UNIT 3 SESSION MANAGEMENT AND DATABASE CONNECTIVITY IN SERVLET

Structure

3.0 INTRODUCTION

In the previous unit, you have already gone through the basics of Servlets in detail which are a server side programming language. As you know that the servlets are used for dynamic web application. Several users interact with such web applications simultaneously. Do you know how to manage this dynamic application among the users? A strategy called session management is applied in these applications. In this unit, you will learn more about session management. In Java Servlet, session is managed through different techniques such as HttpSession object, Cookies, URL rewriting and Hidden Form field. For example, when you check your result on the IGNOU website and put your roll number in the input field, it shows that some roll numbers might appear for your selection. It is all possible by the cookies. You will find explanations and examples of these techniques used in session management.

In addition, this unit introduced you to Servlet Collaboration which is all about sharing information among the servlets. The Servlet-API provides the RequestDispatcher, HttpServletResponse and ServletContext interface to achieve Servlet Collaboration. RequestDispatcher interface is useful for forwarding the request to another web resource and including the resource in the current servlet. HttpServletResponse interface makes available a method sendRedirect to communicate with others. A servlet can also share information among multiple servlets by using setAttribute and getAttibute method of ServletContext. This unit explains to you how the servlets communicate with each other using the methods defined in these three interfaces. Apart from these, this unit also enlightens you about database access using Java Database Connectivity (JDBC) technology.

In the previous unit, you have already gone through the procedure of compiling as well as after compiling servlet made entries in deployment descriptor (web.xml) file and invoked them from a web browser. The previous Unit also defined the creating and running procedure of servlet in NetBeans. This Unit is also devoted to the servlets. So, there is no need to define the same procedures again.

3.1 OBJECTIVES

After going through this unit, you will be able to:

- Describe how to manage session between servlets,
- Connect servlet with database,
- Use forward() and include() method,
- Share information between servlets,
- Use setAttribute() and getAttribute() in servlets, and
- Insert and retrieve data to/from database.

3.2 SESSION MANAGEMENT

In the previous unit, you studied the Java Servlets used to create dynamic content-based web pages or web applications. Dynamic web pages are different from static web pages in which web server creates a web page when a web client or user requests it. For example, when you check your online results on IGNOU website, different pages are generated for different students by the IGNOU web server depending on your enrolment number.

We all know that HTTP is a stateless protocol which is explained in the previous Unit. It means that the HTTP protocol does not remember information when client or user communicates with the server. Whenever you send a request to the server through HTTP, the HTTP treats each request as a new request. But sometimes in web applications, clients are doing some important work such as online shopping, online banking etc. In that situation, it is necessary to know about the client and remember the client's request accordingly. For example, an online banking application system enables many clients to do their different activities like checking account balance(s), obtaining statements and making financial transactions simultaneously. For maintaining each client’s session, you can use session management.

Before going into session management details, you should know about the two important terms, i.e. session and state, which are necessary for implementing business transactions across multiple clients. These are as under:

Session: The server should recognize a series of requests from the same user which form a single working ‘session’. For example, in net banking application, each client can be differentiated from another client by associating a unique identifier in request and response within a specific working session.

State: The server should be able to remember information related to the previous request and other business transaction that are made for that request. For example, in net banking application, state comprises client information such as account number and amount transaction made within the particular session.

Do remember one thing-- session management does not change the nature of HTTP protocol i.e., stateless feature provides a way to remember the client information. Session management is also known as session tracking, which permits Servlet/JSP to maintain information about a series of request from the same client. It is a mechanism to store session information for each client. For example, When a user visits any web application, unique identification information about the user is stored in an object available during the particular session until the user quits the web application.

There are four ways to manage sessions: HttpSession object, Cookies, URL rewriting, and hidden fields.

3.2.1 HttpSession Object

Any dynamic website or web application uses the concept of sessions and Session object to store data for a particular user. For example, when you visit any web application for the first time, you have entered login name and password. This information might be stored in session variable and maintained by the servlet container during your visit to web application so that it can be accessed when needed. When your session is started, the requesting browser is allotted a unique id for each of the clients for identifies the client. This Session object is denoted by javax.http.HttpSession interface.

You have learned about the HttpSession interface under the Servlet API in the previous Unit. Servlet API provides session management through HttpSession interface. This HttpSession interface provides a mechanism to identify a user to examine who is visiting a web application and to store the user information. Servlet Container creates a session id for each user. You can maintain state between the transactions by using methods of HttpSession interface.

