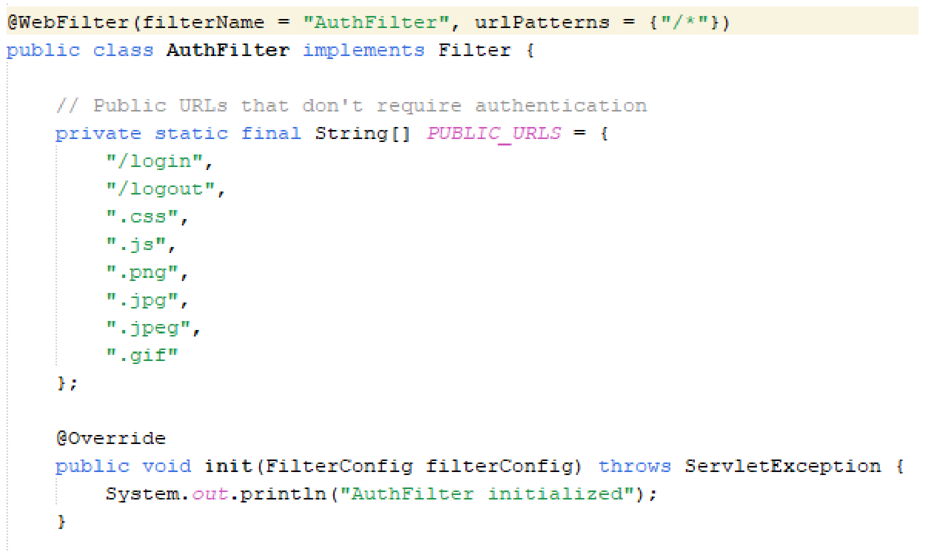
ITCSIU24022 – TRẦN MINH ĐỨC

LAB SESSION 06 (Contd)

EXERCISE 5: AUTHENTICATION FILTER

Task 5.1: Create AuthFilter

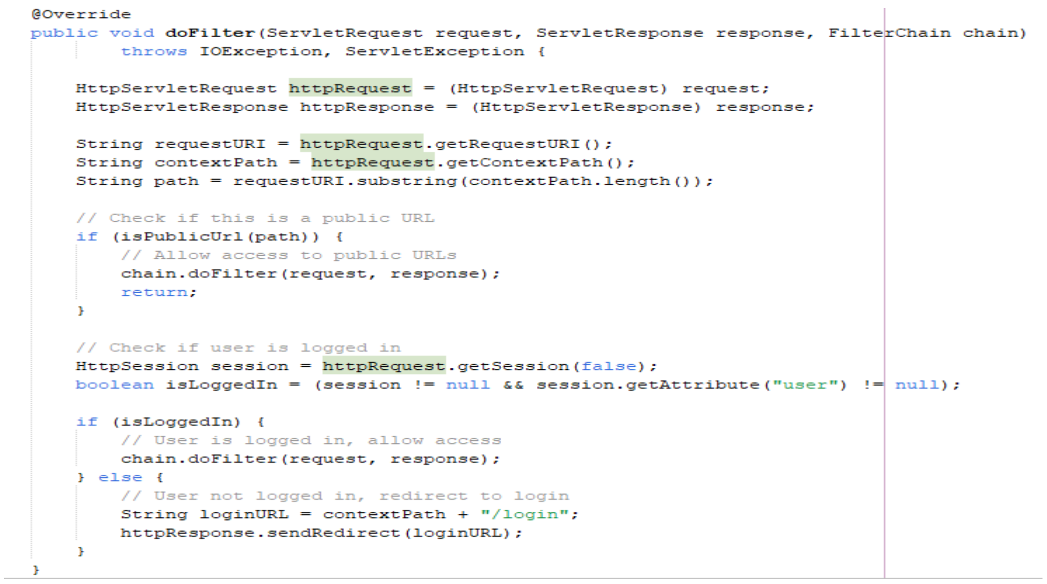
Request all types into the server except the assets. – urlPatterns = (“/*”)



```
@WebFilter(filterName = "AuthFilter", urlPatterns = {"/*"})
public class AuthFilter implements Filter {

    // Public URLs that don't require authentication
    private static final String[] PUBLIC_URLS = {
        "/login",
        "/logout",
        ".css",
        ".js",
        ".png",
        ".jpg",
        ".jpeg",
        ".gif"
    };

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("AuthFilter initialized");
    }
```

