ADVANCED SERVLET and SESSION TRACKING

Instructor:

Learning Goals

Understand Servlet

Can use Servlet to develop web application

Table Content

Section 1

SERVLET EXCEPTION HADLING

Servlet Exception Handling

- We can define custom error handling using a *web.xml* file descriptor in which we can define the following types of policies:
  - ✓ **Status code error handling** – it allows us to map HTTP error codes (**client** and **server**) to a static HTML error page or an error handling servlet
  - ✓ **Exception type error handling** – it allows us to map exception types to static HTML error pages or an error handling servlet
- **Status Code Error Handling with an HTML Page**
- **Example1:**

```
<web-app xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee    http://java.sun.com/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1">
    <error-page>
        <error-code>404</error-code>
        <location>/error-404.html</location> <!-- /src/main/webapp/error-404.html-->
    </error-page>
</web-app>
```

Servlet Exception Handling

- **Exception Type Error Handling with a Servlet**
- **Example 2:**

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee    http://java.sun.com/xml/ns/javaee/web-app_3_1.xsd"
 version="3.1">
  <error-page>
     <exception-type>java.lang.Exception</exception-type>
     <location>/errorHandler</location>
  </error-page>
</web-app>
```

Servlet Exception Handling

- Create class ***ErrorHandlerServlet***, we can access the error details using the error attributes provided in the request:

```
@WebServlet(urlPatterns = "/errorHandler")
public class ErrorHandlerServlet extends HttpServlet {

    @Override
    protected void doGet( HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.setContentType("text/html; charset=utf-8");
        try (PrintWriter writer = resp.getWriter()) {
            writer.write("<html><head><title>Error description</title></head><body>");
            writer.write("<h2>Error description</h2>");
            writer.write("<ul>");
            Arrays.asList(ERROR_STATUS_CODE,  ERROR_EXCEPTION_TYPE,   ERROR_MESSAGE)
              .forEach(e -> writer.write("<li>" + e + ":" + req.getAttribute(e) + " </li>")
            );
            writer.write("</ul>");
            writer.write("</html></body>");
        }
    }
}
```

Section 2

SERVLETCONFIG AND SERVLETCONTEXT

Introduction

- An object of **ServletConfig** is created by the **web container** for each servlet. This object can be used to get configuration information from **web.xml** file.
- If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.
- **Methods of ServletConfig interface:**
  - ✓ **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
  - ✓ **public Enumeration getInitParameterNames():**Returns an enumeration of all the initialization parameter names.
  - ✓ **public String getServletName():**Returns the name of the servlet.
  - ✓ **public ServletContext getServletContext():**Returns an object of ServletContext.

ServletConfig

- **web.xml** file: The **init-param** sub-element of servlet is used to specify the initialization parameter for a servlet.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns=http://java.sun.com/xml/ns/javaee
xmlns:jsp=http://java.sun.com/xml/ns/javaee/jsp
xsi:schemaLocation=http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd version="3.0">
<servlet>
<servlet-name>welcome</servlet-name>
<servlet-class>com.fsoft.controller.WelcomeServlet</servlet-class>
<init-param>
     <param-name>driver</param-name>
     <param-value>com.microsoft.sqlserver.jdbc.SQLServerDriver</param-value>
</init-param>

</servlet>
<welcome-file-list>
     <welcome-file>/views/sign-in.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

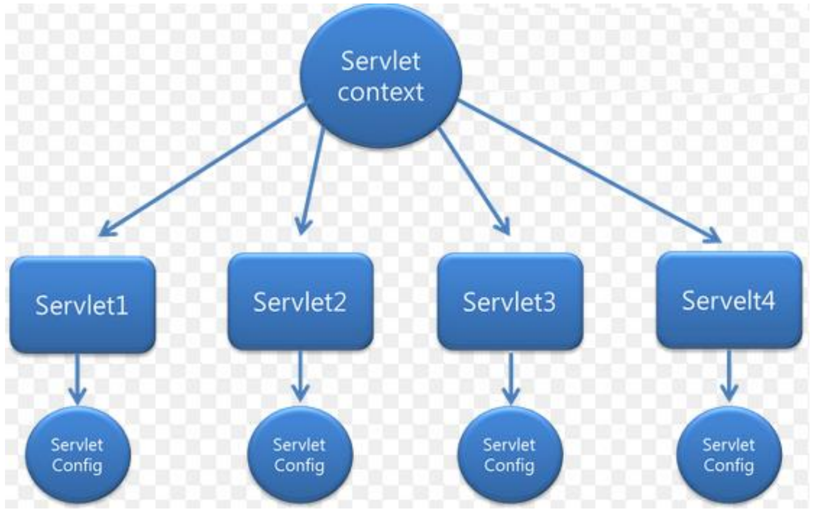
ServletConfig

- **How to get the object of ServletConfig:**
  - ✓ Use **getServletConfig() method** of Servlet interface returns the object of ServletConfig.
  - ✓ Via **init**() method of a Servlet:

```
@WebServlet("/AppServlet")
public class AppServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  @Override
    public void init(ServletConfig config) throws ServletException {
         String driver = config.getInitParameter("driver");
        Log4J.getLogger().info(driver);     }
}
```

ServletContext

- An object of ServletContext is created by the web container at time of deploying the project. This object can be used to **get configuration information from web.xml file**.
- Servlet Context has 3 main methods:
  - ✓ GetAttribute ()
  - ✓ SetAttribute ()
  - ✓ RemoveAttribute ()
- Servlet Context help provides communication between the servlet
- There is only one **ServletContext** object per web application.

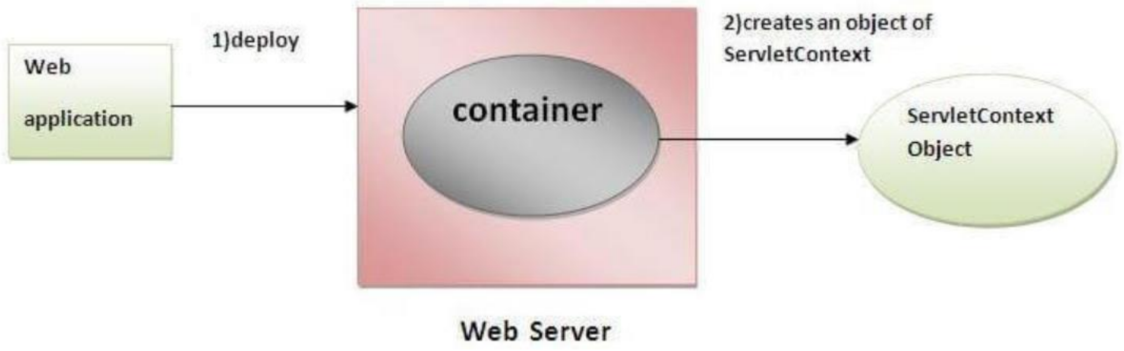


ServletContext

- **Advantage of ServletContext:**
  - ✓ **Easy to maintain** if any information is shared to all the servlet, it is better to make it available for all the servlet.
  - ✓ We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet.
  - ✓ Thus it removes maintenance problem.

ServletContext

- **Usage of ServletContext Interface:**
  - ✓ The object of ServletContext provides an interface between the container and servlet.
