds to any change to the set of attributes attached to a request object.
ServletContextAttributeListener	The context object	Responds to any change to the set of attributes attached to the context object.

- There are **two things** that need to do **to set up a listener** in a web application:
  - Write a class that **implements** the **appropriate listener interface**.
  - **Register** the **class name** in the **web application deployment descriptor**, web.xml.

<listener>

<listener-class>className</listener-class>

</listener>

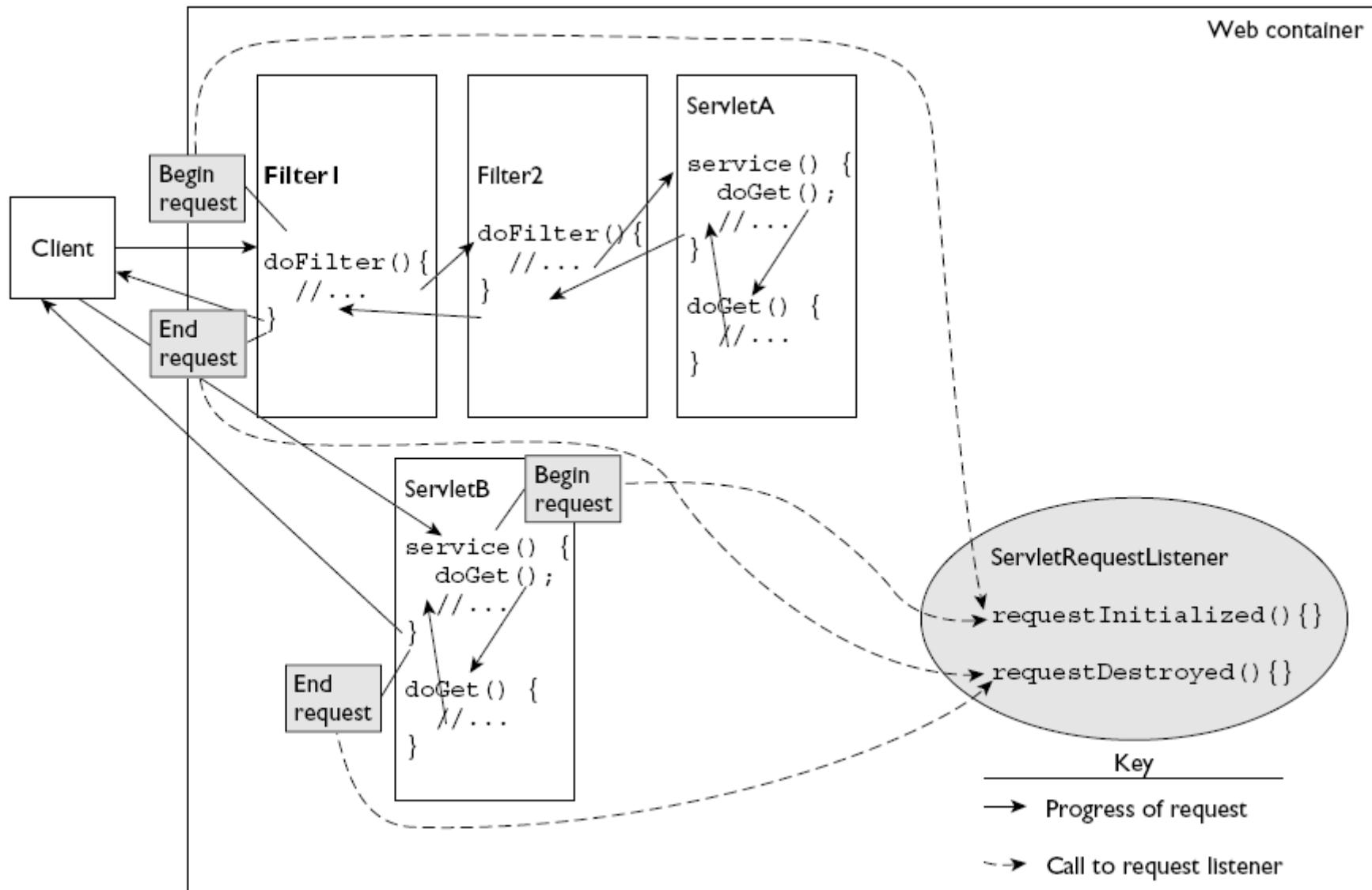
# Appendix

## Request Listeners

- **ServletRequestListener** deals with the **life cycle of each request object**
- A class **implementing** the **ServletRequestListener** interface has 2 methods
  - **requestInitialized()**: is called the **moment** that **any request** in the web container be **comes newly available** (or it is called at the **beginning of any request's scope**)
    - This is at the beginning of a servlet's service() method or earlier than that if filter chain is involved
  - **requestDestroyed()**: is called for each request that **comes to an end** – either at the **end of the servlet's service() method** or at the **end of the doFilter() method** for the first filter in a chain
- **Each** of these **ServletRequestListener** methods **accept** a **ServletRequestEvent** as a parameter. This event object has 2 methods
  - getServletContext()
  - getServletRequest()

# Appendix

## Request Listeners



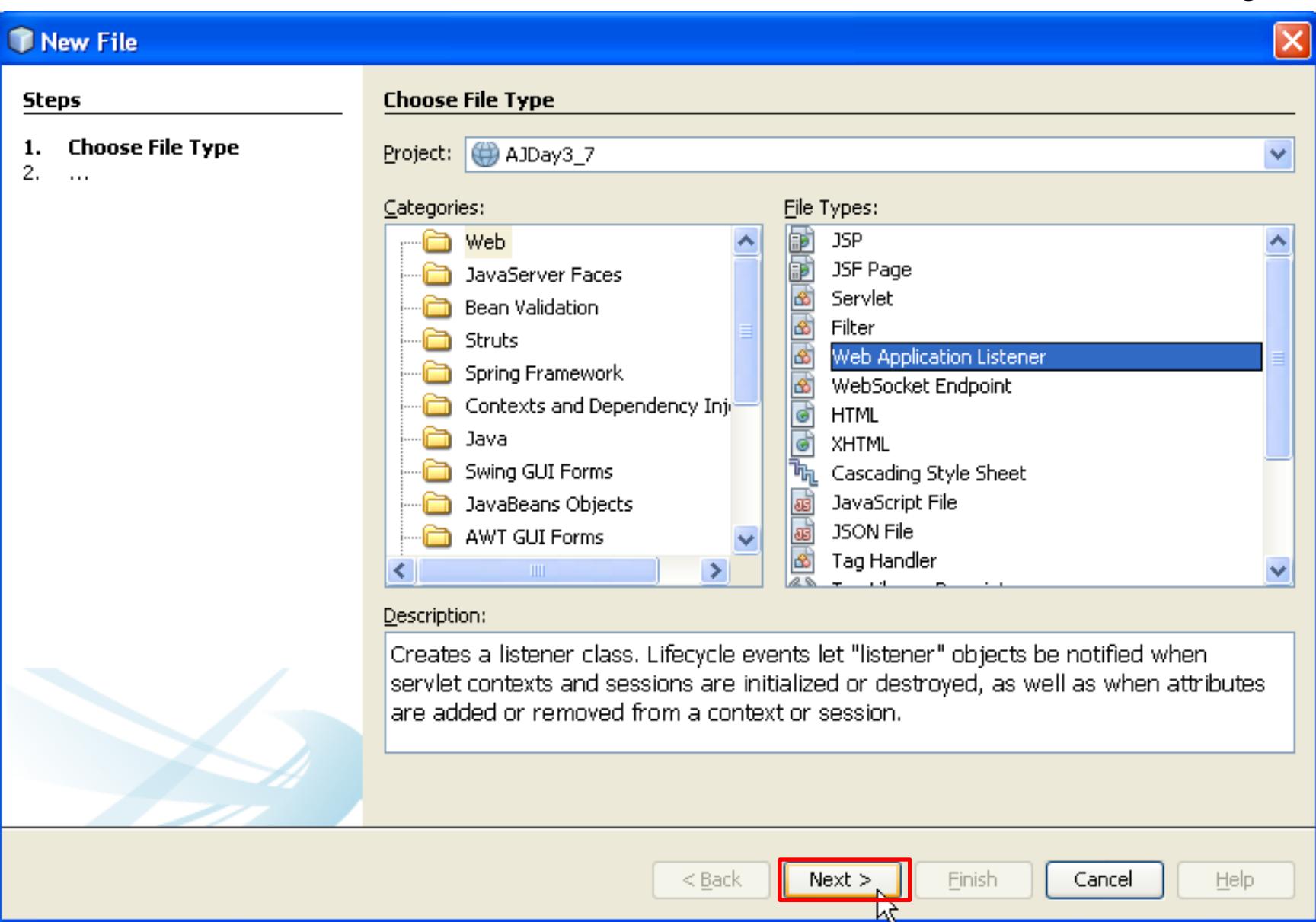
# Appendix

## Request Attribute Listeners

- **ServletRequestAttributeListener** deals with the **life cycle of the attributes attached to request objects**
- A **class implementing the ServletRequestAttributeListener interface has 3 methods**
  - **attributeAdded()**: is called whenever a new attribute is added to any request
  - **attributeRemoved()**: is called whenever an attribute is removed from a request
  - **attributeReplaced()**: is called whenever an attribute is replaced
- Each of these **ServletRequestAttributeListener** methods **accept a ServletRequestAttributeEvent as a parameter**. This event object has **2 methods**
  - **getName()**: returns name of attribute
  - **getValue()**: returns old value of attribute
- The **ServletRequestAttributeEvent** inherits from **ServletRequestEvent**
- The “grandparent” of The **ServletRequestEvent** is **java.util.EventObject**
  - The **getSource()** method returns the object that is the source of the event

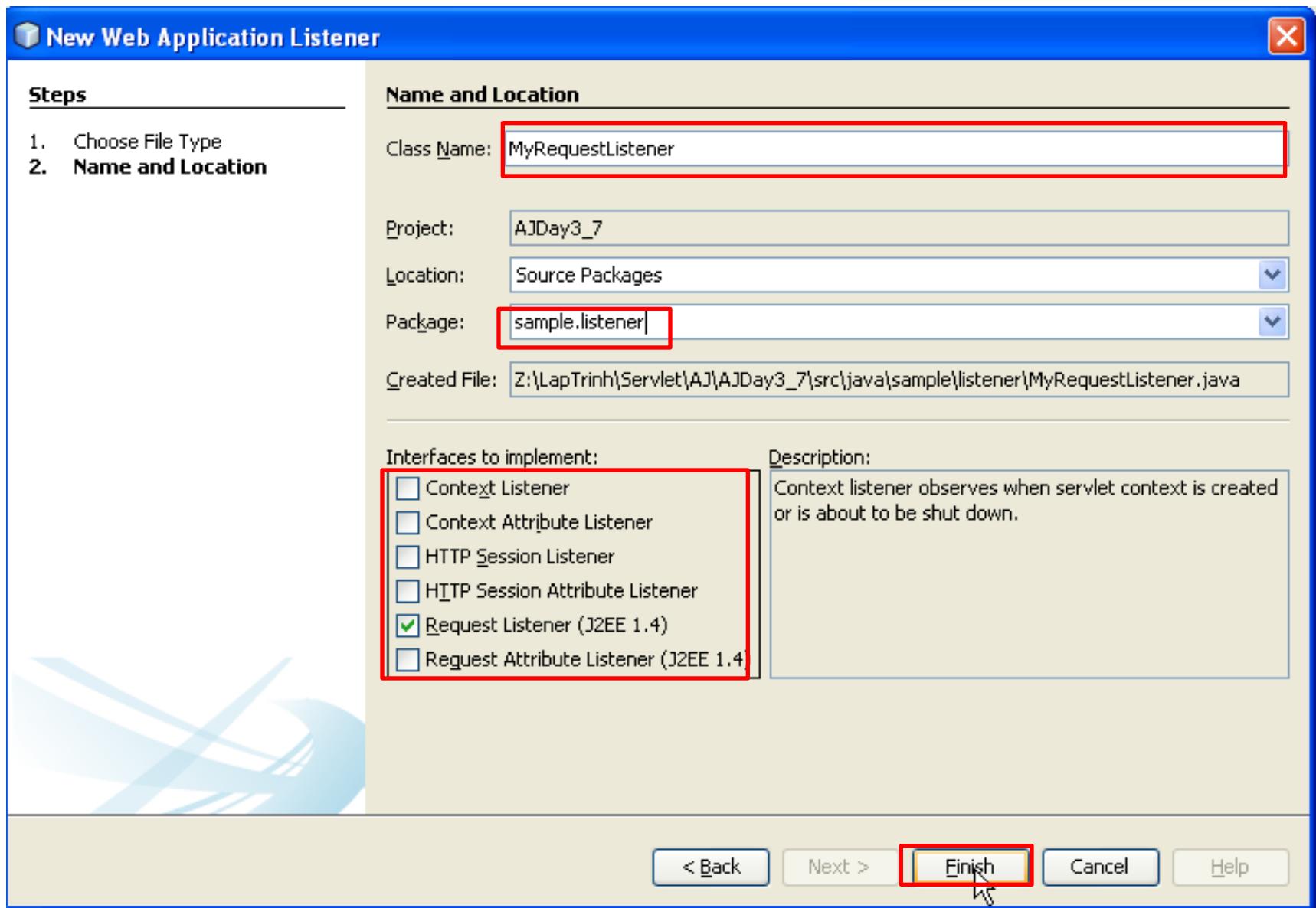
# Appendix

## How to Add Listener to Web Project



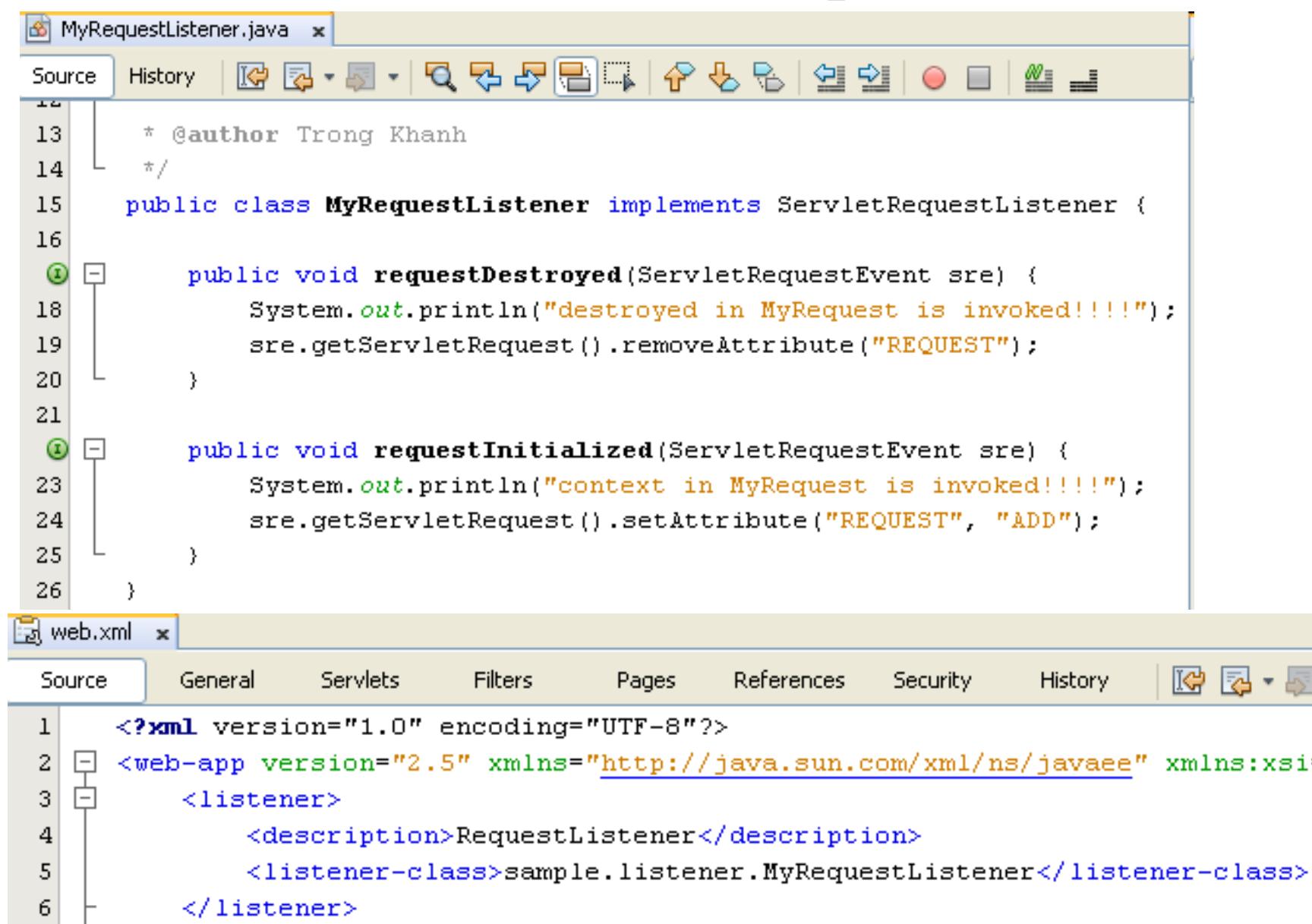
# Appendix

## How to Add Listener to Web Project



# Appendix

## Example



The screenshot shows an IDE interface with two tabs open:

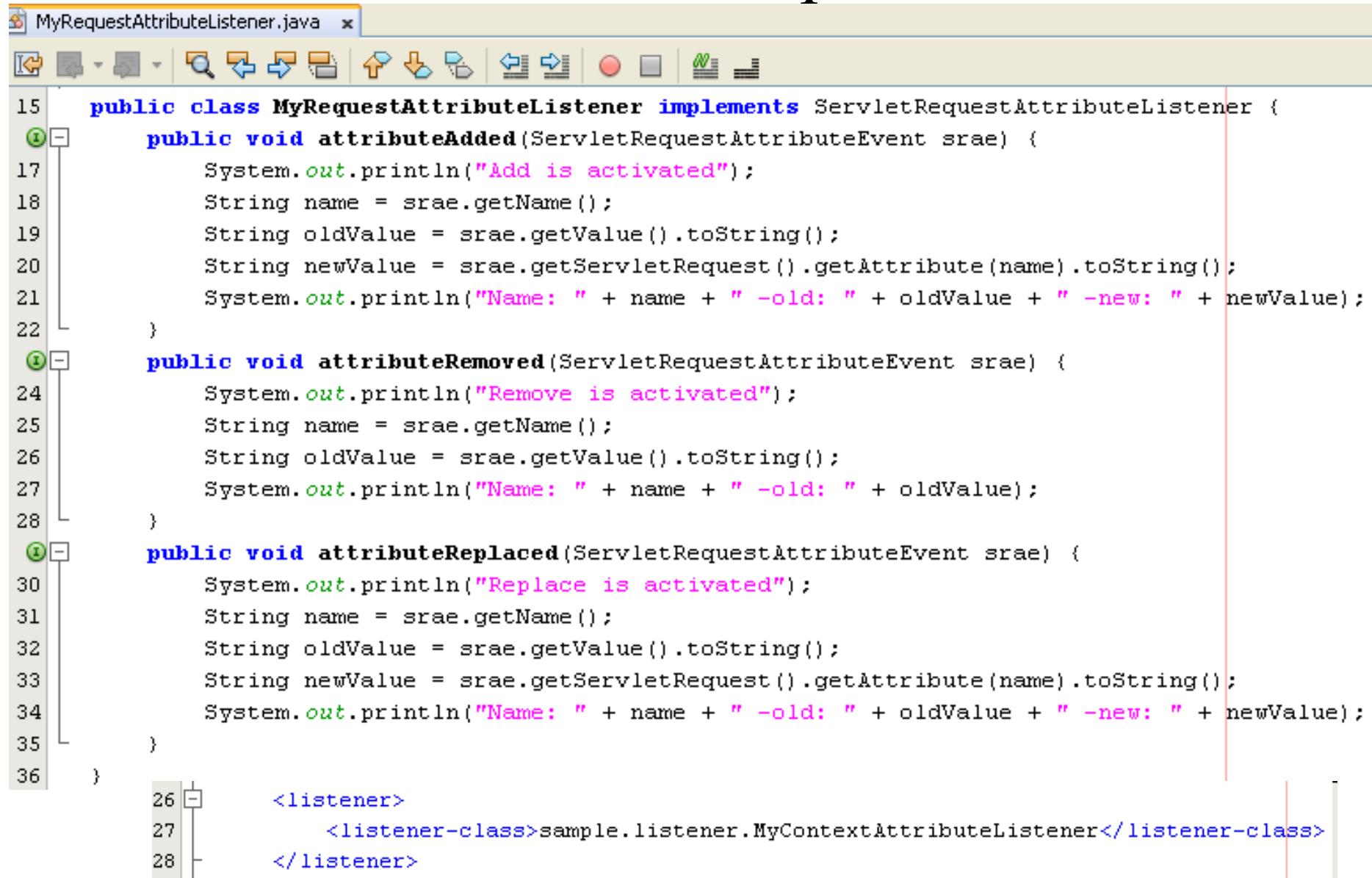
- MyRequestListener.java**: A Java class named `MyRequestListener` that implements the `ServletRequestListener` interface. It contains two methods: `requestDestroyed` and `requestInitialized`. Both methods output a message to the console and manipulate the request attributes.
- web.xml**: A configuration file for a Java Web Application. It defines a listener named "RequestListener" that points to the `MyRequestListener` class.

```
MyRequestListener.java
13  * @author Trong Khanh
14  */
15  public class MyRequestListener implements ServletRequestListener {
16
17      public void requestDestroyed(ServletRequestEvent sre) {
18          System.out.println("destroyed in MyRequest is invoked!!!!");
19          sre.getServletRequest().removeAttribute("REQUEST");
20      }
21
22      public void requestInitialized(ServletRequestEvent sre) {
23          System.out.println("context in MyRequest is invoked!!!!");
24          sre.getServletRequest().setAttribute("REQUEST", "ADD");
25      }
26  }
```

```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi=
3     <listener>
4         <description>RequestListener</description>
5         <listener-class>sample.listener.MyRequestListener</listener-class>
6     </listener>
```

# Appendix

## Example



The screenshot shows a Java code editor with the file `MyRequestAttributeListener.java` open. The code implements the `ServletRequestAttributeListener` interface, handling three events: `attributeAdded`, `attributeRemoved`, and `attributeReplaced`. It prints the name, old value, and new value of each attribute to the console. Below the code, there is a configuration snippet for a web.xml file defining the listener.

```
15  public class MyRequestAttributeListener implements ServletRequestAttributeListener {
16      public void attributeAdded(ServletRequestAttributeEvent srae) {
17          System.out.println("Add is activated");
18          String name = srae.getName();
19          String oldValue = srae.getValue().toString();
20          String newValue = srae.getServletRequest().getAttribute(name).toString();
21          System.out.println("Name: " + name + " -old: " + oldValue + " -new: " + newValue);
22      }
23      public void attributeRemoved(ServletRequestAttributeEvent srae) {
24          System.out.println("Remove is activated");
25          String name = srae.getName();
26          String oldValue = srae.getValue().toString();
27          System.out.println("Name: " + name + " -old: " + oldValue);
28      }
29      public void attributeReplaced(ServletRequestAttributeEvent srae) {
30          System.out.println("Replace is activated");
31          String name = srae.getName();
32          String oldValue = srae.getValue().toString();
33          String newValue = srae.getServletRequest().getAttribute(name).toString();
34          System.out.println("Name: " + name + " -old: " + oldValue + " -new: " + newValue);
35      }
36  }
37  <listener>
38      <listener-class>sample.listener.MyContextAttributeListener</listener-class>
39  </listener>
```

# Appendix

## Example

RequestListenerServlet.java

```

28     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException {
30             response.setContentType("text/html;charset=UTF-8");
31             PrintWriter out = response.getWriter();
32             try {
33                 out.println("<html>");
34                 out.println("<head>");
35                 out.println("<title>Request</title>");
36                 out.println("</head>");
37                 out.println("<body>");
38                 out.println("<h1>Request Processing</h1>");
39
40                 request.setAttribute("VALUE", "ADD");
41                 request.setAttribute("VALUE", "MODIFIED");
42                 request.removeAttribute("VALUE");
43
44                 out.println("REQUEST: " + request.getAttribute("REQUEST"));
45                 out.println("Finished!!!");
46
47                 out.println("</body>");
48                 out.println("</html>");
49             } finally {
50                 out.close();
51             }
52         }

```

Request - Windows Internet Explorer

File Edit View Favorites Tools Help

Favorites Request

**Request Processing**

REQUEST: ADD Finished!!!