Extract the request path (Example)

- Full URL: /StudentApp/student?action=list
- contextPath: /StudentApp
- path becomes: /student?action=list



```
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    HttpServletResponse httpResponse = (HttpServletResponse) response;

    String requestURI = httpRequest.getRequestURI();
    String contextPath = httpRequest.getContextPath();
    String path = requestURI.substring(contextPath.length());

    // Check if this is a public URL
    if (isPublicUrl(path)) {
        // Allow access to public URLs
        chain.doFilter(request, response);
        return;
    }

    // Check if user is logged in
    HttpSession session = httpRequest.getSession(false);
    boolean isLoggedIn = (session != null && session.getAttribute("user") != null);

    if (isLoggedIn) {
        // User is logged in, allow access
        chain.doFilter(request, response);
    } else {
        // User not logged in, redirect to login
        String loginURL = contextPath + "/login";
        httpResponse.sendRedirect(loginURL);
    }
}
```

isPublicUrl : allows all asset filepath to be accessed without going through session

Now check the user if exists by checking its session properties:

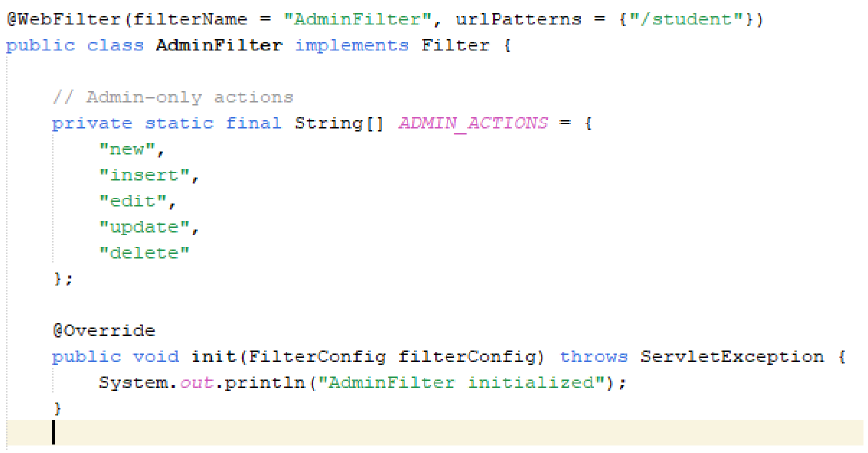
- session exists
- session contains "user" attribute

If true → user is logged in → allow request
If false → user is not logged in → redirect to /login by sendRedirect(loginURL);

EXERCISE 6: ADMIN AUTHORIZATION FILTER

Task 6.1: Create AdminFilter

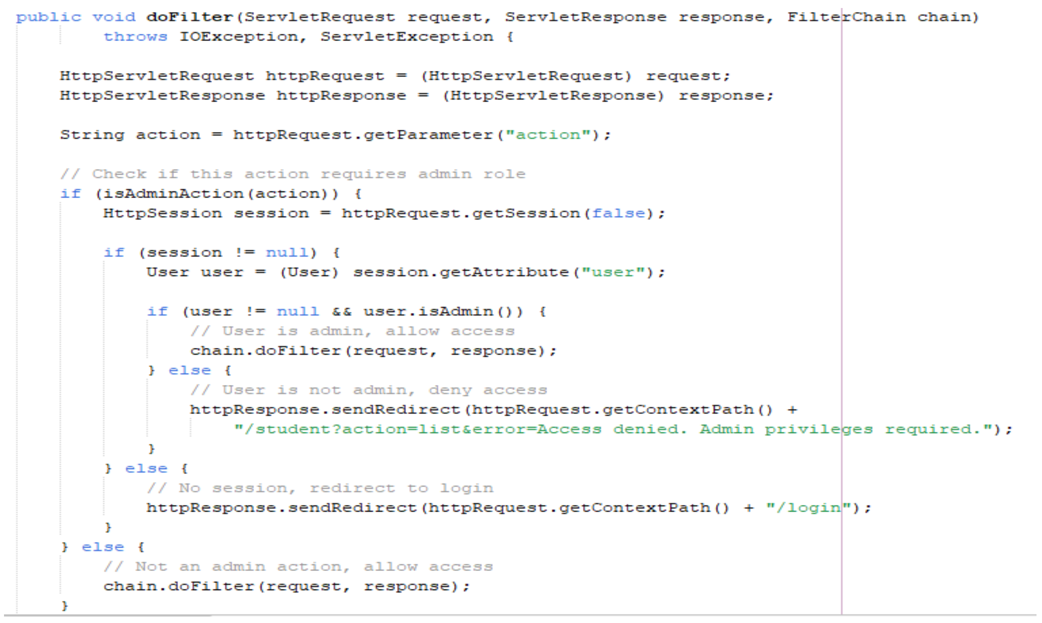
@WebFilter(urlPatterns = {"/student"}) : Only requests hitting /student servlet go through this filter.



```java
@WebFilter(filterName = "AdminFilter", urlPatterns = {"/student"})
public class AdminFilter implements Filter {

    // Admin-only actions
    private static final String[] ADMIN_ACTIONS = {
        "new",
        "insert",
        "edit",
        "update",
        "delete"
    };

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("AdminFilter initialized");
    }
    
```

