

Session Management

Sessions and Listeners

Technique: Error Handling in Servlets

#Session #UrlRewriting #HiddenForm

#SessionTracking #Cookie

#ShoppingCart #ErrorHandle

#JSP

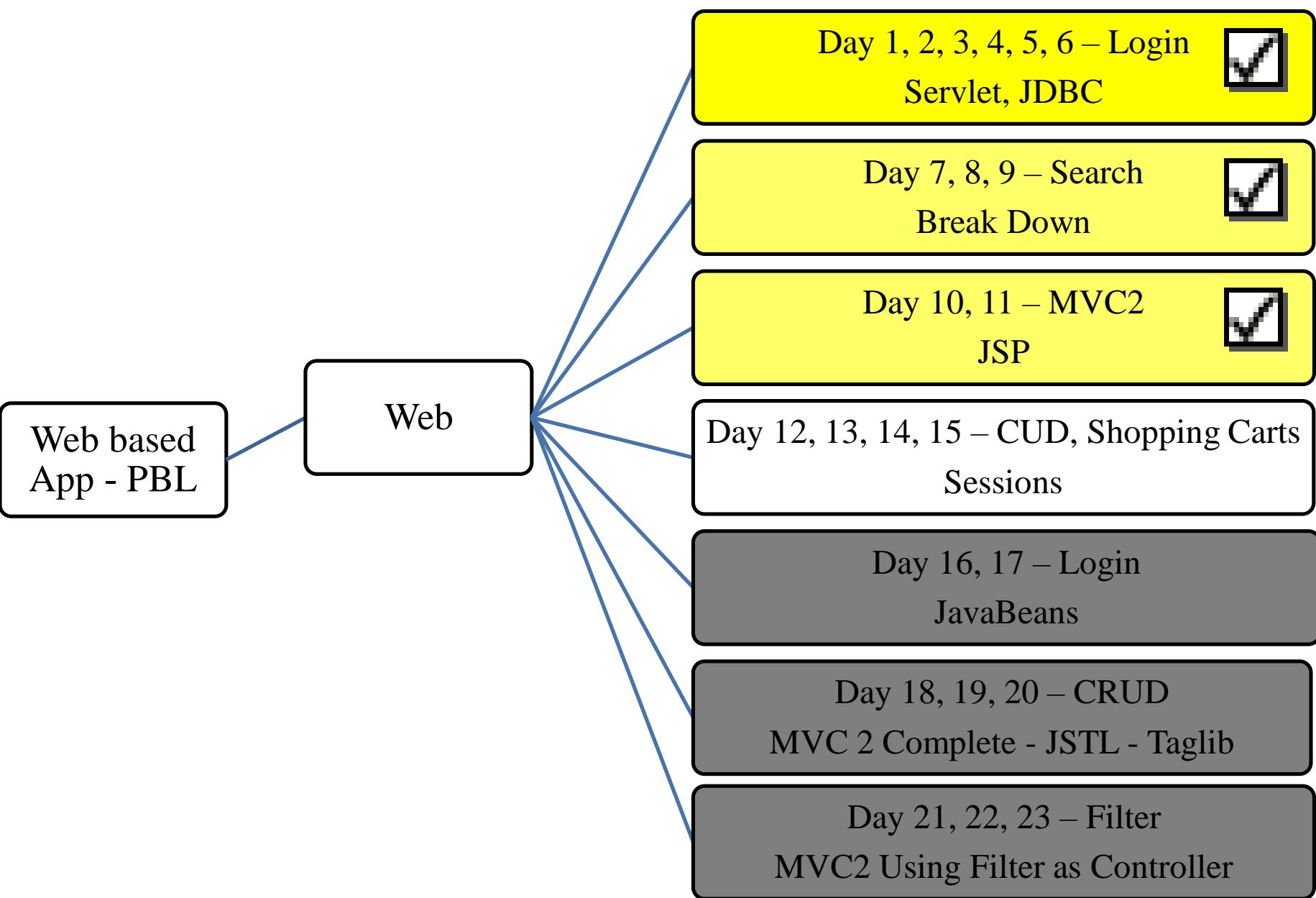
Review

- **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"/>	Delete
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	Delete
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	Delete
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	Delete
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	Delete

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

How to write CRUD Web Application

Expectation



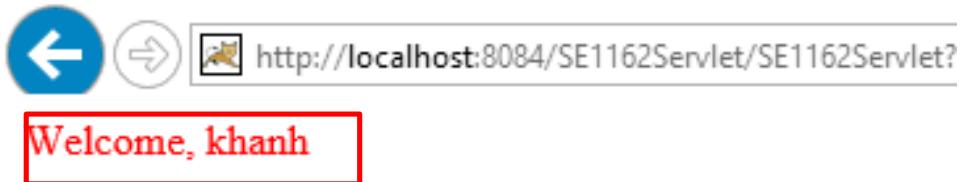
Search Page

Search Value

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

How to write CRUD Web Application

Expectation

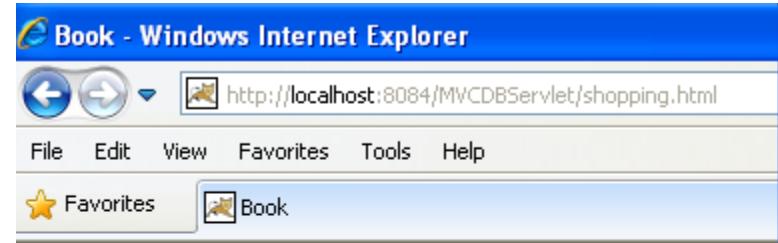
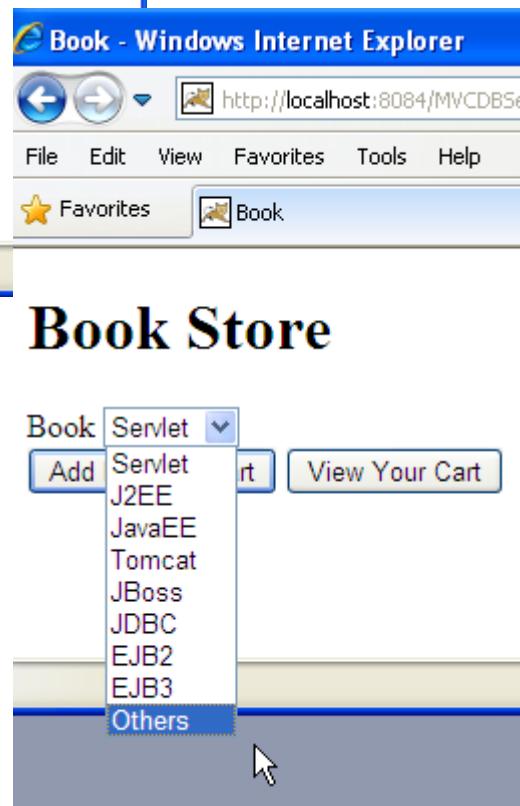
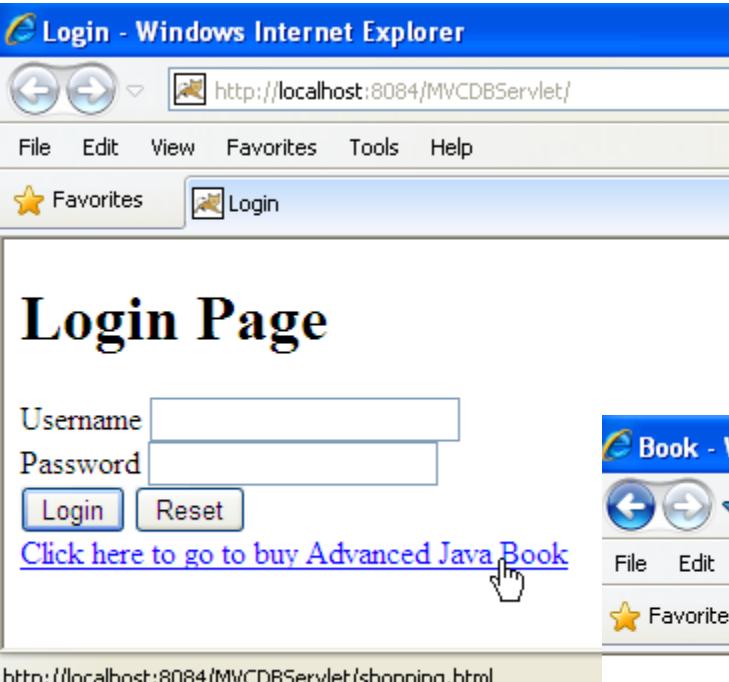


Search Page

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 browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. A toolbar below the menu bar has icons for Back, Forward, Stop, and Refresh, along with a Favorites button and a Carts button. The main content area displays a table titled "Your Cart Items". The table has columns: No., Title, Quantity, and Action. It contains three rows: 1. JavaEE, 1, with an empty checkbox in the Action column. 2. Servlet, 2, with an empty checkbox in the Action column. 3. Tomcat, 2, with an empty checkbox in the Action column. 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 state of the shopping cart after some items have been selected. The URL and browser interface are identical to the first screenshot. The "Your Cart Items" table now shows checked checkboxes in the "Action" column for the first and third items. The second item still has an empty checkbox. The "Add More Item to Cart" and "Remove" buttons are at the bottom.

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 state of the shopping cart after one item has been removed. The URL and browser interface are identical to the previous screenshots. The "Your Cart Items" table now shows only one item: "Servlet" with quantity 2 and an empty checkbox in the "Action" column. The "Add More Item to Cart" and "Remove" buttons are at the bottom.

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

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

How to write CRUD Web Application

Expectation

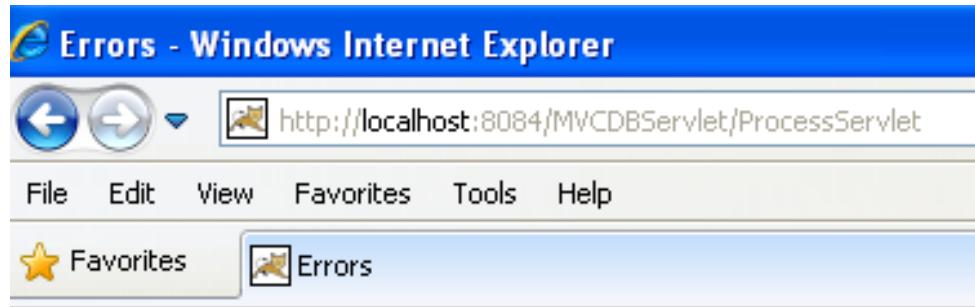


A screenshot of a web browser window. The address bar shows the URL <http://localhost:8084/MVCDBServlet/register.html>. The page title is **Register Page**. The form contains four input fields with validation rules: **Username*** (6 - 12 chars), **Password*** (8 - 20 chars), **Confirm***, and **Full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

Username*	(6 - 12 chars)
Password*	(8 - 20 chars)
Confirm*	
Full name*	(2 - 40 chars)

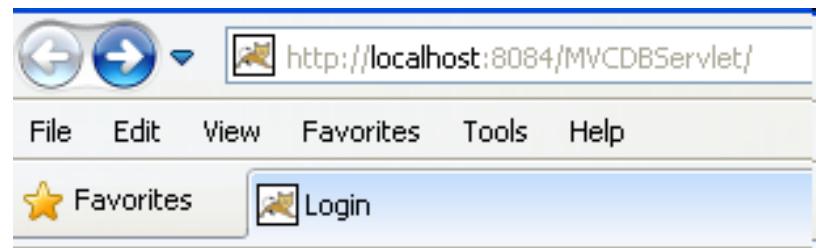
How to write CRUD Web Application

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

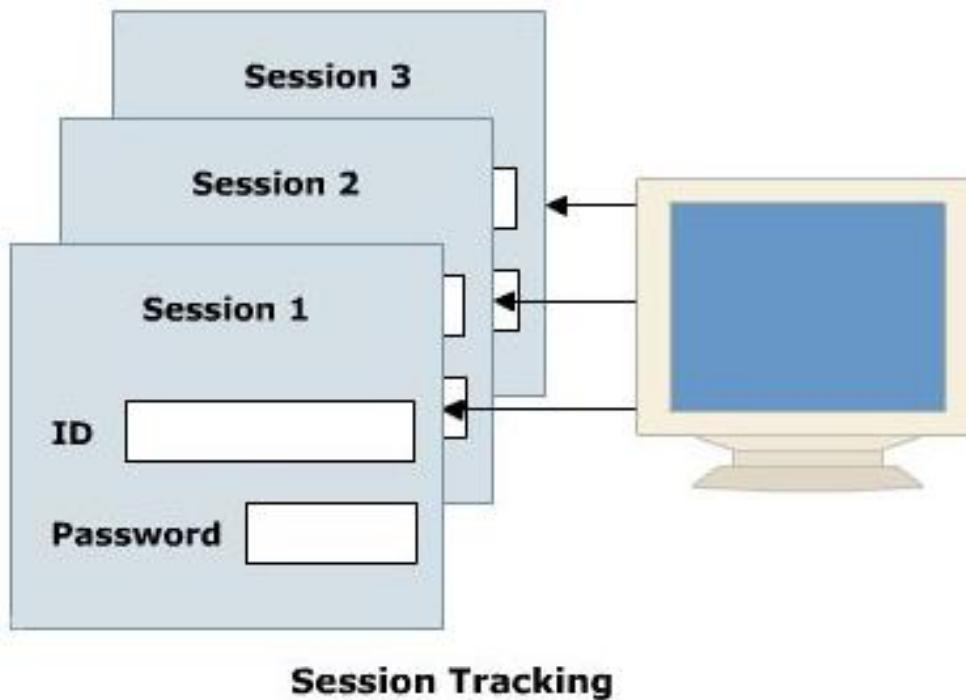
Session

- 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**
 - <a href="<http://localhost:8080/Books?category=java>"> Java Books
 - <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

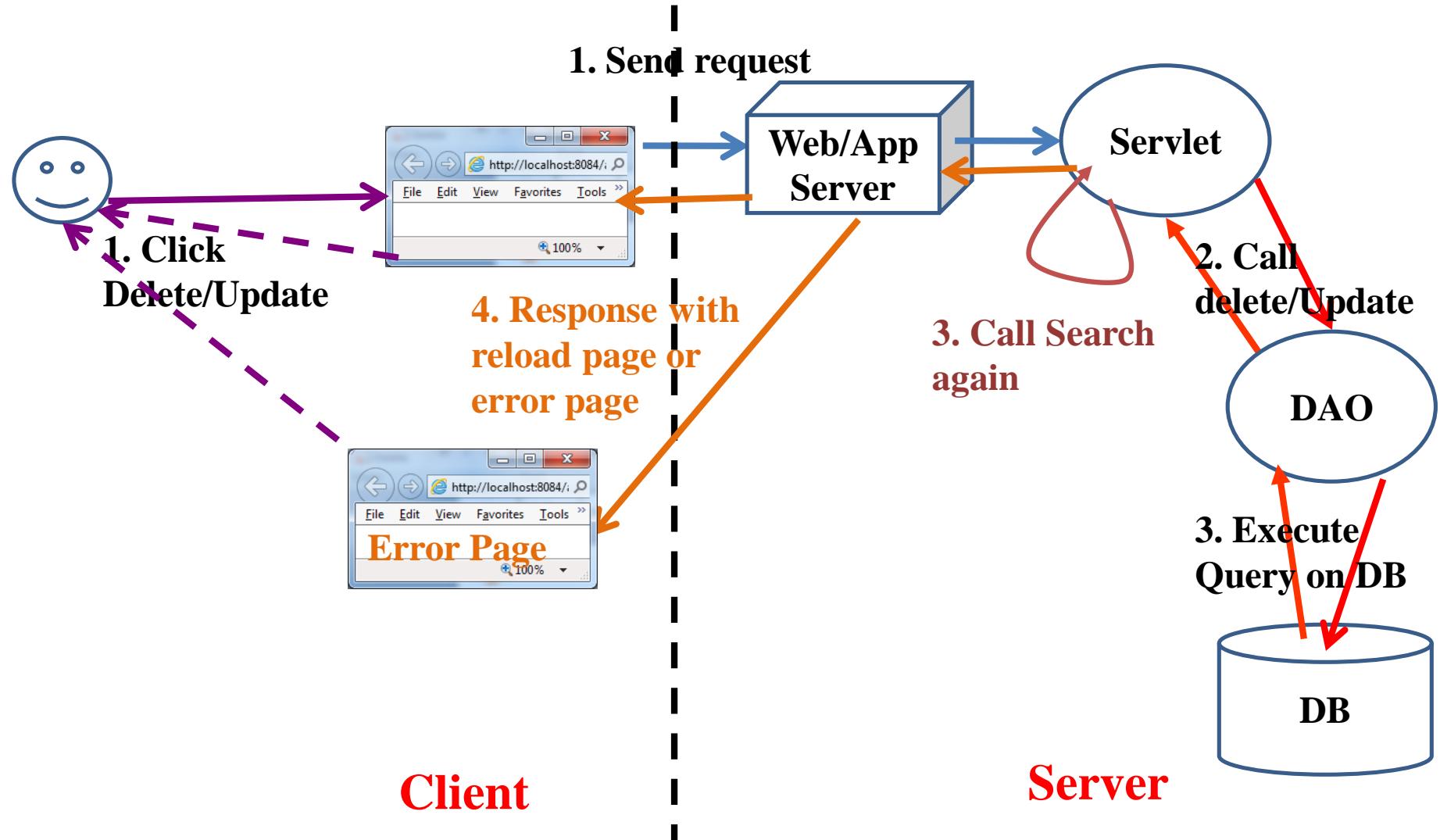
Search Page

Search Value

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

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"/>	Delete
2	khanh	kieu123	Khanh Kieu	<input checked="" type="checkbox"/>	Delete
3	SE1161	123456	Class SE1161	<input type="checkbox"/>	Delete
4	SE1162	123456	Class Se1162	<input type="checkbox"/>	Delete
5	SE1163	123456	Class SE1163	<input type="checkbox"/>	Delete

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

← → 🚗 http://localhost:8084/SE1162Servlet/SE1162Servlet?txtSearchValue=a&btAction=Search

Welcome, khanh

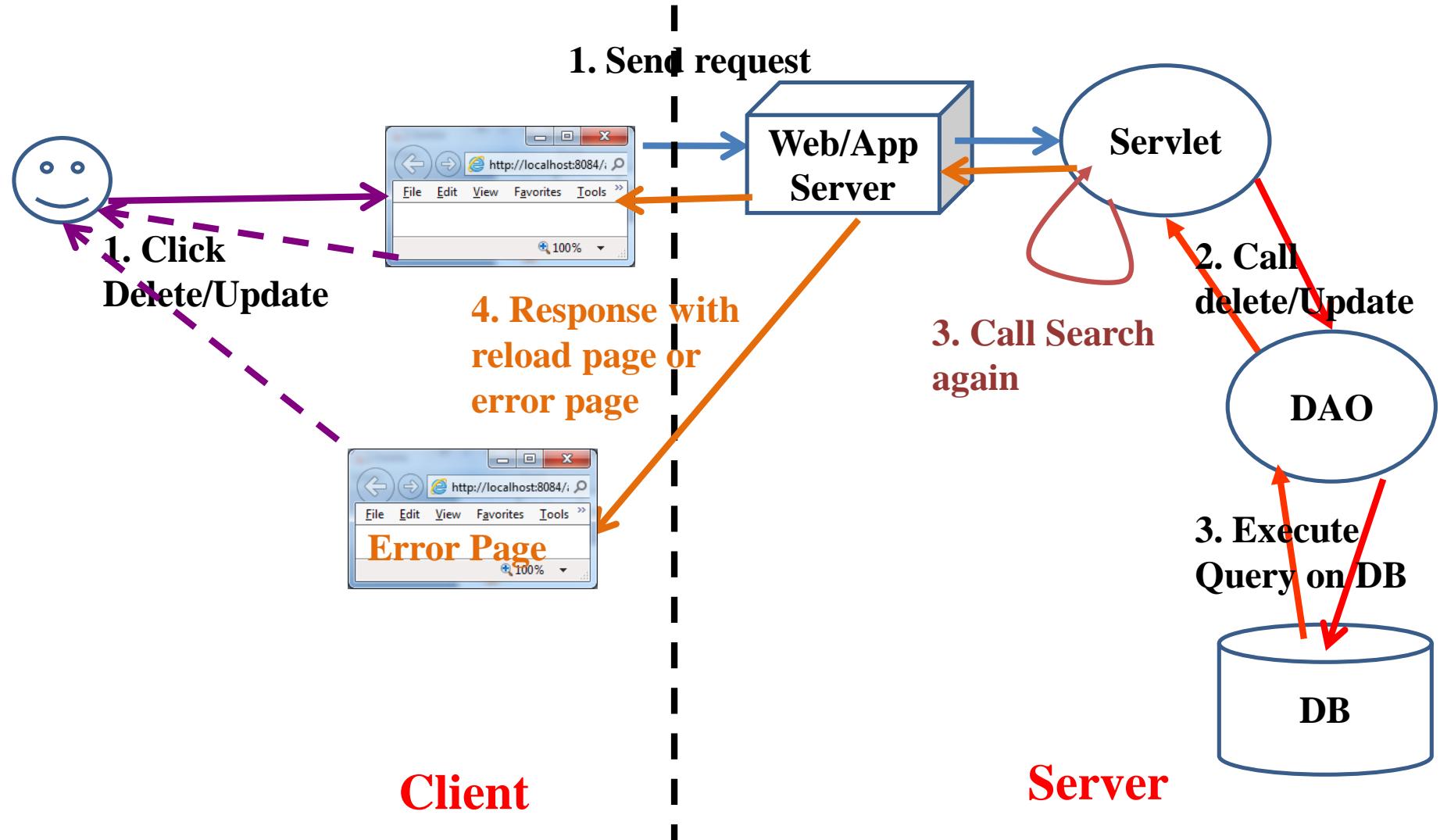
Search Page

Search Value

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

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
addCookie	<ul style="list-style-type: none"> - public void addCookie(cookie1); - Adds field to the HTTP response headers to send cookies to the browser, one at a time - Adds specified cookie to the response - Can be called multiple times to set more than one cookies
setValue	<ul style="list-style-type: none"> - public void setValue(String newValue); - 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
setPath	<ul style="list-style-type: none"> - public void setPath(String path); - 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
setMaxAge	<ul style="list-style-type: none"> - public void setMaxAge(int expiry); - 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

Sessions & Listeners

Cookies

- **Reading Cookie**

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

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

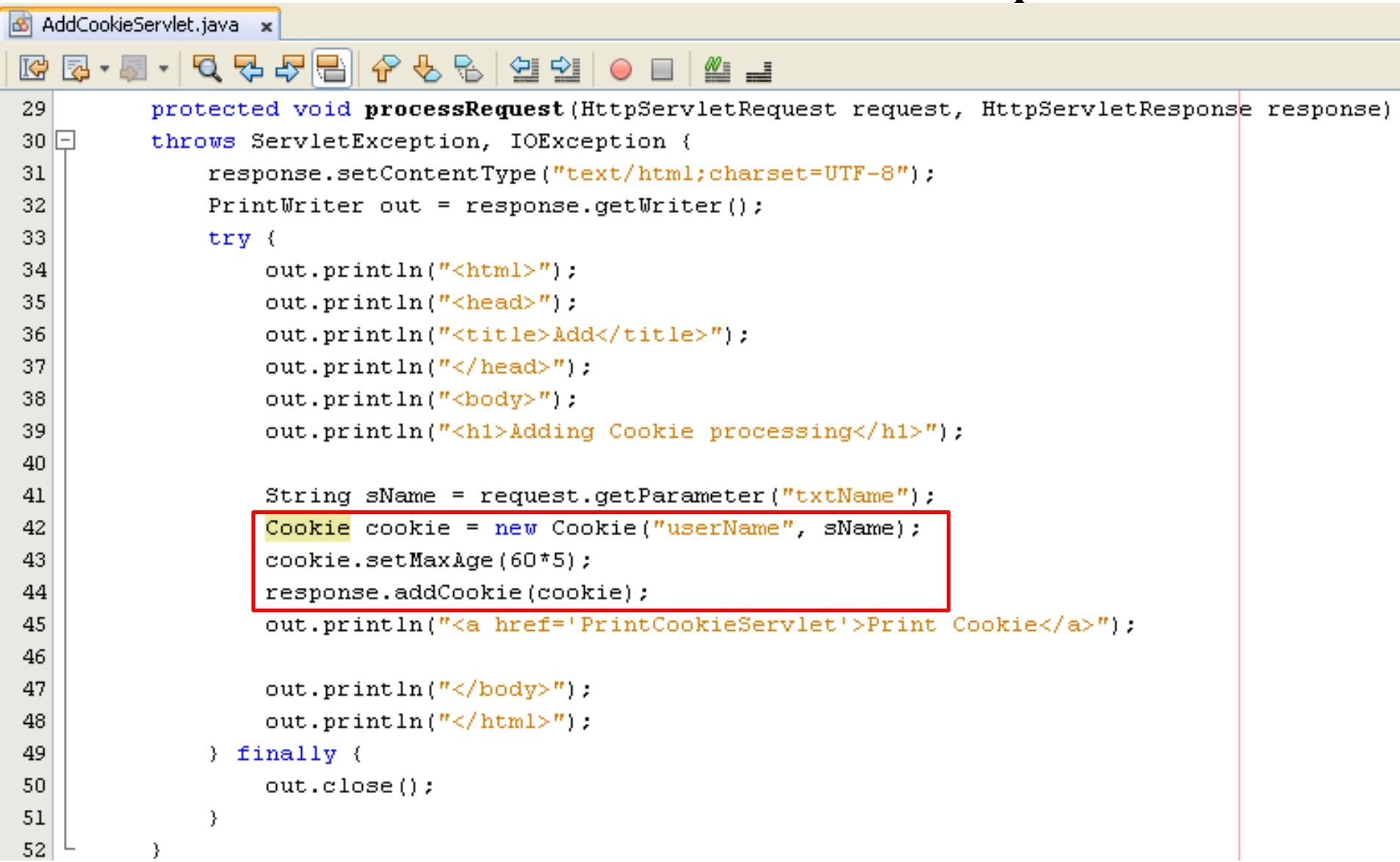
```

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

```

Sessions & Listeners

Cookies – Example

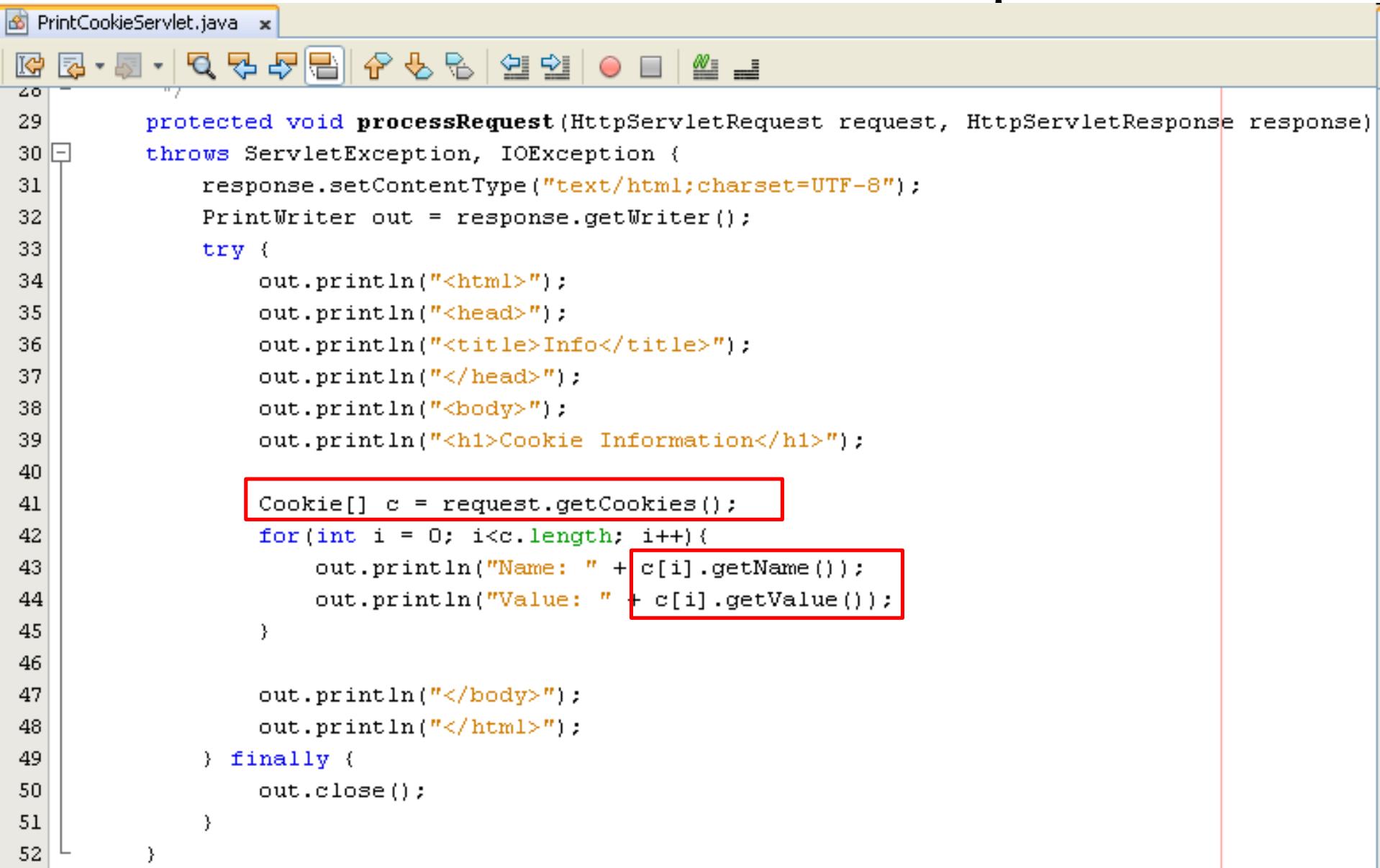


The screenshot shows a Java code editor with the file `AddCookieServlet.java` open. The code implements a `processRequest` method to handle an HTTP request. It prints an HTML page with a title and a heading. It then retrieves a parameter from the request, creates a cookie with the value of that parameter, sets its maximum age to 5 minutes, and adds it to the response. Finally, it prints a link to another servlet. The code is annotated with line numbers from 29 to 52.

```
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>Add</title>");
37             out.println("</head>");
38             out.println("<body>");
39             out.println("<h1>Adding Cookie processing</h1>");
40
41             String sName = request.getParameter("txtName");
42             Cookie cookie = new Cookie("userName", sName);
43             cookie.setMaxAge(60*5);
44             response.addCookie(cookie);
45             out.println("<a href='PrintCookieServlet'>Print Cookie</a>");
46
47             out.println("</body>");
48             out.println("</html>");
49         } finally {
50             out.close();
51         }
52     }
```

Sessions & Listeners

Cookies – Example



The screenshot shows a Java code editor with the file `PrintCookieServlet.java` open. The code is a servlet that prints cookie information. A red box highlights the line `Cookie[] c = request.getCookies();`. Another red box highlights the two lines that print the name and value of each cookie: `out.println("Name: " + c[i].getName());` and `out.println("Value: " + c[i].getValue());`.

```
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 }
```



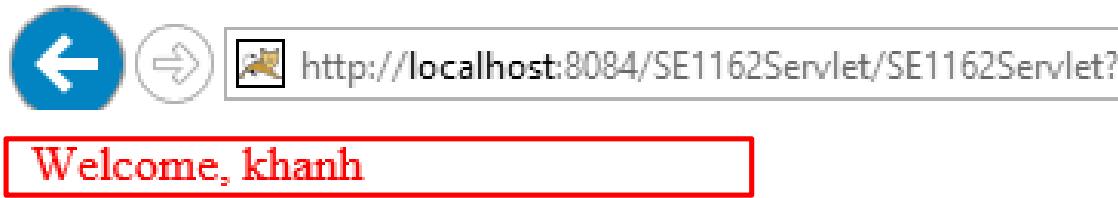
How to write CRUD Web Application

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

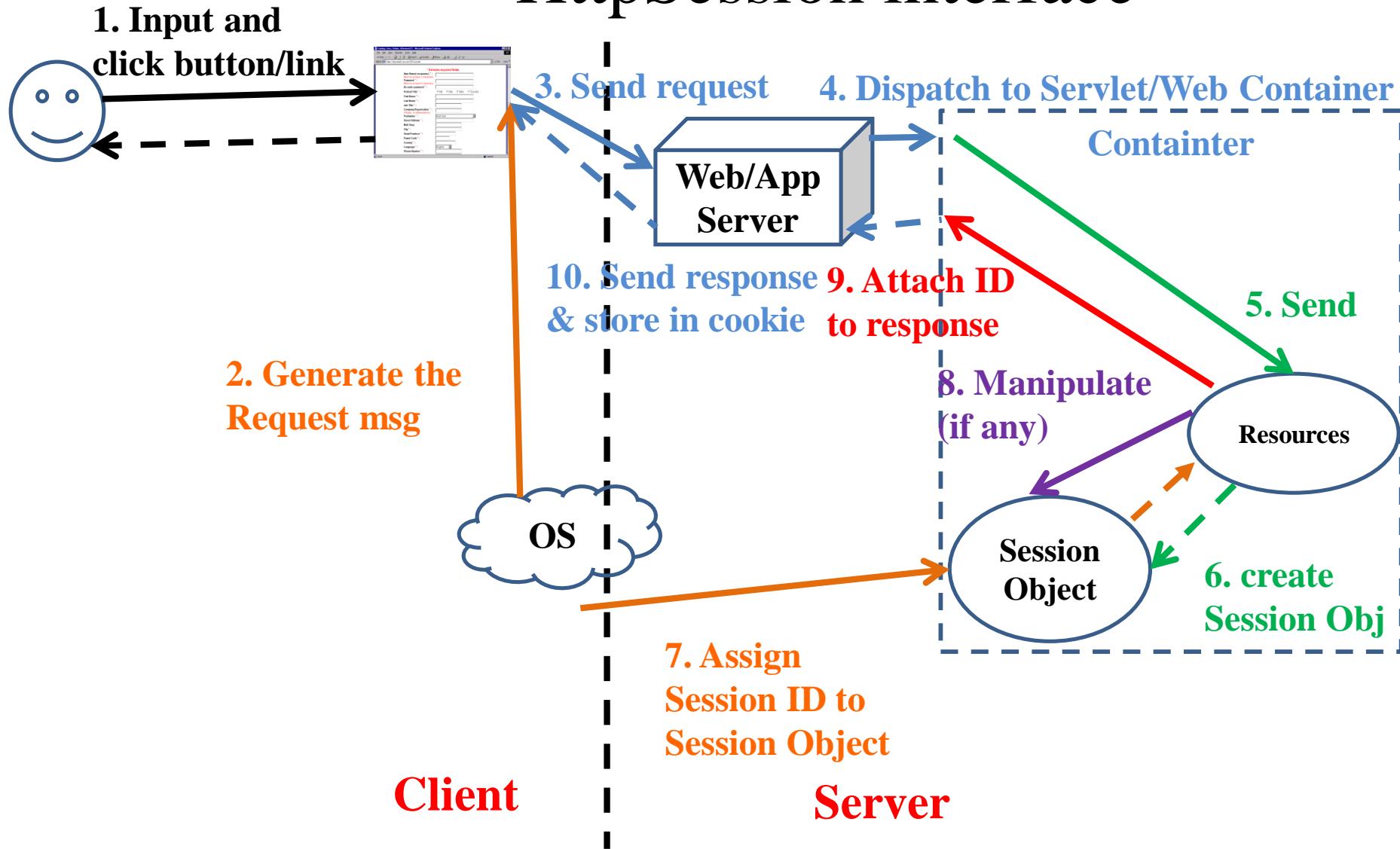
Draw your self

Client

Server

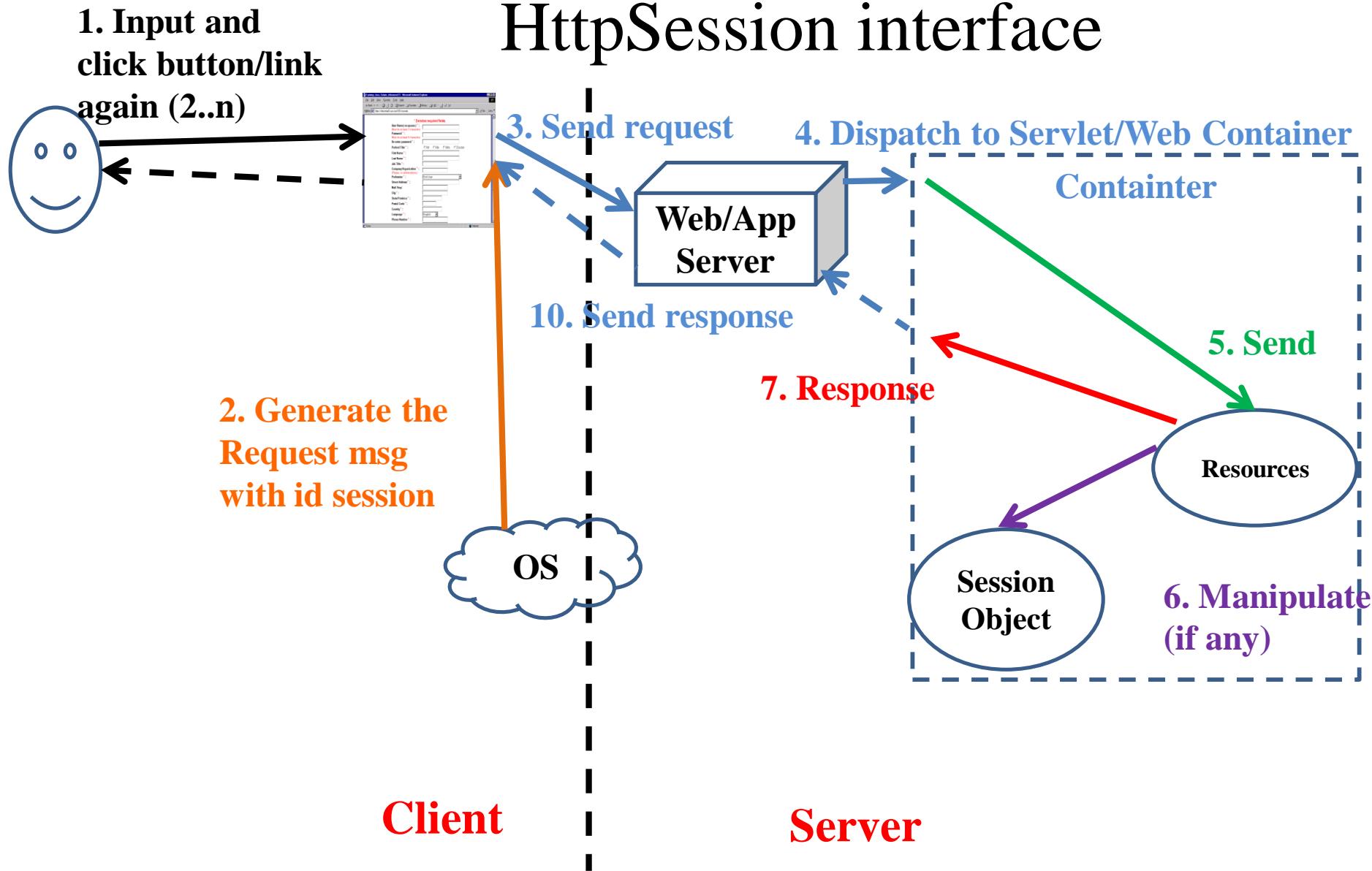
Sessions & Listeners

HttpSession interface



Sessions & Listeners

HttpSession interface



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

Sessions & Listeners

HttpSession interface Methods

Methods	Descriptions
isNew	<ul style="list-style-type: none"> - public boolean isNew() - Returns true if the client is unaware about the session or choose not to be part of the session
invalidate	<ul style="list-style-type: none"> - public void invalidate() - Invalidates the session and the objects bound to the session are bounded. This method throws IllegalStateException if called on already invalidated session - To avoid the hacker from causing any harm - 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