# Appendix

## Example

```
context in MyRequest is invoked!!!!  
Add is activated  
Name: REQUEST -old: ADD -new: ADD  
Replace is activated  
Name: org.apache.catalina.ASYNC_SUPPORTED -old: true -new: false  
Add is activated  
Name: netbeans.monitor.request -old: uri: /AJDay3_7/RequestListenerServlet  
method: GET  
QueryString: null  
Parameters:  
Headers:  
    Name: accept      Value: */*  
    Name: accept-language  Value: vi  
    Name: user-agent      Value: Mozilla/4.0 (compatible; MSIE 8.0; W  
    Name: accept-encoding   Value: gzip, deflate|  
    Name: host          Value: localhost:8084  
    Name: connection      Value: Keep-Alive  
-new: uri: /AJDay3_7/RequestListenerServlet  
method: GET  
QueryString: null  
Parameters:  
Headers:  
    Name: accept      Value: */*  
    Name: accept-language  Value: vi  
    Name: user-agent      Value: Mozilla/4.0 (compatible; MSIE 8.0; W  
    Name: accept-encoding   Value: gzip, deflate  
    Name: host          Value: localhost:8084  
    Name: connection      Value: Keep-Alive
```

# Appendix

## Example

```
Add is activated
Name: netbeans.monitor.monData -old: [MonitorData] -new: [MonitorData]
Add is activated
Name: netbeans.monitor.response -old: org.netbeans.modules.web.monitor.serve
bb5
Add is activated
Name: netbeans.monitor.filter -old: MonitorFilter(ApplicationFilterConfig[nam
ter(ApplicationFilterConfig[name=HTTPMonitorFilter, filterClass=org.netbeans.
Add is activated
Name: VALUE -old: ADD -new: ADD
Replace is activated
Name: VALUE -old: ADD -new: MODIFIED
Remove is activated
Name: VALUE -old: MODIFIED
Remove is activated
Name: netbeans.monitor.request -old: uri: /AJDay3_7/RequestListenerServlet
method: GET
QueryString: null
Parameters:
Headers:
    Name: accept      Value: */
    Name: accept-language  Value: vi
    Name: user-agent        Value: Mozilla/4.0 (compatible; MSIE 8.0; W:
    Name: accept-encoding     Value: gzip, deflate
    Name: host            Value: localhost:8084
    Name: connection       Value: Keep-Alive
```

# Appendix

## Example

Remove is activated

Name: netbeans.monitor.response -old: org.netbeans.

Remove is activated

Name: netbeans.monitor.filter -old: MonitorFilter(I

Remove is activated

Name: netbeans.monitor.monData -old: [MonitorData]

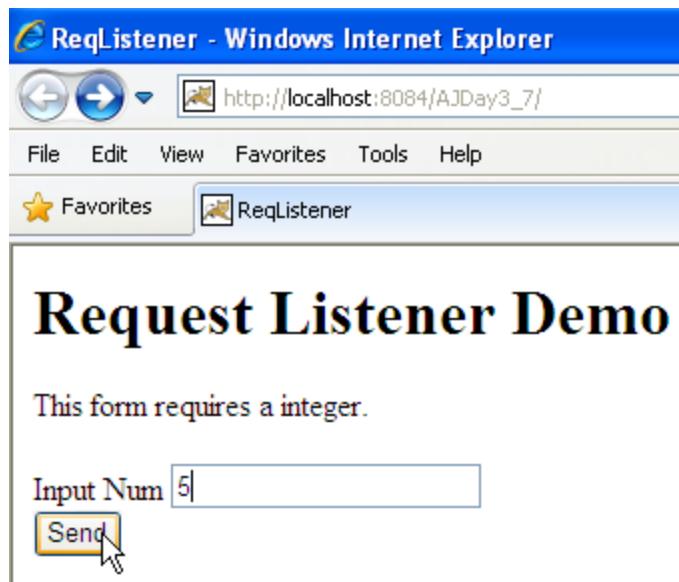
destroyed in MyRequest is invoked!!!!

Remove is activated

Name: REQUEST -old: ADD

# Appendix

## Practices – Example



ReqListener - Windows Internet Explorer  
http://localhost:8084/AJDay3\_7/

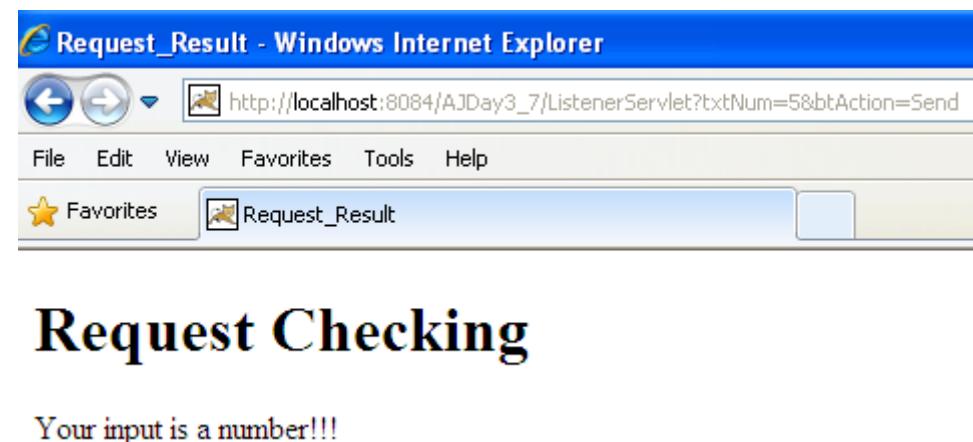
File Edit View Favorites Tools Help

Favorites ReqListener

## Request Listener Demo

This form requires a integer.

Input Num



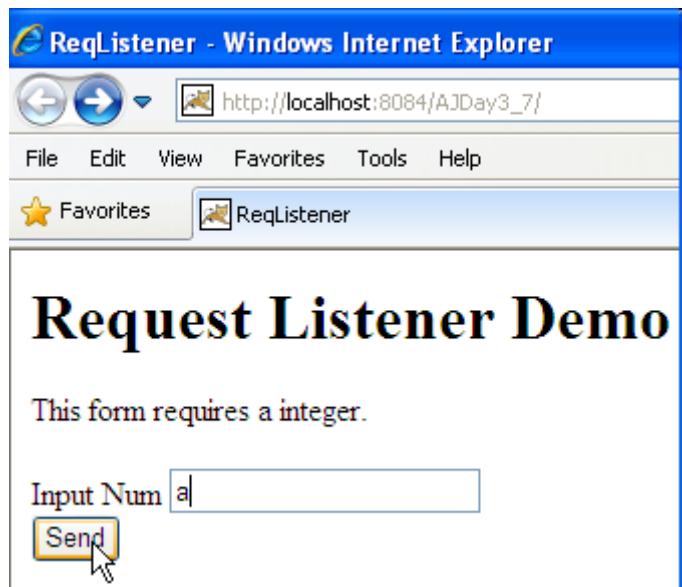
Request\_Result - Windows Internet Explorer  
http://localhost:8084/AJDay3\_7/ListenerServlet?txtNum=5&btAction=Send

File Edit View Favorites Tools Help

Favorites Request\_Result

## Request Checking

Your input is a number!!!



ReqListener - Windows Internet Explorer  
http://localhost:8084/AJDay3\_7/

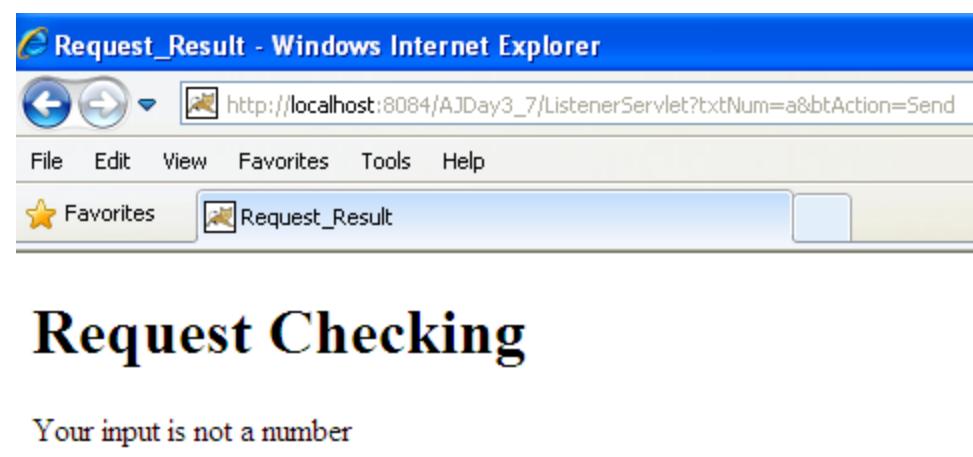
File Edit View Favorites Tools Help

Favorites ReqListener

## Request Listener Demo

This form requires a integer.

Input Num



Request\_Result - Windows Internet Explorer  
http://localhost:8084/AJDay3\_7/ListenerServlet?txtNum=a&btAction=Send

File Edit View Favorites Tools Help

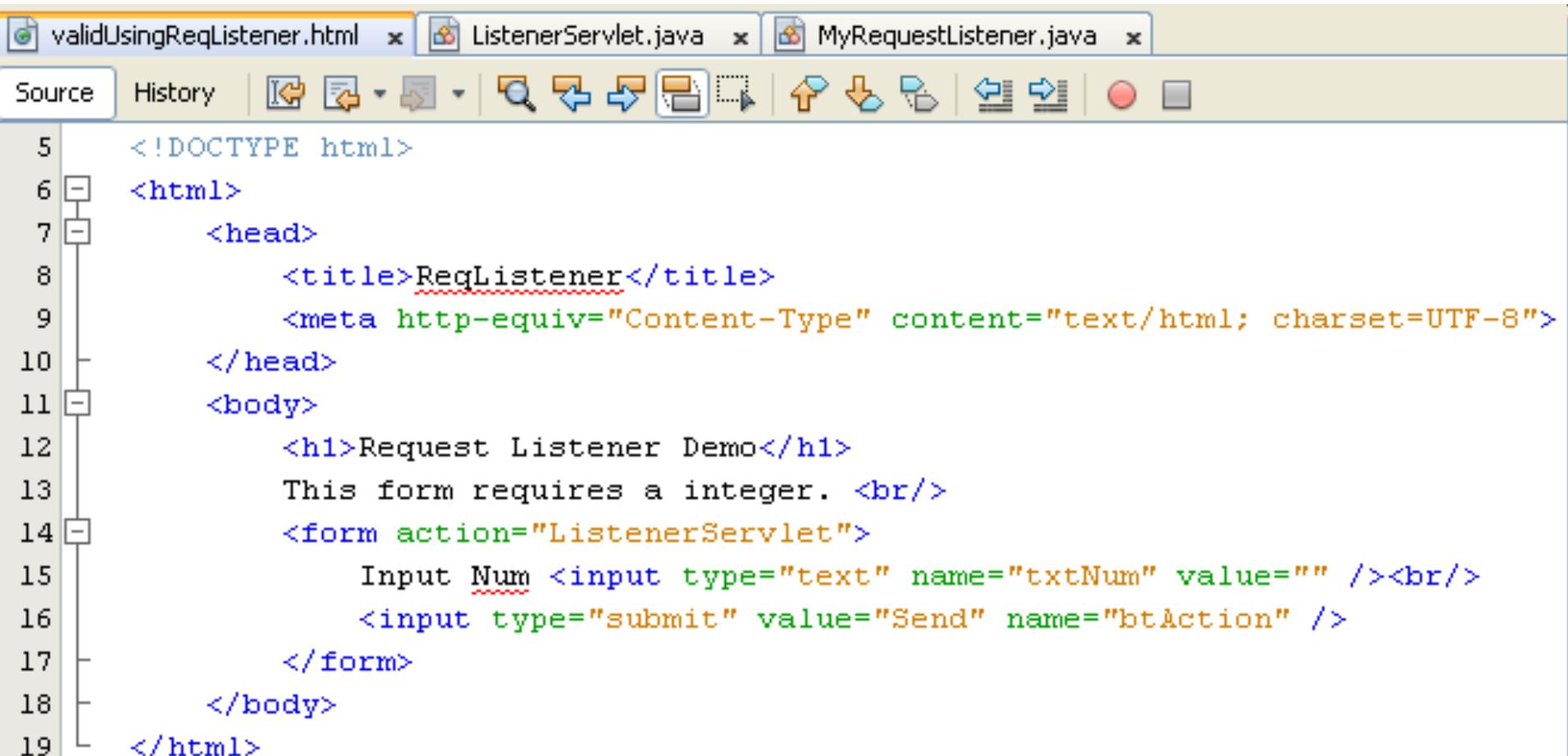
Favorites Request\_Result

## Request Checking

Your input is not a number

# Appendix

## Practices – Example



The screenshot shows a Java Integrated Development Environment (IDE) interface. At the top, there are three tabs: "validUsingReqListener.html", "ListenerServlet.java", and "MyRequestListener.java". Below the tabs is a toolbar with various icons for file operations like Open, Save, and Print.

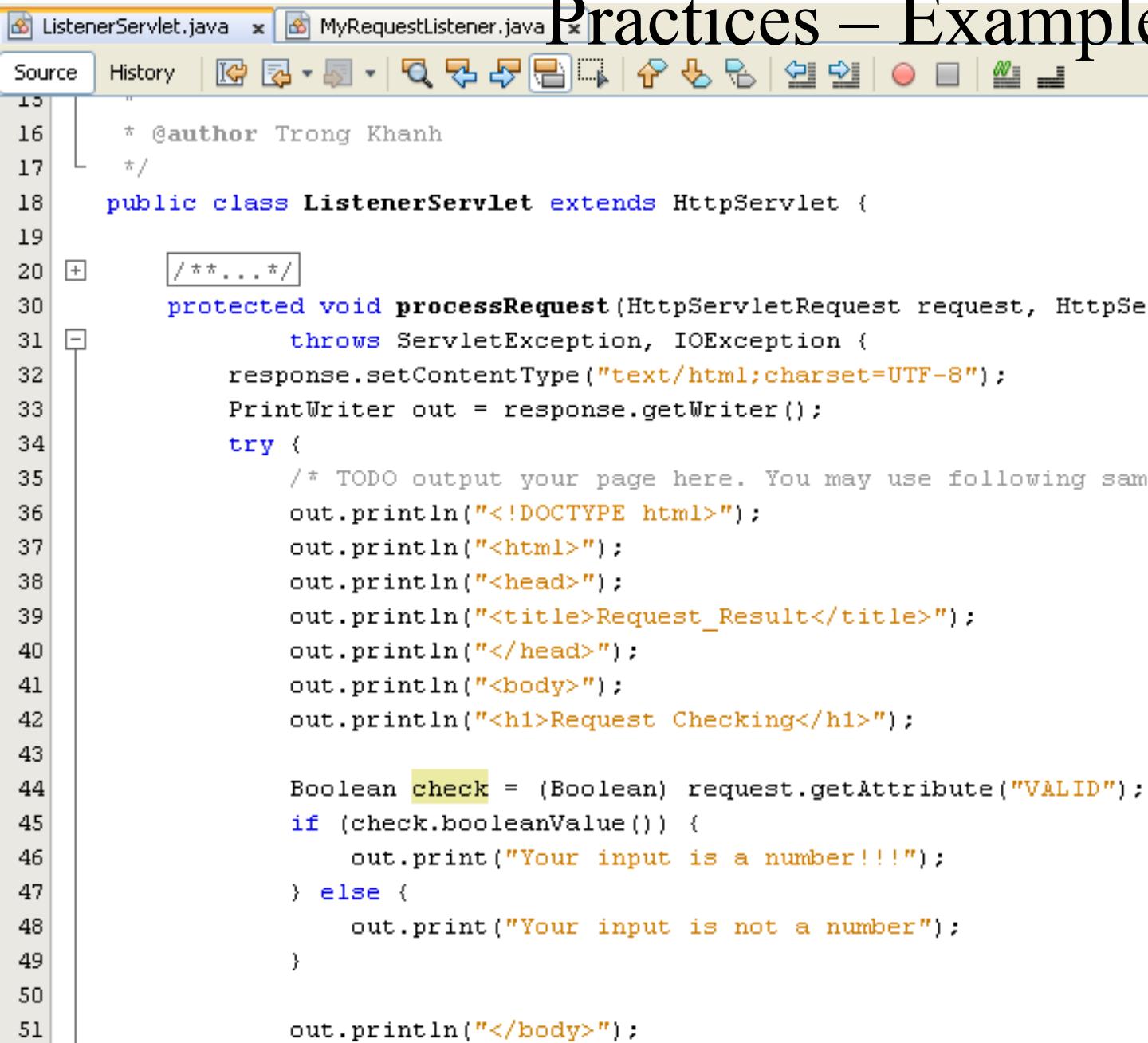
The main area displays the content of the "validUsingReqListener.html" file. The code is as follows:

```
5      <!DOCTYPE html>
6      <html>
7          <head>
8              <title>ReqListener</title>
9              <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10             </head>
11             <body>
12                 <h1>Request Listener Demo</h1>
13                 This form requires a integer. <br/>
14                 <form action="ListenerServlet">
15                     Input Num <input type="text" name="txtNum" value="" /><br/>
16                     <input type="submit" value="Send" name="btAction" />
17                 </form>
18             </body>
19         </html>
```

The code defines an HTML document with a title "ReqListener". It contains a heading "Request Listener Demo", a descriptive message "This form requires a integer.", and a form with a text input field named "txtNum" and a submit button named "btAction". The form's action is set to "ListenerServlet".