Extract the action parameter

Example: /student?action=list, /student?action=new, /student?action=delete&id=5

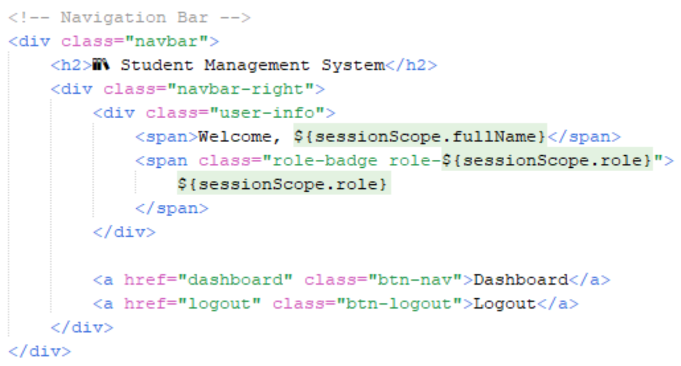


```java
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    HttpServletResponse httpResponse = (HttpServletResponse) response;

    String action = httpRequest.getParameter("action");

    // Check if this action requires admin role
    if (isAdminAction(action)) {
        HttpSession session = httpRequest.getSession(false);

        if (session != null) {
            User user = (User) session.getAttribute("user");

            if (user != null && user.isAdmin()) {
                // User is admin, allow access
                chain.doFilter(request, response);
            } else {
                // User is not admin, deny access
                httpResponse.sendRedirect(httpRequest.getContextPath() +
                    "/student?action=list&error=Access denied. Admin privileges required.");
            }
        } else {
            // No session, redirect to login
            httpResponse.sendRedirect(httpRequest.getContextPath() + "/login");
        }
    } else {
        // Not an admin action, allow access
        chain.doFilter(request, response);
    }
```

Check if the action requires admin privileges – if (isAdminAction(action)).

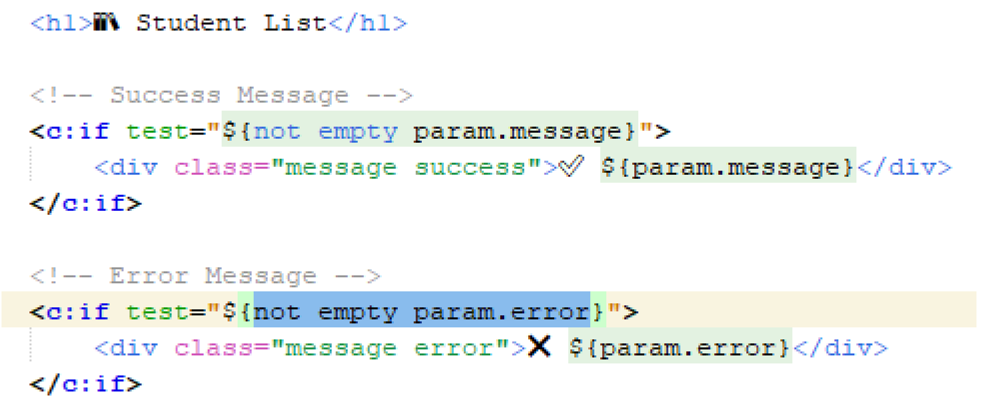
If the action belongs to admin, check the session to verify the account – if yes, do the chain request / response (allow access), otherwise redirect with an error message.