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

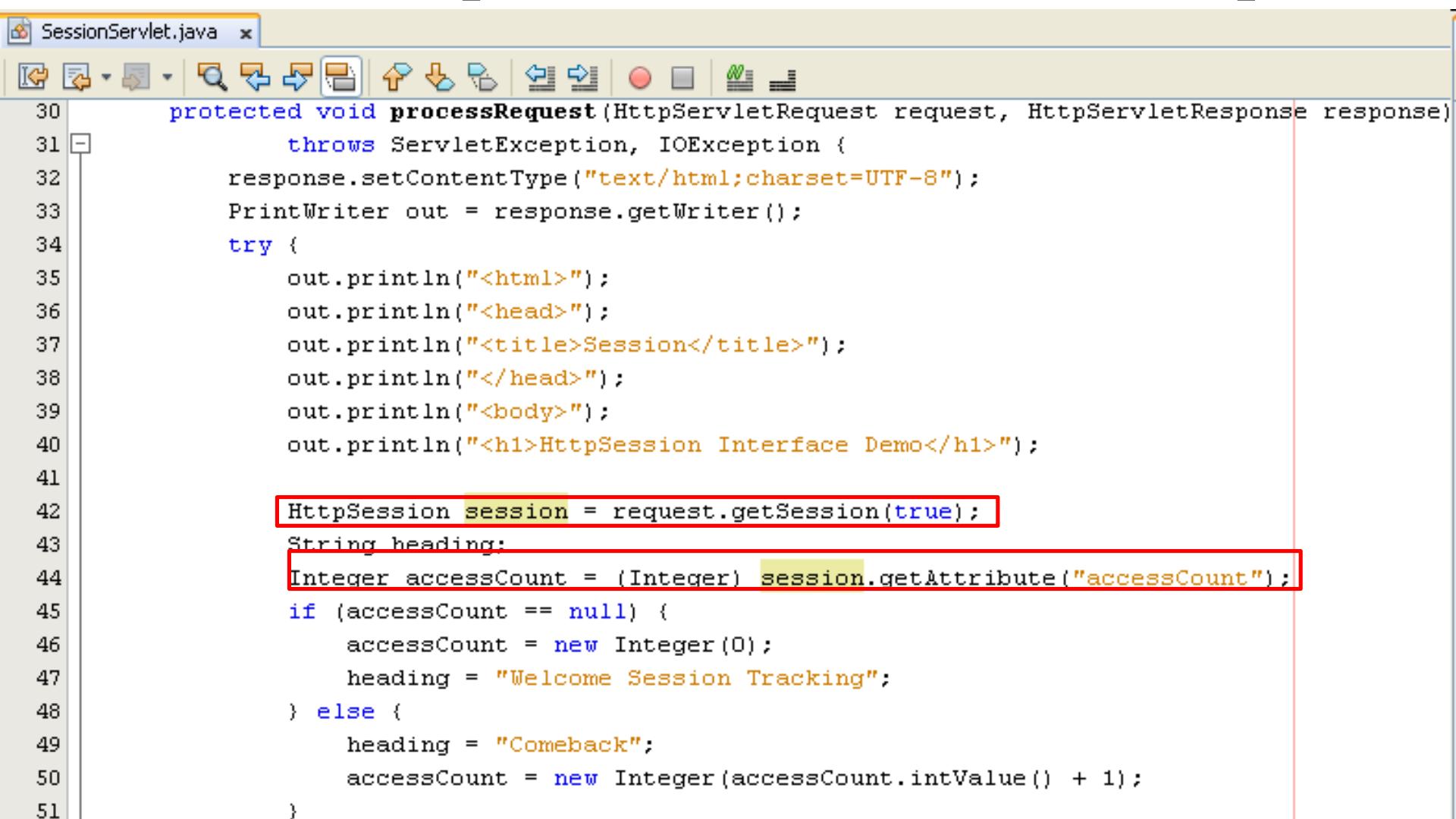
Welcome Session Tracking

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 `HttpSession` interface to track session access counts. Lines 42 and 44 are highlighted with a red box.

```
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

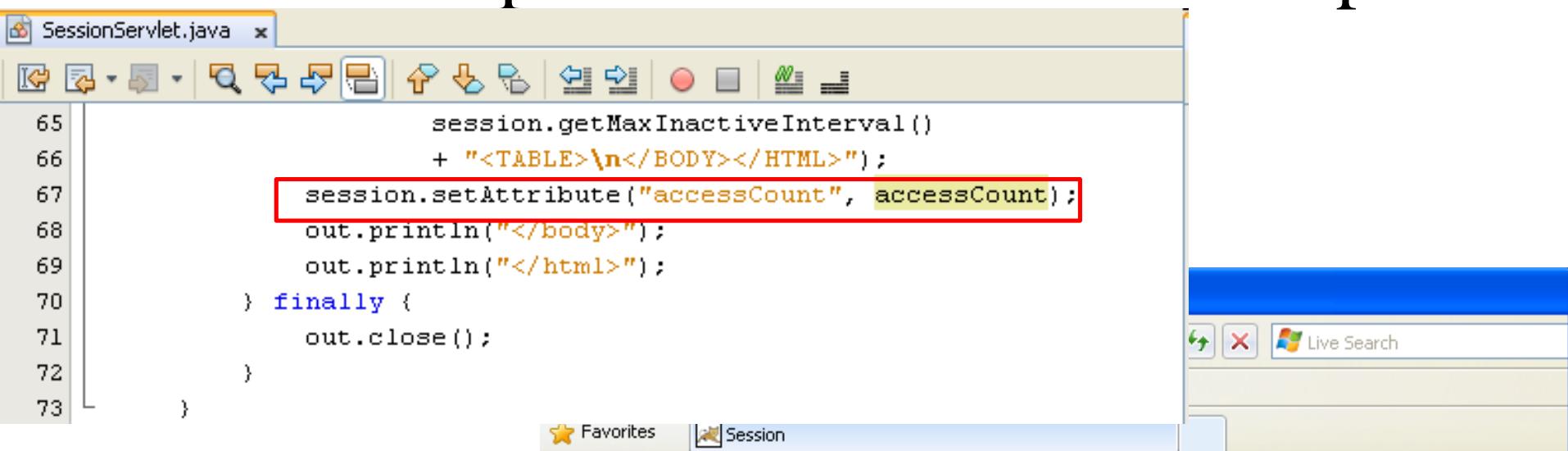
HttpSession interface – Example

```
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 a Java code editor with the file `SessionServlet.java` open. The code demonstrates the use of the `HttpSession` interface. A specific line of code, `session.setAttribute("accessCount", accessCount);`, is highlighted with a red rectangle.

```

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     }

```

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

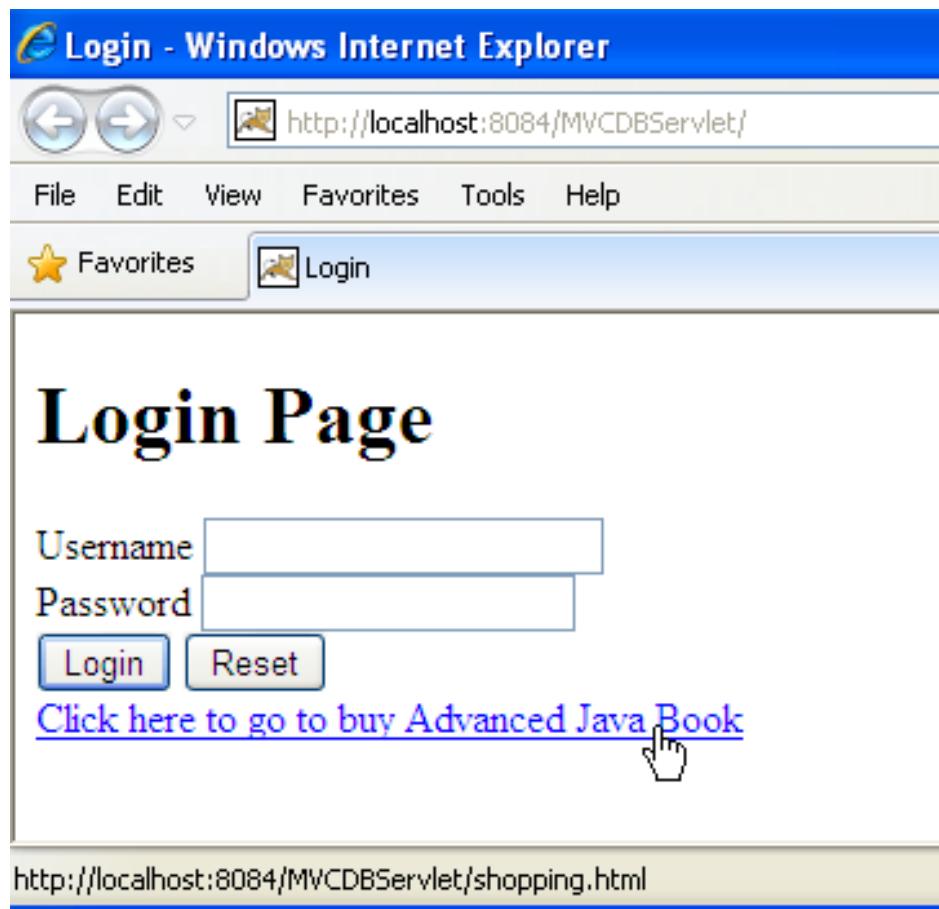
- **Distributed Session**
 - A session is **available** to be **shared between web resources** in a single web application (*e.g. a session *cannot cross web application boundaries**)
- Session Death is **controlled** in one of 3 ways
 - Application Server Global Default
 - Web Application Default (minutes)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



- 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

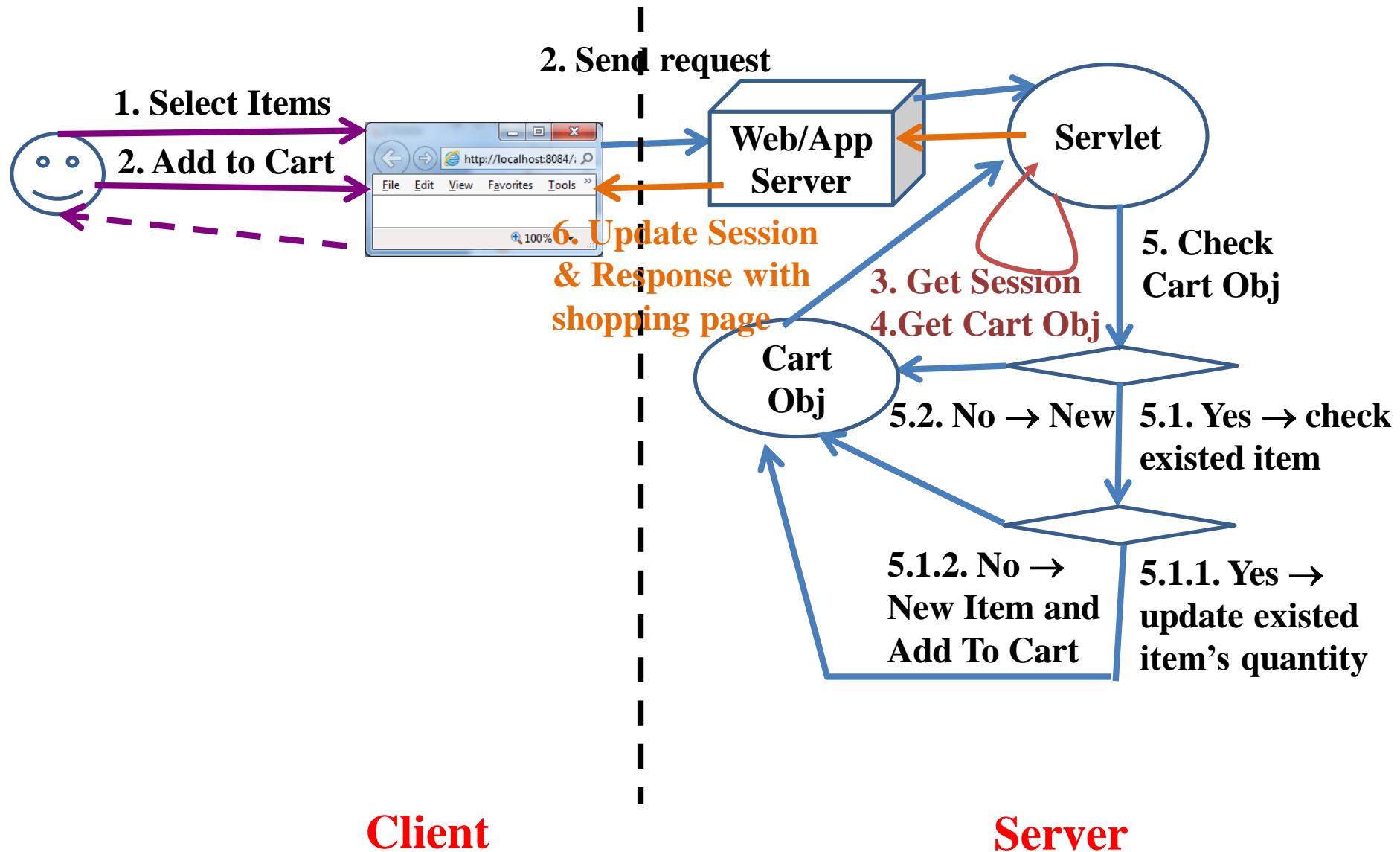
The screenshot shows a Windows Internet Explorer window with the title "Book - Windows Internet Explorer". The URL in the address bar is <http://localhost:8084/MVCDBServlet/shopping.html>. The menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar includes Back, Forward, Stop, Refresh, and Home buttons. A Favorites button is visible, and the Book icon is selected in the toolbar.

The main content area displays the "Book Store" page. At the top right, there is a dropdown menu set to "Book" with options "Servlet", "J2EE", "JavaEE", "Tomcat", "JBoss", "JDBC", "EJB2", "EJB3", and "Others". Below this is a form with two buttons: "Add Book To Cart" and "View Your Cart".

Below the browser window, a larger "Book Store" page is shown. It features a similar header with a dropdown menu set to "Book" and buttons for "Add", "Servlet", "J2EE", "JavaEE", "Tomcat", "JBoss", "JDBC", "EJB2", "EJB3", and "Others". The "Add" button is highlighted with a red border. To its right is a "View Your Cart" button.

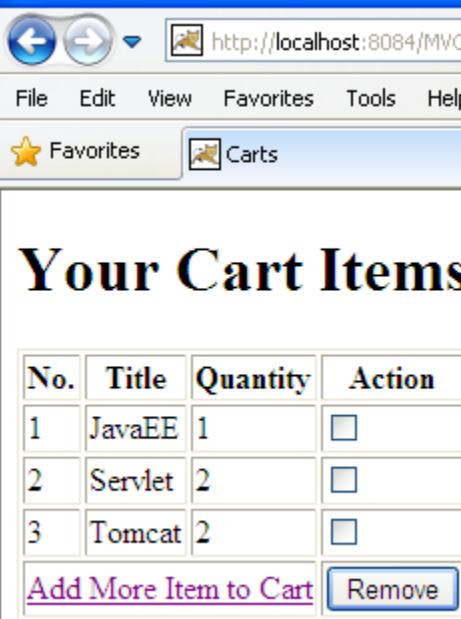
How to write CRUD Web Application

Interactive Server Model – Add To Cart



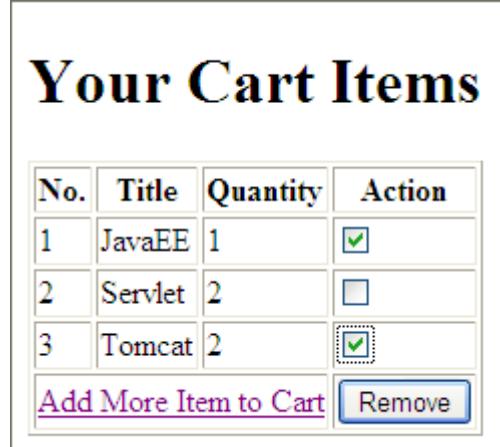
How to write CRUD Web Application

Shopping Cart – View Cart



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](#)

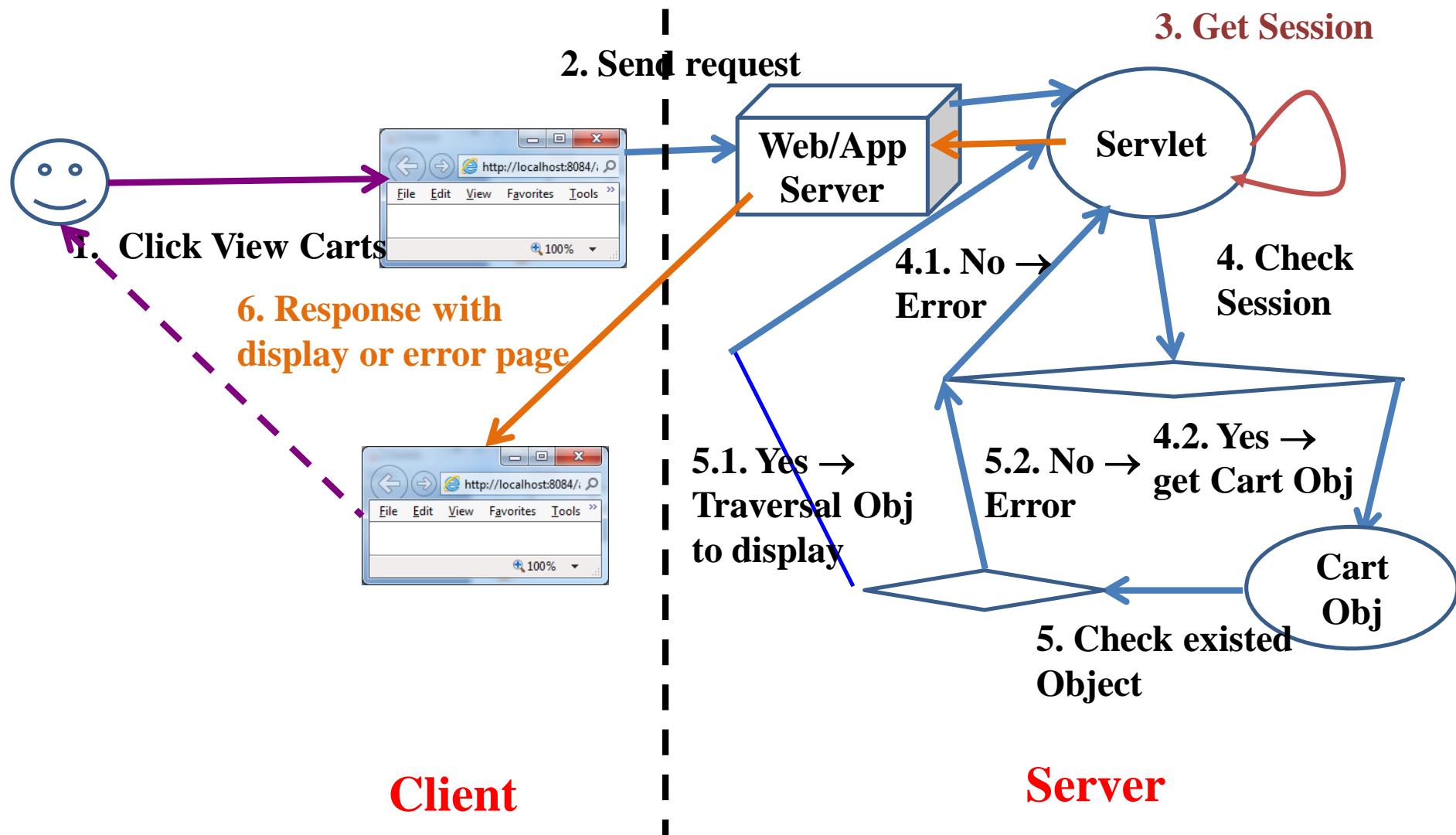


No.	Title	Quantity	Action
1	JavaEE	1	<input checked="" type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input type="checkbox"/>

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

How to write CRUD Web Application

Interactive Server Model – View Cart



How to write CRUD Web Application

Shopping Cart – Remove Cart

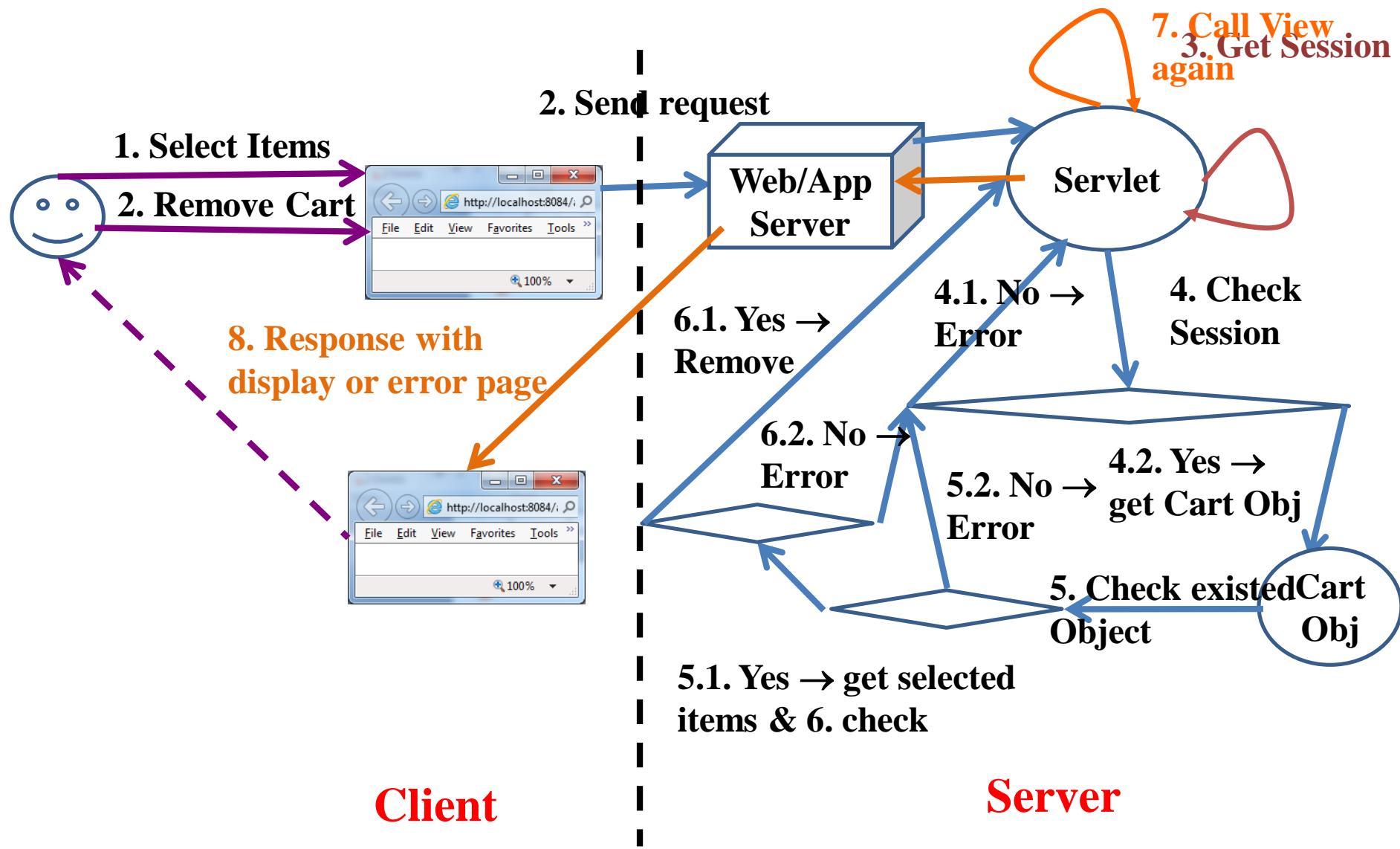
Your Cart Items			
No.	Title	Quantity	Action
1	JavaEE	1	<input checked="" type="checkbox"/>
2	Servlet	2	<input type="checkbox"/>
3	Tomcat	2	<input checked="" type="checkbox"/>
Add More Item to Cart		Remove	

The screenshot shows a web browser window with the URL `http://localhost:8084/MVCDBServlet/ProcessServlet?btAction=View%20Your%20Cart`. The browser's title bar and menu bar are visible. The main content area displays the same "Your Cart Items" table as above, but it now only contains one item: Servlet with quantity 2. The "JavaEE" row has been removed, indicated by the absence of its checkbox in the "Action" column.

No.	Title	Quantity	Action
1	Servlet	2	<input type="checkbox"/>
Add More Item to Cart		Remove	

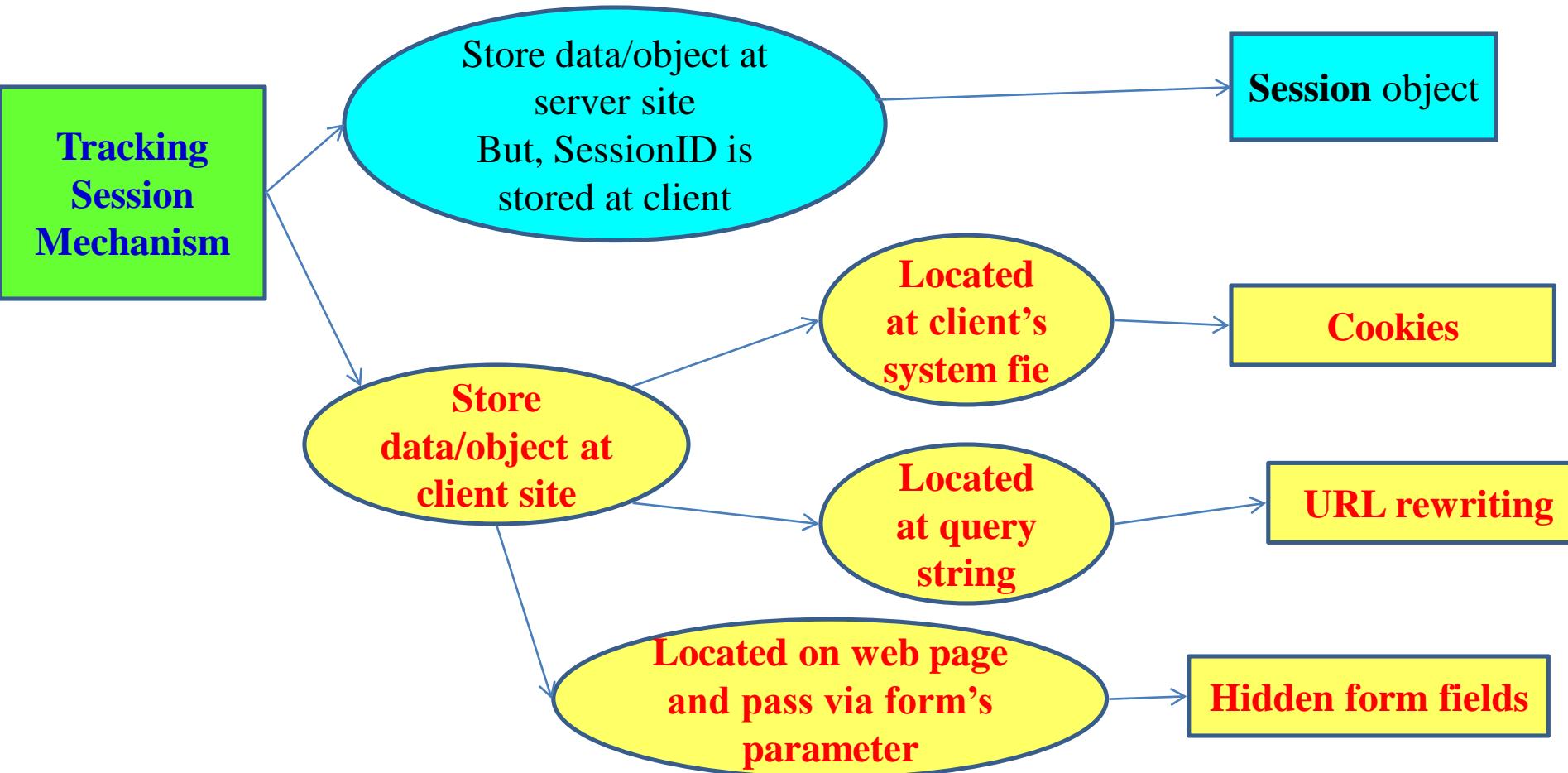
How to write CRUD Web Application

Interactive Server Model – Remove



Sessions & Listeners

Conclusion



Error Handling in Servlet

Reporting Error

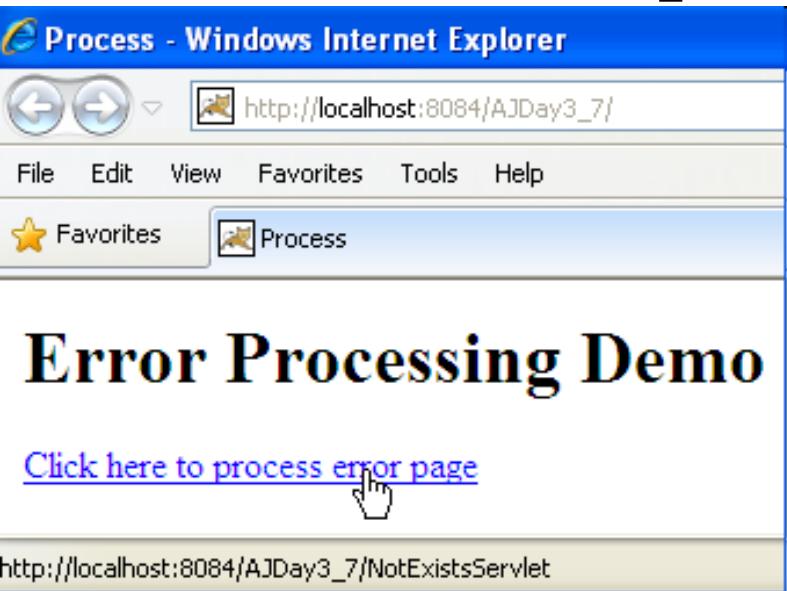
- There are many situations occur an error
 - A requested page may be moved from one location to another.
 - The address may be wrongly typed.
 - The requested page may be forbidden, may be temporarily deleted or correct HTTP version might not have found.
 - There are other situations where an error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

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

Error Handling in Servlet

Reporting Error – Example

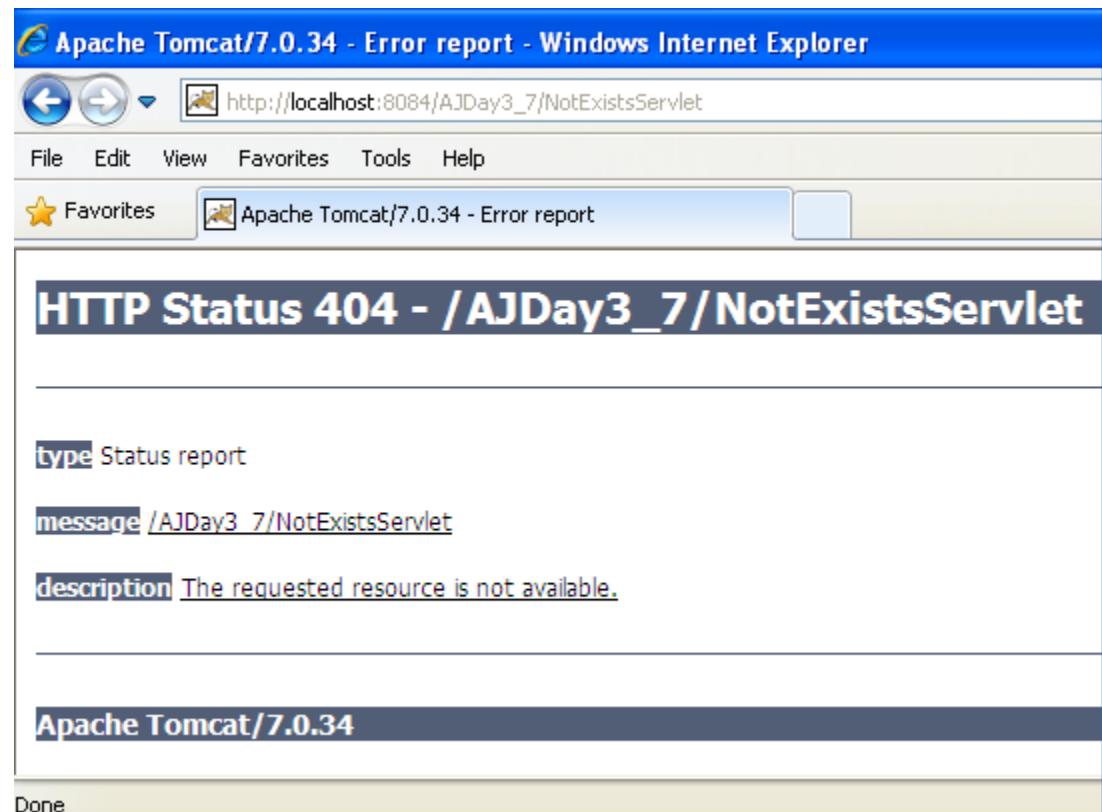
Process - Windows Internet Explorer



The screenshot shows a Windows Internet Explorer window with the following details:

- Title bar: Process - Windows Internet Explorer
- Address bar: http://localhost:8084/AJDay3_7/
- Menu bar: File, Edit, View, Favorites, Tools, Help
- Toolbar: Back, Forward, Stop, Home, Favorites, Process
- Content area:
 - Section title: Error Processing Demo
 - Text: Click here to process error page
 - Link: http://localhost:8084/AJDay3_7/NotExistsServlet

Apache Tomcat/7.0.34 - Error report - Windows Internet Explorer



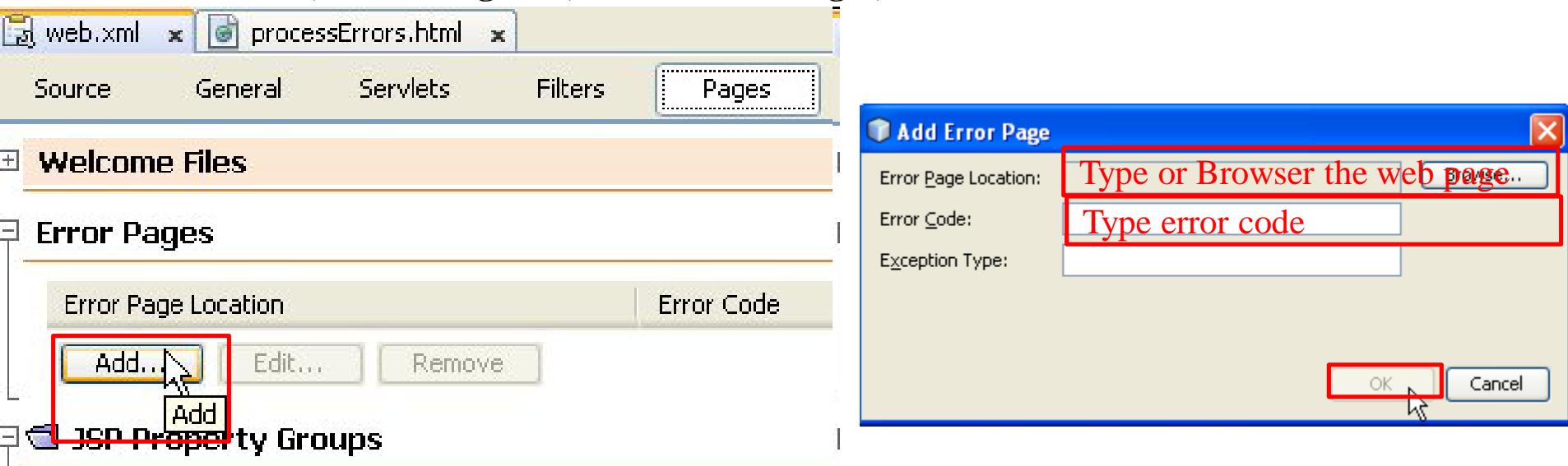
The screenshot shows an Apache Tomcat 7.0.34 error report page with the following details:

- Title bar: Apache Tomcat/7.0.34 - Error report - Windows Internet Explorer
- Address bar: http://localhost:8084/AJDay3_7/NotExistsServlet
- Menu bar: File, Edit, View, Favorites, Tools, Help
- Toolbar: Back, Forward, Stop, Home, Favorites, Apache Tomcat/7.0.34 - Error report
- Content area:
 - Section title: HTTP Status 404 - /AJDay3_7/NotExistsServlet
 - Text: type Status report
 - Text: message /AJDay3_7/NotExistsServlet
 - Text: description The requested resource is not available.
- Footer: Apache Tomcat/7.0.34

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
/processErrors.html	404
Add...	Edit...

