

Q1. A session in Java refers to a mechanism used to maintain state and track interactions between a user and a web application across multiple HTTP requests. sessions help in storing user-specific information on the server side.

The primary role session is to enable the storage and retrieval of user-specific data throughout the user's browsing session by storing those information's on server momentarily.

It starts from the instance the user logs into the application and remains till the user logs out of the application or shuts down the machine. In both cases, the session values are deleted automatically.

Q2.

HTTPSESSION GETSESSION ()	HTTPSESSION GETSESSION (WITH PARAMETERS)
This method always returns a HTTP Session object.	This method provides more control over session management.
If a session already exists for the current request, it returns that session. If no session exists, it automatically creates a new session and returns it.	<p>When create is true, it behaves like get Session() without parameters, returning the existing session or creating a new one if none exists.</p> <p>When create is false, it returns the existing session if one is associated with the request; otherwise, it returns null without creating a new session.</p>
This method is typically used when the developer wants to ensure that a session is always available, regardless of whether it previously existed.	This method is useful when you want to check for an existing session without creating a new one, which can be important for handling specific session-related logic.

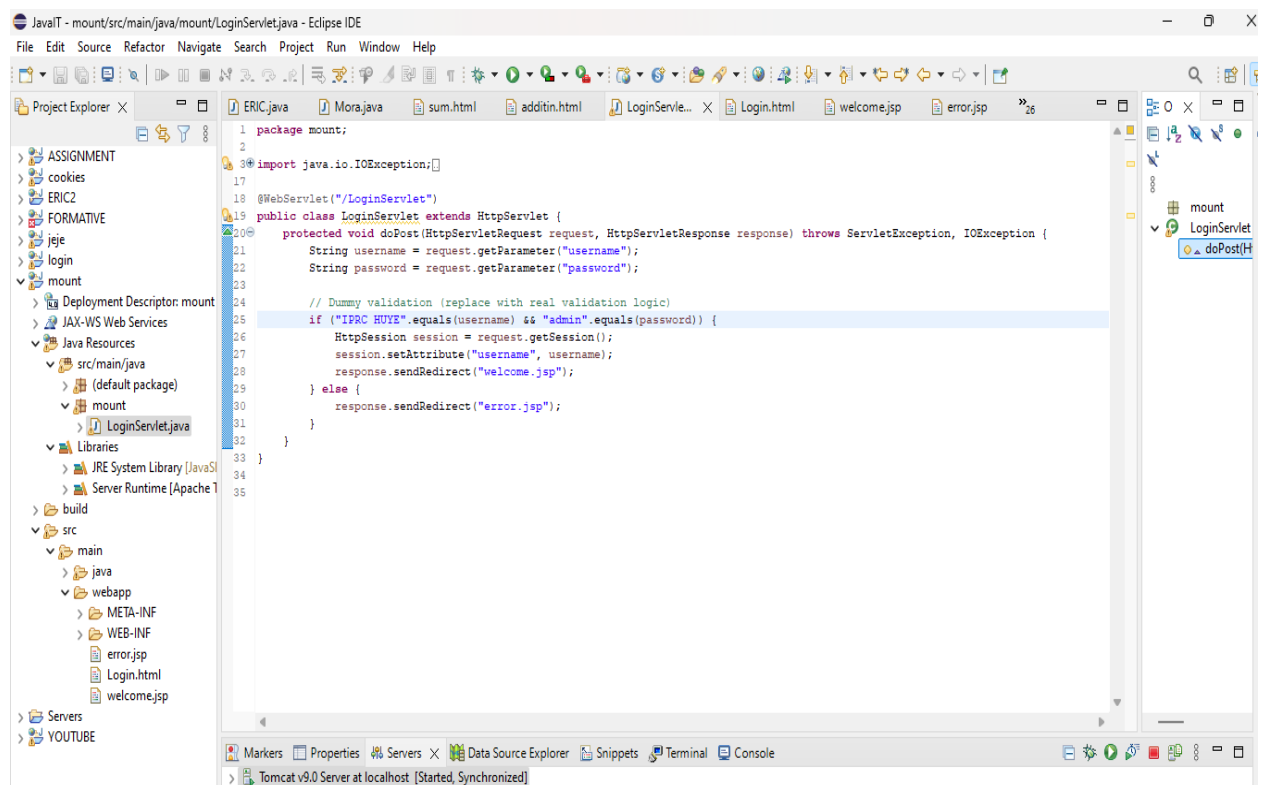
Q3.

- **getId():** Returns the unique session identifier assigned to this session.
- **get Attribute(String name):** Retrieves an object associated with the session, identified by a specified name.
- **set Attribute(String name, Object value):** Binds an object to the session, using the specified name as the key.
- **removeAttribute(String name):** Removes the object bound with the specified name from the session.