  - ✓ The ServletContext object can be used to get configuration information from the web.xml file.
  - ✓ The ServletContext object can be used to set, get or remove attribute from the web.xml file.
  - ✓ The ServletContext object can be used to provide inter-application communication.



https://www.javatpoint.com

ServletContext

- **Methods of ServletContext interface:**
  - ✓ **public String getInitParameter(String name):**

    Returns the parameter value for the specified parameter name.
  - ✓ **public Enumeration getInitParameterNames():**

    Returns the names of the context's initialization parameters.
  - ✓ **public void setAttribute(String name,Object object):**

    sets the given object in the application scope.
  - ✓ **public Object getAttribute(String name):**

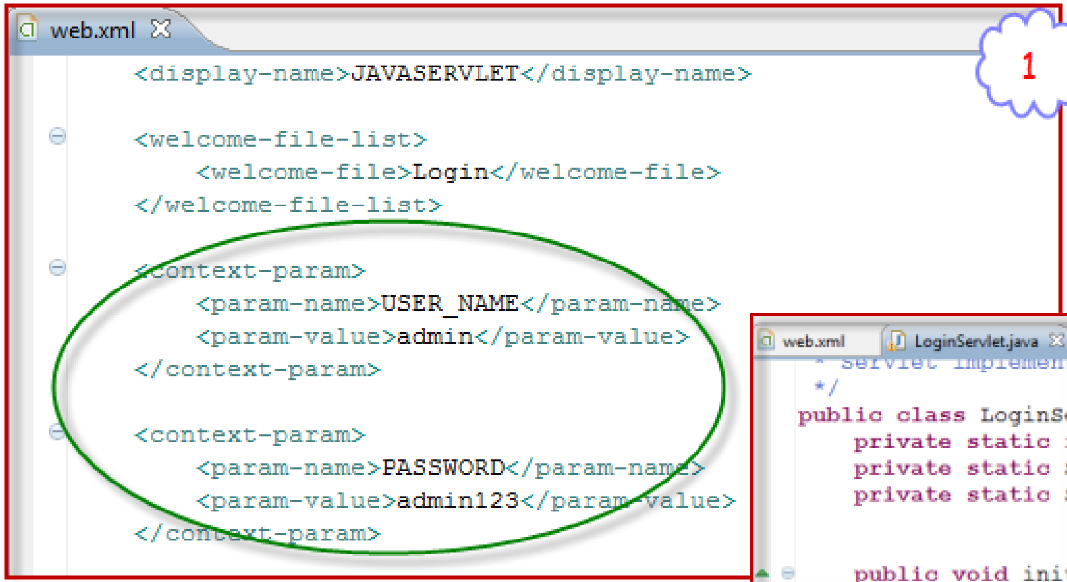
    Returns the attribute for the specified name.
  - ✓ **public Enumeration getInitParameterNames():**

    Returns the names of the context's initialization parameters as an Enumeration of String objects.
  - ✓ **public void removeAttribute(String name):**

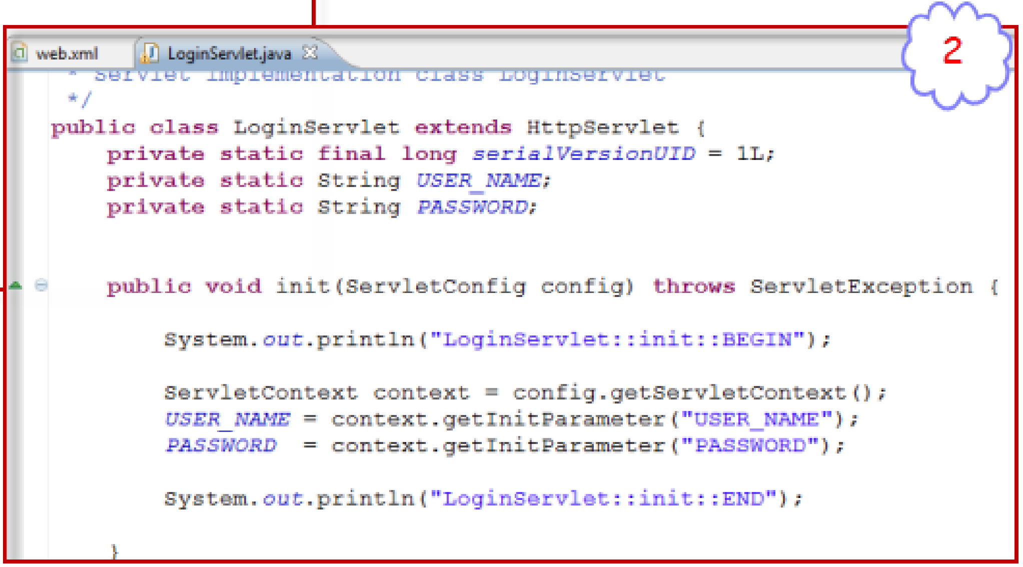
    Removes the attribute with the given name from the servlet context.

https://www.javatpoint.com

ServletContext Example



