Fpt University

Session Management

Sessions and Listeners
Technique: Error Handling in Servlets

#Session #UrlRewriting #HiddenForm
#SessionTracking #Cookie
#ShoppingCart #ErrorHandle
#JSP

Fpt University

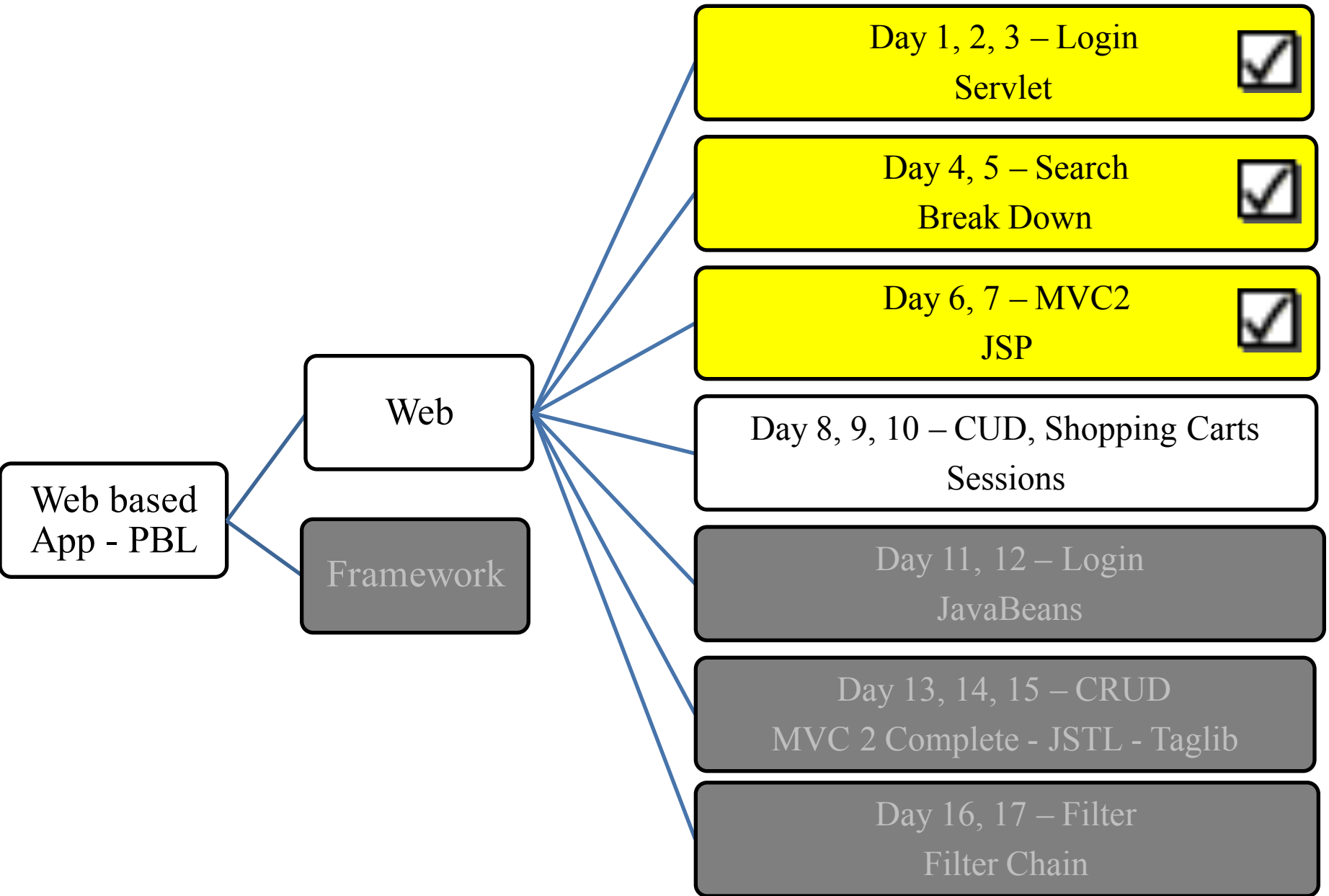
Review

- **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


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

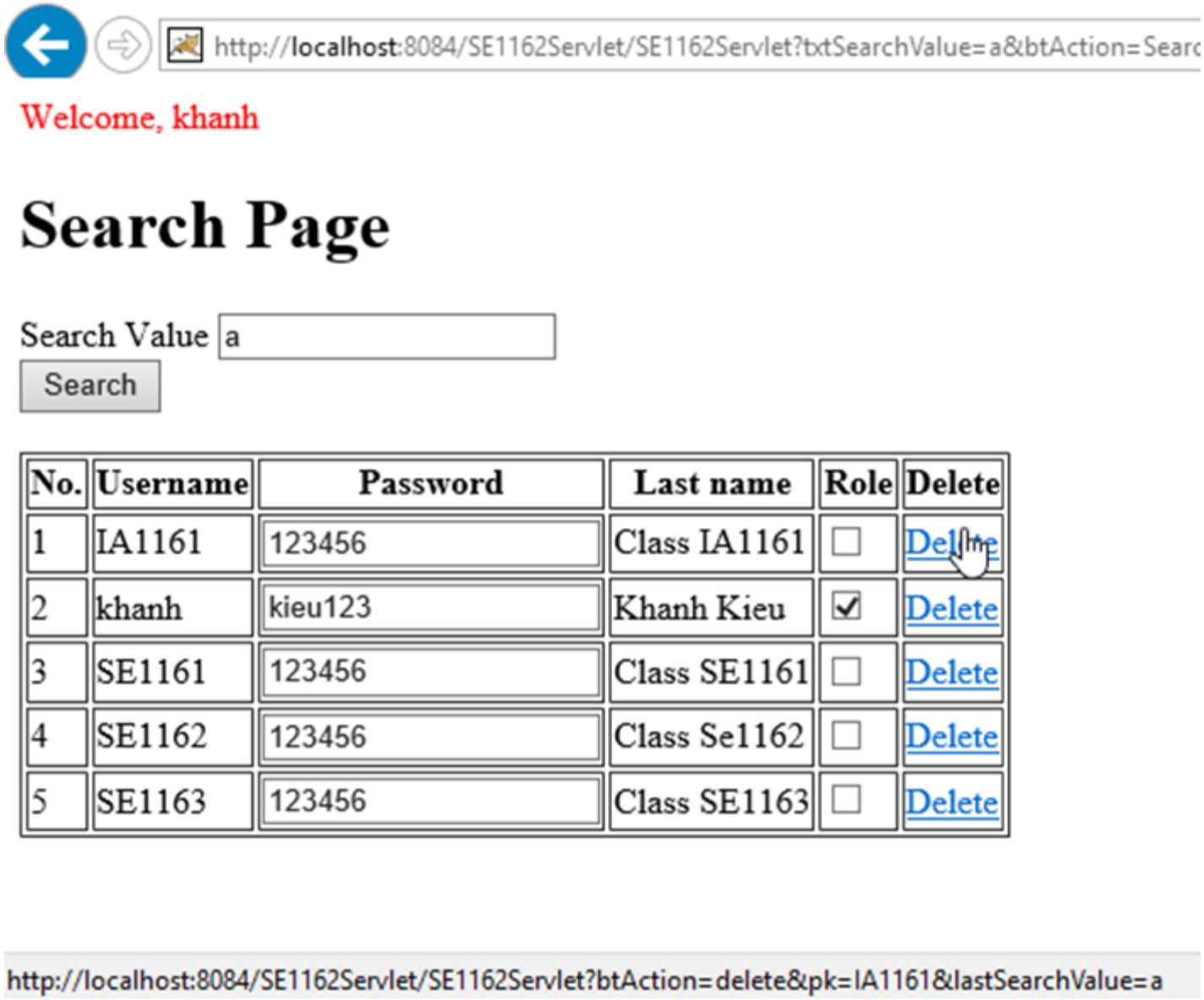
How to write CRUD Web Application

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

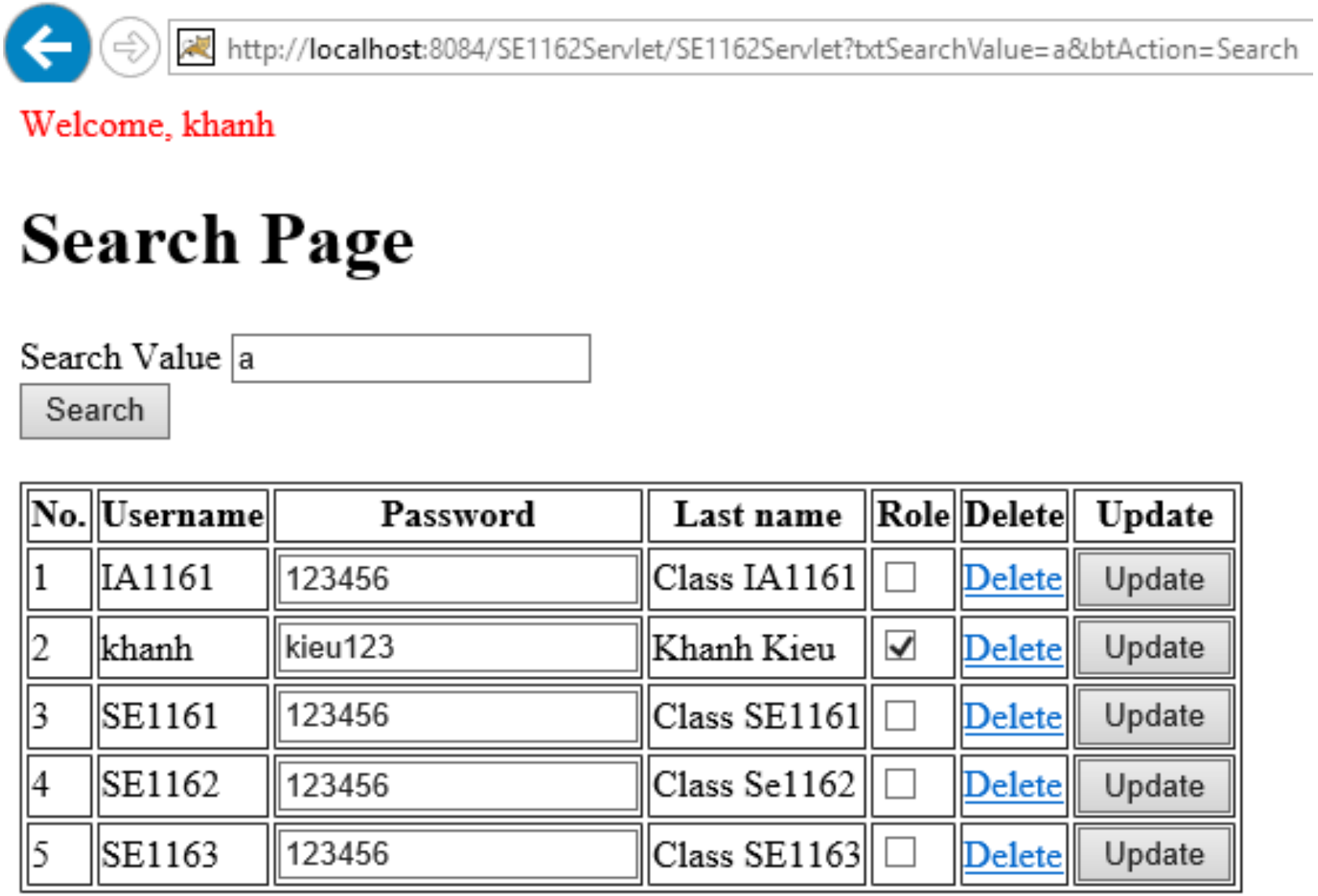
How to write CRUD Web Application

Expectation



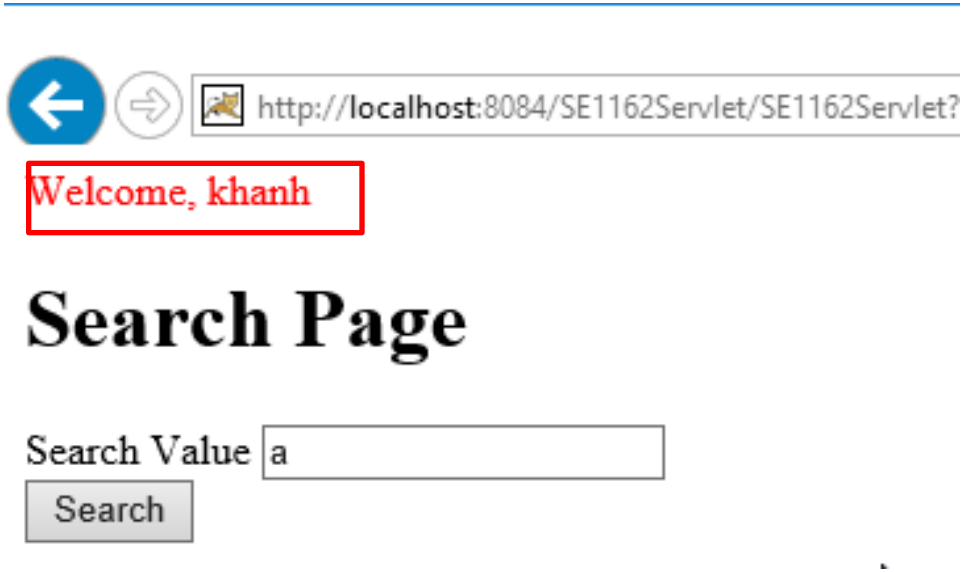
How to write CRUD Web Application

Expectation



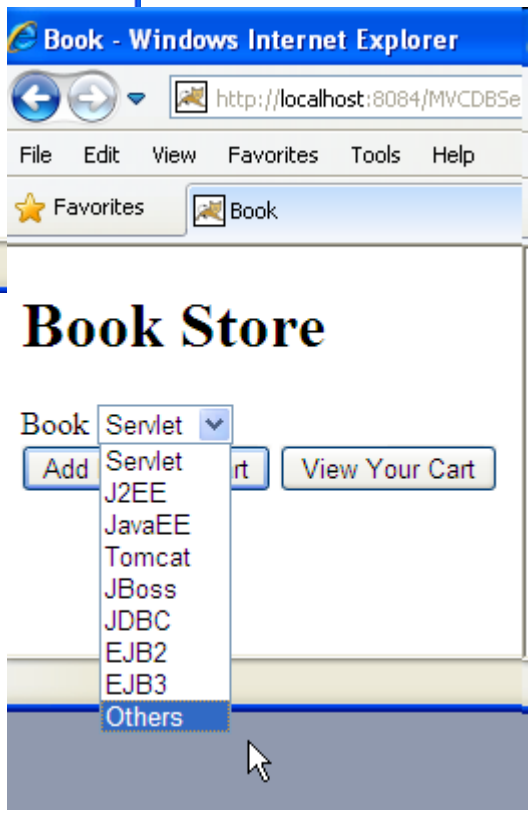
How to write CRUD Web Application

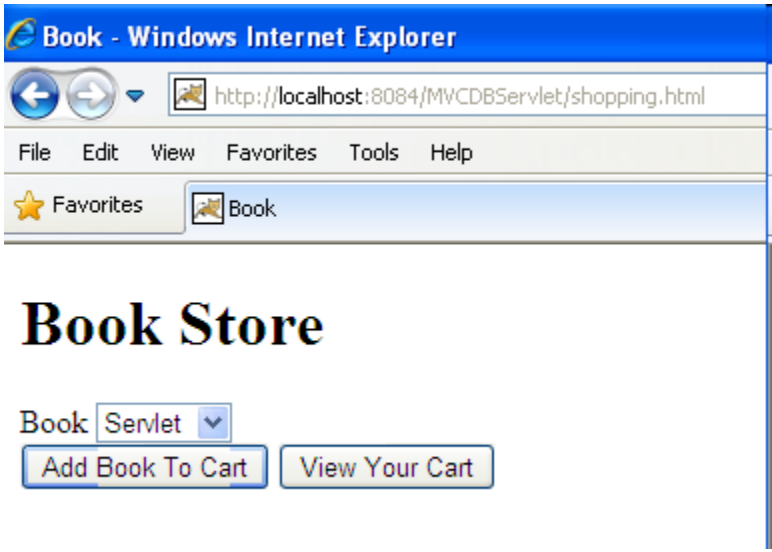
Expectation



How to write CRUD Web Application

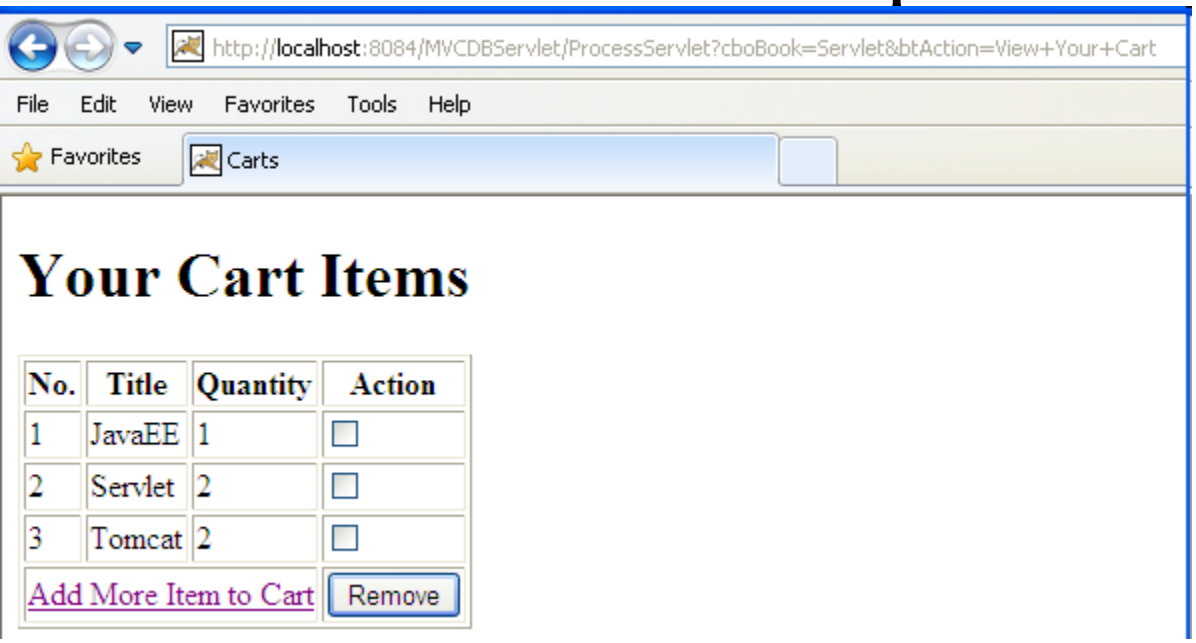
Expectation

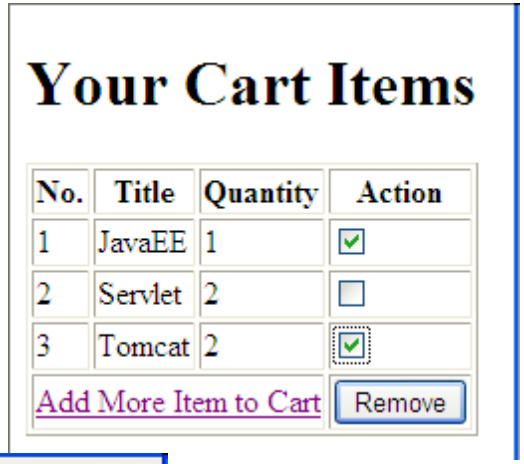


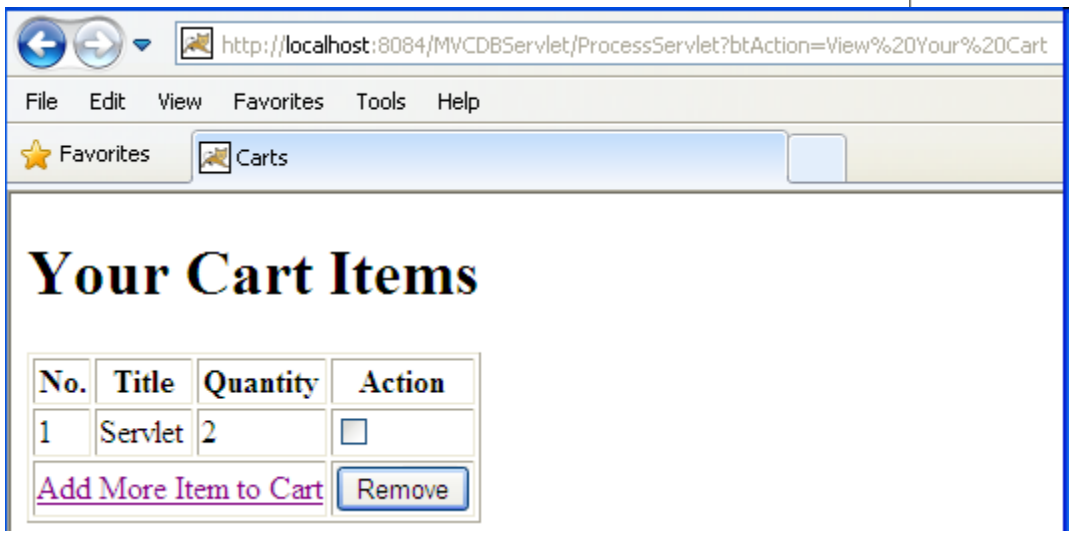


How to write CRUD Web Application

Expectation

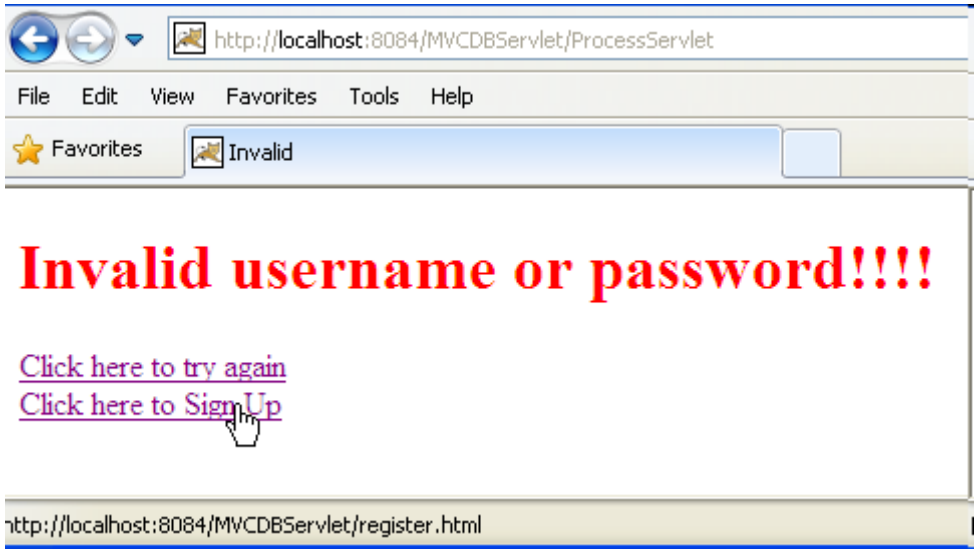


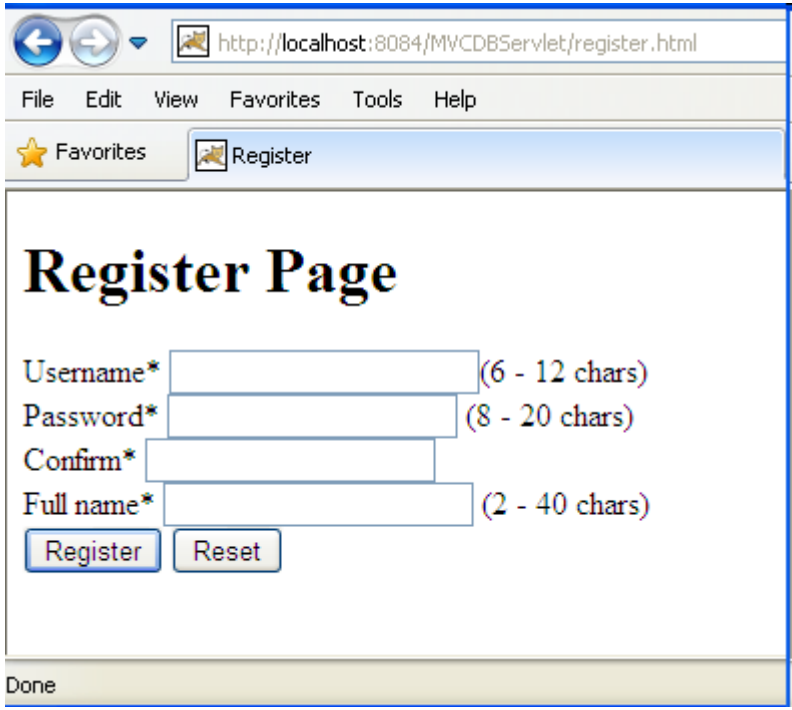




How to write CRUD Web Application

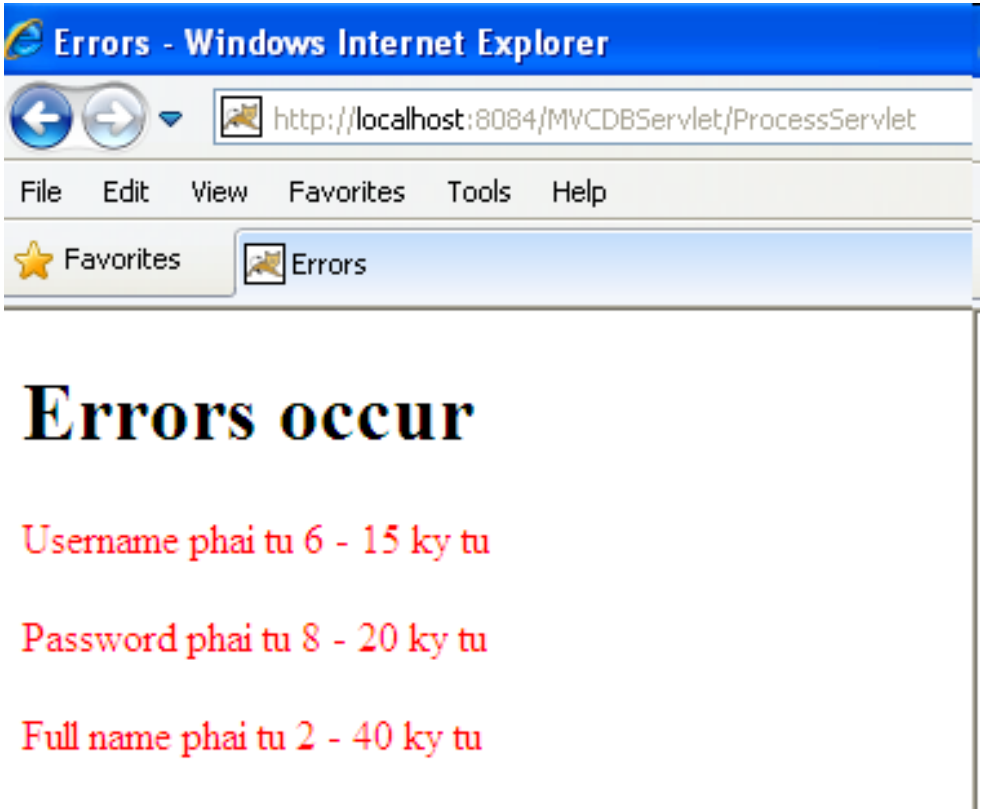
Expectation

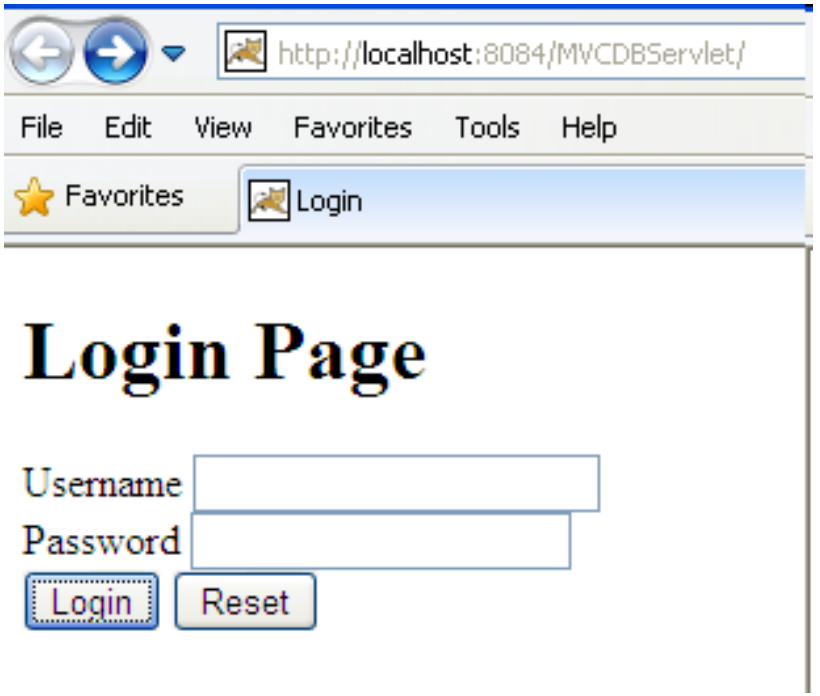




How to write CRUD Web Application

Expectation





Sessions & Listeners

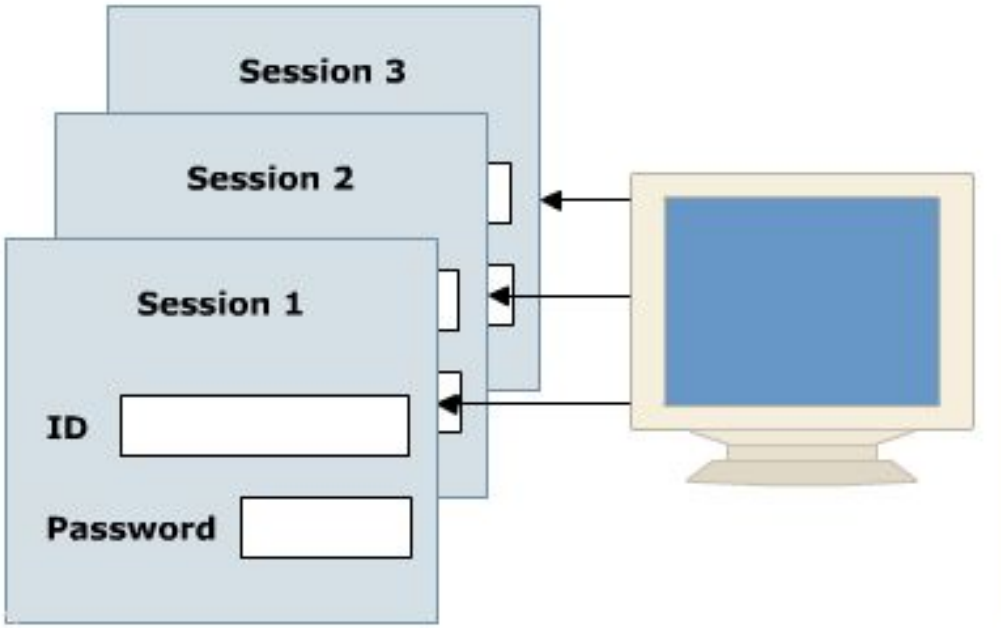
Session

- 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



Session Tracking

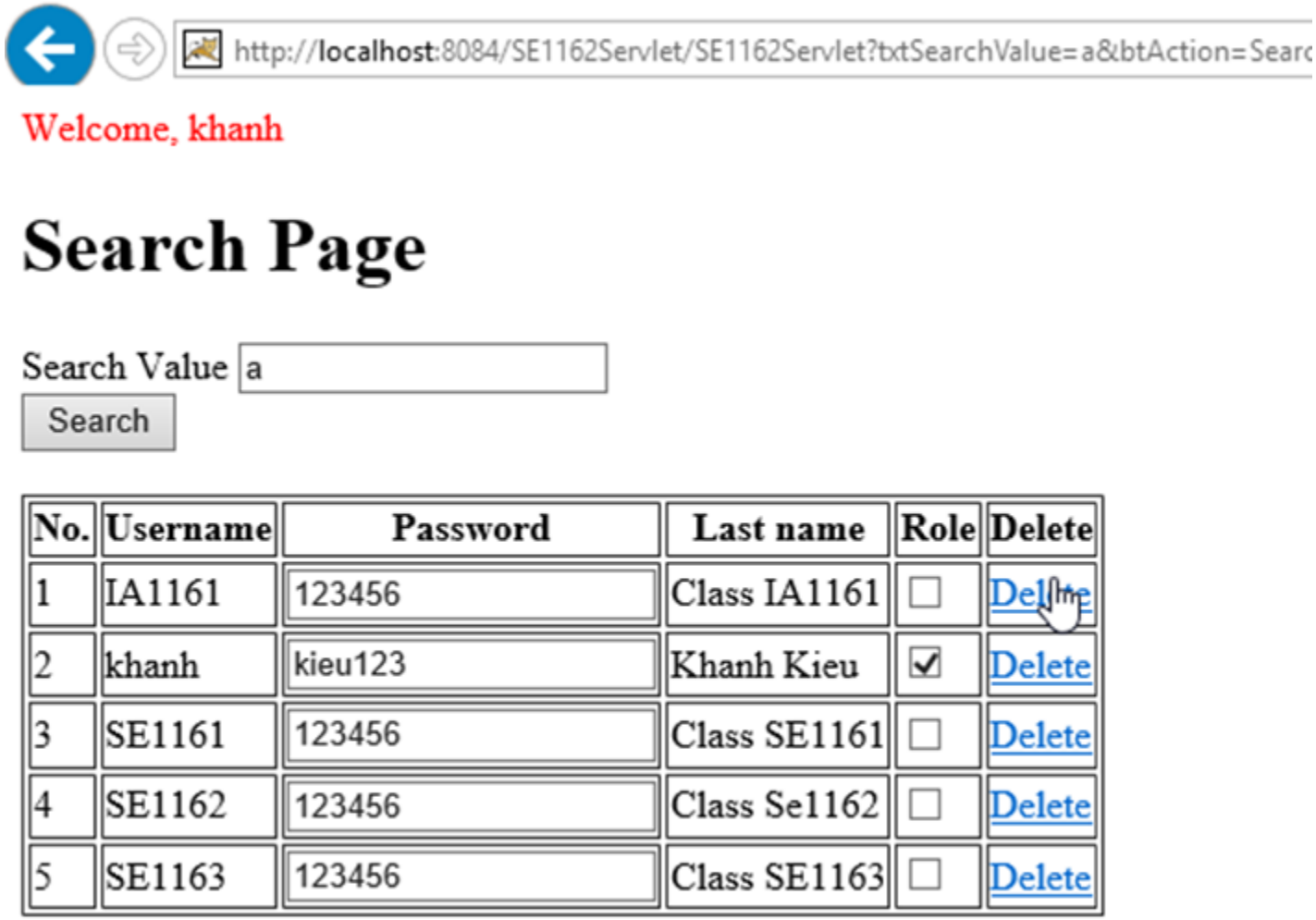
Sessions & Listeners

URL Rewriting

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

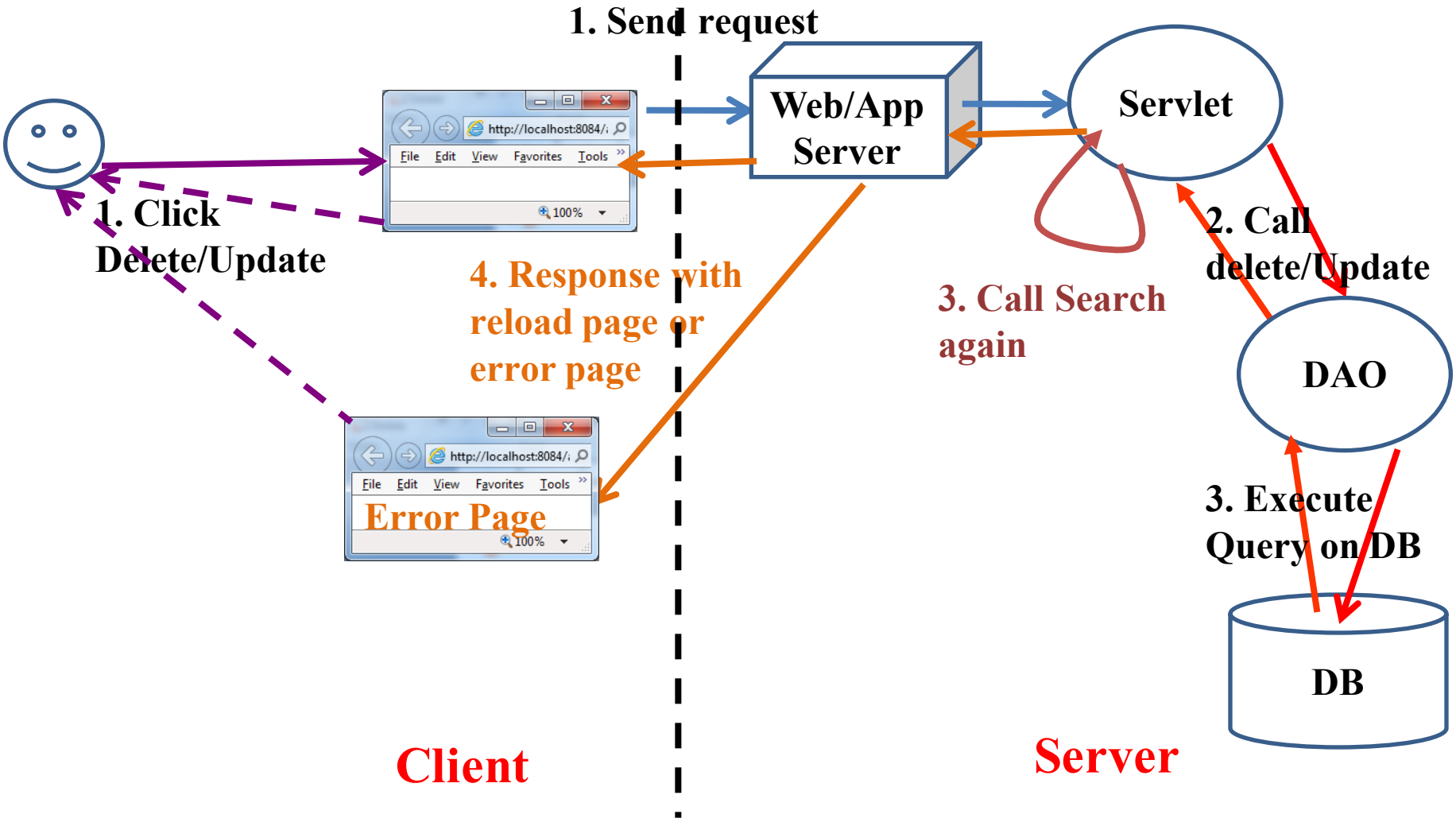
How to write CRUD Web Application

Delete Function



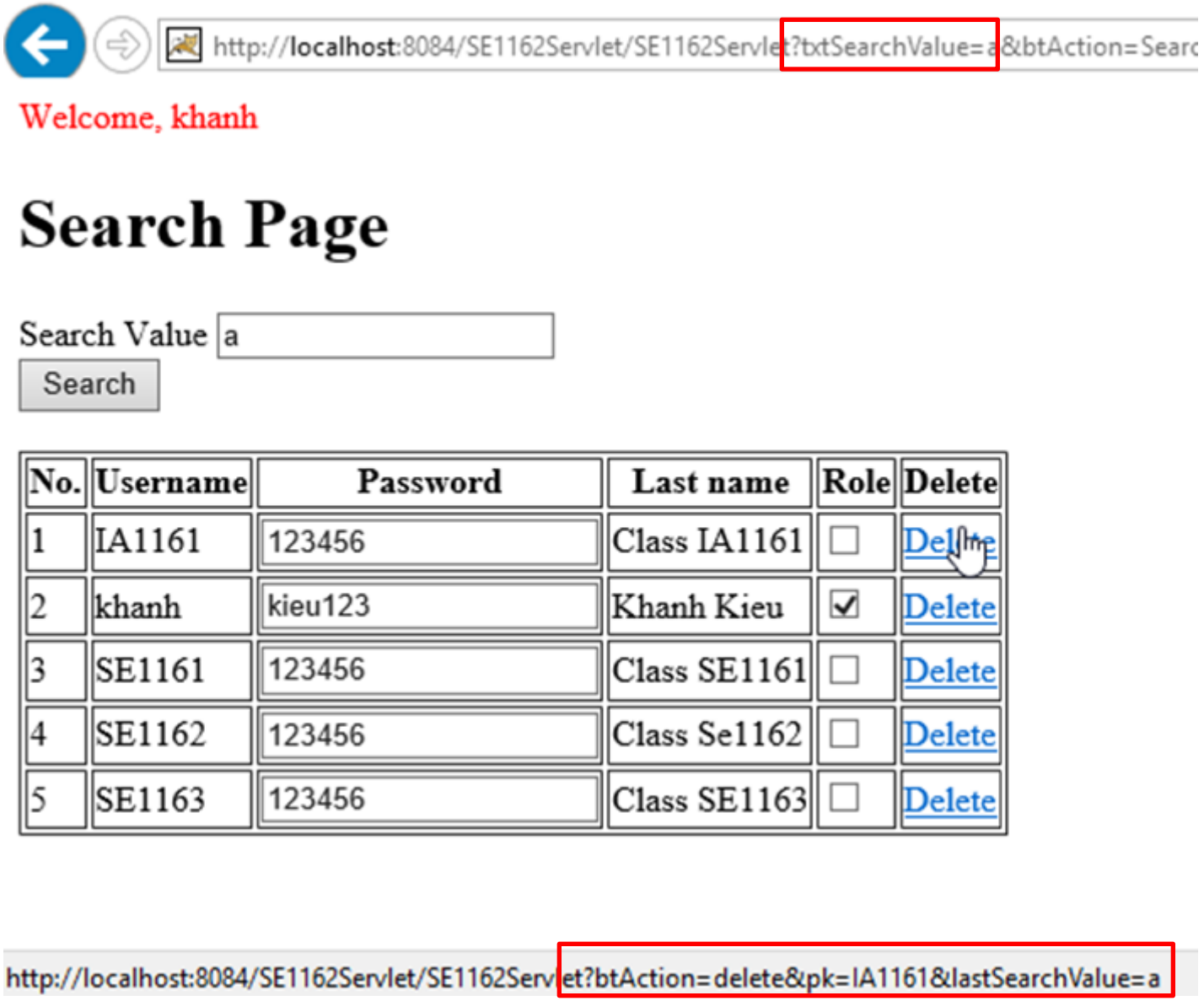
How to write CRUD Web Application

Interactive Server Model



How to write CRUD Web Application

Delete Function



Sessions & Listeners

Hidden Form Fields

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

  <input type="hidden" name="…" value="…">
- **Ex**

  <input type="hidden" name="productId" value="P01">
- **Advantages**
  - **Simplest** way to implement session tracking
  - Displays **nothing** on the HTML page but can be used to hold any kind of data
  - Helps to maintain a **connection between two pages**
- **Disadvantages**:
  - **Work** on the **dynamic pages**.
  - This method of session tracking **displays sensitive information to the user.**

How to write CRUD Web Application

Update Function

Welcome, khanh

Search Page

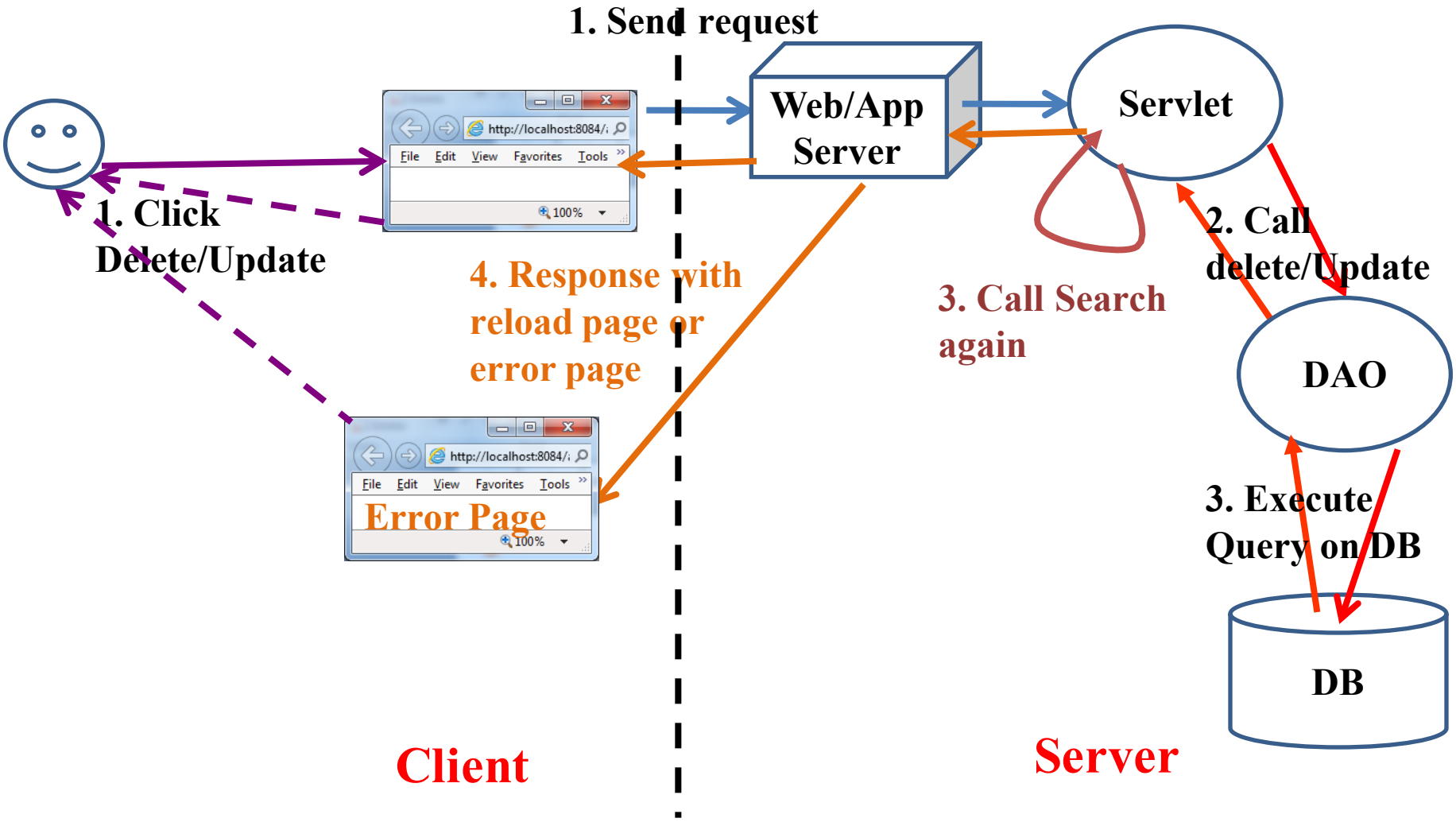
Search Value a

Search

No.	Username	Password	Last name	Role	Delete	Update
1	IA1161	123456	Class IA1161	☐	Delete	Update
2	khanh	kieu123	Khanh Kieu	☑	Delete	Update
3	SE1161	123456	Class SE1161	☐	Delete	Update
4	SE1162	123456	Class Se1162	☐	Delete	Update
5	SE1163	123456	Class SE1163	☐	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\LoggedUserName\Cookies\LoggedUserName@ContextPath[n].txt**