Example -1

The following example will explain to you about session management using the methods defined in the HttpSession interface.

Following servletSession.java is servlet program that uses session tracking to keep track of how many times a particular user has accessed a page and to display some details of the current session such as Session identifier, Creation Time and Last Access Time. The source code of this program is listed below:

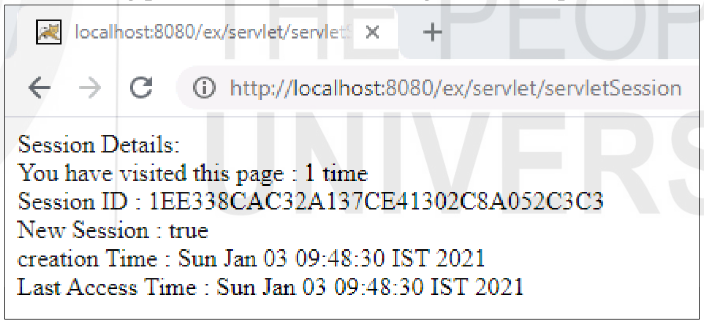
servletSession.java

```
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;
public class servletSession extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws
  ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    HttpSession session = req.getSession(true);
    Integer ct = (Integer)  session.getAttribute("ct");
    if (ct == null)
    { ct = new Integer(1); }
    else { ct = new Integer(ct.intValue()+1); }
    session.setAttribute("count", ct);
    out.println("Session Details: ");
    out.println("<br/>");
    out.println("You have visited this page : " + ct + ((ct.intValue() ==1)? " time" : "  times")
    );
    out.println("<br/>");
    out.println("Session ID : " + session.getId());
    out.println("<br/>");
    out.println("New Session : " + session.isNew() );
    out.println("<br/>");
    out.println("creation Time : " + new Date(session.getCreationTime()));
    out.println("<br/>");
    out.println("Last Access Time : " + new Date(session.getLastAccessedTime()));
  }
}
```

You already have learnt about the running procedure of servlet and methods included in the above servlet program in Unit 2 Block 1 of this course. Compile the above servlet, put the class file in the classes folder, and create appropriate entry in the web.xml file under the WEB-INF folder in your 'webapps' folder. Now, you can start your web server and run the program from your browser.

When you access this program for the first time, the visiting counter will be one and the new Session will 'true'. When visiting counter increases in numbers the new session will be 'false'. This program will also display some details of the current session such as Session identifier, Creation Time and Last Access Time.

Output of the above servlet program is displayed in the following figure-1.
Creating and running procedures of Sevlet are given in the previous Unit of this block.


localhost:8080/ex/servlet/servlet
http://localhost:8080/ex/servlet/servletSession

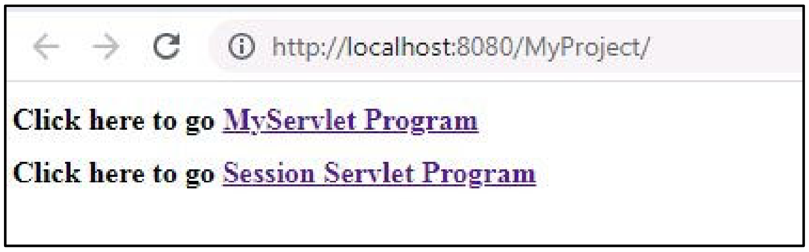
```
Session Details:
You have visited this page : 1 time
Session ID : 1EE338CAC32A137CE41302C8A052C3C3
New Session : true
creation Time : Sun Jan 03 09:48:30 IST 2021
Last Access Time : Sun Jan 03 09:48:30 IST 2021
```

Figure 1: Output of Servlet by using HttpSession Object

In the similar way, you can create above servlet program and add the code in index.html file like the following:

```
<h1>Click here to go <a href="servletSession" >Session Servlet Program</a></h1>
```

When you will run your Project in NetBeans it is displayed as output like the figure-2:



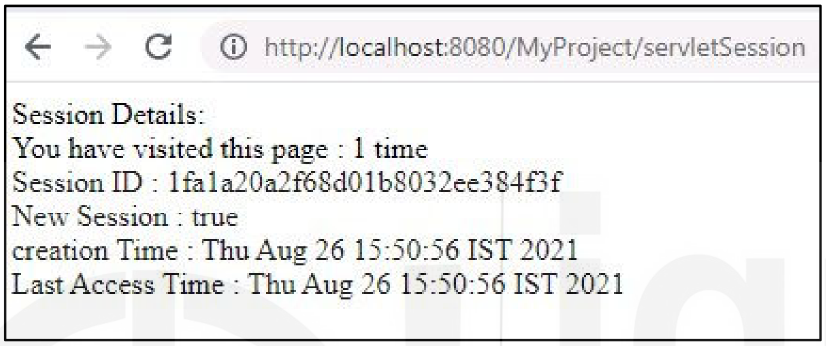
http://localhost:8080/MyProject/

Click here to go MyServlet Program

Click here to go Session Servlet Program

Figure 2: Output Screen of Welcome Index Page

When you will click on 'Session Servlet Program' link, then servlet will run and give output as Figure-3:



http://localhost:8080/MyProject/servletSession

```
Session Details:
You have visited this page : 1 time
Session ID : 1fa1a20a2f68d01b8032ee384f3f
New Session : true
creation Time : Thu Aug 26 15:50:56 IST 2021
Last Access Time : Thu Aug 26 15:50:56 IST 2021
```

Figure 3: Output Screen of Session Servlet program

3.2.2 Cookies

In the last section, you have studied session management through HttpSession object in Servlet. Here you will learn about Cookies. This is another session management technique. Cookies are a small part of information like a name, a single value and optional attributes such as comment, path and domain qualifiers, a maximum age, and a version number. A servlet sends this cookie information to a web browser. It is saved by the browser and later sent back to the server. The browser probably supports 20 cookies for each Web server, 300 cookies total and may limit cookie size to 4 KB each. You can uniquely identify a client through cookie's value, so cookies are generally used for session management. If the client disables the cookies then it won't work and you cannot maintain a session with cookies.

There are two types of cookies such as Non-persistent cookie and Persistent cookie in servlets. The **Non-persistent cookie** is effective for a single session only and removed each time when the user closes the browser. On the contrary, **Persistent cookie** is effective for multiple sessions, and it is removed only when user logouts.

A cookie is indicated by the Cookie class in the javax.servlet.http package. You can create a cookie by calling Cookie class like the following:

```
Cookie c = new Cookie("userid", "socis");
```

You can send the cookie to the client browser by using addCookie() method of the HttpServletResponse interface like the following:

```
response.addCookie(c);
```

You can retrieve cookies from request using getCookie() of the HttpServletRequest interface like the following:

```
request.getCookie(c);
```

Example -2

The following example demonstrates to you how to create cookies and how these cookies are transferred to another servlet. This example contains one HTML form (cookieForm.html) and two servlet programs (CreateCookieServlet.java and GetCookieServlet.java).

The source codes of the programs are given below:

cookieForm.html

```
<form action="../servlet/CreateCookieServlet" method="post">
Name:<input type="text" name="uname"/><br/>
<input type="submit" value="Submit"/>
</form>
```

CreateCookieServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CreateCookieServlet extends HttpServlet
{
  public void doPost(HttpServletRequest request, HttpServletResponse response)
  {
    try
    {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String name=request.getParameter("uname");
      out.print("Welcome "+name);
      out.print(",  Submit your data for GetCookieServlet");
      //create cookie object
      Cookie c=new Cookie("uname",name);
       //add cookie in the response
      response.addCookie(c);
      out.println("<form action='../servlet/GetCookieServlet'  method='post'>");
      out.println("<input type='submit' value='Submit data'>");
      out.println("</form>");
    }
    catch(Exception e){System.out.println(e);}
  }
}
```

GetCookieServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetCookieServlet extends HttpServlet
{
  public void doPost(HttpServletRequest request, HttpServletResponse response)
  {
   try
   {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     Cookie c[] = request.getCookies();
     out.println("Welcome in SOCIS "+c[0].getValue());
   }
   catch(Exception e){System.out.println(e);}
  }
}
```

Now, you can compile both the servlets program and place class files in the classes folder of the your web application. Also, make an entry in the deployment descriptor file and first run the HTML form program from your browser. When you submit your data by clicking submit button of the HTML program (see figure-4), the control is transferred to CreateCookieServlet program (see figure-5).