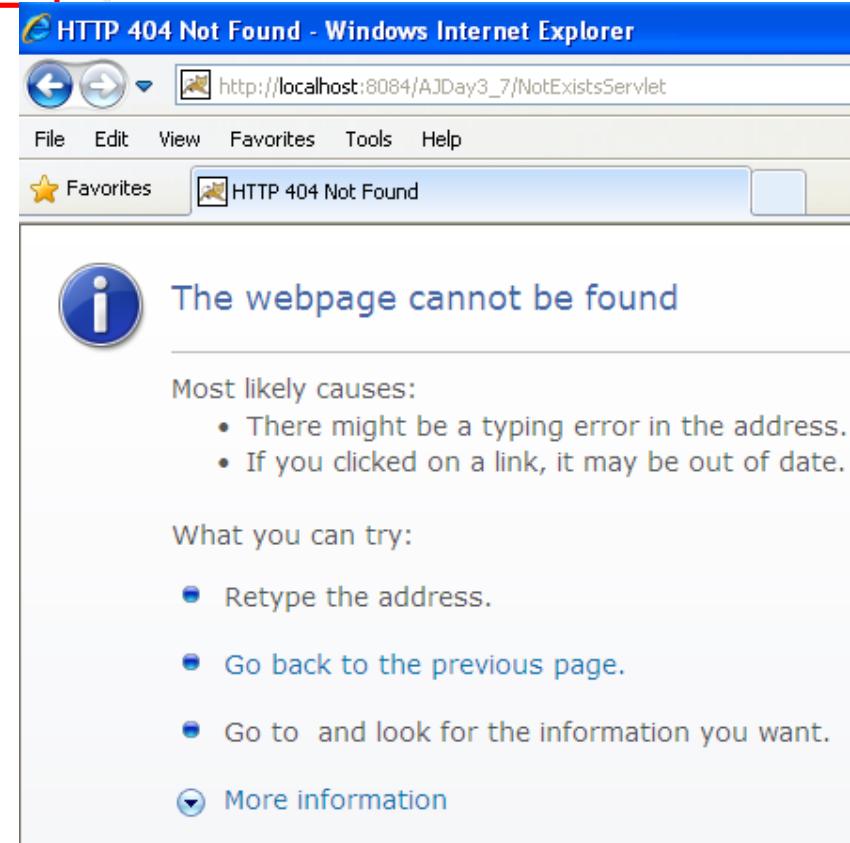
Error Handling in Servlet

Reporting Error – Example

web.xml

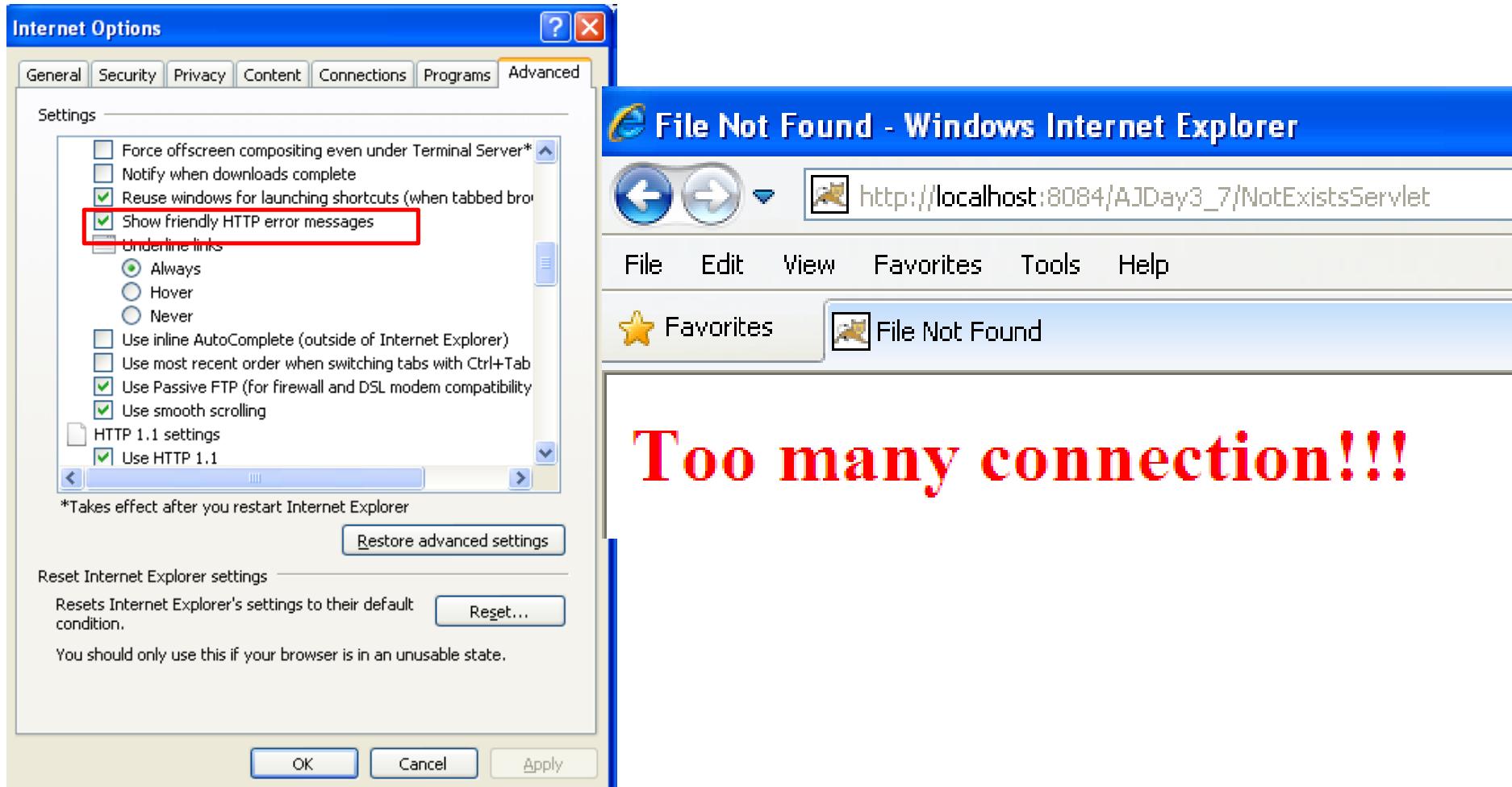
Source General Servlets Filters Pages References Security

```
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

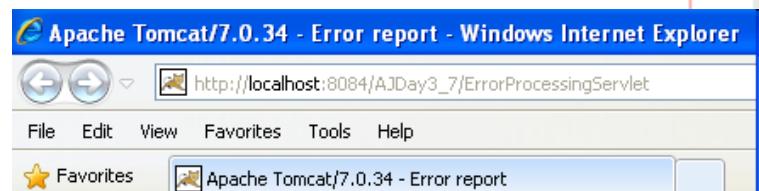
Reporting Error – Example

Source History | 

```

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

```



HTTP Status 500 - For input string: "a"

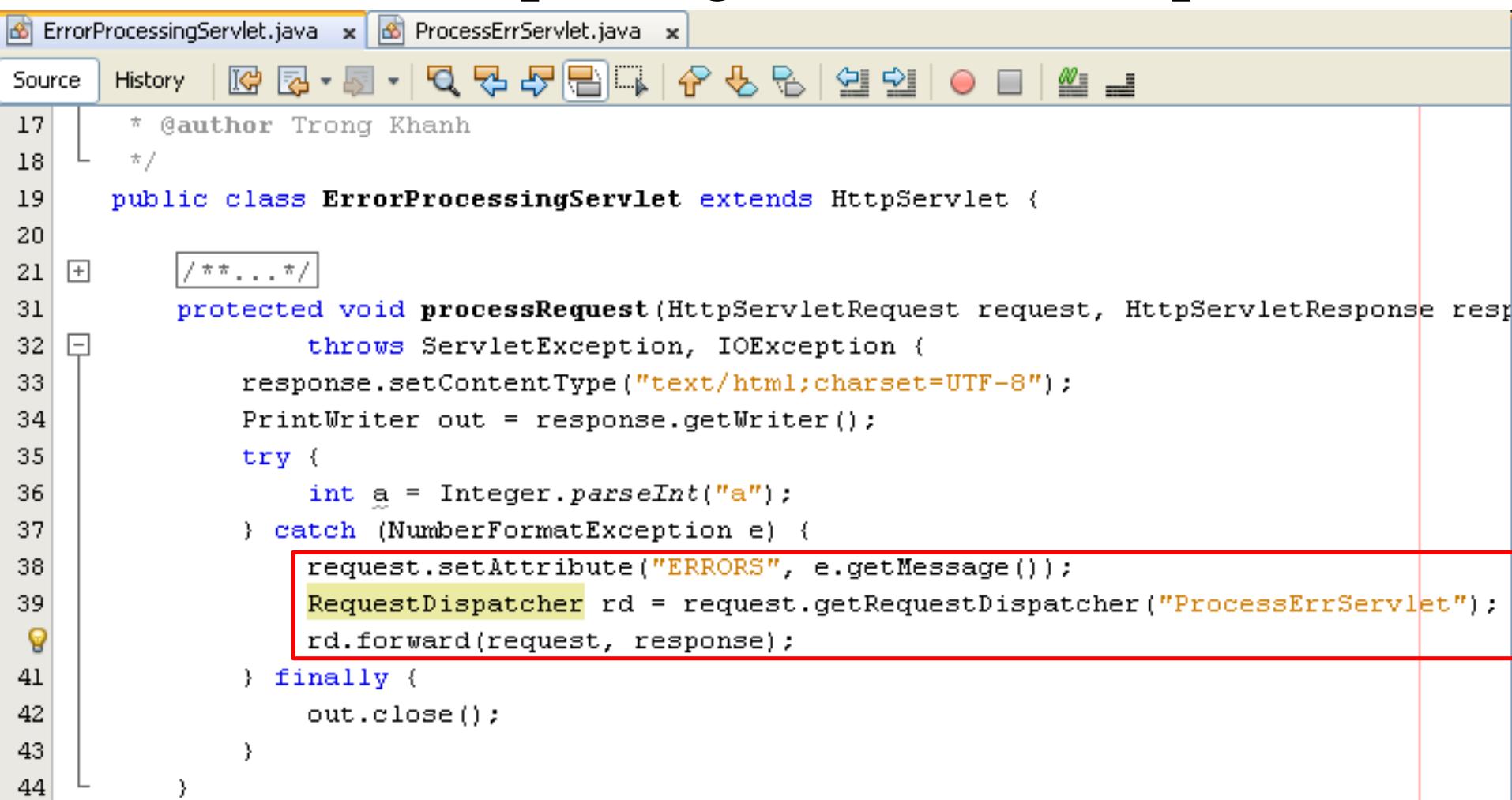
type Status report

message For input string: "a"

description The server encountered an internal error that prevented it from fulfilling this request.

Error Handling in Servlet

Reporting Error – Example



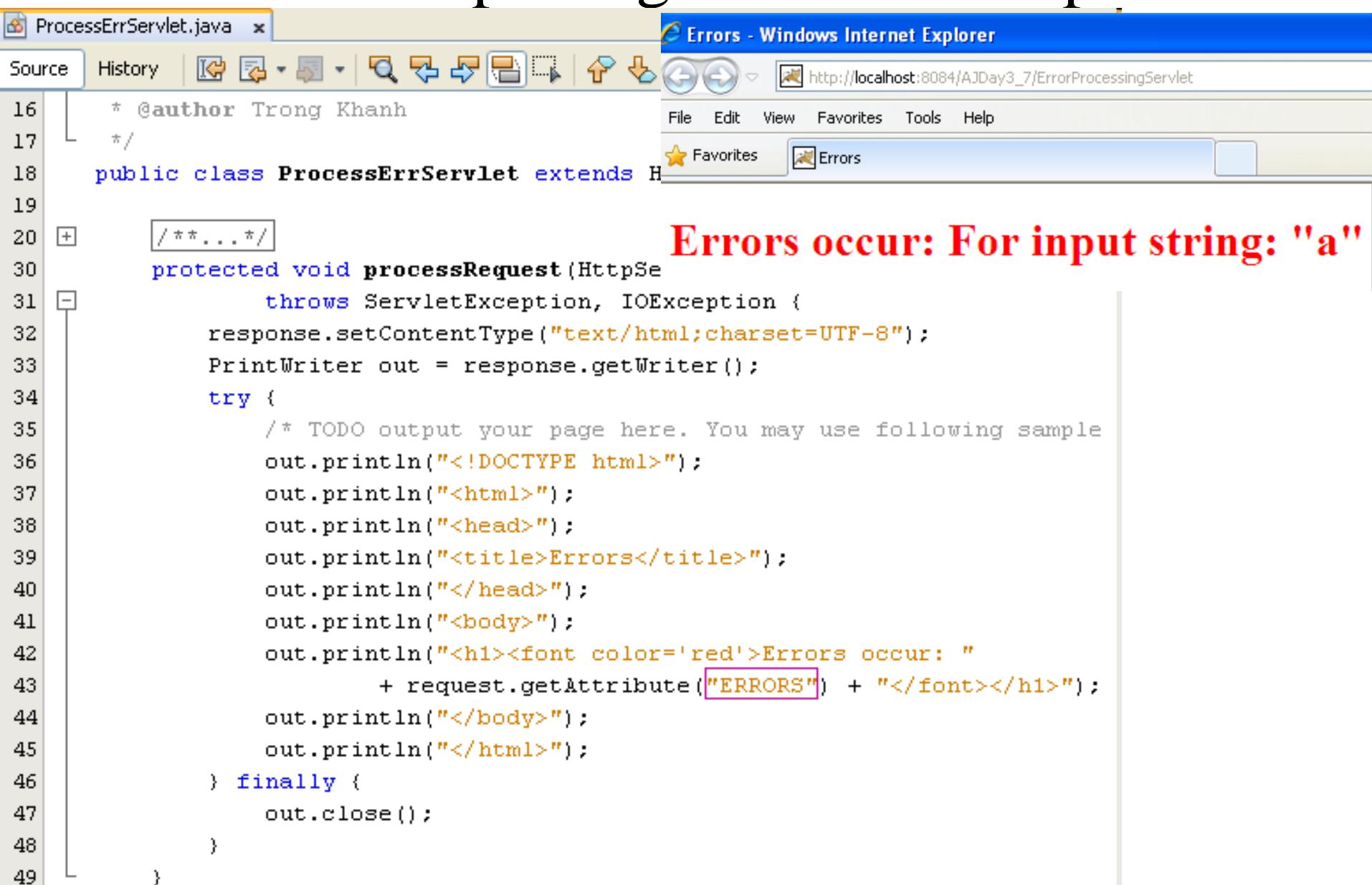
The screenshot shows a Java code editor with two tabs: "ErrorProcessingServlet.java" and "ProcessErrServlet.java". The "ErrorProcessingServlet.java" tab is active, displaying the following code:

```
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 section of code from line 33 to line 66 is highlighted with a red rectangle.

Error Handling in Servlet

Reporting Error – Example



The screenshot shows a Java IDE (NetBeans) on the left displaying the code for `ProcessErrServlet.java`, and a web browser (Internet Explorer) on the right showing the resulting page.

Code (ProcessErrServlet.java):

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

Browser Output:

Errors - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites Errors

http://localhost:8084/AJDay3_7/ErrorProcessingServlet

Errors occur: For input string: "a"

Error Handling in Servlet

Reporting Error – Example

Reporting Error – Example

The screenshot shows a Java IDE interface with the title "DisplayErrorServlet.java" in the top bar. The code editor displays the following Java code:

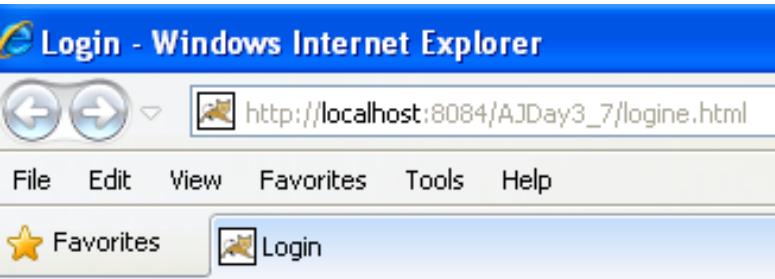
```
15
16     * @author Trong Khanh
17     */
18
19     public class DisplayErrorServlet extends HttpServlet {
20
21         /**
22
23         * @param request the servlet request
24         * @param response the servlet response
25         * @throws ServletException if an error occurs
26         */
27
28         protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30
31             response.setContentType("text/html;charset=UTF-8");
32
33             PrintWriter out = response.getWriter();
34
35             try {
36                 String action = request.getParameter("action");
37
38                 if (action == null) {
39
40                     response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);
41
42                     String url = "http://" + request.getLocalName() + ":"
43                         + request.getLocalPort() + request.getContextPath() +
44                         "/logine.html";
45
46                     response.setHeader("Location", url);
47
48                 }
49
50             } finally {
51                 out.close();
52             }
53
54         }
55
56     }
```

The code highlights several parts of the code with different colors and boxes:

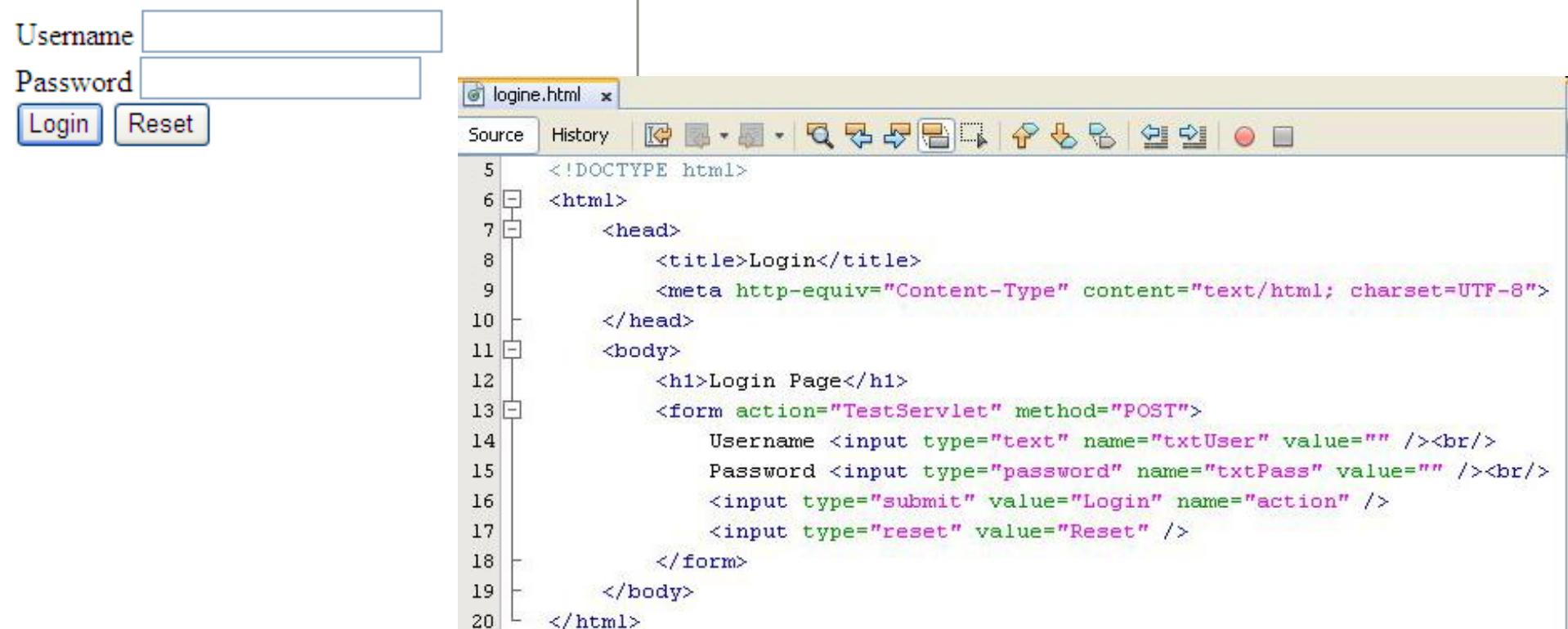
- Line 30: `protected void processRequest(HttpServletRequest request, HttpServletResponse response)` is highlighted in yellow.
- Line 37: `response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);` is highlighted in green.
- Line 38: The URL construction starting with `String url = "http://" + request.getLocalName() + ":"` is highlighted in blue.
- Line 41: `response.setHeader("Location", url);` is highlighted in red.

Error Handling in Servlet

Reporting Error – Example



Login Page

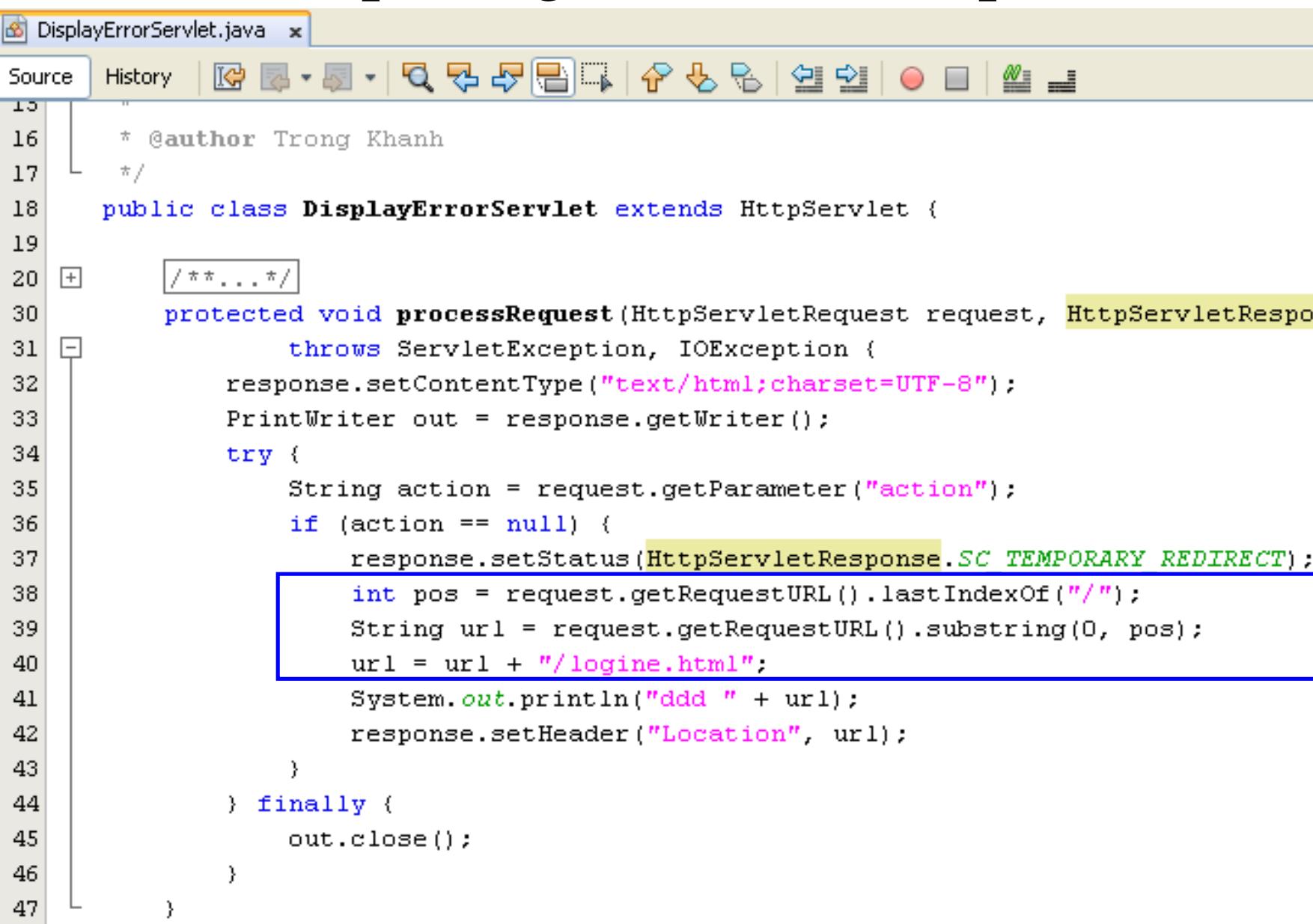


A screenshot of a code editor window titled "logine.html". The tab bar shows "Source" and "History". The code editor interface includes a toolbar with various icons for file operations like Open, Save, Copy, Paste, Find, and Print. The code itself is a simple HTML form:

```
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



The screenshot shows a Java code editor with the file `DisplayErrorServlet.java` open. The code implements a `HttpServlet` to handle requests and perform a temporary redirect if the parameter `action` is null.

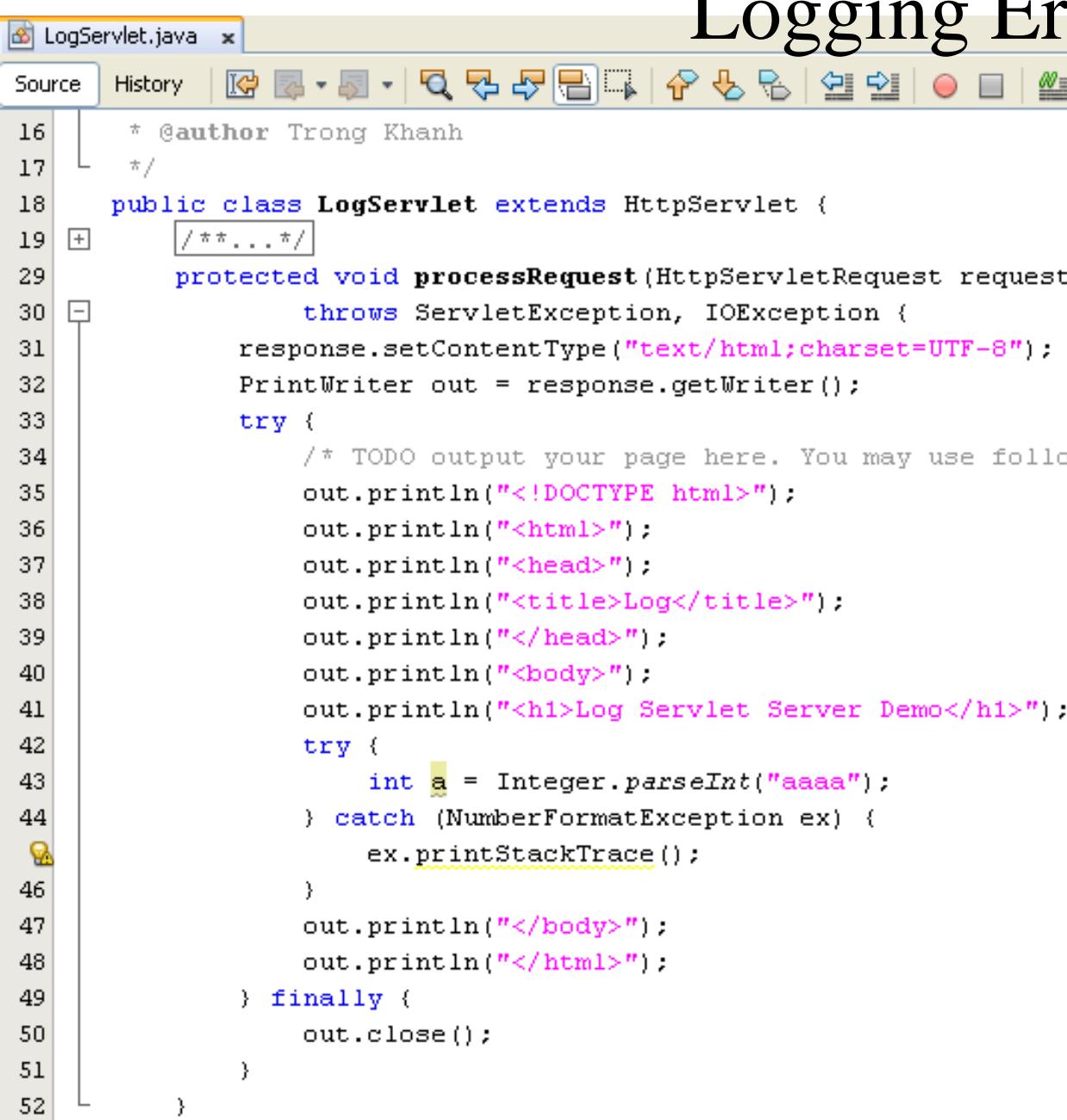
```
15
16     * @author Trong Khanh
17     */
18    public class DisplayErrorServlet extends HttpServlet {
19
20        /**
21         * ...
22         */
23
24        protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25            throws ServletException, IOException {
26            response.setContentType("text/html;charset=UTF-8");
27            PrintWriter out = response.getWriter();
28            try {
29                String action = request.getParameter("action");
30                if (action == null) {
31                    response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);
32                    int pos = request.getRequestURL().lastIndexOf("/");
33                    String url = request.getRequestURL().substring(0, pos);
34                    url = url + "/logine.html";
35                    System.out.println("ddd " + url);
36                    response.setHeader("Location", url);
37                }
38            } finally {
39                out.close();
40            }
41        }
42    }
```

Annotations and highlights in the code:

- `HttpServletResponse` and `SC TEMPORARY_REDIRECT` are highlighted in yellow.
- A blue rectangular box highlights the code block from line 37 to line 74, which performs the redirect logic.

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 an error. It includes a try-catch block that prints the stack trace of a `NumberFormatException`.

```
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

LogServlet.java

```

16  * @author Trong Khanh
17  */
18  public class LogServlet extends HttpServlet {
19      /**
20      */
21
22      protected void processRequest(HttpServletRequest request
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
28              out.println("<!DOCTYPE html>");
29              out.println("<html>");
30              out.println("<head>");
31              out.println("<title>Log</title>");
32              out.println("</head>");
33              out.println("<body>");
34              out.println("<h1>Log Servlet Server Demo</h1>");
35              try {
36                  int a = Integer.parseInt("aaaa");
37              } catch (NumberFormatException ex) {
38                  ex.printStackTrace();
39              }
40              out.println("</body>");
41              out.println("</html>");
42          } finally {
43              out.close();
44          }
45      }
46      /**
47      */
48      protected void doGet(HttpServletRequest request,
49          HttpServletResponse response) throws ServletException,
50          IOException {
51          processRequest(request, response);
52      }
53  }

```

Log - Windows Internet Explorer

File Edit View Favorites Tools Help

Favorites Log

Log Servlet Server Demo

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:**

...

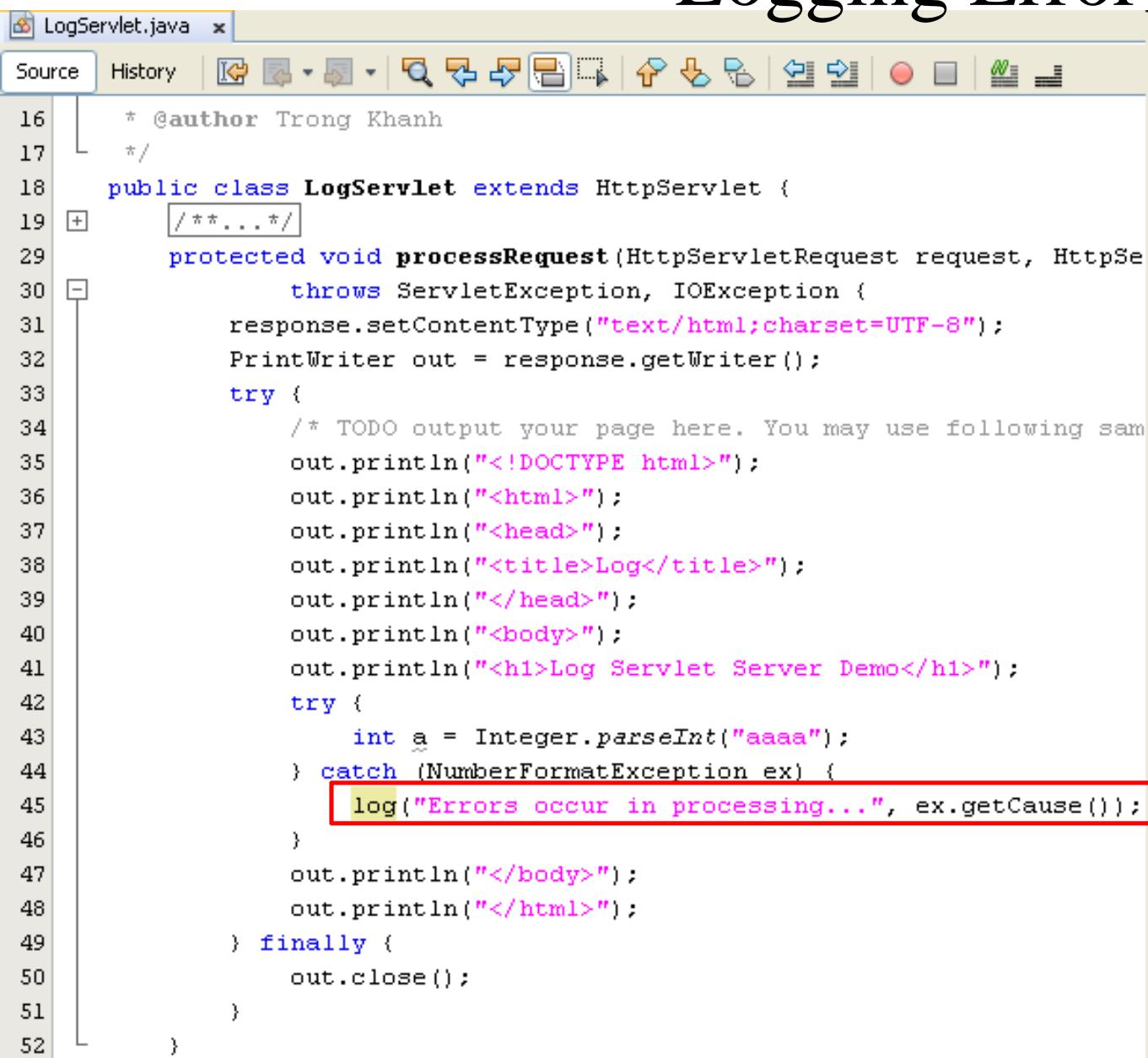
```
log("Servlet is not found ");
response.sendError(response.SC_INTERNAL_SERVER_ERROR, "The requested
page ["+ page + "] not found.");
```

...

- **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. 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 rules: **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 is a **Done** button.

