JSP/SERVLET ADVANCED

Instructor: DieuNT1

Learning Goals

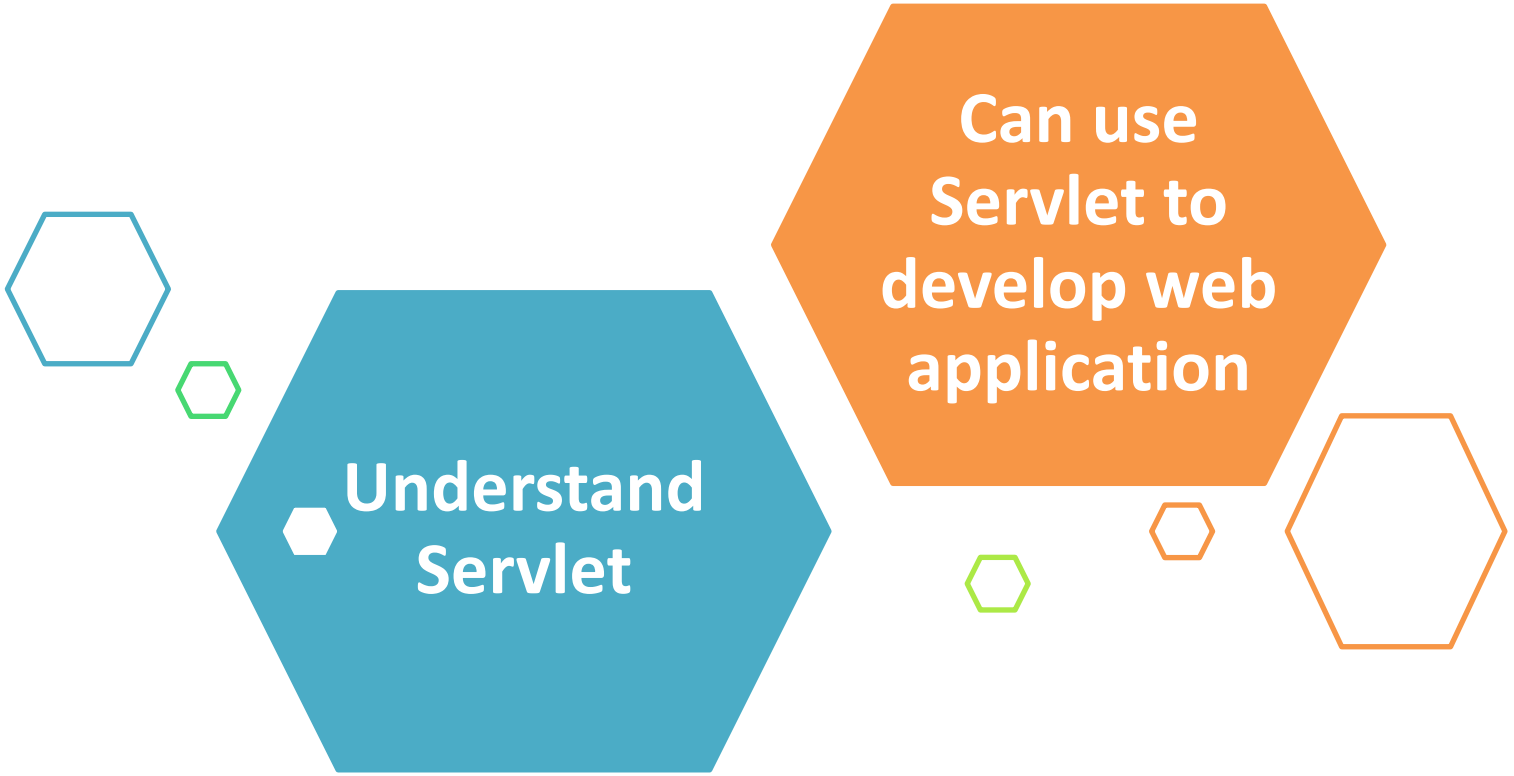


Table Content

- Exception Handling
- JSP/ Servlet Session Tracking
- Servlet FILTER
- Q&A

Section 1

EXCEPTION HANDLING

Exception Handling

- **Exceptions** are errors that can occur in a JSP page.
- **isErrorPage** attribute of page directive[chỉ thị] is used to construct an error page.
- This attribute is used with the page directive at the **beginning of the JSP page.**
  - ✓ Value of this attribute is either **true** or **false.**
- If you want to handle the exceptions that occurs on the execution of the JSP page you may use the page directive attribute **isErrorPage="true“.**
- **Syntax :**

```
<%@ page isErrorPage=“true” %>
```