# Appendix

## Practices – Example

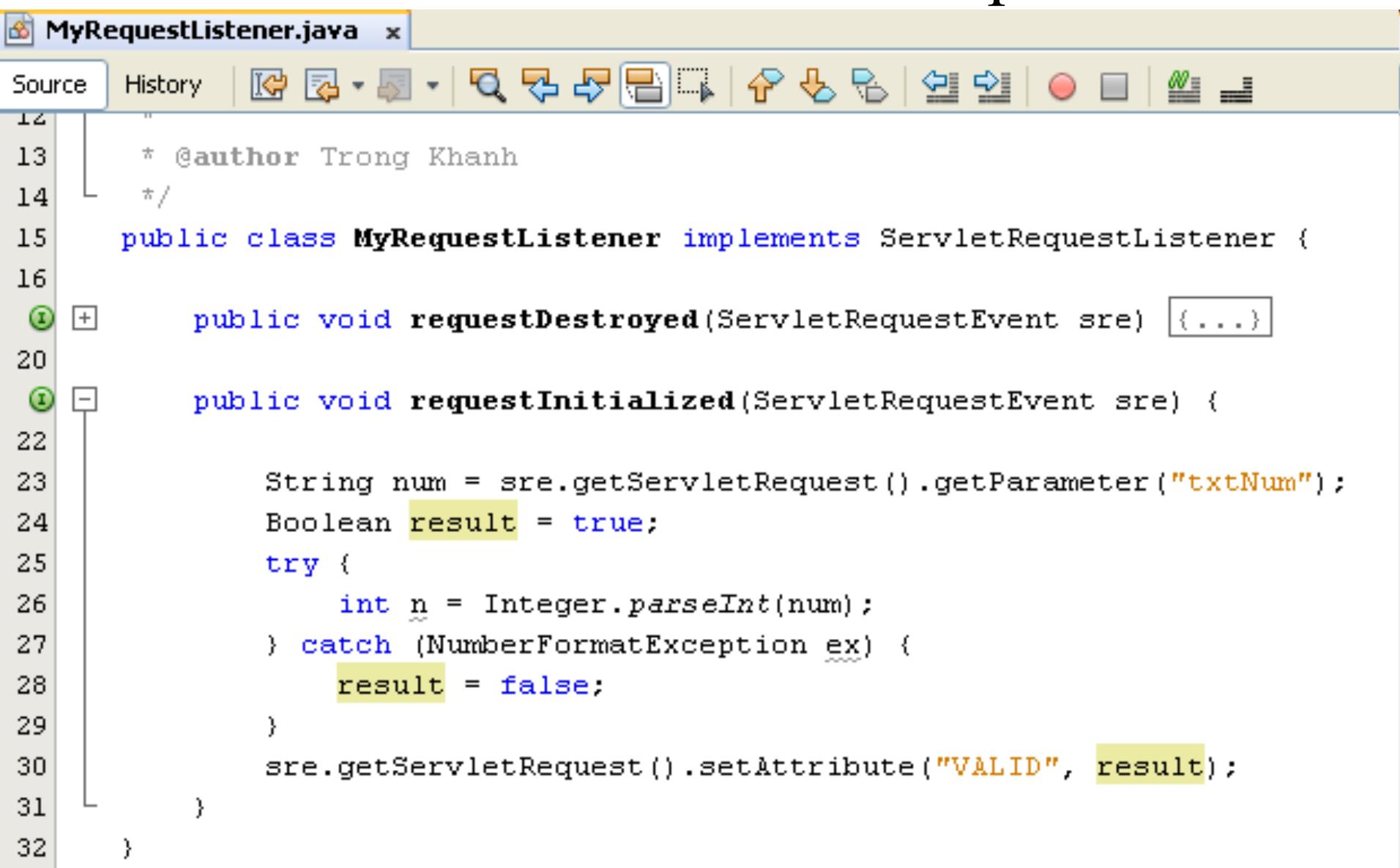


The screenshot shows a Java code editor with two tabs: ListenerServlet.java and MyRequestListener.java. The ListenerServlet.java tab is active, displaying the following code:

```
15
16     * @author Trong Khanh
17     */
18    public class ListenerServlet extends HttpServlet {
19
20        /**
21         * ...
22         */
23
24        protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25                throws ServletException, IOException {
26            response.setContentType("text/html;charset=UTF-8");
27            PrintWriter out = response.getWriter();
28            try {
29                /* TODO output your page here. You may use following sample code */
30                out.println("<!DOCTYPE html>");
31                out.println("<html>");
32                out.println("<head>");
33                out.println("<title>Request_Result</title>");
34                out.println("</head>");
35                out.println("<body>");
36                out.println("<h1>Request Checking</h1>");
37
38                Boolean check = (Boolean) request.getAttribute("VALID");
39                if (check.booleanValue()) {
40                    out.print("Your input is a number!!!");
41                } else {
42                    out.print("Your input is not a number");
43                }
44
45                out.println("</body>");
46            } catch (Exception e) {
47                e.printStackTrace();
48            }
49        }
50
51    }
```

# Appendix

## Practices – Example



The screenshot shows a Java code editor with the file `MyRequestListener.java` open. The code implements the `ServletRequestListener` interface, handling `requestDestroyed` and `requestInitialized` events. It retrieves a parameter from the request, parses it to an integer, and sets a boolean attribute `VALID` based on the result.

```
12
13     * @author Trong Khanh
14 */
15 public class MyRequestListener implements ServletRequestListener {
16
17     public void requestDestroyed(ServletRequestEvent sre) { ... }
18
19     public void requestInitialized(ServletRequestEvent sre) {
20
21
22         String num = sre.getServletRequest().getParameter("txtNum");
23         Boolean result = true;
24         try {
25             int n = Integer.parseInt(num);
26         } catch (NumberFormatException ex) {
27             result = false;
28         }
29         sre.getServletRequest().setAttribute("VALID", result);
30     }
31 }
32 }
```

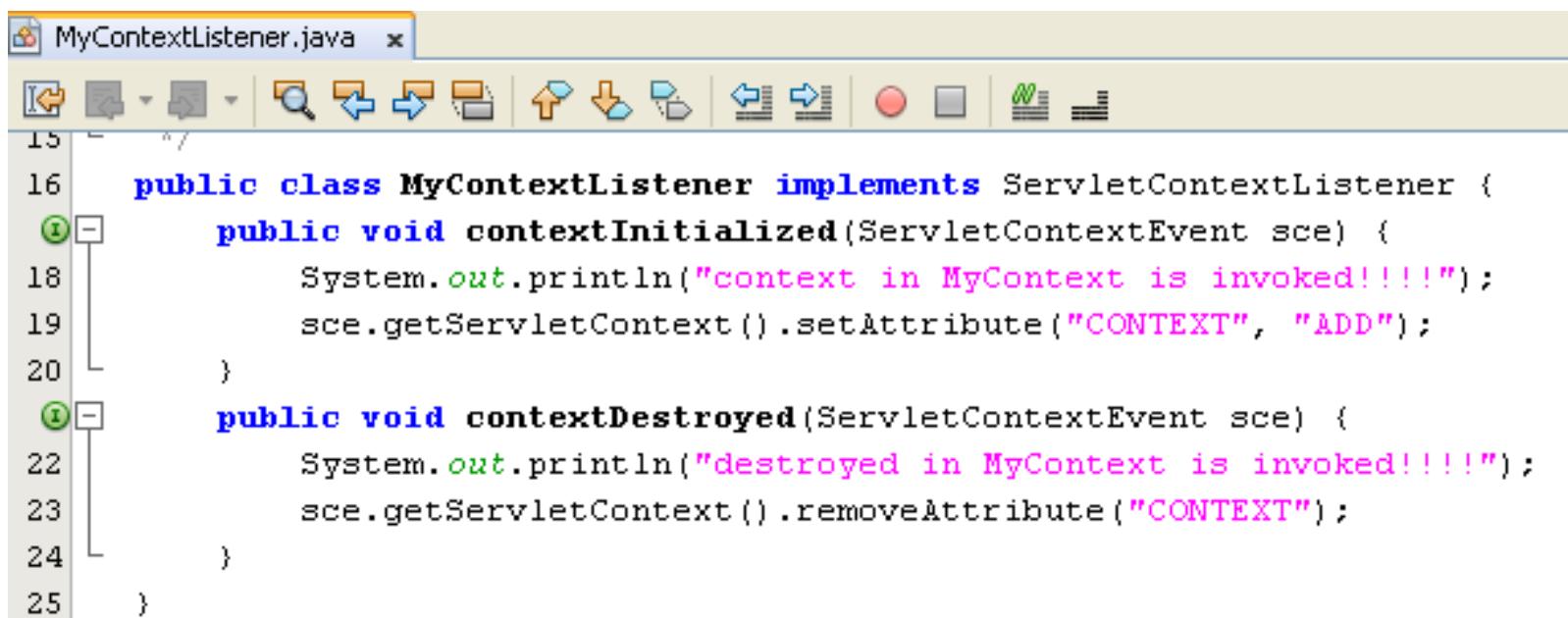
# Appendix

## Context Listener

- Sessions have 02 listeners:
  - **ServletContextListener**
    - Receive notifications about **changes** to the  **servlet context** of the Web application
      - **contextInitialized()**: gets **called before** any servlet's **init()** method or any filter's **doFilter()** method
      - **contextDestroyed()**: gets called **after** the **servlet's** or **filter's** **destroy()** method
      - Both of methods get passed a **ServletContextEvent** object that provides the **getServletContext()** method
    - **ServletContextAttributeListener**
      - Recieves a **notification** about any **modifications** made to the **attribute list** on the servlet context of a web application
      - Has the same trio of methods as **ServletRequestAttributeListener**

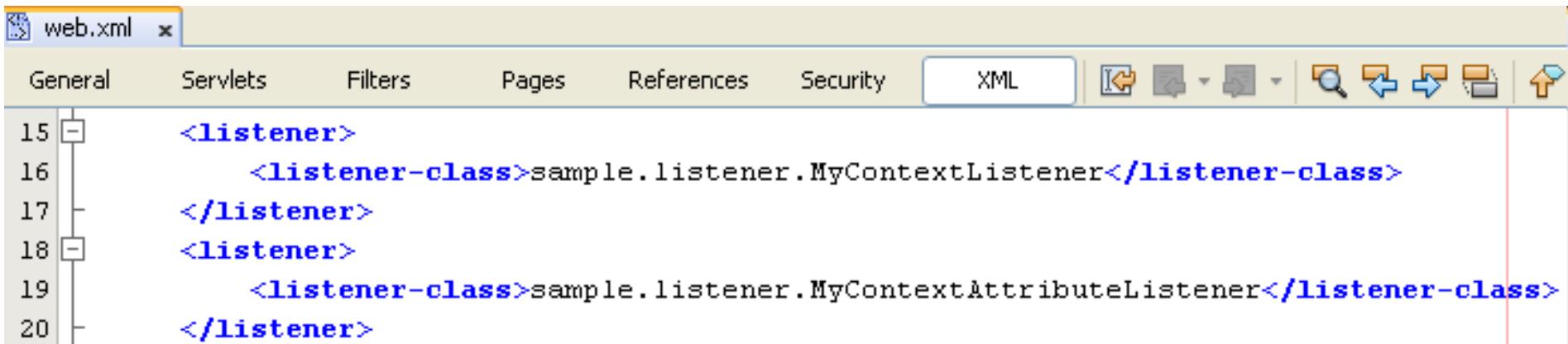
# Appendix

## Example



MyContextListener.java

```
15  /*
16   * public class MyContextListener implements ServletContextListener {
17   *     public void contextInitialized(ServletContextEvent sce) {
18   *         System.out.println("context in MyContext is invoked!!!!");
19   *         sce.getServletContext().setAttribute("CONTEXT", "ADD");
20   *     }
21   *     public void contextDestroyed(ServletContextEvent sce) {
22   *         System.out.println("destroyed in MyContext is invoked!!!!");
23   *         sce.getServletContext().removeAttribute("CONTEXT");
24   *     }
25 }
```

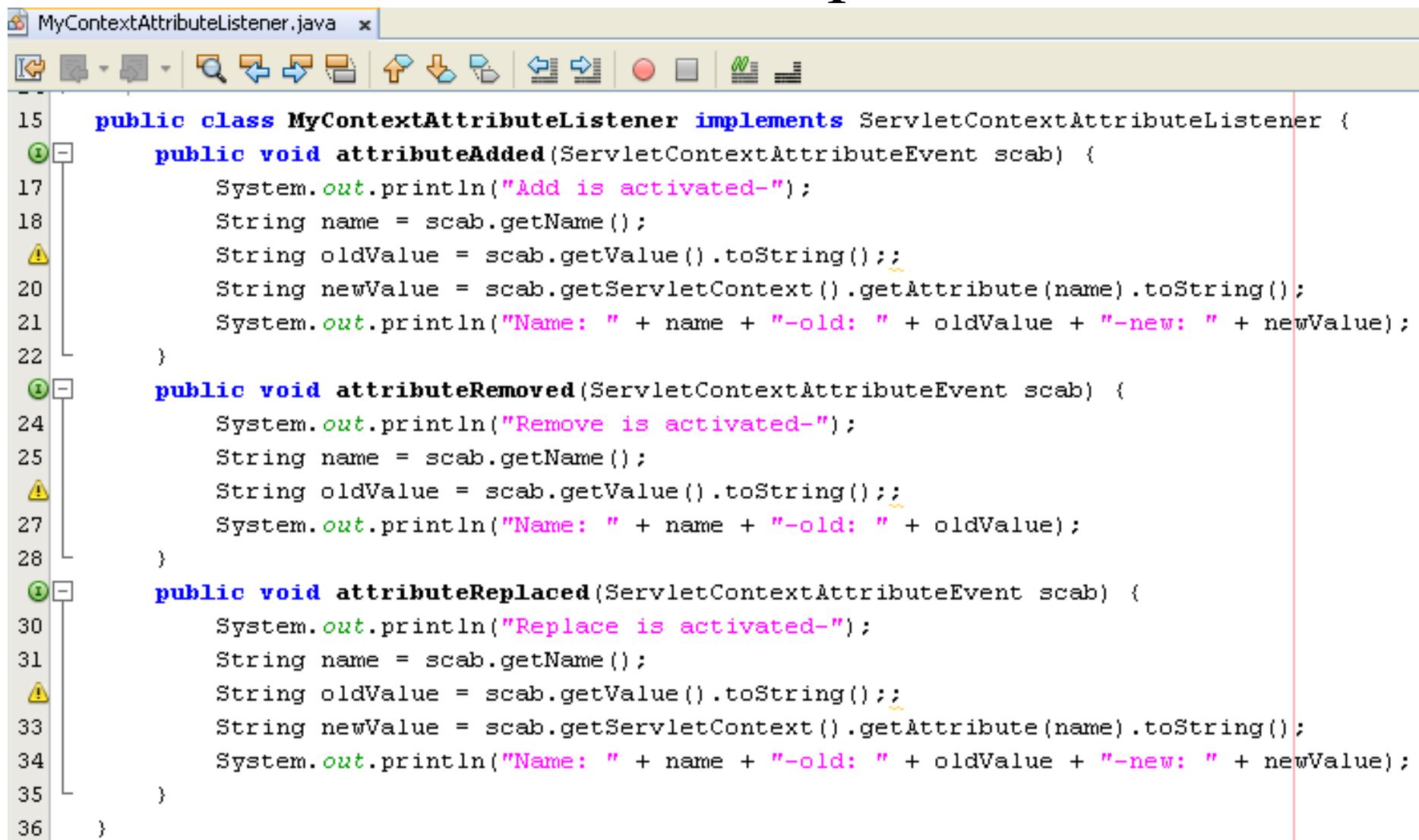


General Servlets Filters Pages References Security XML

```
15 <listener>
16   <listener-class>sample.listener.MyContextListener</listener-class>
17 </listener>
18 <listener>
19   <listener-class>sample.listener.MyContextAttributeListener</listener-class>
20 </listener>
```

# Appendix

## Example

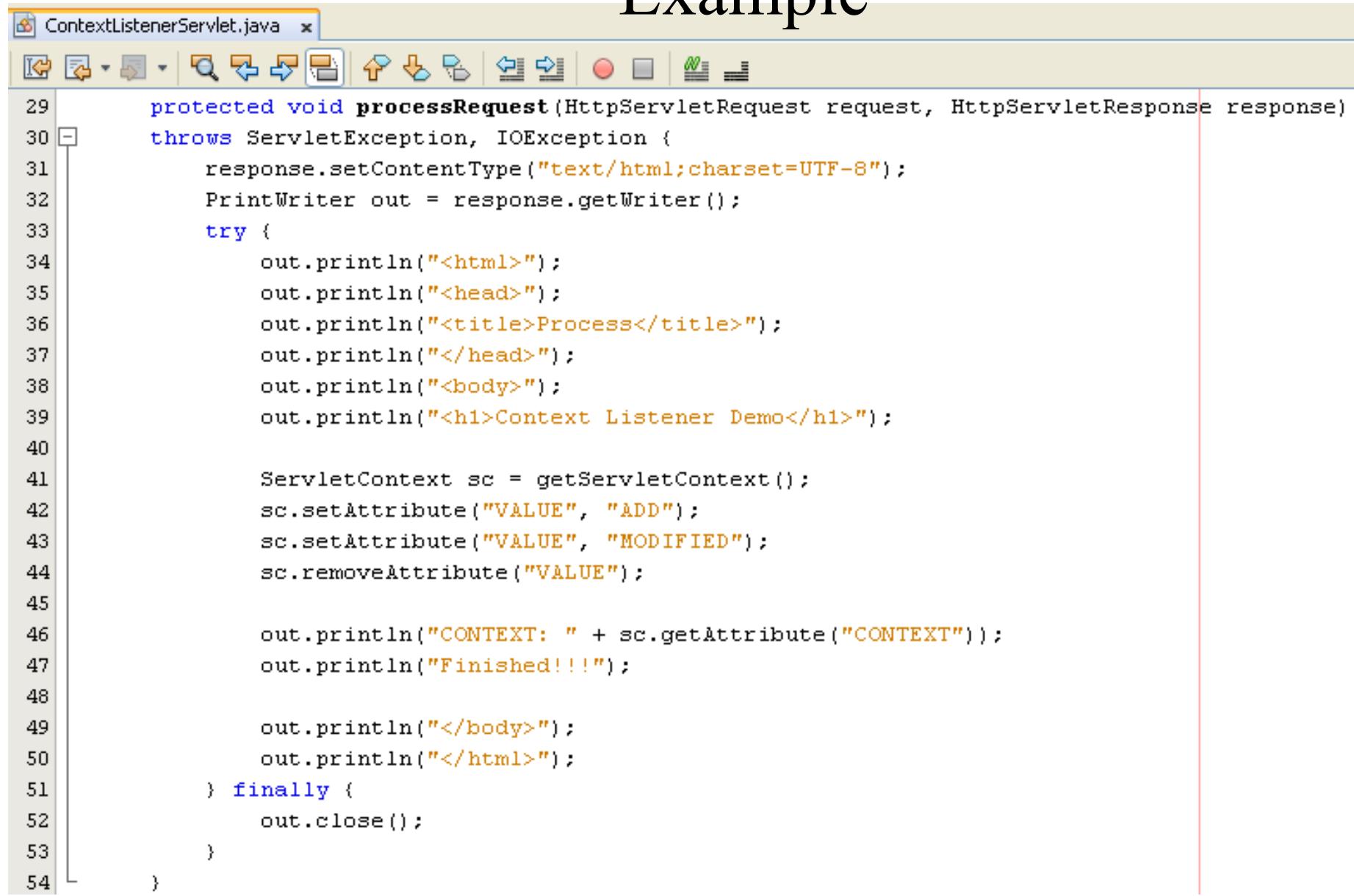


The screenshot shows a Java code editor with the file `MyContextAttributeListener.java` open. The code implements the `ServletContextAttributeListener` interface, handling three events: attributeAdded, attributeRemoved, and attributeReplaced. It prints the name of the attribute, its old value, and its new value to the console.

```
15  public class MyContextAttributeListener implements ServletContextAttributeListener {  
16      public void attributeAdded(ServletContextAttributeEvent scab) {  
17          System.out.println("Add is activated-");  
18          String name = scab.getName();  
19          String oldValue = scab.getValue().toString();  
20          String newValue = scab.getServletContext().getAttribute(name).toString();  
21          System.out.println("Name: " + name + "-old: " + oldValue + "-new: " + newValue);  
22      }  
23      public void attributeRemoved(ServletContextAttributeEvent scab) {  
24          System.out.println("Remove is activated-");  
25          String name = scab.getName();  
26          String oldValue = scab.getValue().toString();  
27          System.out.println("Name: " + name + "-old: " + oldValue);  
28      }  
29      public void attributeReplaced(ServletContextAttributeEvent scab) {  
30          System.out.println("Replace is activated-");  
31          String name = scab.getName();  
32          String oldValue = scab.getValue().toString();  
33          String newValue = scab.getServletContext().getAttribute(name).toString();  
34          System.out.println("Name: " + name + "-old: " + oldValue + "-new: " + newValue);  
35      }  
36  }
```

# Appendix

## Example



The screenshot shows a Java code editor with the file "ContextListenerServlet.java" open. The code implements a servlet context listener. It sets the response content type to "text/html; charset=UTF-8", prints an HTML page with a title and body containing "Process Listener Demo", and interacts with the servlet context by adding, modifying, and removing attributes named "VALUE". Finally, it prints the value of the "CONTEXT" attribute and closes the output stream.

```
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>Process</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Context Listener Demo</h1>");

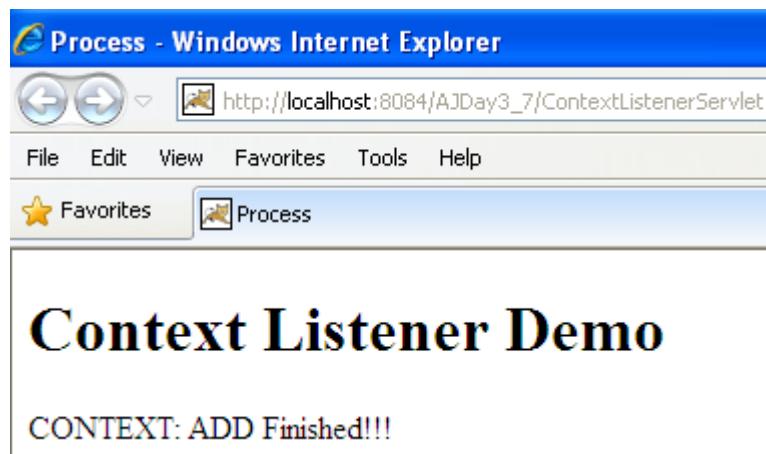
        ServletContext sc = getServletContext();
        sc.setAttribute("VALUE", "ADD");
        sc.setAttribute("VALUE", "MODIFIED");
        sc.removeAttribute("VALUE");

        out.println("CONTEXT: " + sc.getAttribute("CONTEXT"));
        out.println("Finished!!!!");

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

# Appendix

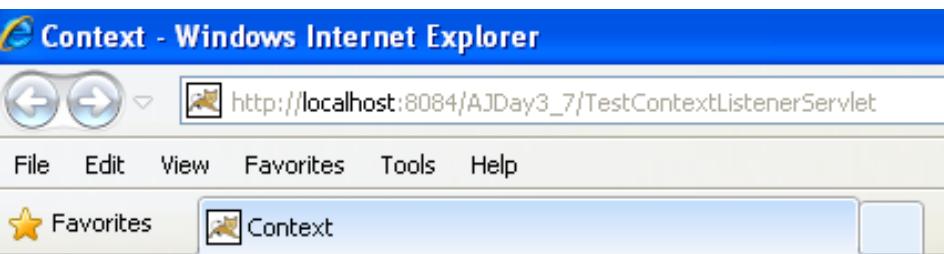
## Example



```
context in MyContext is invoked!!!!
Add is activated-
Name: CONTEXT-old: ADD-new: ADD
Add is activated-
Name: org.apache.jasper.compiler.TldLocationsCache-old: org.apache.jasper..i
Add is activated
Name: VALID -old: false -new: false
Replace is activated
Name: org.apache.catalina.ASYNC_SUPPORTED -old: true -new: false
Add is activated
Name: netbeans.monitor.request -old: uri: /AJDay3_7/ContextListenerServlet
method: GET
QueryString: null
Parameters:
Headers:
```

# Appendix

## Practices – Example



### Test Context Listener Servlet

Test DB Connection is Closed: false