    - **C:\Users\LoggedUserName\AppData\Local\Microsoft\Windows\Temporary Internet Files \LoggedUserName@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	- **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	- **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	- **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	- **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

Fpt University

Sessions & Listeners
Cookies

```
// Create cookies for first and last names.
Cookie firstName = new Cookie("first_name",
        request.getParameter("first_name"));
Cookie lastName = new Cookie("last_name",
        request.getParameter("last_name"));

// Set expiry date after 24 Hrs for both the cookies.
firstName.setMaxAge(60 * 60 * 24);
lastName.setMaxAge(60 * 60 * 24);

// Add both the cookies in the response header.
response.addCookie(firstName);
response.addCookie(lastName);
```

Sessions & Listeners
Cookies

- **Reading Cookie**

Methods	Descriptions
getCookies	- **Cookie [] cookies = request.getCookies();** - Returns an array containing all of the Cookie objects the client sends with the request
getMaxAge	- **public int getMaxAge();** - Returns the maximum age of the cookie. - Returns an integer which specify the maximum age of the cookies in seconds
getValue	- **public String getValue();** - Returns the value of the cookie
getName	- **public String getName()** - Returns the name of cookie. Once the cookie has been created its name cannot be changed
getPath	- **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

- **Reading Cookie**

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

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

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

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

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

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

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

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

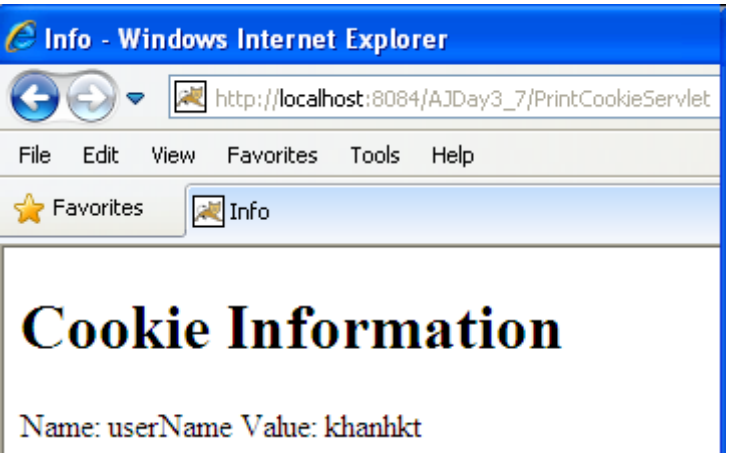
Sessions & Listeners
Cookies

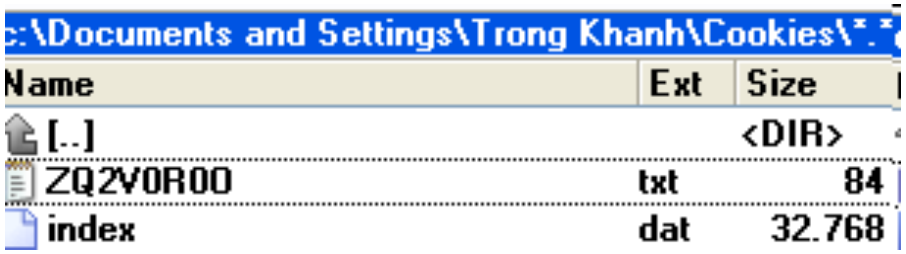
- **Delete Cookie**

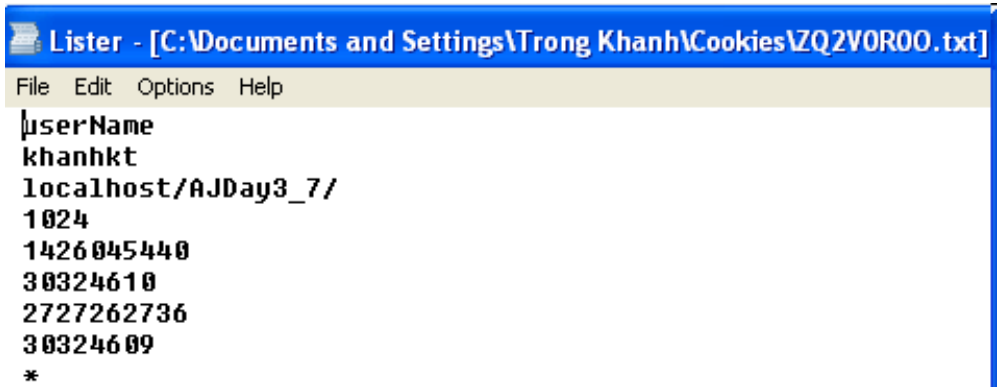
```
public class DeleteCookie extends HttpServlet {

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

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

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

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

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

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

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

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

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

Sessions & Listeners

Cookies – Example







Sessions & Listeners

Cookies – Example

AddCookieServlet.java

```
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

PrintCookieServlet.java

```
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      }
```