EXERCISE 7: ROLE-BASED UI



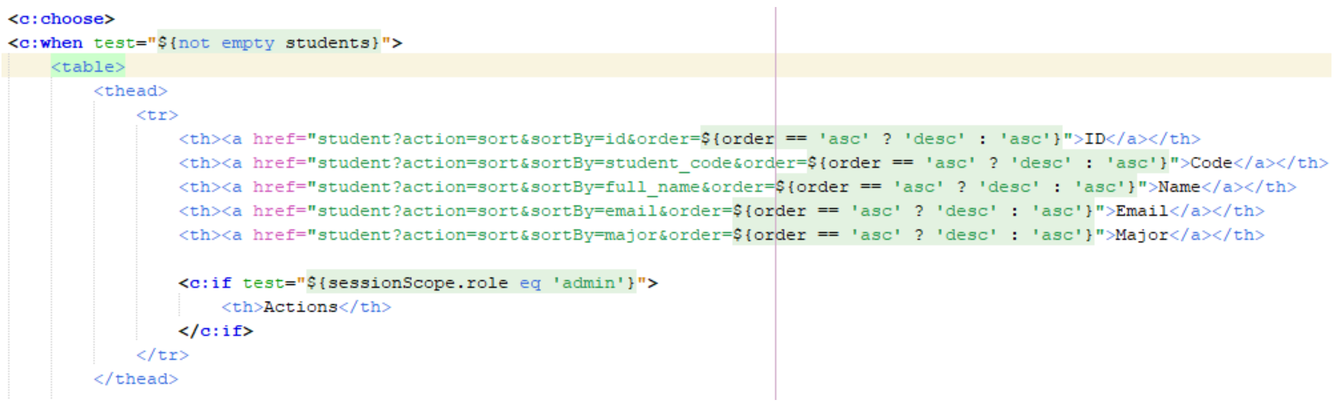
```
<!-- Navigation Bar -->
<div class="navbar">
    <h2>📚 Student Management System</h2>
    <div class="navbar-right">
        <div class="user-info">
            <span>Welcome, ${sessionScope.fullName}</span>
            <span class="role-badge role-${sessionScope.role}">
                ${sessionScope.role}
            </span>
        </div>

        <a href="dashboard" class="btn-nav">Dashboard</a>
        <a href="logout" class="btn-logout">Logout</a>
    </div>
</div>
```

Navigation Bar With User Info: Display username who logged in from session, additionally show role badges

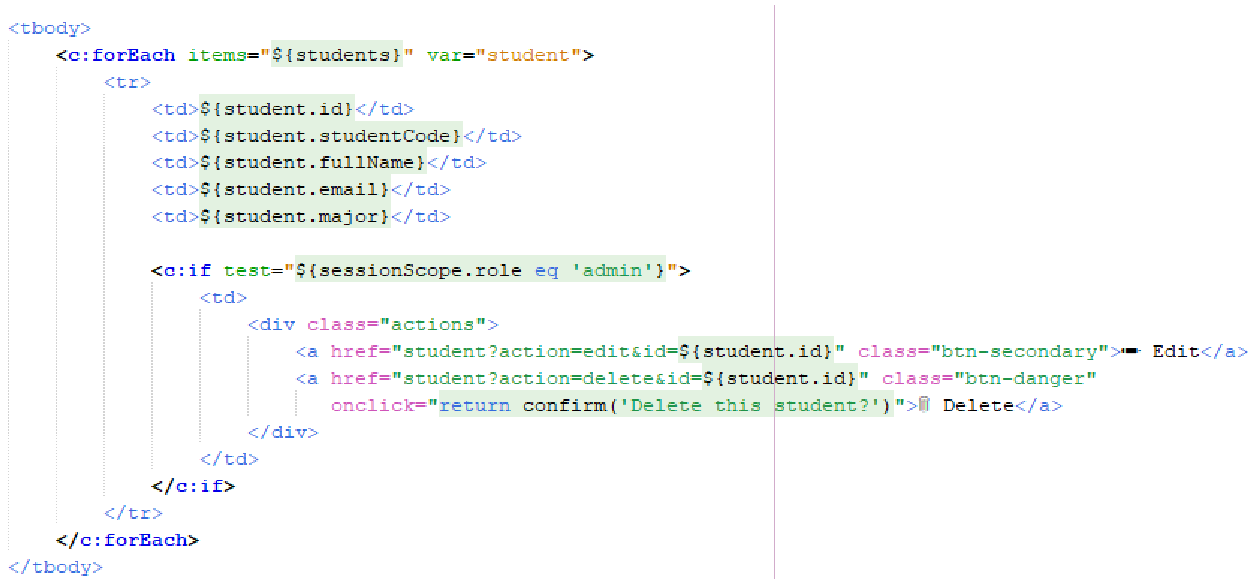


```
<h1>📚 Student List</h1>

<!-- Success Message -->
<c:if test="${not empty param.message}">
    <div class="message success">✅ ${param.message}</div>
</c:if>

<!-- Error Message -->
<c:if test="${not empty param.error}">
    <div class="message error">❌ ${param.error}</div>
</c:if>
```