```
14
15     * @author Trong Khanh
16     */
17     public class MyContextListener implements ServletContextListener {
18
19         /**
20          * @param sce
21          */
22         public void contextInitialized(ServletContextEvent sce) {
23             Connection con = DBUtils.makeConnection();
24             sce.getServletContext().setAttribute("CON", con);
25         }
26
27         /**
28          * @param sce
29         */
30         public void contextDestroyed(ServletContextEvent sce) {
31             ...
32         }
33     }
34 }
```

# Appendix

## Practices – Example

# Appendix

## Session Listeners Declared in DD

- Have **02 listeners**:
  - **HttpSessionListener**
    - Implements the changes to the list of active sessions in Web application
    - **sessionCreated()** method: is called whenever a **new session** is provided (*can say that after the **getSession()** method*)
    - **sessionDestroyed()**: is called at the **end of the sessions** (*within the call **invalidate()** or session time out but before the session become invalid*)
    - Both of methods get passed a **HttpSessionEvent object** that provides the **getSession()** method
  - **HttpSessionAttributeListener**
    - Is **called** whenever **some changes** are made to the **attribute** list on the **servlet session** of a Web application
    - Is used to **notify when** an **attribute** has been **added, removed or replaced by another attribute**
    - Has the **same trio of** methods as **ServletRequestAttributeListener** that are passed the **HttpSessionBindingEvent** (is inherited from **HttpSessionEvent**)

# Appendix

## Practices – Example

 Login - Windows Internet Explorer

http://localhost:8084/AJDay3\_7/

File Edit View Favorites Tools Help

Favorites Login

### Login Page

Username

Password

 Result - Windows Internet Explorer

http://localhost:8084/AJDay3\_7/SessionListenerServlet

File Edit View Favorites Tools Help

Favorites Result

### Session Listener Demo

### Welcome, khanh

This website is accessed: 1 times

 Login - Windows Internet Explorer

http://localhost:8084/AJDay3\_7/

File Edit View Favorites Tools Help

Favorites Login

### Login Page

Username

Password

 Result - Windows Internet Explorer

http://localhost:8084/AJDay3\_7/SessionListenerServlet

File Edit View Favorites Tools Help

Favorites Result

### Session Listener Demo

### Welcome, test

This website is accessed: 2 times

# Appendix

## Practices – Example

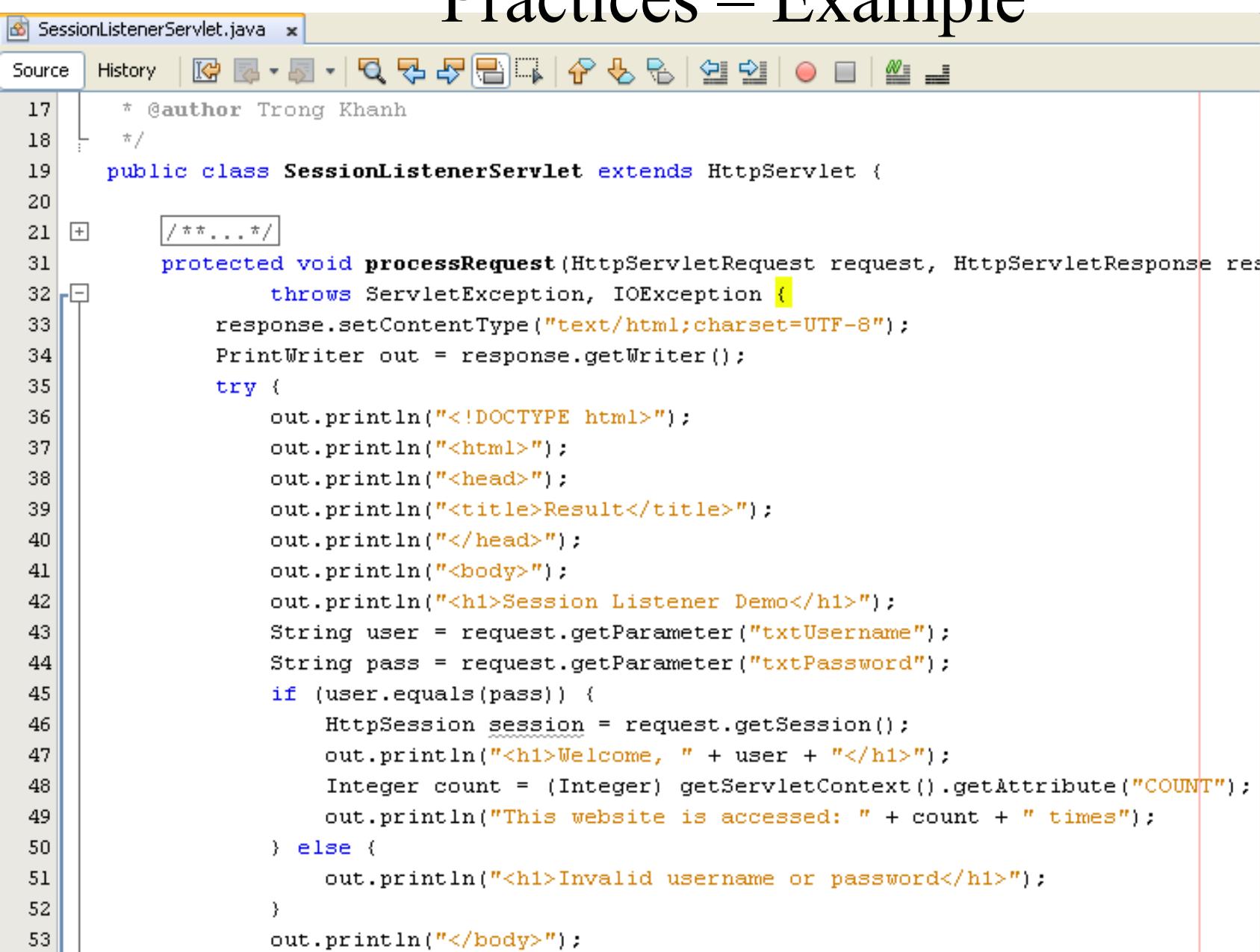
loginCountSession.html

Source History

```
5   <!DOCTYPE html>
6   <html>
7       <head>
8           <title>Login</title>
9           <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10      </head>
11      <body>
12          <h1>Login Page</h1>
13          <form action="SessionListenerServlet" method="POST">
14              Username <input type="text" name="txtUsername" value="" /><br/>
15              Password <input type="password" name="txtPassword" value="" /><br/>
16              <input type="submit" value="Login" name="btAction" />
17              <input type="reset" value="Reset" />
18          </form>
19      </body>
20  </html>
```

# Appendix

## Practices – Example



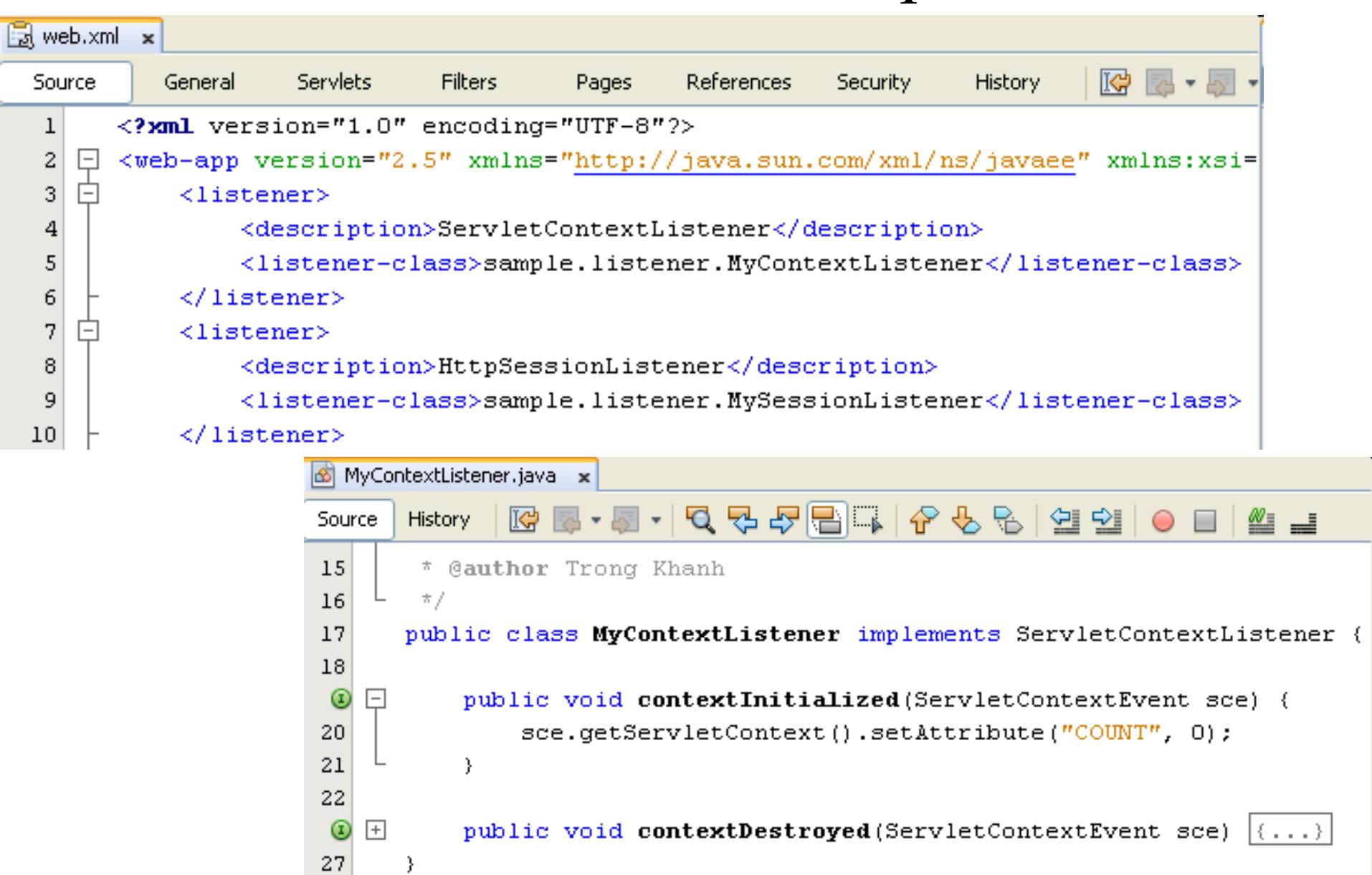
The screenshot shows a Java code editor with the file `SessionListenerServlet.java` open. The code implements a `HttpServlet` that handles session listeners. It prints an HTML page with a welcome message and the count of website accesses.

```
SessionListenerServlet.java
Source History | Back Forward | Find | Replace | Open | Save | Print | Copy | Paste | Cut | Delete | Select All | Undo | Redo | New | Open In Browser | Run | Stop | Refresh | Help | Exit |
```

```
17  * @author Trong Khanh
18  */
19  public class SessionListenerServlet extends HttpServlet {
20
21  /**
22  */
23  protected void processRequest(HttpServletRequest request, HttpServletResponse res
24  throws ServletException, IOException {
25      response.setContentType("text/html;charset=UTF-8");
26      PrintWriter out = response.getWriter();
27      try {
28          out.println("<!DOCTYPE html>");
29          out.println("<html>");
30          out.println("<head>");
31          out.println("<title>Result</title>");
32          out.println("</head>");
33          out.println("<body>");
34          out.println("<h1>Session Listener Demo</h1>");
35          String user = request.getParameter("txtUsername");
36          String pass = request.getParameter("txtPassword");
37          if (user.equals(pass)) {
38              HttpSession session = request.getSession();
39              out.println("<h1>Welcome, " + user + "</h1>");
40              Integer count = (Integer) getServletContext().getAttribute("COUNT");
41              out.println("This website is accessed: " + count + " times");
42          } else {
43              out.println("<h1>Invalid username or password</h1>");
44          }
45          out.println("</body>");
46      } catch (Exception e) {
47          e.printStackTrace();
48      }
49  }
50  
```

# Appendix

## Practices – Example



The screenshot shows a Java Development Environment (IDE) interface with two open files:

- web.xml**: A configuration file for a Java Web Application. It defines two listeners: `ServletContextListener` and `HttpSessionListener`, both implemented by the class `sample.listener.MyContextListener`. The XML code is as follows:

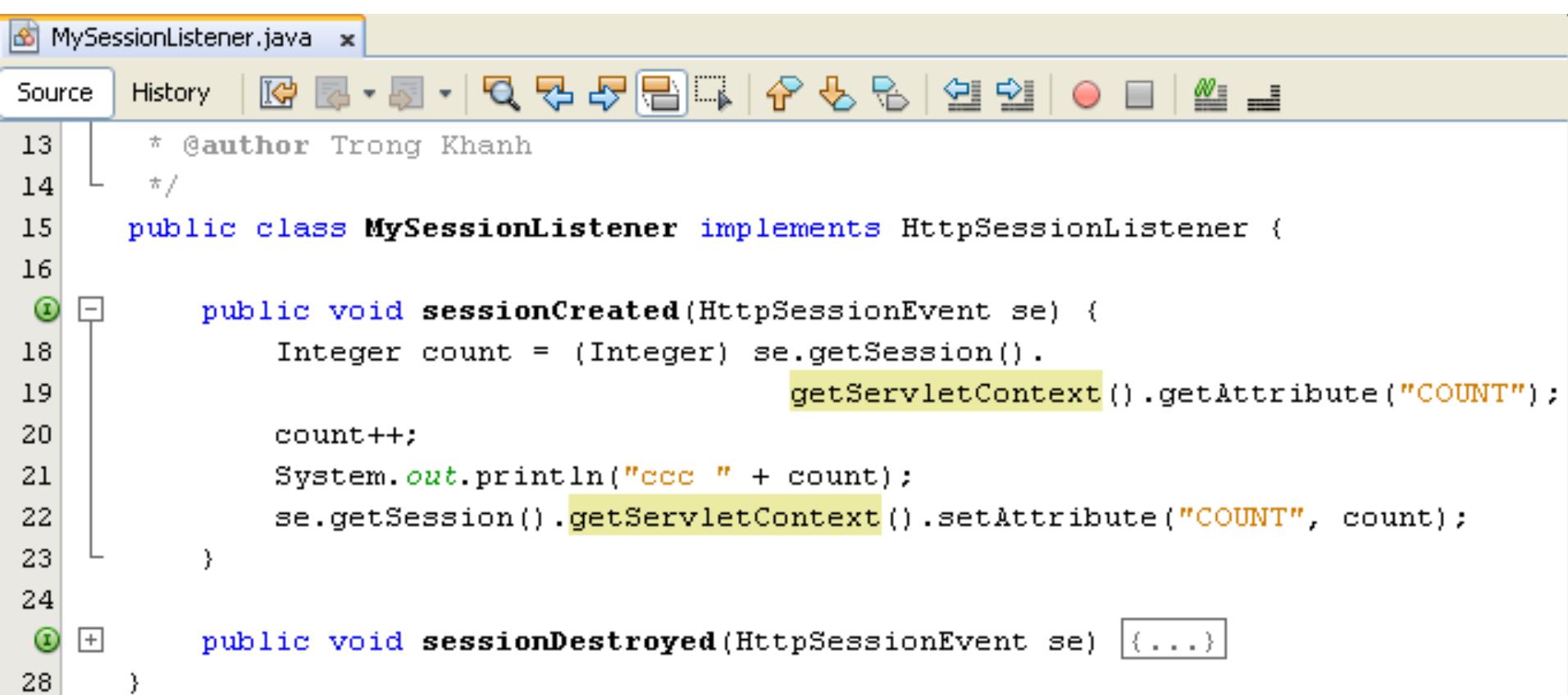
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi=
  <listener>
    <description>ServletContextListener</description>
    <listener-class>sample.listener.MyContextListener</listener-class>
  </listener>
  <listener>
    <description>HttpSessionListener</description>
    <listener-class>sample.listener.MySessionListener</listener-class>
  </listener>
```

- MyContextListener.java**: A Java source code file containing the implementation of the `ServletContextListener` interface. The code includes annotations for the author and implements the required methods `contextInitialized` and `contextDestroyed`.

```
15 * @author Trong Khanh
16 */
17 public class MyContextListener implements ServletContextListener {
18
19     public void contextInitialized(ServletContextEvent sce) {
20         sce.getServletContext().setAttribute("COUNT", 0);
21     }
22
23     public void contextDestroyed(ServletContextEvent sce) { ... }
24 }
```

# Appendix

## Practices – Example



The screenshot shows a Java code editor window titled "MySessionListener.java". The code implements the HttpSessionListener interface. It contains two methods: sessionCreated and sessionDestroyed. The sessionCreated method increments a counter and prints it to the console. The sessionDestroyed method is currently empty. The code is annotated with comments and uses standard Java syntax.

```
13 * @author Trong Khanh
14 */
15 public class MySessionListener implements HttpSessionListener {
16
17     public void sessionCreated(HttpSessionEvent se) {
18         Integer count = (Integer) se.getSession().
19                         getServletContext().getattribute("COUNT");
20         count++;
21         System.out.println("ccc " + count);
22         se.getSession().getServletContext().setAttribute("COUNT", count);
23     }
24
25     public void sessionDestroyed(HttpSessionEvent se) { ... }
26 }
```

# Appendix

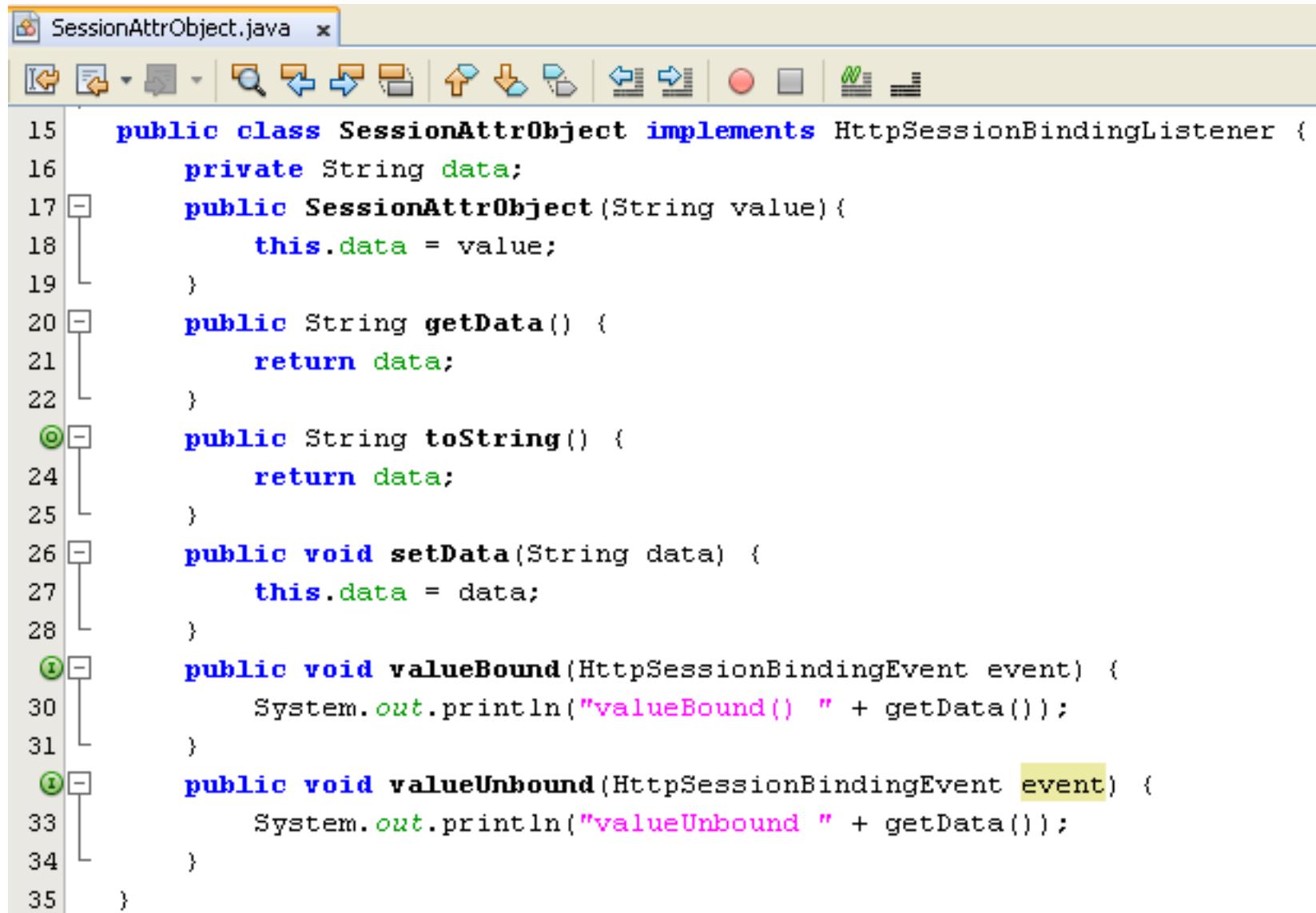
## Session Listeners Not Declared in DD

- Have **02 listeners:**
  - **HttpSessionBindingListener**
    - **Notifies** the **object** when it is being **bound** to or **unbound** from a **session**
    - This **notification** can be the **result** of a **forced unbinding** of an **attribute** from a **session** by **the programmer**, **invalidation of the session** or due to **timing out of session**
    - This **implementation** do **not require** any **configuration** within the deployment descriptor of the Web application
    - **Notes:** The object data types not implemented in **BindingListener** don't fire any events!