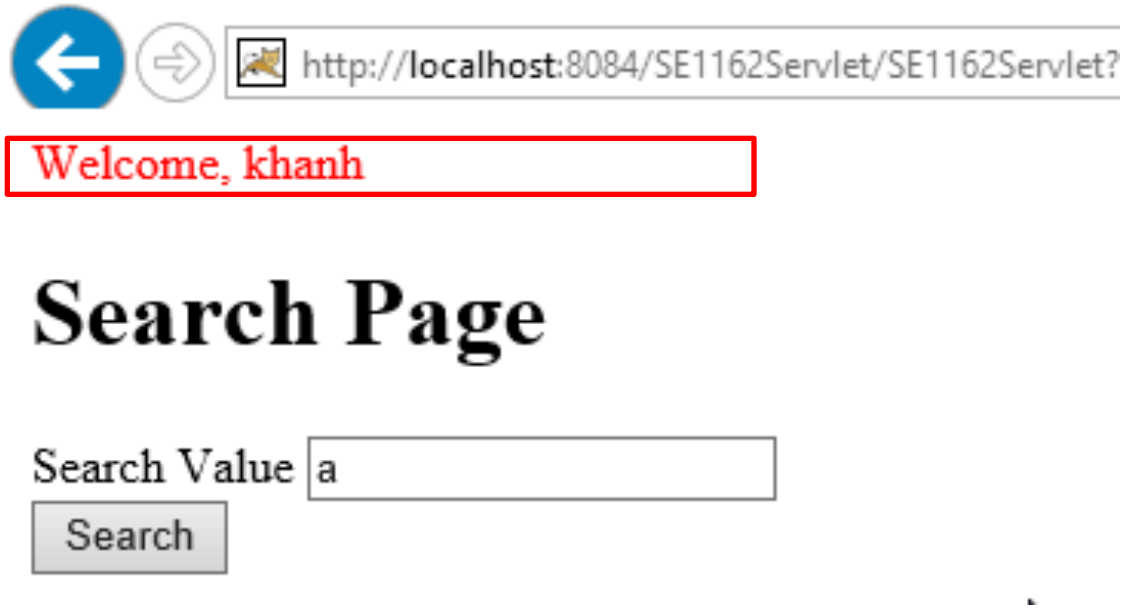
How to write CRUD Web Application

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

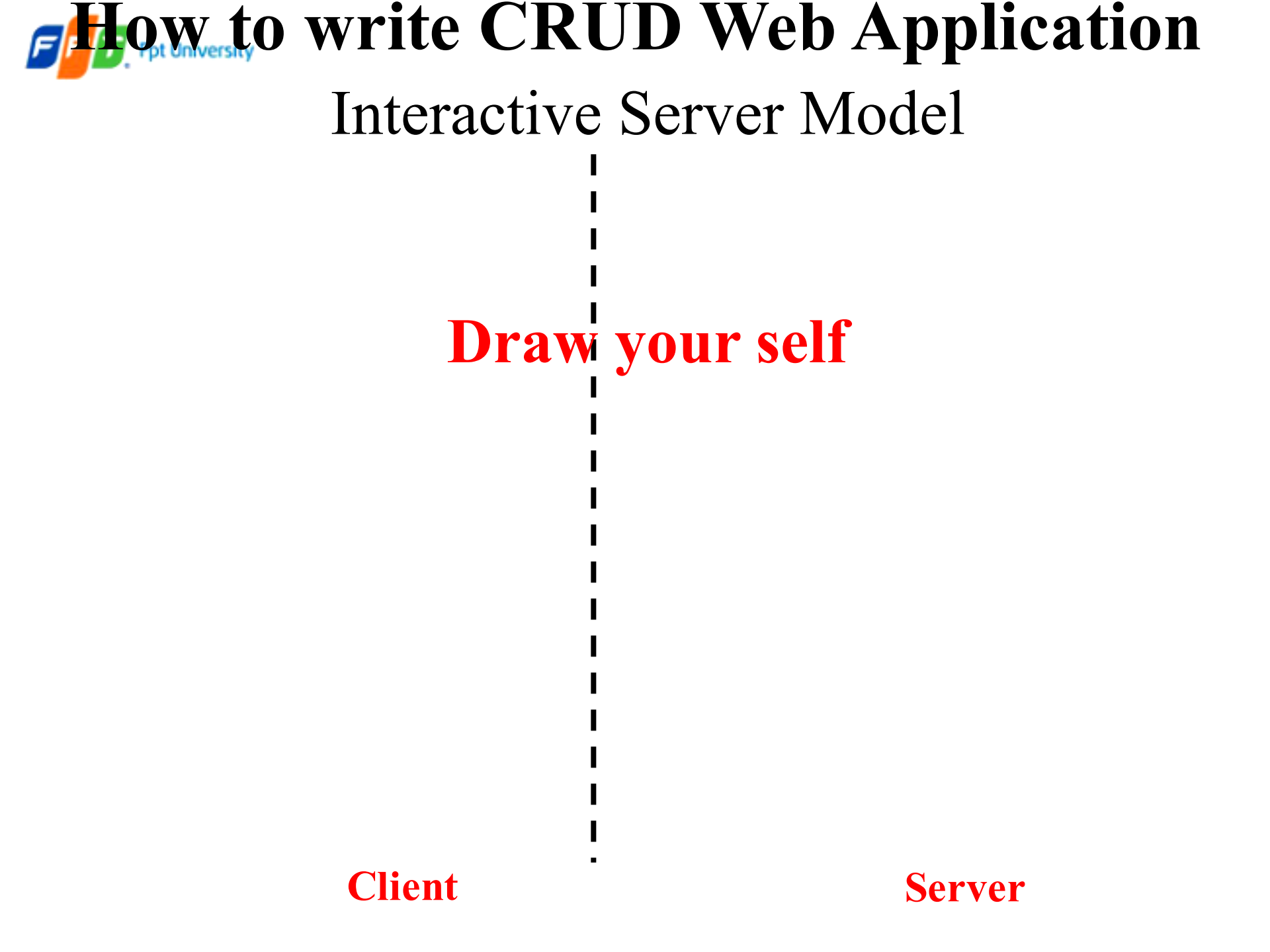
How to write CRUD Web Application

Store Info



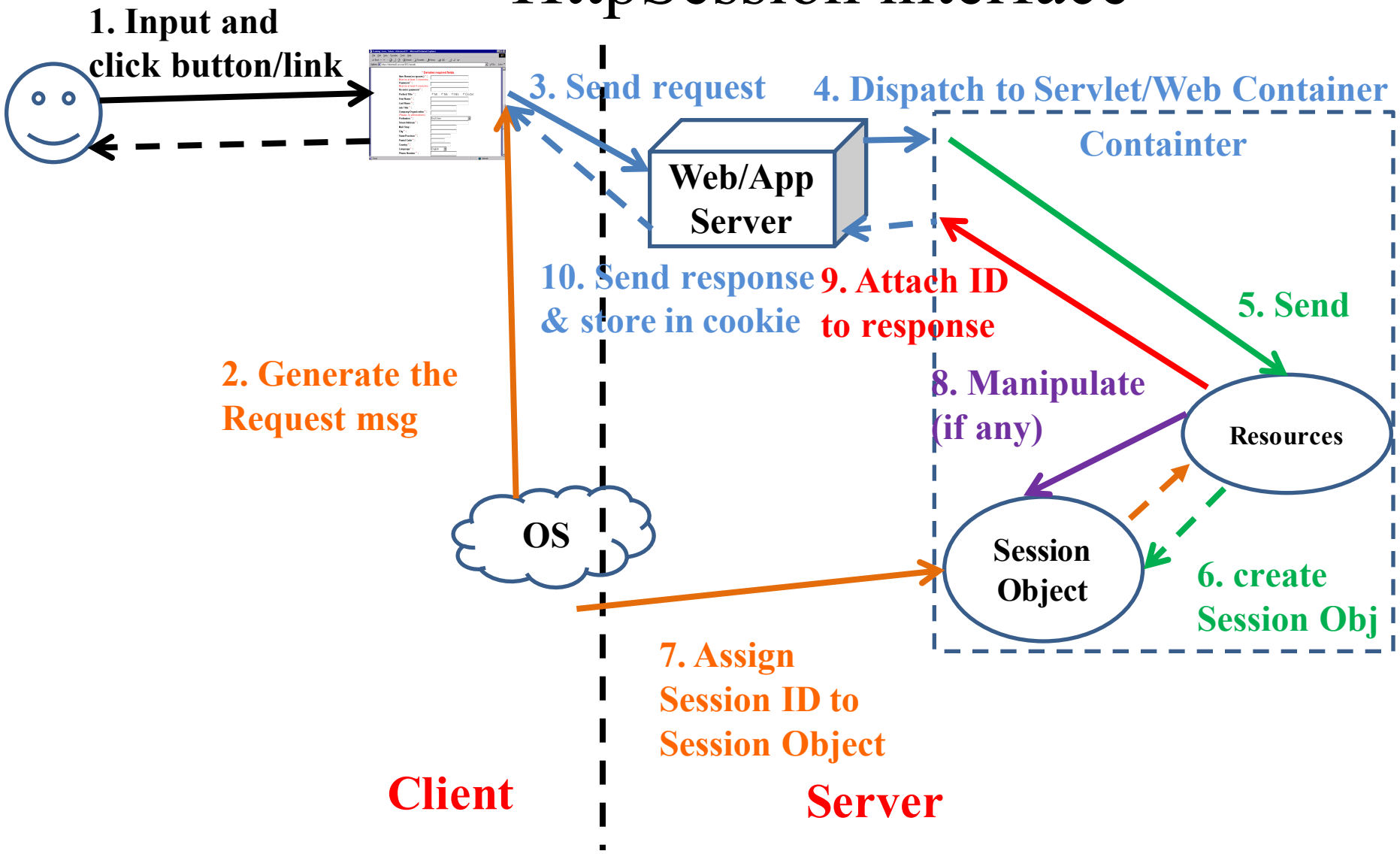
How to write CRUD Web Application

Interactive Server Model



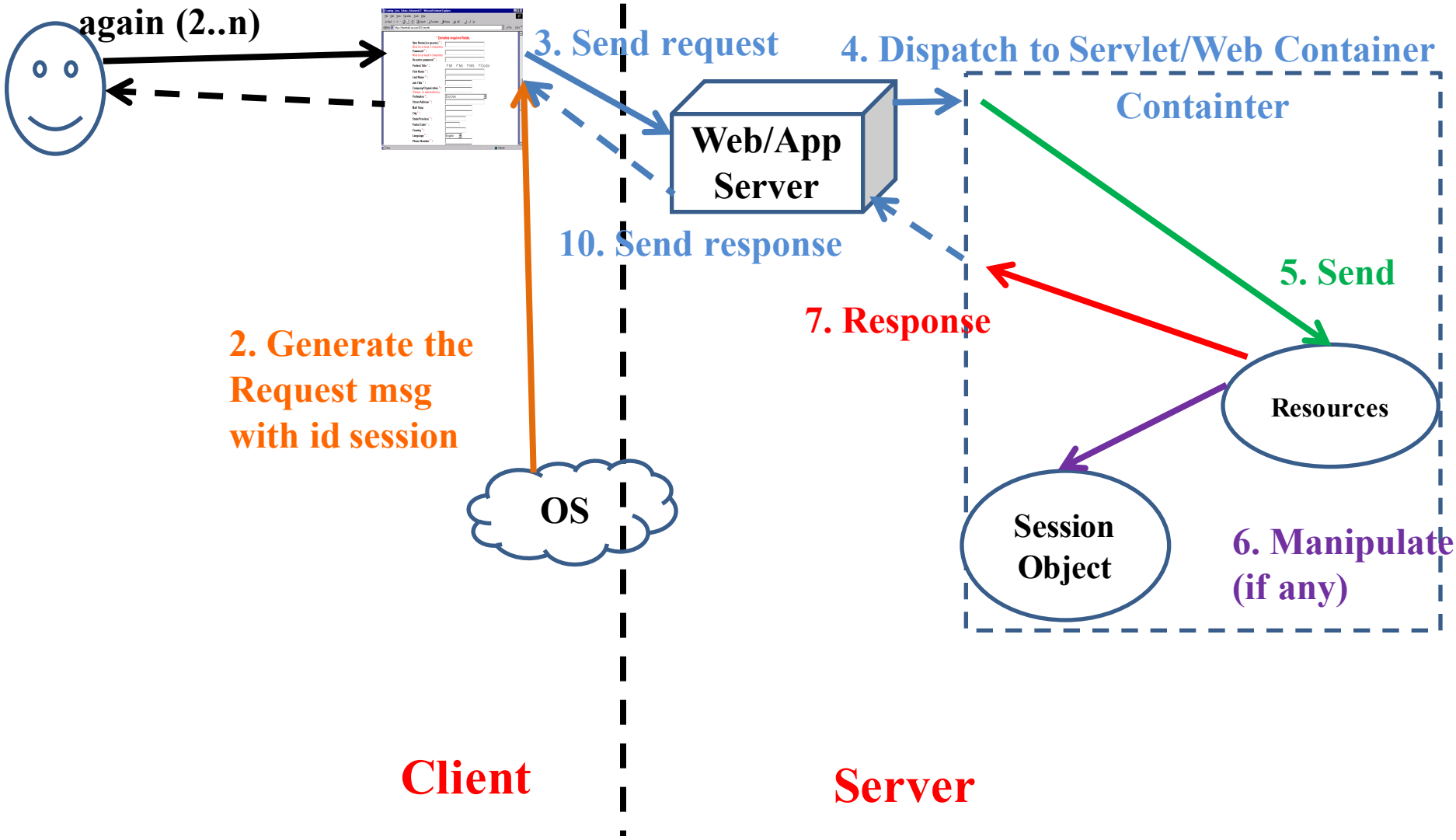
Sessions & Listeners

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**()

Fpt University

Sessions & Listeners
HttpSession interface Methods

Methods	Descriptions
getSession	- **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	- **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	- **public long getCreationTime()** - Returns the creation time of session.
getLastAccessedTime	- **public long getLastAccessedTime()** - Returns the last accessed Time of session
getMaxInactiveInterval	- **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	- **public void setMaxInactiveInterval(int interval)** - Specifies the time, in seconds, between the client requests before the servlet container invalidates the current session

Fpt University

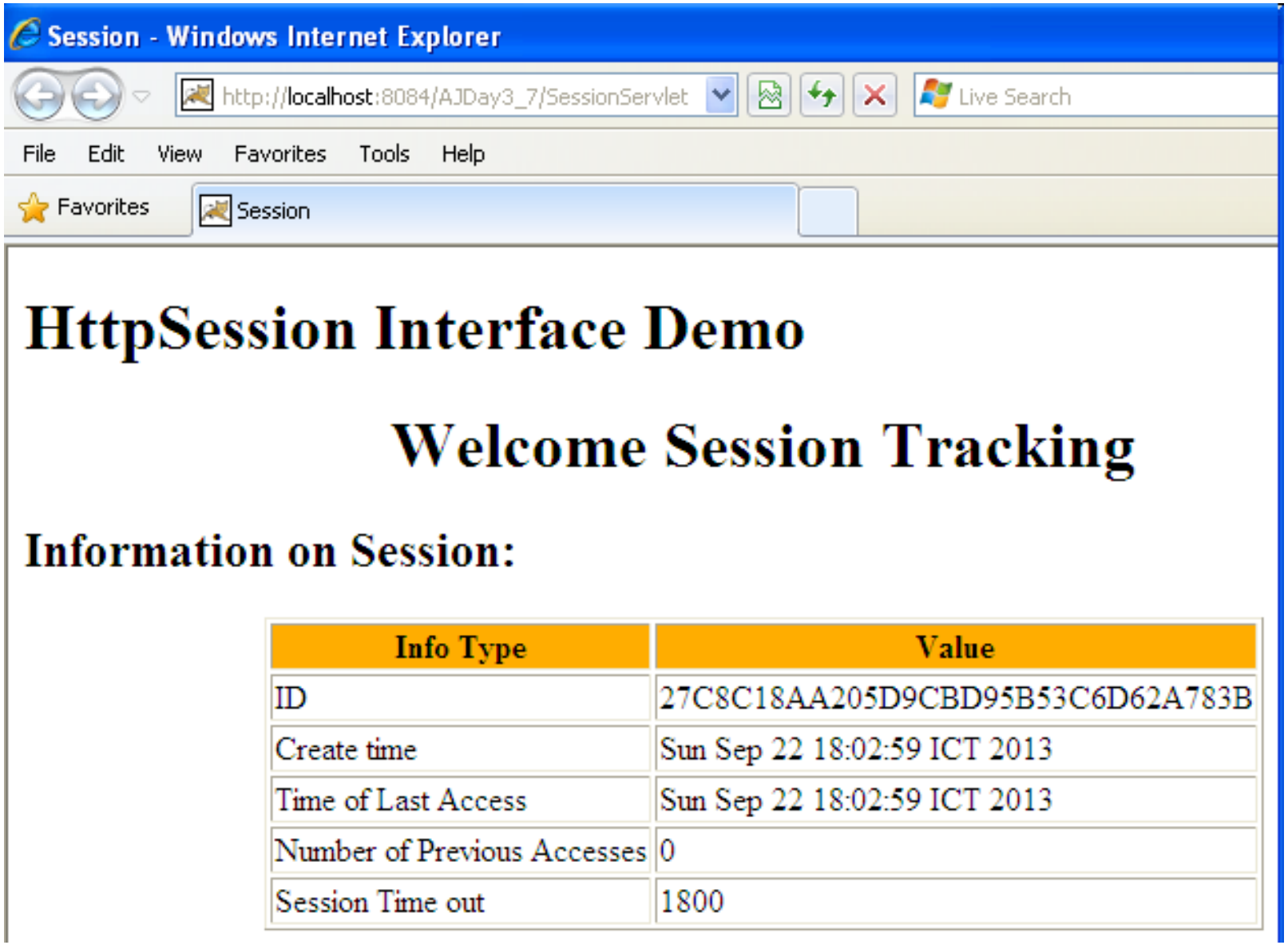
Sessions & Listeners

HttpSession interface Methods

Methods	Descriptions
isNew	- **public boolean isNew()** - Returns true if the client is unaware about the session or choose not to be part of the session
invalidate	- **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



Sessions & Listeners
HttpSession interface – Example

SessionServlet.java

```
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              out.println("<html>");
36              out.println("<head>");
37              out.println("<title>Session</title>");
38              out.println("</head>");
39              out.println("<body>");
40              out.println("<h1>HttpSession Interface Demo</h1>");
41
42              HttpSession session = request.getSession(true);
43              String heading;
44              Integer accessCount = (Integer) session.getAttribute("accessCount");
45              if (accessCount == null) {
46                  accessCount = new Integer(0);
47                  heading = "Welcome Session Tracking";
48              } else {
49                  heading = "Comeback";
50                  accessCount = new Integer(accessCount.intValue() + 1);
51              }
```