Username*	(6 - 12 chars)
Password*	(8 - 20 chars)
Confirm*	
Full name*	(2 - 40 chars)

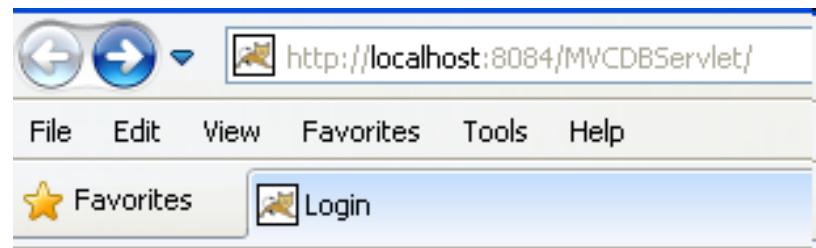
How to write CRUD Web Application

Validation



Errors occur

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



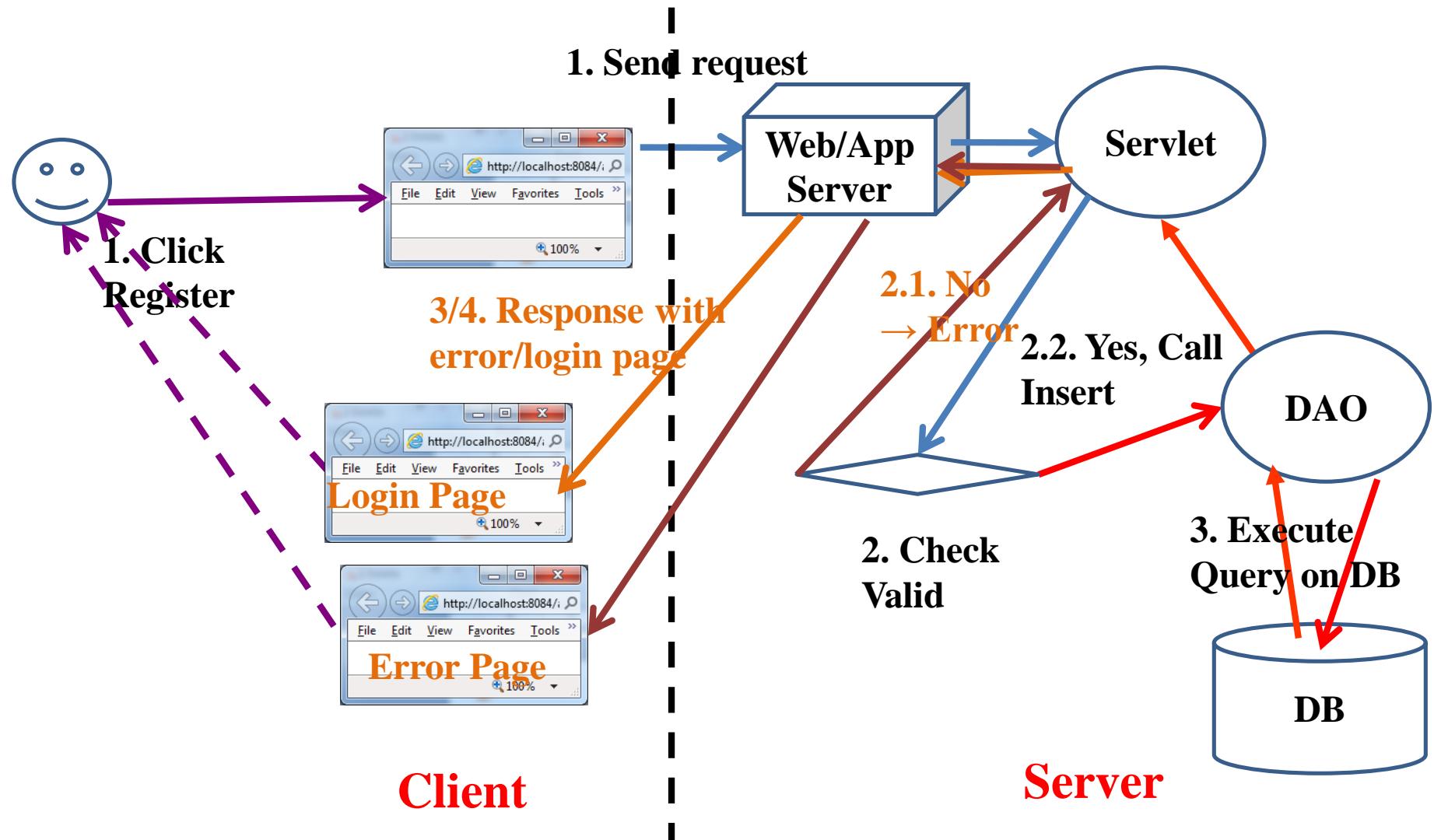
Login Page

Username

Password

How to write CRUD Web Application

Interactive Server Model



Summary

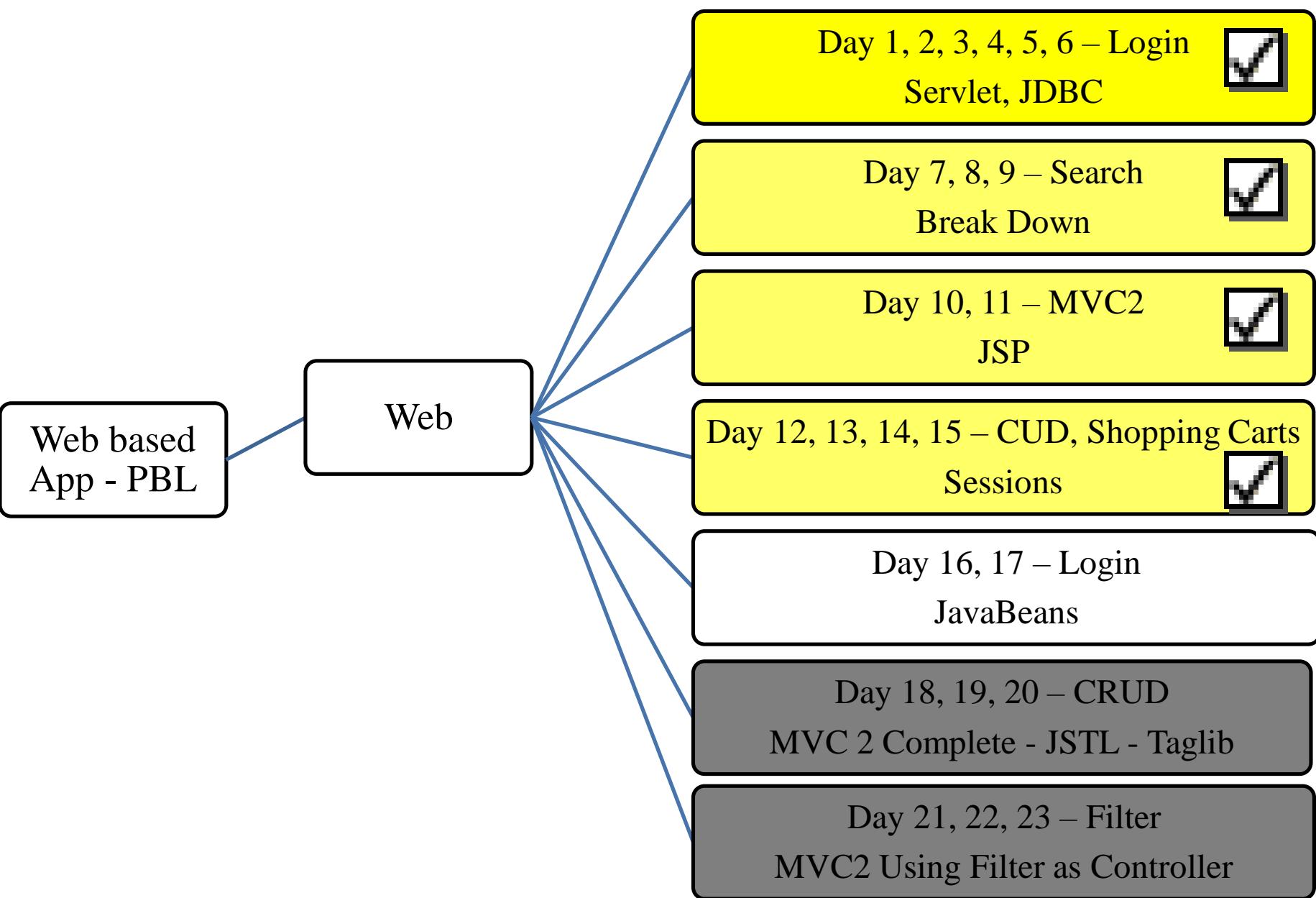
- **How to write CRUD Web Application**
 - Session Tracking Techniques
 - Manipulate DB Techniques in Web Application
 - Break down structure component in building web application
- **Techniques: Error Handling in Servlets**
 - Reporting Errors
 - Logging Errors
 - Users Errors vs. System Errors

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

The image displays three windows illustrating a shopping cart application using cookies:

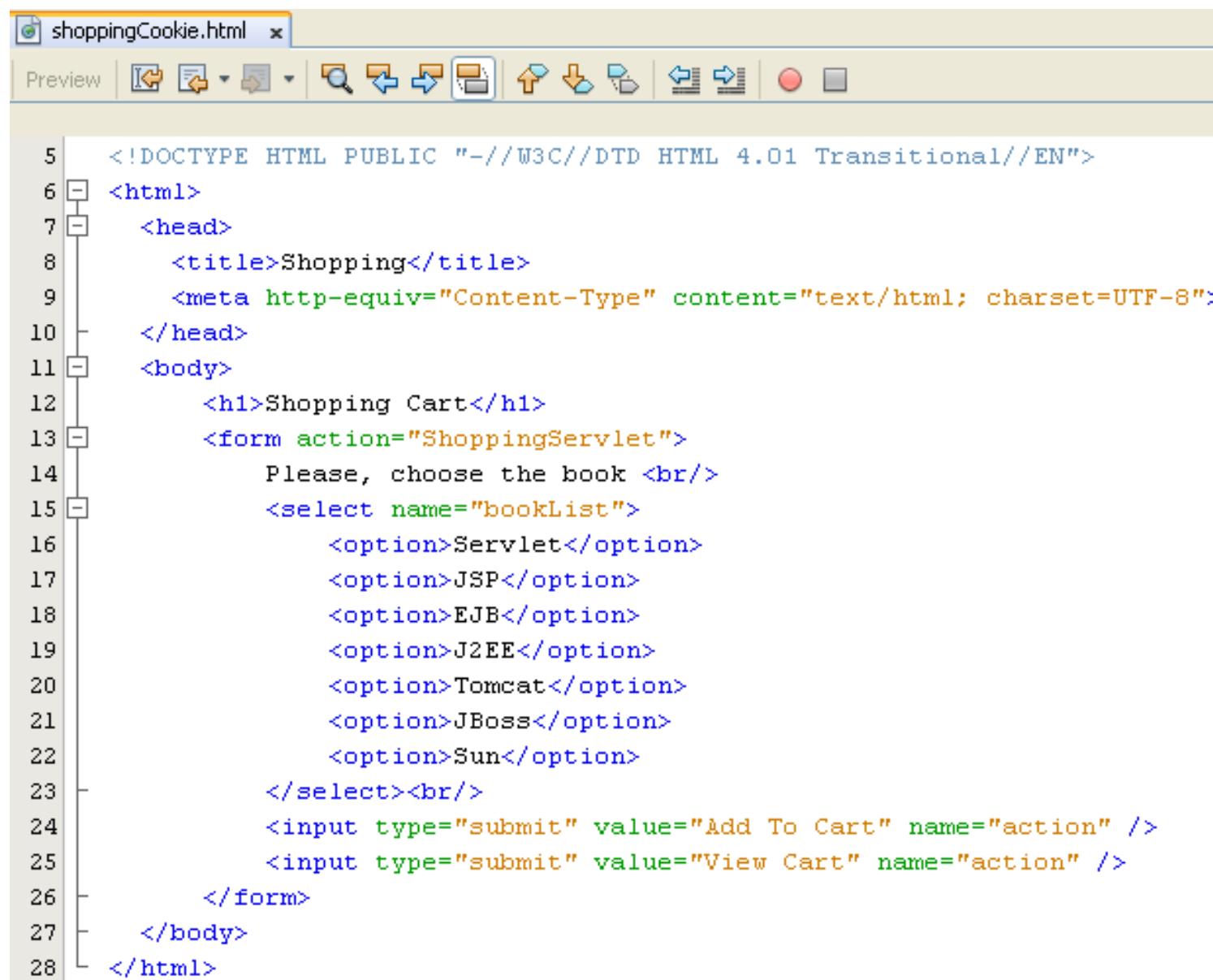
- Browser Window 1:** Shows the URL http://localhost:8084/AJDay3_7/shoppingCookie.html. The page title is "Shopping Cart". It displays a message: "Please, choose the book" followed by a dropdown menu set to "Servlet" and two buttons: "Add To Cart" and "View Cart".
- Browser Window 2:** Shows the URL http://localhost:8084/AJDay3_7/ShoppingServlet?action=View%20Cart. The page title is "Your Shopping Cart". It displays a table showing the items in the shopping cart:

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

 Below the table are two buttons: "Add More Cart" and "Remove Cart".
- Notepad Window:** Shows a list of session IDs and their corresponding URLs. The list includes:
 - Servlet3localhost/AJDay3_7/102480359193630401284210789232
 - 210703923230401283*Tomcat2localhost/AJDay3_7/1024883591936304012842184699232
 - 30401283*J2EE1localhost/AJDay3_7/1024913591936304012842210789232
 - 30401283*JSP1localhost/AJDay3_7/1024933591936304012842237199232
 - 30401283*Sun2localhost/AJDay3_7/10241003591936304012842303289232
 - 30401283*JBoss0localhost/AJDay3_7/10241083591936304012842379539232
 - 30401283*

Appendix

Shopping Cart using Cookies

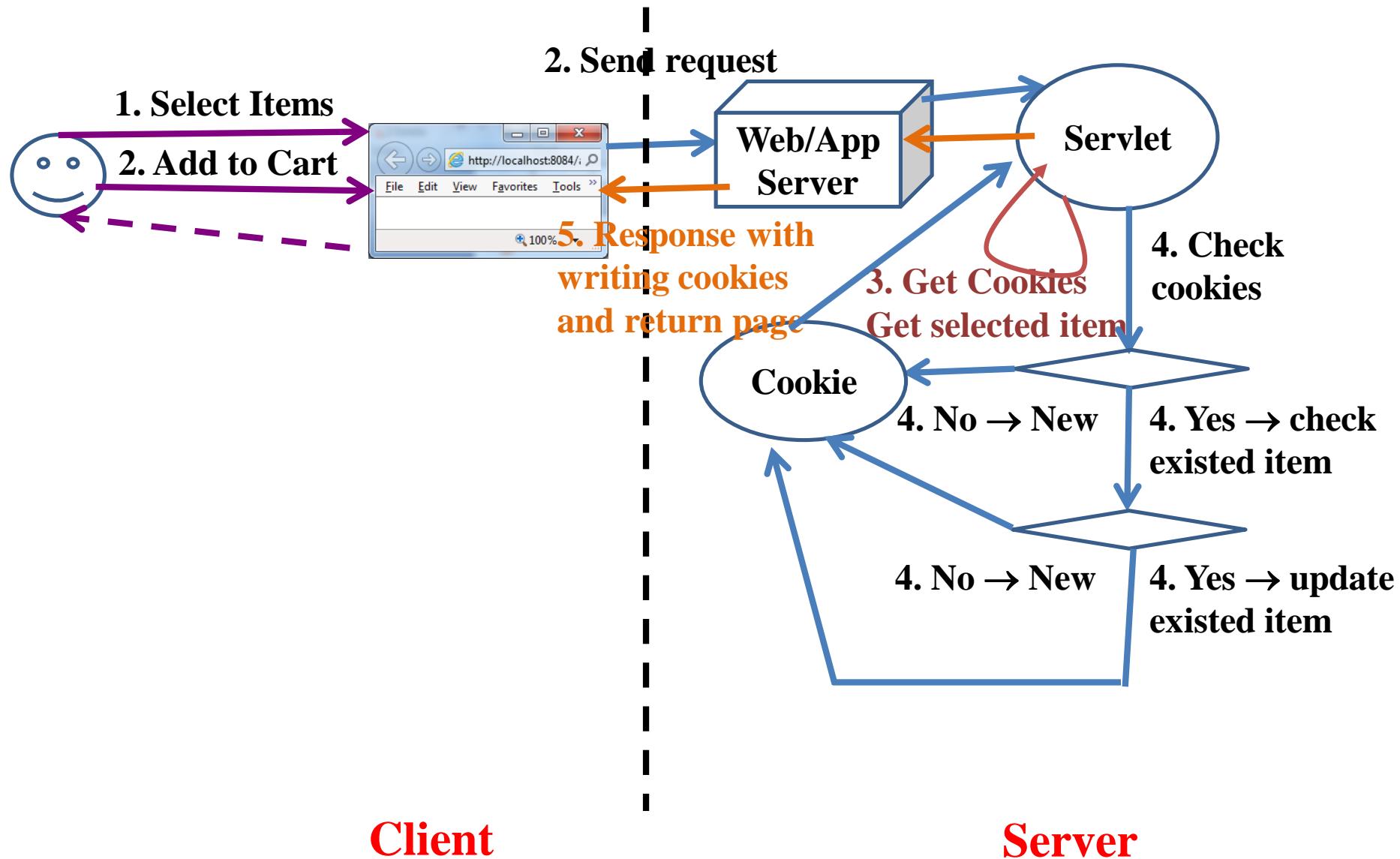


The screenshot shows a code editor window titled "shoppingCookie.html". The window has a toolbar with various icons for file operations like Open, Save, and Print. The main area displays the following HTML code:

```
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>
```

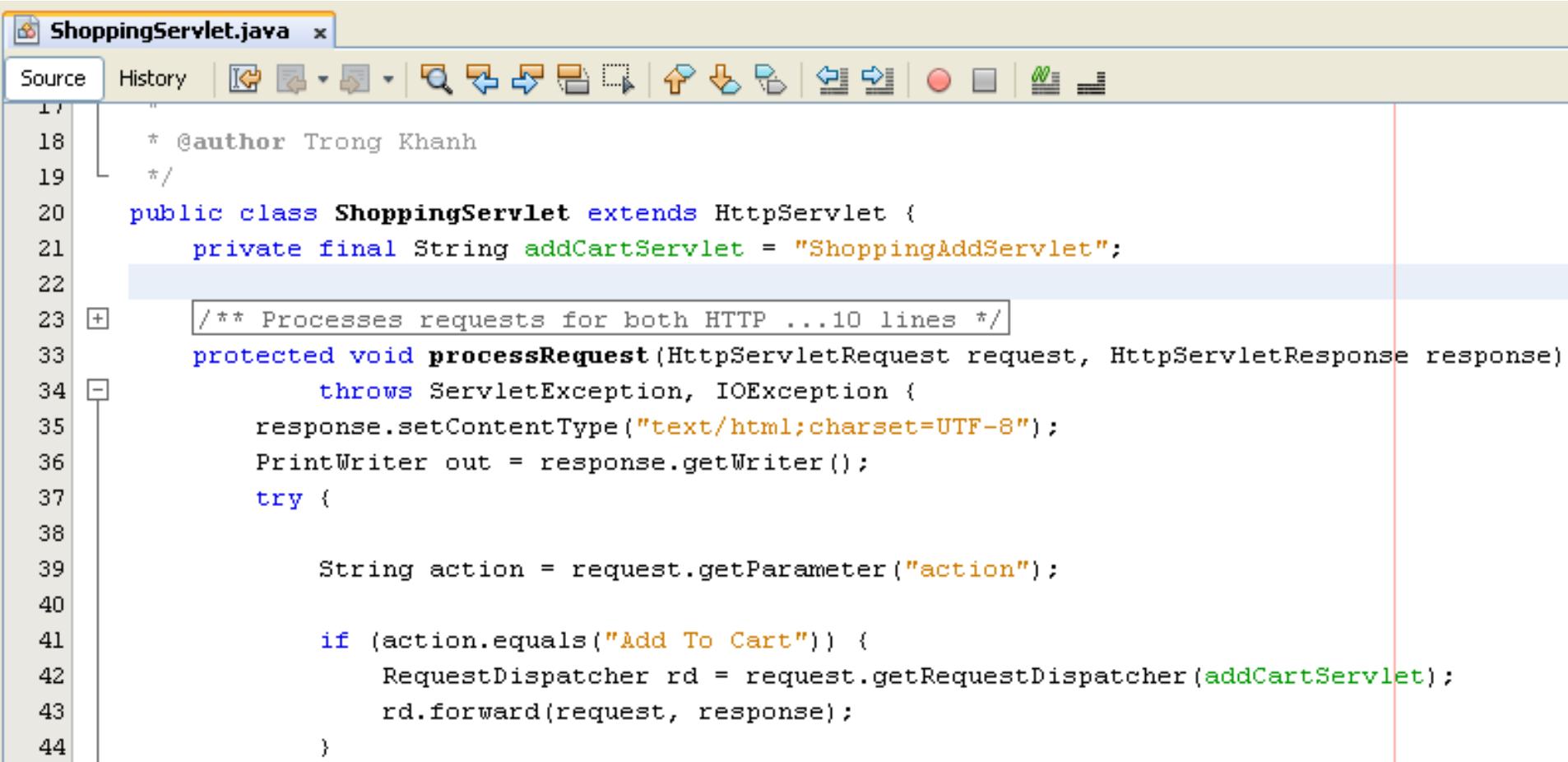
Appendix

Interactive Server Model



Appendix

Shopping Cart using Cookies

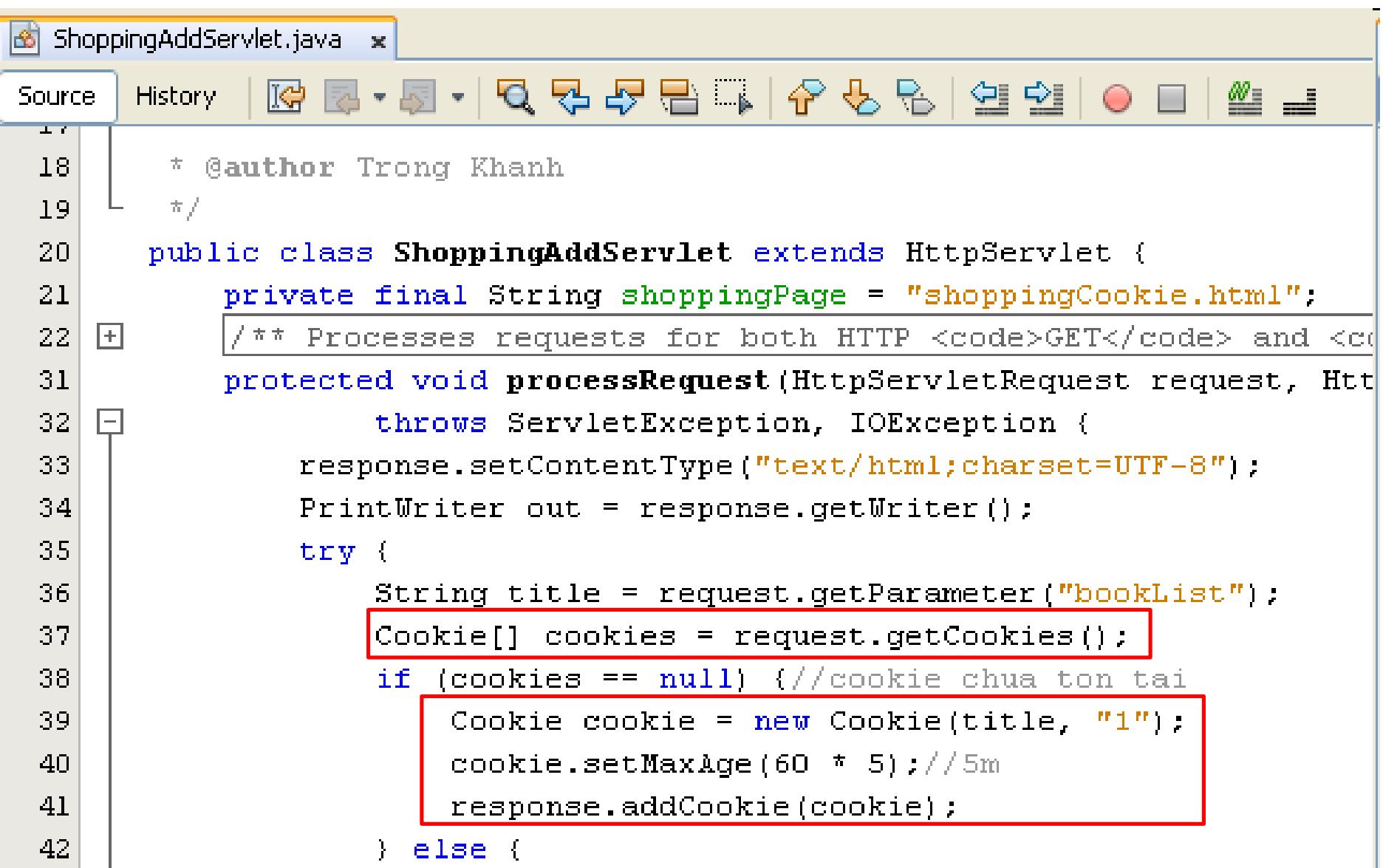


The screenshot shows a Java code editor window titled "ShoppingServlet.java". The code implements a HttpServlet for a shopping cart application, utilizing RequestDispatcher and Cookies.

```
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 servlet for adding items to a shopping cart using cookies.

```
17  
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>  
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                String title = request.getParameter("bookList");  
37                Cookie[] cookies = request.getCookies();  
38                if (cookies == null) // cookie chua ton tai  
39                    Cookie cookie = new Cookie(title, "1");  
40                    cookie.setMaxAge(60 * 5); // 5m  
41                    response.addCookie(cookie);  
42            } else {  
43            }  
44        }  
45    }
```

The code highlights several sections of the code with red boxes:

- A red box surrounds the line `Cookie[] cookies = request.getCookies();`.
- A red box surrounds the entire block of code starting with `if (cookies == null)` up to `response.addCookie(cookie);`.

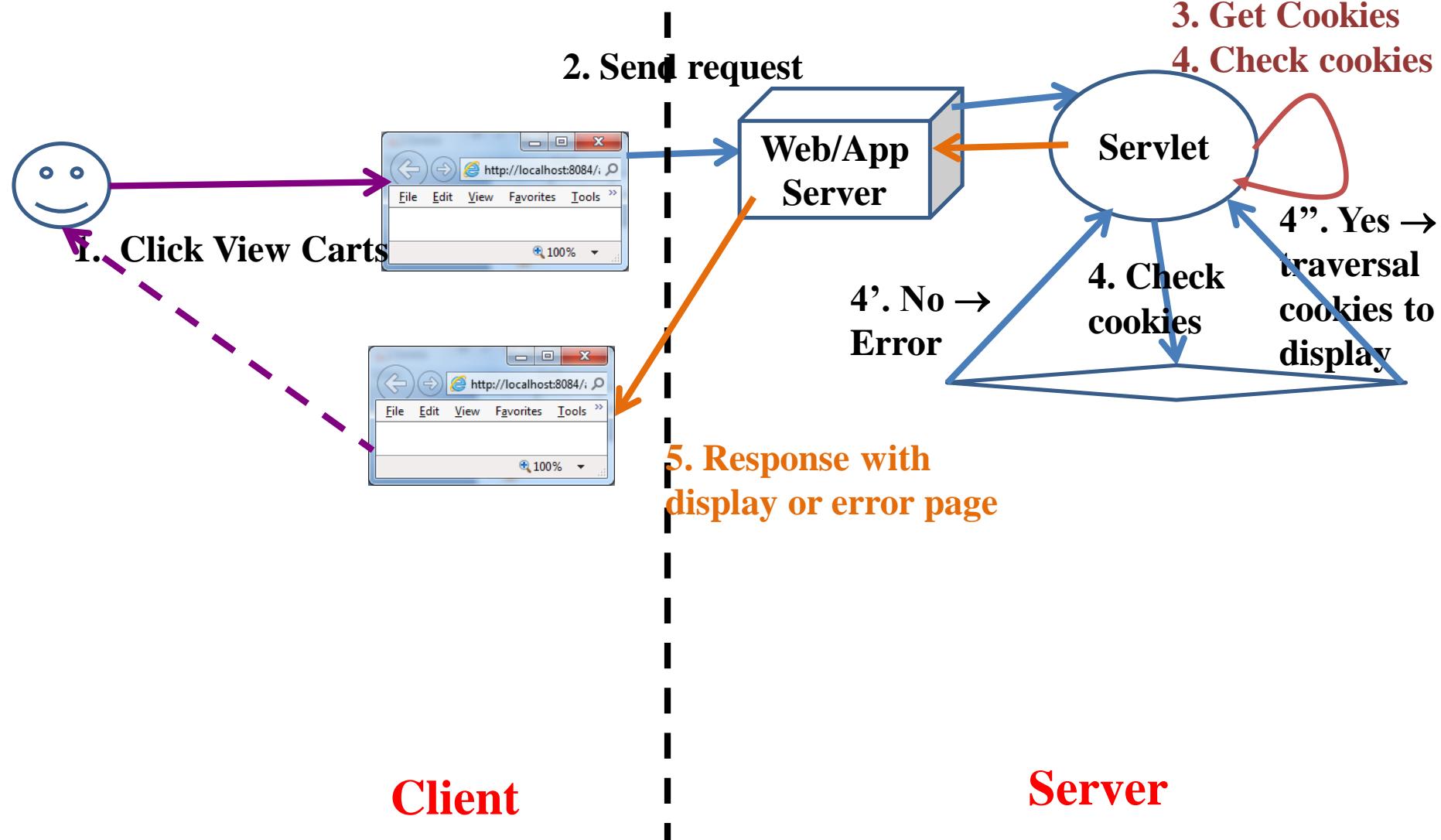
Appendix

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 }
```

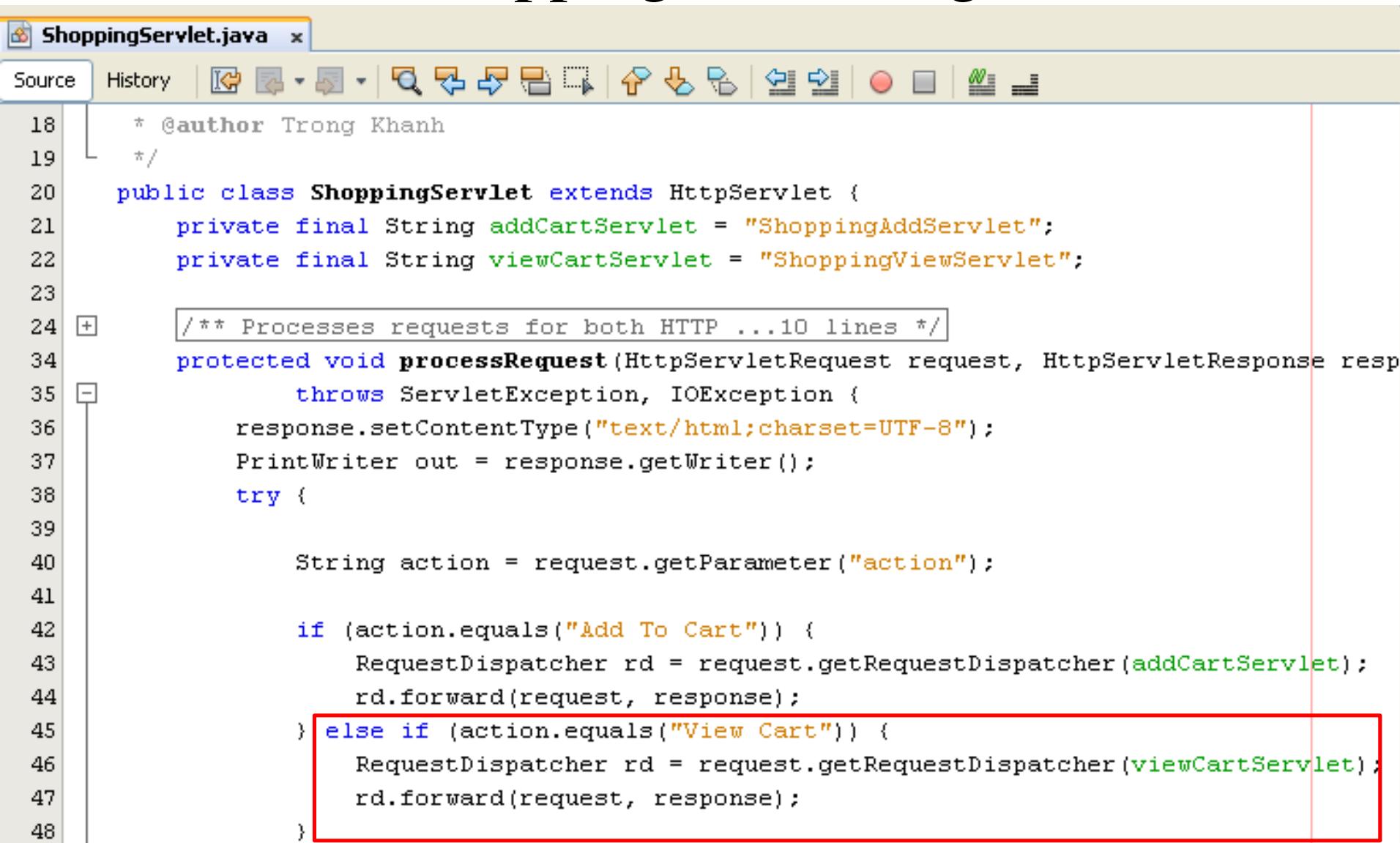
Appendix

Interactive Server Model



Appendix

Shopping Cart using Cookies

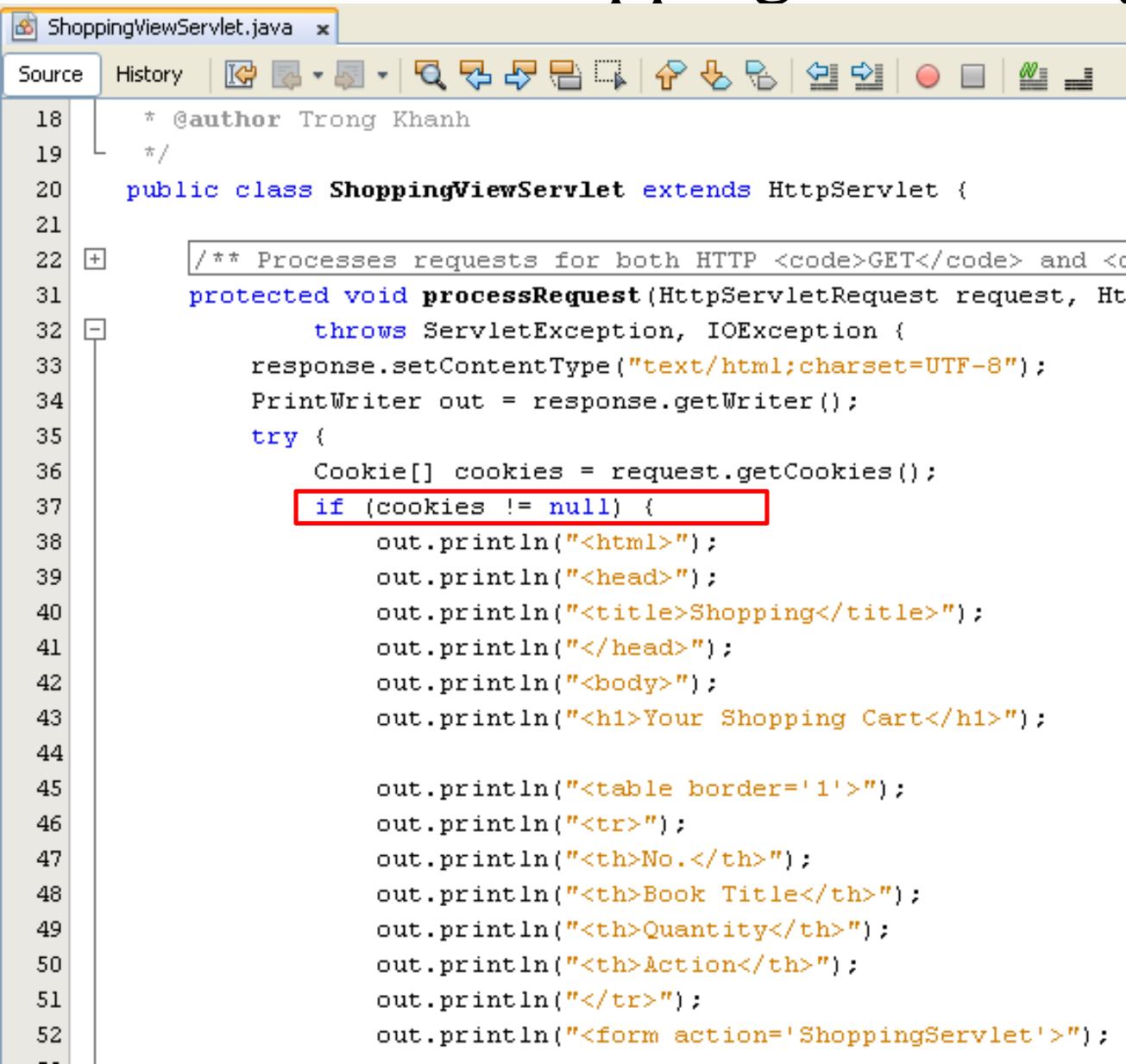


The screenshot shows a Java code editor with the file `ShoppingServlet.java` open. The code implements a `HttpServlet` to handle shopping cart requests. It defines two private final strings for the `ShoppingAddServlet` and `ShoppingViewServlet`. The `processRequest` method processes both HTTP requests. It retrieves the `action` parameter from the request. If the action is "Add To Cart", it uses a `RequestDispatcher` to forward the request to the `addCartServlet`. If the action is "View Cart", it uses a `RequestDispatcher` to forward the request to the `viewCartServlet`. The code is annotated with comments and imports.

```
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 response
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. A red box highlights the `if (cookies != null)` condition at line 37.