Exception Handling

- The JSP page traps and handles request time errors.
- Unhandled exceptions are **forwarded to the error page**
- **Syntax:**

```
<%@ page errorPage="errorpage.jsp" %>
```

Exception Handling

Translation time

- ✓ Occurs when the JSP source file is **converted to servlets** class file. The JSP engine handles translation time errors.
- ✓ This translation can happen:
  - After the JSP has been deployed into the JSP container and before the client requests the JSP.
  - When the client requests a JSP.

In the first case, error processing is implementation-dependent, and the JSP specification **does not cover** this.

For HTTP protocols, the error status code 500 is returned.

Exception Handling

❖ Request time:

- ✓ Occurs during the **processing of the request**. Request time errors are the runtime errors that throw exceptions.

Code snippet

```
<%@ page isErrorPage="true"%>
<html>
<head>
<title>Error Page</title>
</head>
<body>
     <h3>Due to following reasons an error has occurred</h3>
     <ul>
     <li><%=exception.getClass()%></li>
     <li><%=exception.getMessage()%></li>
     </ul>
</body>
</html>
```

Makes JSP page an error handler

Returns error message

ErrorPage.jsp

Exception Handling

Code snippet

```
<%@ page errorPage="ErrorPage.jsp"%>
<html>
<head>
<title>Form</title>
</head>
<body>
	<form>
		<table>
			<tr>
				<td>Enter a number :</td>
				<td><input type="text" name="number" /></td>
				<td><input type="submit" value="Submit" /></td>
			</tr>
		</table>
	</form>
	<%
	    String num = request.getParameter("number");
	    if (num != null) {
	        String number = num.trim();
	        int no = Integer.parseInt(number);
	        int value = 100 / no;
	    }
	%>
</body>
</html>
```

Forwards the unhandled exception to errorpage.jsp

Code occurs error

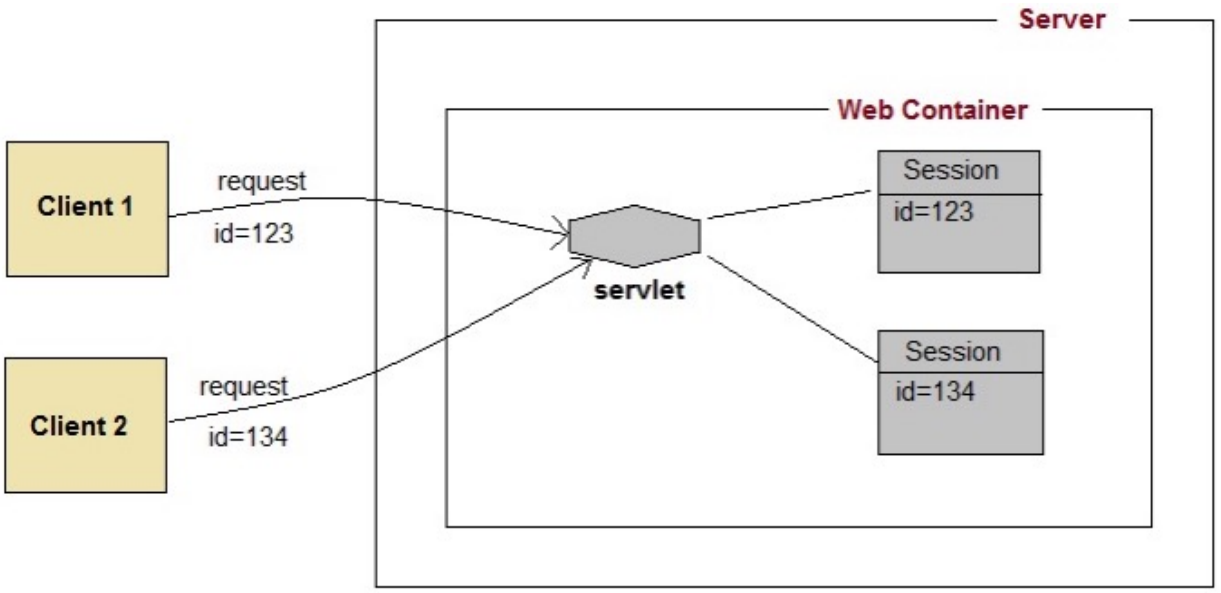
ExceptionHandlingPage.jsp: to transfer control to the error page