```xml
    <display-name>JAVASERVLET</display-name>

    <welcome-file-list>
        <welcome-file>Login</welcome-file>
    </welcome-file-list>

    <context-param>
        <param-name>USER_NAME</param-name>
        <param-value>admin</param-value>
    </context-param>

    <context-param>
        <param-name>PASSWORD</param-name>
        <param-value>admin123</param-value>
    </context-param>
```



```java
 * Servlet implementation class LoginServlet
 */
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static String USER_NAME;
    private static String PASSWORD;


    public void init(ServletConfig config) throws ServletException {

        System.out.println("LoginServlet::init::BEGIN");

        ServletContext context = config.getServletContext();
        USER_NAME = context.getInitParameter("USER_NAME");
        PASSWORD  = context.getInitParameter("PASSWORD");

        System.out.println("LoginServlet::init::END");

    }
```

Servlet Context
Sample

- We can change the init() method of the as below:

```
// set up database connection and prepare SQL statements
public void init( ServletConfig config ) throws ServletException
{
   String dbDriver, dbURL;
   ServletContext context = config.getServletContext();

   dbDriver = context.getInitParameter("DB_Driver");
   dbURL = context.getInitParameter("DB_URL");

   // attempt database connection and create PreparedStatements
   // ...
}
```

- Then we need to amend the web.xml file to specify the initial context parameters:

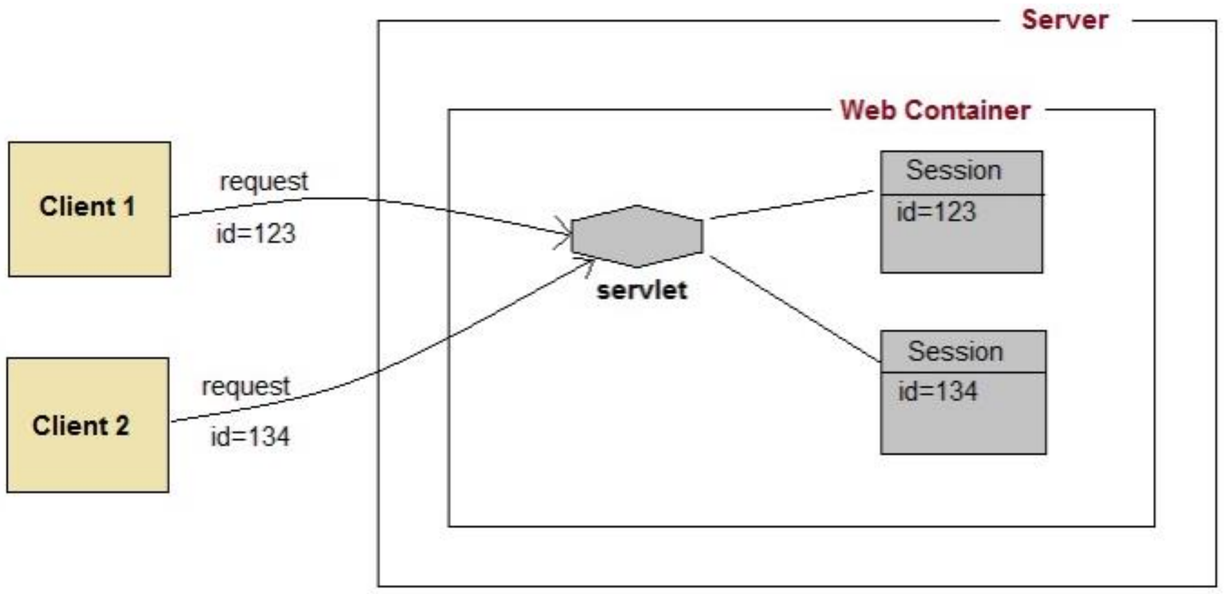
```
<context-param>
    <param-name>DB_URL</param-name>
    <param-value>jdbc:mysql://localhost:3306/test</param-value>
</context-param>