```
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 runtime 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'>");
```

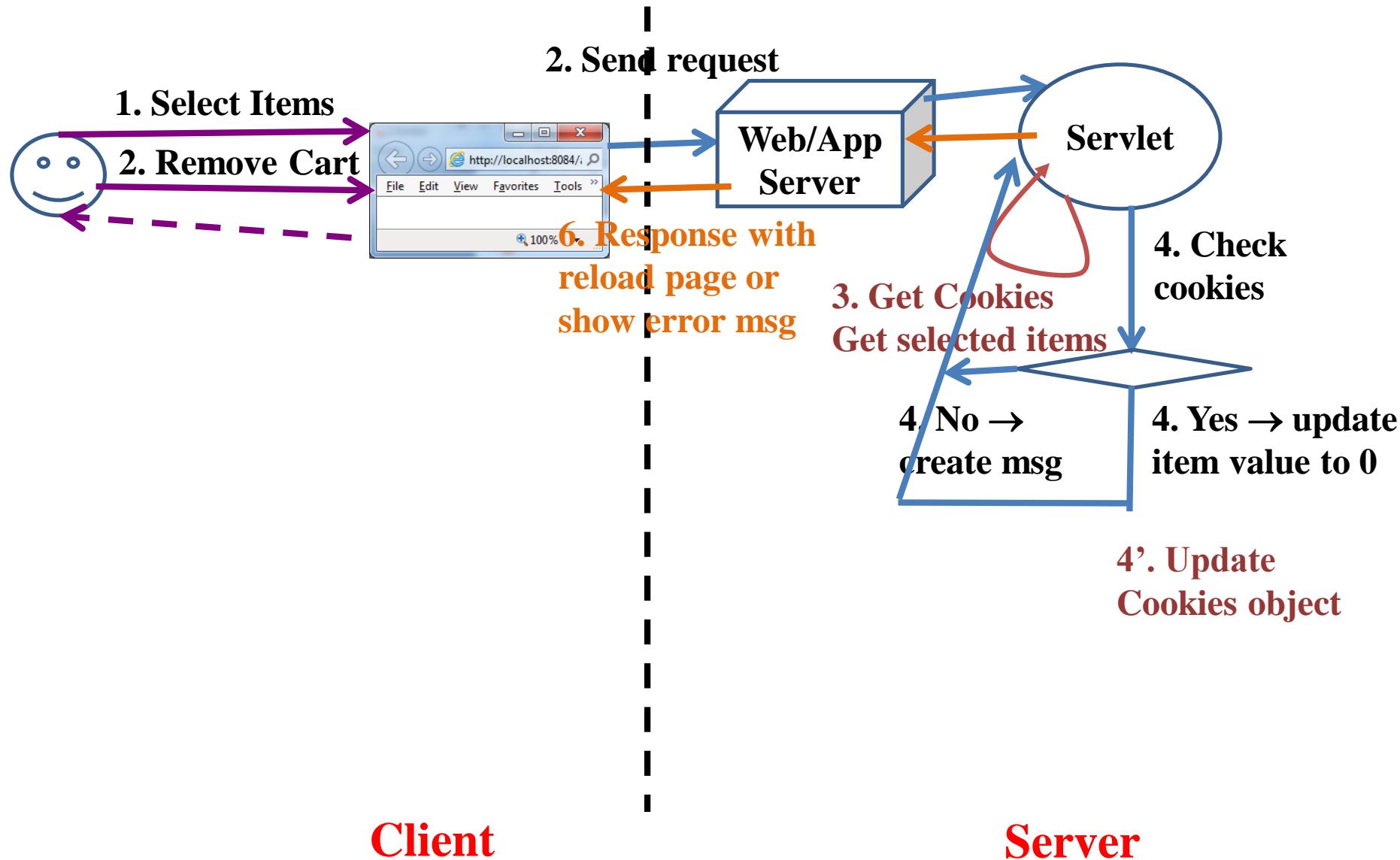
Appendix

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 }
```

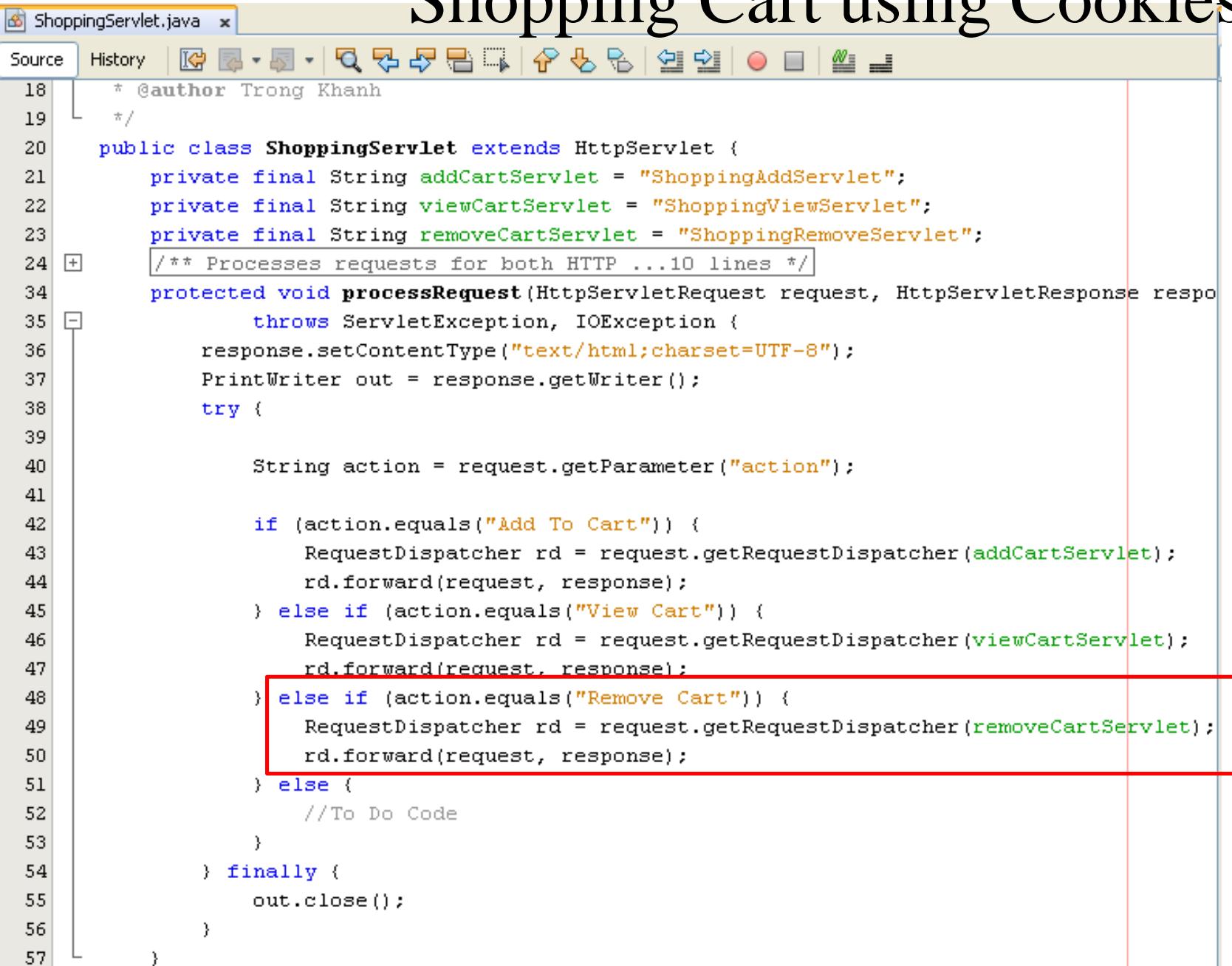
Appendix

Interactive Server Model



Appendix

Shopping Cart using Cookies

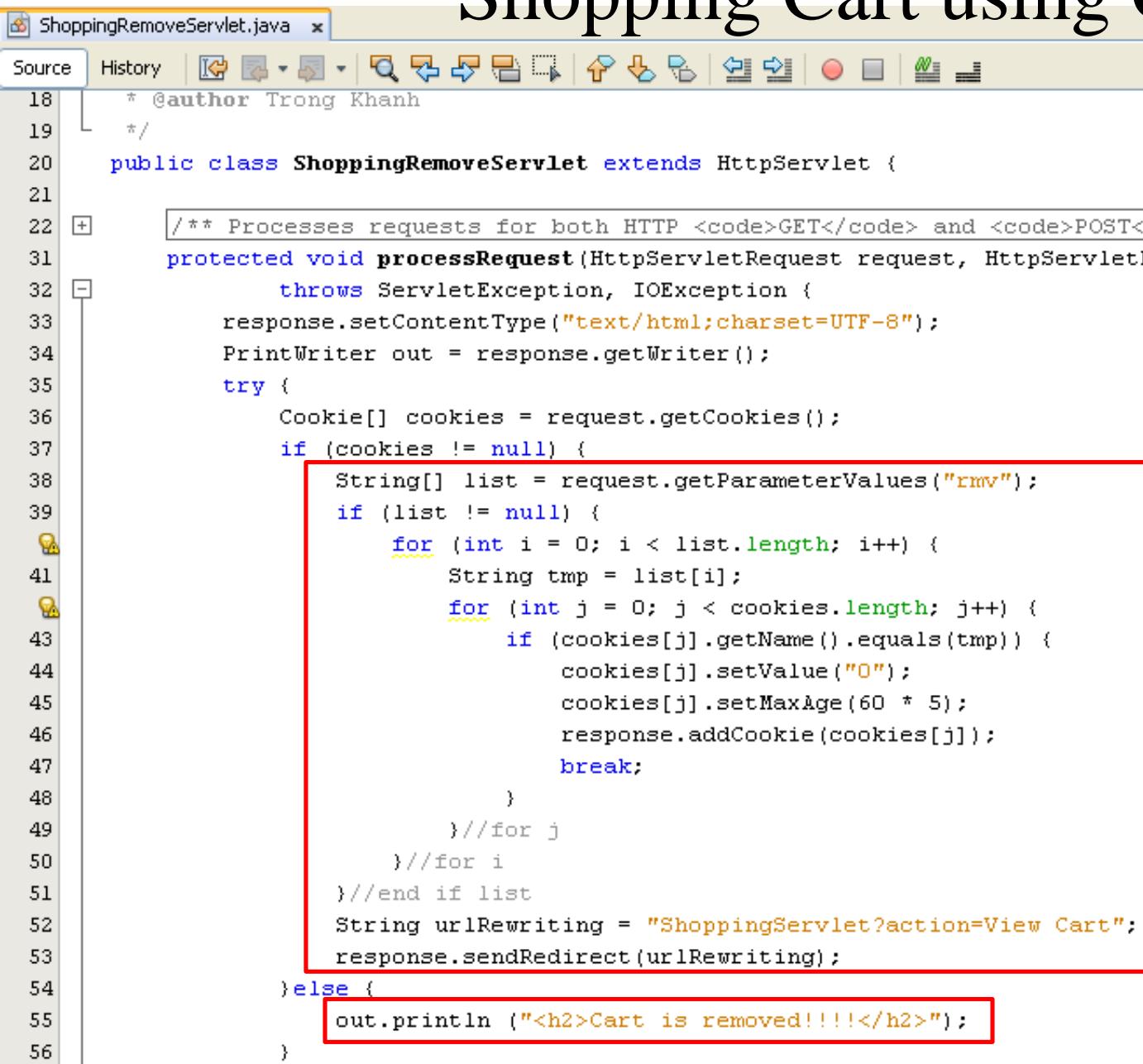


The screenshot shows a Java code editor with the file `ShoppingServlet.java` open. The code implements a `HttpServlet` that handles requests for adding items to a cart, viewing the cart, and removing items from the cart. The code uses `RequestDispatcher` to forward requests to other servlets: `ShoppingAddServlet` for adding, `ShoppingViewServlet` for viewing, and `ShoppingRemoveServlet` for removing. 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



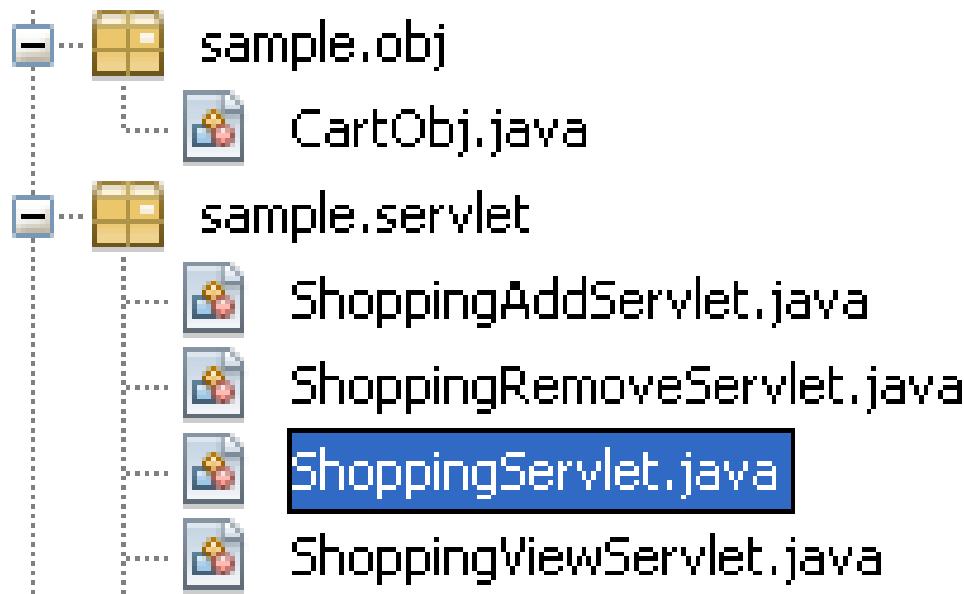
The screenshot shows a Java code editor with the file `ShoppingRemoveServlet.java` open. The code implements a `HttpServlet` to handle requests for removing items from a shopping cart using cookies.

```
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 }
```

A red box highlights the logic for removing items from the cookie-based cart. Another red box highlights the success message printed to the response.

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>

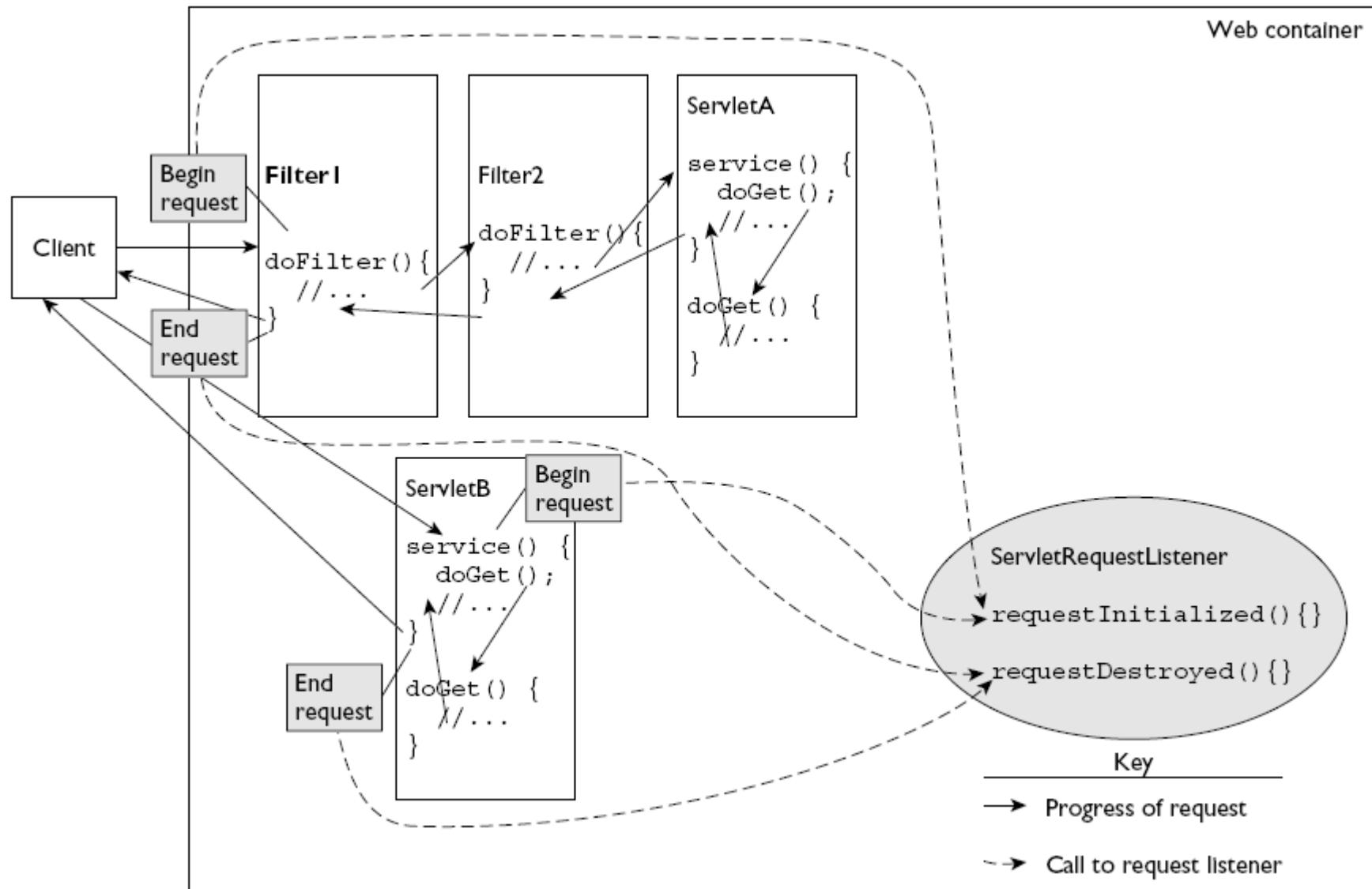
Appendix

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



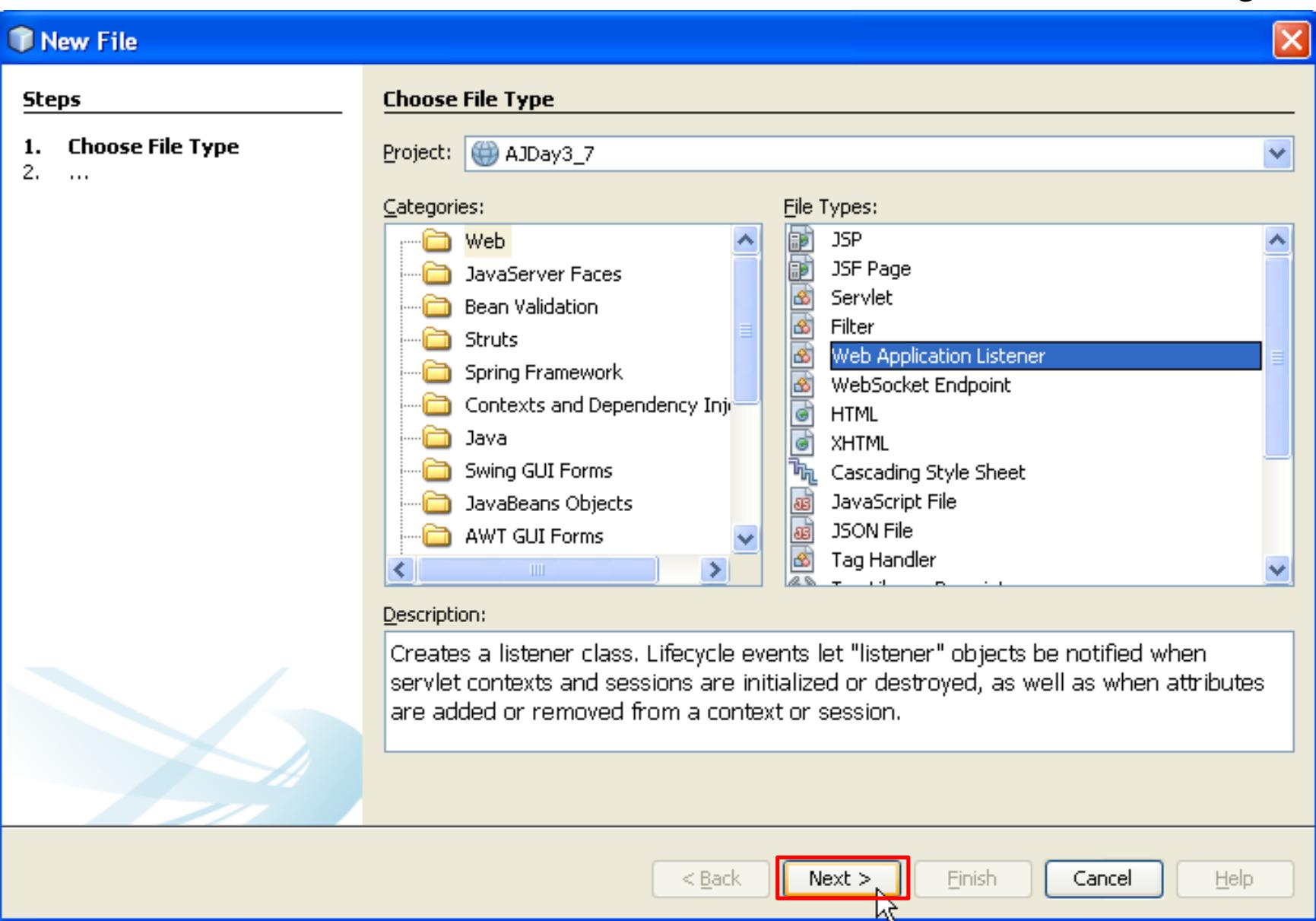
Appendix

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

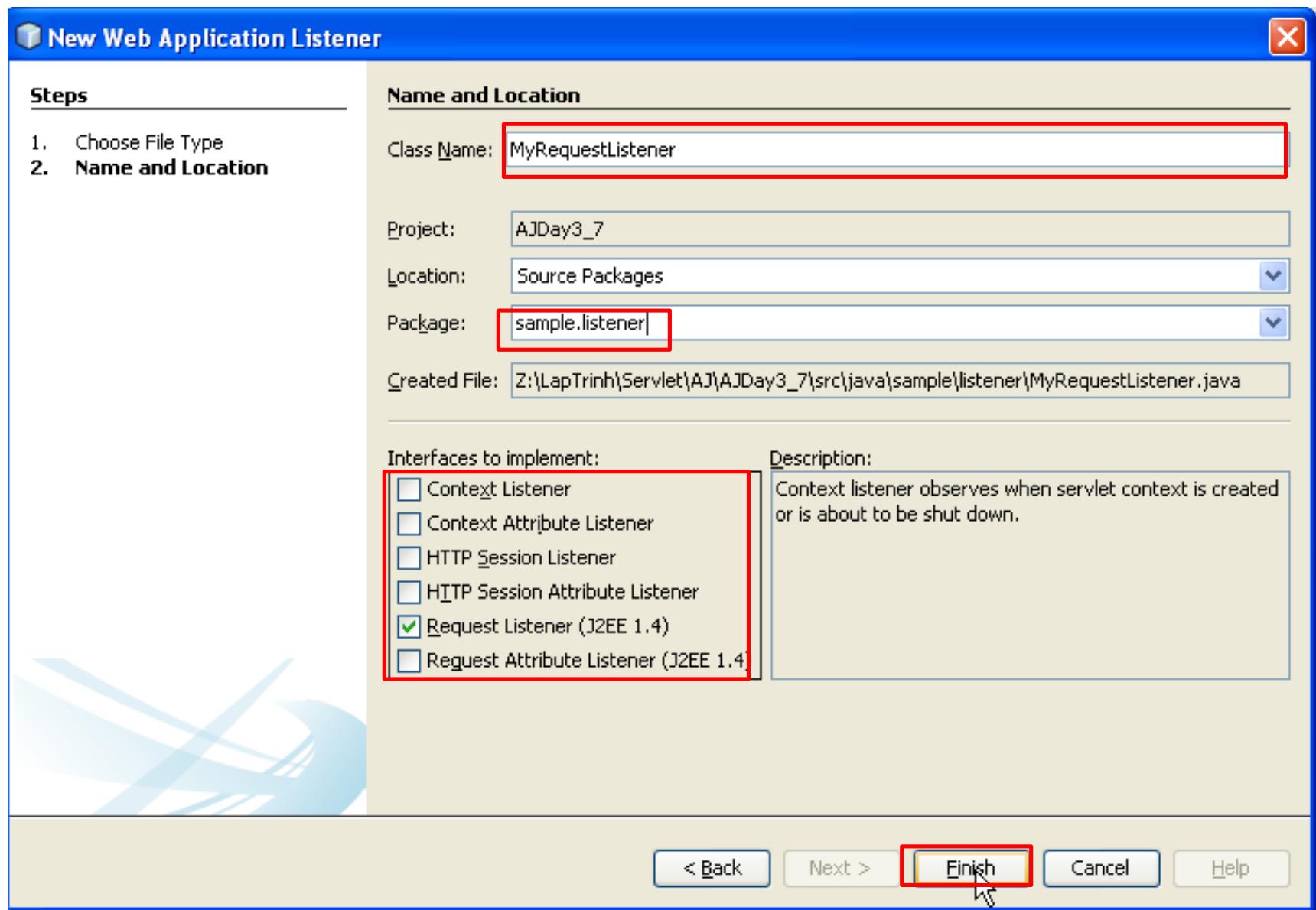
Appendix

How to Add Listener to Web Project



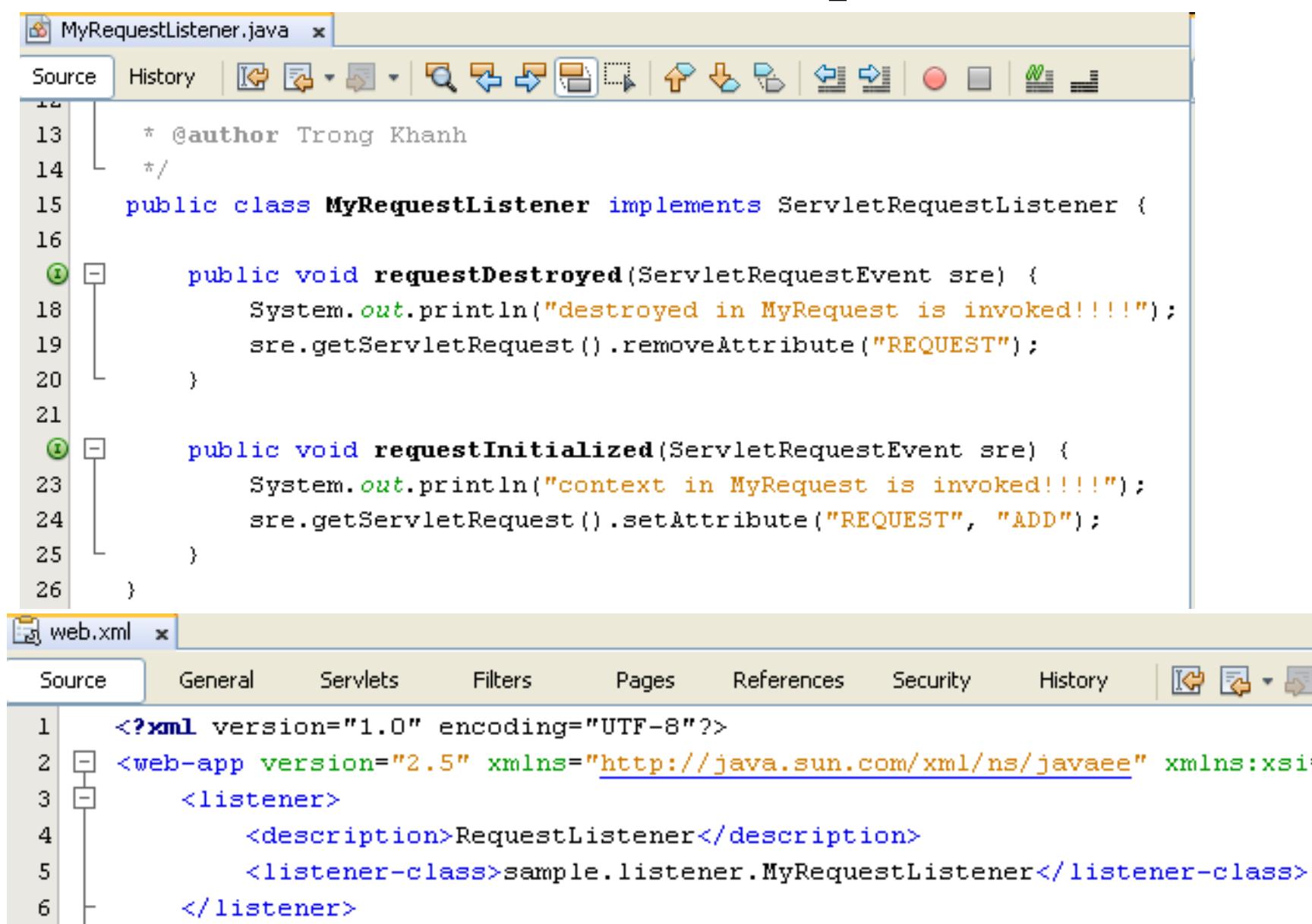
Appendix

How to Add Listener to Web Project



Appendix

Example



The screenshot shows a Java code editor and a configuration file in an IDE.

Java Code (MyRequestListener.java):

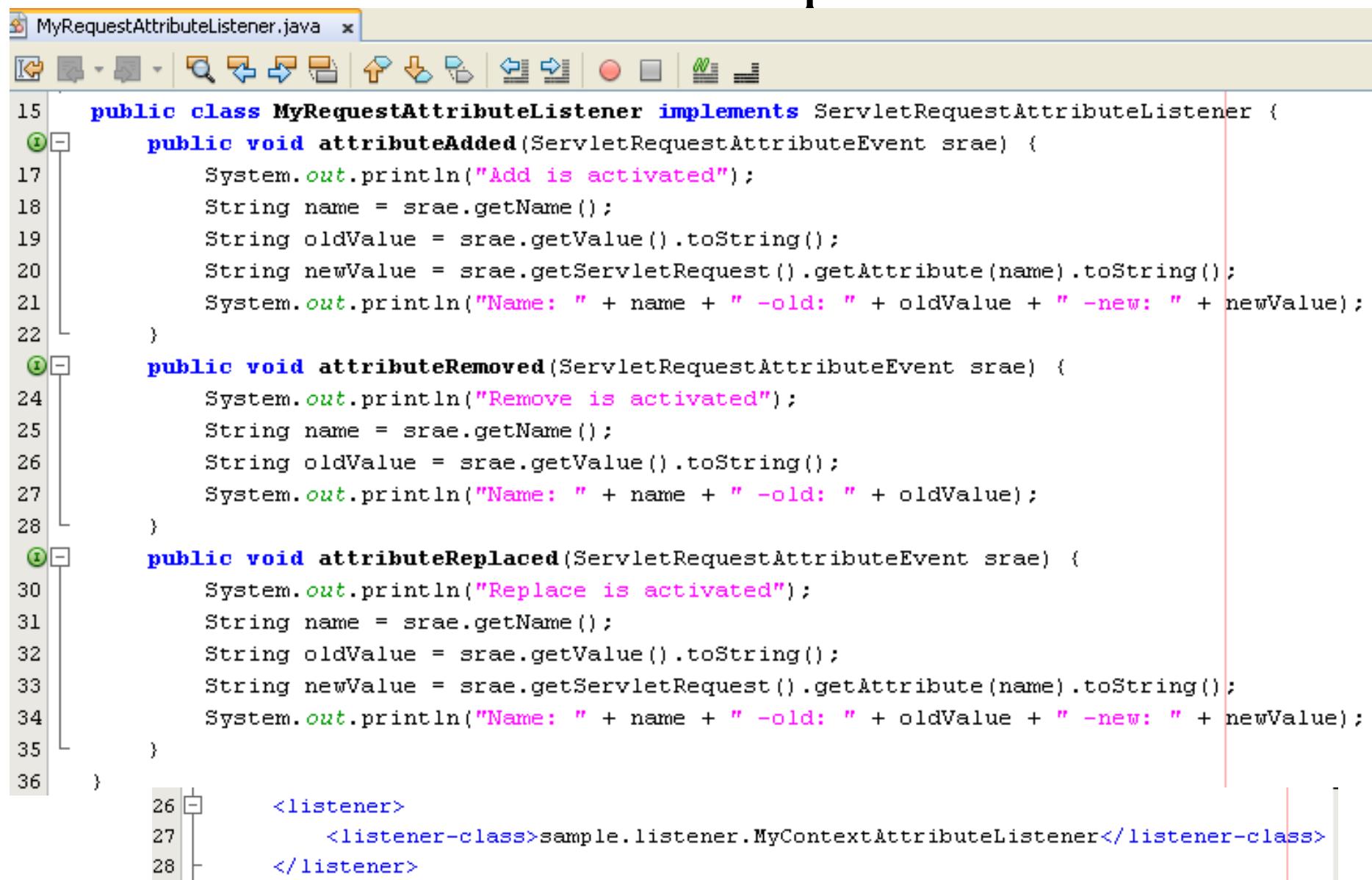
```
13 * @author Trong Khanh
14 */
15 public class MyRequestListener implements ServletRequestListener {
16
17     public void requestDestroyed(ServletRequestEvent sre) {
18         System.out.println("destroyed in MyRequest is invoked!!!!");
19         sre.getServletRequest().removeAttribute("REQUEST");
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 }
```

Configuration File (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="http://www.w3.org/2001/XMLSchema-instance">
3     <listener>
4         <description>RequestListener</description>
5         <listener-class>sample.listener.MyRequestListener</listener-class>
6     </listener>
```

Appendix

Example

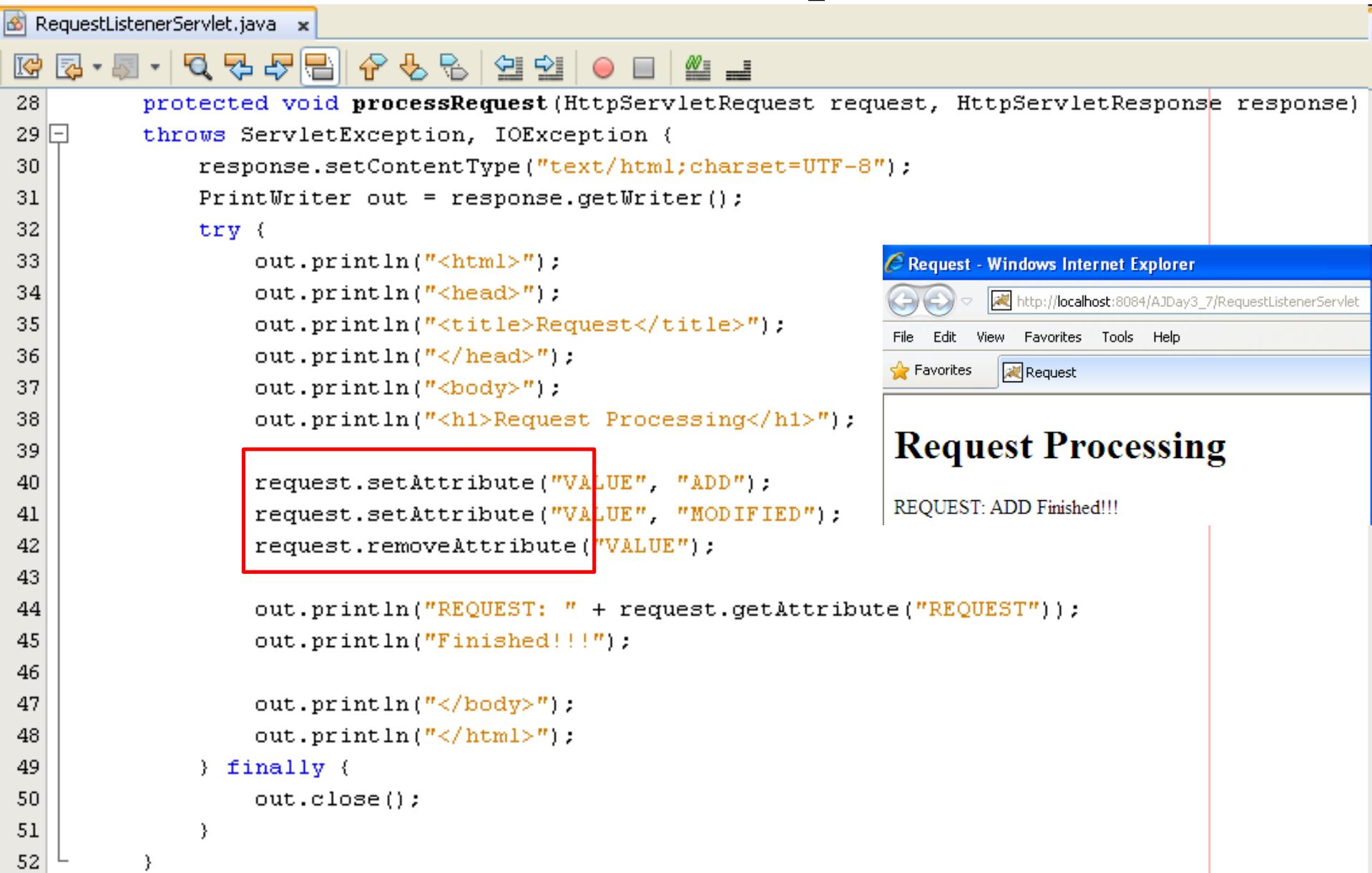


The screenshot shows a Java code editor with the file `MyRequestAttributeListener.java` open. The code implements the `ServletRequestAttributeListener` interface, containing three methods: `attributeAdded`, `attributeRemoved`, and `attributeReplaced`. Each method prints the name of the attribute, its old value, and its new value to the console. Below the code, there is a configuration snippet for a web.xml file that registers the listener.