Section 2

SESSION TRACKING

Introduction

- **HTTP protocol** and **Web Servers** are stateless, what it means is that for **web server every request is a new request** to process.
  - ✓ But sometimes in web applications, we should know **who the client** is and process the request accordingly[phù hợp].
- **Session Management** is a mechanism used by the Web container to store session information for a particular user:
  - ✓ **Cookies**
  - ✓ **Hidden form field**
  - ✓ **URL Rewriting**
  - ✓ **HttpSession**



Advantages

- **Remember** user **IDs** and **password**.
- To **track** visitors on a Web site for better service and new features.
- Cookies enable **efficient** ad processing.

Disadvantages

- ✓ The size and number of cookies stored are **limited**.
- ✓ Personal information is **exposed** to the other users.
- ✓ Cookies fails to work if the **security** level is set too high in the Internet browser.

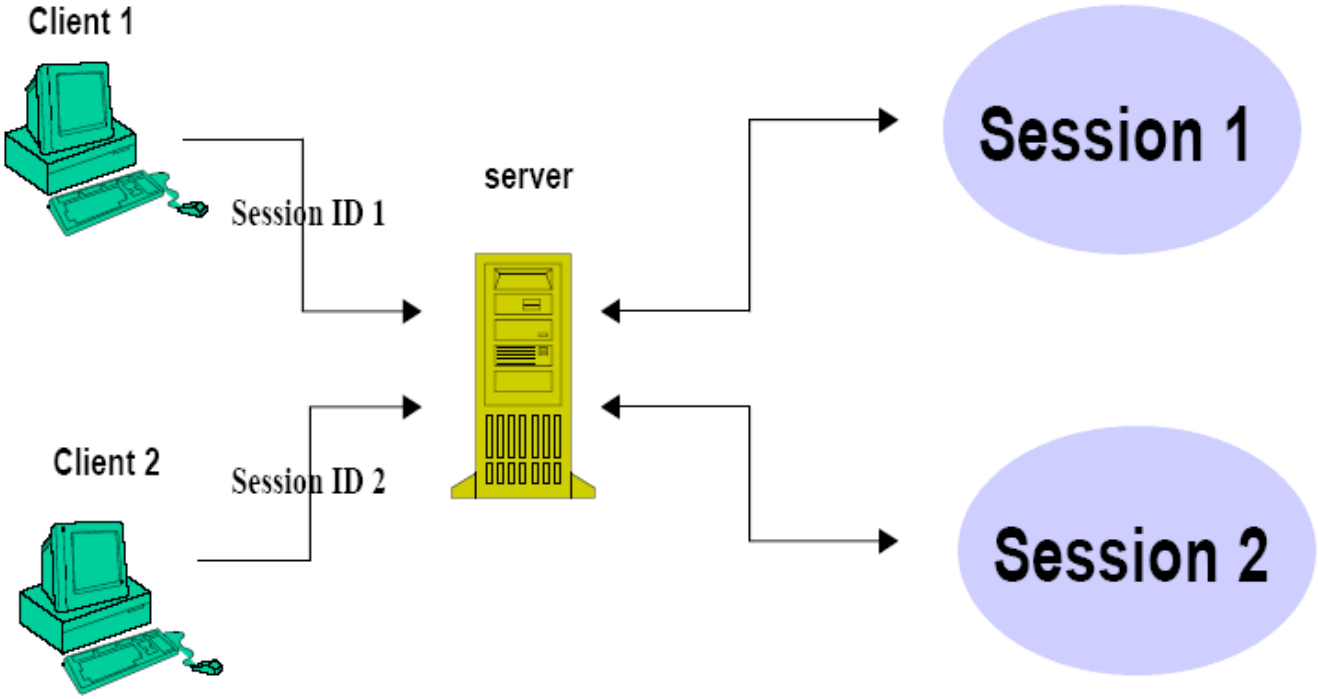
Servlet `CookieServlet`

- ✓ Handles both `get` and `post` requests

Session Tracking

HttpSession

- The servlet API has a built-in support for session tracking.
- Session objects live on the server.
  - ✓ Each user has associated an HttpSession object—one user/session
  - ✓ HttpSession object operates like a hashtable