Error message display

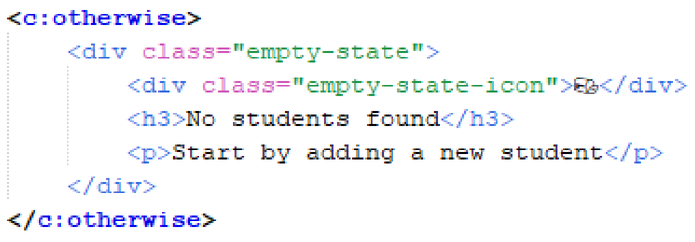


```
<c:choose>
<c:when test="${not empty students}">
    <table>
        <thead>
            <tr>
                <th><a href="student?action=sort&sortBy=id&order=${order == 'asc' ? 'desc' : 'asc'}">ID</a></th>
                <th><a href="student?action=sort&sortBy=student_code&order=${order == 'asc' ? 'desc' : 'asc'}">Code</a></th>
                <th><a href="student?action=sort&sortBy=full_name&order=${order == 'asc' ? 'desc' : 'asc'}">Name</a></th>
                <th><a href="student?action=sort&sortBy=email&order=${order == 'asc' ? 'desc' : 'asc'}">Email</a></th>
                <th><a href="student?action=sort&sortBy=major&order=${order == 'asc' ? 'desc' : 'asc'}">Major</a></th>

                <c:if test="${sessionScope.role eq 'admin'}">
                    <th>Actions</th>
                </c:if>
            </tr>
        </thead>
```



```
<tbody>
    <c:forEach items="${students}" var="student">
        <tr>
            <td>${student.id}</td>
            <td>${student.studentCode}</td>
            <td>${student.fullName}</td>
            <td>${student.email}</td>
            <td>${student.major}</td>

            <c:if test="${sessionScope.role eq 'admin'}">
                <td>
                    <div class="actions">
                        <a href="student?action=edit&id=${student.id}" class="btn-secondary">✏ Edit</a>
                        <a href="student?action=delete&id=${student.id}" class="btn-danger"
                           onclick="return confirm('Delete this student?')">🗑 Delete</a>
                    </div>
                </td>
            </c:if>
        </tr>
    </c:forEach>
</tbody>
```

Perform the condition when – otherwise <c:when test = “${not empty students}”>. Show messages when no students is found.



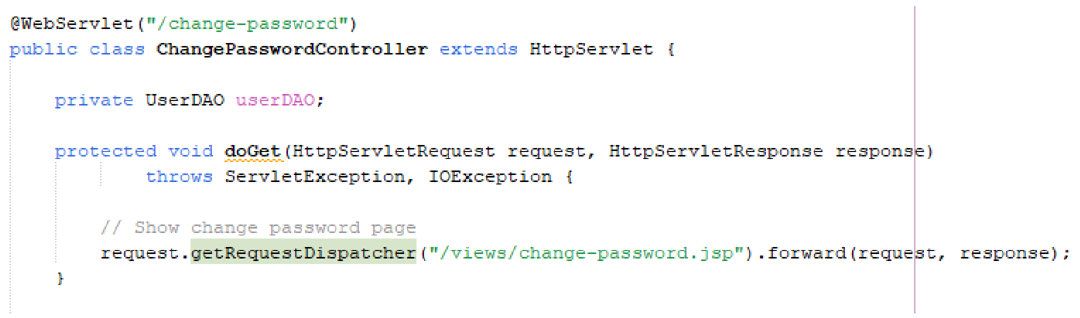
```
<c:otherwise>
    <div class="empty-state">
        <div class="empty-state-icon">📭</div>
        <h3>No students found</h3>
        <p>Start by adding a new student</p>
    </div>
</c:otherwise>
```

Check condition on whether the session user is admin: ${sessionScope.role eq ‘admin’} - JSTL equality check, which only show the header Actions and buttons of Edit and Delete towards admins.

