Ch-2

Session Tracking Approaches

Http is a stateless protocol. Each request is independent of the previous one. However, in some application it is necessary to save the information. So that it can be collected from several interactions between a browser and a server.

A Session is temporary small unique connection between a server and the client enabling it to identify that user across multiple page requests or visit to that site.

It provides following mechanism to track the user state across requests.

(1) URL Rewriting : URL rewriting is baesd on the idea of inserting a unique ID in each URL of the response from the server.

That is, while generating the response to the first request, the server inserts this ID in each URL.

When the client submits a request to one such URL, the browser sends this ID back to the server.

Example of create servlet file with name exurlrewrit.java

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public void doGet(HttpServletRequest req,HttpServletResponse res) throws ServletException{

        res.setContentType(“text/html”);

        printWriter out=res.getWriter();

        String contextPath=res.getContextPath();

        stringencodingUrl=res.encodeURL (contextpath +”Hello World.html”);

        out.println(“<html>”);

        out.println(“<head>”);

        out.println(“<title> URL Rewriter</title>”);

        out.println(“</head>”);
```

```
        out.println("<body>");
        out.println("<h2> This page will use URL rewriting if neccessary</h2>");
        out.println("Go to the HelloWorld page<a href=\" "+encodeUrl
+"\">here<\a>.");
        out.println("</body>");
        out.println("</html>");
        }
public void doPost(HttpServletRequest rq, HttpServletResponse res) throws
ServletException
        {
                doGet(req,res);
        }
}
```

create following html file with name HelloWorld.html

```
<html>
<head>
        <title>URL Rewrite</title>
</head>
        <body>
                <H1> Hello World </H1>
        </body>
</html>
```

(2) Hidden Form Field : Other way to support session tracking is to use hidden form fields.

As the name implies, these are fields added to an HTML form that are not displayed in the client's browser.

Advantage of hidden form field :

It only works for a sequence of dynamically generated forms.

There are no browser shutdowns.

Example of create servlet file with name Hiddenservlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public void doGet(HttpServletRequest req,HttpServletREsponse res) throws ServletException
{
        res.setContentType(“text/html;charset=UTF-8”);
        printWriter out=res.getWriter();
        string str=req.getQueryString();
string[] newstr=str.split(“ & ”);
        out.println(“<html>”);
        out.println(“<head>”);
        out.println(“<title>HiddenFormField</title>”);
        out.println(“</head>”);
        out.println(“<body>”);
        out.println(“<b>”+newstr[0]+”<b>”+newstr[1]);
         out.println(“</body>”);
         out.println(“</html>”);
        }
 public void doPost(HttpServletRequest rq, HttpServletResponse res) throws ServletException
{
                doGet(req,res);
        } }
```

create following html file with name HelloWorld.html

```
<html>
<head>
    <title>URL Rewrite</title>
</head>
    <body>
        <H1> Hello World </H1>
    </body>
</html>
```

(3) Cookies : cookies provide a better alternatives to explicit URL rewriting, because cookies are not sent as query string but are exchanged within the bodies of HTTP requests and response.

All modern browsers can recognize and receive cookies from web server, and then send them back along with requests.

A cookies is a sent via the HTTP request and HTTP response header. A cookie has following parameters :

Parameter	Description
Name :	Name of cookie
Value :	A value of the cookie
Comment :	Comment Explain purpose of cookie
Max-Age :	Set lifetime of cookie
Domain :	domain to which cookie should be sent.
Path :	path to which cookie should be sent
Secure :	specifies if the cookie should be sent securely via HTTP
Version :	version of the cookie protocol.

- Some of the Methods of cookie class as given below :
- 1. Set Methods :

(1) setComment()

(2) setDomain()

(3) setMaxAge()

(4) setpath()

(5) setSecure()

(6) setValue()

(7) setVersion()

- 2. Get Methods :

(1) getComment()

(2) getDomain()

(3) getMaxAge()

(5) getPath()

(6) getsecure()

(7) getvalue()

Create servlet file name with SetCookie.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SetCookies extends HttpServlet
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException
{
        for(int i=0; i<3;i++)
        {
                cookie cookie=new cookie("s-cookie"+i,"c-  val-s"+i);
```