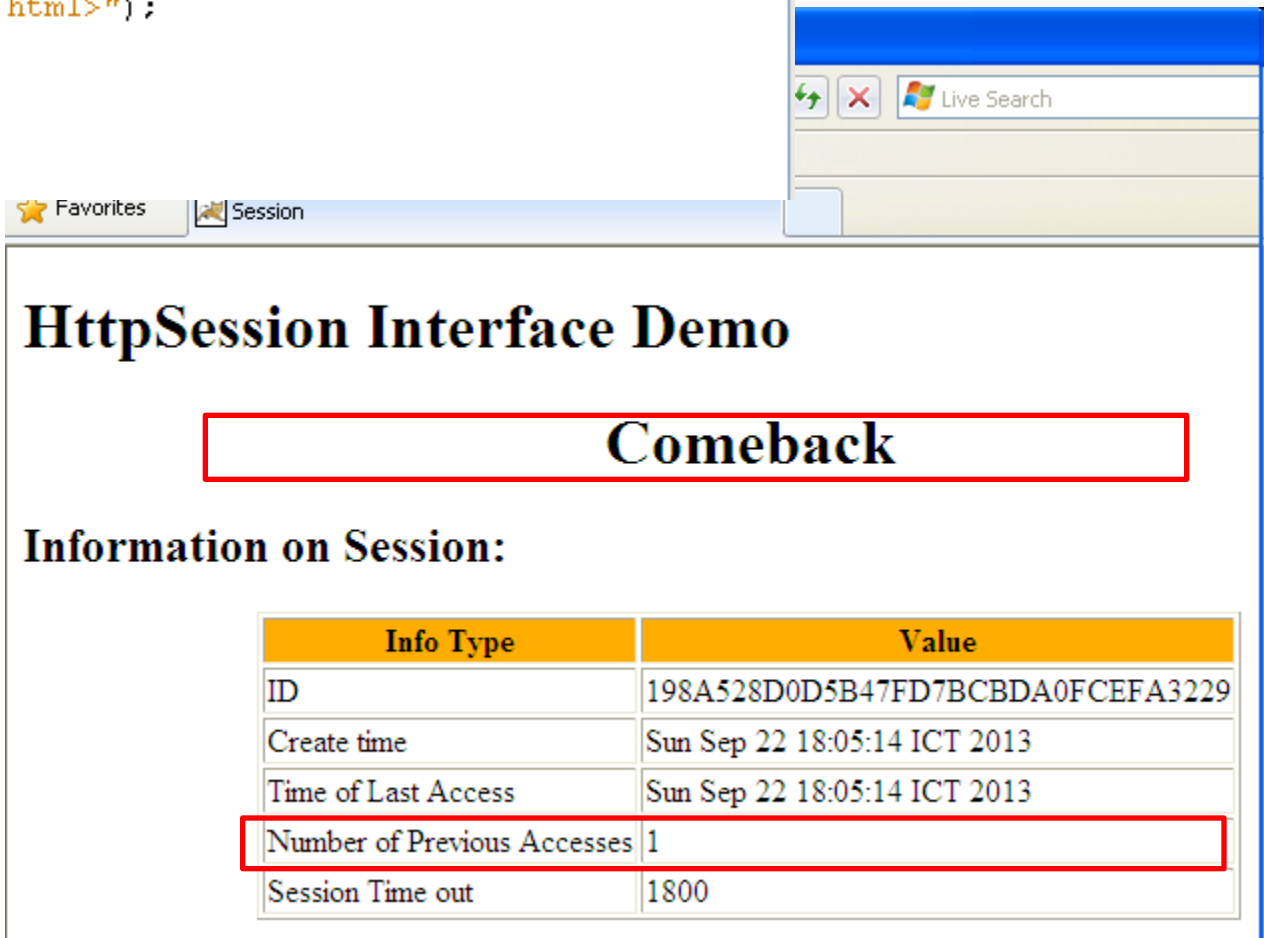
Sessions & Listeners
HttpSession interface – Example

```
        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="
                + "\"#FFADOO\">\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

SessionServlet.java

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



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

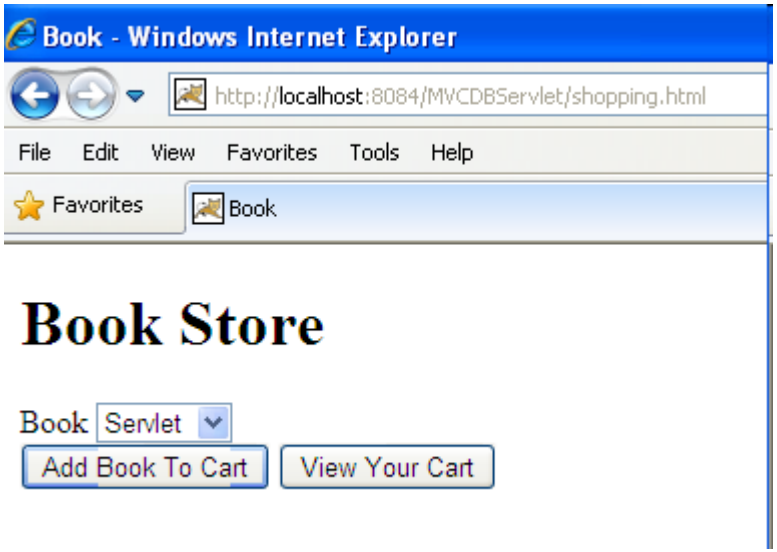
  - Individual Session Setting using **setMaxInactivateInterval**() method
    - A negative value supplied as an argument causes the session to never expire
- Other Session APIs
  - **HttpSession.getServletContext**() returns the SessionContext that the session is attached

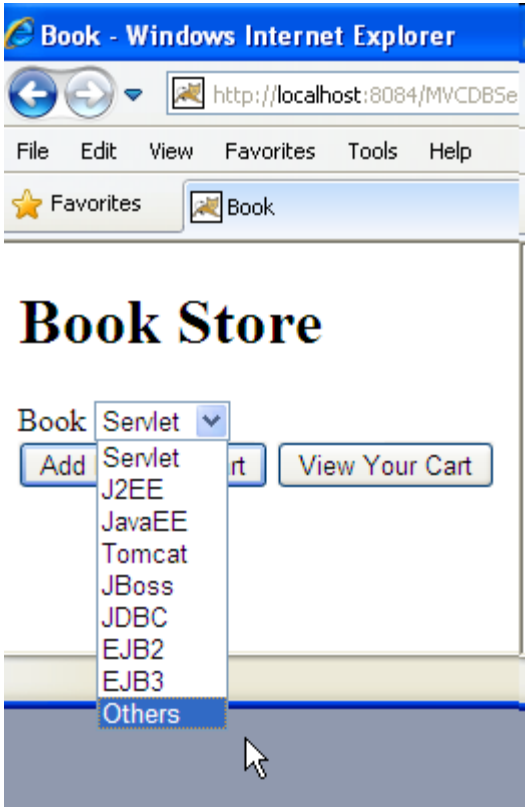
How to write CRUD Web Application

Shopping Cart

How to write CRUD Web Application

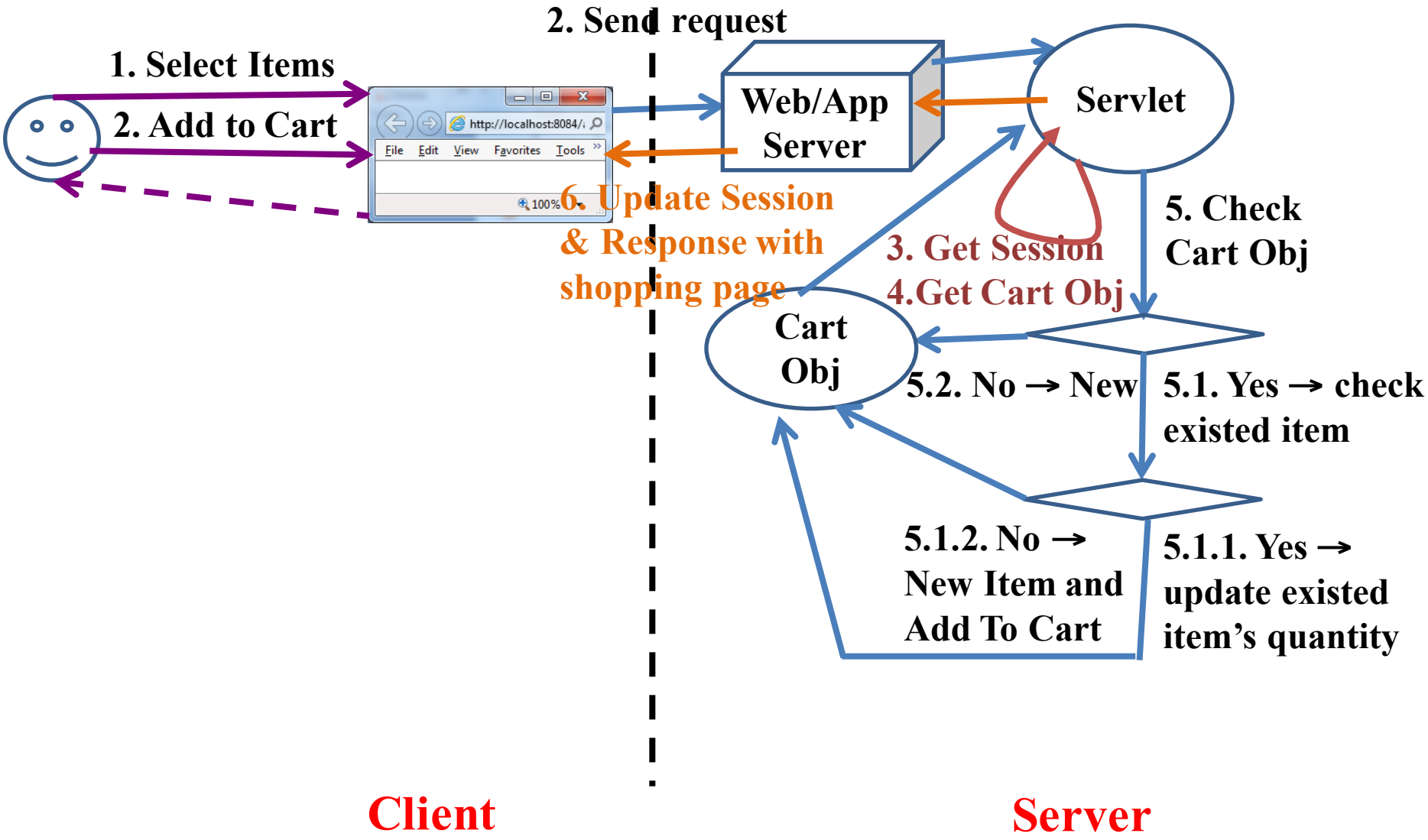
Shopping Cart – Add To Cart





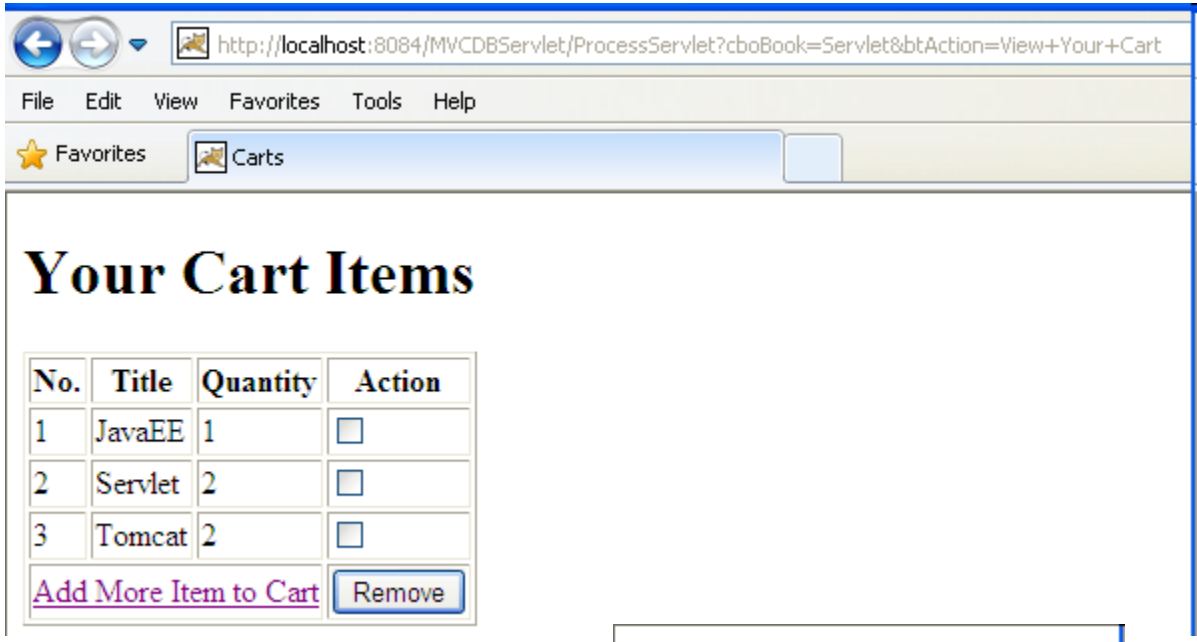
How to write CRUD Web Application

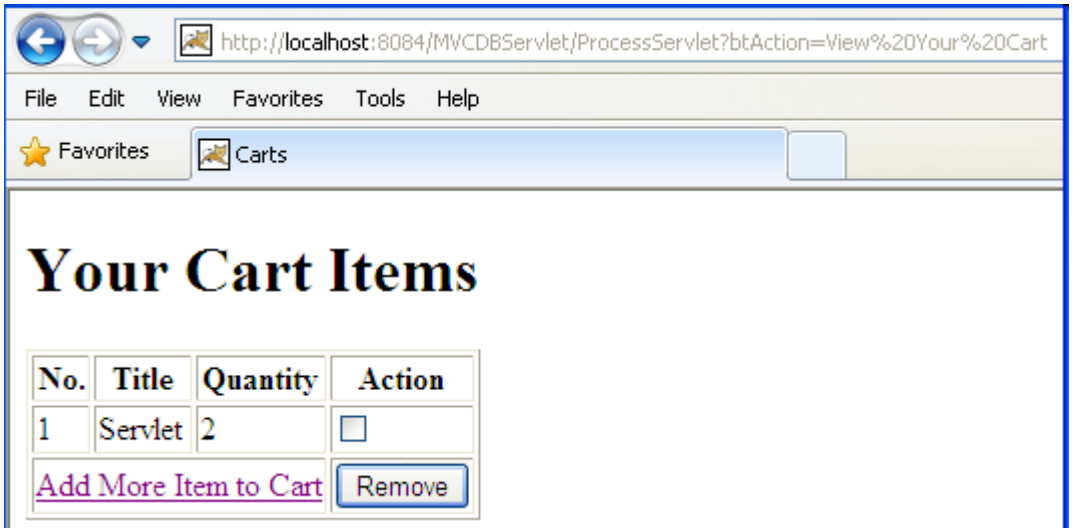
Interactive Server Model – Add To Cart

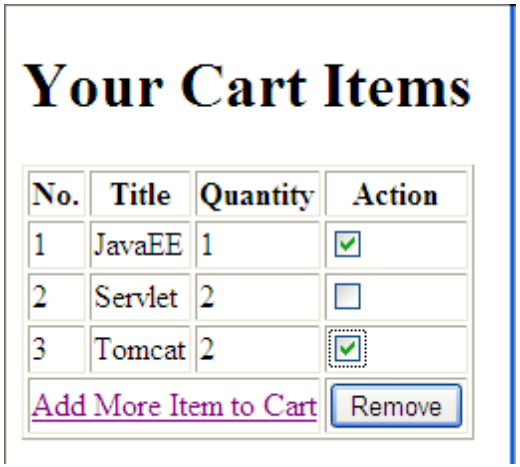


How to write CRUD Web Application

Shopping Cart – View Cart

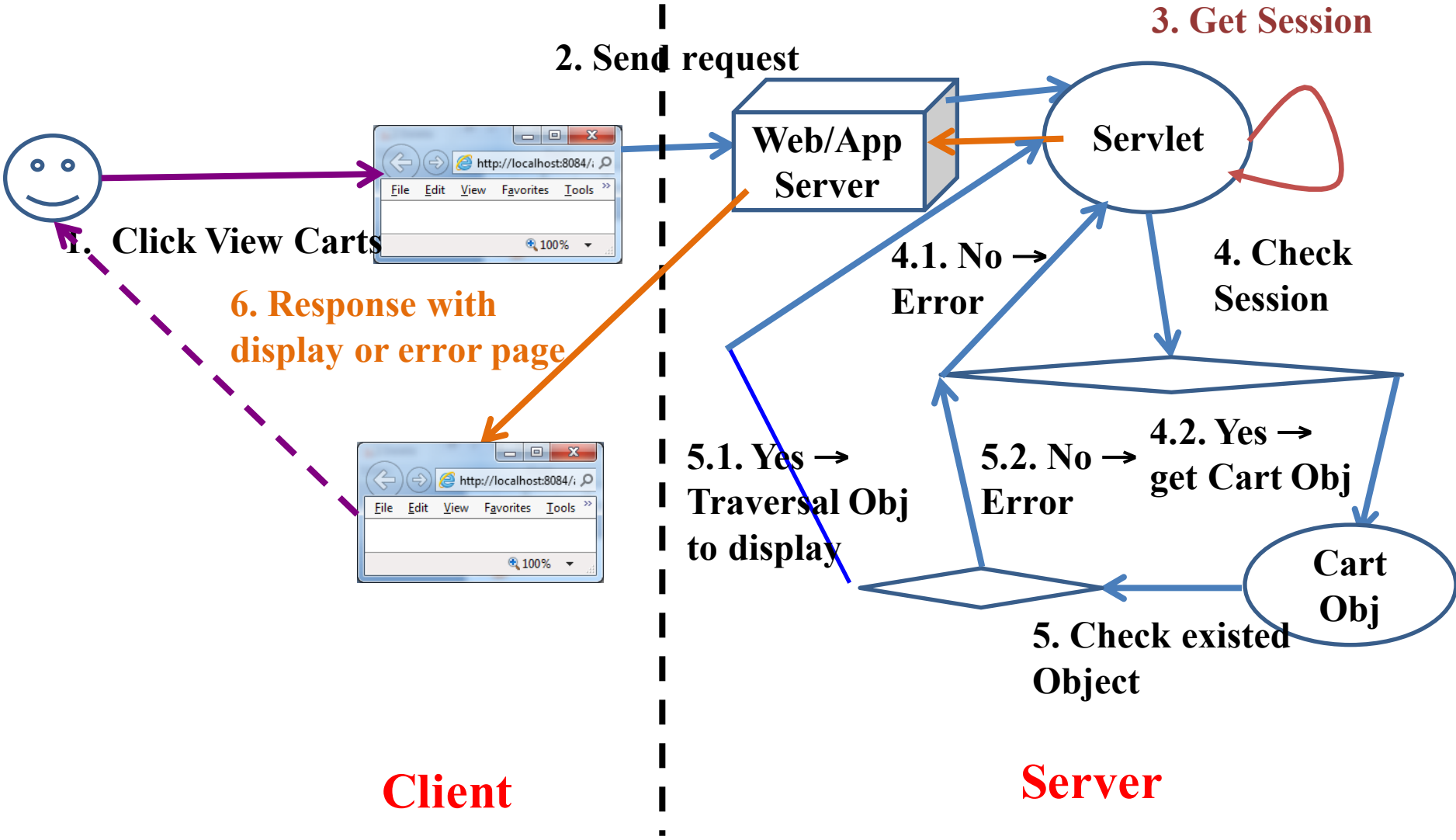






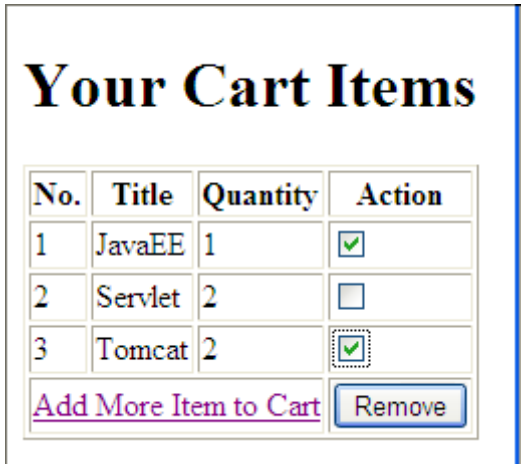
How to write CRUD Web Application

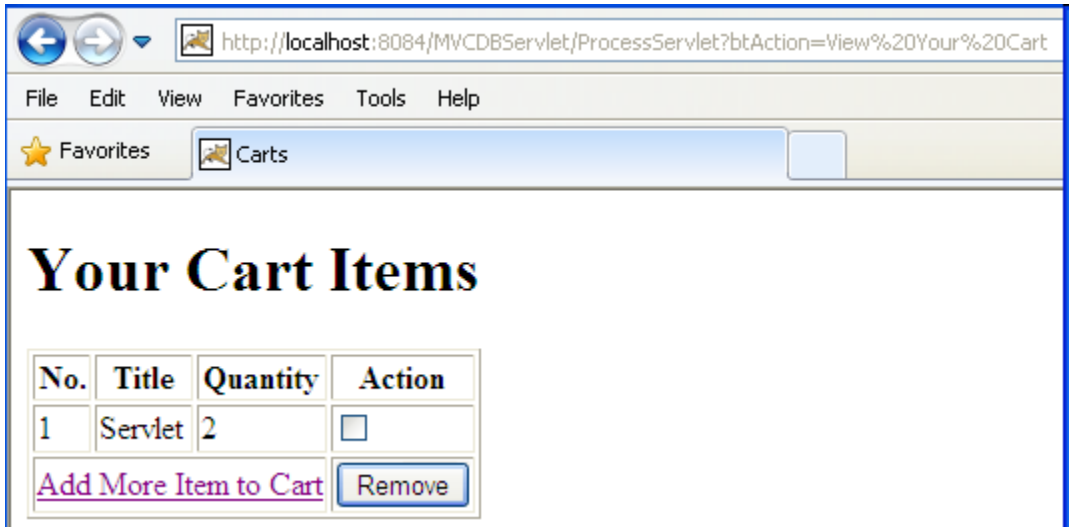
Interactive Server Model – View Cart



How to write CRUD Web Application

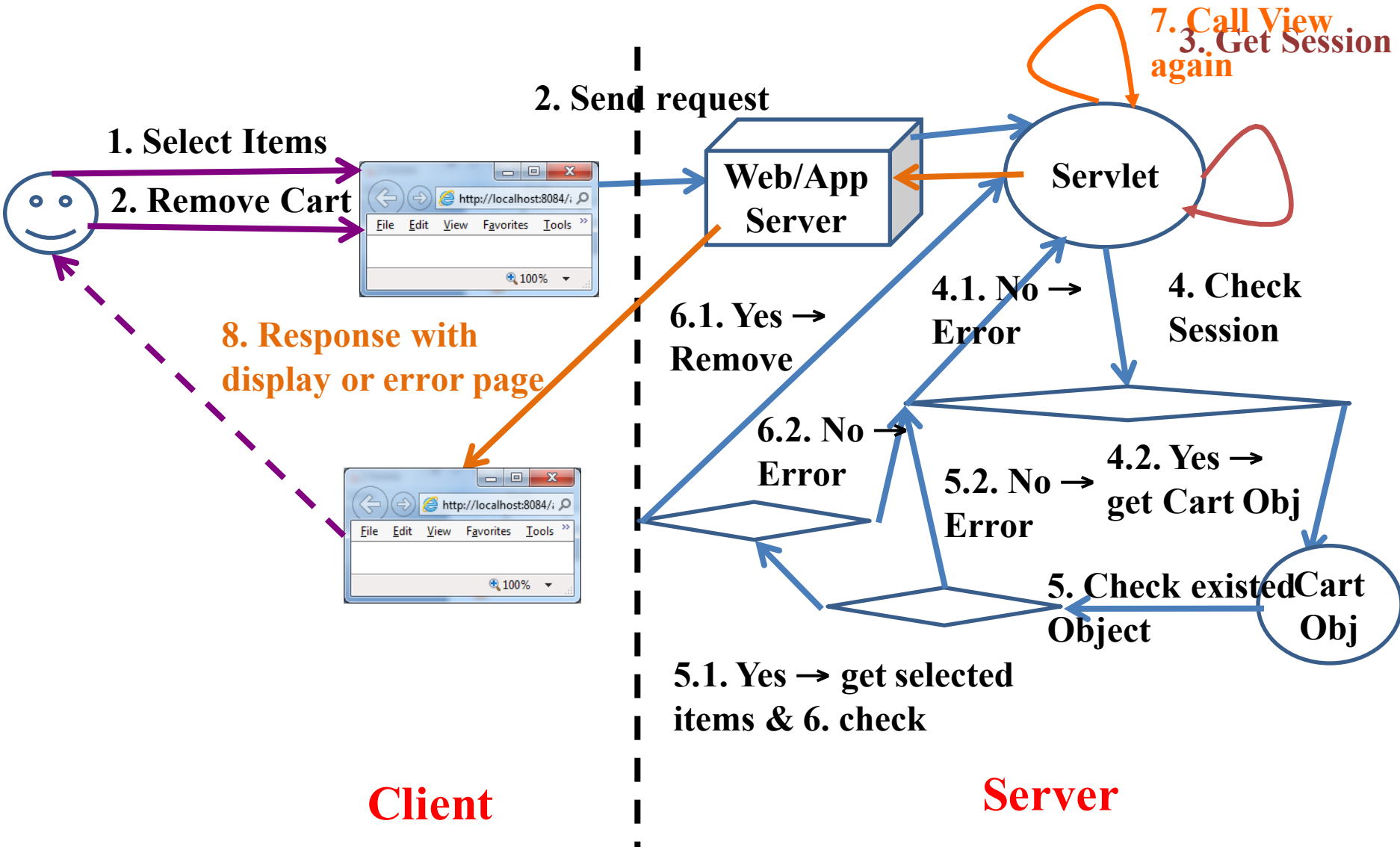
Shopping Cart – Remove Cart





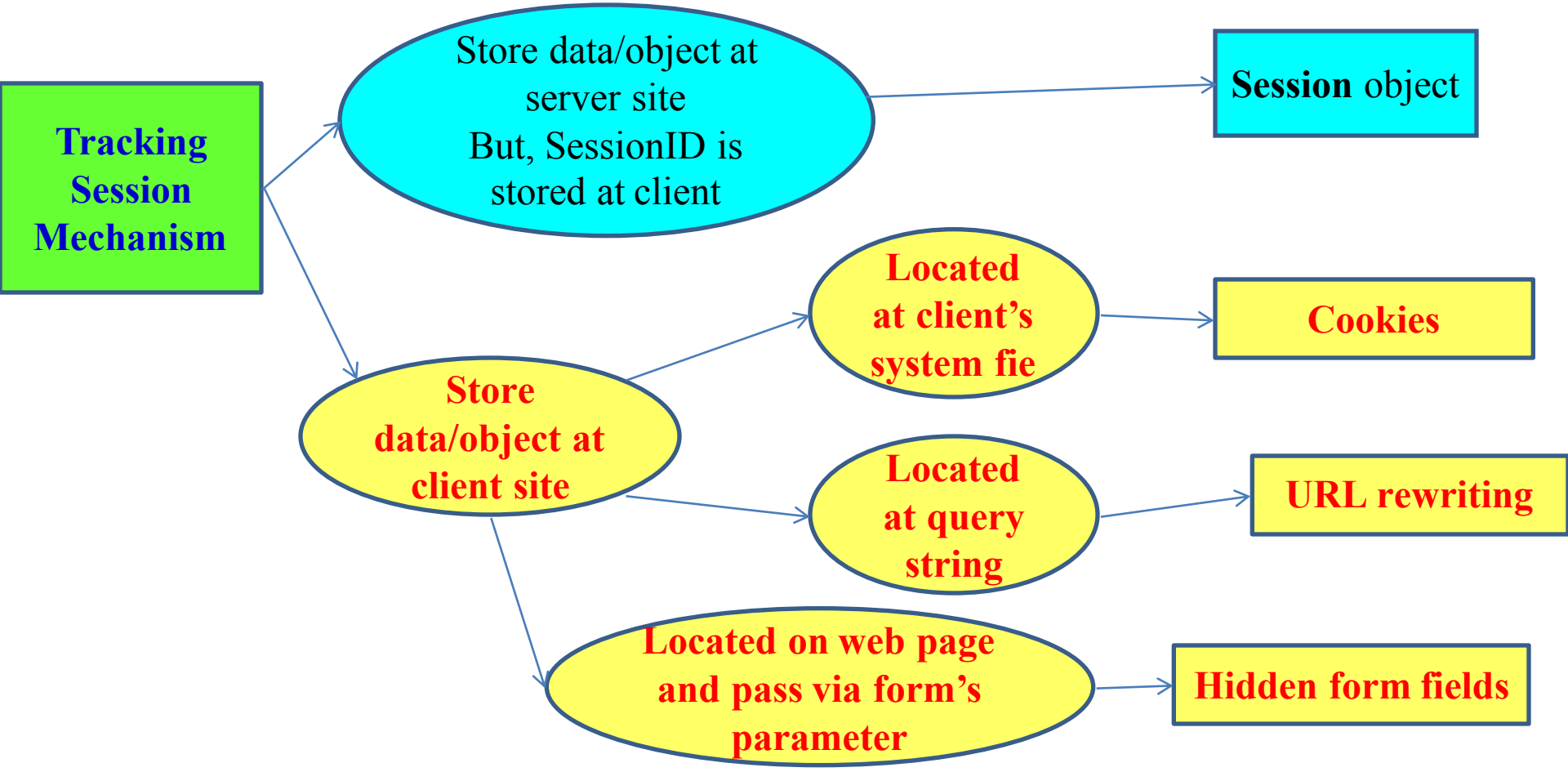
How to write CRUD Web Application

Interactive Server Model – Remove



Sessions & Listeners

Conclusion



Error Handling in Servlet

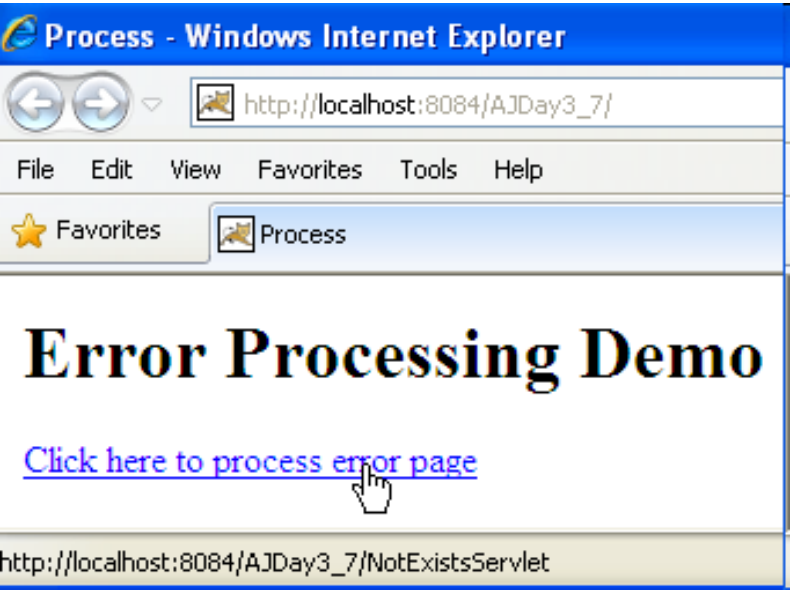
Reporting Error

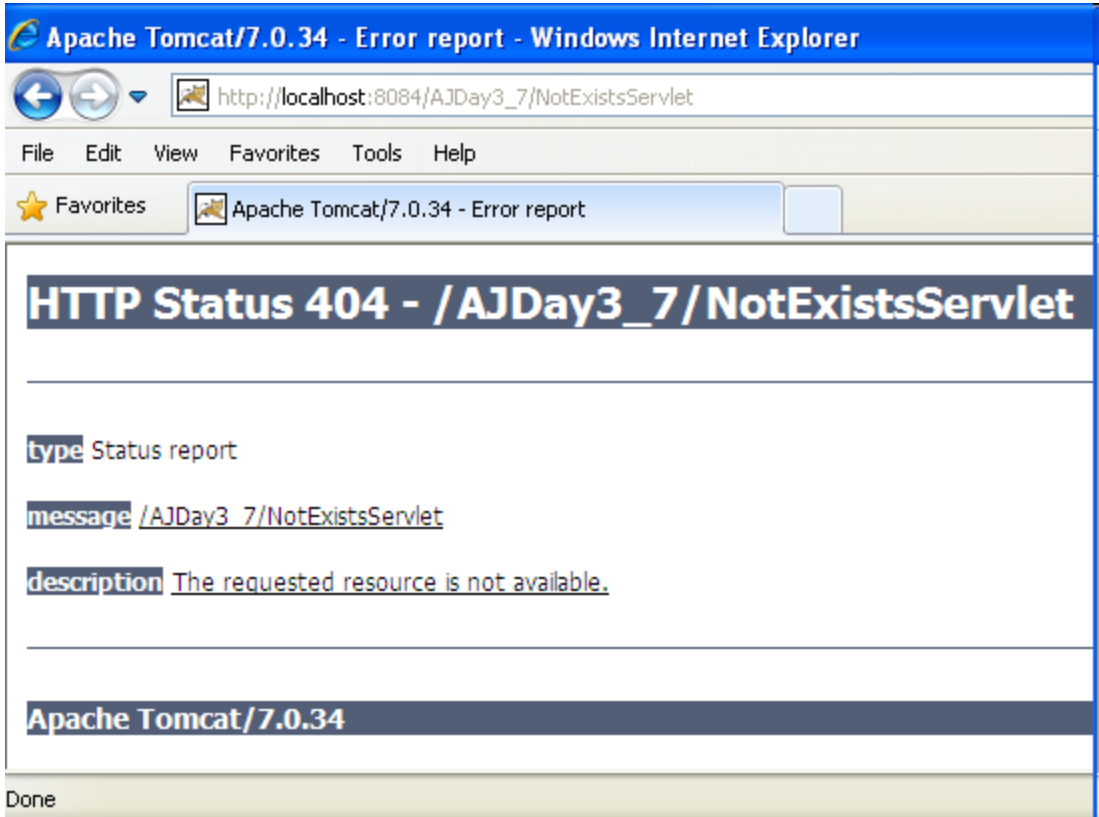
- **There are many situations** occur an error
  - A **requested page** may be **moved from one** location **to another**.
  - The **address** may be **wrongly typed.**
  - The requested page may be **forbidden**, may be **temporarily deleted** or **correct** HTTP version might **not have found**.
 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

Methods	Descriptions
sendError	- **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	- **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

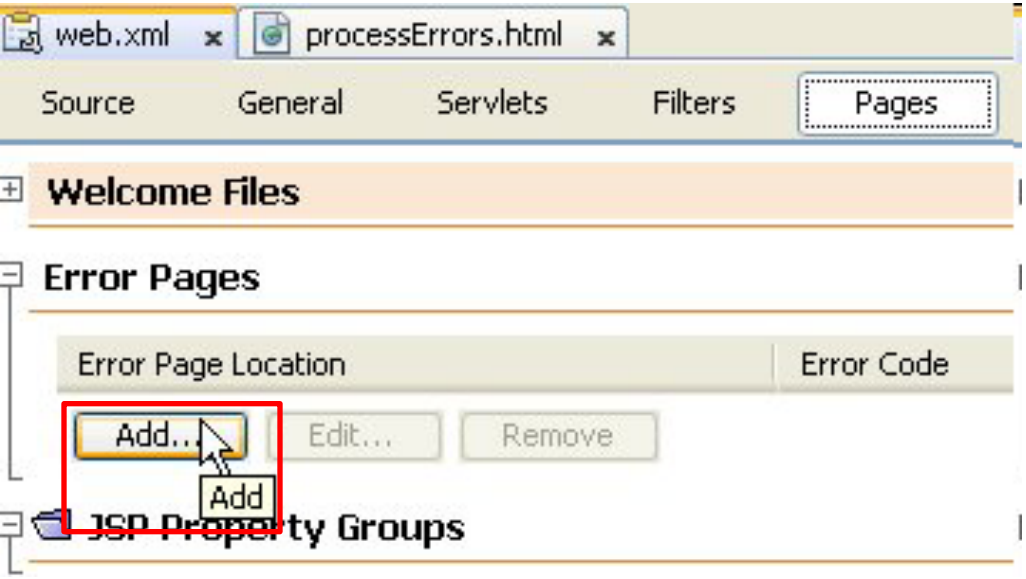
Reporting Error – Example

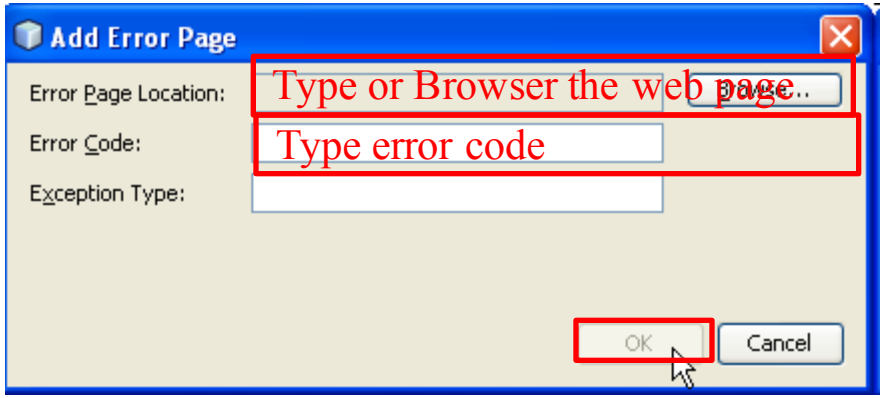


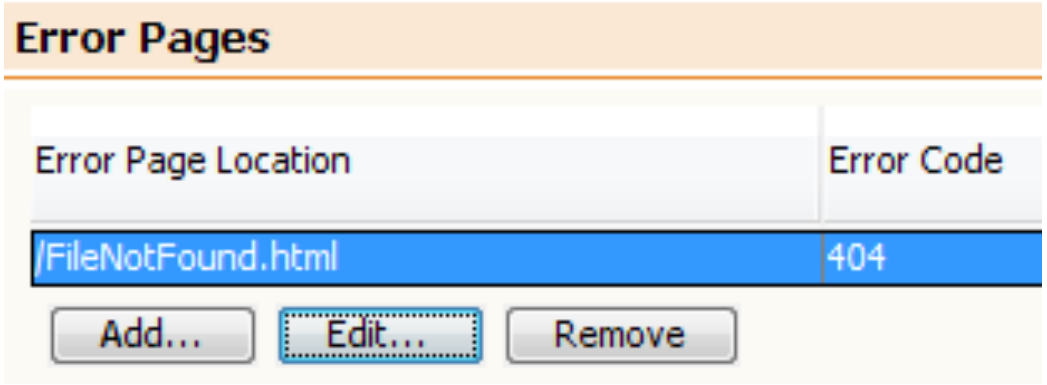


Error Handling in Servlet
Reporting Error – Example

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







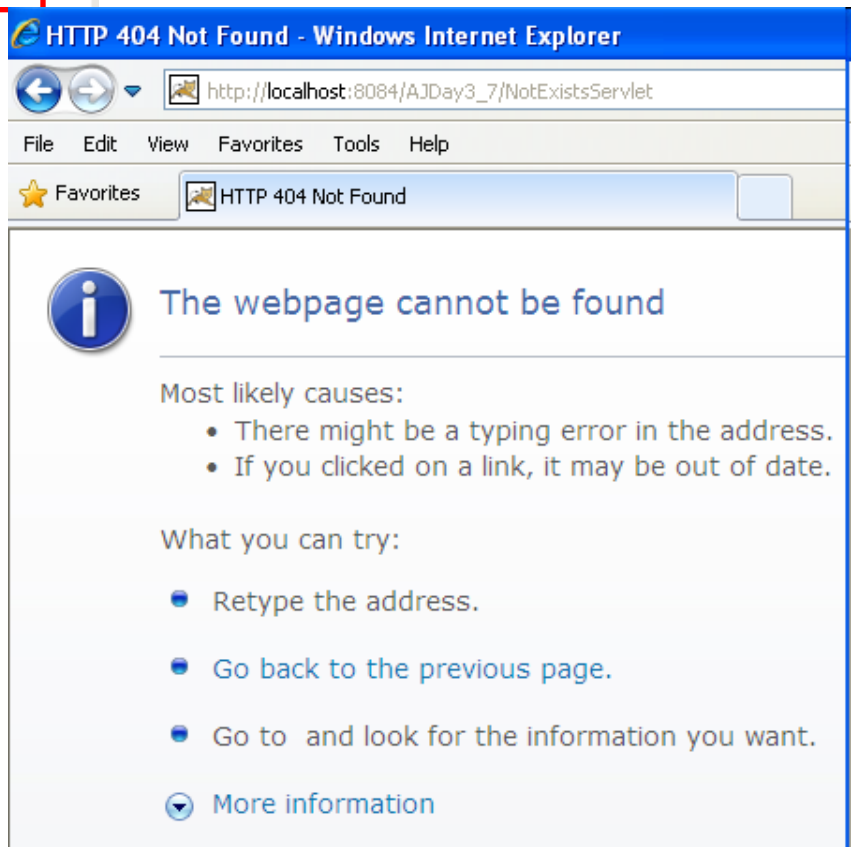
Error Handling in Servlet

Reporting Error – Example

web.xml

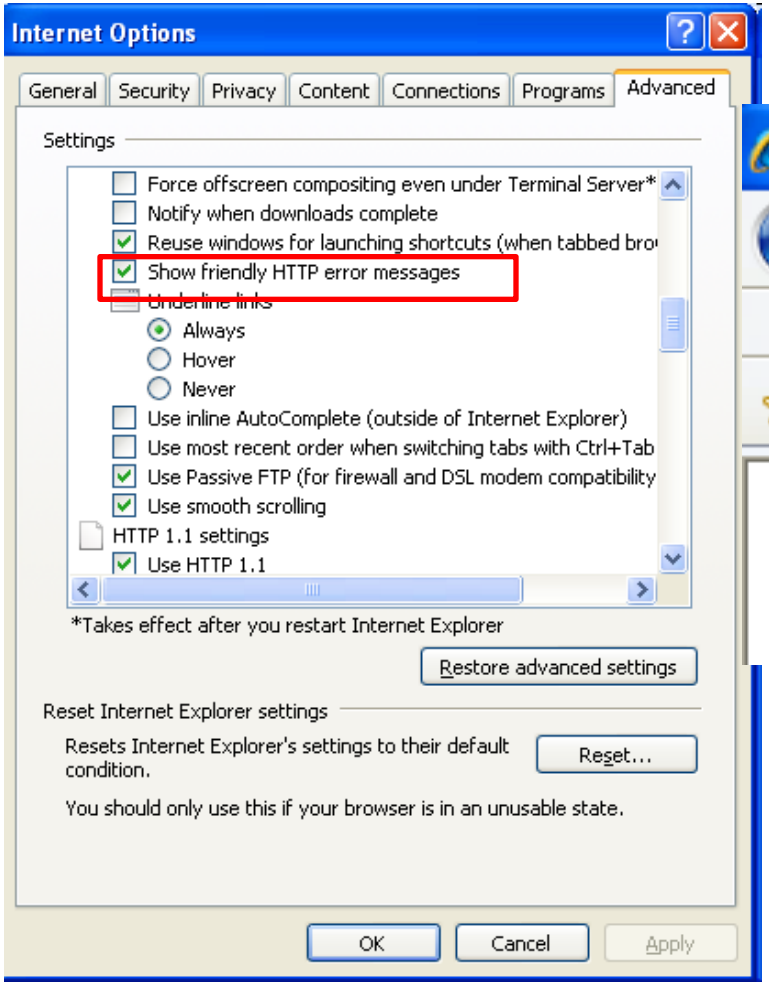
Source | General | Servlets | Filters | Pages | References | Secu

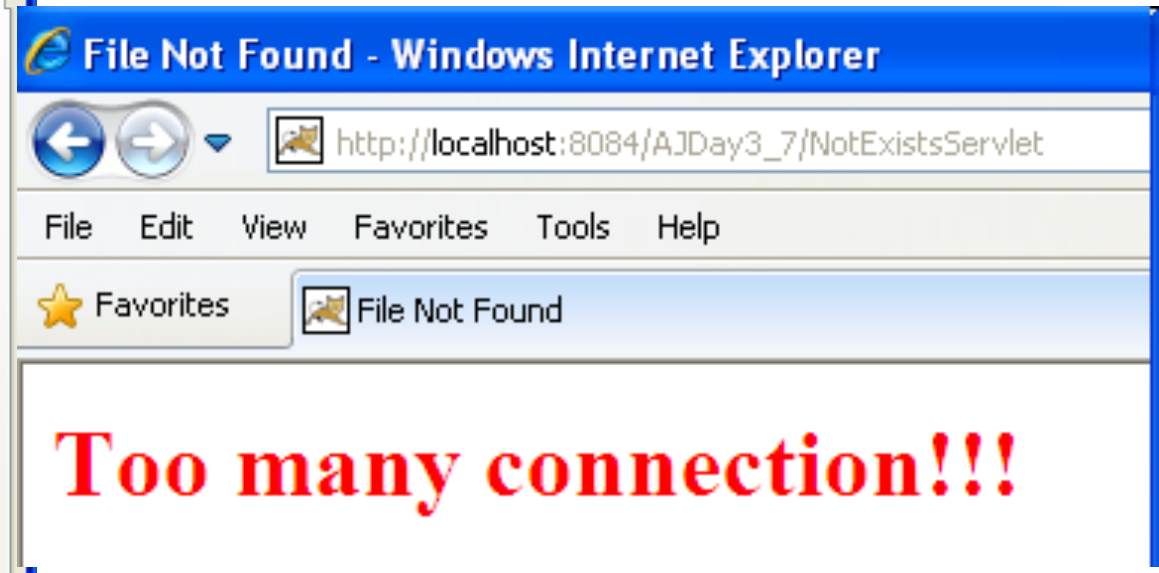
```
172     <welcome-file-list>
173         <welcome-file>logine.html</welcome-file>
174     </welcome-file-list>
175     <error-page>
176         <error-code>404</error-code>
177         <location>/fileNotFound.html</location>
178     </error-page>
179 </web-app>
```



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

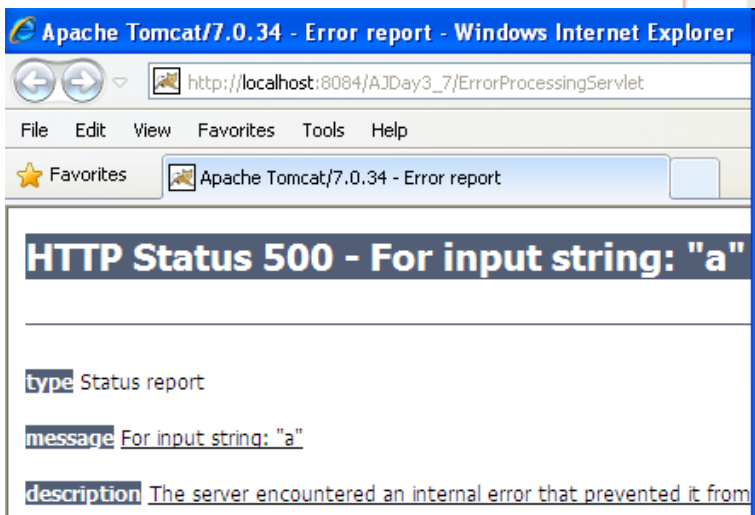
Error Handling in Servlet

Reporting Error – Example

ErrorProcessingServlet.java

Source | History

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



Error Handling in Servlet

Reporting Error – Example

ErrorProcessingServlet.java | ProcessErrServlet.java

Source | History