Methods	Descriptions
<b>valueBound</b>	<ul style="list-style-type: none"> <li>- <b>public void valueBound(HttpSessionBindingEvent se);</b></li> <li>- <b>Notifies</b> the object in being <b>bound</b> to a <b>session</b> and is responsible for identification of the session</li> </ul>
<b>valueUnbound</b>	<ul style="list-style-type: none"> <li>- <b>public void valueUnbound(HttpSessionBindingEvent se);</b></li> <li>- <b>Notifies</b> the <b>object</b> on being <b>unbound</b> from a <b>session</b> and is responsible for identification of the session</li> </ul>

# Appendix

## Session Listeners Not Declared in DD - Example

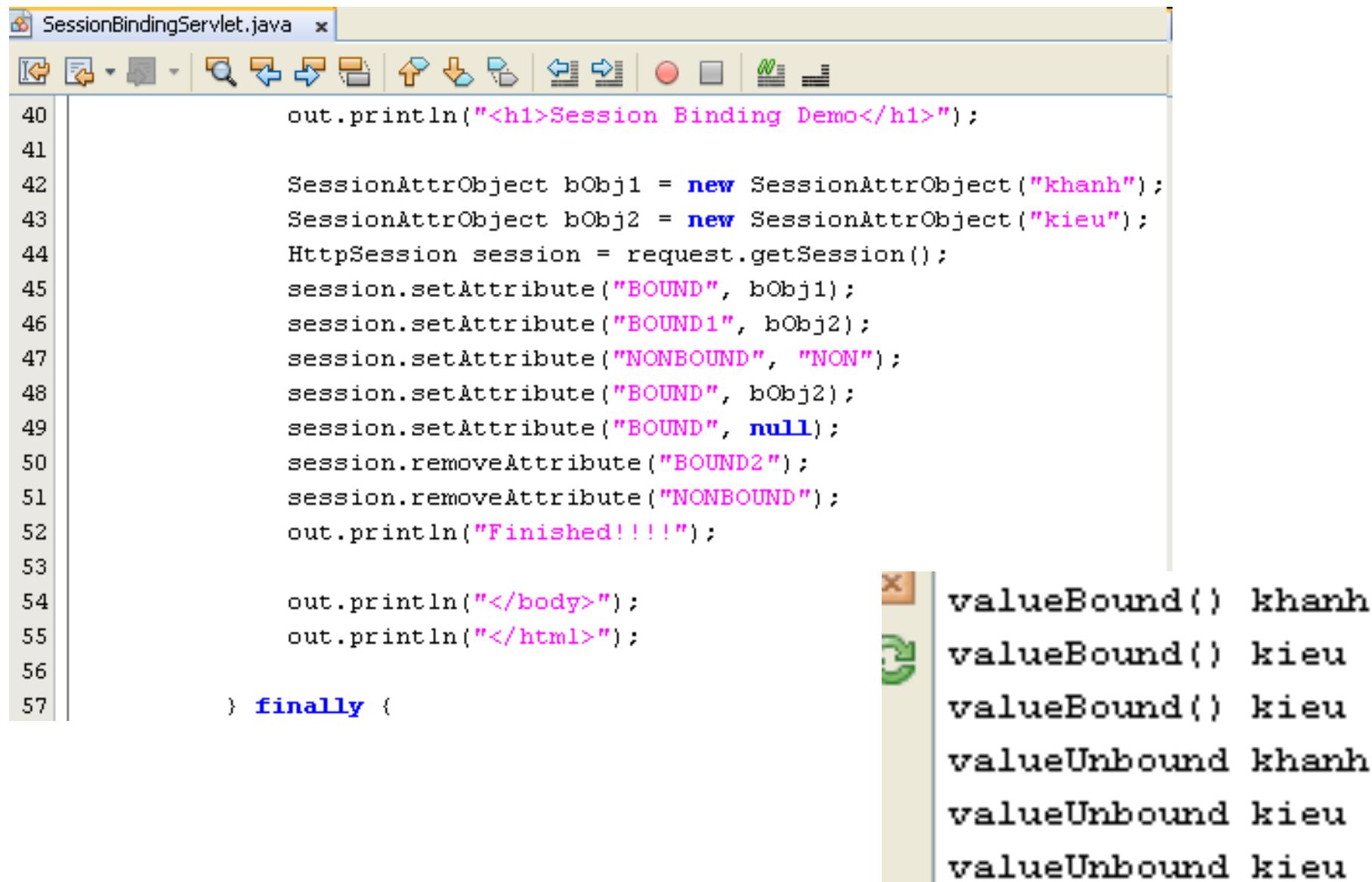


The screenshot shows a Java code editor window with the file "SessionAttrObject.java" open. The code implements the HttpSessionBindingListener interface. It contains private data, constructor, getters, setters, and two listener methods: valueBound and valueUnbound, each printing the current session value to the console.

```
15  public class SessionAttrObject implements HttpSessionBindingListener {  
16      private String data;  
17      public SessionAttrObject(String value){  
18          this.data = value;  
19      }  
20      public String getData(){  
21          return data;  
22      }  
23      public String toString(){  
24          return data;  
25      }  
26      public void setData(String data){  
27          this.data = data;  
28      }  
29      public void valueBound(HttpSessionBindingEvent event){  
30          System.out.println("valueBound() " + getData());  
31      }  
32      public void valueUnbound(HttpSessionBindingEvent event){  
33          System.out.println("valueUnbound " + getData());  
34      }  
35  }
```

# Appendix

## Session Listeners Not Declared in DD - Example



The screenshot shows an IDE interface with a Java file named `SessionBindingServlet.java` open. The code demonstrates session attribute binding and unbinding. On the right, the execution output shows the values of session listeners.

```
SessionBindingServlet.java
```

```
40     out.println("<h1>Session Binding Demo</h1>");
41
42     SessionAttrObject bObj1 = new SessionAttrObject("khanh");
43     SessionAttrObject bObj2 = new SessionAttrObject("kieu");
44     HttpSession session = request.getSession();
45     session.setAttribute("BOUND", bObj1);
46     session.setAttribute("BOUND1", bObj2);
47     session.setAttribute("NONBOUND", "NON");
48     session.setAttribute("BOUND", bObj2);
49     session.setAttribute("BOUND", null);
50     session.removeAttribute("BOUND2");
51     session.removeAttribute("NONBOUND");
52     out.println("Finished!!!!");
53
54     out.println("</body>");
55     out.println("</html>");
56
57 } finally {
```

Action	Value
valueBound()	khanh
valueBound()	kieu
valueBound()	kieu
valueUnbound	khanh
valueUnbound	kieu
valueUnbound	kieu

# Appendix

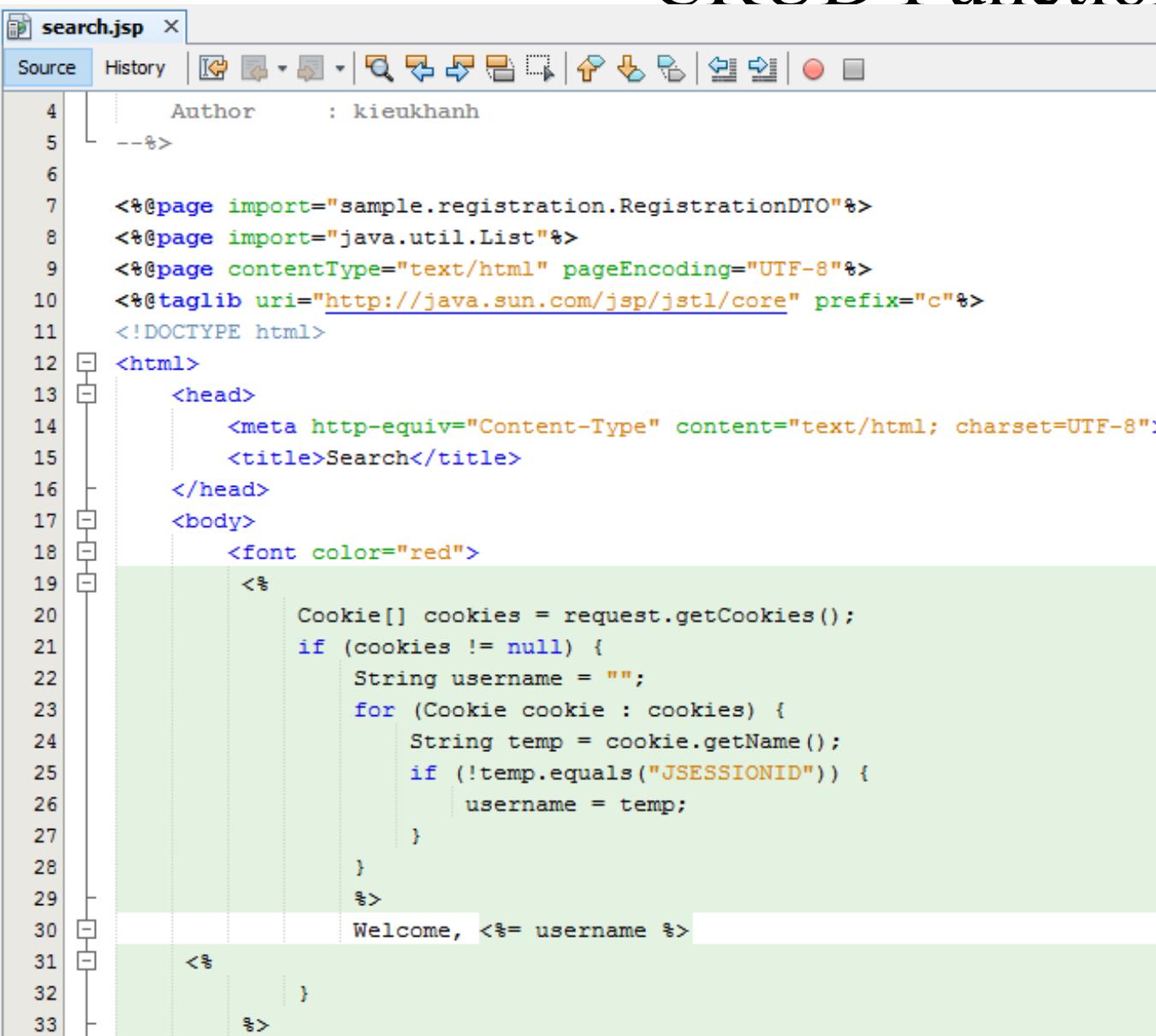
## Session Listeners Not Declared in DD

- Have **02 listeners** (cont)
  - **HttpSessionActivationListener** (*receives events when a value object is transported across JVMs*).
    - **Stateful** session (activated and passivated)
    - Is **implemented** when a **container migrates** the **session between VM** or **persists sessions** and is **not required** any **configuration within the deployment descriptor**

Methods	Descriptions
<b>sessionDidActivate</b>	<ul style="list-style-type: none"> <li>- <b>public void sessionDidActivate(HttpSessionEvent se);</b></li> <li>- Provides notification that the session has just been activated.</li> </ul>
<b>sessionWillPassivate</b>	<ul style="list-style-type: none"> <li>- <b>public void sessionWillPassivate(HttpSessionEvent se);</b></li> <li>- Provide notification that the session is about to be passivated.</li> </ul>

# Appendix

## CRUD Function



The screenshot shows a Java Server Page (JSP) editor with the file name "search.jsp". The code is written in JSP syntax, combining HTML, Java scriptlets, and JSTL tags.

```
4     Author      : kieukhanh
5     --%>
6
7     <%@page import="sample.registration.RegistrationDTO"%>
8     <%@page import="java.util.List"%>
9     <%@page contentType="text/html" pageEncoding="UTF-8"%>
10    <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
11    <!DOCTYPE html>
12    <html>
13        <head>
14            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15            <title>Search</title>
16        </head>
17        <body>
18            <font color="red">
19                <%
20                    Cookie[] cookies = request.getCookies();
21                    if (cookies != null) {
22                        String username = "";
23                        for (Cookie cookie : cookies) {
24                            String temp = cookie.getName();
25                            if (!temp.equals("JSESSIONID")) {
26                                username = temp;
27                            }
28                        }
29                    %>
30                Welcome, <%= username %>
31                <%
32                %>
33            </font>

```

The code includes imports for RegistrationDTO and List, sets the page encoding to UTF-8, and defines a JSTL core tag library. It starts with an HTML document type declaration and an HTML tag. Inside the body, it contains an  tag with red text. A scriptlet block (<%) retrieves cookies from the request. If there are any, it iterates through them to find a cookie named "JSESSIONID". If found, it sets the variable `username` to its value. Finally, it displays the welcome message followed by the `username` value.

# Appendix

## CRUD Function

```
35      </font>
36      <h1>Search Page</h1>
37      <form action="SE1162Servlet">
38          Search Value <input type="text" name="txtSearchValue"
39                      value="" /><br/>
40          <input type="submit" value="Search" name="btAction" />
41      </form>
42
43      <br/>
44
45      <%
46          String searchValue = request.getParameter("txtSearchValue");
47
48          if (searchValue != null) {
49              List<RegistrationDTO> result =
50                  (List<RegistrationDTO>) request.getAttribute("SEARCHRESULT");
51
52          if (result != null) {
53              %>
54              <table border="1">
55                  <thead>
56                      <tr>
57                          <th>No.</th>
58                          <th>Username</th>
59                          <th>Password</th>
60                          <th>Lastname</th>
61                          <th>Role</th>
62                          <th>Delete</th>
63                          <th>Update</th>
64                  </tr>
```

# Appendix

## CRUD Function

```
65 </thead>
66 <tbody>
67 <%
68     int count = 0;
69     for (RegistrationDTO dto : result) {
70         String urlRewriting = "SE1162Servlet?btAction=delete&pk="
71             + dto.getUsername()
72             + "&lastSearchValue="
73             + searchValue;
74     %>
75     <form action="SE1162Servlet">
76         <tr>
77             <td>
78                 <%= ++count %>
79             </td>
80             <td>
81                 <%= dto.getUsername() %>
82                 <input type="hidden" name="txtUsername"
83                     value="<%= dto.getUsername() %>" />
84             </td>
85             <td>
86                 <input type="text" name="txtPassword"
87                     value="<%= dto.getPassword() %>" />
88             </td>
89             <td>
90                 <%= dto.getLastname() %>
91             </td>
```

## Appendix

## CRUD Function

```
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
```

I

```
<td>
    <input type="checkbox" name="chkRole" value="ADMIN"
        <%
            if (dto.isRole()) {
                %>
                checked="checked"
        <%
        }
        %>
    />
</td>
<td>
    <a href="<% urlRewriting %>">Delete</a>
</td>
<td>
    <input type="hidden" name="lastSearchValue"
        value="<% searchValue %>" />
    <input type="submit" value="Update" name="btAction" />
</td>
</tr>
</form>
<%
    }
%>
</tbody>
</table>
```

<%

```
122
123
124
125
126
127
128
```

} else {

%>

```
<%
}
//end if search Value
%>
</body>
</html>
```

# Appendix

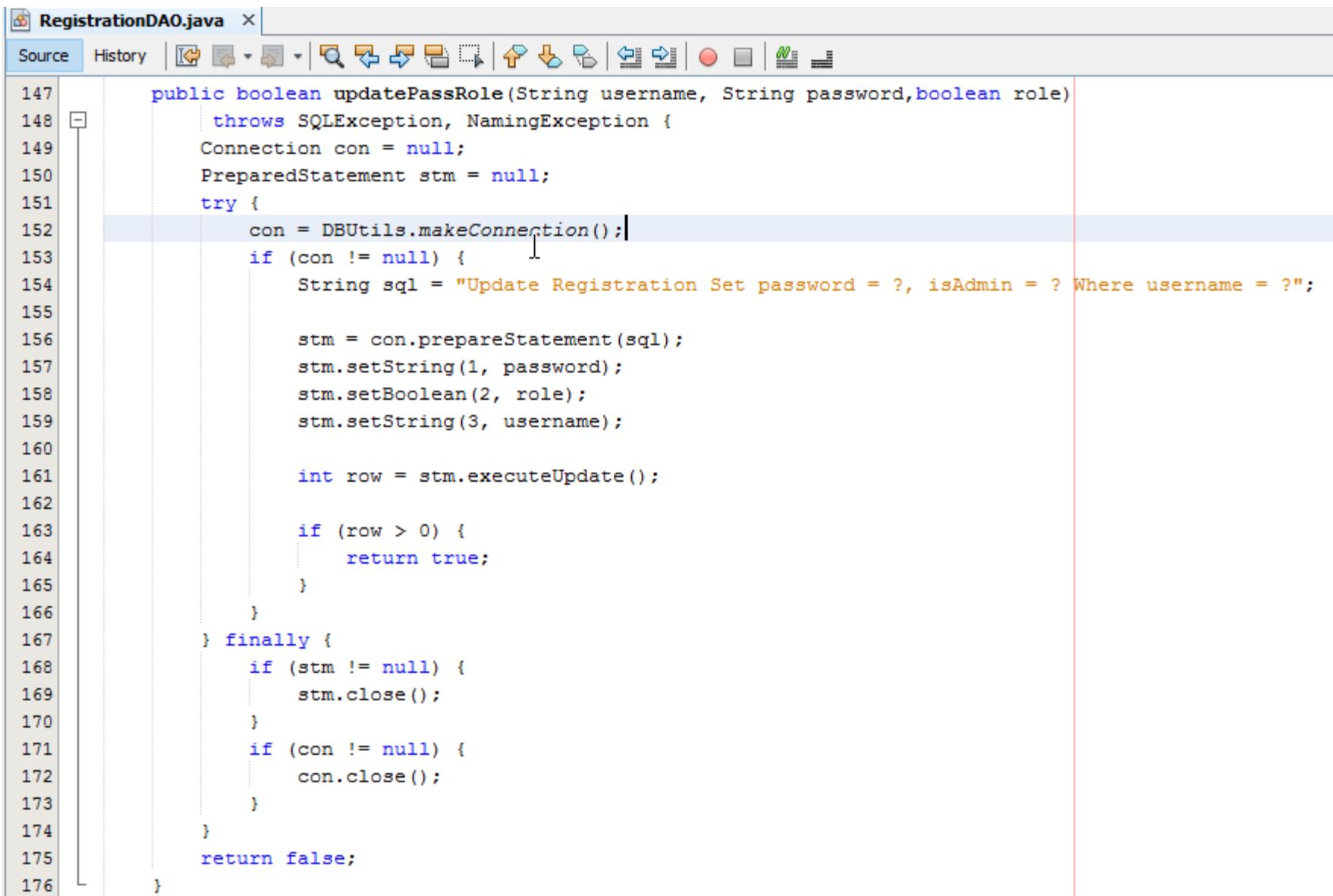
## CUD Function

The screenshot shows a Java code editor with the file `RegistrationDAO.java` open. The code implements a `deleteRecord` method that performs a delete operation on a database table named `Registration`. The code uses JDBC to establish a connection, prepare a statement, and execute the delete query. It handles resource cleanup by closing the statement and connection in the finally block. The code is annotated with line numbers from 113 to 142.

```
113     public boolean deleteRecord(String username)
114         throws SQLException, NamingException {
115     Connection con = null;
116     PreparedStatement stm = null;
117     try {
118         con = DBUtils.makeConnection();
119
120         if (con != null) {
121             String sql = "Delete From Registration Where username = ?";
122
123             stm = con.prepareStatement(sql);
124             stm.setString(1, username);
125
126             int row = stm.executeUpdate();
127
128             if (row > 0) {
129                 return true;
130             }
131         }
132     } finally {
133         if (stm != null) {
134             stm.close();
135         }
136         if (con != null) {
137             con.close();
138         }
139     }
140
141     return false;
142 }
```

# Appendix

## CRUD Function

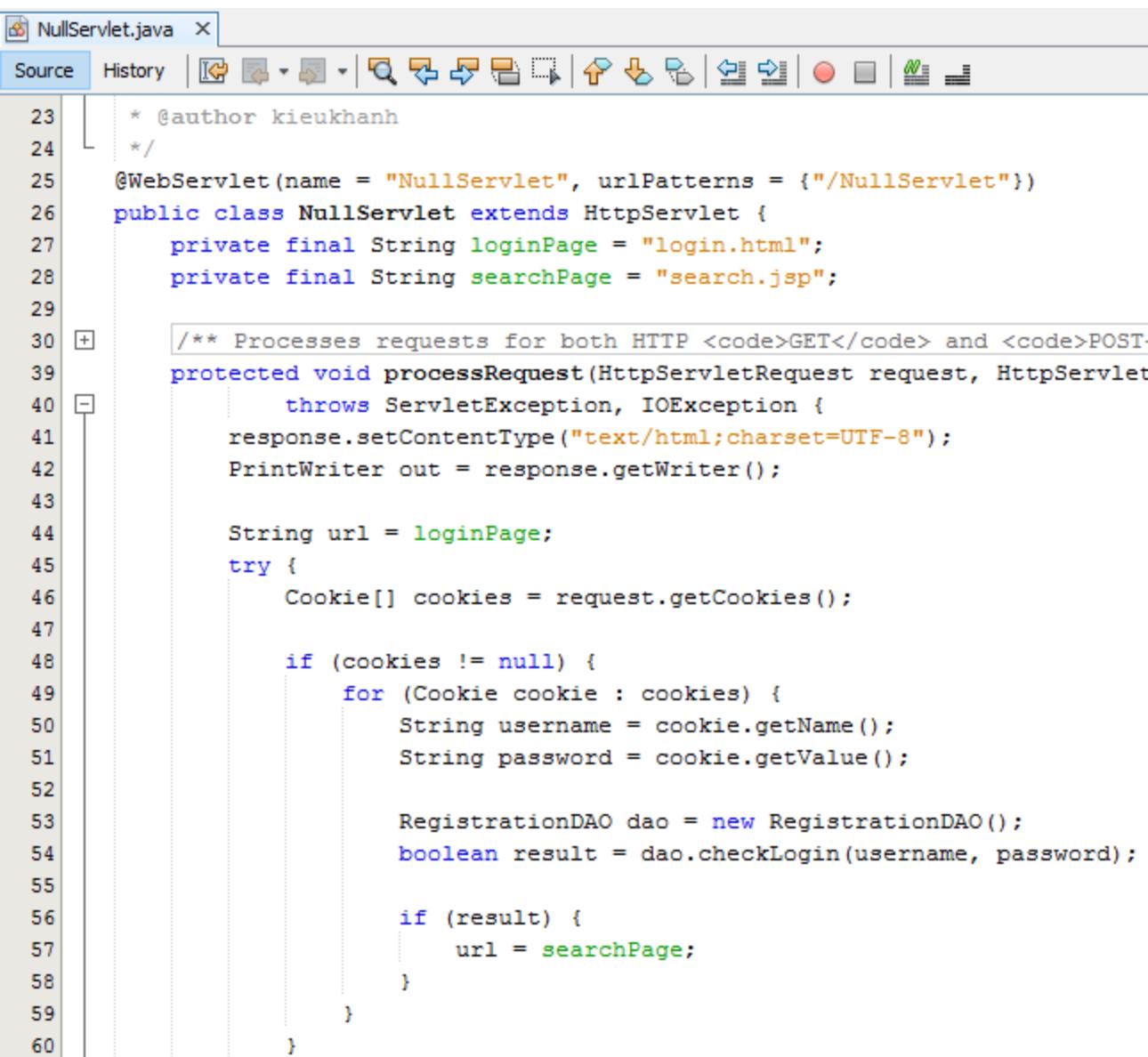


The screenshot shows a Java code editor with the file `RegistrationDAO.java` open. The code implements a `updatePassRole` method. It uses JDBC to connect to a database, prepare a statement, and execute an update query to change a user's password and role. If the update is successful, it returns `true`; otherwise, it returns `false`.

```
147     public boolean updatePassRole(String username, String password,boolean role)
148         throws SQLException, NamingException {
149     Connection con = null;
150     PreparedStatement stm = null;
151     try {
152         con = DBUtils.makeConnection();
153         if (con != null) {
154             String sql = "Update Registration Set password = ?, isAdmin = ? Where username = ?";
155
156             stm = con.prepareStatement(sql);
157             stm.setString(1, password);
158             stm.setBoolean(2, role);
159             stm.setString(3, username);
160
161             int row = stm.executeUpdate();
162
163             if (row > 0) {
164                 return true;
165             }
166         }
167     } finally {
168         if (stm != null) {
169             stm.close();
170         }
171         if (con != null) {
172             con.close();
173         }
174     }
175     return false;
176 }
```