```
res.addCookie(cookie);
            cookie cookie=new cookie("persistcookie"+i,      "c-val-p"+i);
            cookie.setMaxAge(3600);
             res.addCookie(cookie);
      }
            res.setContentType("text/html");
            PrintWriter out=res.getWriter();
            out.println("<html><body>\n"+ <h1   align=\"center\">"+ setting
cookies" +        "<\h1>\n"+"there are six cookies.\n"+"to   see them, visit
\n"+"<ahref=\"Show  Cookies\">\n"+"</body></html>");   }  }
Create servlet file name with ShowCookie.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class ShowCookies extends HttpServlet
{
      PrintWriter out;
      public void doGet(HttpServletRequest req, HttpServletResponse
res)throws IOException
{
      try
      {
res.setContentType("text/html");
      out=res.getWriter();
      out.println("<html><body>\n"+<h1 align=\"center\">+" Active
cookies"+"</h1>\n"+ "<table border=1 align= \"center"\>\n"
+"<tr>\n"+"<th>Cookie Name\n" +"<th>Cookie value");
Cookie[] c=req.getCookie();
```

```
Cookie c;
for(i=0;i<c.length;i++)
{
    c=cookies[i];
out.println("<tr>\n"+"<td>"+cookie.getName()+"\n"+"<td>"+cookie.getValue(
));
}
out.println("</table></body></html>");
}
    catch(Exception e)
    {
        out.println(e.getMessage());
    }
```

4. Session tracking API

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below −

```
HttpSession session = request.getSession();
```

You need to call *request.getSession()* before you send any document content to the client.

public void setAttribute(String name, Object value)

This method binds an object to this session, using the name specified.

public String getId()

This method returns a string containing the unique identifier assigned to this session.

public Object getAttribute(String name)

This method returns the object bound with the specified name in this session, or null if no object is bound under the name.

public void removeAttribute(String name)

This method removes the object bound with the specified name from this session.

Servlet Collaboration

What is Servlet Collaboration?

The exchange of information among servlets of a particular Java web application is known as Servlet Collaboration. This enables passing/sharing information from one servlet to the other through method invocations.

What are the principle ways provided by Java to achieve Servlet Collaboration?

The servlet api provides two interfaces namely:

1) **javax.servlet.RequestDispatcher**
2) **javax.servlet.http.HttpServletResponse**

These two interfaces include the methods responsible for achieving the objective of sharing information between servlets.

Using RequestDispatcher Interface

The RequestDispatcher interface provides the option of dispatching the client's request to another web resource, which could be an HTML page, another servlet, JSP etc. It provides the following two methods:

public void forward(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The forward() method is used to transfer the client request to another resource (HTML file, servlet, jsp etc). When this method is called, the control is transferred to the next resource called.

public void include(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The include() method is used to include the contents of the calling resource into the called one. When this method is called, the control still remains with the calling resource. It simply includes the processed output of the calling resource into the called one.

Example of using RequestDispatcher for Servlet Collaboration

The following example explains how to use RequestDispatcher interface to achieve Servlet Collaboration:

FirstServlet.java

```
import java.io.IOException;

import javax.servlet.*;

import javax.servlet.http.*;

@WebServlet("/firstServlet")

public class FirstServlet extends HttpServlet {

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

      RequestDispatcher includeDispatcher =
      request.getRequestDispatcher("/secondServlet");

      includeDispatcher.include(request, response);

       RequestDispatcher forwardDispatcher =
       request.getRequestDispatcher("/thirdServlet");

      forwardDispatcher.forward(request, response);

   }

}
```

SecondServlet.java

```
import java.io.IOException;

import javax.servlet.*;

import javax.servlet.http.*;

@WebServlet("/secondServlet")

public class SecondServlet extends HttpServlet {

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

      response.getWriter().println("This is the SecondServlet");

   }

}
```

ThirdServlet.java

```
import java.io.IOException;

import javax.servlet.*;

import javax.servlet.http.*;

@WebServlet("/thirdServlet")

public class ThirdServlet extends HttpServlet {

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

      response.getWriter().println("This is the ThirdServlet");

   }

}
```

Using HttpServletResponse Interface

The HttpServletResponse interface is entrusted with managing Http responses. To achieve servlet collaboration, it uses the following method:

public void sendRedirect(String URL)throws IOException;

This method is used redirect response to another resource, which may be a servlet, jsp or an html file. The argument accepted by it, is a URL which can be both, absolute and relative. It works on the client side and uses the browser's URL bar to make a request.

Example of using sendRedirect() for redirection

The following example of a web application created using servlet takes the text written in the text field in the webpage, and directs it to the servlet. The servlet then redirects it to google, which then produces search results based on the text written.

index.html