- **invalidate():** Invalidates the session, removing all attributes bound to it. This method is often used during user logout.
- **getCreationTime():** Returns the time when the session was created, expressed in milliseconds since midnight January 1, 1970 UTC.
- **getLastAccessedTime():** Returns the last time the session was accessed, also expressed in milliseconds.
- **setMaxInactiveInterval(int interval):** Sets the maximum time interval in seconds that the session will be kept open between client accesses. After this time, the session is invalidated.
- **getMaxInactiveInterval():** Returns the maximum time interval, in seconds, that the session will remain open between client accesses.

4. PRACTICAL QUESTIONS

SOURCE CODE ALL

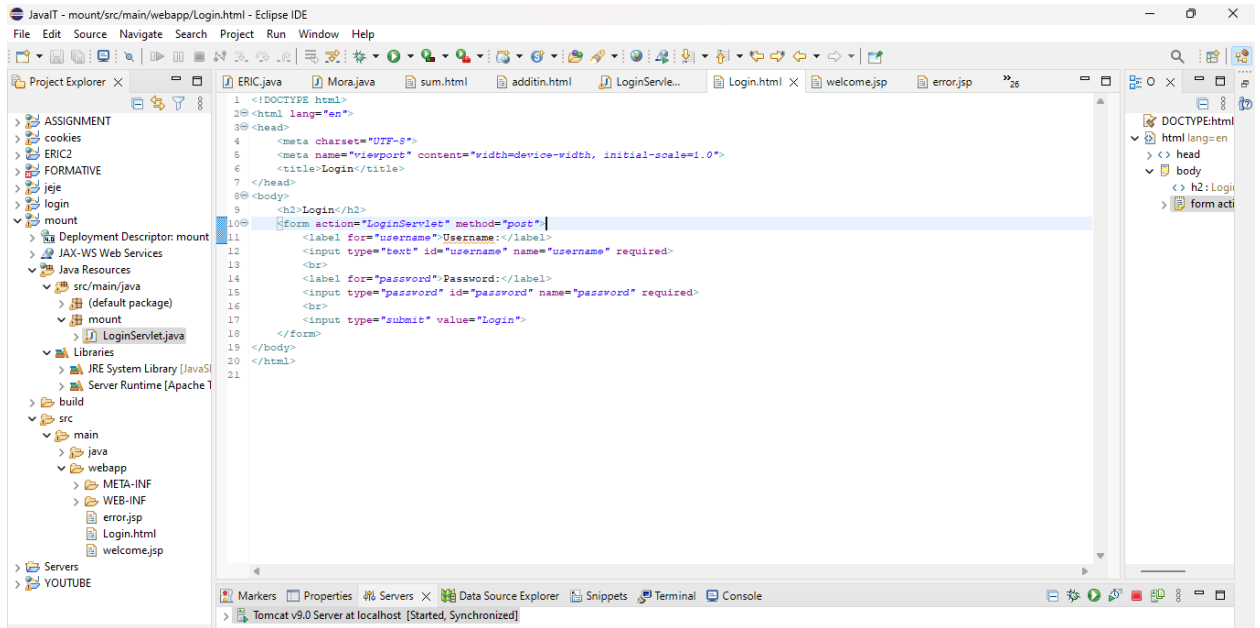


```

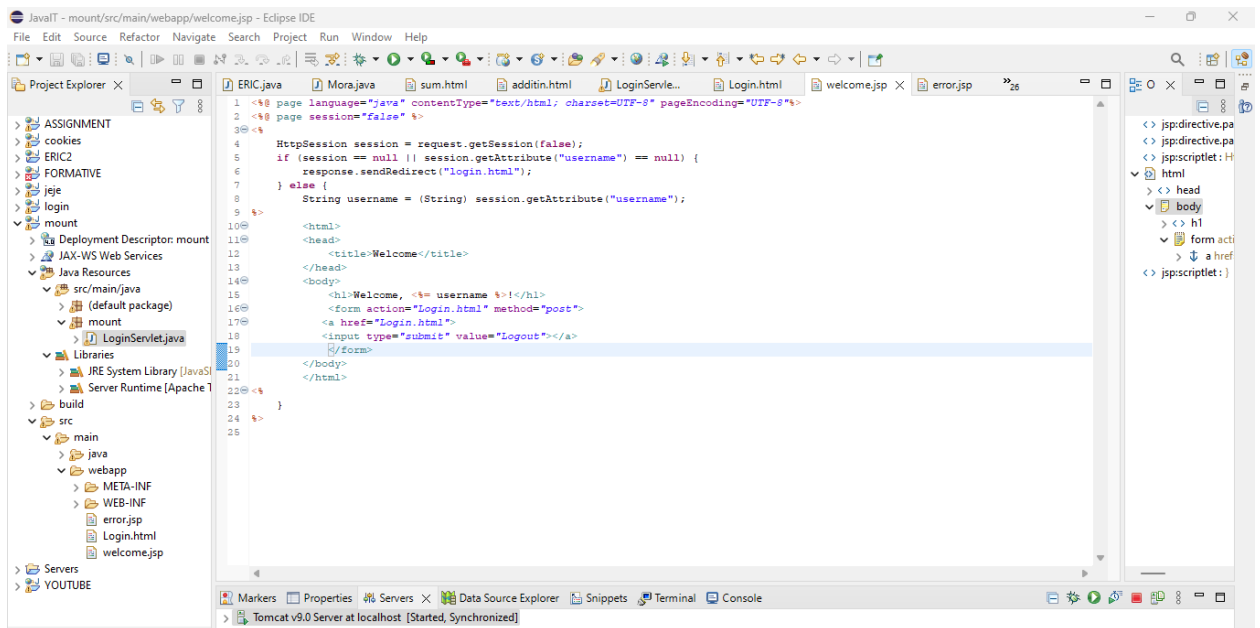
1 package mount;
2
3 import java.io.IOException;
4
5 @WebServlet("/LoginServlet")
6 public class LoginServlet extends HttpServlet {
7     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
8         String username = request.getParameter("username");
9         String password = request.getParameter("password");
10
11         // Dummy validation (replace with real validation logic)
12         if ("IPRC HUYE".equals(username) && "admin".equals(password)) {
13             HttpSession session = request.getSession();
14             session.setAttribute("username", username);
15             response.sendRedirect("welcome.jsp");
16         } else {
17             response.sendRedirect("error.jsp");
18         }
19     }
20 }