Edit and Delete buttons wrapped in <c:if>, subsequentially perform the visibility of Edit & Delete to each students <c: forEach items=”${students}” var = “student”>. The form will ask for confirmation of deleting this student by promping a message.

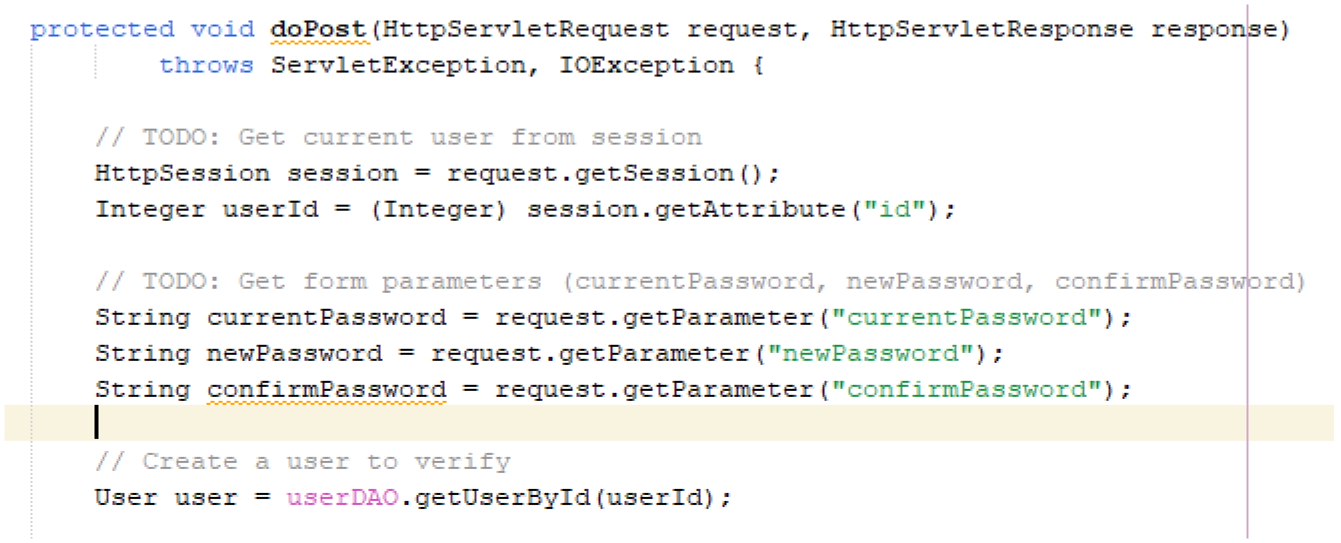
EXERCISE 8: CHANGE PASSWORD FUNCTIONALITY

Show the change password page – forward a request internally on the server to another resource (JSP, HTML, another servlet, etc.).



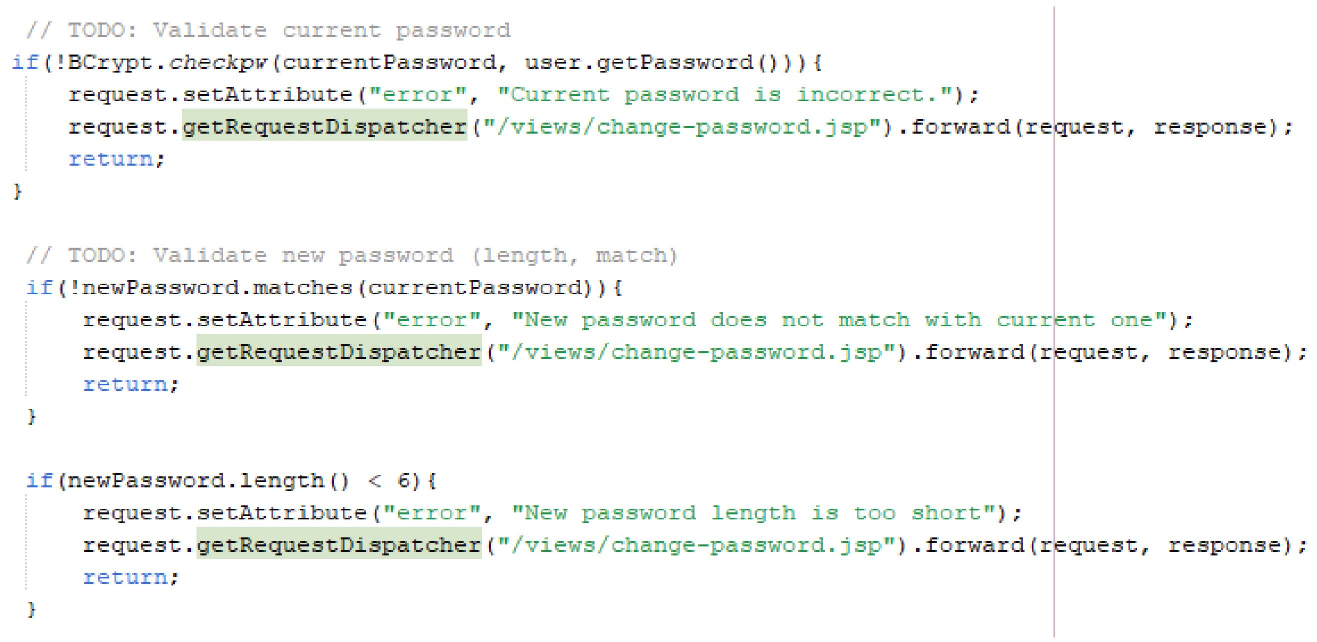
```java
@WebServlet("/change-password")
public class ChangePasswordController extends HttpServlet {

    private UserDAO userDAO;

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

        // Show change password page
        request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
    }
```

