Lab Sheet: Enterprise Java Web App – Asynchronous Interaction, Filters & Validation

Module: Development of Enterprise Applications (DEA)
Topic: Advanced Library Management System with AJAX, Filters, and Validations (MVC)
Duration: 3 Hours

🎯 Learning Outcomes

By the end of this lab, students will:

- Use **AJAX** for asynchronous operations (e.g., live book search).
- Implement **filters** for access control (e.g., restrict Admin pages).
- Perform **form validation** on both client (JavaScript) and server side.
- Integrate all these features within the MVC design pattern.

📘 Scenario: Real-Time Enhanced Library System

You are continuing the Library System project by adding:

1. **Live Book Search** with AJAX.
2. **Access Control** using a Servlet Filter (prevent students from accessing admin pages).
3. **Client-side + Server-side validations** for book and user forms.

🛠️ Prerequisites

- Previous lab's system completed.
- Database configured with books, users, and reservations tables.

🗂️ Updated Folder Structure

css

WebContent/

├— login.jsp

```
├── dashboard.jsp
├── searchBook.jsp
├── bookList.jsp
├── addBook.jsp
├── error.jsp
├── js/
│   └── validation.js
├── ajax/
│   └── liveSearch.jsp

src/
├── controller/
│   └── LoginServlet.java
│   └── BookServlet.java
│   └── SearchBookServlet.java
│   └── LogoutServlet.java
│   └── filter/
│       └── AuthFilter.java
├── dao/
├── model/
```

📝 Step-by-Step Instructions

1️⃣ Live Book Search Using AJAX

- **Frontend (searchBook.jsp)**

jsp

```
<script src="js/validation.js"></script>
```

```
<input type="text" id="searchBox" onkeyup="searchBooks()" placeholder="Search books...">
<div id="results"></div>
```

- **JavaScript (validation.js)**

```
javascript


function searchBooks() {
   let query = document.getElementById("searchBox").value;
   let xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
         document.getElementById("results").innerHTML = this.responseText;
      }
   };
   xhttp.open("GET", "ajax/liveSearch.jsp?q=" + query, true);
   xhttp.send();
}
```

- **liveSearch.jsp**

```
jsp


<%@ page import="java.sql.*, your.package.dao.BookDAO" %>
<%
   String q = request.getParameter("q");
   List<Book> books = new BookDAO().searchBooks(q);
   for (Book book : books) {
%>
      <p><%= book.getTitle() %> - <%= book.getAuthor() %></p>
<% } %>
```

- **BookDAO.java** (add searchBooks(String query))

2️⃣ Add Filter to Restrict Access

- **Create AuthFilter.java**

java

```
@WebFilter("/addBook.jsp")
public class AuthFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        HttpSession session = ((HttpServletRequest) request).getSession();
        User user = (User) session.getAttribute("user");
        if (user == null || !"Admin".equals(user.getRole())) {
            ((HttpServletResponse) response).sendRedirect("error.jsp");
        } else {
            chain.doFilter(request, response);
        }
    }
}
```

3️⃣ Add Validations to Forms

- **addBook.jsp**

jsp

```
<form action="addBook" method="post" onsubmit="return validateBookForm()">
    Title: <input type="text" name="title" id="title"/><br/>
    Author: <input type="text" name="author" id="author"/><br/>
    <input type="submit" value="Add Book"/>
</form>
```

- **JavaScript (validation.js)**

javascript

```
function validateBookForm() {
    let title = document.getElementById("title").value;
    let author = document.getElementById("author").value;
    if (title === "" || author === "") {
        alert("Both title and author must be filled out.");
        return false;
    }
    return true;
}
```

- **Server-side check (BookServlet.java)**

```
java

if (title == null || title.trim().isEmpty() || author == null || author.trim().isEmpty()) {
    request.setAttribute("error", "Invalid input!");
    request.getRequestDispatcher("addBook.jsp").forward(request, response);
    return;
}
```

🧪 **Testing Instructions**

1. Log in with a Student account and attempt to access addBook.jsp → should redirect to error.
2. Log in as Admin and add books using valid and invalid inputs.
3. Perform a live search and verify that results update as you type.
4. Check that invalid form inputs trigger JS alerts and prevent submission.

🧠 **Extra Learning Extension (Bonus)**

- Add CAPTCHA to login page.
- Allow dynamic drop-down filters in the search page.
- Use JSON instead of JSP in AJAX response and parse via JavaScript.

- Integrate Bootstrap for better styling.

✅ **Evaluation Criteria (Total: 60 Marks)**

Component	Marks
AJAX live search	15
Filter implementation (AuthFilter)	10
JavaScript validation	10
Server-side input check	10
Folder structure & MVC adherence	10
Code readability & reusability	5
Total	**60**