```

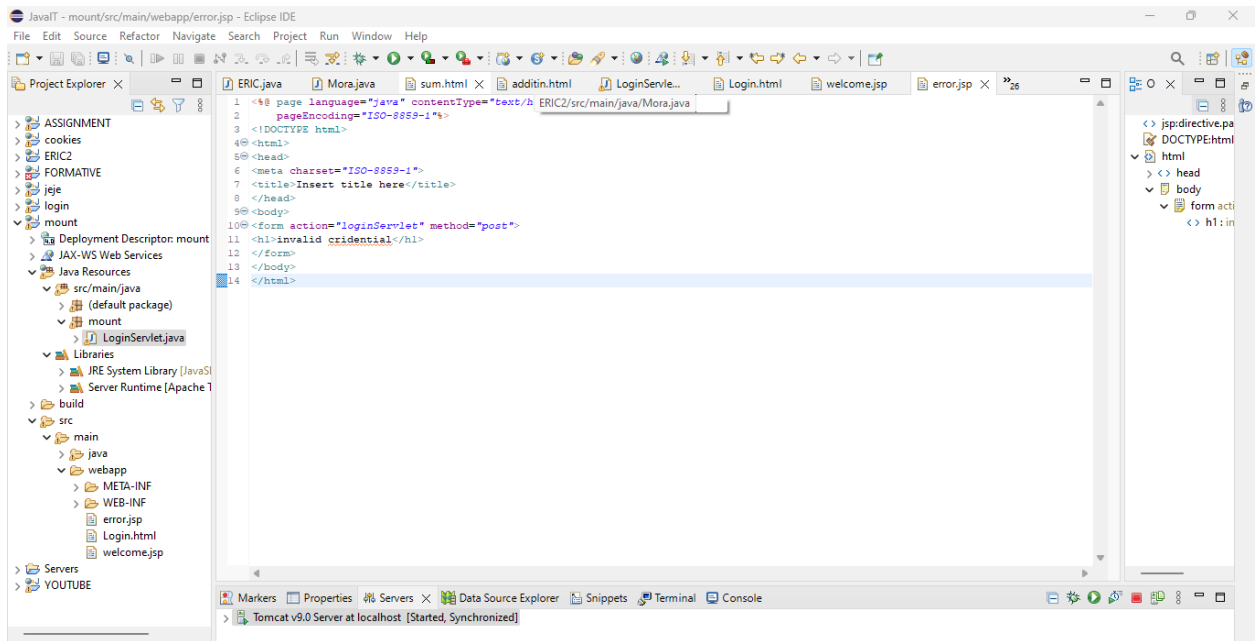
SOURCE CODE TWO



SOURCE CODE THREE

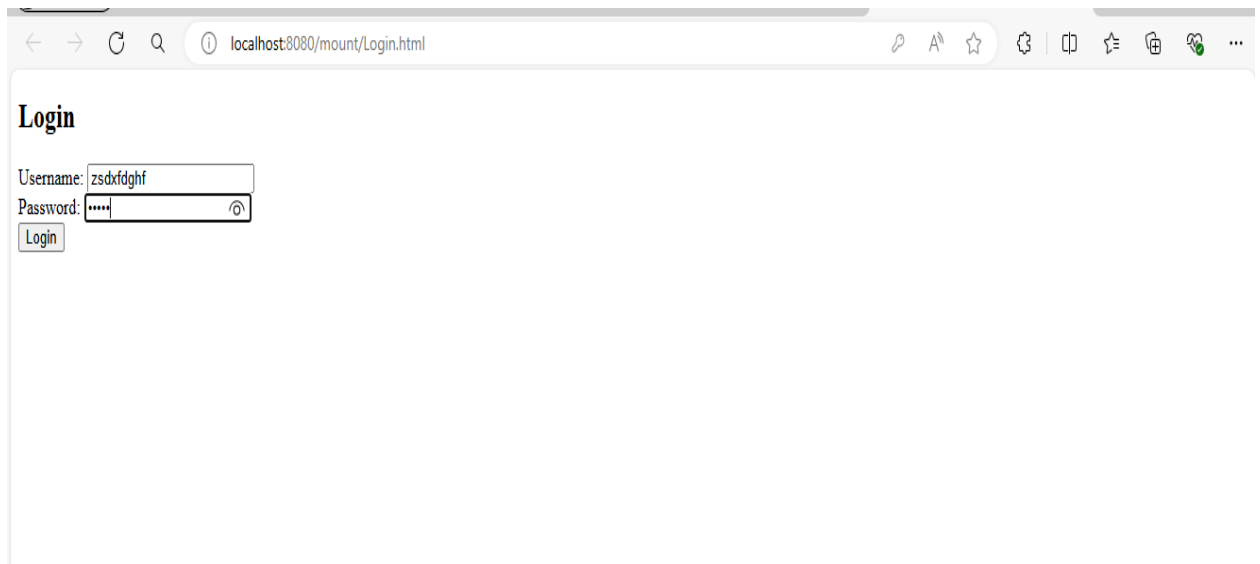


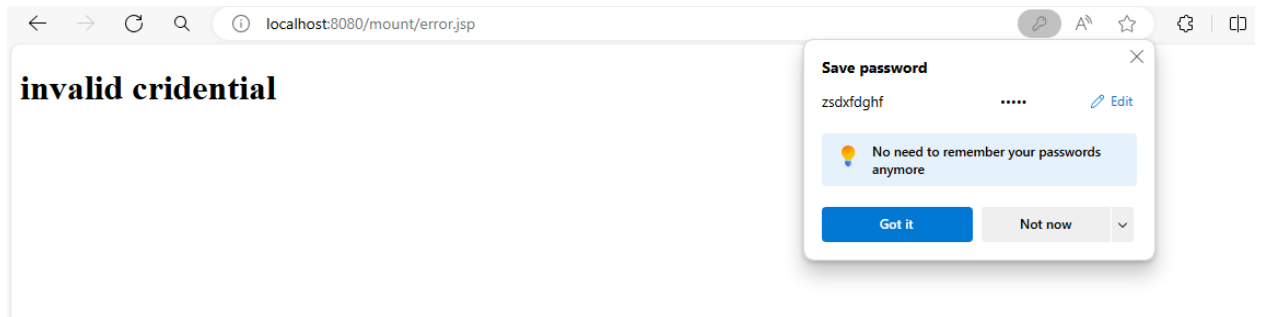
SOURCE CODE FOUR