<context-param>
    <param-name>DB_Driver</param-name>
    <param-value>com.mysql.jdbc.Driver</param-value>
</context-param>
```

Section 3

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 fiel**
  - ✓ **URL Rewriting**
  - ✓ **HttpSession**



Session Tracking

Cookie in Servlet

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- **How Cookie works:**
  - ✓ By default, each request is considered as a new request.
  - ✓ In cookies technique, we add cookie with response from the servlet.
  - ✓ So cookie is stored in the **cache of the browser**. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.
- **Types of Cookie:**
  - ✓ Non-persistent cookie: It is **valid for single session** only. It is removed each time when user closes the browser
  - ✓ Persistent cookie: It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Session Tracking

Cookie in Servlet

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

Cookie in Servlet

- Cookie class: **javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides a lot of useful methods for cookies.
- **Constructors**:

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

- **Methods**:

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Session Tracking

Cookie in Servlet

- Other methods required for using Cookies:
  - ✓ **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object.
  - ✓ **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser.
- How to create Cookie?

```
Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object
response.addCookie(ck);//adding cookie in the response
```

- How to delete Cookie?

```
Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

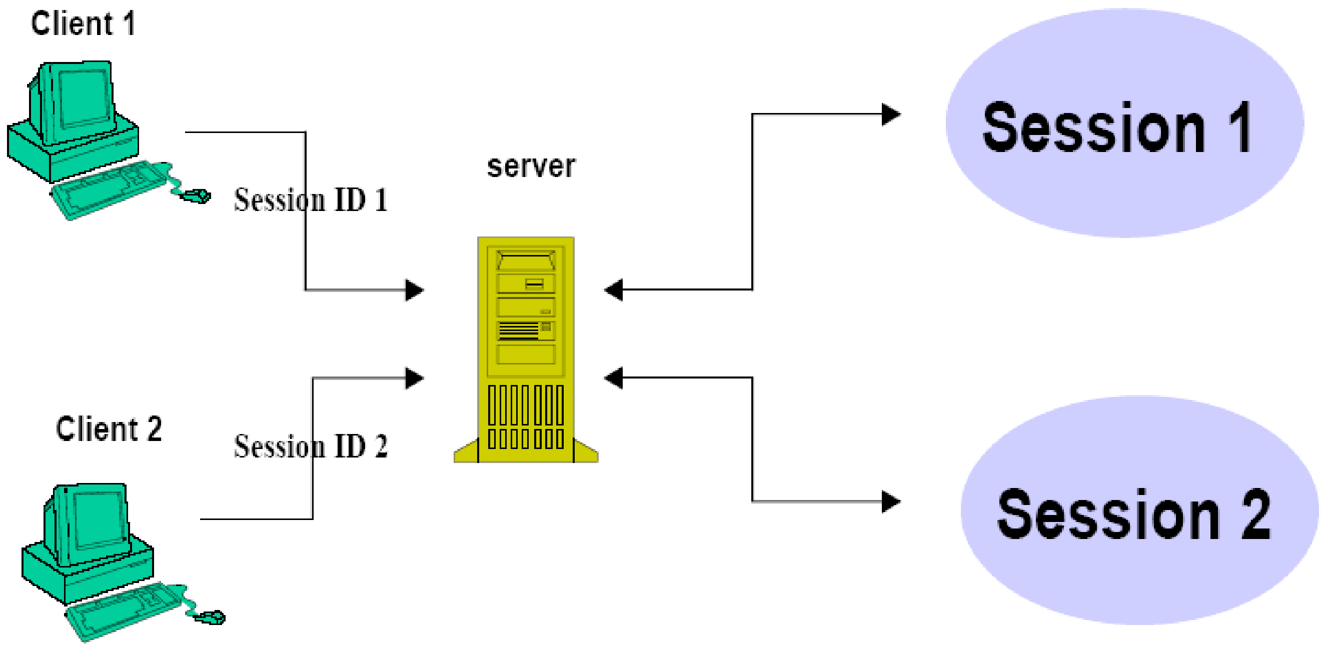
- How to get Cookies?