```
public class MyRequestAttributeListener implements ServletRequestAttributeListener {
    public void attributeAdded(ServletRequestAttributeEvent srae) {
        System.out.println("Add is activated");
        String name = srae.getName();
        String oldValue = srae.getValue().toString();
        String newValue = srae.getServletRequest().getAttribute(name).toString();
        System.out.println("Name: " + name + " -old: " + oldValue + " -new: " + newValue);
    }
    public void attributeRemoved(ServletRequestAttributeEvent srae) {
        System.out.println("Remove is activated");
        String name = srae.getName();
        String oldValue = srae.getValue().toString();
        System.out.println("Name: " + name + " -old: " + oldValue);
    }
    public void attributeReplaced(ServletRequestAttributeEvent srae) {
        System.out.println("Replace is activated");
        String name = srae.getName();
        String oldValue = srae.getValue().toString();
        String newValue = srae.getServletRequest().getAttribute(name).toString();
        System.out.println("Name: " + name + " -old: " + oldValue + " -new: " + newValue);
    }
}
<listener>
    <listener-class>sample.listener.MyContextAttributeListener</listener-class>
</listener>
```

Appendix

Example



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

Code Editor (RequestListenerServlet.java):

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

        request.setAttribute("VALUE", "ADD");
        request.setAttribute("VALUE", "MODIFIED");
        request.removeAttribute("VALUE");

        out.println("REQUEST: " + request.getAttribute("REQUEST"));
        out.println("Finished!!!");

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

Browser Window (Request - Windows Internet Explorer):

http://localhost:8084/AJDay3_7/RequestListenerServlet

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.server.bb5
Add is activated
Name: netbeans.monitor.filter -old: MonitorFilter(ApplicationFilterConfig[name=MonitorFilter, filterClass=org.netbeans.modules.web.monitor.MonitorFilter])
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; Windows NT 6.1; .NET CLR 2.0.50727)
    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]

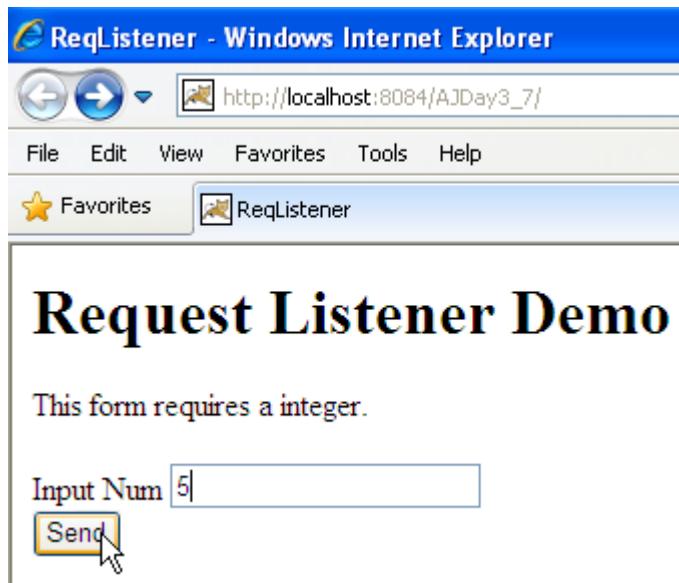
destroyed in MyRequest is invoked!!!!

Remove is activated

Name: REQUEST -old: ADD

Appendix

Practices – Example



Reqlistener - Windows Internet Explorer
http://localhost:8084/AJDay3_7/

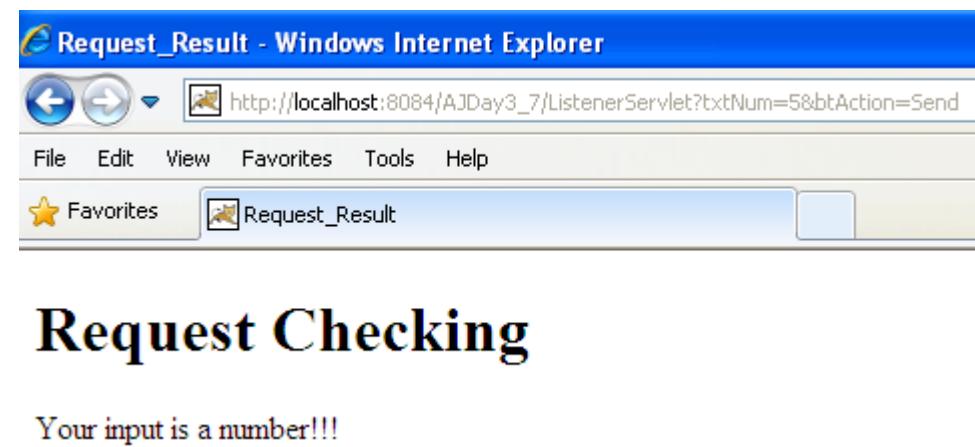
File Edit View Favorites Tools Help

Favorites ReqListener

Request Listener Demo

This form requires a integer.

Input Num



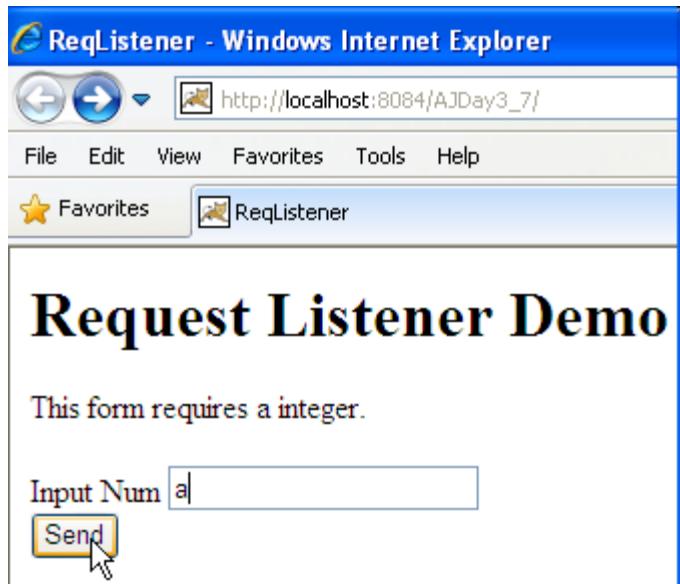
Request_Result - Windows Internet Explorer
http://localhost:8084/AJDay3_7/ListenerServlet?txtNum=5&btAction=Send

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/

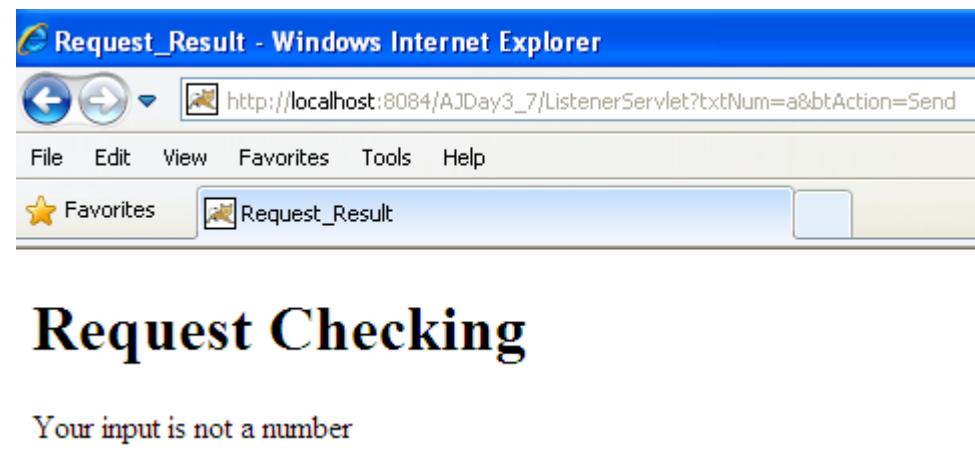
File Edit View Favorites Tools Help

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

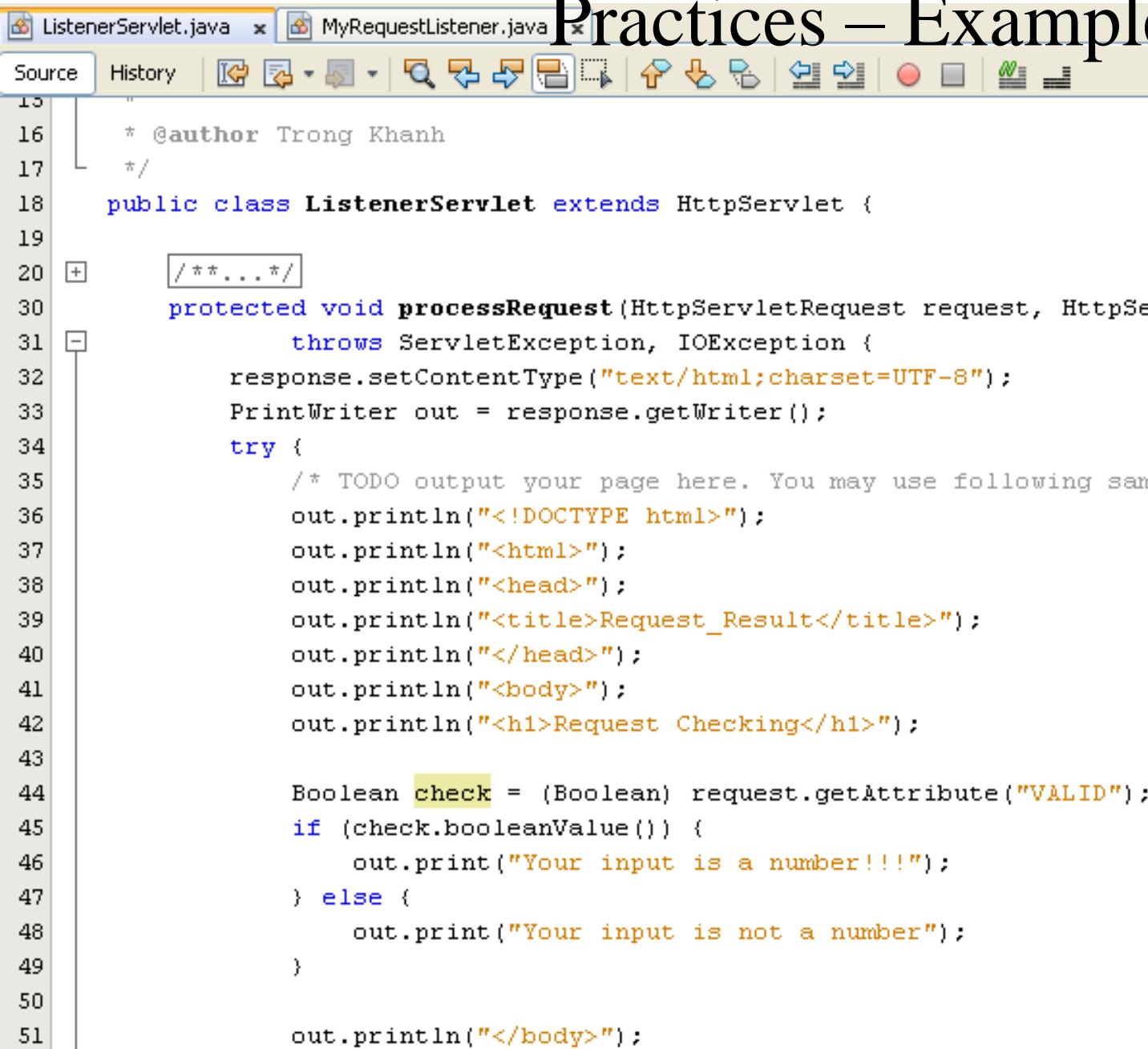


The screenshot shows a Java Integrated Development Environment (IDE) interface. At the top, there are three tabs: "validUsingReqListener.html", "ListenerServlet.java", and "MyRequestListener.java". Below the tabs is a toolbar with various icons for file operations like Open, Save, and Print. The main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

Appendix

Practices – Example



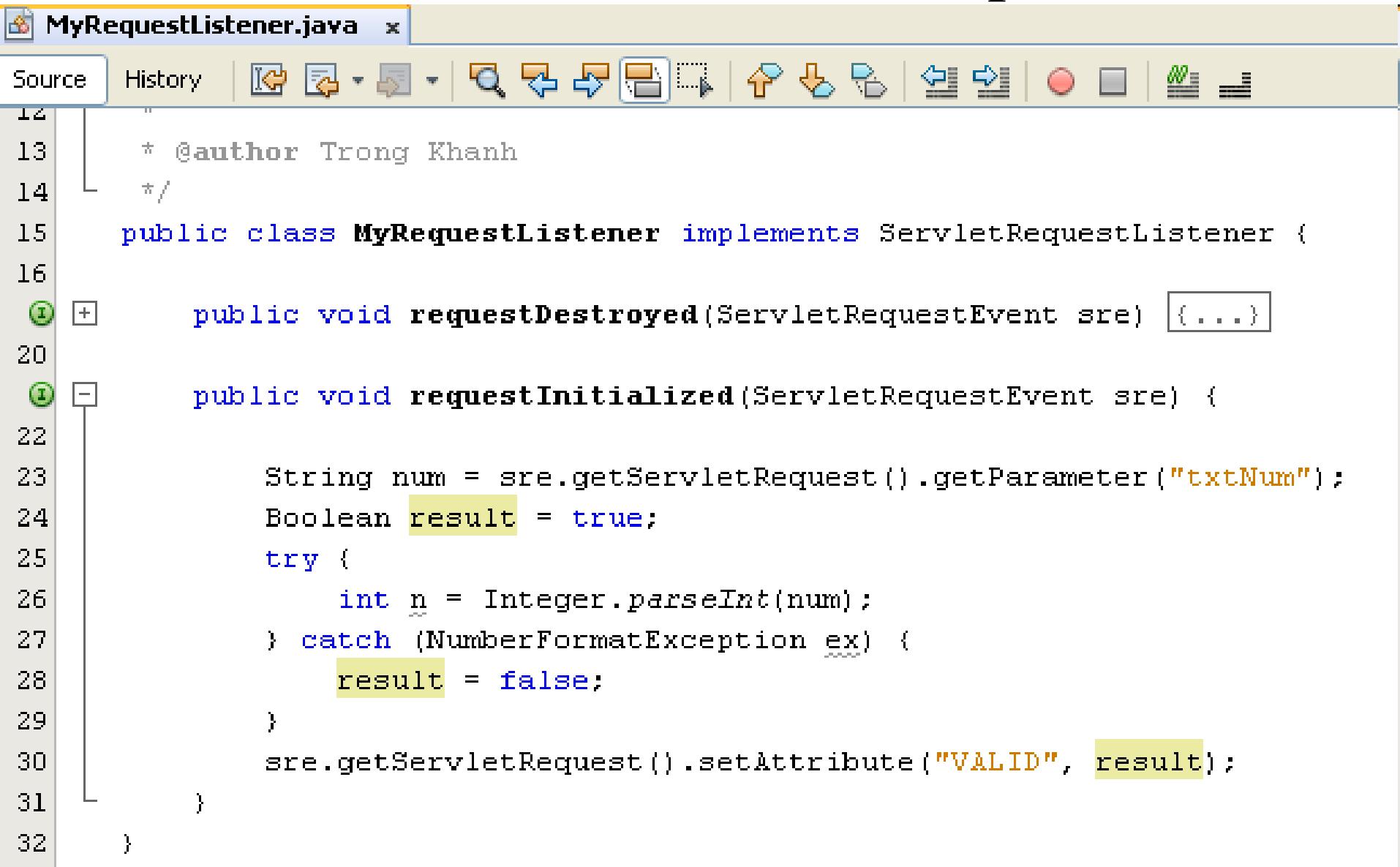
The screenshot shows a Java code editor with two tabs: ListenerServlet.java and MyRequestListener.java. The ListenerServlet.java tab is active, displaying the following code:

```
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

MyRequestListener.java



```
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 }
```

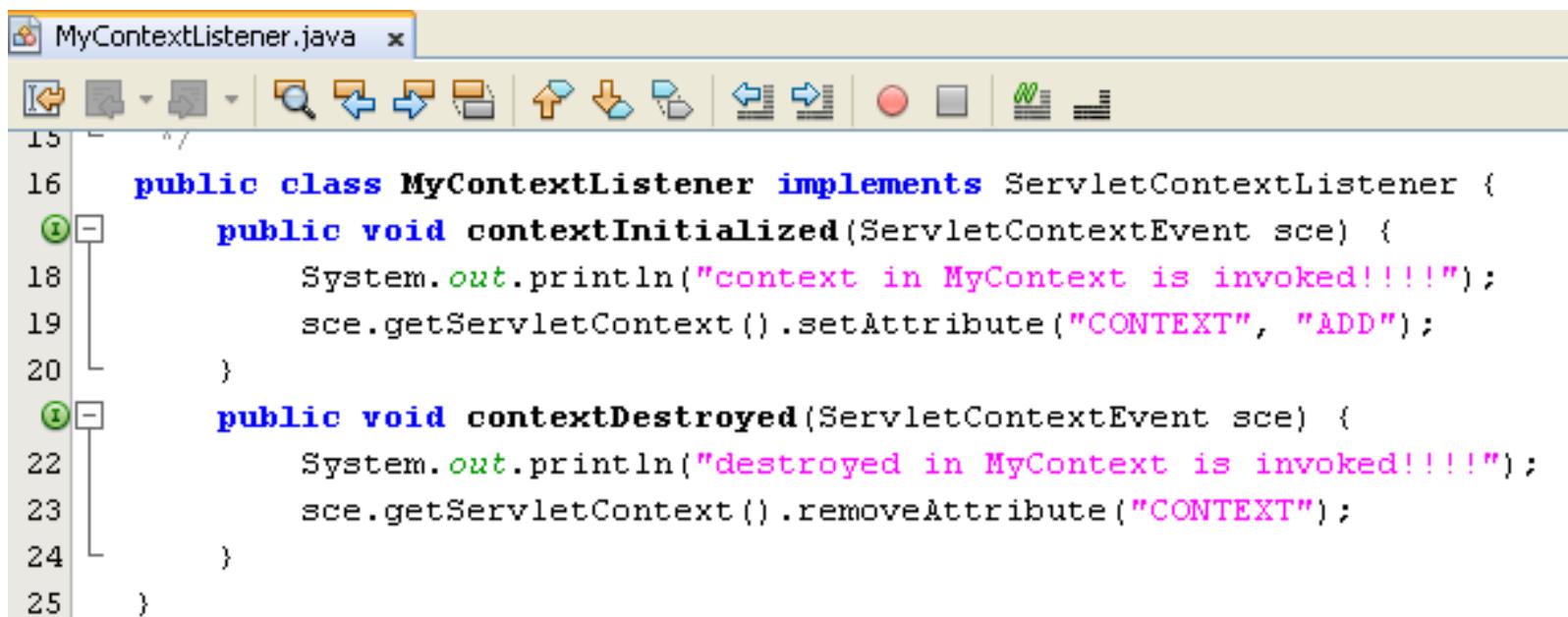
Appendix

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**

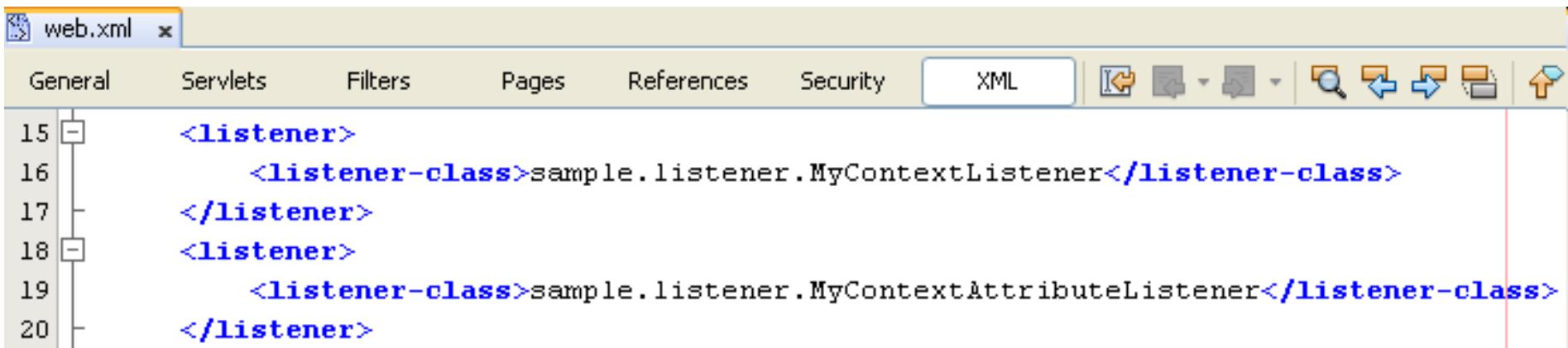
Appendix

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 }
```

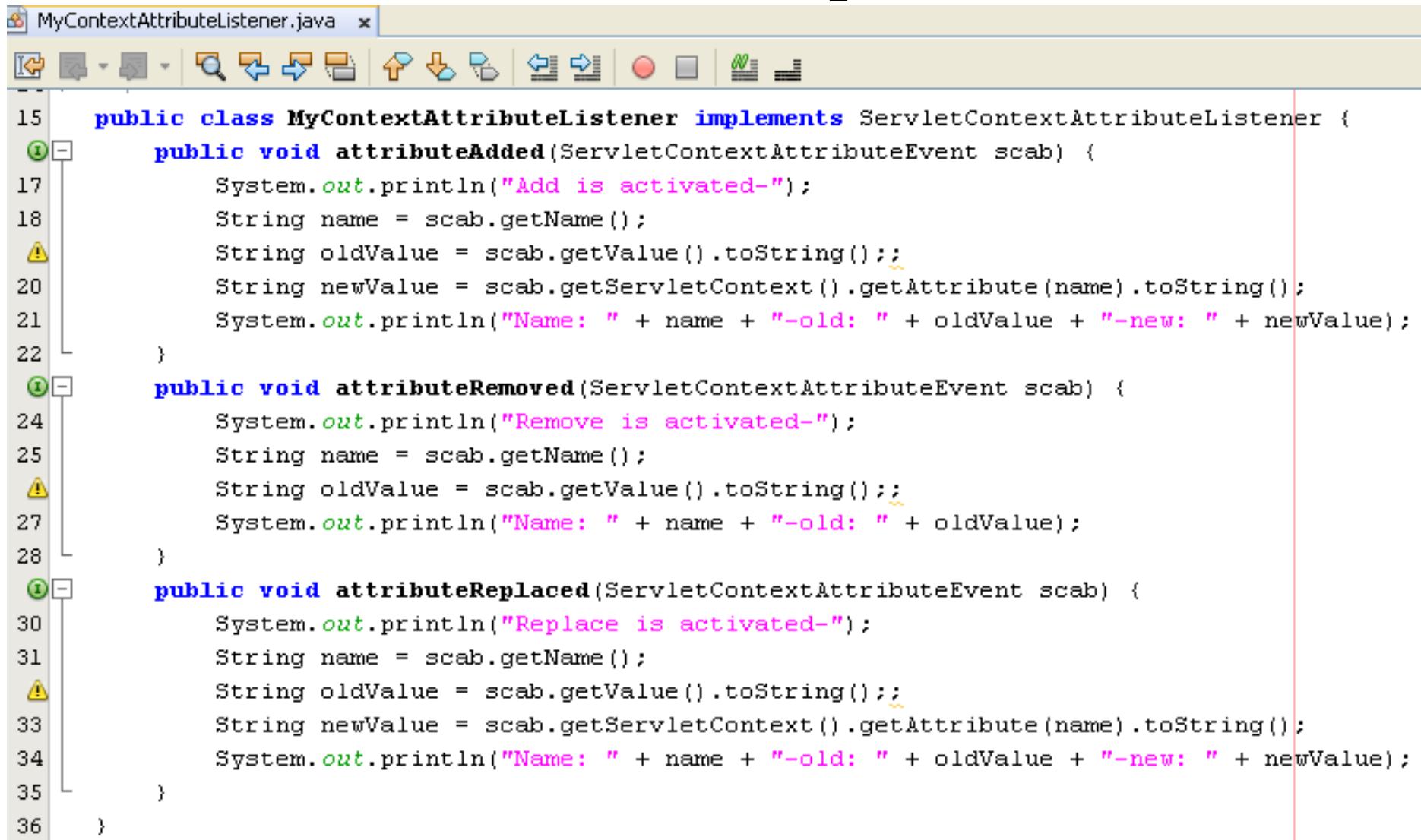


web.xml

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

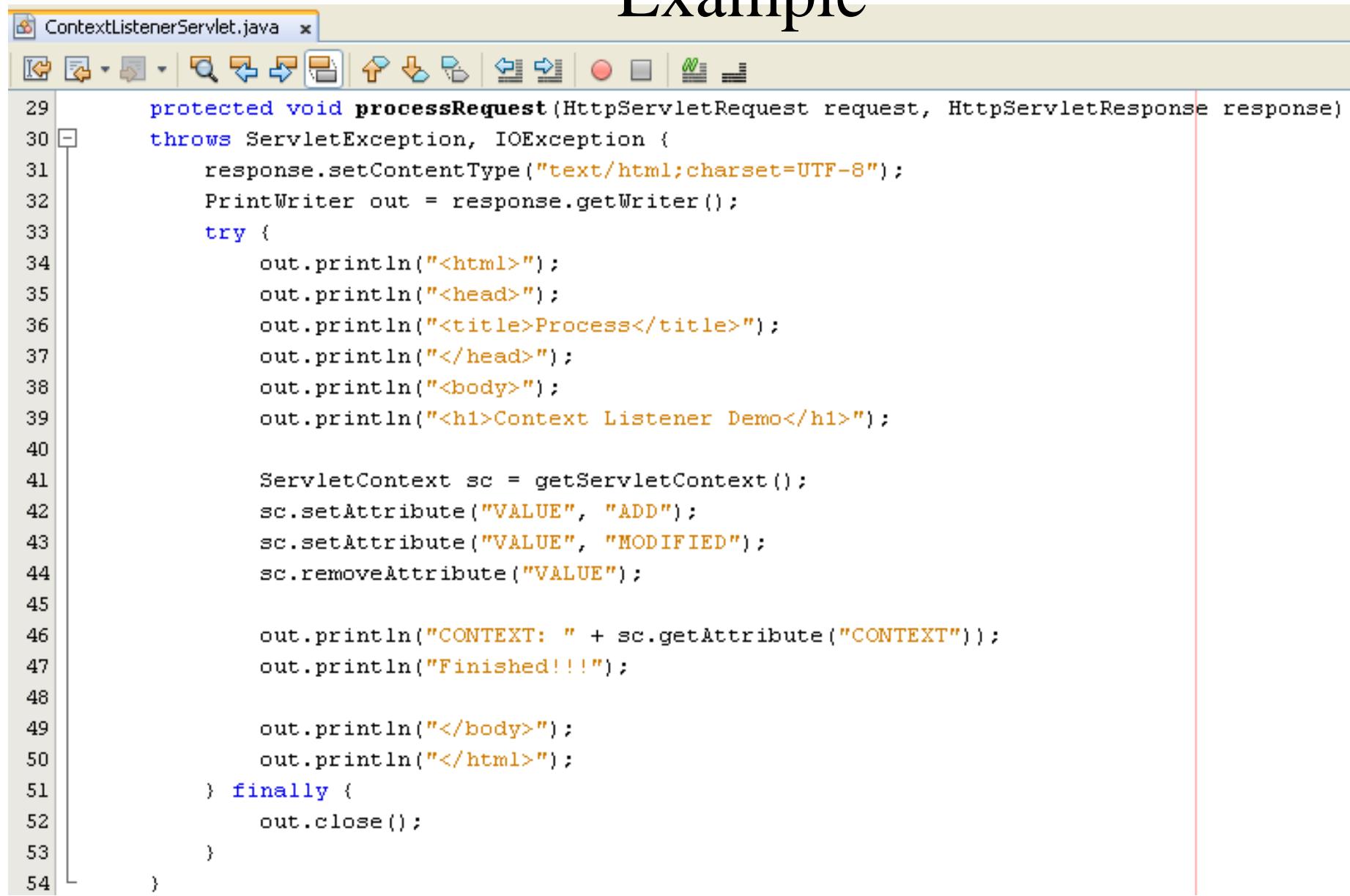


The screenshot shows a Java code editor with the file `MyContextAttributeListener.java` open. The code implements the `ServletContextAttributeListener` interface, handling three events: `attributeAdded`, `attributeRemoved`, and `attributeReplaced`. It prints the name of the attribute, its old value,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

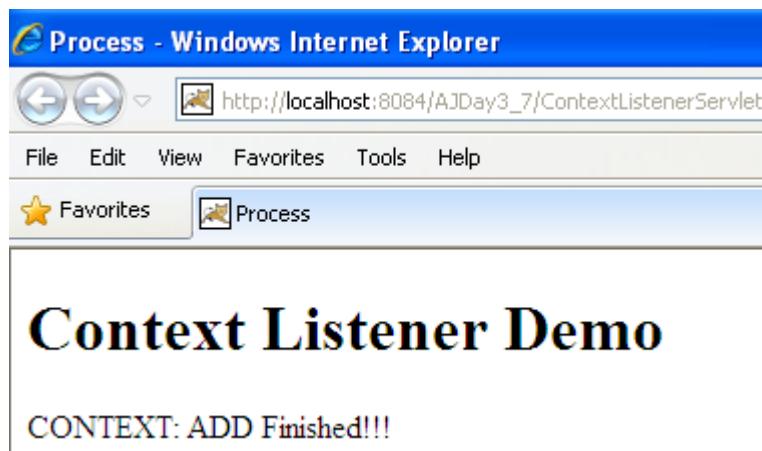
        ServletContext sc = getServletContext();
        sc.setAttribute("VALUE", "ADD");
        sc.setAttribute("VALUE", "MODIFIED");
        sc.removeAttribute("VALUE");

        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...
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

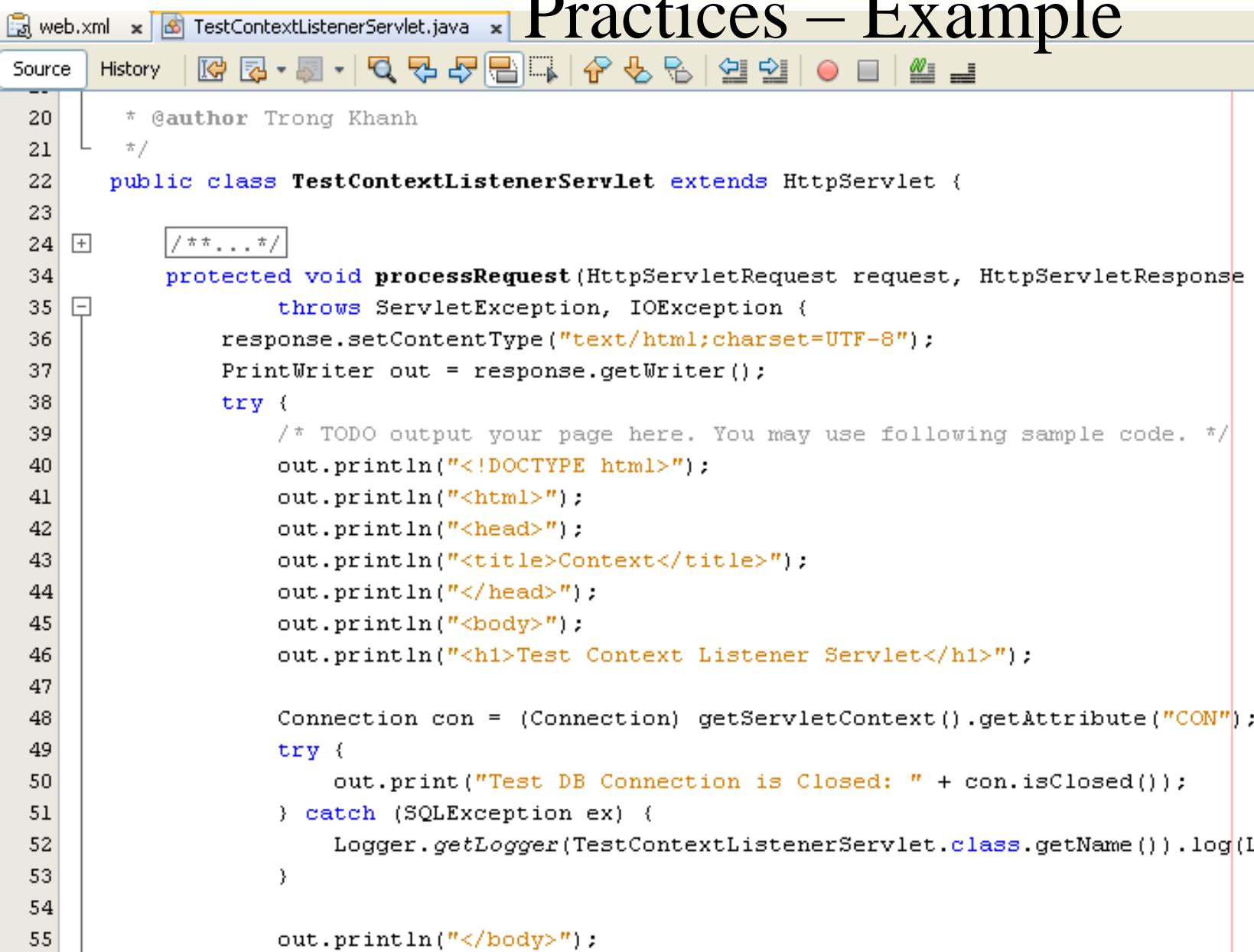
Practices – Example



```
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) { ... }
```

Appendix

Practices – Example



The screenshot shows a Java code editor with the following details:

- File Tabs:** web.xml (closed), TestContextListenerServlet.java (open).
- Toolbar:** Includes tabs for Source, History, and various file operations like Open, Save, Find, and Copy.
- Code Content:** The code is a Java servlet named TestContextListenerServlet. It prints an HTML page and retrieves a database connection from the servlet context.

```
20 * @author Trong Khanh
21 */
22 public class TestContextListenerServlet extends HttpServlet {
23
24     /**
25      * ...
26     */
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         try {
32             /* TODO output your page here. You may use following sample code. */
33             out.println("<!DOCTYPE html>");
34             out.println("<html>");
35             out.println("<head>");
36             out.println("<title>Context</title>");
37             out.println("</head>");
38             out.println("<body>");
39             out.println("<h1>Test Context Listener Servlet</h1>");
40
41             Connection con = (Connection) getServletContext().getAttribute("CON");
42             try {
43                 out.print("Test DB Connection is Closed: " + con.isClosed());
44             } catch (SQLException ex) {
45                 Logger.getLogger(TestContextListenerServlet.class.getName()).log(L
46             }
47
48             out.println("</body>");
49         }
50     }
51 }
```

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

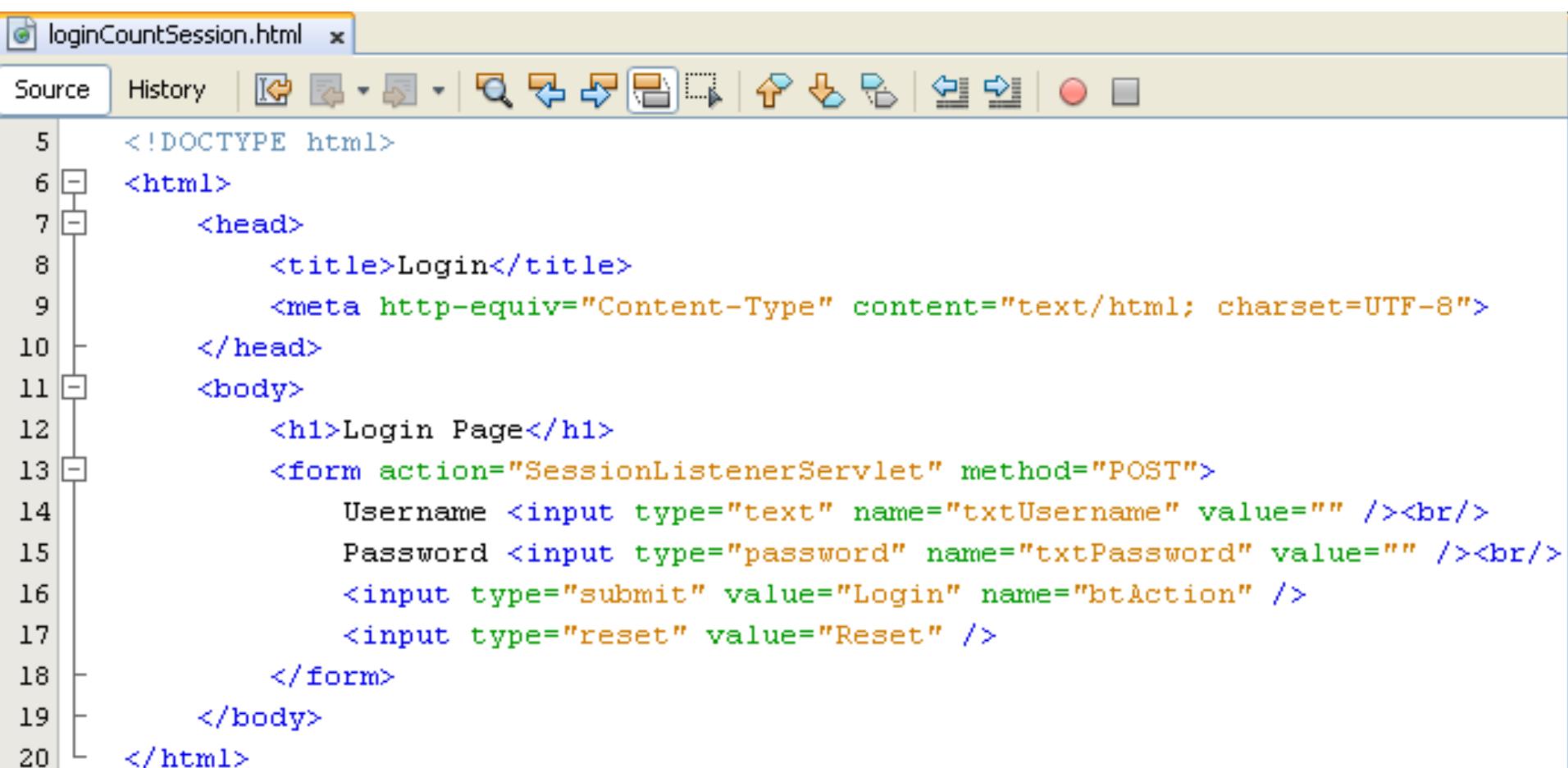
Session Listener Demo

Welcome, test

This website is accessed: 2 times

Appendix

Practices – Example



The screenshot shows a web browser window with the title "loginCountSession.html". The tab bar also includes "Source" and "History". The toolbar contains various icons for file operations like Open, Save, Print, and Find.

```
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

Practices – Example

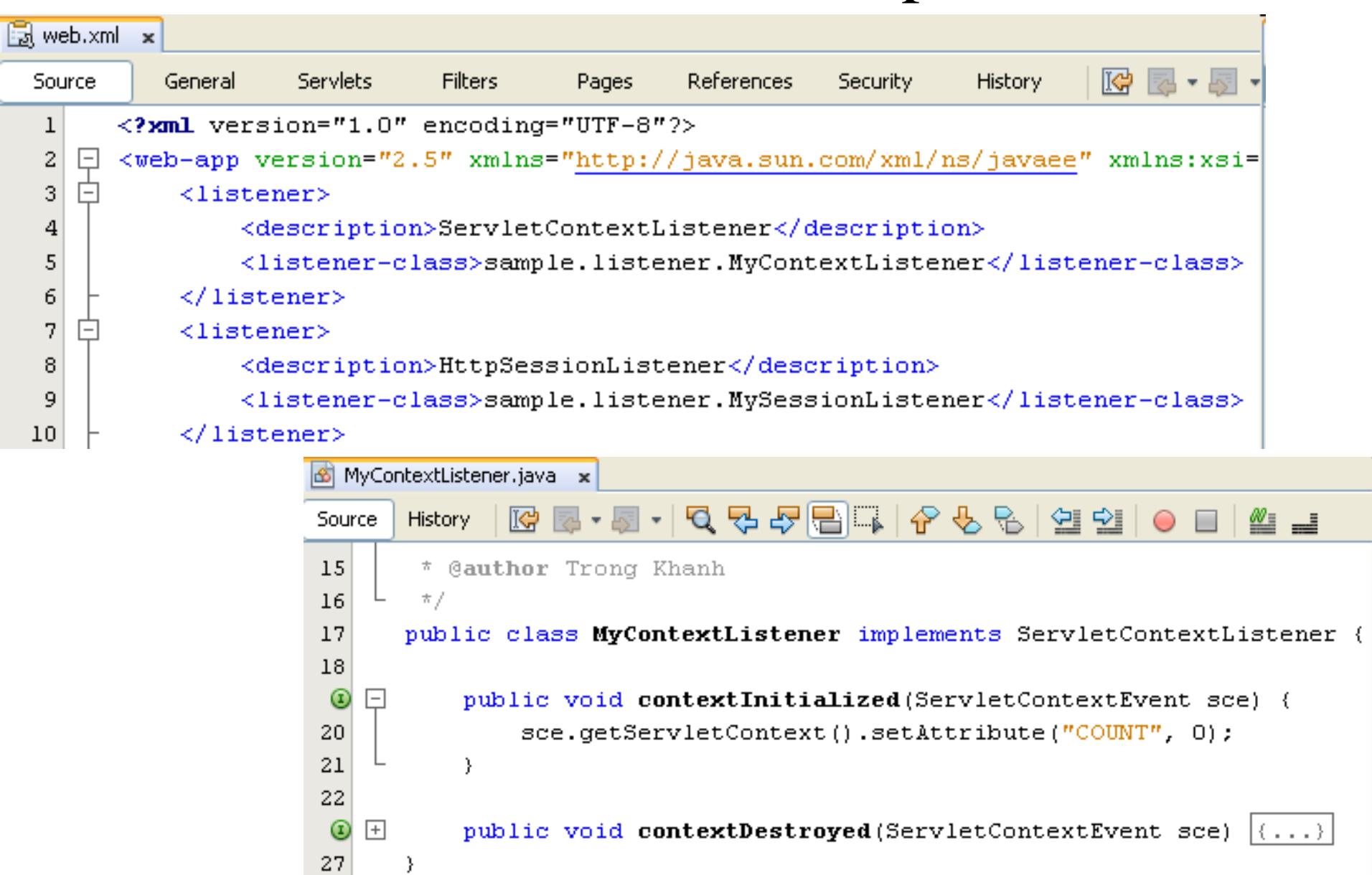
The screenshot shows a Java IDE interface with the following details:

- Title Bar:** SessionListenerServlet.java
- Toolbar:** Includes icons for Source, History, and various file operations like Open, Save, Find, and Copy.
- Code Editor:** Displays the Java code for SessionListenerServlet. The code handles session listeners and prints a welcome message or error based on user input.

```
17     * @author Trong Khanh
18     */
19     public class SessionListenerServlet extends HttpServlet {
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             out.println("<!DOCTYPE html>");
30             out.println("<html>");
31             out.println("<head>");
32             out.println("<title>Result</title>");
33             out.println("</head>");
34             out.println("<body>");
35             out.println("<h1>Session Listener Demo</h1>");
36             String user = request.getParameter("txtUsername");
37             String pass = request.getParameter("txtPassword");
38             if (user.equals(pass)) {
39                 HttpSession session = request.getSession();
40                 out.println("<h1>Welcome, " + user + "</h1>");
41                 Integer count = (Integer) getServletContext().getAttribute("COUNT");
42                 out.println("This website is accessed: " + count + " times");
43             } else {
44                 out.println("<h1>Invalid username or password</h1>");
45             }
46             out.println("</body>");
47         } catch (Exception e) {
48             e.printStackTrace();
49         }
50     }
51
52     // Other methods and annotations omitted for brevity
53 }
```

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="http://www.w3.org/2001/XMLSchema-instance">
    <listener>
        <description>ServletContextListener</description>
        <listener-class>sample.listener.MyContextListener</listener-class>
    </listener>
    <listener>
        <description>HttpSessionListener</description>
        <listener-class>sample.listener.MySessionListener</listener-class>
    </listener>
</web-app>
```

- 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`.

```
* @author Trong Khanh
*/
public class MyContextListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent sce) {
        sce.getServletContext().setAttribute("COUNT", 0);
    }
    public void contextDestroyed(ServletContextEvent sce) {
    }
}
```

Appendix

Practices – Example

The screenshot shows a Java code editor with the file `MySessionListener.java` open. The code implements the `HttpSessionListener` interface. It contains two methods: `sessionCreated` and `sessionDestroyed`. In the `sessionCreated` method, it retrieves the current session, gets the `ServletContext`, and increments a counter attribute named "COUNT". It then prints the value to `System.out` and updates the session attribute. The `sessionDestroyed` method is partially visible at the bottom.