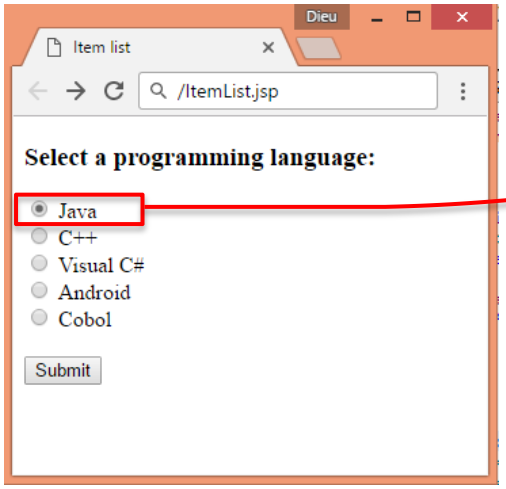


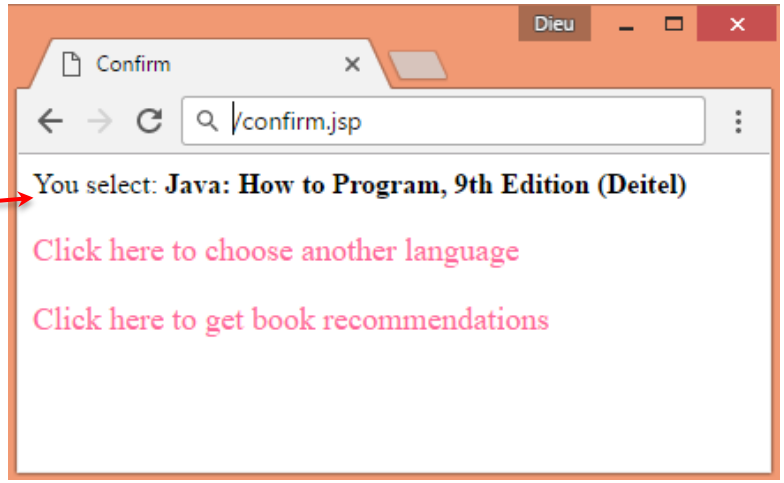
Using Session Object

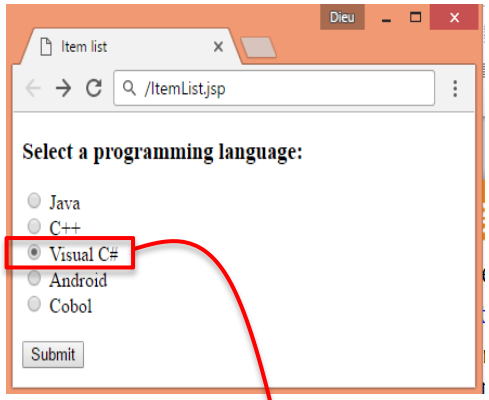
- To get a user's existing or new session object:
  - ✓ HttpSession session = request.getSession(true);
  - ✓ "true" means the server should create a new session object if necessary
- To store or retrieve an object in the session:
  - ✓ Stores values: setAttribute("cartItem", cart);
  - ✓ Retrieves values: getAttribute("cartItem");