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
   out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());
   //printing name and value of cookie
}
```

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



Session Tracking

HttpSession

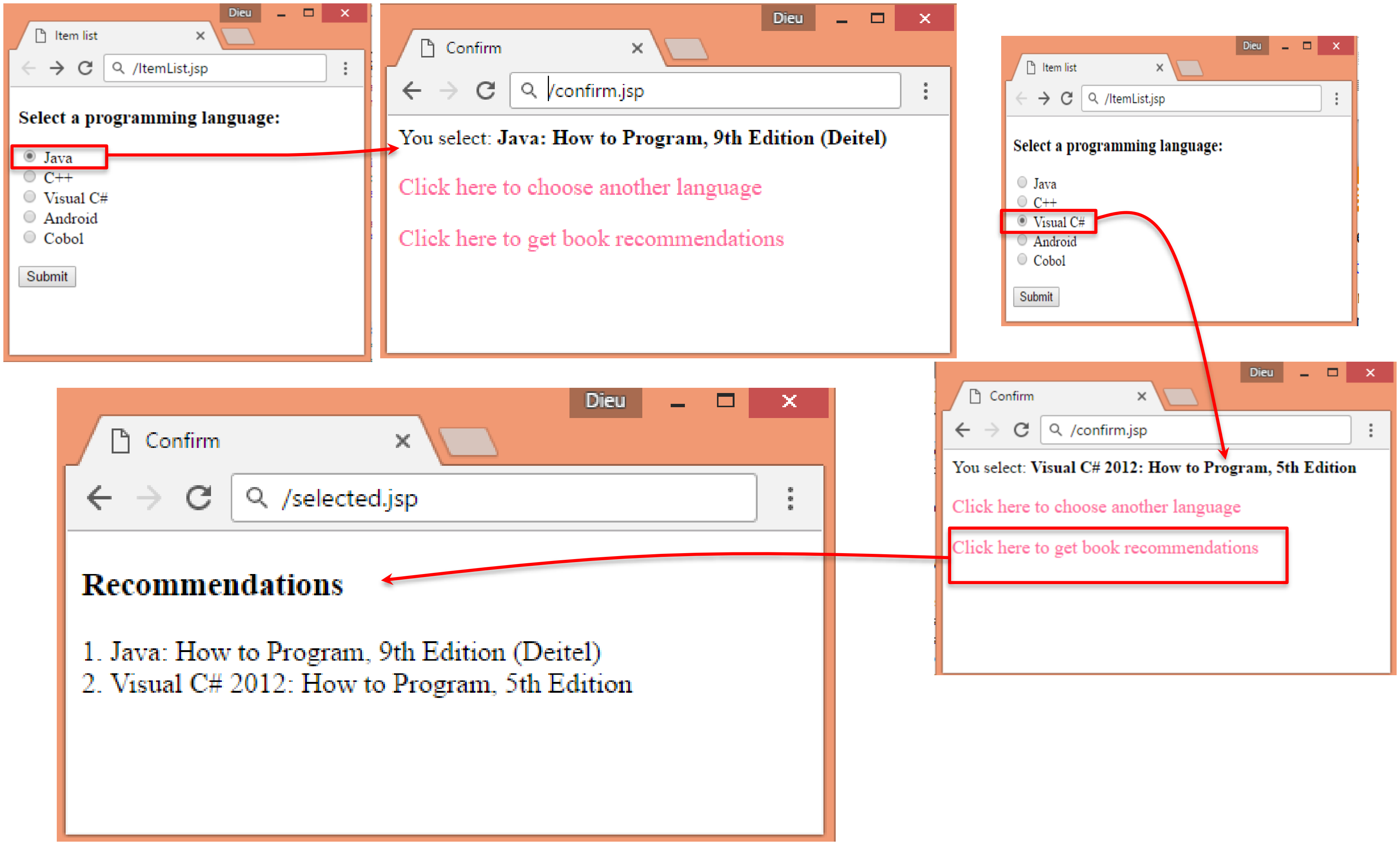
- The container creates a session id for each user. Container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:
  - ✓ bind objects
  - ✓ view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.
- **How to get the HttpSession object?**
  - ✓ **public HttpSession getSession():** Returns the current session associated with this request, or if the request does not have a session, creates one.
  - ✓ **public HttpSession getSession(boolean create):** Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

Session Tracking

HttpSession

- **Methods of HttpSession**
  - ✓ **public String getId():**Returns a string containing the unique identifier value.
  - ✓ **public void setAttribute(String name, Object value)**: Binds the object with a name and stores the name/value pair as an attribute of the HttpSession object. If an attribute already exists, then this method replaces the existing attributes.
  - ✓ **public Object getAttribute(String name)**: Returns the String object specified in the parameter, from the session object. If no object is found for the specified attribute, then the getAttribute() method returns null.
  - ✓ **public Enumeration getAttributeNames()**: Returns an Enumeration that contains the name of all the objects that are bound as attributes to the session object.
  - ✓ **public void removeAttribute(String name)**: Removes the given attribute from session.
  - ✓ **setMaxInactiveInterval(int interval)**: Sets the session inactivity time in seconds. This is the time in seconds that specifies how long a sessions remains active since last request received from client.
  - ✓ **public void invalidate():**Invalidates this session then unbinds any objects bound to it.

Session Tracking
HttpSession



Select a programming language:

● Java
○ C++
○ Visual C#
○ Android
○ Cobol

You select: **Java: How to Program, 9th Edition (Deitel)**

Click here to choose another language

Click here to get book recommendations

Select a programming language:

○ Java
○ C++
● Visual C#
○ Android
○ Cobol

You select: **Visual C# 2012: How to Program, 5th Edition**

Click here to choose another language

Click here to get book recommendations

Recommendations

1. Java: How to Program, 9th Edition (Deitel)
2. Visual C# 2012: How to Program, 5th Edition

Session Tracking

HttpSession

❖ **Example:**

✓ **login.html**