```
<html>

<head>

<body>

<form action="search" method="GET">

<input type="text" name="name">
```

```
<input type="submit" value="search">
</form>
</body>
</html>
```

```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*
public class MySearcher extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse
   response) throws ServletException, IOException
   {
      String name = request.getParameter("name");
      response.sendRedirect("https://www.google.co.in/#q=" + name);
      // response redirected to google.com
   }
}
```

forward()	sendRedirect()
It works on the server side	It works on the client side
It sends the same request and response objects to another resource.	It always send a new request
It works only within the server.	It can be used within and outside the server

Ch-1

Que - Difference Between execute(), executeQuery(), qxecuteUpdate()

executeQuery()	executeUpdate()	execute()
This method is used to execute the SQL statements which retrieve some data from the database.	This method is used to execute the SQL statements which update or modify the database.	This method can be used for any kind of SQL statements.
This method returns a ResultSet object which contains the results returned by the query.	This method returns an int value which represents the number of rows affected by the query. This value will be the 0 for the statements which return nothing.	This method returns a boolean value. TRUE indicates that query returned a ResultSet object and FALSE indicates that query returned an int value or returned nothing.
This method is used to execute only select queries.	This method is used to execute only non-select queries.	This method can be used for both select and non-select queries.
Ex : SELECT	Ex : DML → INSERT, UPDATE and DELETE DDL → CREATE, ALTER	This method can be used for any type of SQL statements.

Que – DatabaseMetaDeta interface

The **DatabaseMetaData** interface provides methods to get information about the database you have connected with like, database name, database driver version, maximum column length etc...

Following are some methods of **DatabaseMetaData** class.

Method	Description
getDriverName()	Retrieves the name of the current JDBC driver
getDriverVersion()	Retrieves the version of the current JDBC driver
getUserName()	Retrieves the user name.
getDatabaseProductName()	Retrieves the name of the current database.
getDatabaseProductVersion()	Retrieves the version of the current database.
getNumericFunctions()	Retrieves the list of the numeric functions available with this database.
getStringFunctions()	Retrieves the list of the numeric functions available with this database.
getSystemFunctions()	Retrieves the list of the system functions available with this database.
getTimeDateFunctions()	Retrieves the list of the time and date functions available with this database.
getURL()	Retrieves the URL for the current database.

Following example demonstrates the usage of DatabaseMetaData class.

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
public class DatabaseMetadataExample {
   public static void main(String args[])throws Exception {
      //Getting the connection
      String mysqlUrl = "jdbc:mysql://localhost/sampleDB";
      Connection con = DriverManager.getConnection(mysqlUrl, "root", "password");
      System.out.println("Connection established......");

      //Creating the DatabaseMetaData object
      DatabaseMetaData dbMetadata = con.getMetaData();
      //invoke the supportsBatchUpdates() method.
      boolean bool = dbMetadata.supportsBatchUpdates();

      if(bool) {
         System.out.println("Underlying database supports batch updates");
      } else {
         System.out.println("Underlying database doesnt supports batch updates");
      }

      //Retrieving the driver name
      System.out.println(dbMetadata.getDriverName());
      //Retrieving the driver version
      System.out.println(dbMetadata.getDriverVersion());
      //Retrieving the user name
      System.out.println(dbMetadata.getUserName());
      //Retrieving the URL
      System.out.println(dbMetadata.getURL());
      //Retrieving the list of numeric functions
      System.out.println("Numeric functions: "+dbMetadata.getNumericFunctions());
      System.out.println("");
      //Retrieving the list of String functions
```

```
    System.out.println("String functions: "+dbMetadata.getStringFunctions());
    System.out.println("");
    //Retrieving the list of system functions
    System.out.println("System functions: "+dbMetadata.getSystemFunctions());
    System.out.println("");
    //Retrieving the list of time and date functions
    System.out.println("Time and Date funtions: "+dbMetadata.getTimeDateFunctions());
  }
}
```

Que – Methods of ResultSet interface

Commonly used methods of ResultSet interface

1) public boolean next():	is used to move the cursor to the one row next from the current position.
2) public boolean previous():	is used to move the cursor to the one row previous from the current position.
3) public boolean first():	is used to move the cursor to the first row in result set object.
4) public boolean last():	is used to move the cursor to the last row in result set object.
5) public boolean absolute(int row):	is used to move the cursor to the specified row number in the ResultSet object.
6) public boolean relative(int row):	is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.
7) public int getInt(int columnIndex):	is used to return the data of specified column index of the current row as int.
8) public int getInt(String columnName):	is used to return the data of specified column name of the current row as int.
9) public String getString(int columnIndex):	is used to return the data of specified column index of the current row as String.
10) public String getString(String columnName):	is used to return the data of specified column name of the current row as String.

Let's see the simple example of ResultSet interface to retrieve the data of 3rd row.

```
1. import java.sql.*;
2. class FetchRecord{
3. public static void main(String args[])throws Exception{
4.
```

```
5. Class.forName("oracle.jdbc.driver.OracleDriver");
6. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
   m","oracle");
7. Statement stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
   _UPDATABLE);
8. ResultSet rs=stmt.executeQuery("select * from emp765");
9.
10. //getting the record of 3rd row
11. rs.absolute(3);
12. System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));
13.
14. con.close();
15. }}
```