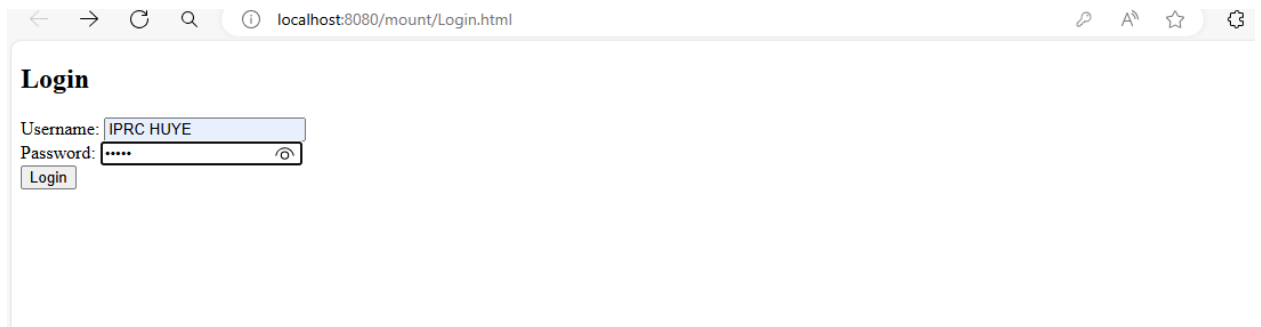
OUT PUT ALL

When credential invalid





When credential correct



When a user clicks on logout, he can be back on login form page