```
<form action="FirstServlet">
          User Name:<input type="text" name="userName"/><br/>
          Password:<input type="password" name="userPassword"/><br/>
          <input type="submit" value="Submit"/>
</form>
```

✓ **FirstServlet.java**

```
@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){
        try{

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String userName =request.getParameter("userName");
        out.print("Welcome "+ userName);

        HttpSession session=request.getSession();
        session.setAttribute("uname", userName);

        out.print("<br><a href= 'SecondServlet'>Visit</a>");

        out.close();

        }catch(Exception e){System.out.println(e);}
     }

}
```

Session Tracking

HttpSession

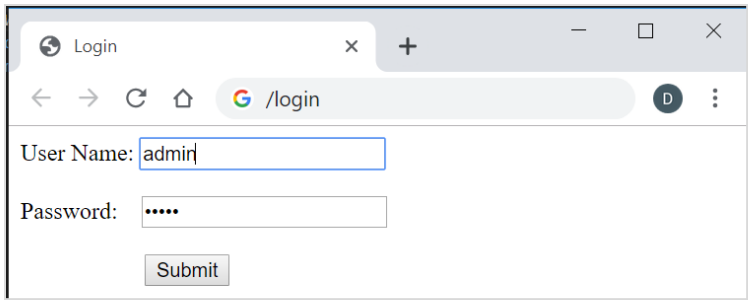
❖ **Example:**

- ✓ **SecondServlet.java**

```java
@WebServlet("/SecondServlet")
public class SecondServlet extends HttpServlet {

public void doGet(HttpServletRequest request,
                                   HttpServletResponse response)
        try{

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        HttpSession session=request.getSession(false);
        String userName =(String)session.getAttribute("uname");
        out.print("Hello " + userName);

        out.close();