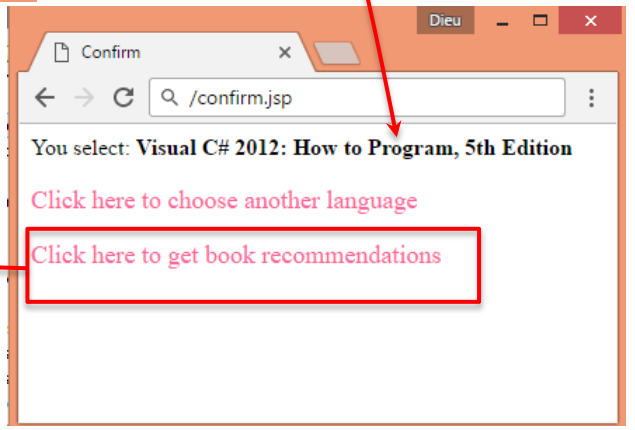
Session Tracking

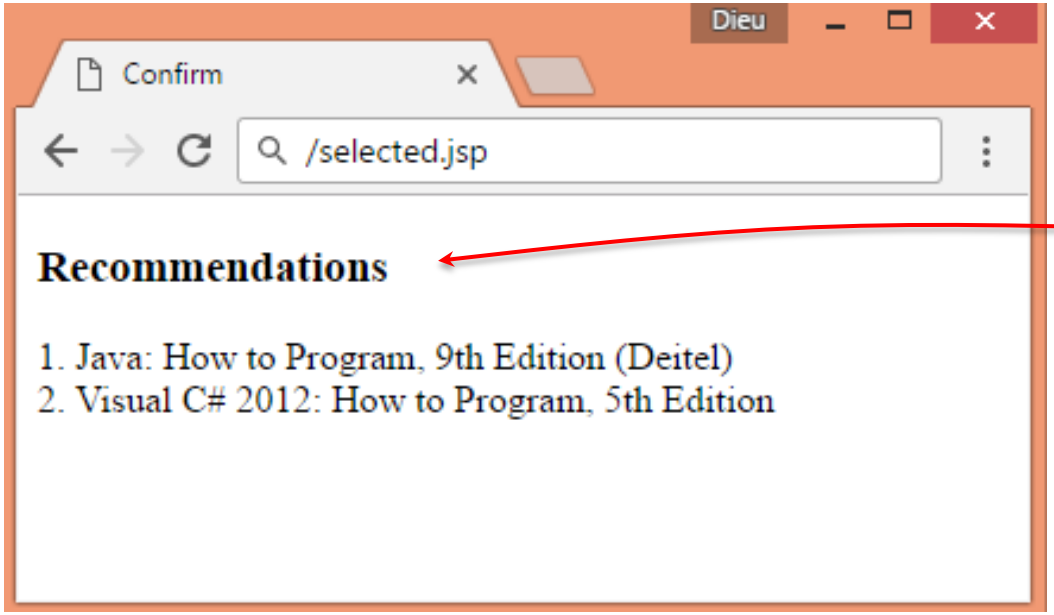
HttpSession











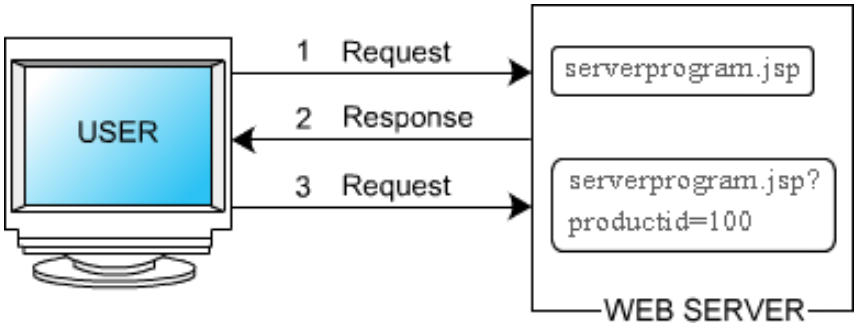
- Append a **token** or **identifier to the URL.** We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&

- When the **user clicks the hyperlink,** the **parameter** name/value pairs **will be passed to the server.** We can use **getParameter()** method to obtain a parameter value.
- **Advantage of URL Rewriting**
  - ✓ It will always work whether **cookie is disabled or not** (browser independent).
  - ✓ Extra **form submission is not required on each pages.**
- **Disadvantage of URL Rewriting**
  - ✓ It will work **only with links.**
  - ✓ It can send Only **textual information.**

Session Tracking

URL Rewriting



The session ID is encoded in the URLs that are created by the JSP pages

```
<b>Search results for books</b>
<form method="post" action="serverprogram.jsp">
     // Provides check box for different products
     <input type="checkbox" name="productID" value="100">
     CD MP3 Converter Kit For Your CAR<br>
     <input type="checkbox“ name="productID" value="101">
     Front Loading Car MP3/CD Player With Anti Shock    Memory and FM<br>
     <input type="checkbox“ name="productID" value="102">
     CAR/Home DVD/VCD/MP3 Playerwith anti shock for     Indian Roads<br>
     // Submits the user input to URL
     <input type="submit“ name="Submit" value="Add to Cart"><br>
</form>
```