Create a session to verify the user information and get all necessary parameters (old and new – confirmed password). All changes must go through the if – else statement .



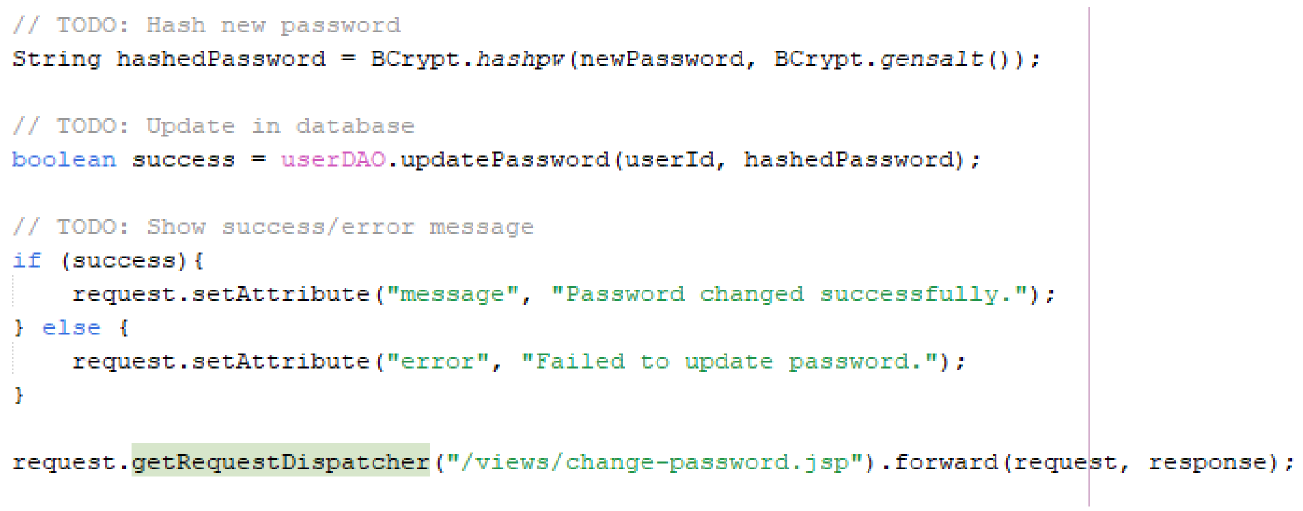
```java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // TODO: Get current user from session
    HttpSession session = request.getSession();
    Integer userId = (Integer) session.getAttribute("id");

    // TODO: Get form parameters (currentPassword, newPassword, confirmPassword)
    String currentPassword = request.getParameter("currentPassword");
    String newPassword = request.getParameter("newPassword");
    String confirmPassword = request.getParameter("confirmPassword");

    // Create a user to verify
    User user = userDAO.getUserById(userId);
```

Validate the current matches and comparison checks before checking its length (weak or not).