```
17      * @author Trong Khanh
18      */
19     public class ErrorProcessingServlet extends HttpServlet {
20
21         /**...*/
31         protected void processRequest(HttpServletRequest request, HttpServletResponse resp
32                 throws ServletException, IOException {
33             response.setContentType("text/html;charset=UTF-8");
34             PrintWriter out = response.getWriter();
35             try {
36                 int a = Integer.parseInt("a");
37             } catch (NumberFormatException e) {
38                 request.setAttribute("ERRORS", e.getMessage());
39                 RequestDispatcher rd = request.getRequestDispatcher("ProcessErrServlet");
                   rd.forward(request, response);
41             } finally {
42                 out.close();
43             }
44     }
```

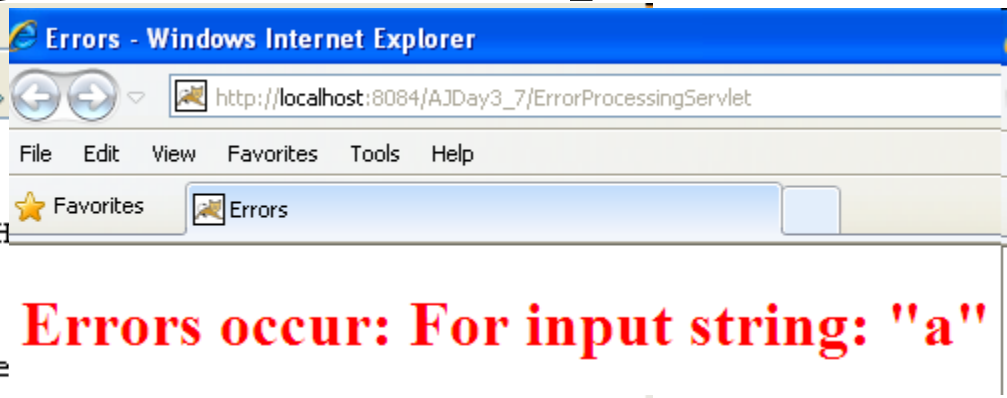
Error Handling in Servlet

Reporting Error – Example

ProcessErrServlet.java

Source | History

```
16      * @author Trong Khanh
17      */
18     public class ProcessErrServlet extends H
19
20         /**...*/
30         protected void processRequest(HttpSe
31                 throws ServletException, IOException {
32             response.setContentType("text/html;charset=UTF-8");
33             PrintWriter out = response.getWriter();
34             try {
35                 /* TODO output your page here. You may use following sample
36                 out.println("<!DOCTYPE html>");
37                 out.println("<html>");
38                 out.println("<head>");
39                 out.println("<title>Errors</title>");
40                 out.println("</head>");
41                 out.println("<body>");
42                 out.println("<h1><font color='red'>Errors occur: "
43                         + request.getAttribute("ERRORS") + "</font></h1>");
44                 out.println("</body>");
45                 out.println("</html>");
46             } finally {
47                 out.close();
48             }
49         }
```



Error Handling in Servlet

Reporting Error – Example

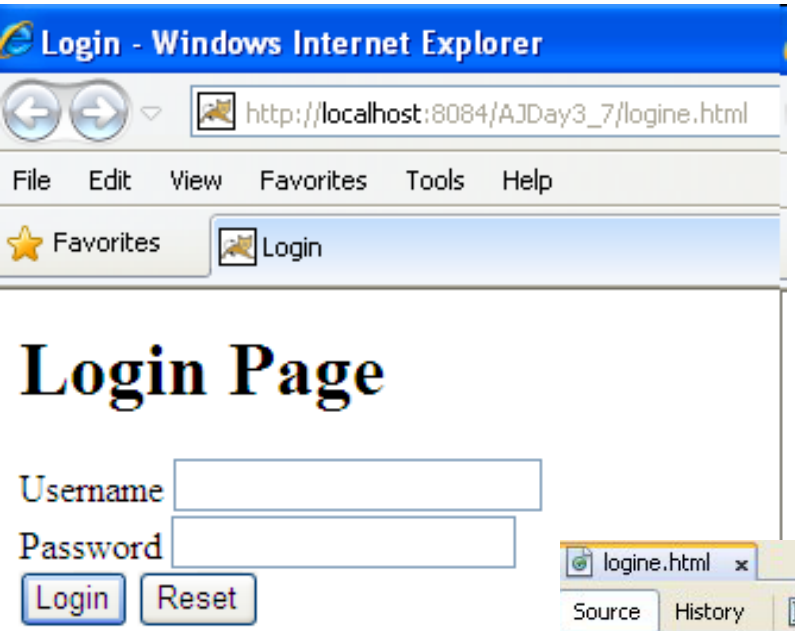
DisplayErrorServlet.java

Source | History

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

Error Handling in Servlet

Reporting Error – Example



logine.html

Source | History

```
5   <!DOCTYPE html>
6   <html>
7       <head>
8           <title>Login</title>
9           <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10      </head>
11      <body>
12          <h1>Login Page</h1>
13          <form action="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

DisplayErrorServlet.java

Source | History

```
16   * @author Trong Khanh
17   */
18  public class DisplayErrorServlet extends HttpServlet {
19
20      /**...*/
30      protected void processRequest(HttpServletRequest request, HttpServletRespo
31              throws ServletException, IOException {
32          response.setContentType("text/html;charset=UTF-8");
33          PrintWriter out = response.getWriter();
34          try {
35              String action = request.getParameter("action");
36              if (action == null) {
37                  response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);
38                  int pos = request.getRequestURL().lastIndexOf("/");
39                  String url = request.getRequestURL().substring(0, pos);
40                  url = url + "/logine.html";
41                  System.out.println("ddd " + url);
42                  response.setHeader("Location", url);
43              }
44          } finally {
45              out.close();
46          }
47      }
```

Error Handling in Servlet

Logging Error

LogServlet.java

Source | History

```
16      * @author Trong Khanh
17      */
18     public class LogServlet extends HttpServlet {
19         /**...*/
29         protected void processRequest(HttpServletRequest request
30                 throws ServletException, IOException {
31             response.setContentType("text/html;charset=UTF-8");
32             PrintWriter out = response.getWriter();
33             try {
34                 /* TODO output your page here. You may use follo
35                 out.println("<!DOCTYPE html>");
36                 out.println("<html>");
37                 out.println("<head>");
38                 out.println("<title>Log</title>");
39                 out.println("</head>");
40                 out.println("<body>");
41                 out.println("<h1>Log Servlet Server Demo</h1>");
42                 try {
43                     int a = Integer.parseInt("aaaa");
44                 } catch (NumberFormatException ex) {
                       ex.printStackTrace();
46                 }
47                 out.println("</body>");
48                 out.println("</html>");
49             } finally {
50                 out.close();
51             }
52         }
```

Error Handling in Servlet

Logging Error

LogServlet.java

Source | History

```
16      * @author Trong Khanh
17      */
18     public class LogServlet extends HttpServlet {
19         /**...*/
29         protected void processRequest(HttpServletRequest reque
30                 throws ServletException, IOException {
31             response.setContentType("text/html;charset=UTF-8");
32             PrintWriter out = response.getWriter();
33             try {
34                 /* TODO output your page here. You may use follo
35                 out.println("<!DOCTYPE html>");
36                 out.println("<html>");
37                 out.println("<head>");
38                 out.println("<title>Log</title>");
39                 out.println("</head>");
40                 out.println("<body>");
41                 out.println("<h1>Log Servlet Server Demo</h1>");
42                 try {
43                     int a = Integer.parseInt("aaaa");
44                 } catch (NumberFormatException ex) {
                       ex.printStackTrace();
46                 }
47                 out.println("<
48                 out.println("<
49             } finally {
50                 out.close();
51             }
52         }
```

Log - Windows Internet Explorer

http://localhost:8084/AJDay3_7/LogServlet

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(NumberFormatE
        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

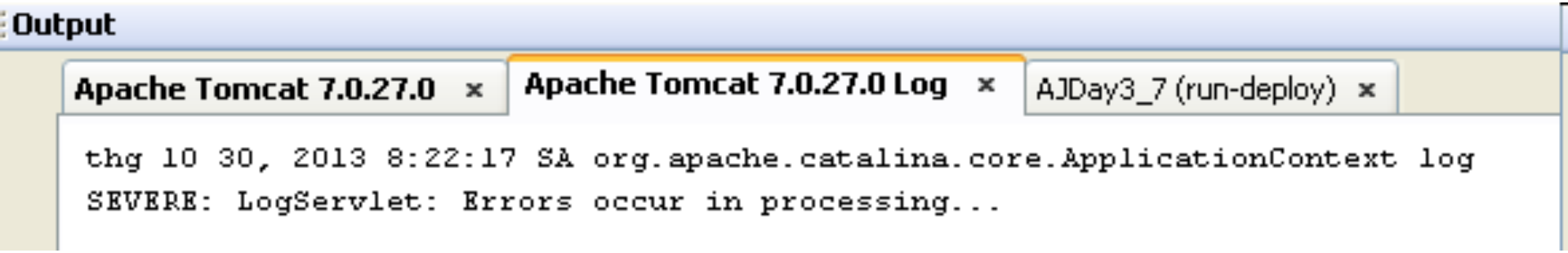
LogServlet.java

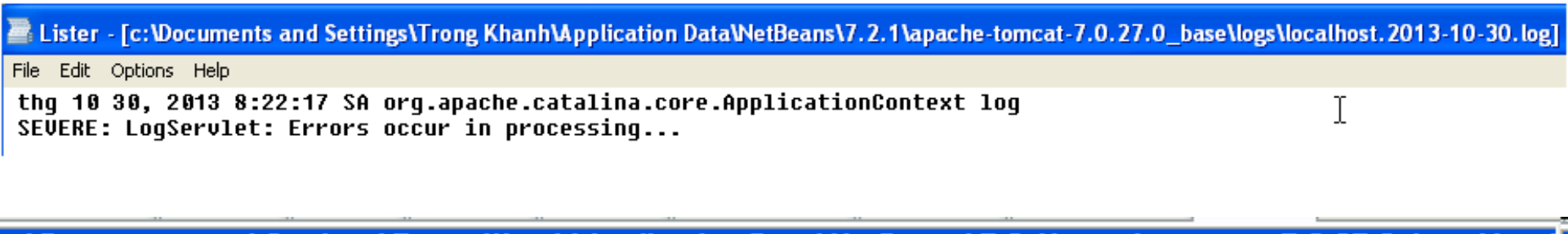
Source | History

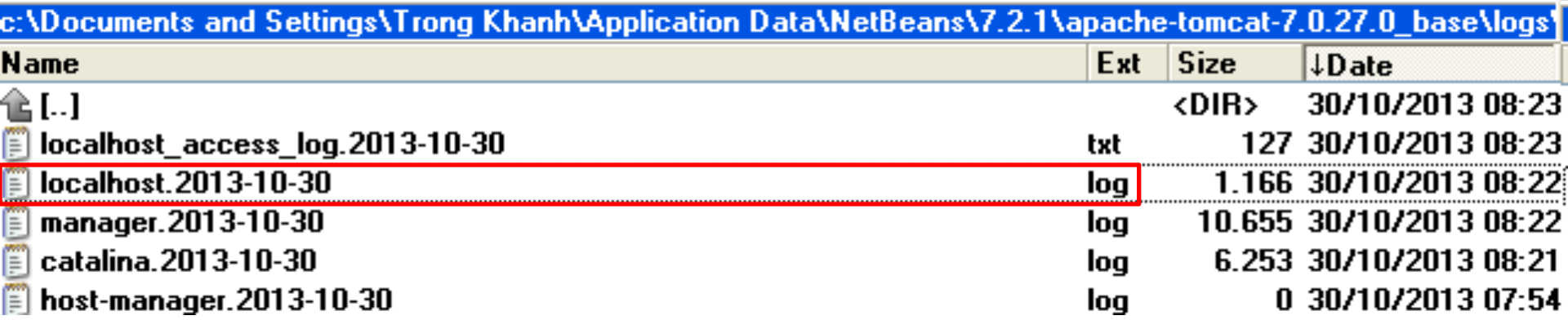
```
16      * @author Trong Khanh
17      */
18     public class LogServlet extends HttpServlet {
19         /**...*/
29         protected void processRequest(HttpServletRequest request, HttpSe
30                 throws ServletException, IOException {
31             response.setContentType("text/html;charset=UTF-8");
32             PrintWriter out = response.getWriter();
33             try {
34                 /* TODO output your page here. You may use following sam
35                 out.println("<!DOCTYPE html>");
36                 out.println("<html>");
37                 out.println("<head>");
38                 out.println("<title>Log</title>");
39                 out.println("</head>");
40                 out.println("<body>");
41                 out.println("<h1>Log Servlet Server Demo</h1>");
42                 try {
43                     int a = Integer.parseInt("aaaa");
44                 } catch (NumberFormatException ex) {
45                     log("Errors occur in processing...", ex.getCause());
46                 }
47                 out.println("</body>");
48                 out.println("</html>");
49             } finally {
50                 out.close();
51             }
52         }
```