        }catch(Exception e){System.out.println(e);}
    }


}
```

Session Tracking
HttpSession

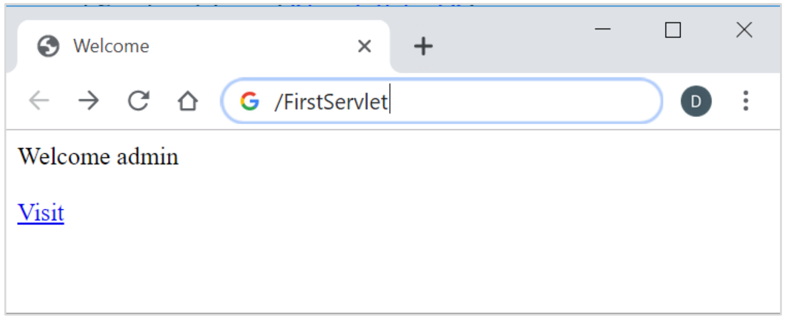
- **login.html**



User Name: admin

Password: •••••

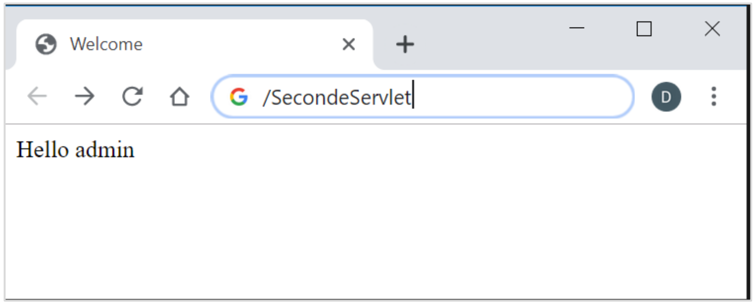
- **After clicking Submit:**



Welcome admin

Visit

- **After clicking Visit:**



Hello admin

Session Tracking

URL Rewriting

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

  url?name1=value1&name2=value2&

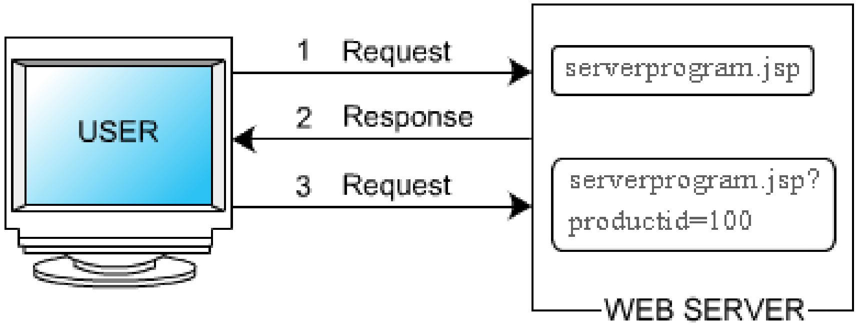
- When the **user clicks the hyperlink**, the **parameter** name/value pairs **will be passed to the server**. We can use **getParameter()** method to obtain a parameter value.

- **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

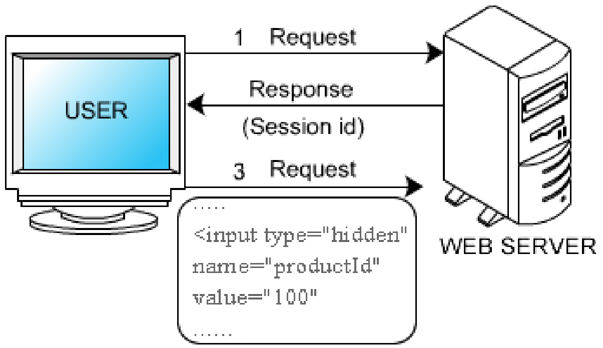
Hidden Form Fields

- 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>
```

Summary

- ◊ **Servlet Exception Hadling**
- ◊ **ServletConfig and servletcontext**
- ◊ **Servlet Session Tracking**
- ◊ **Q&A**

Thank you