```
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

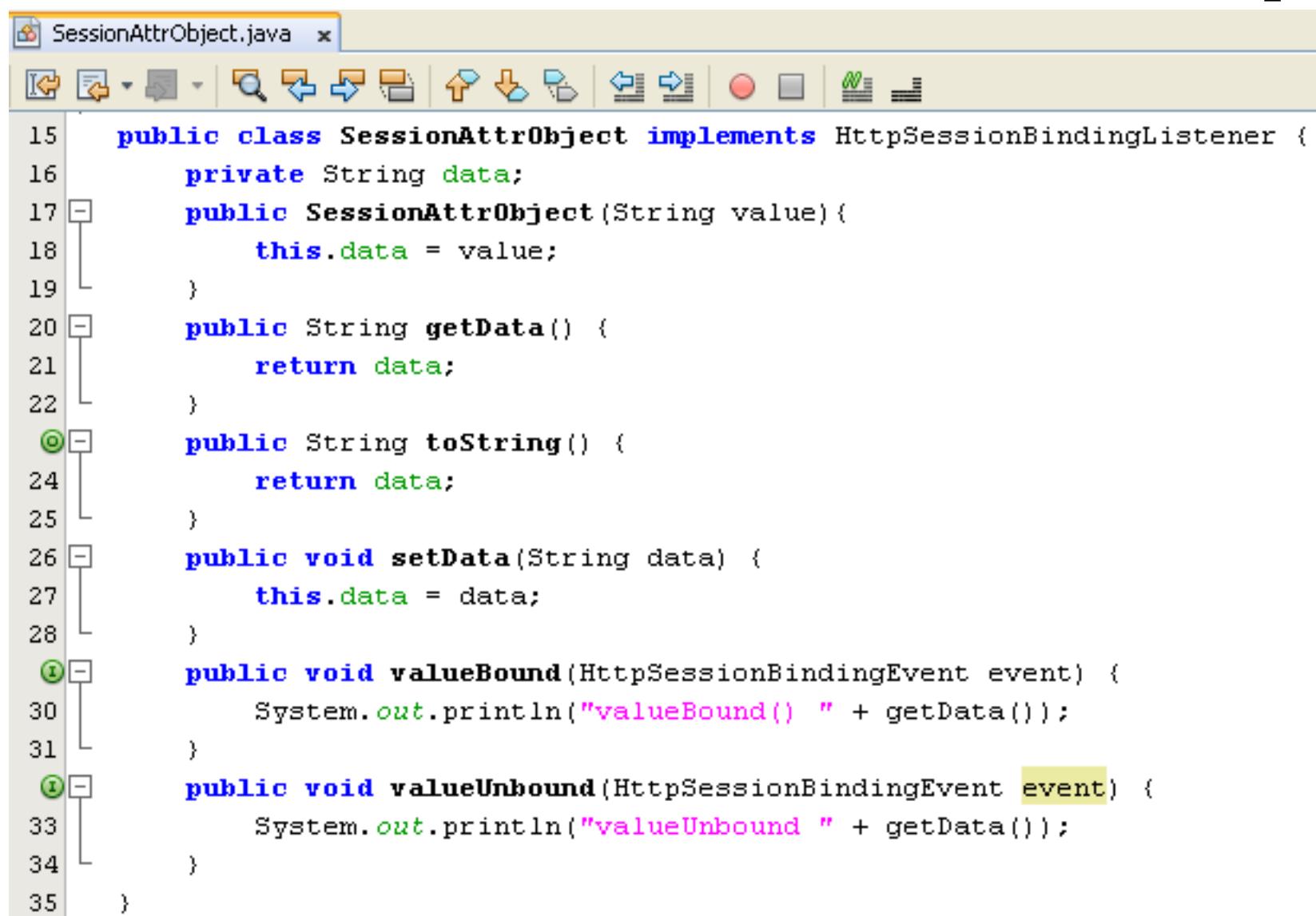
Session Listeners Not Declared in DD

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

Appendix

Session Listeners Not Declared in DD - Example

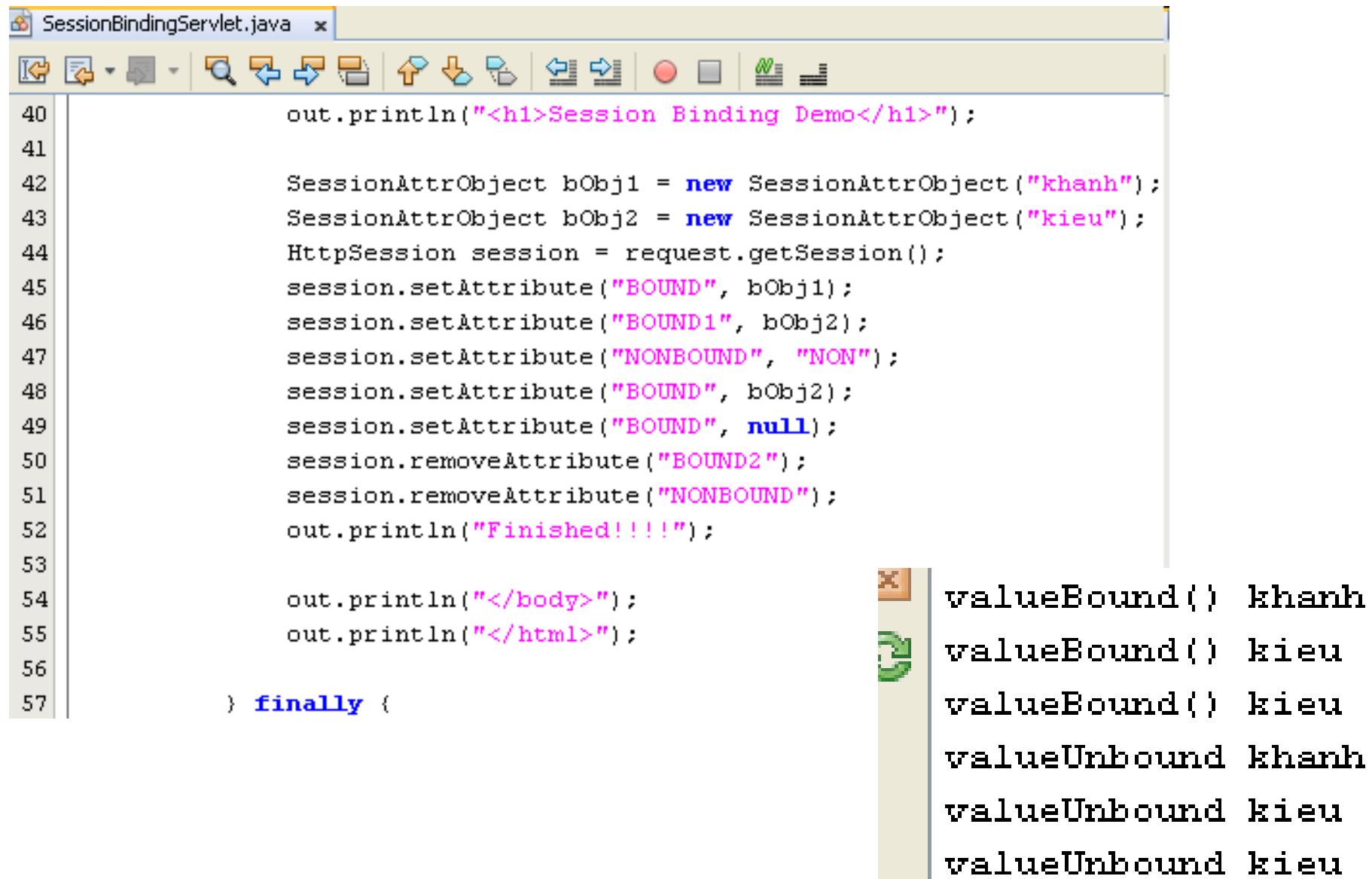


The screenshot shows a Java code editor window with the file "SessionAttrObject.java" open. The code implements the HttpSessionBindingListener interface. It contains a private attribute "data", a constructor that initializes "data" to the provided value, and methods to get and set "data". It also implements the two listener methods: valueBound and valueUnbound, both of which print the current value of "data" 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 manipulation and its effect on session listeners. The code includes logic to set various session attributes (e.g., `BOUND`, `NONBOUND`) and remove them, followed by a final print statement.

```
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 {
```

The right side of the interface displays the output of the code execution, showing the sequence of events triggered by the session attribute changes:

- valueBound() khanh
- valueBound() kieu
- valueBound() kieu
- valueUnbound khanh
- valueUnbound kieu
- valueUnbound kieu

Appendix

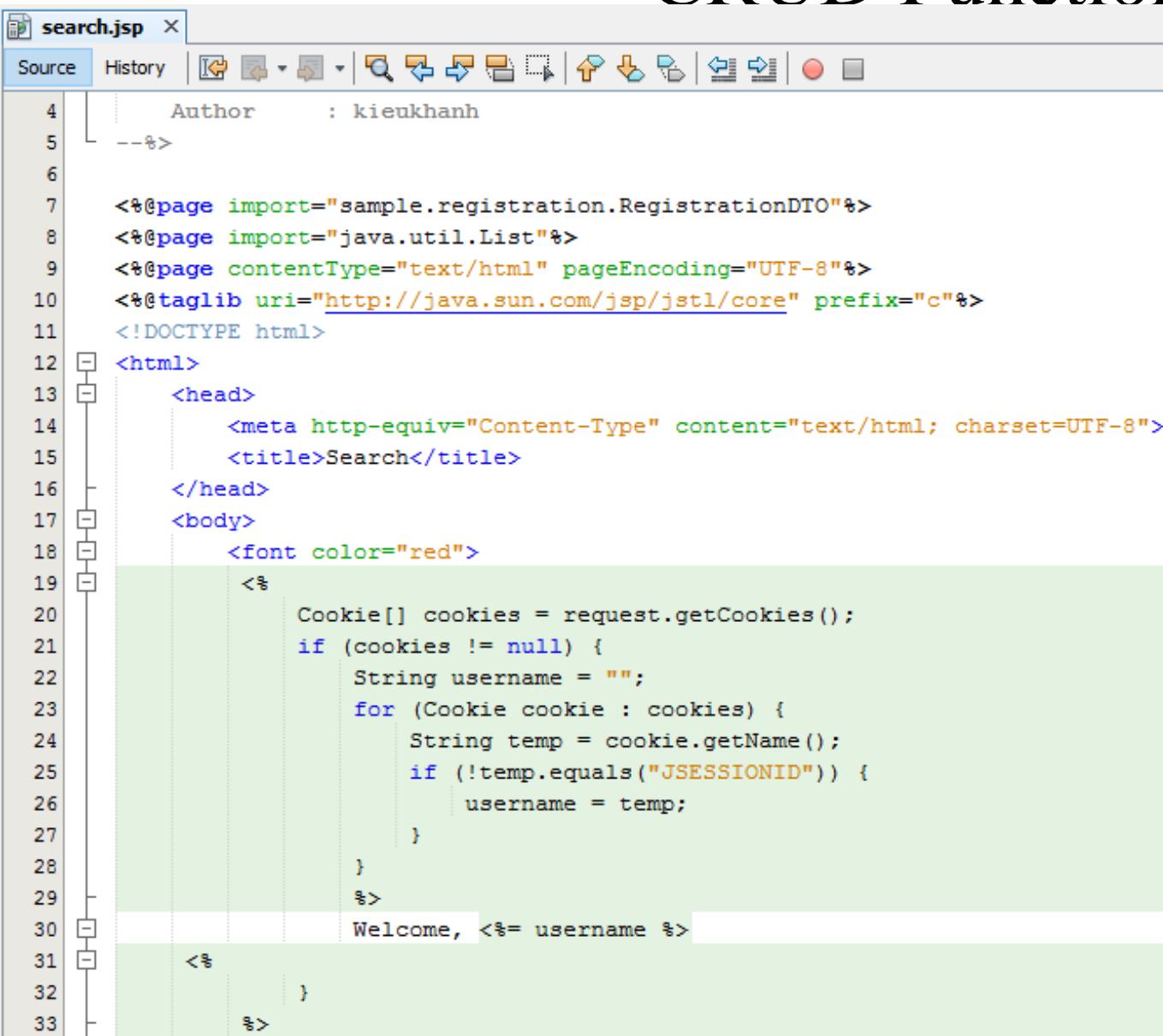
Session Listeners Not Declared in DD

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

Appendix

CRUD Function



The screenshot shows a JSP file named "search.jsp" in an IDE. The code implements a simple search function that retrieves the user's name from a cookie and displays a welcome message.

```
Author      : kieukhanh
--%>

<%@page import="sample.registration.RegistrationDTO"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Search</title>
    </head>
    <body>
        <font color="red">
            <%
                Cookie[] cookies = request.getCookies();
                if (cookies != null) {
                    String username = "";
                    for (Cookie cookie : cookies) {
                        String temp = cookie.getName();
                        if (!temp.equals("JSESSIONID")) {
                            username = temp;
                        }
                    }
                }
                %>
                Welcome, <%= username %>
            <%
                %>
        </font>
    </body>
</html>
```

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

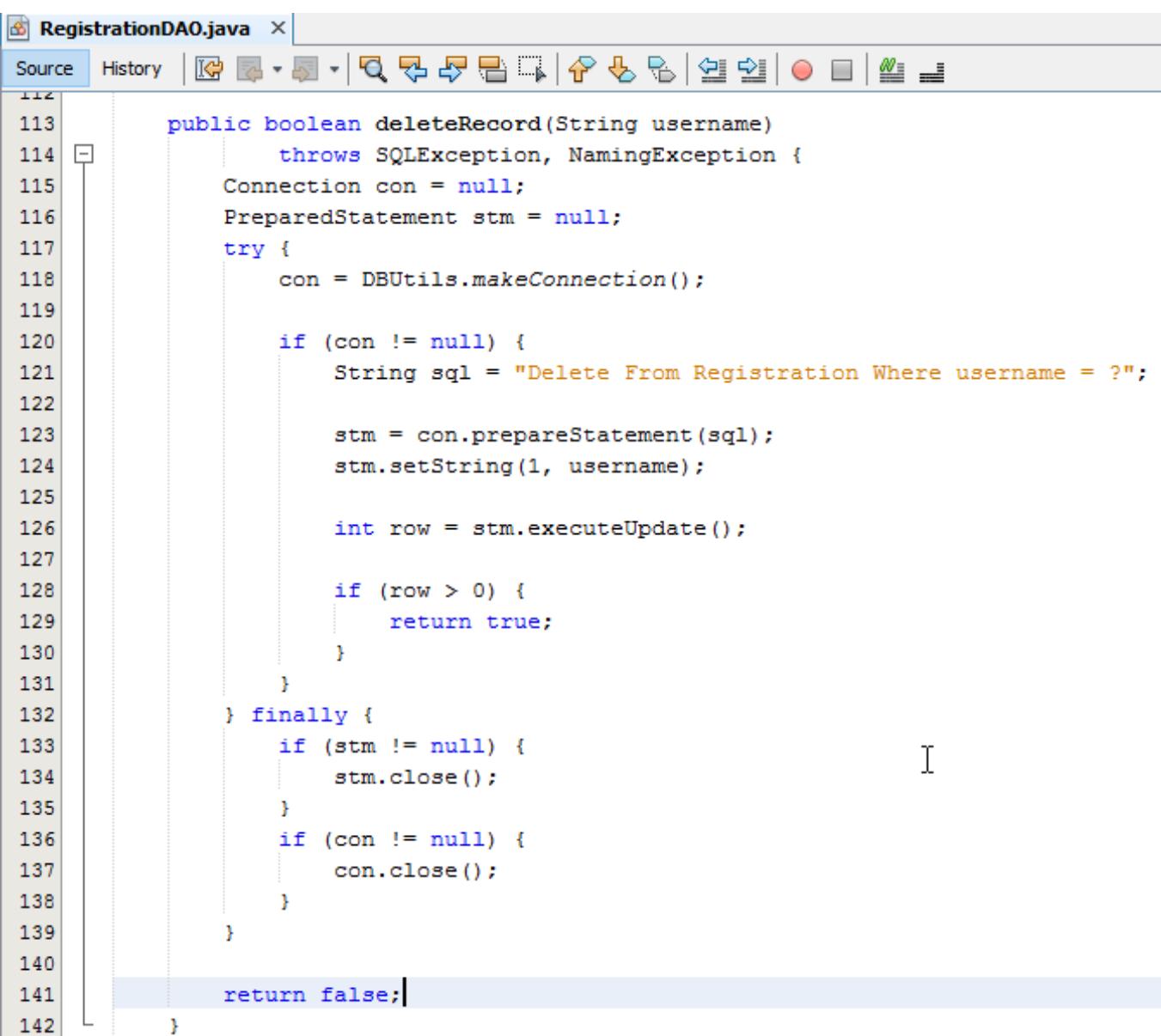
```

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
122
123
124
125
126
127
128
  <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>
<%
} else {
%>
<h2>No record is matched!!!</h2>
<%
}
//end if search Value
%>
</body>
</html>

```

Appendix

CUD Function

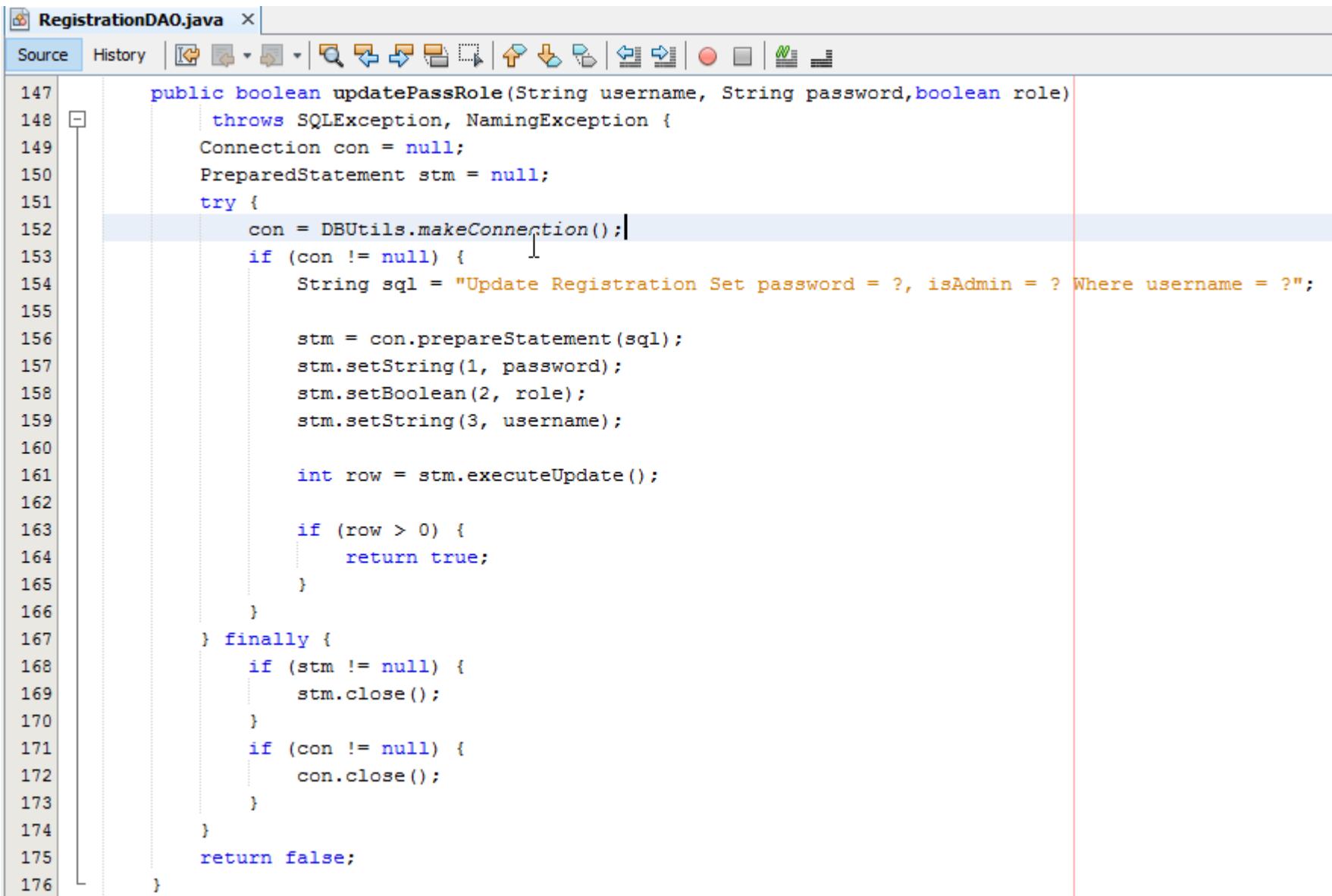


The screenshot shows a Java code editor with the file `RegistrationDAO.java` open. The code implements a `deleteRecord` method that performs a delete operation on a database table named `Registration`. The code uses JDBC to establish a connection, prepare a statement, and execute the delete query. It returns `true` if the record is successfully deleted, and `false` otherwise. 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

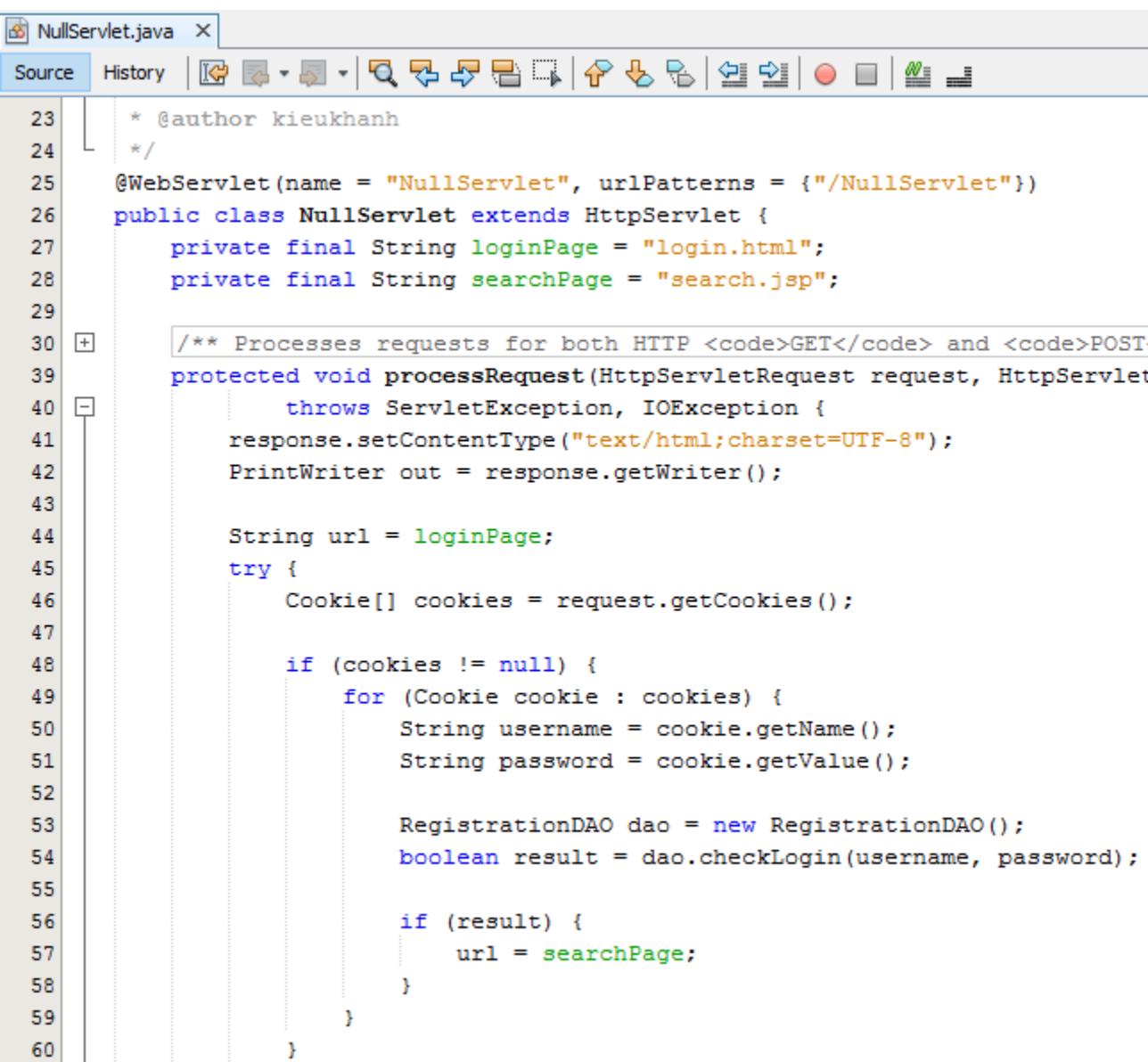


The screenshot shows a Java code editor with the file `RegistrationDAO.java` open. The code implements a `updatePassRole` method that updates a user's password and role in a database. The code uses JDBC and PreparedStatement.

```
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 window titled "NullServlet.java". The code is a servlet named NullServlet that processes requests for both GET and POST methods. It retrieves cookies from the request, checks if they are null, and if not, iterates through them to get the username and password. It then creates a new instance of RegistrationDAO and calls its checkLogin method with the provided credentials. If the result is true, it sets the URL to searchPage; otherwise, it remains loginPage. The code uses standard Java syntax with annotations like @WebServlet and imports for HttpServletRequest, HttpServletResponse, and PrintWriter.

```
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     protected void processRequest(HttpServletRequest request, HttpServlet
33             throws ServletException, IOException {
34         response.setContentType("text/html;charset=UTF-8");
35         PrintWriter out = response.getWriter();
36
37         String url = loginPage;
38         try {
39             Cookie[] cookies = request.getCookies();
40
41             if (cookies != null) {
42                 for (Cookie cookie : cookies) {
43                     String username = cookie.getName();
44                     String password = cookie.getValue();
45
46                     RegistrationDAO dao = new RegistrationDAO();
47                     boolean result = dao.checkLogin(username, password);
48
49                     if (result) {
50                         url = searchPage;
51                     }
52                 }
53             }
54         } finally {
55             out.close();
56         }
57     }
58
59 }
60 }
```

Appendix

CRUD Function

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

Appendix

CRUD Function

The screenshot shows a Java code editor with the following details:

- Title Bar:** The title bar displays "SE1162Servlet.java" and a close button.
- Toolbar:** A standard toolbar with icons for file operations like Open, Save, Print, and a magnifying glass for search.
- Code Area:** The main area contains the Java code for the `SE1162Servlet`. The code defines a class that extends `HttpServlet`, initializes various service methods, and implements a `processRequest` method to handle HTTP requests based on a button parameter.

```
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 = deleteItemServlet;
51             } else if (button.equals("update")) {
52                 url = updatePassRoleServlet;
53             } else if (button.equals("create")) {
54                 url = createAccountServlet;
55             }
56         } catch (IOException ex) {
57             ex.printStackTrace();
58         }
59     }
60
61     // Add methods for login, search, etc.
62 }
```

Appendix

CRUD Function

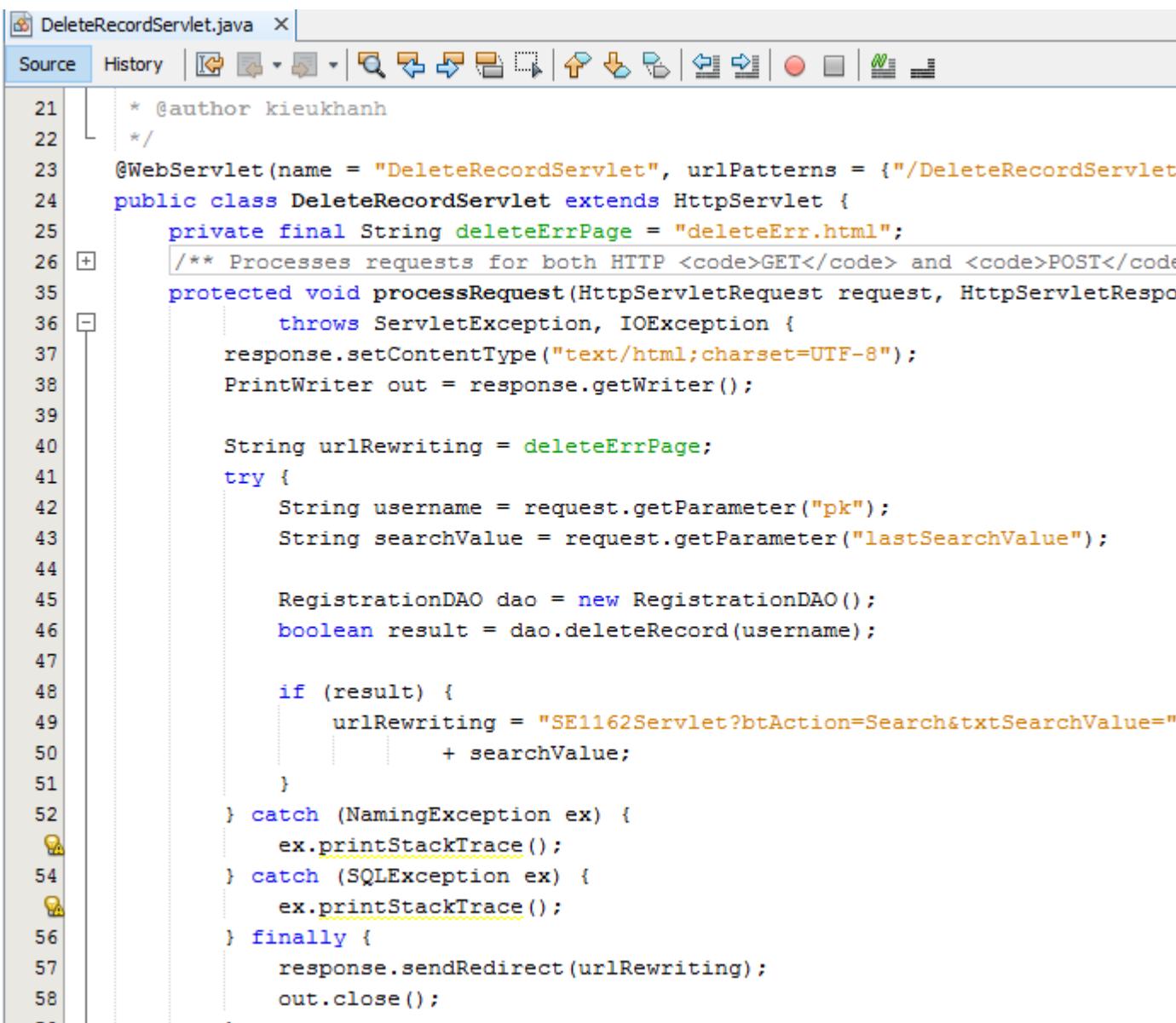
```
56     url = deleteRecordServlet;
57 } else if (button.equals("Update")) {
58     url = updatePassRoleServlet;
59 } else if (button.equals("Add Book To Your Cart")) {
60     url = addItemServlet;
61 } else if (button.equals("View Your Cart")) {
62     url = viewCartPage;
63 } else if (button.equals("Remove 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     out.close();
73 }
74 }
75 }
```

Appendix

CRUD Function

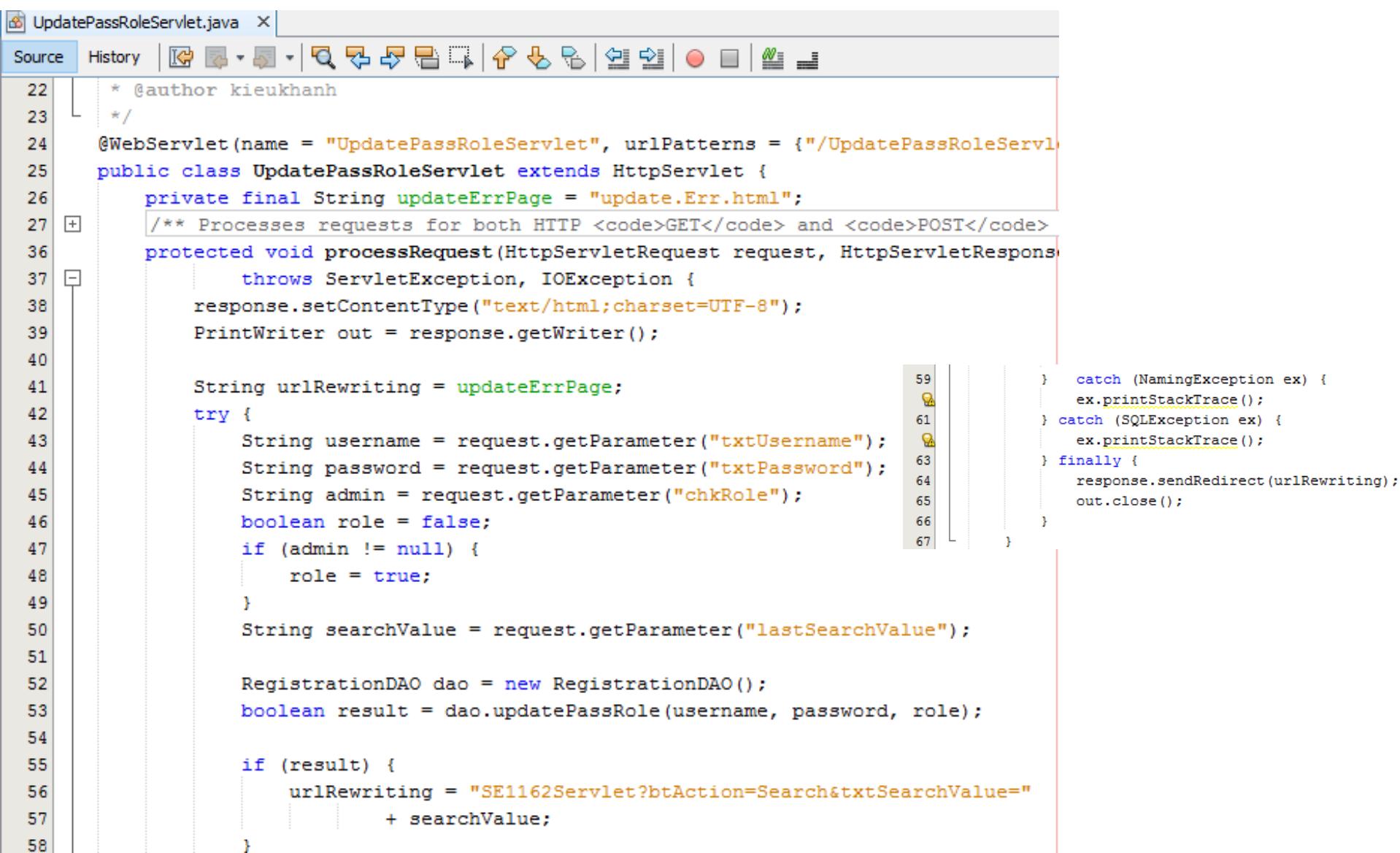


The screenshot shows a Java code editor window with the file `DeleteRecordServlet.java` open. The code implements a `HttpServlet` for handling both GET and POST requests. It uses parameters to identify a user and search value, interacts with a `RegistrationDAO` to delete a record, and then performs a redirect with the updated URL.

```
1  * @author kieukhanh
2  */
3  @WebServlet(name = "DeleteRecordServlet", urlPatterns = {" /DeleteRecordServlet"})
4  public class DeleteRecordServlet extends HttpServlet {
5      private final String deleteErrPage = "deleteErr.html";
6      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
7      protected void processRequest(HttpServletRequest request, HttpServletResponse
8          throws ServletException, IOException {
9              response.setContentType("text/html;charset=UTF-8");
10             PrintWriter out = response.getWriter();
11
12             String urlRewriting = deleteErrPage;
13             try {
14                 String username = request.getParameter("pk");
15                 String searchValue = request.getParameter("lastSearchValue");
16
17                 RegistrationDAO dao = new RegistrationDAO();
18                 boolean result = dao.deleteRecord(username);
19
20                 if (result) {
21                     urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
22                         + searchValue;
23                 }
24             } catch (NamingException ex) {
25                 ex.printStackTrace();
26             } catch (SQLException ex) {
27                 ex.printStackTrace();
28             } finally {
29                 response.sendRedirect(urlRewriting);
30                 out.close();
31             }
32         }
33     }
34 }
```

Appendix

CRUD Function

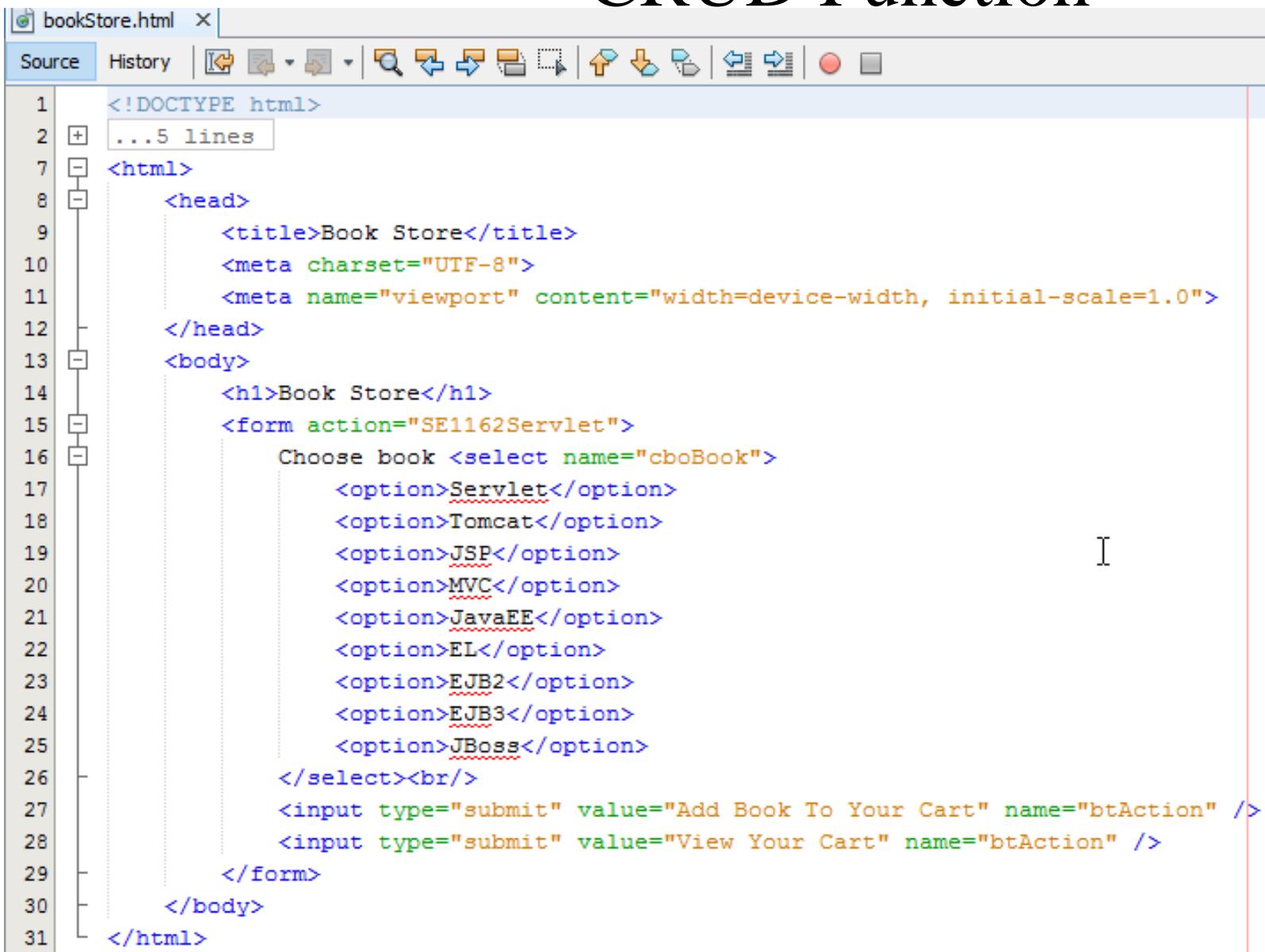


The screenshot shows a Java code editor with the file `UpdatePassRoleServlet.java` open. The code implements a servlet for updating user roles. It uses parameters from the request to update a database via a `RegistrationDAO` object. The code includes exception handling for `NamingException` and `SQLException`.