Error Handling in Servlet

Logging Error

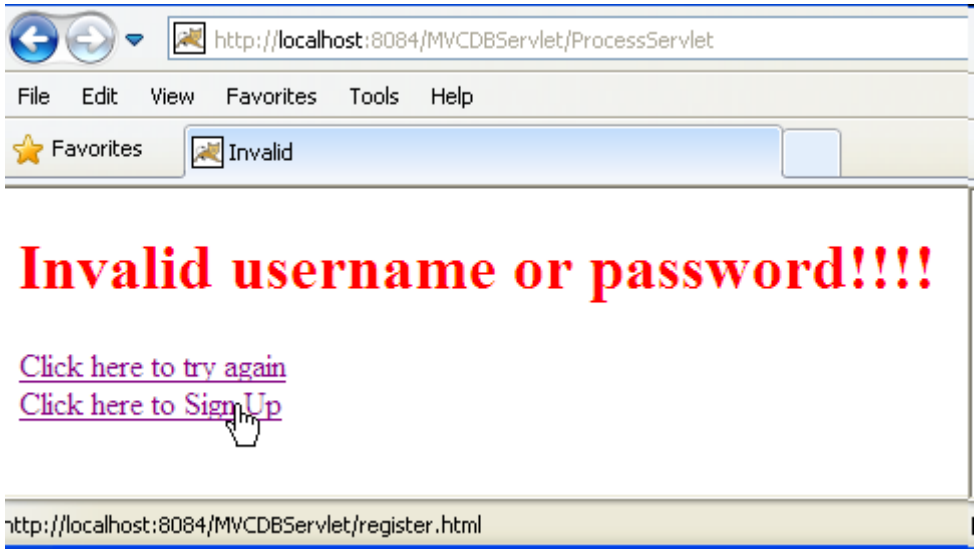


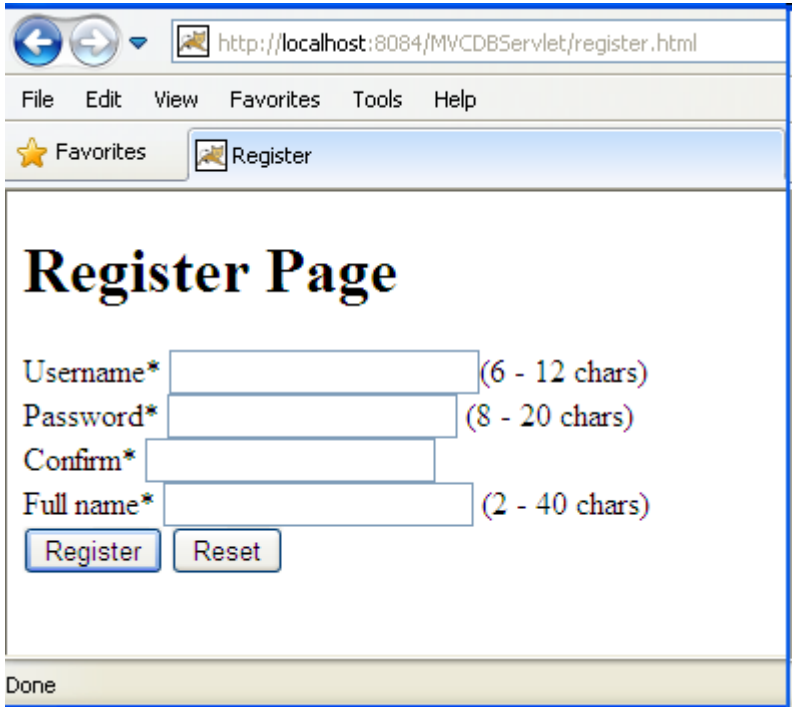




How to write CRUD Web Application

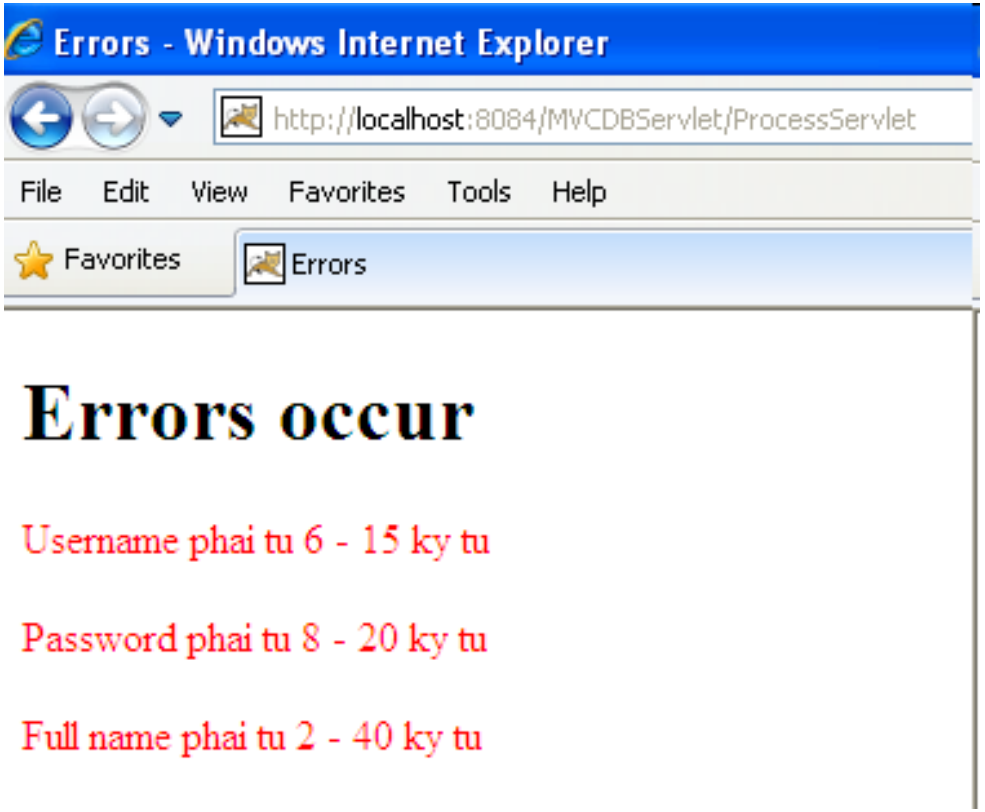
Register Functions

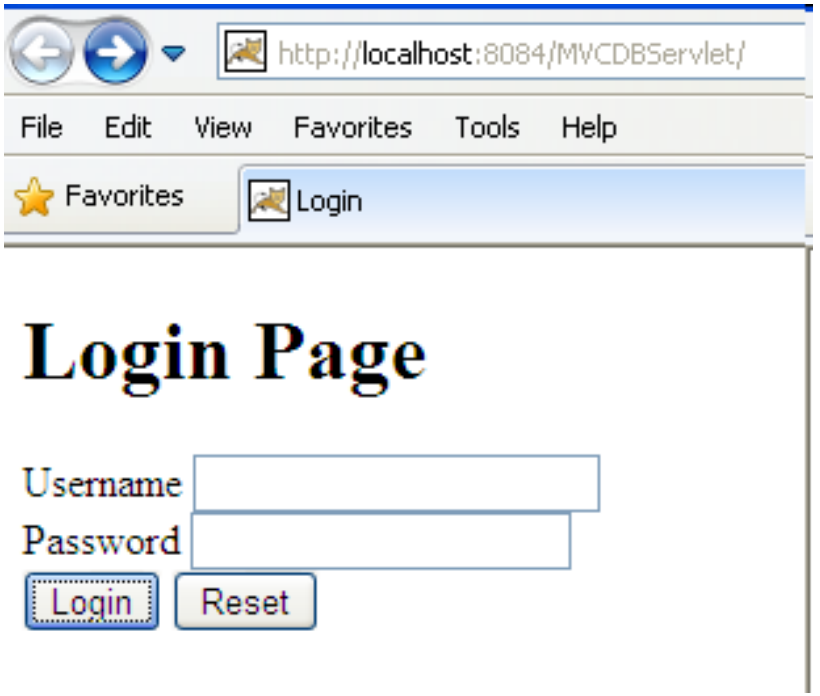




How to write CRUD Web Application

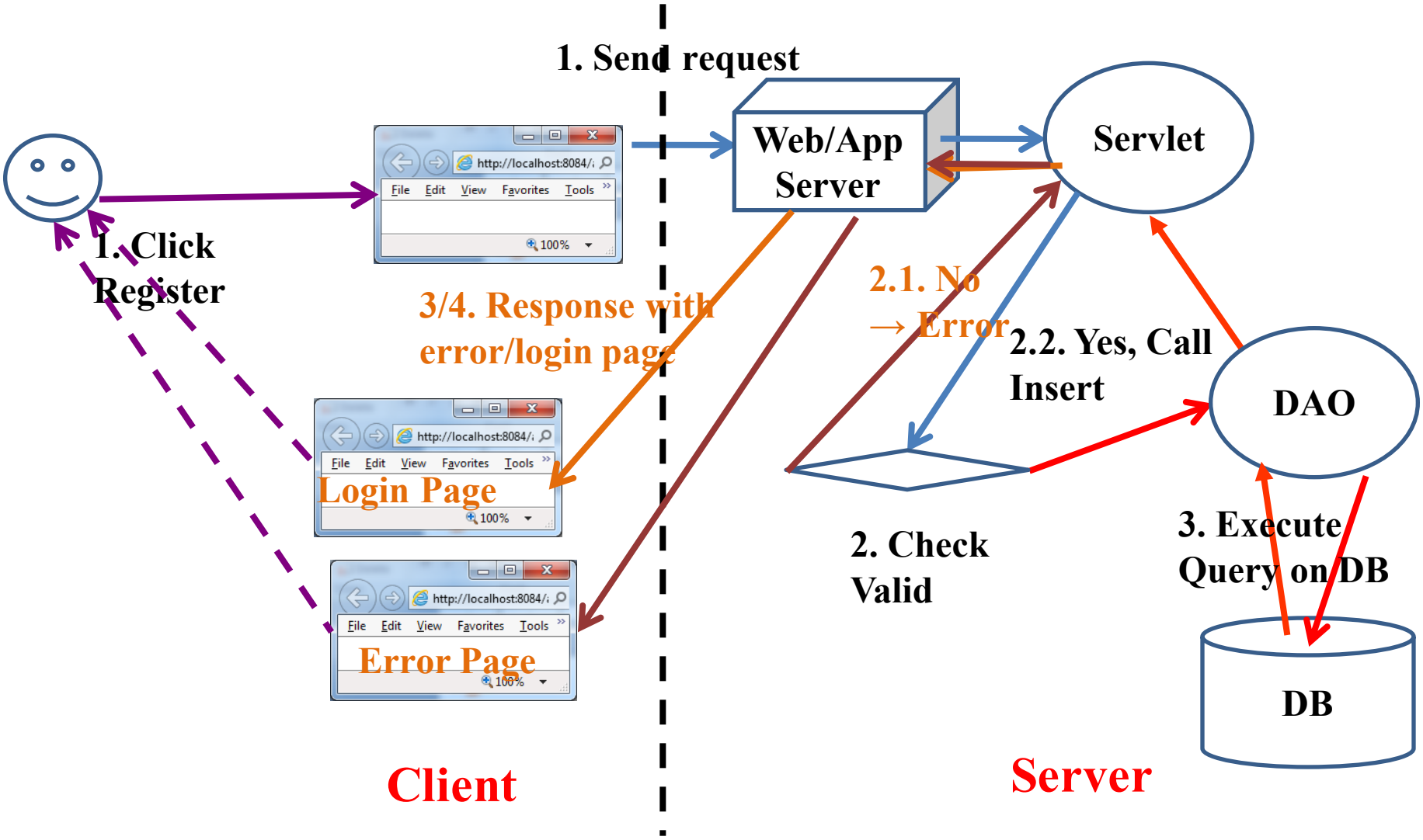
Validation





How to write CRUD Web Application

Interactive Server Model



Fpt University

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

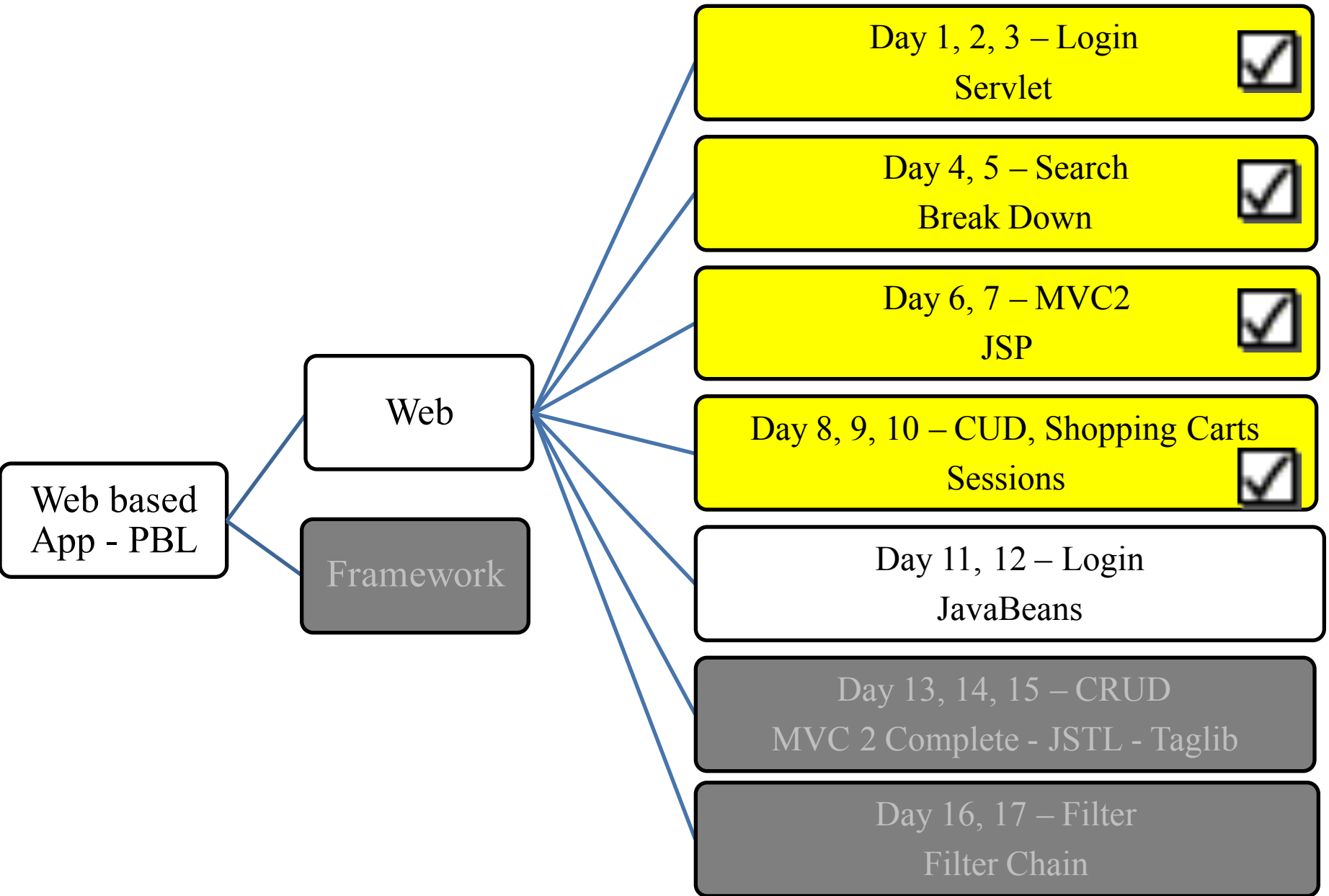
Q&A

Fpt University

Next Lecture

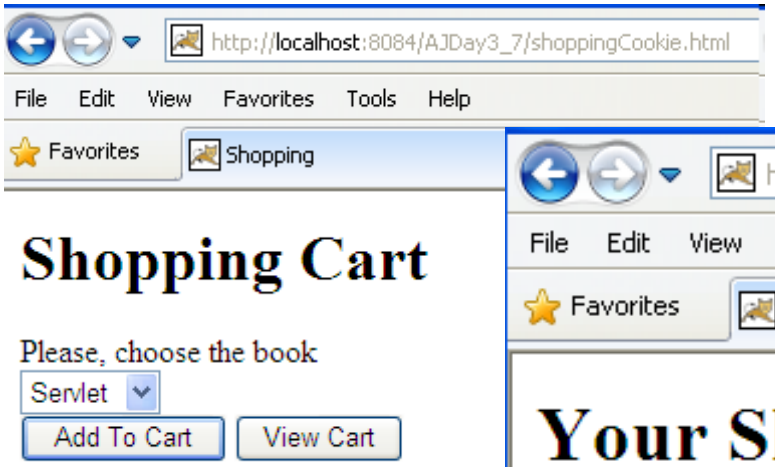
- **JSP Standard Actions**
  - JavaBeans
  - Standard Actions
- **Dispatcher Mechanism**
  - Including, Forwarding, and Parameters
  - Vs. Dispatcher in Servlets
- **EL – Expression Languages**
  - What is EL?
  - How to use EL in JSP?

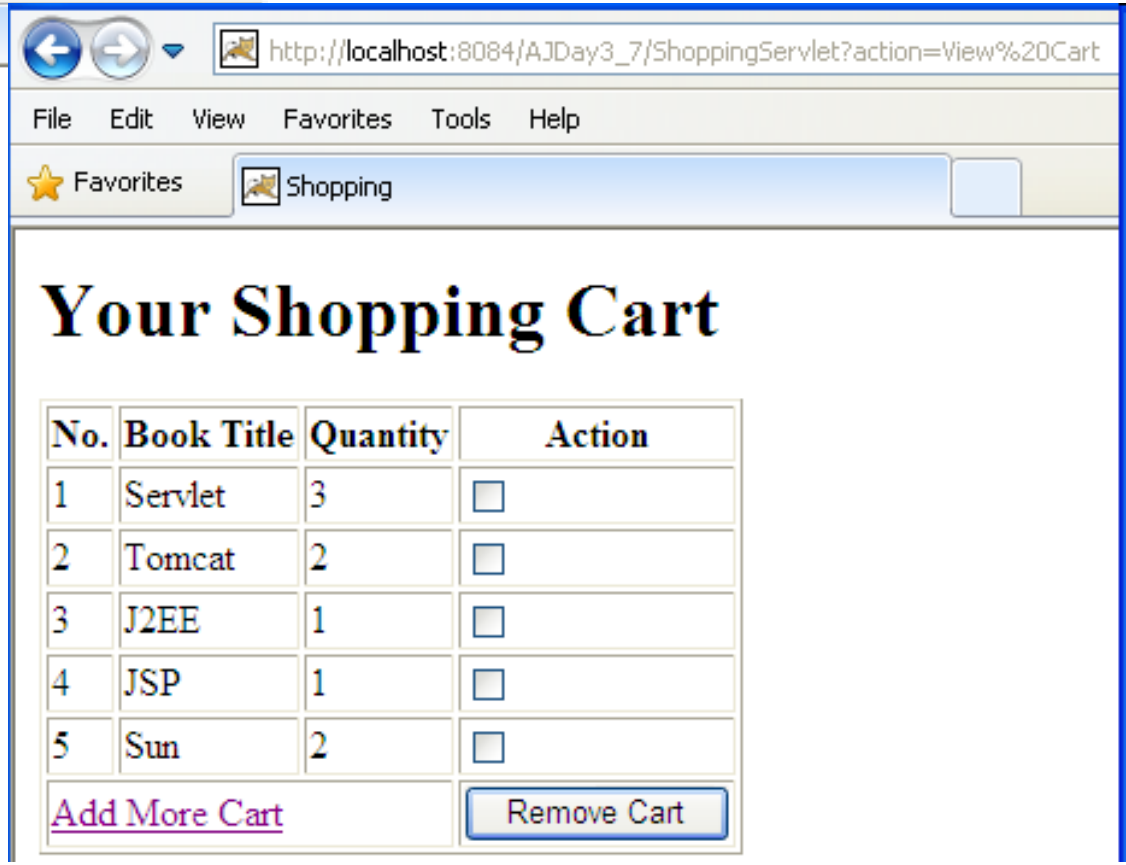
Next Lecture

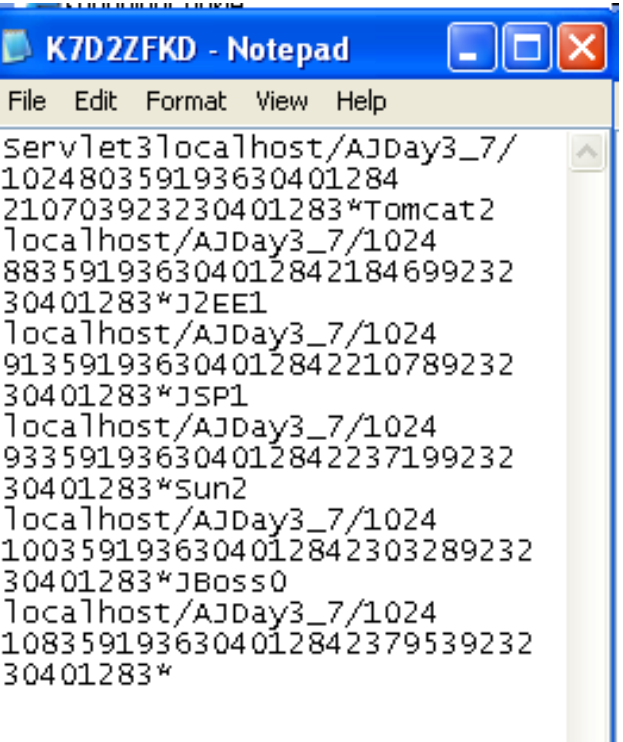


Appendix

Shopping Cart using Cookies







Appendix

Shopping Cart using Cookies

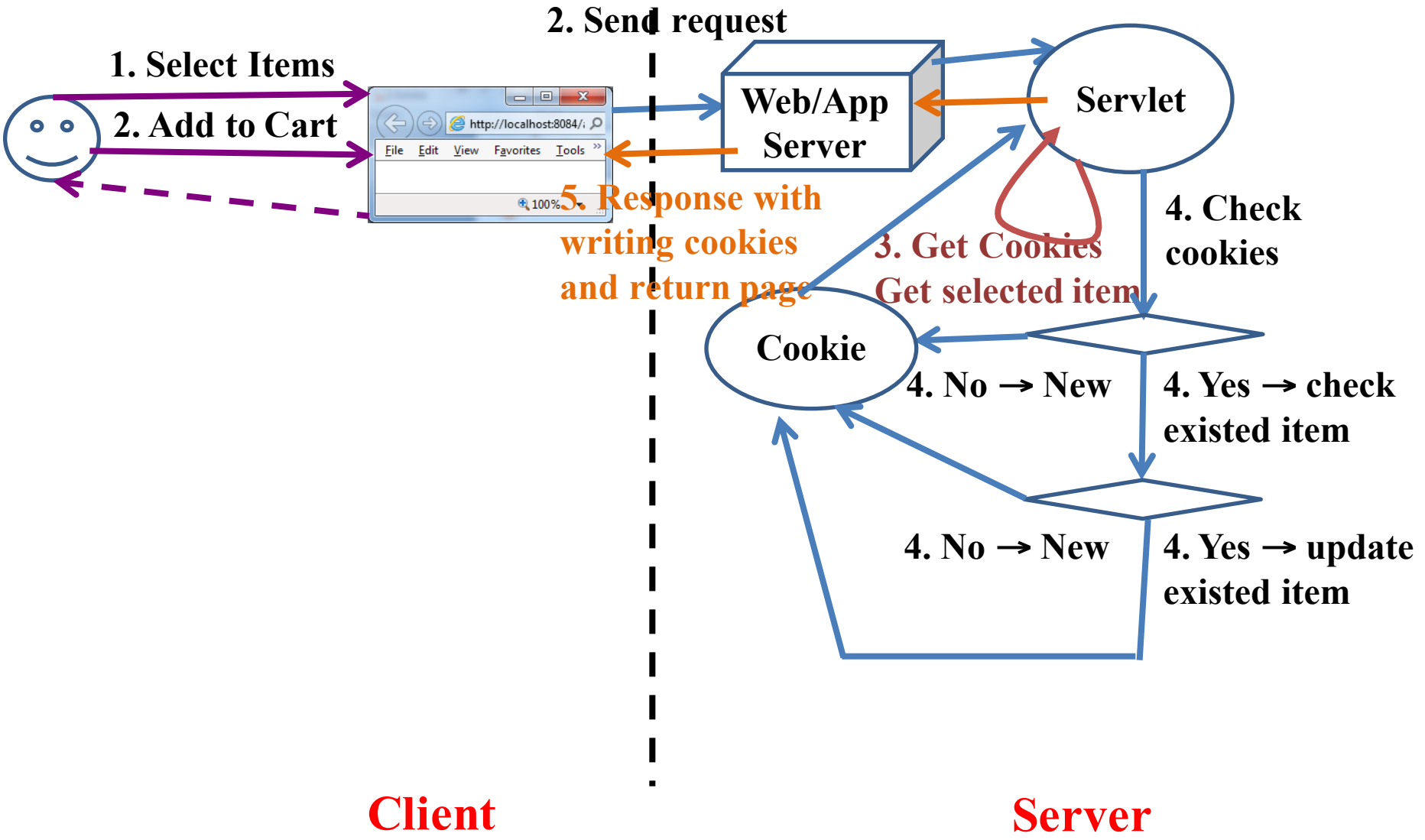
shoppingCookie.html

Preview

```
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

ShoppingServlet.java

Source | History

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

Appendix

Shopping Cart using Cookies

ShoppingAddServlet.java

Source | History