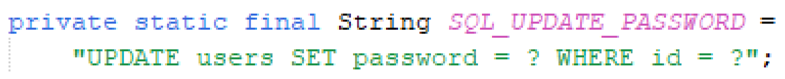
```java
 // TODO: Validate current password
if(!BCrypt.checkpw(currentPassword, user.getPassword())){
    request.setAttribute("error", "Current password is incorrect.");
    request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
    return;
}

 // TODO: Validate new password (length, match)
 if(!newPassword.matches(currentPassword)){
     request.setAttribute("error", "New password does not match with current one");
     request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
     return;
 }

 if(newPassword.length() < 6){
     request.setAttribute("error", "New password length is too short");
     request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
     return;
 }
```

Encoding the new password and perform SQL execution (thourgh JDBC connection), embedded with a boolean tracker to show success / message.

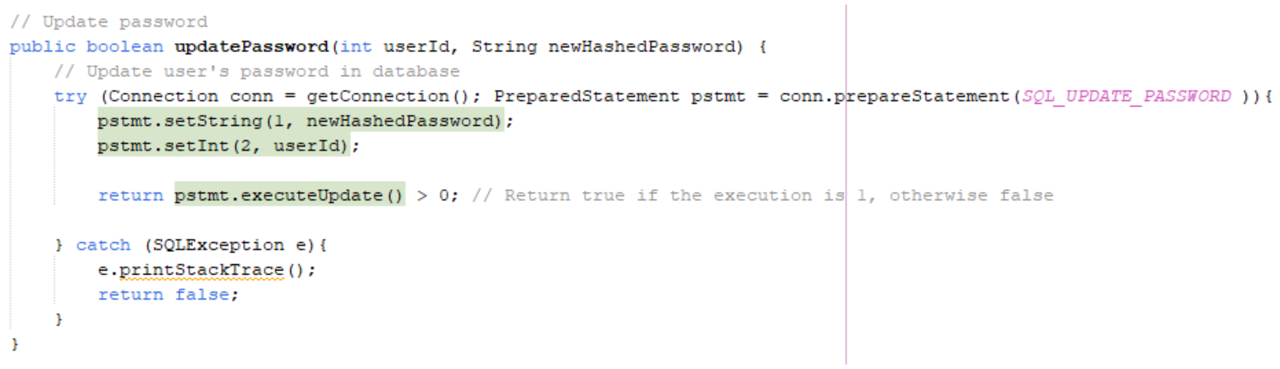


```java
// TODO: Hash new password
String hashedPassword = BCrypt.hashpw(newPassword, BCrypt.gensalt());

// TODO: Update in database
boolean success = userDAO.updatePassword(userId, hashedPassword);

// TODO: Show success/error message
if (success){
    request.setAttribute("message", "Password changed successfully.");
} else {
    request.setAttribute("error", "Failed to update password.");
}

request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
```

Update Password in UserDAO



```java
private static final String SQL_UPDATE_PASSWORD =
    "UPDATE users SET password = ? WHERE id = ?";
```



```java
// Update password
public boolean updatePassword(int userId, String newHashedPassword) {
    // Update user's password in database
    try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(SQL_UPDATE_PASSWORD )){
        pstmt.setString(1, newHashedPassword);
        pstmt.setInt(2, userId);

        return pstmt.executeUpdate() > 0; // Return true if the execution is 1, otherwise false

    } catch (SQLException e){
        e.printStackTrace();
        return false;
    }
}
```

change-password.jsp – Password changing dialog with the error / message display by getting attribute from request, embedded by Java field <%%>.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Change Password</title>

        <!-- Bootstrap theme -->
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

        <!--  -->
        <style>
            .password-box {
                max-width: 400px;
                margin: 40px auto;
            }
        </style>

    </head>
    <body>
        <div class = "password-box">

            <h3 class="text-center mb-4">Change Password</h3>

            <!-- Display change status -->
            <%
                String error = (String) request.getAttribute("error");
                String message = (String) request.getAttribute("message");
                if (error != null){ // If error raises
            %>

            <div class="alert alert-danger"><%= error %></div>

            <% } else if (message != null) { %>

            <div class="alert alert-success"><%= message %></div>

            <% } %>

            <form action="change-password" method="post">
                <input type="password" name="currentPassword" placeholder="Current Password">
                <input type="password" name="newPassword" placeholder="New Password">
                <input type="password" name="confirmPassword" placeholder="Confirm Password">
                <button type="submit">Change Password</button>
            </form>
        </div>
    </body>
</html>
```