URL of server side program

Session Tracking

URL Rewriting

```
// URL for server side program after the user selects a product
// and goes to another page
<form method="post" action="serverprogram.jsp?productID=102">
    // Provides check box for different products
    <input type="checkbox" name="productID" value="150">
           DVD Player with built in Amplifier <br>
    <input type="checkbox" name="productID" value="160">
           Ultra Slim DVD Player Multi Region 5.1 Digital<br>
    // Submits input to the URL
    <input type="submit" name="Submit" value="Add to Cart"> <br>
</form>
```

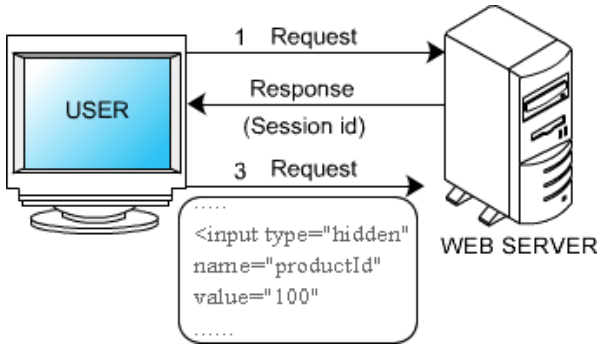
Session Tracking

Hidden Form Fields

- We **store the information in the hidden field** and **get it** from another servlet.
- This approach is better if we have to **submit form** in all the pages and we don't want to depend on the browser.

```
<input type="hidden" name="uname" value="Vimal Jaiswal">
```

- **Advantage of Hidden Form Field**
  - ✓ It will always work whether cookie is disabled or not.
- **Disadvantage of Hidden Form Field:**
  - ✓ It is maintained at server side.
  - ✓ Extra **form submission** is required on each pages.
  - ✓ Only textual information can be used.
- When the user **visits the next page**, the server side program reads all the parameters that a user passes in the previous form



Session Tracking

Hidden Form Fields

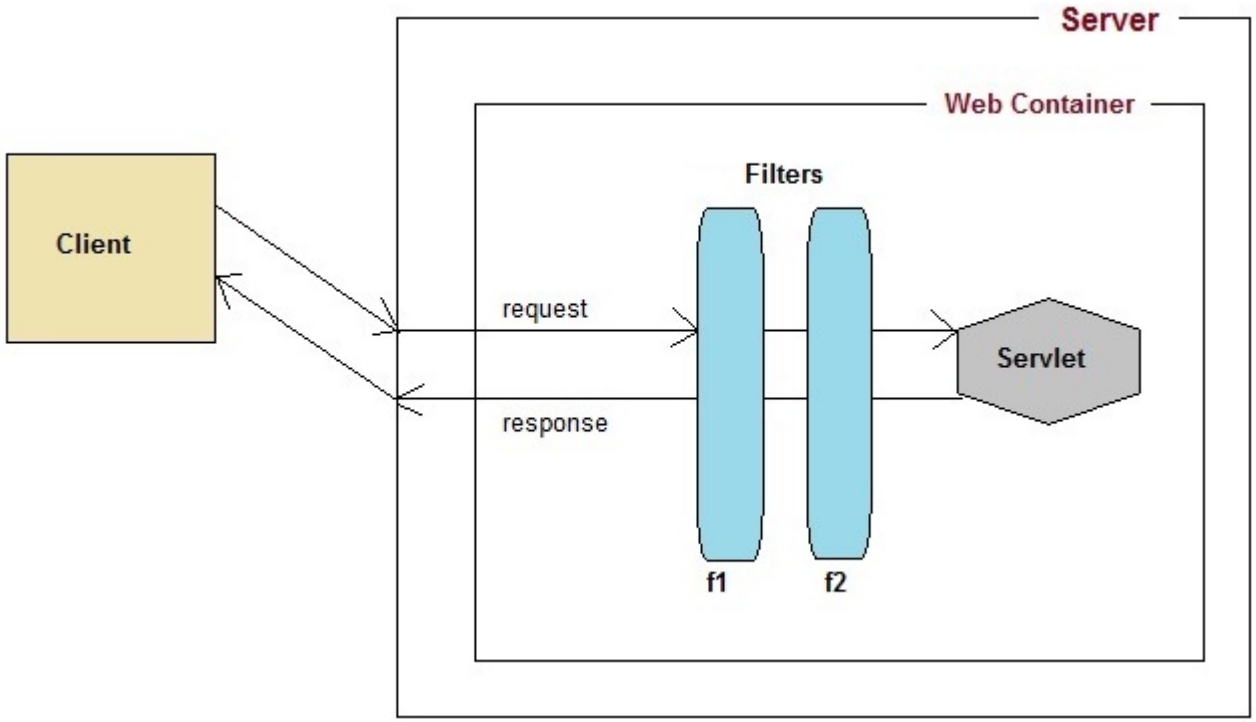
```
<b>Search results for books</b>
<form method="post" action="serverprogram.jsp">
    // Hidden input field
    <input type="hidden" name="productID" value="100">
    // Provides check box for user input
    <input type="checkbox" name="productID" value="150">
            DVD Player with Built in Amplifier <br>
    <input type="checkbox" name="productID“ value="160">
            Ultra Slim DVD Player Multi Region 5.1 Digital<br>
    // Submits user input to the server side program
    <input type="submit" name="Submit" value="Add to Cart"><br>
</form>
```