```
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 <c
31         protected void processRequest(HttpServletRequest request, Htt
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 {
```

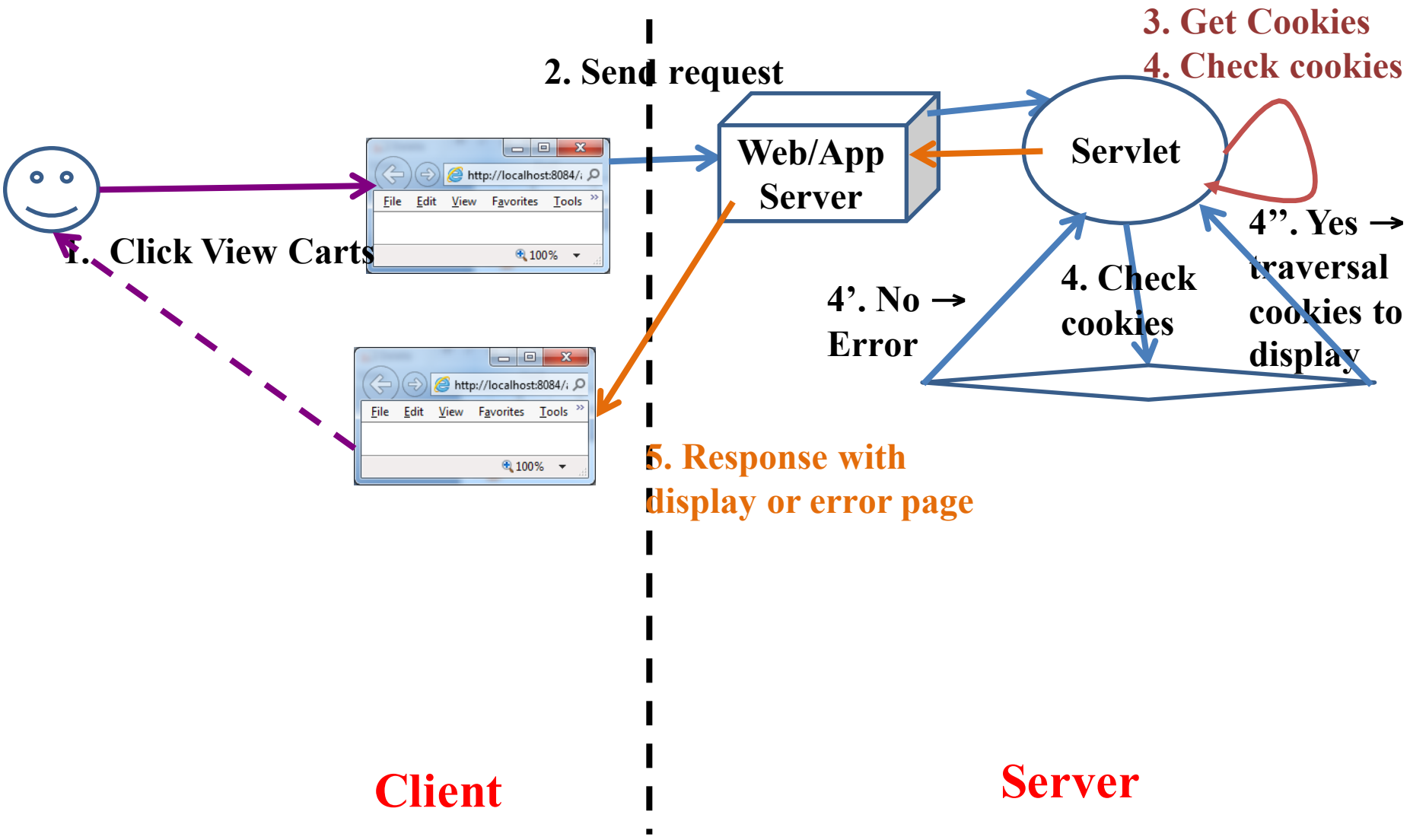
Appendix

Shopping Cart using Cookies

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

Appendix

Interactive Server Model



Appendix

Shopping Cart using Cookies

ShoppingServlet.java

Source | History

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

Appendix

Shopping Cart using Cookies

ShoppingViewServlet.java

Source | History

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

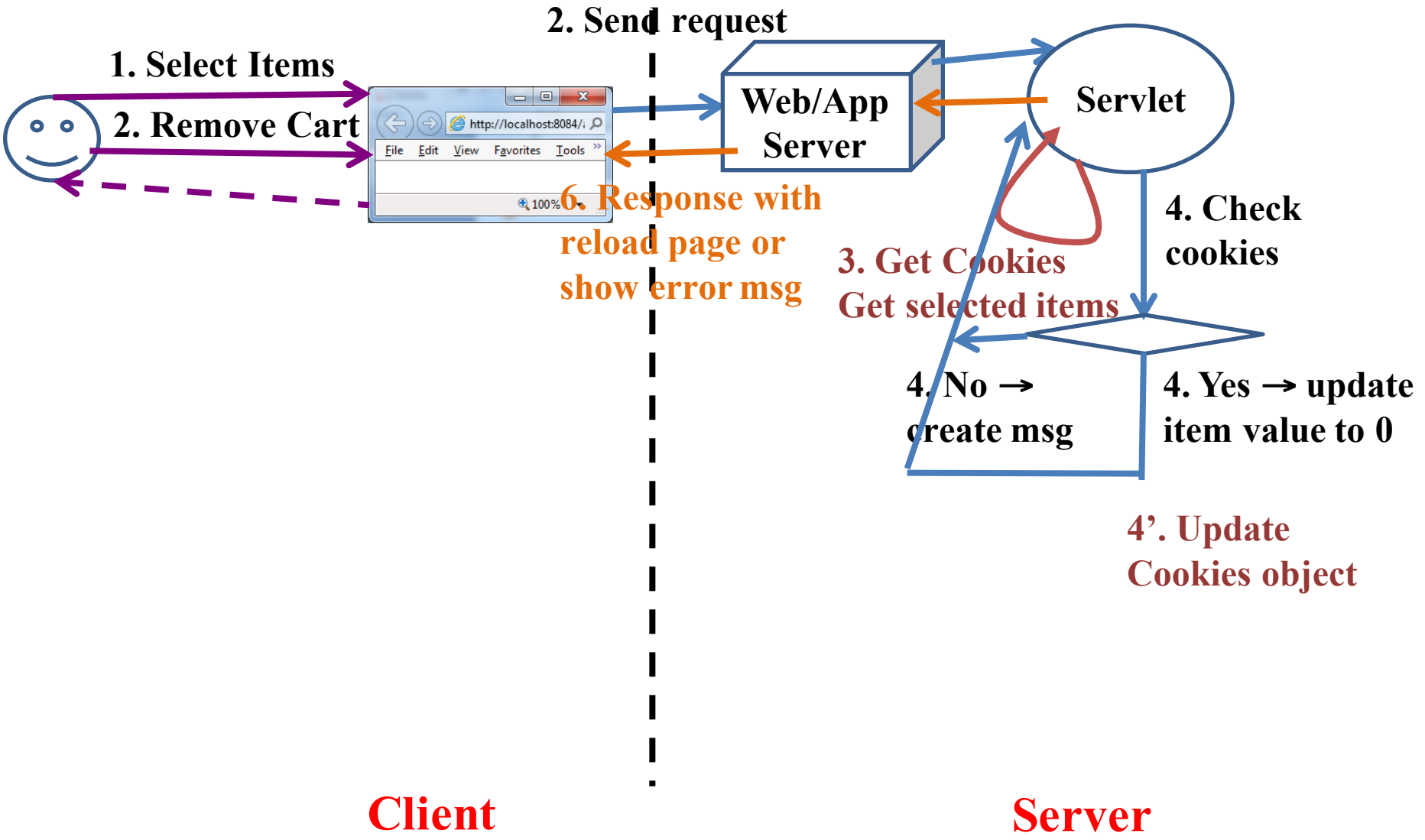
Appendix

Shopping Cart using Cookies

```
53
54                  int count = 1;
                    for (int i = 0; i < cookies.length; i++) {
56                      int tmp = Integer.parseInt(cookies[i].getValue());
57                      if (tmp > 0) {
58                          out.println("<tr>");
59                          out.println("<td>" + count++ + "</td>");
60                          out.println("<td>" + cookies[i].getName() + "</td>");
61                          out.println("<td>" + cookies[i].getValue() + "</td>");
62                          out.println("<td><input type='checkbox' name='rmv' value='"
63                                  + cookies[i].getName() + "' /></td>");
64                          out.println("</tr>");
65                      }
66                  }
67
68                  out.println("<tr>");
69                  out.println("<td colspan='3'><a href='shoppingCookie.html'>Add More Cart</a></td>");
70                  out.println("<td><input type='submit' value='Remove Cart' name='action' /></td>");
71                  out.println("</tr>");
72                  out.println("</form>");
73                  out.println("</table>");
74
75                  out.println("</body>");
76                  out.println("</html>");
77                  return;
78              }
79              out.println ("<h2>Cart is removed or No items in cart</h2>");
80          } finally {
81              out.close();
82          }
83      }
```

Appendix

Interactive Server Model



Appendix
Shopping Cart using Cookies

ShoppingServlet.java

Source | History

```
18      * @author Trong Khanh
19      */
20     public class ShoppingServlet extends HttpServlet {
21         private final String addCartServlet = "ShoppingAddServlet";
22         private final String viewCartServlet = "ShoppingViewServlet";
23         private final String removeCartServlet = "ShoppingRemoveServlet";
24         /** Processes requests for both HTTP ...10 lines */
34         protected void processRequest(HttpServletRequest request, HttpServletResponse respo
35                 throws ServletException, IOException {
36             response.setContentType("text/html;charset=UTF-8");
37             PrintWriter out = response.getWriter();
38             try {
39
40                 String action = request.getParameter("action");
41
42                 if (action.equals("Add To Cart")) {
43                     RequestDispatcher rd = request.getRequestDispatcher(addCartServlet);
44                     rd.forward(request, response);
45                 } else if (action.equals("View Cart")) {
46                     RequestDispatcher rd = request.getRequestDispatcher(viewCartServlet);
47                     rd.forward(request, response);
48                 } else if (action.equals("Remove Cart")) {
49                     RequestDispatcher rd = request.getRequestDispatcher(removeCartServlet);
50                     rd.forward(request, response);
51                 } else {
52                     //To Do Code
53                 }
54             } finally {
55                 out.close();
56             }
57         }
```

Appendix
Shopping Cart using Cookies

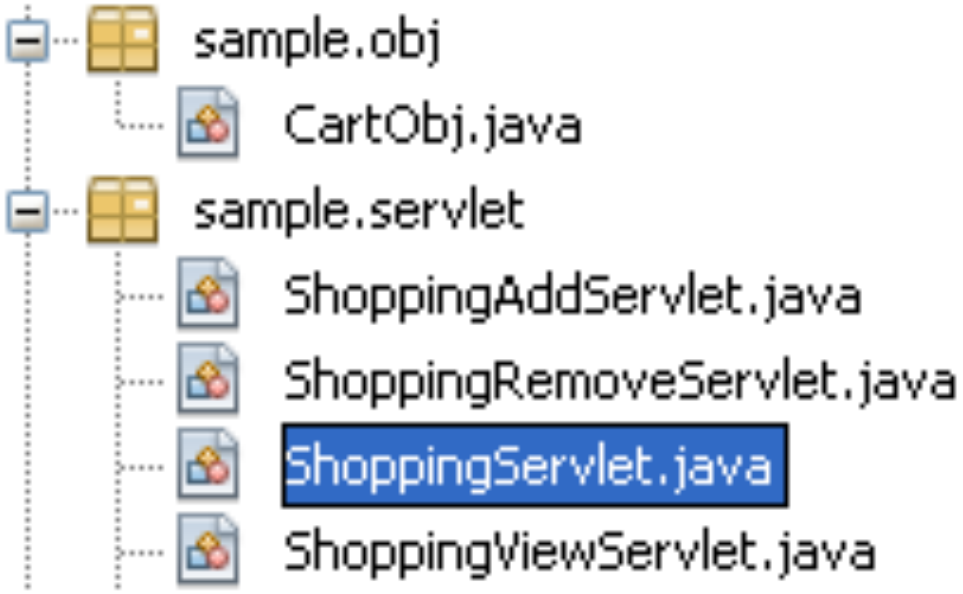
ShoppingRemoveServlet.java

Source | History

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

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.

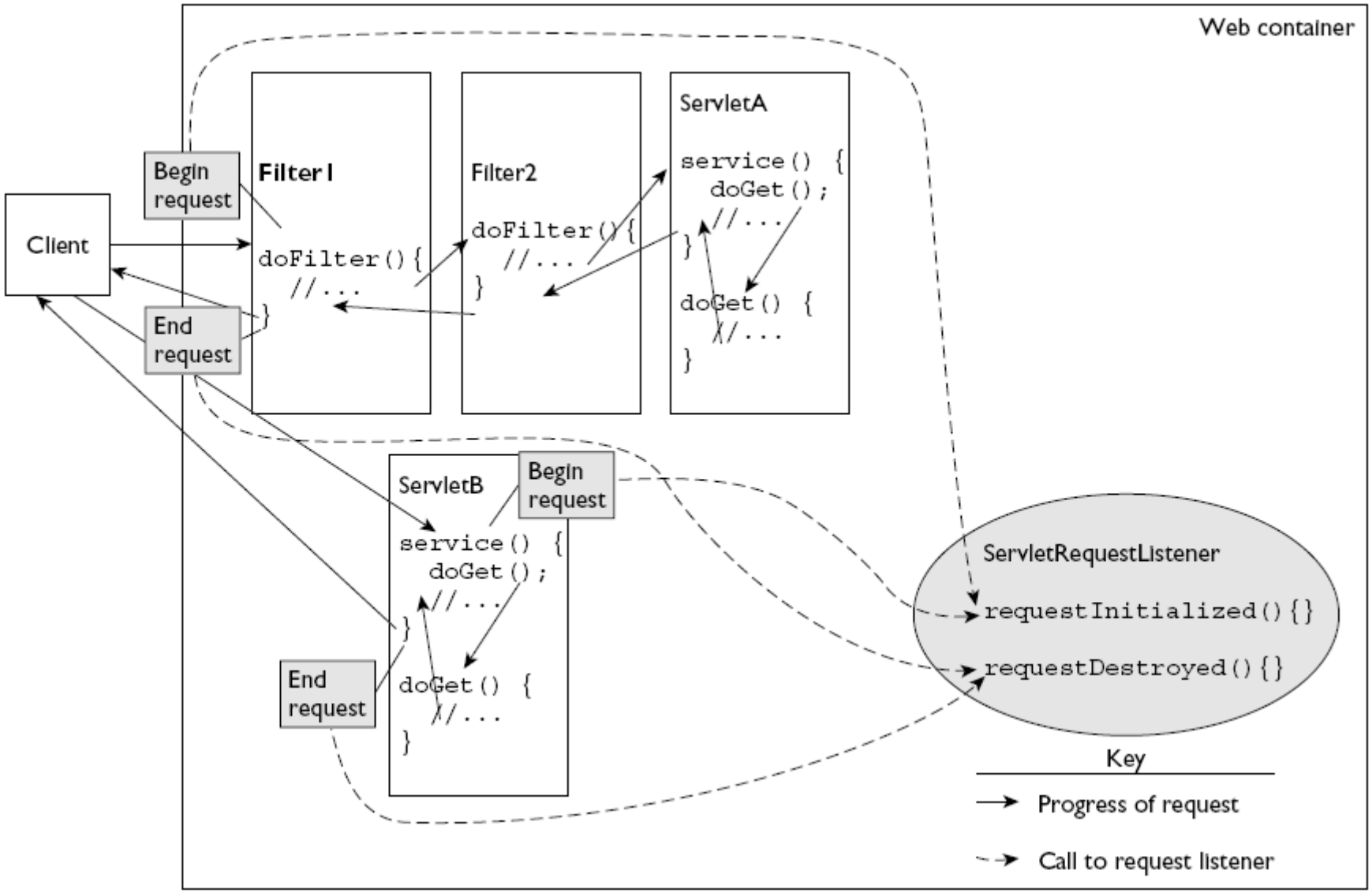
```
<listener>
    <listener-class>className</listener-class>
</listener>
```

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



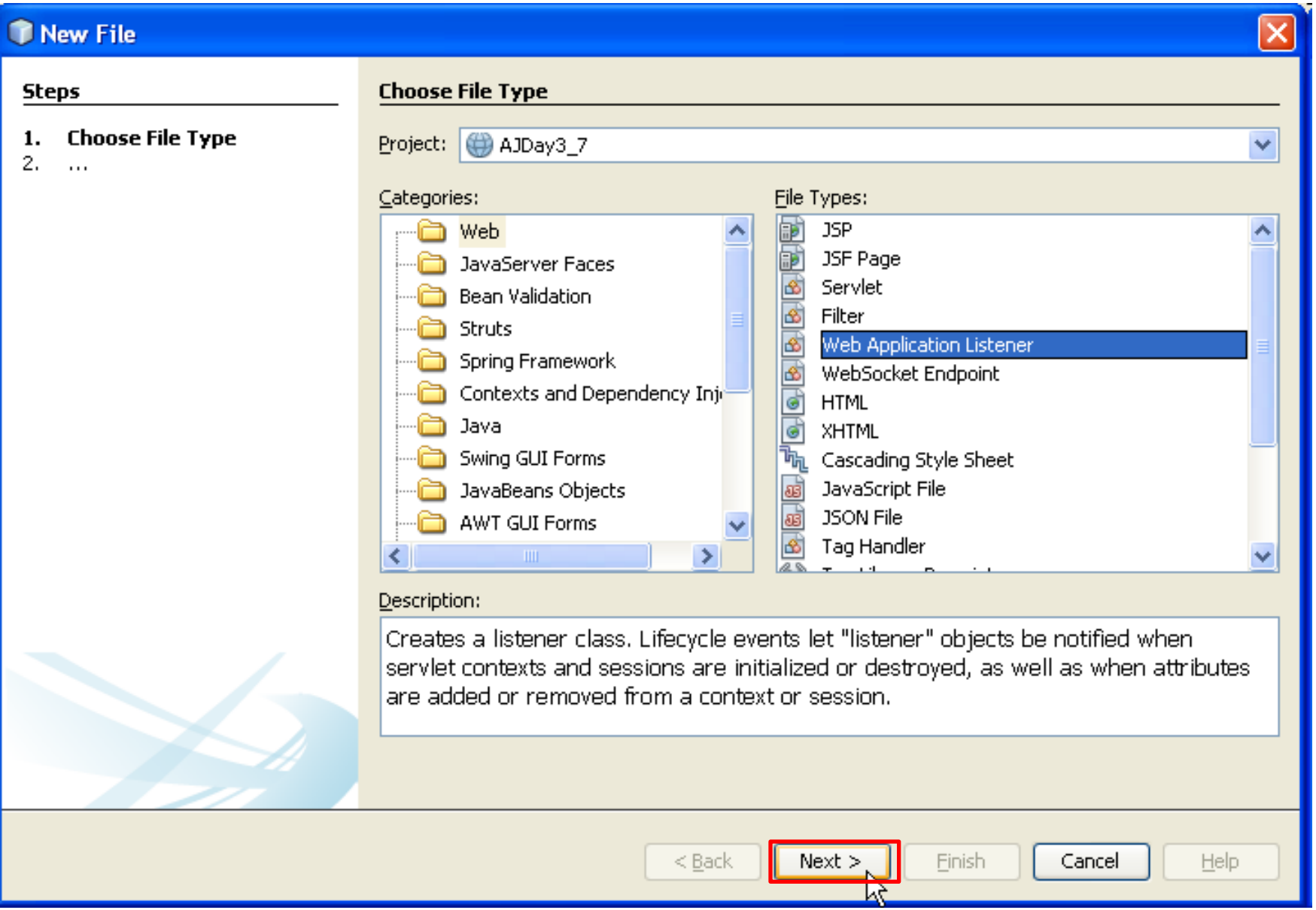
Appendix

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 ServetRequestEvent
- The "grandparent" of The ServletRequestEvent is java.util.EventObject
  - The getSource() method returns the object that is the source of the event

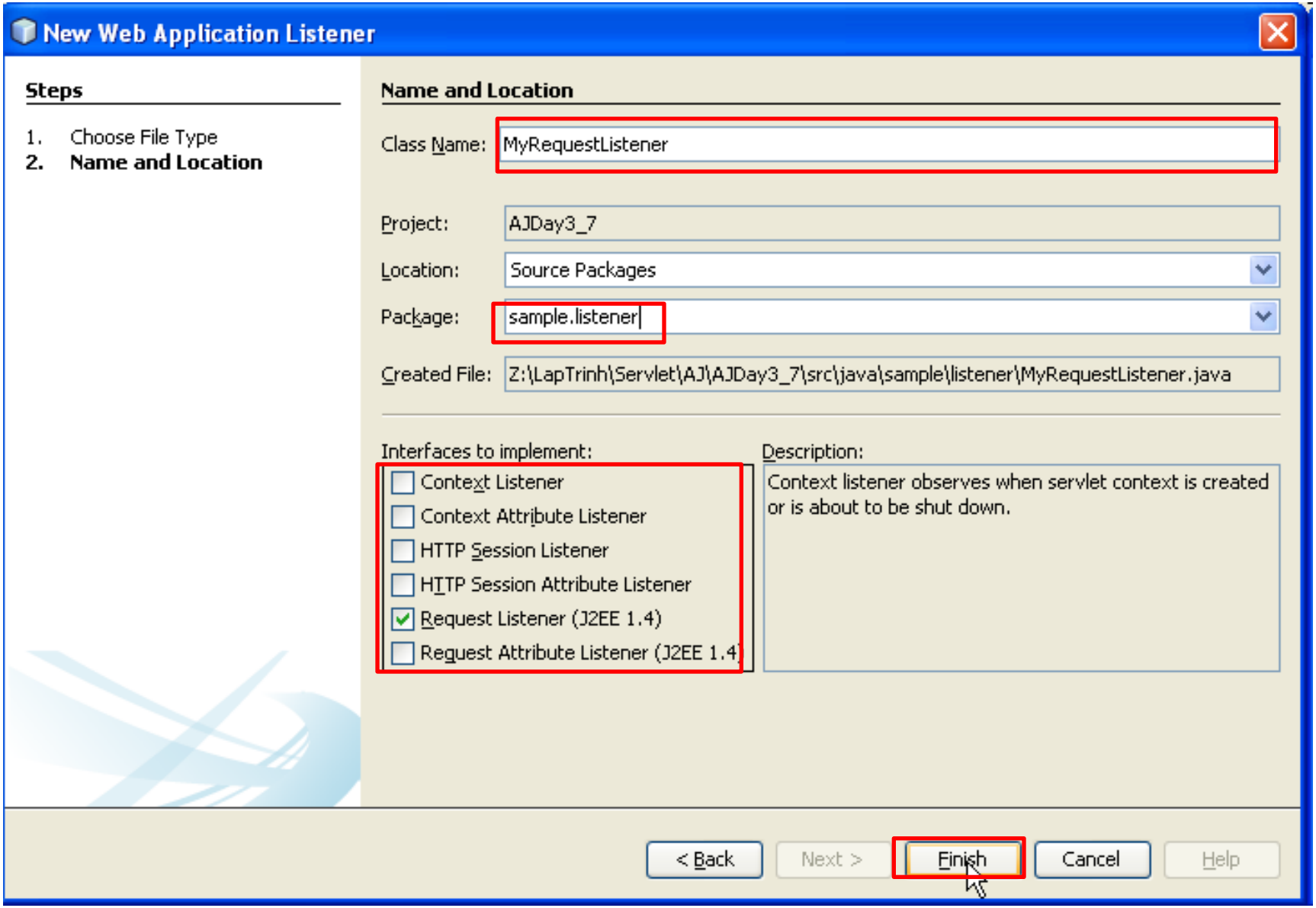
Appendix

How to Add Listener to Web Project



Appendix

How to Add Listener to Web Project



Appendix
Example

MyRequestListener.java

Source | History

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

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

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

Appendix
Example

MyRequestAttributeListener.java

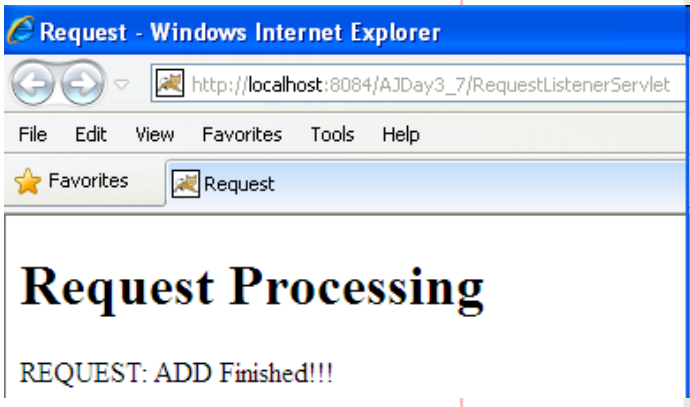
```
15  public class MyRequestAttributeListener implements ServletRequestAttributeListener {
        public void attributeAdded(ServletRequestAttributeEvent srae) {
17          System.out.println("Add is activated");
18          String name = srae.getName();
19          String oldValue = srae.getValue().toString();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

```
26      <listener>
27          <listener-class>sample.listener.MyContextAttributeListener</listener-class>
28      </listener>
```

Appendix
Example

RequestListenerServlet.java

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



Appendix
Example

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

Appendix
Example

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

Fpt University

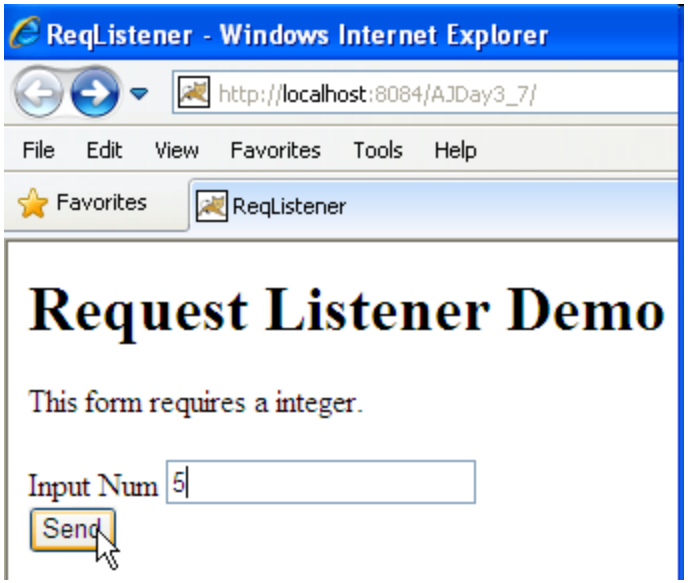
Appendix

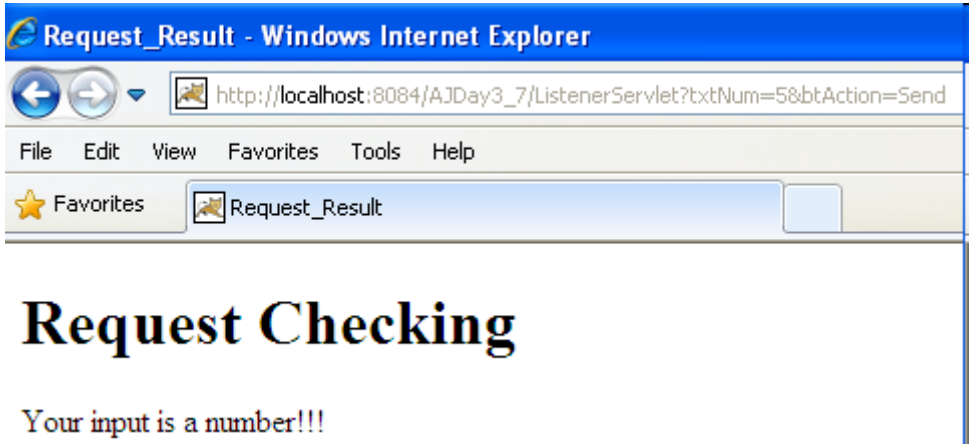
Example

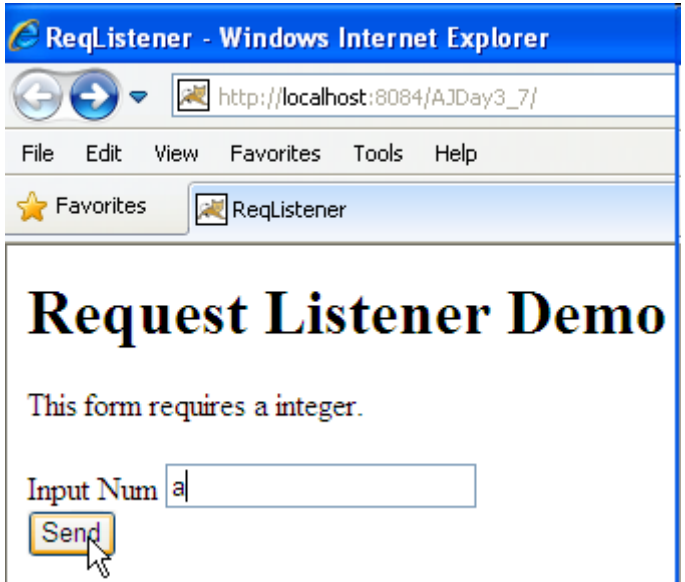
```
Remove is activated
Name: netbeans.monitor.response -old: org.netbeans.
Remove is activated
Name: netbeans.monitor.filter -old: MonitorFilter(A
Remove is activated
Name: netbeans.monitor.monData -old: [MonitorData]
destroyed in MyRequest is invoked!!!!
Remove is activated
Name: REQUEST -old: ADD
```