# Appendix

## CRUD Function



The screenshot shows a Java code editor with the file `NullServlet.java` open. The code implements a `HttpServlet` that handles both GET and POST requests. It checks for cookies containing a username and password, then uses a `RegistrationDAO` to check if the login is valid. If valid, it sets the response URL to `search.jsp`.

```
23 * @author kieukhanh
24 */
25 @WebServlet(name = "NullServlet", urlPatterns = {"/NullServlet"})
26 public class NullServlet extends HttpServlet {
27     private final String loginPage = "login.html";
28     private final String searchPage = "search.jsp";
29
30     /**
31      * Processes requests for both HTTP <code>GET</code> and <code>POST<
32     */
33     protected void processRequest(HttpServletRequest request, HttpServletResponse
34             throws ServletException, IOException {
35         response.setContentType("text/html;charset=UTF-8");
36         PrintWriter out = response.getWriter();
37
38         String url = loginPage;
39         try {
40             Cookie[] cookies = request.getCookies();
41
42             if (cookies != null) {
43                 for (Cookie cookie : cookies) {
44                     String username = cookie.getName();
45                     String password = cookie.getValue();
46
47                     RegistrationDAO dao = new RegistrationDAO();
48                     boolean result = dao.checkLogin(username, password);
49
50                     if (result) {
51                         url = searchPage;
52                     }
53                 }
54             }
55         } finally {
56             out.close();
57         }
58     }
59
60 }
```

# Appendix

## CRUD Function

```
61
62      } catch (NamingException ex) {
63          ex.printStackTrace();
64      } catch (SQLException ex) {
65          ex.printStackTrace();
66      } finally {
67          RequestDispatcher rd = request.getRequestDispatcher(url);
68          rd.forward(request, response);
69
70          out.close();
71      }
72 }
```

## Appendix

# CRUD Function

The screenshot shows a Java code editor with the file `SE1162Servlet.java` open. The code defines a servlet that processes requests for various actions like login, search, and delete. It uses several private final variables to store the names of other servlets and pages. The `processRequest` method checks the value of the `btAction` parameter to determine which subsequent page or servlet to forward to.

```
18 * @author kieukhanh
19 */
20 public class SE1162Servlet extends HttpServlet {
21     private final String loginServlet = "LoginServlet";
22     private final String loginPage = "login.html";
23     private final String searchServlet = "SearchServlet";
24     private final String deleteRecordServlet = "DeleteRecordServlet";
25     private final String updatePassRoleServlet = "UpdatePassRoleServlet";
26     private final String nullServlet = "NullServlet";
27     private final String addItemServlet = "AddItemServlet";
28     private final String viewCartPage = "viewCart.jsp";
29     private final String deleteItemServlet = "DeleteItemServlet";
30     private final String createAccountServlet = "CreateAccountServlet";
31
32     /**
33      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
34      */
35     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
36             throws ServletException, IOException {
37         response.setContentType("text/html;charset=UTF-8");
38         PrintWriter out = response.getWriter();
39
40         String button = request.getParameter("btAction");
41         String url = loginPage;
42         try {
43             if (button == null) {
44                 url = nullServlet;
45             } else if (button.equals("Login")) {
46                 url = loginServlet;
47             } else if (button.equals("Search")) {
48                 url = searchServlet;
49             } else if (button.equals("delete")) {
50                 url = deleteRecordServlet;
51             } else if (button.equals("update")) {
52                 url = updatePassRoleServlet;
53             } else if (button.equals("addItem")) {
54                 url = addItemServlet;
55             } else if (button.equals("viewCart")) {
56                 url = viewCartPage;
57             } else if (button.equals("deleteItem")) {
58                 url = deleteItemServlet;
59             } else if (button.equals("createAccount")) {
60                 url = createAccountServlet;
61             }
62         } catch (IOException ex) {
63             ex.printStackTrace();
64         }
65     }
66
67     // Add your custom methods here
68 }
```

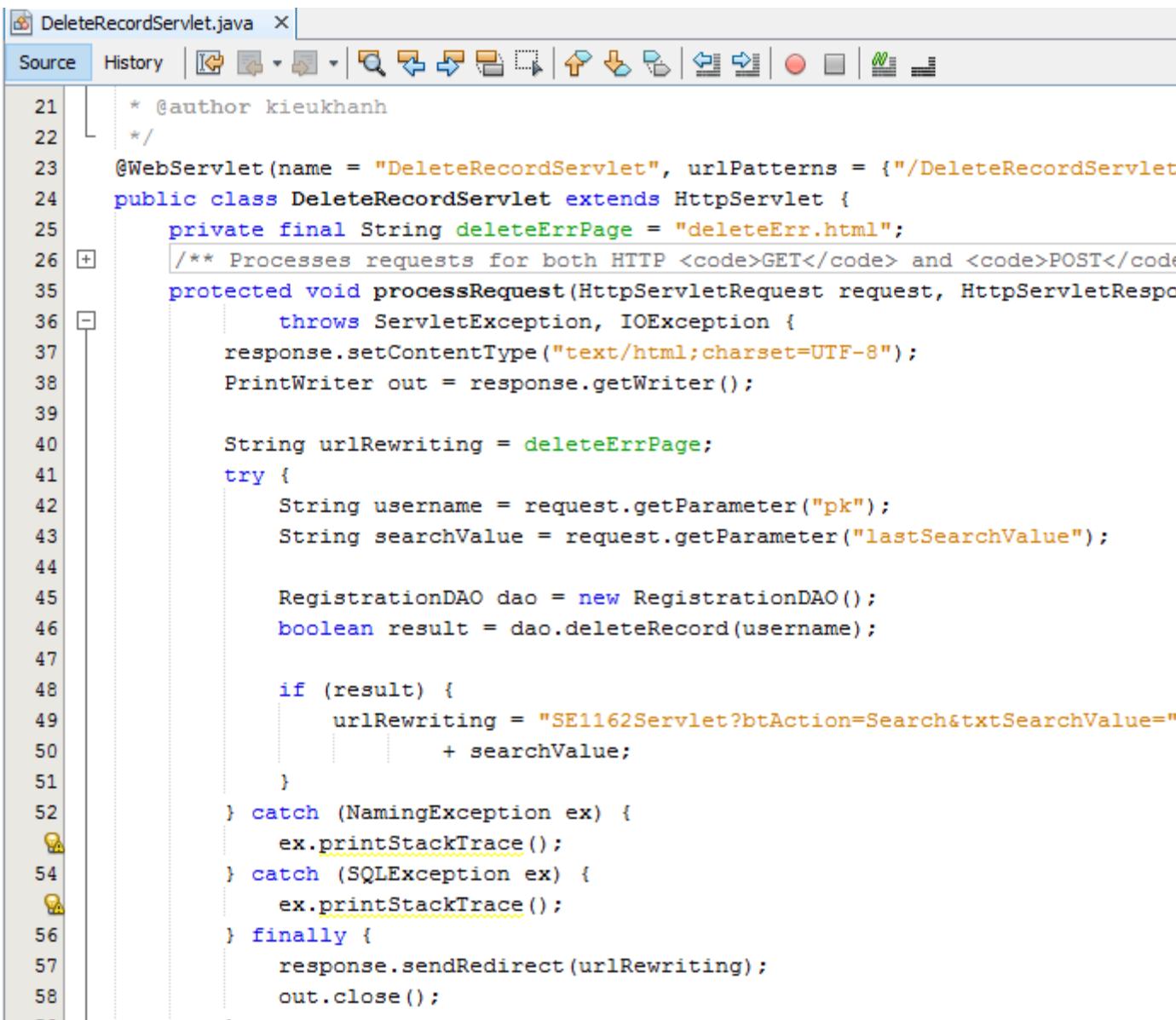
# Appendix

## CRUD Function

```
56 |         url = deleteRecordServlet;
57 |     } else if (button.equals("Update")) {
58 |         url = updatePassRoleServlet;
59 |     } else if (button.equals("Add Book To Your Cart")) {
60 |         url = addItemServlet;
61 |     } else if (button.equals("View Your Cart")) {
62 |         url = viewCartPage;
63 |     } else if (button.equals("Remove Selected Items")) {
64 |         url = deleteItemServlet;
65 |     } else if (button.equals("Create New Account")) {
66 |         url = createAccountServlet;
67 |
68 |
69 | } finally {
70 |     RequestDispatcher rd = request.getRequestDispatcher(url);
71 |     rd.forward(request, response);
72 |
73 |     out.close();
74 | }
75 }
```

# Appendix

## CRUD Function

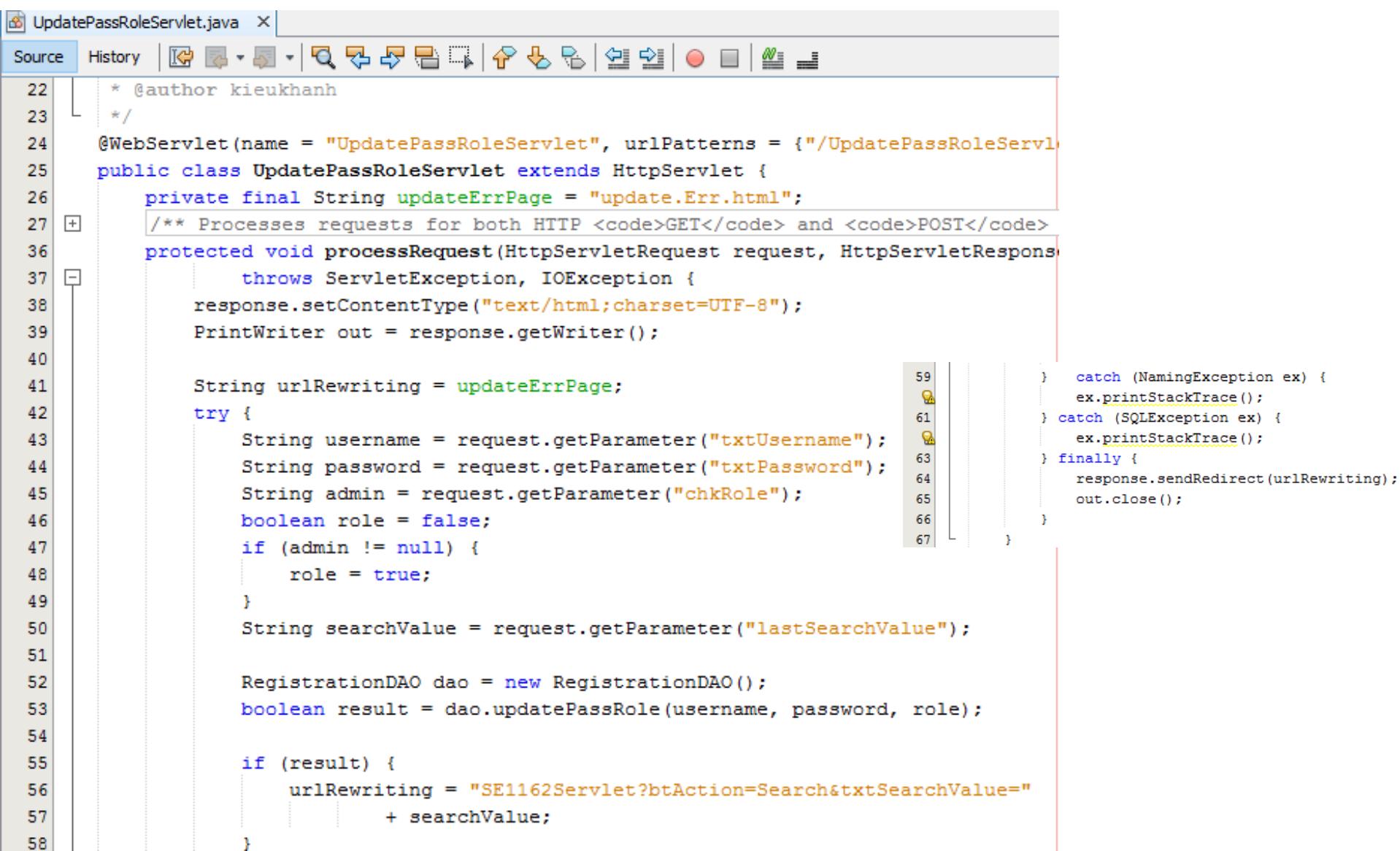


The screenshot shows a Java code editor with the file `DeleteRecordServlet.java` open. The code implements a `HttpServlet` for deleting records from a database. It uses `JSP` for URL rewriting and `RegistrationDAO` for database operations. The code includes error handling for `NamingException` and `SQLException`.

```
21  * @author kieukhanh
22  */
23  @WebServlet(name = "DeleteRecordServlet", urlPatterns = {" /DeleteRecordServlet"})
24  public class DeleteRecordServlet extends HttpServlet {
25      private final String deleteErrPage = "deleteErr.html";
26      /**
27       * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
28       */
29      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30          throws ServletException, IOException {
31          response.setContentType("text/html;charset=UTF-8");
32          PrintWriter out = response.getWriter();
33
34          String urlRewriting = deleteErrPage;
35          try {
36              String username = request.getParameter("pk");
37              String searchValue = request.getParameter("lastSearchValue");
38
39              RegistrationDAO dao = new RegistrationDAO();
40              boolean result = dao.deleteRecord(username);
41
42              if (result) {
43                  urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
44                      + searchValue;
45              }
46          } catch (NamingException ex) {
47              ex.printStackTrace();
48          } catch (SQLException ex) {
49              ex.printStackTrace();
50          } finally {
51              response.sendRedirect(urlRewriting);
52              out.close();
53          }
54      }
55  }
```

# Appendix

## CRUD Function

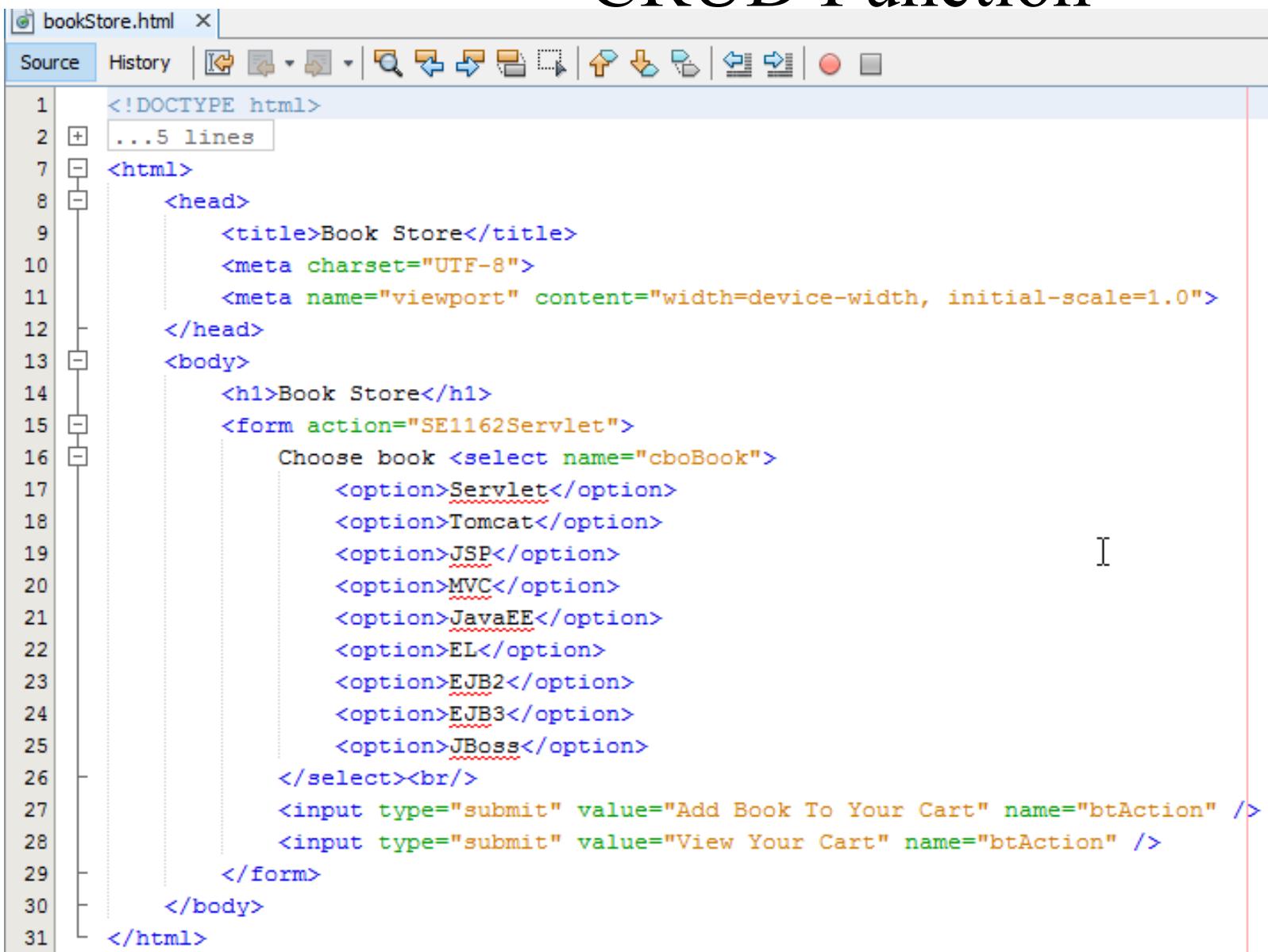


The screenshot shows a Java code editor with the file `UpdatePassRoleServlet.java` open. The code implements a `HttpServlet` for updating user roles. It uses `request.getParameter` to get user input and `RegistrationDAO` to perform the update operation. The code includes exception handling for `NamingException` and `SQLException`.

```
22 * @author kieukhanh
23 */
24 @WebServlet(name = "UpdatePassRoleServlet", urlPatterns = {"/UpdatePassRoleServlet"})
25 public class UpdatePassRoleServlet extends HttpServlet {
26     private final String updateErrPage = "update.Err.html";
27     /**
28      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
29      */
30     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31             throws ServletException, IOException {
32         response.setContentType("text/html;charset=UTF-8");
33         PrintWriter out = response.getWriter();
34
35         String urlRewriting = updateErrPage;
36         try {
37             String username = request.getParameter("txtUsername");
38             String password = request.getParameter("txtPassword");
39             String admin = request.getParameter("chkRole");
40             boolean role = false;
41             if (admin != null) {
42                 role = true;
43             }
44             String searchValue = request.getParameter("lastSearchValue");
45
46             RegistrationDAO dao = new RegistrationDAO();
47             boolean result = dao.updatePassRole(username, password, role);
48
49             if (result) {
50                 urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
51                             + searchValue;
52             }
53         } catch (NamingException ex) {
54             ex.printStackTrace();
55         } catch (SQLException ex) {
56             ex.printStackTrace();
57         } finally {
58             response.sendRedirect(urlRewriting);
59             out.close();
60         }
61     }
62 }
```

# Appendix

## CRUD Function



The screenshot shows a code editor window with the file "bookStore.html" open. The code is an HTML page for a book store. It includes a title, meta tags for charset and viewport, and a form with a dropdown menu for selecting a book. The dropdown menu lists various Java technologies: Servlet, Tomcat, JSP, MVC, JavaEE, EL, EJB2, EJB3, and JBoss. The code is color-coded for syntax highlighting.

```
<!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="SE1162Servlet">
            Choose book <select name="cboBook">
                <option>Servlet</option>
                <option>Tomcat</option>
                <option>JSP</option>
                <option>MVC</option>
                <option>JavaEE</option>
                <option>EL</option>
                <option>EJB2</option>
                <option>EJB3</option>
                <option>JBoss</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

## CRUD Function

CartObj.java X

Source History | 

```
14 * @author kieukhanh
15 */
16 public class CartObj implements Serializable {
17     private String customerId;
18     private Map<String, Integer> items;
19
20     public String getCustomerId() {
21         return customerId;
22     }
23
24     public void setCustomerId(String customerId) {
25         this.customerId = customerId;
26     }
27
28     public Map<String, Integer> getItems() {
29         return items;
30     }
31
32     public void addItemToCart(String title) {
33         if (this.items == null) {
34             this.items = new HashMap<String, Integer>();
35         }
36
37         int quantity = 1;
38         if (this.items.containsKey(title)) {
39             quantity = this.items.get(title) + 1;
40         }
41
42         this.items.put(title, quantity);
43     }
}
```

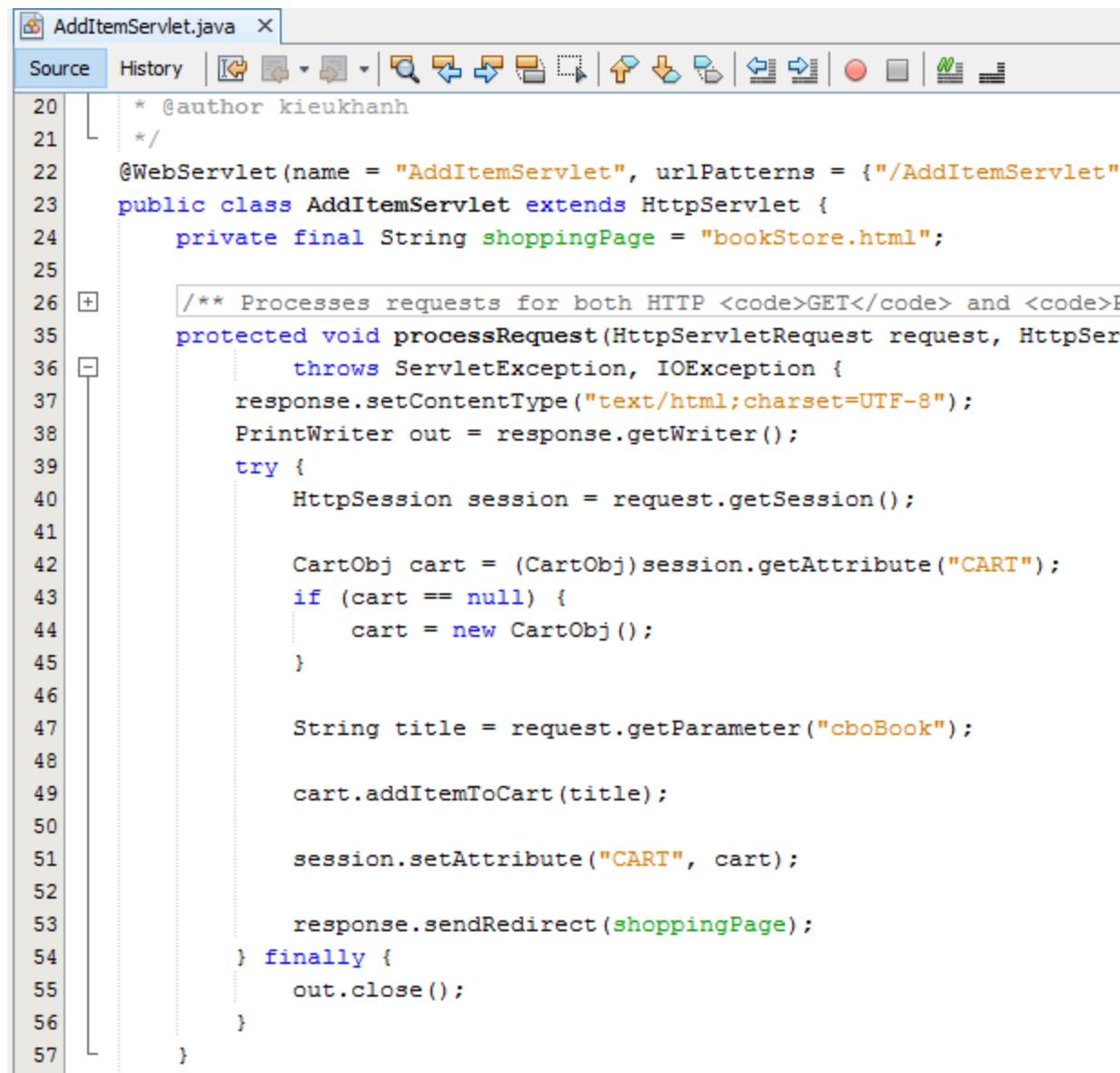
# Appendix

## CRUD Function