```
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     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 = updateErrPage;
35         try {
36             String username = request.getParameter("txtUsername");
37             String password = request.getParameter("txtPassword");
38             String admin = request.getParameter("chkRole");
39             boolean role = false;
40             if (admin != null) {
41                 role = true;
42             }
43             String searchValue = request.getParameter("lastSearchValue");
44
45             RegistrationDAO dao = new RegistrationDAO();
46             boolean result = dao.updatePassRole(username, password, role);
47
48             if (result) {
49                 urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
50                         + searchValue;
51             }
52         } catch (NamingException ex) {
53             ex.printStackTrace();
54         } catch (SQLException ex) {
55             ex.printStackTrace();
56         } finally {
57             response.sendRedirect(urlRewriting);
58             out.close();
59         }
60     }
61
62     // Other methods and annotations...
63
64     /**
65      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
66     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
67         processRequest(request, response);
68     }
69
70     /**
71      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
72     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
73         processRequest(request, response);
74     }
75 }
```

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-related technologies: Servlet, Tomcat, JSP, MVC, JavaEE, EL, EJB2, EJB3, and JBoss. There are also two submit buttons for adding a book to the cart or viewing the cart.

```
<!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     }
}
```

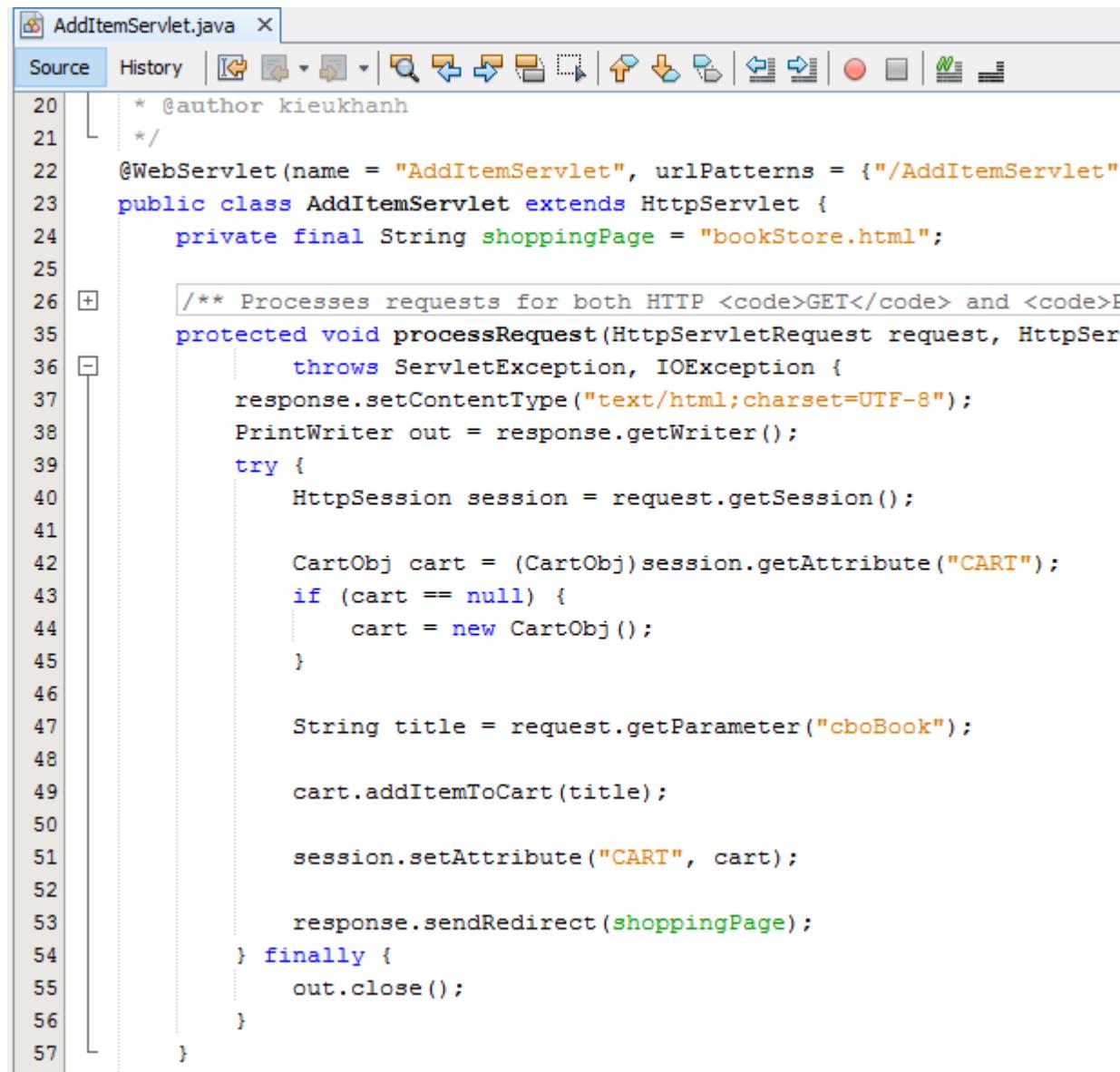
Appendix

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 `HttpSession` to manage the cart.

```
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>P(
27      protected void processRequest(HttpServletRequest request, HttpServletRe
28          throws ServletException, IOException {
29              response.setContentType("text/html;charset=UTF-8");
30              PrintWriter out = response.getWriter();
31              try {
32                  HttpSession session = request.getSession();
33
34                  CartObj cart = (CartObj)session.getAttribute("CART");
35                  if (cart == null) {
36                      cart = new CartObj();
37                  }
38
39                  String title = request.getParameter("cboBook");
40
41                  cart.addItemToCart(title);
42
43                  session.setAttribute("CART", cart);
44
45                  response.sendRedirect(shoppingPage);
46              } finally {
47                  out.close();
48              }
49      }
50
51
52
53
54
55
56
57 }
```

Appendix

CRUD Function

Appendix

CRUD Function

```
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
```

```
</thead>
<tbody>
<form action="SE1162Servlet">
    <%
        Map<String, Integer> items = cart.getItems();
        int count = 0;
        for (Map.Entry item : items.entrySet()) {
            %
    <tr>
        <td>
            <%= ++count %>
        </td>
        <td>
            <%= item.getKey() %>
        </td>
        <td>
            <%= item.getValue() %>
        </td>
        <td>
            <input type="checkbox" name="chkItem" value="<%= item.getKey() %>" />
        </td>
    </tr>
    %
    //end for
%
<tr>
    <td colspan="3">
        <a href="bookStore.html">Add More Items to Your Cart</a>
    </td>
```

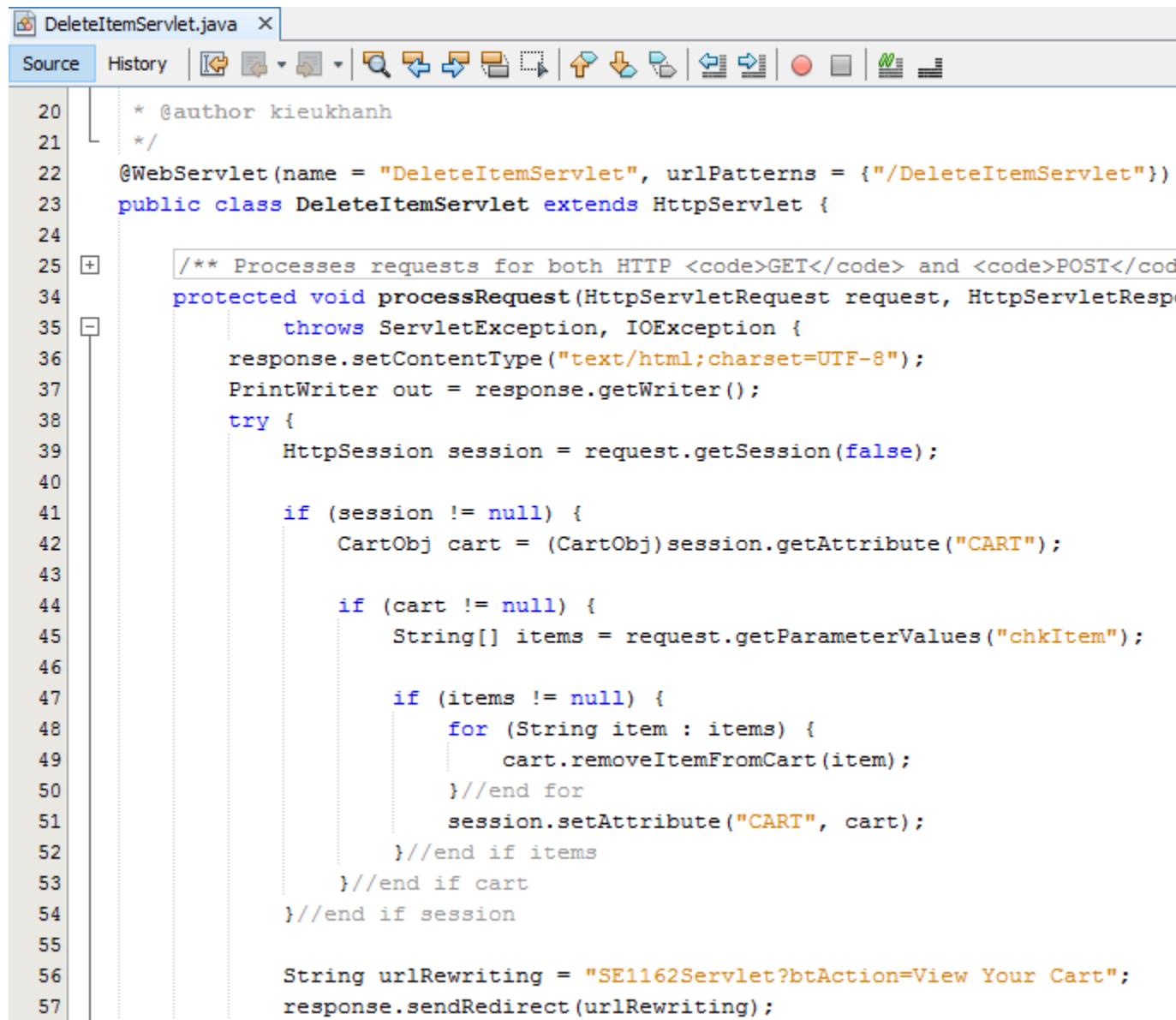
Appendix

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

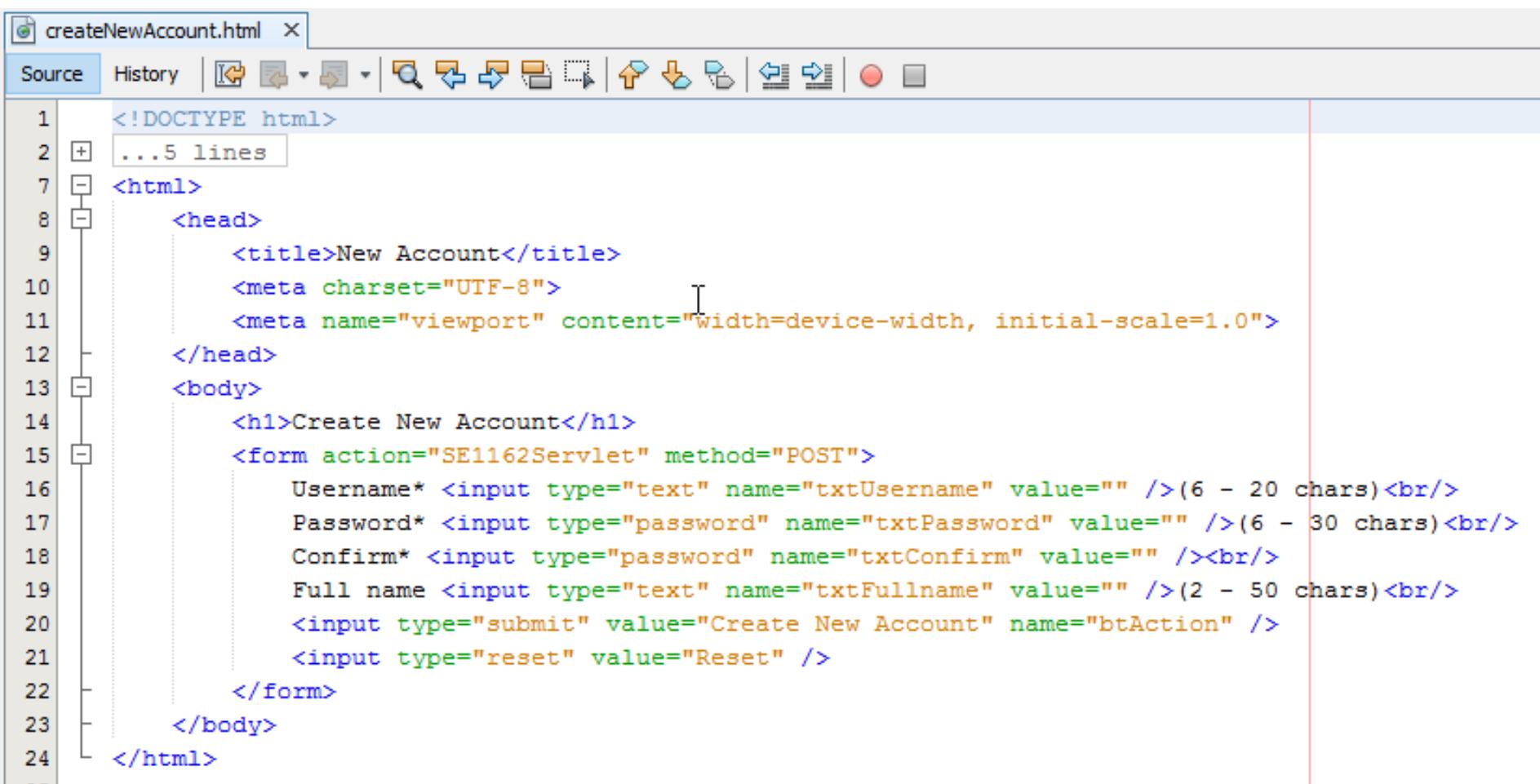


The screenshot shows a Java code editor window titled "DeleteItemServlet.java". The code is a servlet for deleting items from a shopping cart. It uses annotations for web service methods and handles both GET and POST requests. It retrieves a session, gets the "CART" attribute, and removes items based on selected checkboxes. Finally, it updates the session and 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      * @param request the servlet request
27      * @param response the servlet response
28      * @throws ServletException if a servlet-specific error occurs
29      * @throws IOException if an I/O error occurs
30      */
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(false);
37
38              if (session != null) {
39                  CartObj cart = (CartObj)session.getAttribute("CART");
40
41                  if (cart != null) {
42                      String[] items = request.getParameterValues("chkItem");
43
44                      if (items != null) {
45                          for (String item : items) {
46                              cart.removeItemFromCart(item);
47                          } //end for
48                          session.setAttribute("CART", cart);
49                      } //end if items
50                  } //end if cart
51              } //end if session
52
53
54
55
56              String urlRewriting = "SE1162Servlet?btAction=View Your Cart";
57              response.sendRedirect(urlRewriting);
58          }
59      }
60
61      // <editor-fold defaultstate="collapsed" desc="HttpServlet methods">
62      /**
63       * Handles the HTTP <code>GET</code> method.
64       * @param request the servlet request
65       * @param response the servlet response
66       * @throws ServletException if a servlet-specific error occurs
67       * @throws IOException if an I/O error occurs
68       */
69      @Override
70      protected void doGet(HttpServletRequest request, HttpServletResponse response)
71          throws ServletException, IOException {
72          processRequest(request, response);
73      }
74
75      /**
76       * Handles the HTTP <code>POST</code> method.
77       * @param request the servlet request
78       * @param response the servlet response
79       * @throws ServletException if a servlet-specific error occurs
80       * @throws IOException if an I/O error occurs
81       */
82      @Override
83      protected void doPost(HttpServletRequest request, HttpServletResponse response)
84          throws ServletException, IOException {
85          processRequest(request, response);
86      }
87
88      // 
89  }
```

Appendix

CRUD Function

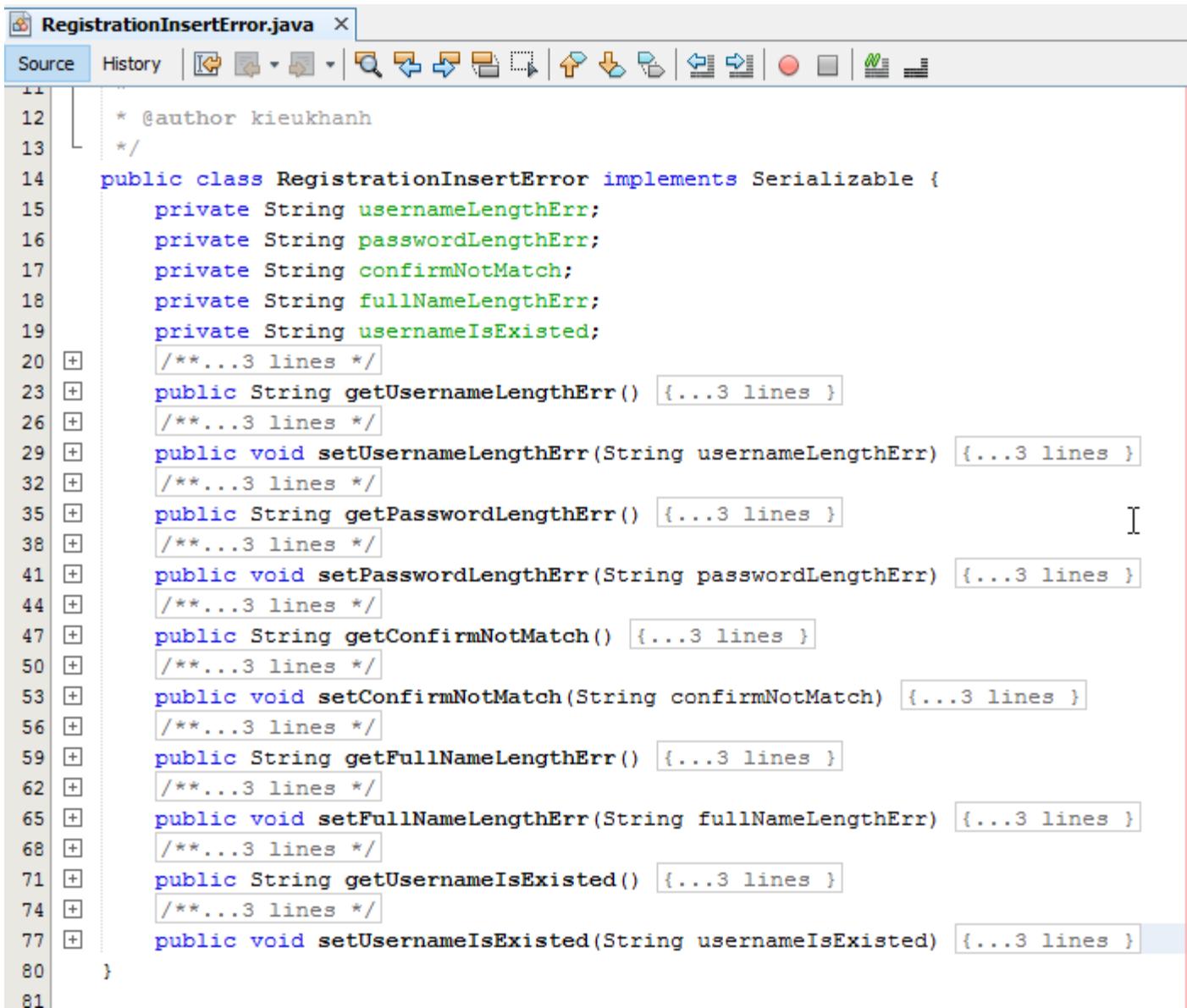


The screenshot shows a code editor window with the title bar "createNewAccount.html X". The menu bar includes "Source" and "History". Below the menu is a toolbar with various icons for file operations. The main area displays an HTML file with line numbers on the left. The code defines a form for creating a new account, including fields for Username, Password, Confirm, Full name, and action 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="SE11625Servlet" 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

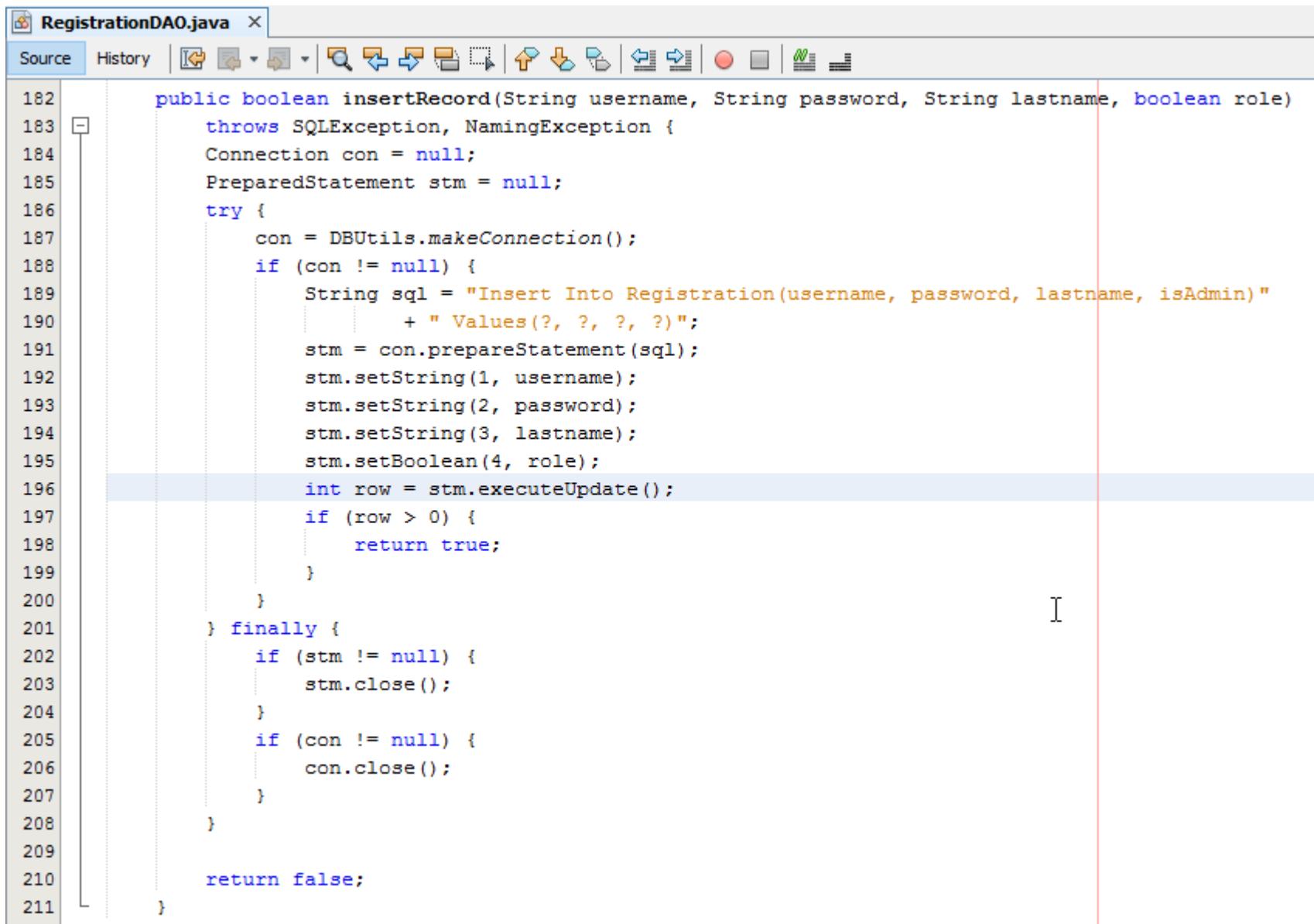


The screenshot shows a Java code editor window with the file `RegistrationInsertError.java` open. The code defines a class `RegistrationInsertError` that implements `Serializable`. The class contains several private fields and corresponding public getters and setters for error messages related to registration validation. The code is annotated with `/* @author kieukhanh */`.

```
11
12     * @author kieukhanh
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      * ...3 lines */
22     public String getUsernameLengthErr() { ...3 lines }
23     /**
24      * ...3 lines */
25     public void setUsernameLengthErr(String usernameLengthErr) { ...3 lines }
26     /**
27      * ...3 lines */
28     public String getPasswordLengthErr() { ...3 lines }
29     /**
30      * ...3 lines */
31     public void setPasswordLengthErr(String passwordLengthErr) { ...3 lines }
32     /**
33      * ...3 lines */
34     public String getConfirmNotMatch() { ...3 lines }
35     /**
36      * ...3 lines */
37     public void setConfirmNotMatch(String confirmNotMatch) { ...3 lines }
38     /**
39      * ...3 lines */
40     public String getFullNameLengthErr() { ...3 lines }
41     /**
42      * ...3 lines */
43     public void setFullNameLengthErr(String fullNameLengthErr) { ...3 lines }
44     /**
45      * ...3 lines */
46     public String getUsernameIsExisted() { ...3 lines }
47     /**
48      * ...3 lines */
49     public void setUsernameIsExisted(String usernameIsExisted) { ...3 lines }
50 }
```

Appendix

CRUD Function

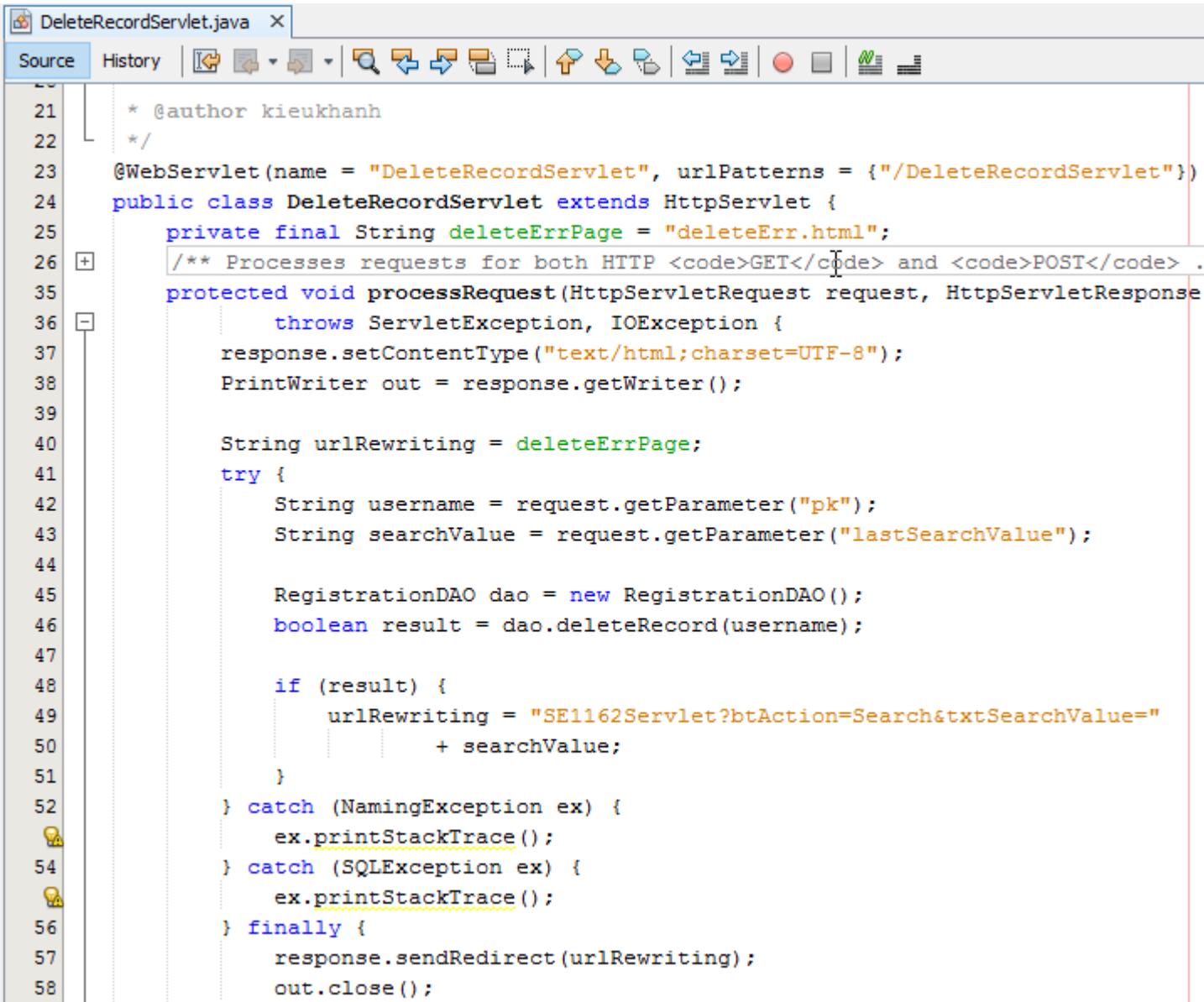


The screenshot shows a Java code editor window titled "RegistrationDAO.java". The code is a Java class containing a single method, insertRecord, which performs an insert operation into a database table named "Registration". The code uses JDBC to make a connection, prepare a statement, and execute an update query. It returns true if the insertion was successful and false otherwise. The code editor has a toolbar with various icons for file operations, and the code is displayed with line numbers on the left.

```
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

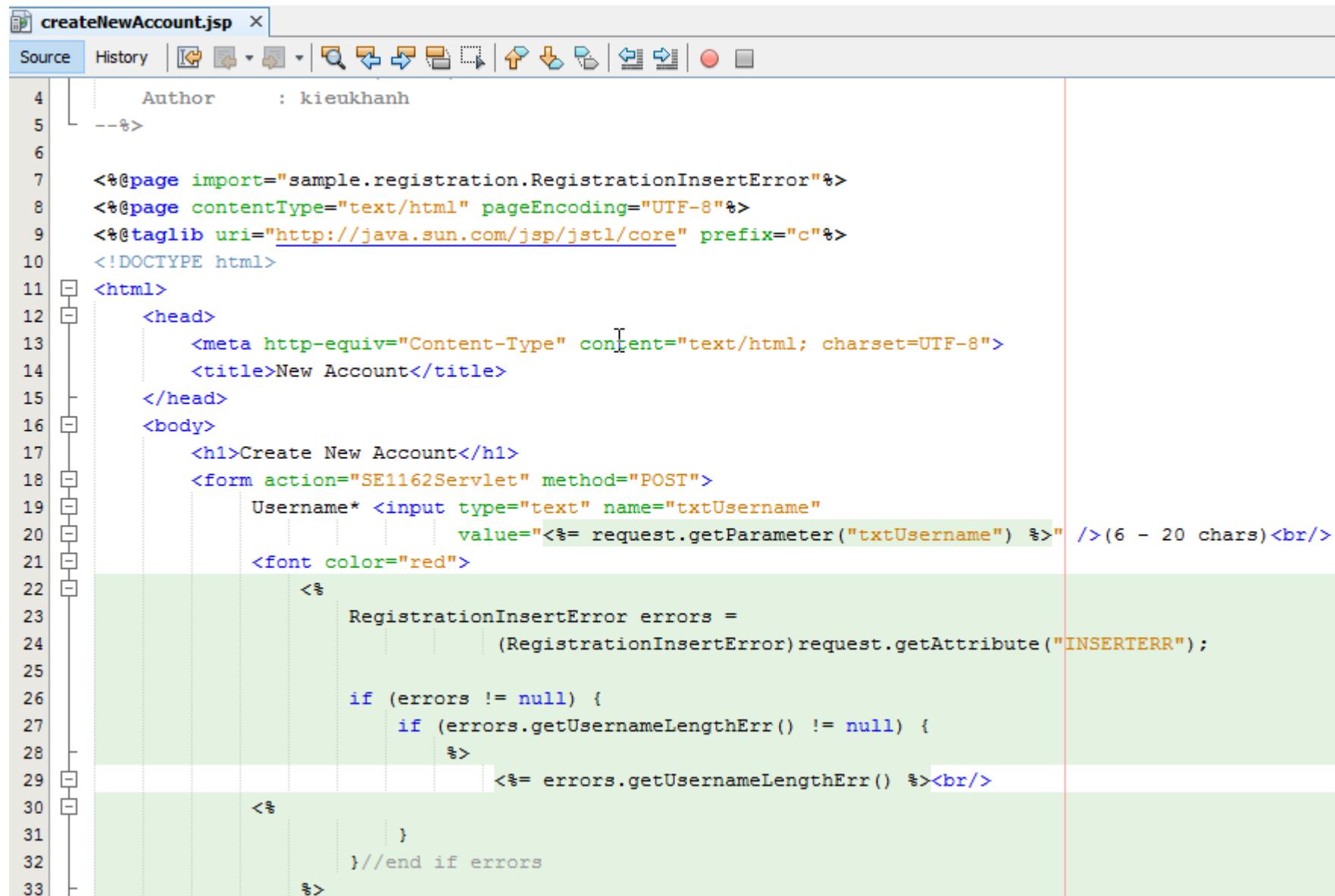


The screenshot shows a Java code editor with the file `DeleteRecordServlet.java` open. The code implements a `HttpServlet` for deleting records from a database. It uses `JSP` for URL rewriting and `RegistrationDAO` for database operations. The code includes error handling for `NamingException` and `SQLException`.

```
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 Server Page (JSP) editor window with the file name 'createNewAccount.jsp'. The code is written in JSP syntax, using Java code within scriptlets and JSTL tags for database interaction.

```
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           }
33         //end if errors
34       <%>
```

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">
```

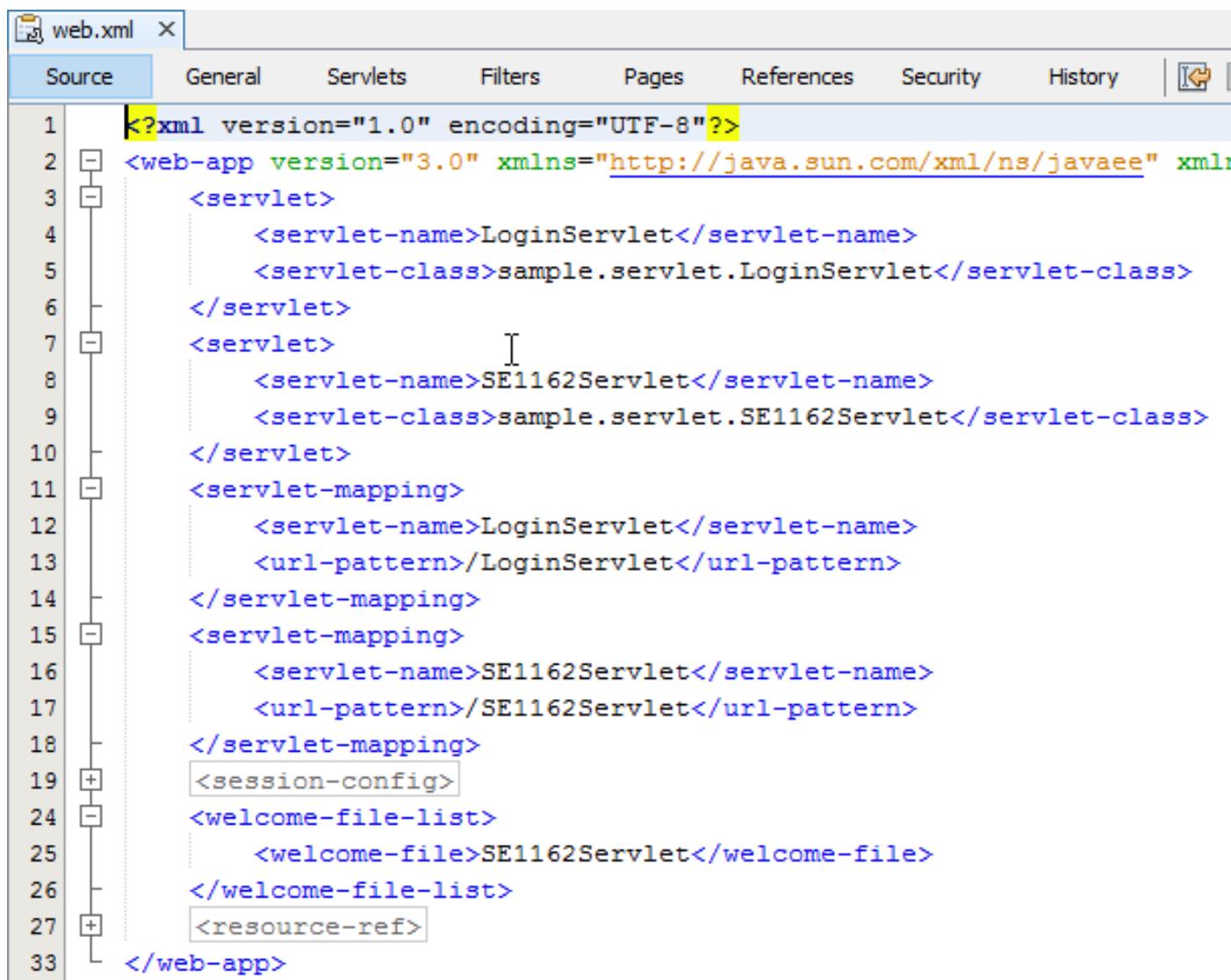
Appendix

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



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" 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>
```