Section 3

SERVLET FILTER

Introduction to Filter API

- **Filters** are compontents that you can use and configure to perform some filtering tasks.
  - ✓ Filter is used for pre-processing of requests and post-processing of responses.



Introduction to Filter API (2/2)

- For creating a filter, we must implement **Filter** interface. Filter interface gives the following life cycle methods for a filter:
  - ✓ void **init**(FilterConfig filterConfig): invoked by the web container to indicate to a filter that it is being placed into service.
  - ✓ void **doFilter**(ServletRequest request, ServletResponse response, FilterChain chain): invoked by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
  - ✓ void **destroy**(): invoked by the web container to indicate to a filter that it is being taken out of service.

Deployment Descriptor

```
<web-app ...>
<filter>
<filter-name>MyFilter</filter-name>
<filter-class>MyFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>MyFilter</filter-name>
<url-pattern>..</url-pattern> or <servlet-name>.</servlet-name>
</filter-mapping>
</web-app>
```

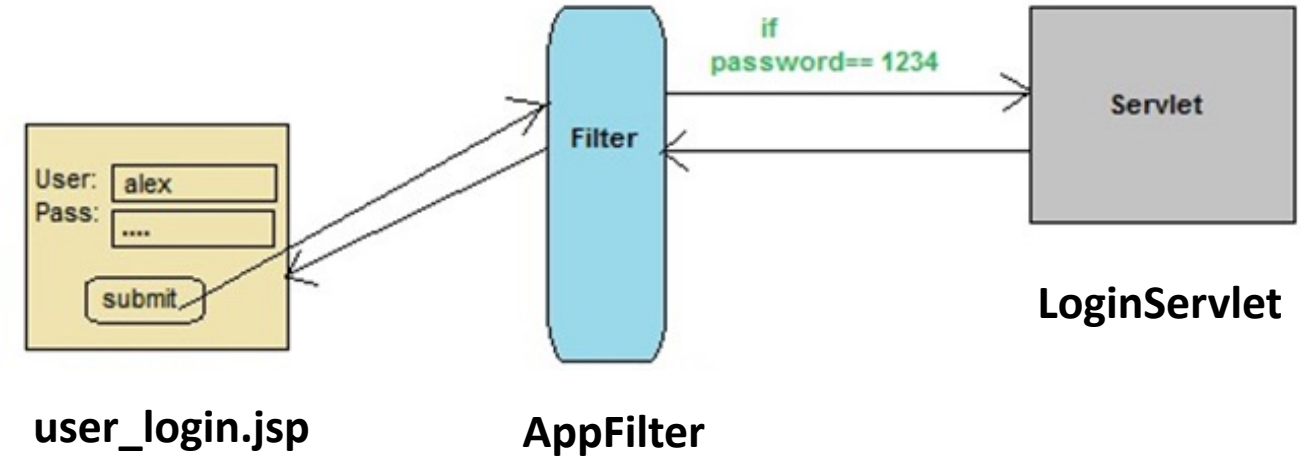
<filter-name> tag is used to give a internal name to your filter

<filter-class> declares the filter that you have created

Either the <url-pattern> or the <servlet-name> element is mandatory which web app resource will use this filter

Practical time

- In this example we are using Filter to authenticate:



Summary

- Introduction to Servlet
- Servlet API
- Servlet Request and Response
- Servlet Context
- Create servlet in Eclipse IDE
- Servlet FILTER

Thank you