```
44
45     public void removeItemFromCart(String title) {
46         if (this.items == null) {
47             return;
48         }
49
50         if (this.items.containsKey(title)) {
51             this.items.remove(title);
52             if (this.items.isEmpty()) {
53                 this.items = null;
54             }
55         }
56     }
57 }
58 }
```

# Appendix

## CRUD Function

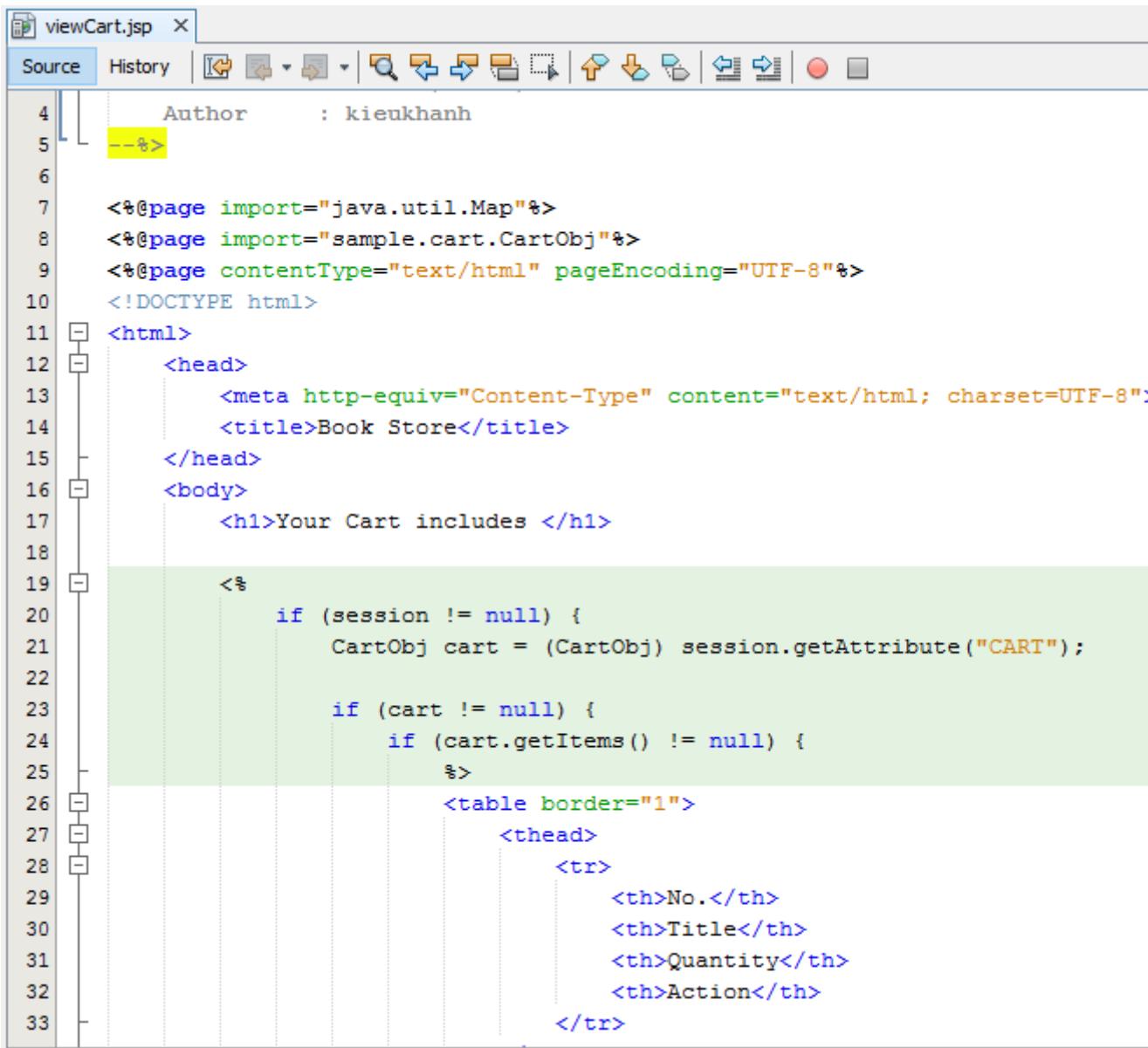


The screenshot shows a Java code editor with the file `AddItemServlet.java` open. The code implements a `HttpServlet` for adding items to a shopping cart. It uses session attributes and `PrintWriter` to respond to requests.

```
20  * @author kieukhanh
21  */
22 @WebServlet(name = "AddItemServlet", urlPatterns = {" /AddItemServlet"})
23  public class AddItemServlet extends HttpServlet {
24      private final String shoppingPage = "bookStore.html";
25
26      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
27       * @param request the servlet request
28       * @param response the servlet response
29       * @throws ServletException if a servlet-specific error occurs
30       * @throws IOException if an I/O error occurs
31      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32          throws ServletException, IOException {
33          response.setContentType("text/html;charset=UTF-8");
34          PrintWriter out = response.getWriter();
35          try {
36              HttpSession session = request.getSession();
37
38              CartObj cart = (CartObj) session.getAttribute("CART");
39              if (cart == null) {
40                  cart = new CartObj();
41              }
42
43              String title = request.getParameter("cboBook");
44
45              cart.addItemToCart(title);
46
47              session.setAttribute("CART", cart);
48
49              response.sendRedirect(shoppingPage);
50          } finally {
51              out.close();
52          }
53      }
54  }
```

# Appendix

## CRUD Function



```
viewCart.jsp X
Source History | 
4     Author      : kieukhanh
5     --%>
6
7     <%@page import="java.util.Map"%>
8     <%@page import="sample.cart.CartObj"%>
9     <%@page contentType="text/html" pageEncoding="UTF-8"%>
10    <!DOCTYPE html>
11    <html>
12        <head>
13            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14            <title>Book Store</title>
15        </head>
16        <body>
17            <h1>Your Cart includes </h1>
18
19            <%
20                if (session != null) {
21                    CartObj cart = (CartObj) session.getAttribute("CART");
22
23                    if (cart != null) {
24                        if (cart.getItems() != null) {
25                            %>
26                            <table border="1">
27                                <thead>
28                                    <tr>
29                                        <th>No.</th>
30                                        <th>Title</th>
31                                        <th>Quantity</th>
32                                        <th>Action</th>
33                                    </tr>
```

# Appendix

## CRUD Function

```
34      -> </thead>
35      -> <tbody>
36      -> <form action="SE1162Servlet">
37      ->     <%>
38      ->         Map<String, Integer> items = cart.getItems();
39      ->         int count = 0;
40      ->         for (Map.Entry item : items.entrySet()) {
41      ->             <%>
42      ->                 <tr>
43      ->                     <td>
44      ->                         <%= ++count %>
45      ->                     .</td>
46      ->                     <td>
47      ->                         <%= item.getKey() %>
48      ->                     </td>
49      ->                     <td>
50      ->                         <%= item.getValue() %>
51      ->                     </td>
52      ->                     <td>
53      ->                         <input type="checkbox" name="chkItem" value="<%= item.getKey() %>" />
54      ->                     </td>
55      ->                 </tr>
56      ->             <%>
57      ->                 }//end for
58      ->             <%>
59      ->             <tr>
60      ->                 <td colspan="3">
61      ->                     <a href="bookStore.html">Add More Items to Your Cart</a>
62      ->                 </td>
```

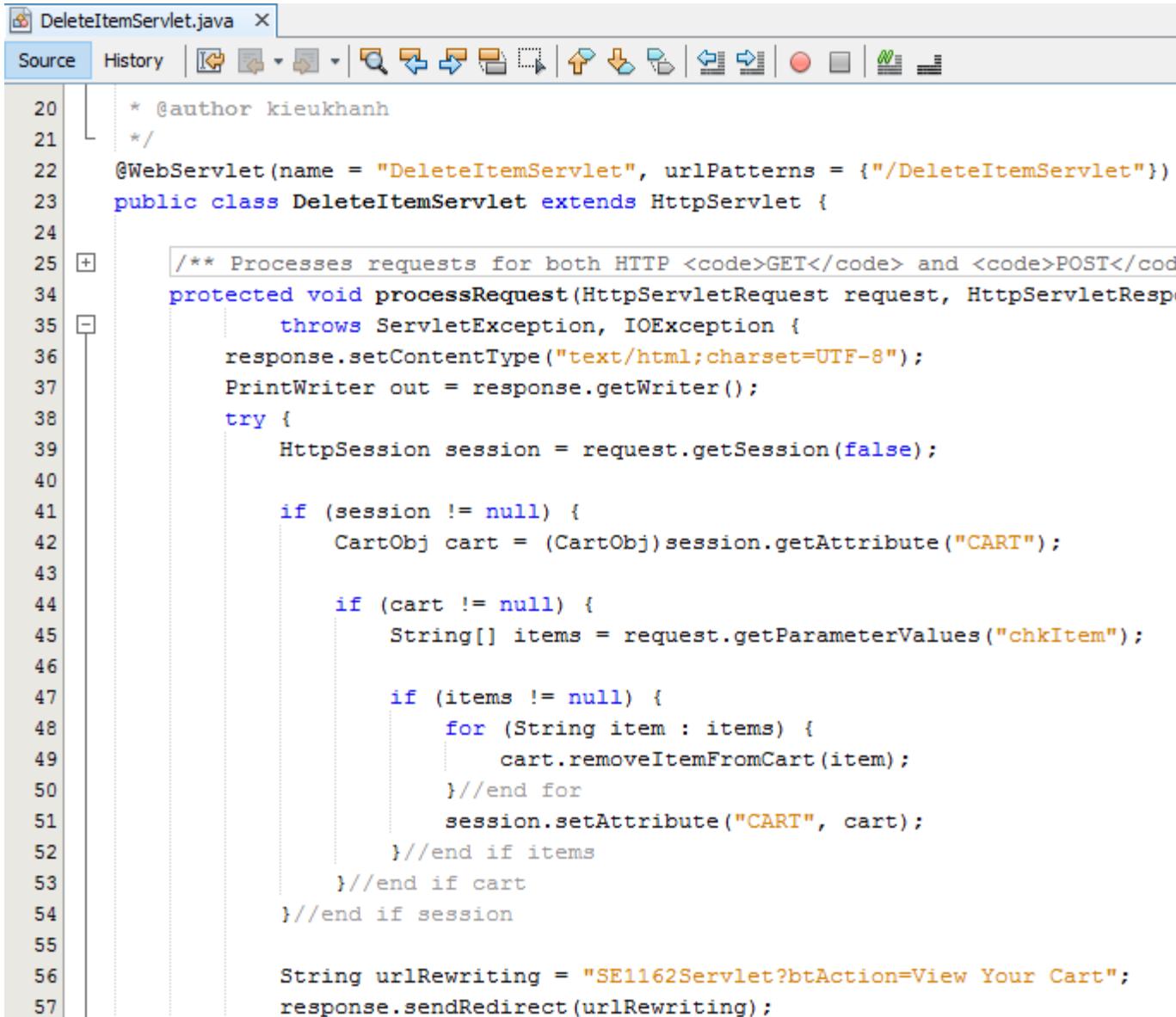
# Appendix

## CRUD Function

```
63      <td>
64          <input type="submit" value="Remove Selected Items" name="btAction" />
65      </td>
66      </tr>
67  </form>
68  </tbody>
69 </table>
70
71  <%
72      return;
73  %>//end items
74  %>//end cart
75  %>//end session
76  %>
77
78      <h2>No cart is existed</h2>
79  </body>
80 </html>
```

# Appendix

## CRUD Function

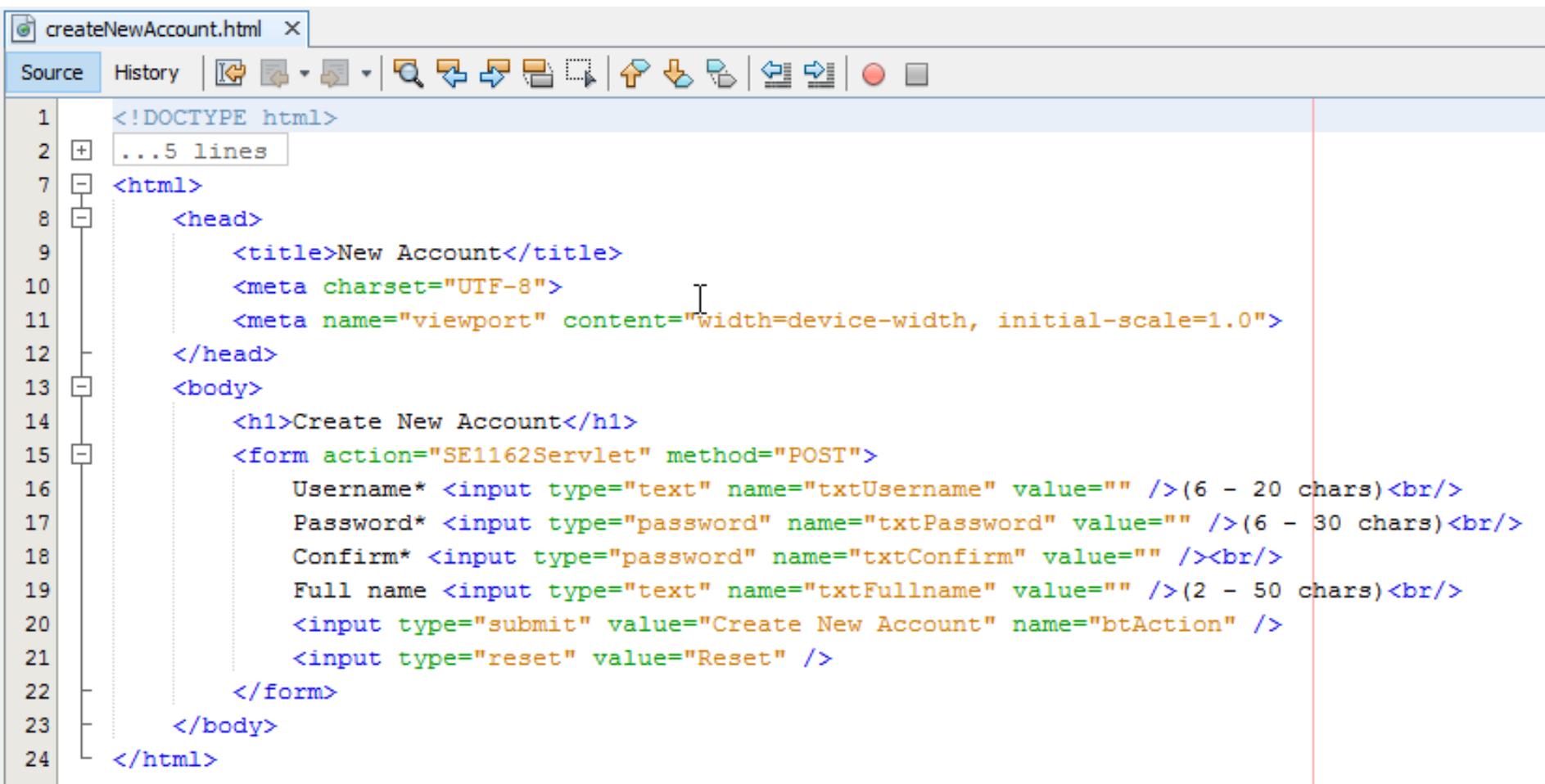


The screenshot shows a Java code editor window with the title bar "DeleteItemServlet.java". The menu bar includes "Source" and "History". Below the menu is a toolbar with various icons for file operations like open, save, and search. The code itself is a Java servlet named DeleteItemServlet. It processes both GET and POST requests. It retrieves a session, gets a cart attribute, and removes items from the cart based on checked items. Finally, it sends a redirect response.

```
20  * @author kieukhanh
21  */
22  @WebServlet(name = "DeleteItemServlet", urlPatterns = {" /DeleteItemServlet"})
23  public class DeleteItemServlet extends HttpServlet {
24
25      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
26      protected void processRequest(HttpServletRequest request, HttpServletResponse
27          throws ServletException, IOException {
28      response.setContentType("text/html;charset=UTF-8");
29      PrintWriter out = response.getWriter();
30      try {
31          HttpSession session = request.getSession(false);
32
33          if (session != null) {
34              CartObj cart = (CartObj) session.getAttribute("CART");
35
36              if (cart != null) {
37                  String[] items = request.getParameterValues("chkItem");
38
39                  if (items != null) {
40                      for (String item : items) {
41                          cart.removeItemFromCart(item);
42                      } //end for
43                      session.setAttribute("CART", cart);
44                  } //end if items
45              } //end if cart
46          } //end if session
47
48          String urlRewriting = "SE1162Servlet?btAction=View Your Cart";
49          response.sendRedirect(urlRewriting);
50      } //end try
51      catch (Exception e) {
52          e.printStackTrace();
53      }
54  } //end processRequest
55
56  /**
57  * @param request the request from the client
58  * @param response the response to the client
59  * @throws ServletException if an error occurs
60  * @throws IOException if an error occurs
61  */
62  protected void doGet(HttpServletRequest request, HttpServletResponse response)
63      throws ServletException, IOException {
64      processRequest(request, response);
65  }
66
67  /**
68  * @param request the request from the client
69  * @param response the response to the client
70  * @throws ServletException if an error occurs
71  * @throws IOException if an error occurs
72  */
73  protected void doPost(HttpServletRequest request, HttpServletResponse response)
74      throws ServletException, IOException {
75      processRequest(request, response);
76  }
77
78 }
```

# Appendix

## CRUD Function

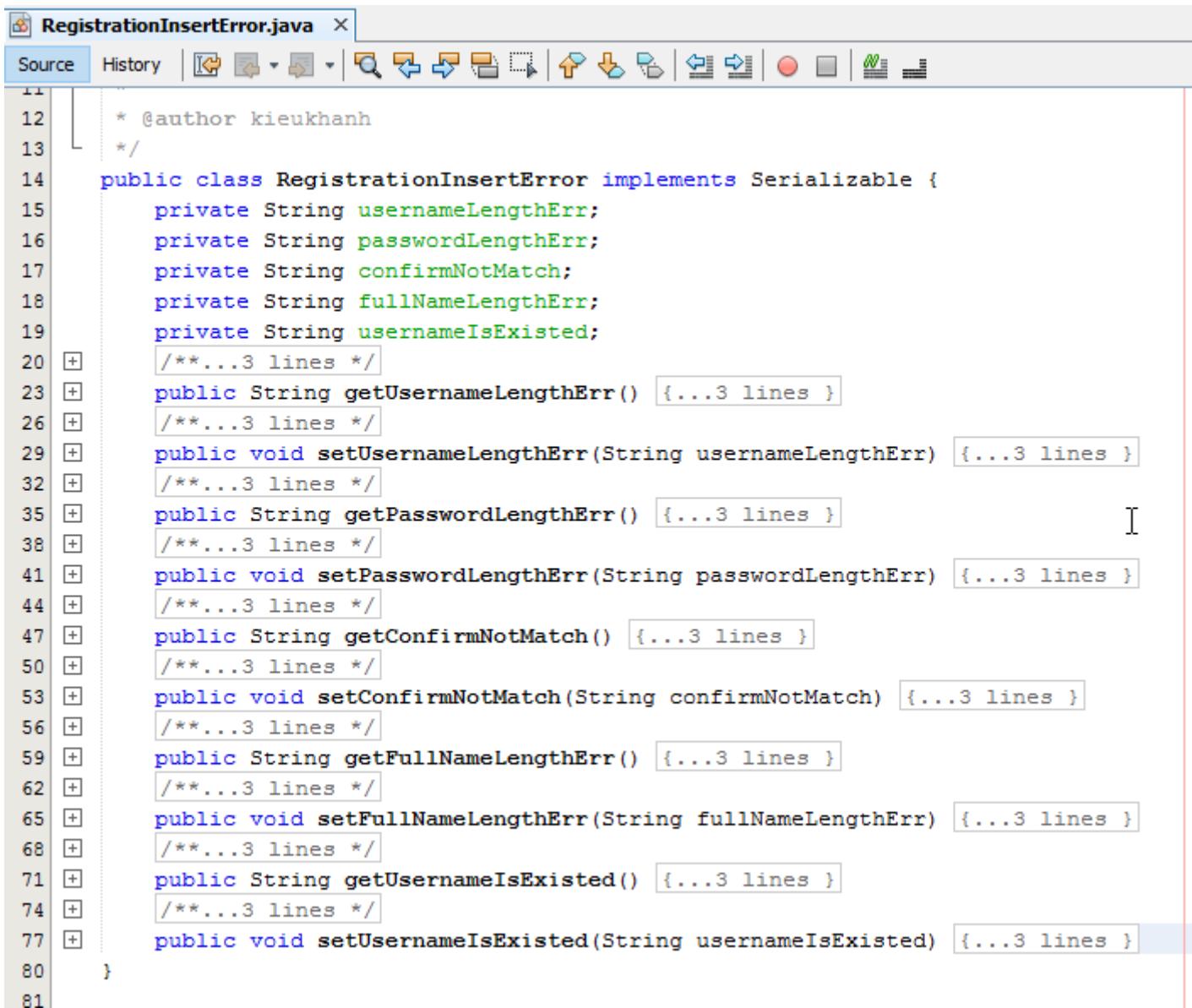


The screenshot shows a code editor window with the file 'createNewAccount.html' open. The code is an HTML form for creating a new account, structured with a head and body section. The head contains meta tags for the title, charset, and viewport. The body contains an h1 header, a form with various input fields (username, password, confirm, full name), and submit/reset buttons.