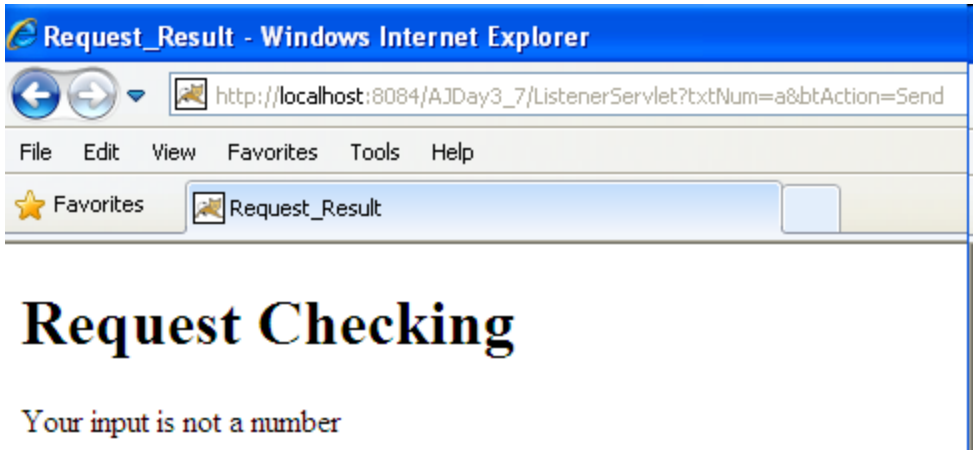
Appendix

Practices – Example









Appendix

Practices – Example

validUsingReqListener.html | ListenerServlet.java | MyRequestListener.java

Source | History

```
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

ListenerServlet.java | MyRequestListener.java

Source | History

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

    /**...*/
    protected void processRequest(HttpServletRequest request, HttpSe
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sam
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Request_Result</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Request Checking</h1>");

            Boolean check = (Boolean) request.getAttribute("VALID");
            if (check.booleanValue()) {
                out.print("Your input is a number!!!");
            } else {
                out.print("Your input is not a number");
            }

            out.println("</body>");
```

Appendix

Practices – Example

MyRequestListener.java

Source | History

```
 * @author Trong Khanh
 */
public class MyRequestListener implements ServletRequestListener {

    public void requestDestroyed(ServletRequestEvent sre) {...}

    public void requestInitialized(ServletRequestEvent sre) {

        String num = sre.getServletRequest().getParameter("txtNum");
        Boolean result = true;
        try {
            int n = Integer.parseInt(num);
        } catch (NumberFormatException ex) {
            result = false;
        }
        sre.getServletRequest().setAttribute("VALID", result);
    }
}
```

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 {
        public void contextInitialized(ServletContextEvent sce) {
18          System.out.println("context in MyContext is invoked!!!!");
19          sce.getServletContext().setAttribute("CONTEXT", "ADD");
20      }
        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

MyContextAttributeListener.java

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

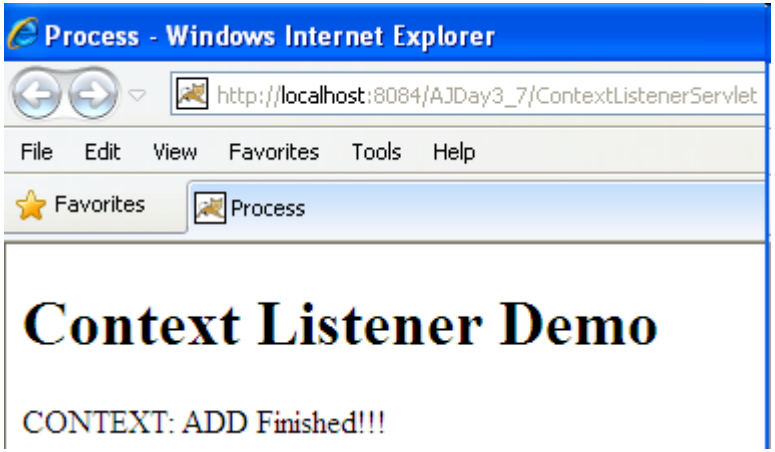
Example

ContextListenerServlet.java

```
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>Process</title>");
37              out.println("</head>");
38              out.println("<body>");
39              out.println("<h1>Context Listener Demo</h1>");
40
41              ServletContext sc = getServletContext();
42              sc.setAttribute("VALUE", "ADD");
43              sc.setAttribute("VALUE", "MODIFIED");
44              sc.removeAttribute("VALUE");
45
46              out.println("CONTEXT: " + sc.getAttribute("CONTEXT"));
47              out.println("Finished!!!");
48
49              out.println("</body>");
50              out.println("</html>");
51          } finally {
52              out.close();
53          }
54      }
```

Appendix

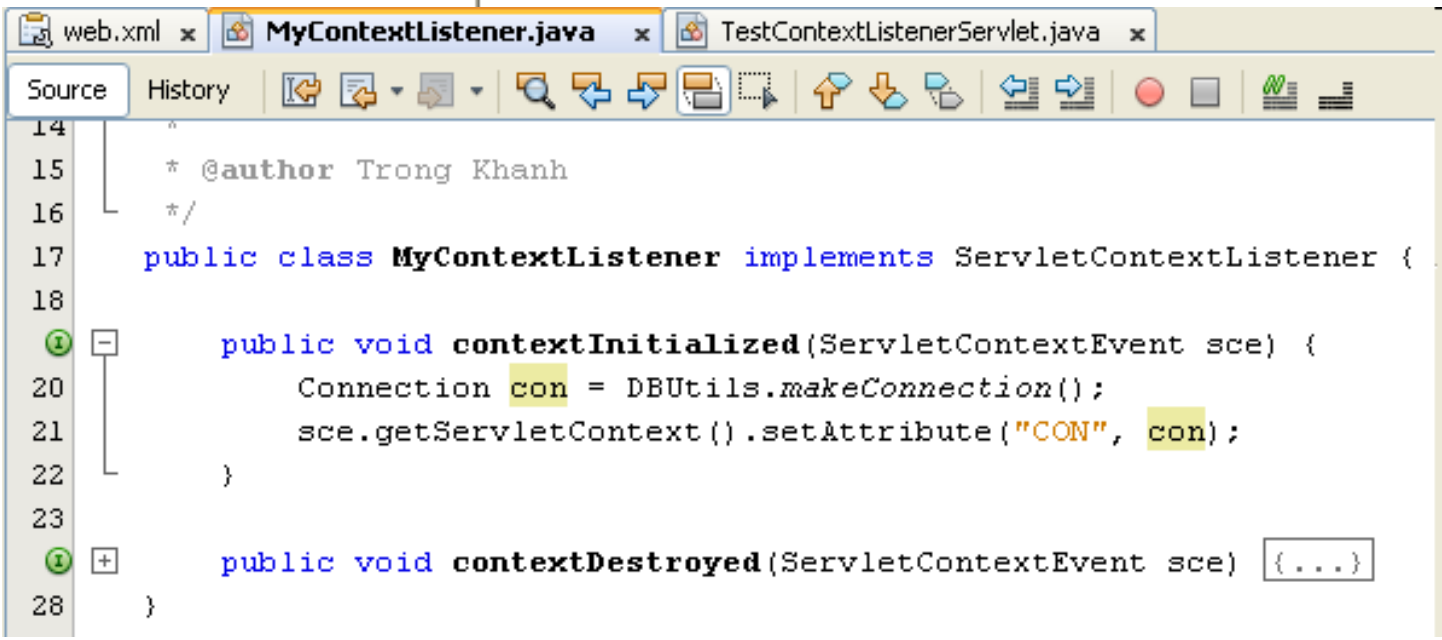
Example




```
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



Appendix

Practices – Example

web.xml | TestContextListenerServlet.java

Source | History

```
20      * @author Trong Khanh
21      */
22     public class TestContextListenerServlet extends HttpServlet {
23
24         /**...*/
34         protected void processRequest(HttpServletRequest request, HttpServletResponse
35                 throws ServletException, IOException {
36             response.setContentType("text/html;charset=UTF-8");
37             PrintWriter out = response.getWriter();
38             try {
39                 /* TODO output your page here. You may use following sample code. */
40                 out.println("<!DOCTYPE html>");
41                 out.println("<html>");
42                 out.println("<head>");
43                 out.println("<title>Context</title>");
44                 out.println("</head>");
45                 out.println("<body>");
46                 out.println("<h1>Test Context Listener Servlet</h1>");
47
48                 Connection con = (Connection) getServletContext().getAttribute("CON");
49                 try {
50                     out.print("Test DB Connection is Closed: " + con.isClosed());
51                 } catch (SQLException ex) {
52                     Logger.getLogger(TestContextListenerServlet.class.getName()).log(L
53                 }
54
55                 out.println("</body>");
```

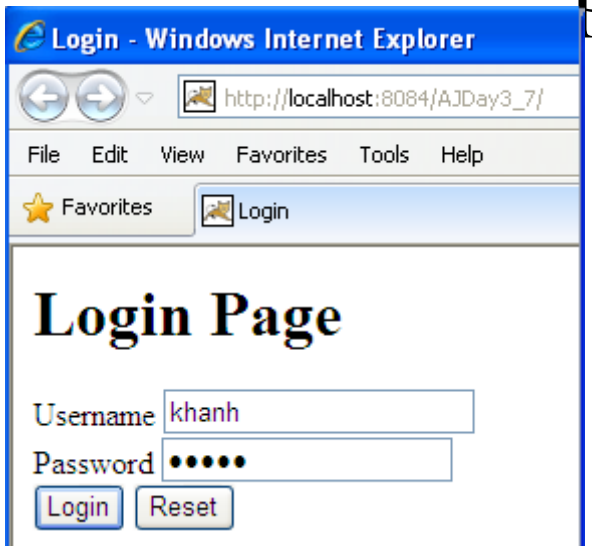
Appendix

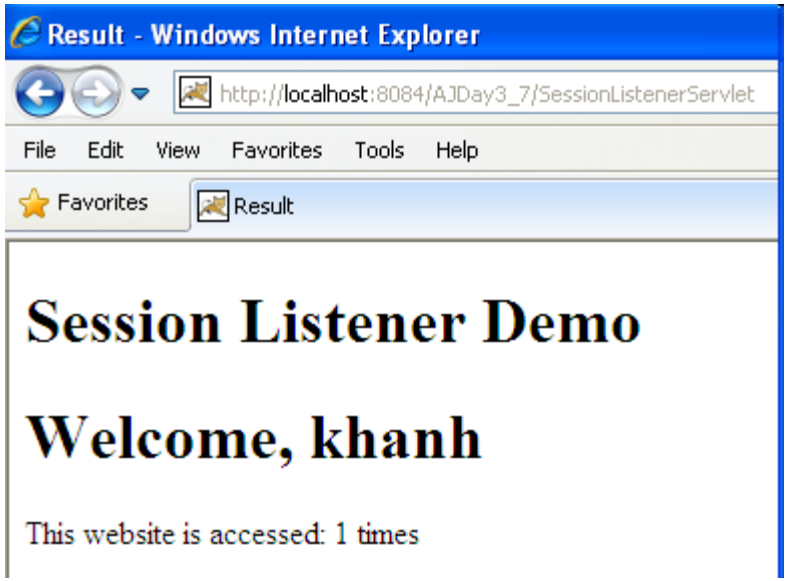
Session Listeners Declared in DD

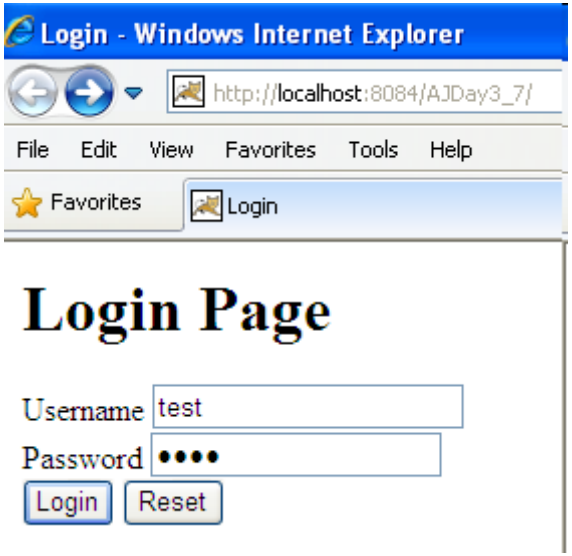
- Have **02 listeners:**
  - **HttpSessionListener**
    - Implements the changes to the list of active sessions in Web application
    - **sessionCreated**() method: is called whenever a **new session** is provided (*can say that **after** the **getSession()** method*)
    - **sessionDestroyed**(): 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)

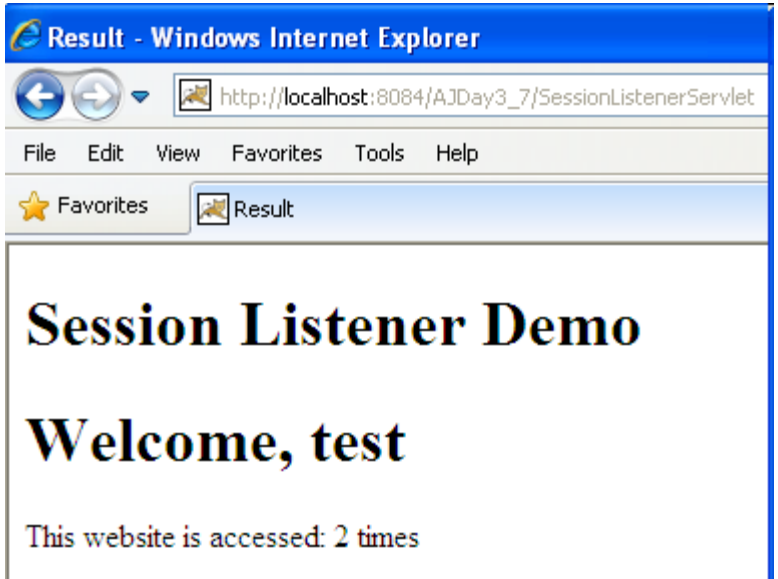
Appendix

Practices – Example









Appendix

Practices – Example

loginCountSession.html

Source | History

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

Appendix
Practices – Example

SessionListenerServlet.java

Source | History

```
17    * @author Trong Khanh
18    */
19   public class SessionListenerServlet extends HttpServlet {
20
21       /**...*/
31       protected void processRequest(HttpServletRequest request, HttpServletResponse res
32               throws ServletException, IOException {
33           response.setContentType("text/html;charset=UTF-8");
34           PrintWriter out = response.getWriter();
35           try {
36               out.println("<!DOCTYPE html>");
37               out.println("<html>");
38               out.println("<head>");
39               out.println("<title>Result</title>");
40               out.println("</head>");
41               out.println("<body>");
42               out.println("<h1>Session Listener Demo</h1>");
43               String user = request.getParameter("txtUsername");
44               String pass = request.getParameter("txtPassword");
45               if (user.equals(pass)) {
46                   HttpSession session = request.getSession();
47                   out.println("<h1>Welcome, " + user + "</h1>");
48                   Integer count = (Integer) getServletContext().getAttribute("COUNT");
49                   out.println("This website is accessed: " + count + " times");
50               } else {
51                   out.println("<h1>Invalid username or password</h1>");
52               }
53               out.println("</body>");
```

Appendix
Practices – Example

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

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

MyContextListener.java

Source | History

```
15      * @author Trong Khanh
16      */
17  public class MyContextListener implements ServletContextListener {
18
19      public void contextInitialized(ServletContextEvent sce) {
20          sce.getServletContext().setAttribute("COUNT", 0);
21      }
22
23      public void contextDestroyed(ServletContextEvent sce) {...}
27  }
```

Appendix
Practices – Example

MySessionListener.java

Source | History

```
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) {...}
28     }
```

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	- **public void valueBound(HttpSessionBindingEvent se);** - **Notifies** the object in being **bound** to a **session** and is responsible for identification of the session
valueUnbound	- **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

SessionAttrObject.java

```
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      }
        public String toString() {
24          return data;
25      }
26      public void setData(String data) {
27          this.data = data;
28      }
        public void valueBound(HttpSessionBindingEvent event) {
30          System.out.println("valueBound() " + getData());
31      }
        public void valueUnbound(HttpSessionBindingEvent event) {
33          System.out.println("valueUnbound " + getData());
34      }
35  }
```

Appendix

Session Listeners Not Declared in DD - Example

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 {
```

```
valueBound() khanh
valueBound() kieu
valueBound() kieu
valueUnbound khanh
valueUnbound kieu
valueUnbound kieu
```

Appendix

Session Listeners Not Declared in DD

- Have **02 listeners** (cont)
  - **HttpSesssionActivationListener** (*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	- **public void sessionDidActivate(HttpSessionEvent se);** - Provides notification that the session has just been activated.
sessionWillPassivate	- **public void sessionWillPassivate(HttpSessionEvent se);** - Provide notification that the session is about to be passivated.

Appendix

CRUD Function

search.jsp

Source | History

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

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                                  <td>
93                                      <input type="checkbox" name="chkRole" value="ADMIN"
94                                             <%
95                                              if (dto.isRole()) {
96                                                  %>
97                                                  checked="checked"
98                                              <%
99                                              }
100                                            %>
101                                            />
102                                 </td>
103                                 <td>
104                                     <a href="<%= urlRewriting %>">Delete</a>
105                                 </td>
106                                 <td>
107                                     <input type="hidden" name="lastSearchValue"
108                                            value="<%= searchValue %>" />
109                                     <input type="submit" value="Update" name="btAction" />
110                                 </td>
111                             </tr>
112                         </form>
113                             <%
114                                 }
115                             %>
116                         </tbody>
117                     </table>
118
119             <%
120                     } else {
121                         %>
```

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

Appendix

CUD Function

RegistrationDAO.java

Source | History

```
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

RegistrationDAO.java

Source | History

```
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

NullServlet.java

Source | History

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

Appendix

CRUD Function

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

Appendix

CRUD Function

SE1162Servlet.java

Source | History

```
 * @author kieukhanh
 */
public class SE1162Servlet extends HttpServlet {
    private final String loginServlet = "LoginServlet";
    private final String loginPage = "login.html";
    private final String searchServlet = "SearchServlet";
    private final String deleteRecordServlet = "DeleteRecordServlet";
    private final String updatePassRoleServlet = "UpdatePassRoleServlet";
    private final String nullServlet = "NullServlet";
    private final String addItemServlet = "AddItemServlet";
    private final String viewCartPage = "viewCart.jsp";
    private final String deleteItemServlet = "DeleteItemServlet";
    private final String createAccountServlet = "CreateAccountServlet";

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

        String button = request.getParameter("btAction");
        String url = loginPage;
        try {
            if (button == null) {
                url = nullServlet;
            } else if (button.equals("Login")) {
                url = loginServlet;
            } else if (button.equals("Search")) {
                url = searchServlet;
            } else if (button.equals("delete")) {
```

Appendix

CRUD Function

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

Appendix

CRUD Function

DeleteRecordServlet.java

Source | History

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

Appendix

CRUD Function

UpdatePassRoleServlet.java

Source | History

```
22      * @author kieukhanh
23      */
24     @WebServlet(name = "UpdatePassRoleServlet", urlPatterns = {"/UpdatePassRoleServl
25     public class UpdatePassRoleServlet extends HttpServlet {
26         private final String updateErrPage = "update.Err.html";
27         /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
36         protected void processRequest(HttpServletRequest request, HttpServletRespons
37                 throws ServletException, IOException {
38             response.setContentType("text/html;charset=UTF-8");
39             PrintWriter out = response.getWriter();
40
41             String urlRewriting = updateErrPage;
42             try {
43                 String username = request.getParameter("txtUsername");
44                 String password = request.getParameter("txtPassword");
45                 String admin = request.getParameter("chkRole");
46                 boolean role = false;
47                 if (admin != null) {
48                     role = true;
49                 }
50                 String searchValue = request.getParameter("lastSearchValue");
51
52                 RegistrationDAO dao = new RegistrationDAO();
53                 boolean result = dao.updatePassRole(username, password, role);
54
55                 if (result) {
56                     urlRewriting = "SE1162Servlet?btAction=Search&txtSearchValue="
57                             + searchValue;
58                 }
59             }   catch (NamingException ex) {
                    ex.printStackTrace();
61             } catch (SQLException ex) {
                    ex.printStackTrace();
63             } finally {
64                 response.sendRedirect(urlRewriting);
65                 out.close();
66             }
67         }
```

Appendix

CRUD Function

bookStore.html

Source | History

```
1  <!DOCTYPE html>
2  ...5 lines
7  <html>
8      <head>
9          <title>Book Store</title>
10         <meta charset="UTF-8">
11         <meta name="viewport" content="width=device-width, initial-scale=1.0">
12     </head>
13     <body>
14         <h1>Book Store</h1>
15         <form action="SE1162Servlet">
16             Choose book <select name="cboBook">
17                 <option>Servlet</option>
18                 <option>Tomcat</option>
19                 <option>JSP</option>
20                 <option>MVC</option>
21                 <option>JavaEE</option>
22                 <option>EL</option>
23                 <option>EJB2</option>
24                 <option>EJB3</option>
25                 <option>JBoss</option>
26             </select><br/>
27             <input type="submit" value="Add Book To Your Cart" name="btAction" />
28             <input type="submit" value="View Your Cart" name="btAction" />
29         </form>
30     </body>
31 </html>
```

Appendix

CRUD Function

CartObj.java

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

AddItemServlet.java

Source | History

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

Appendix

CRUD Function

viewCart.jsp

Source | History

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

Appendix

CRUD Function

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

Appendix

CRUD Function

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

Appendix

CRUD Function

DeleteItemServlet.java

Source | History

```
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</cod
34         protected void processRequest(HttpServletRequest request, HttpServletRespo
35                 throws ServletException, IOException {
36             response.setContentType("text/html;charset=UTF-8");
37             PrintWriter out = response.getWriter();
38             try {
39                 HttpSession session = request.getSession(false);
40
41                 if (session != null) {
42                     CartObj cart = (CartObj)session.getAttribute("CART");
43
44                     if (cart != null) {
45                         String[] items = request.getParameterValues("chkItem");
46
47                         if (items != null) {
48                             for (String item : items) {
49                                 cart.removeItemFromCart(item);
50                             }//end for
51                             session.setAttribute("CART", cart);
52                         }//end if items
53                     }//end if cart
54                 }//end if session
55
56                 String urlRewriting = "SE1162Servlet?btAction=View Your Cart";
57                 response.sendRedirect(urlRewriting);
```

Appendix

CRUD Function

createNewAccount.html

Source | History

```
1   <!DOCTYPE html>
2   ...5 lines
7   <html>
8       <head>
9           <title>New Account</title>
10          <meta charset="UTF-8">
11          <meta name="viewport" content="width=device-width, initial-scale=1.0">
12      </head>
13      <body>
14          <h1>Create New Account</h1>
15          <form action="SE1162Servlet" method="POST">
16              Username* <input type="text" name="txtUsername" value="" />(6 - 20 chars)<br/>
17              Password* <input type="password" name="txtPassword" value="" />(6 - 30 chars)<br/>
18              Confirm* <input type="password" name="txtConfirm" value="" /><br/>
19              Full name <input type="text" name="txtFullname" value="" />(2 - 50 chars)<br/>
20              <input type="submit" value="Create New Account" name="btAction" />
21              <input type="reset" value="Reset" />
22          </form>
23      </body>
24  </html>
```

Appendix

CRUD Function

RegistrationInsertError.java

Source | History

```
 * @author kieukhanh
 */
public class RegistrationInsertError implements Serializable {
    private String usernameLengthErr;
    private String passwordLengthErr;
    private String confirmNotMatch;
    private String fullNameLengthErr;
    private String usernameIsExisted;
    /**...3 lines */
    public String getUsernameLengthErr() {...3 lines }
    /**...3 lines */
    public void setUsernameLengthErr(String usernameLengthErr) {...3 lines }
    /**...3 lines */
    public String getPasswordLengthErr() {...3 lines }
    /**...3 lines */
    public void setPasswordLengthErr(String passwordLengthErr) {...3 lines }
    /**...3 lines */
    public String getConfirmNotMatch() {...3 lines }
    /**...3 lines */
    public void setConfirmNotMatch(String confirmNotMatch) {...3 lines }
    /**...3 lines */
    public String getFullNameLengthErr() {...3 lines }
    /**...3 lines */
    public void setFullNameLengthErr(String fullNameLengthErr) {...3 lines }
    /**...3 lines */
    public String getUsernameIsExisted() {...3 lines }
    /**...3 lines */
    public void setUsernameIsExisted(String usernameIsExisted) {...3 lines }
}
```

Appendix

CRUD Function

RegistrationDAO.java

Source | History

```
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 
210         return false;
211     }
```

Appendix

CRUD Function

DeleteRecordServlet.java

Source | History

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

Appendix

CRUD Function

createNewAccount.jsp

Source | History

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

Appendix

CRUD Function

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

Appendix

CRUD Function

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

Appendix

CRUD Function

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmln
3      <servlet>
4          <servlet-name>LoginServlet</servlet-name>
5          <servlet-class>sample.servlet.LoginServlet</servlet-class>
6      </servlet>
7      <servlet>
8          <servlet-name>SE1162Servlet</servlet-name>
9          <servlet-class>sample.servlet.SE1162Servlet</servlet-class>
10     </servlet>
11     <servlet-mapping>
12         <servlet-name>LoginServlet</servlet-name>
13         <url-pattern>/LoginServlet</url-pattern>
14     </servlet-mapping>
15     <servlet-mapping>
16         <servlet-name>SE1162Servlet</servlet-name>
17         <url-pattern>/SE1162Servlet</url-pattern>
18     </servlet-mapping>
19     <session-config>
24     <welcome-file-list>
25         <welcome-file>SE1162Servlet</welcome-file>
26     </welcome-file-list>
27     <resource-ref>
33 </web-app>
```