```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>New Account</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="SE1162Servlet" 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

## CRUD Function

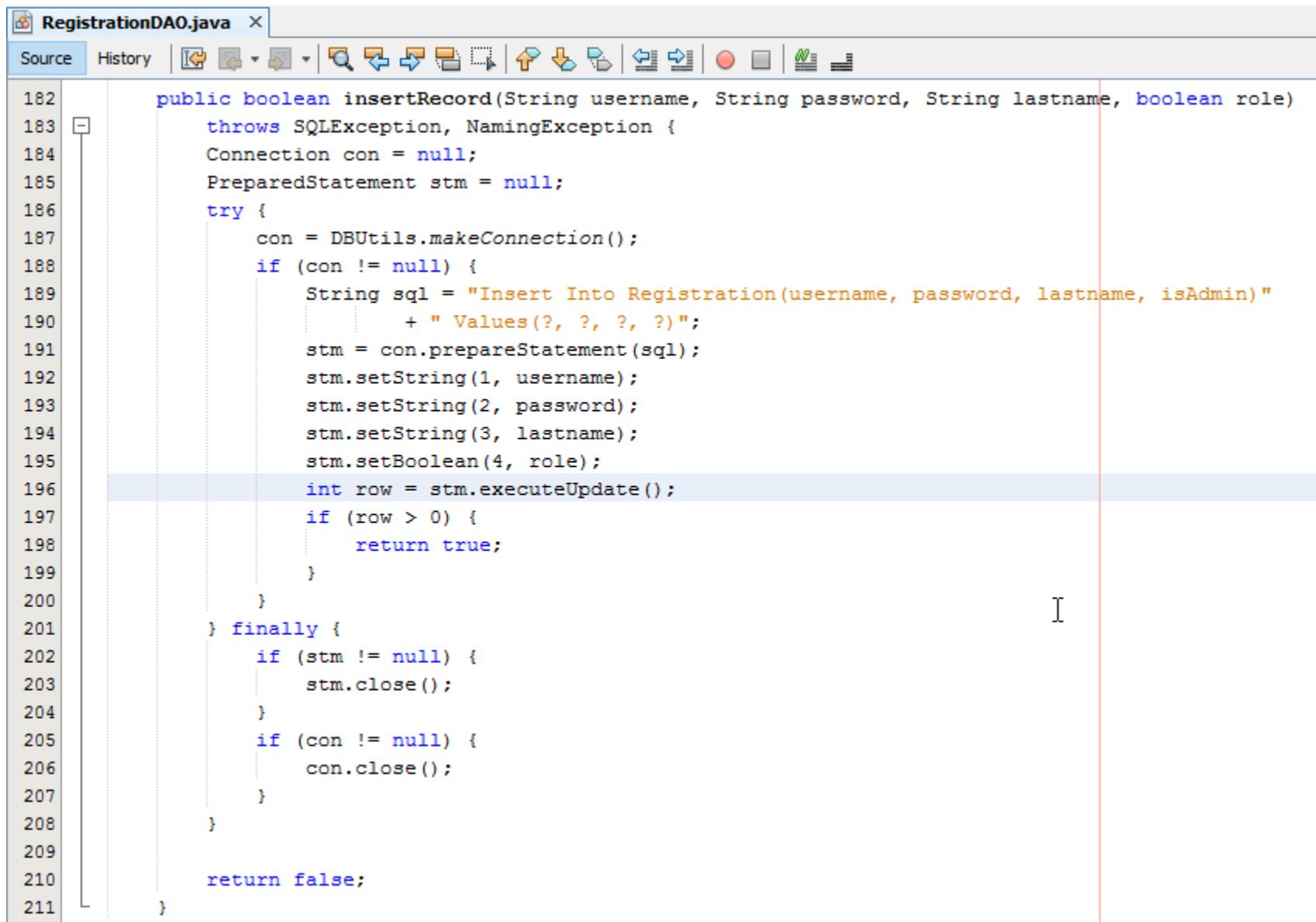


The screenshot shows a Java code editor window titled "RegistrationInsertError.java". The code implements the Serializable interface and contains methods for setting and getting various error messages related to registration. The code is heavily annotated with Javadoc-style comments. The editor interface includes tabs for "Source" and "History", and a toolbar with various icons for file operations like cut, copy, paste, and search.

```
11 * @author kieukhanh
12 */
13
14 public class RegistrationInsertError implements Serializable {
15     private String usernameLengthErr;
16     private String passwordLengthErr;
17     private String confirmNotMatch;
18     private String fullNameLengthErr;
19     private String usernameIsExisted;
20     /**
21      * @author kieukhanh
22      */
23     public String getUsernameLengthErr() { ...3 lines }
24     /**
25      * @author kieukhanh
26      */
27     public void setUsernameLengthErr(String usernameLengthErr) { ...3 lines }
28     /**
29      * @author kieukhanh
30      */
31     public String getPasswordLengthErr() { ...3 lines }
32     /**
33      * @author kieukhanh
34      */
35     public void setPasswordLengthErr(String passwordLengthErr) { ...3 lines }
36     /**
37      * @author kieukhanh
38      */
39     public String getConfirmNotMatch() { ...3 lines }
40     /**
41      * @author kieukhanh
42      */
43     public void setConfirmNotMatch(String confirmNotMatch) { ...3 lines }
44     /**
45      * @author kieukhanh
46      */
47     public String getFullNameLengthErr() { ...3 lines }
48     /**
49      * @author kieukhanh
50      */
51     public void setFullNameLengthErr(String fullNameLengthErr) { ...3 lines }
52     /**
53      * @author kieukhanh
54      */
55     public String getUsernameIsExisted() { ...3 lines }
56     /**
57      * @author kieukhanh
58      */
59     public void setUsernameIsExisted(String usernameIsExisted) { ...3 lines }
```

# Appendix

## CRUD Function

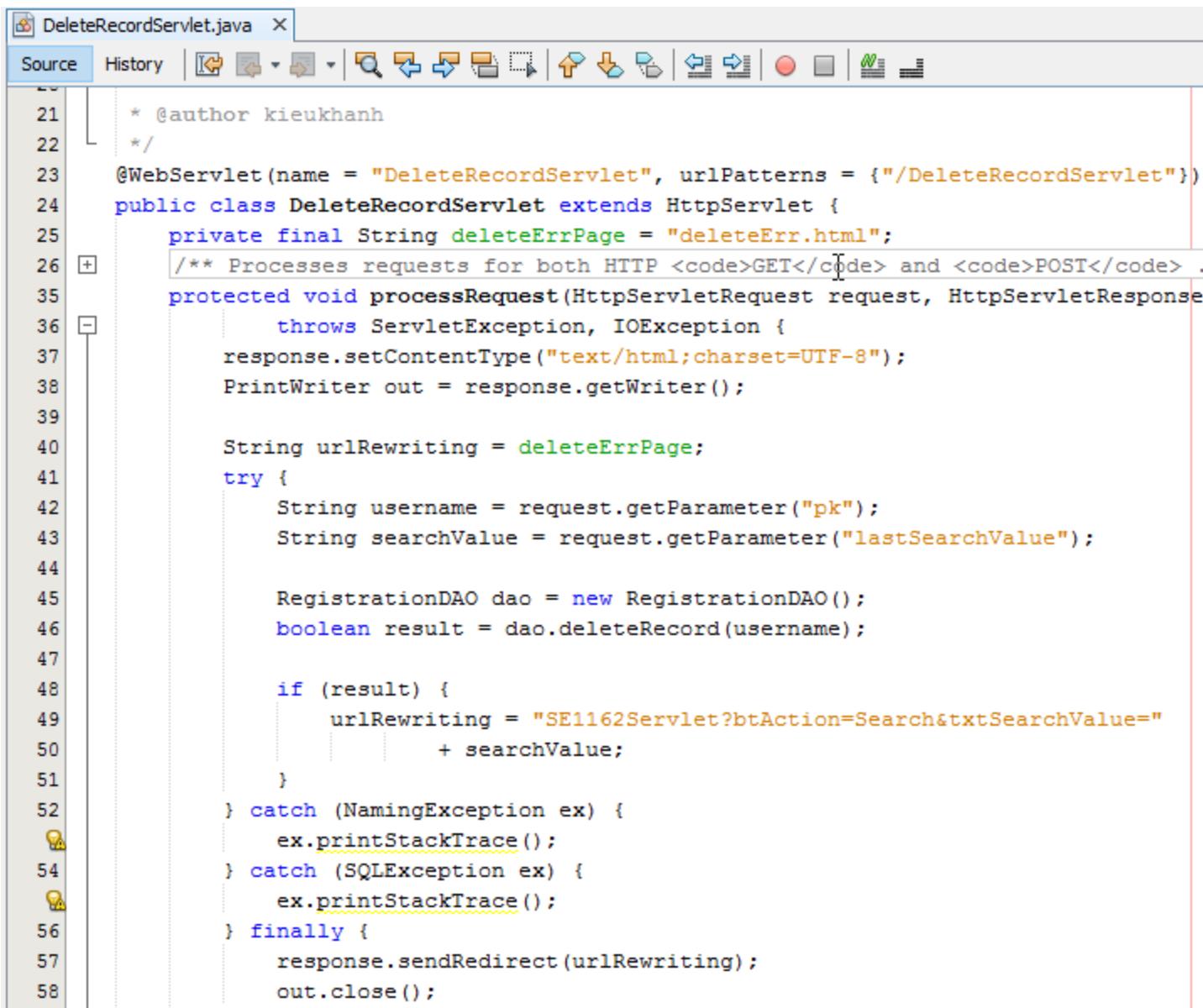


The screenshot shows a Java code editor window titled "RegistrationDAO.java". The code is a Java class containing a single method, `insertRecord`, which performs an insert operation into a database table named "Registration". The code uses JDBC to connect to the database and execute an SQL statement. The code editor has a toolbar at the top with various icons for file operations like new, open, save, and search. The code itself is color-coded for syntax, with numbers on the left indicating line numbers.

```
182     public boolean insertRecord(String username, String password, String lastname, boolean role)
183         throws SQLException, NamingException {
184     Connection con = null;
185     PreparedStatement stm = null;
186     try {
187         con = DBUtils.makeConnection();
188         if (con != null) {
189             String sql = "Insert Into Registration(username, password, lastname, isAdmin)"
190                     + " Values(?, ?, ?, ?)";
191             stm = con.prepareStatement(sql);
192             stm.setString(1, username);
193             stm.setString(2, password);
194             stm.setString(3, lastname);
195             stm.setBoolean(4, role);
196             int row = stm.executeUpdate();
197             if (row > 0) {
198                 return true;
199             }
200         }
201     } finally {
202         if (stm != null) {
203             stm.close();
204         }
205         if (con != null) {
206             con.close();
207         }
208     }
209     return false;
210 }
```

# Appendix

## CRUD Function

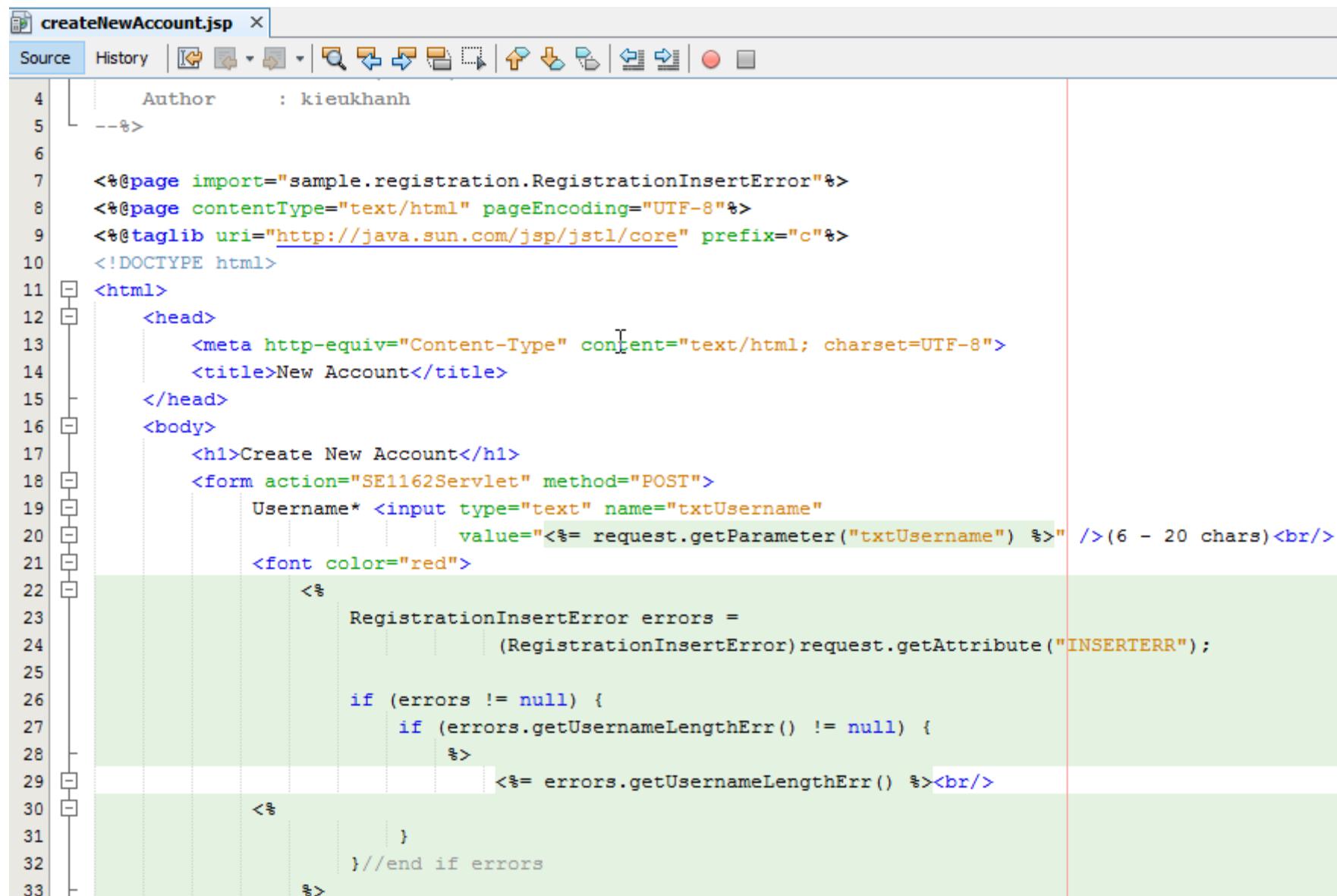


The screenshot shows a Java code editor window titled "DeleteRecordServlet.java". The code is a servlet for deleting records from a database. It uses annotations for web services and handles both GET and POST requests. It retrieves parameters from the request, interacts with a RegistrationDAO object to delete a record, and then performs URL rewriting to redirect the user. Error handling is included for NamingException and SQLException.

```
21 * @author kieukhanh
22 */
23 @WebServlet(name = "DeleteRecordServlet", urlPatterns = {" /DeleteRecordServlet"})
24 public class DeleteRecordServlet extends HttpServlet {
25     private final String deleteErrPage = "deleteErr.html";
26     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> .
27     protected void processRequest(HttpServletRequest request, HttpServletResponse
28             throws ServletException, IOException {
29         response.setContentType("text/html;charset=UTF-8");
30         PrintWriter out = response.getWriter();
31
32         String urlRewriting = deleteErrPage;
33         try {
34             String username = request.getParameter("pk");
35             String searchValue = request.getParameter("lastSearchValue");
36
37             RegistrationDAO dao = new RegistrationDAO();
38             boolean result = dao.deleteRecord(username);
39
40             if (result) {
41                 urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
42                             + searchValue;
43             }
44         } catch (NamingException ex) {
45             ex.printStackTrace();
46         } catch (SQLException ex) {
47             ex.printStackTrace();
48         } finally {
49             response.sendRedirect(urlRewriting);
50             out.close();
51         }
52     }
53 }
54
55
56
57
58 }
```

# Appendix

## CRUD Function



The screenshot shows a Java Server Page (JSP) editor window with the file name 'createNewAccount.jsp'. The code is written in JSP syntax, using Java and JSTL tags.

```
4     Author      : kieukhanh
5 ---%>
6
7 <%@page import="sample.registration.RegistrationInsertError"%>
8 <%@page contentType="text/html" pageEncoding="UTF-8"%>
9 <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
10 <!DOCTYPE html>
11 <html>
12   <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>New Account</title>
15   </head>
16   <body>
17     <h1>Create New Account</h1>
18     <form action="SE1162Servlet" method="POST">
19       Username* <input type="text" name="txtUsername"
20                  value="<% request.getParameter("txtUsername") %>" />(6 - 20 chars)<br/>
21       <font color="red">
22         <%
23           RegistrationInsertError errors =
24             (RegistrationInsertError)request.getAttribute("INSERTERR");
25
26           if (errors != null) {
27             if (errors.getUsernameLengthErr() != null) {
28               <%>
29               <%= errors.getUsernameLengthErr() %><br/>
30             <%>
31             }
32           //end if errors
33         <%>
```

# Appendix

## CRUD Function

```
34          </font>
35          Password* <input type="password" name="txtPassword" value="" />(6 - 30 chars)<br/>
36          <font color="red">
37              <%
38                  if (errors != null) {
39                      if (errors.getPasswordLengthErr() != null) {
40                          %>
41                          <%= errors.getPasswordLengthErr()%><br/>
42              <%
43                  }
44                  //end if errors
45              %>
46          </font>
47          Confirm* <input type="password" name="txtConfirm" value="" /><br/>
48          <font color="red">
49              <%
50                  if (errors != null) {
51                      if (errors.getConfirmNotMatch() != null) {
52                          %>
53                          <%= errors.getConfirmNotMatch()%><br/>
54              <%
55                  }
56                  //end if errors
57              %>
58          </font>
59          Full name <input type="text" name="txtFullname"
60                      value=<%= request.getParameter("txtFullname") %>" />(2 - 50 chars)<br/>
61          <font color="red">
```

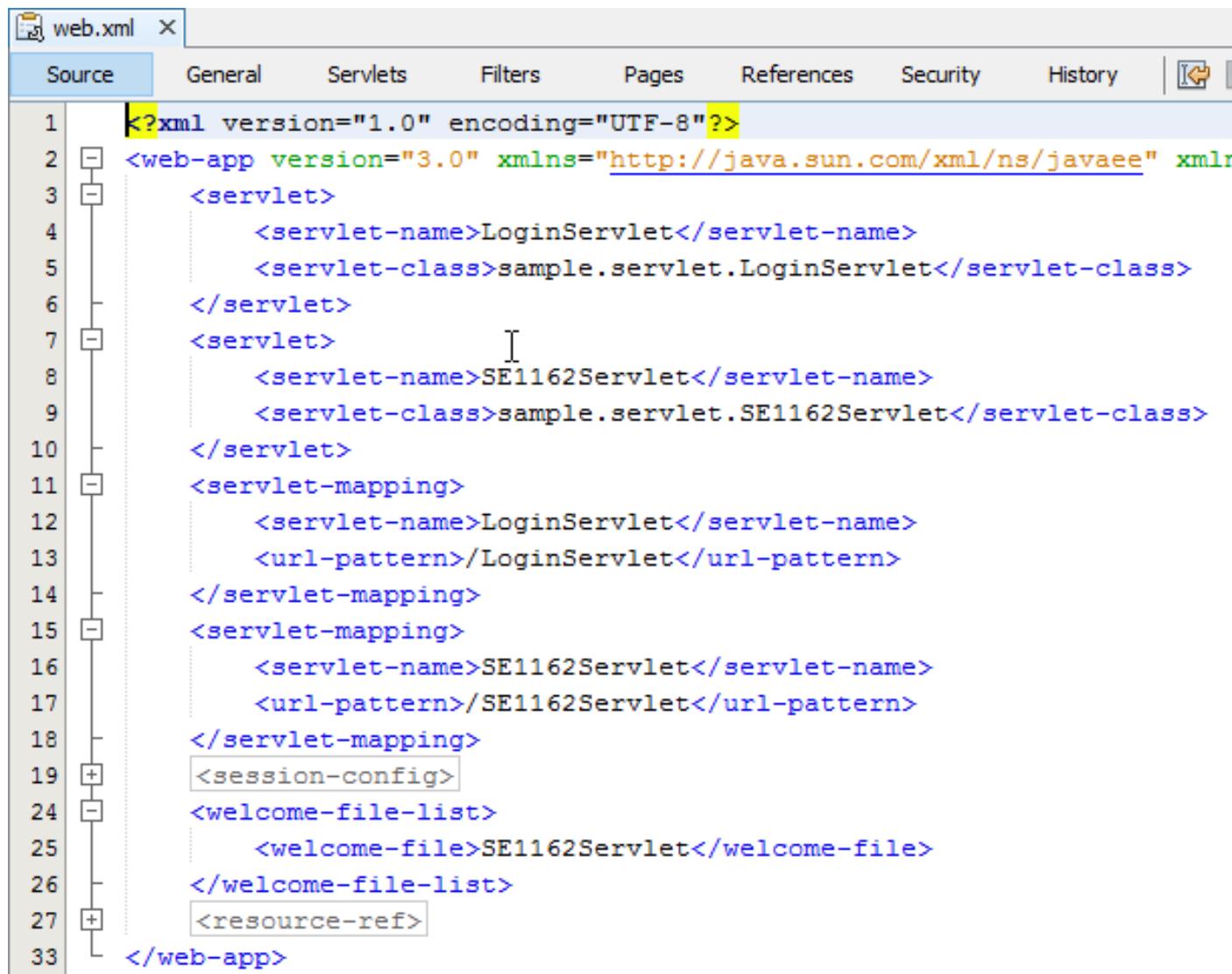
# Appendix

## CRUD Function

```
62      <%
63          if (errors != null) {
64              if (errors.getFullNameLengthErr() != null) {
65                  %>
66                  <%= errors.getFullNameLengthErr()%><br/>
67              <%
68                  }
69              //end if errors
70          %>
71      </font>
72      <input type="submit" value="Create New Account" name="btAction" />
73      <input type="reset" value="Reset" />
74  </form><br/>
75  <font color="red">
76      <%
77          if (errors != null) {
78              if (errors.getUsernameIsExisted() != null) {
79                  %>
80                  <%= errors.getUsernameIsExisted()%><br/>
81              <%
82                  }
83              //end if errors
84          %>
85      </font>
86  </body>
87 </html>
```

# Appendix

## CRUD Function



The screenshot shows the `web.xml` configuration file for a Java Web Application. The file defines two servlets: `LoginServlet` and `SE1162Servlet`, and maps them to specific URLs. It also includes session configuration and a welcome file list.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>sample.servlet.LoginServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>SE1162Servlet</servlet-name>
        <servlet-class>sample.servlet.SE1162Servlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/LoginServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SE1162Servlet</servlet-name>
        <url-pattern>/SE1162Servlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <welcome-file-list>
            <welcome-file>SE1162Servlet</welcome-file>
        </welcome-file-list>
        <resource-ref>
    </web-app>
```