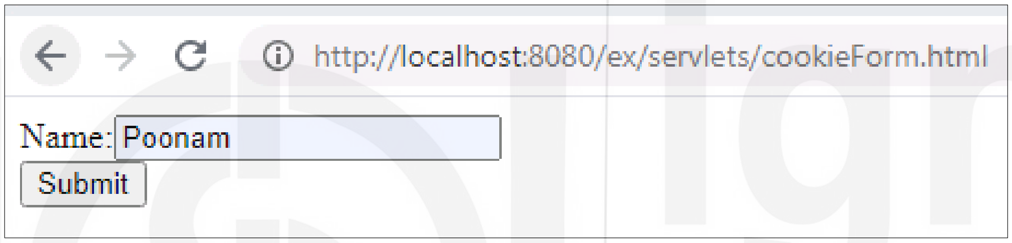
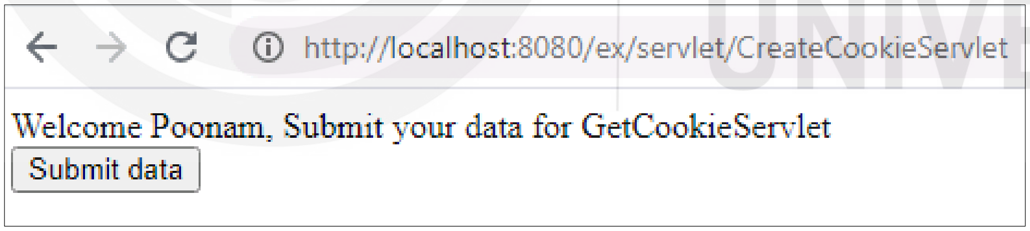


Figure 4: Cookie HTML Form

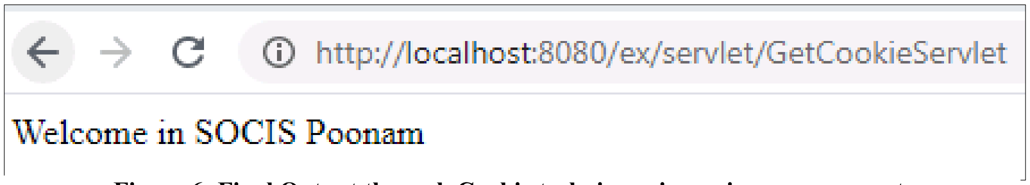
The process of creation and sending of cookies are included in CreateCookieServlet program. Now at this stage, form data is fetched and displayed on screen. When you click on ‘Submit data’ button, the control goes to the GetCookieServlet program (see Figure-5)



```
Welcome Poonam, Submit your data for GetCookieServlet
```

Figure 5: Intermediate output of CreateCookieServlet program

Now, the GetCookieServlet program fetches your data from CreateCookieServlet program and displays this as an output on your monitor screen like the following figure-6.



```
Welcome in SOCIS Poonam
```

Figure 6: Final Output through Cookie techniques in session management

3.2.3 URL Writing

In the previous section, you have learnt two session management techniques i.e. HttpSession Object and Cookies. The third technique that you can use to maintain user sessions is by using URL writing. In this approach, the token (parameter) is embedded in each URL. When client submits request using such URLs, the token is retransmitted to the server. In each dynamically generated page, server embeds an extra query parameter or extra path information. If a browser does not support cookies then in that case URL rewriting technique is the best alternative for session management.
Using URL Writing technique, you can send parameter name/value pairs like the following:

http://myserver.com?name=xyz&age=20

When you click on URL, the parameter name/value pair will transfer to the servlet. The servlet will fetch this parameter by using getParameter() method of HttpServletRequest interface from the requested URL and use it for session management.

Example -3

The following example will show you how to work URL Writing technique with session management. This example comprises the 3 programs such as html form (URLWritingForm.html) and two Servlet program (CreateURLServlet.java and FetchURLServlet.java). The source codes of programs are listed below:

URLWritingForm.html

```
<html>
<head><title>URL Writing Session FORM</title></head>
<body>
<form action="../servlet/CreateURLServlet" method="post">
<fieldset>
<legend>Student Details</legend>
Student Name :<input type="text" name="uname">  <br>
Password: <input type="password" name="pwd"> <br>
<input type="submit" value="Submit">
 </fieldset>
</form>
</body></html>
```

CreateURLServlet.java

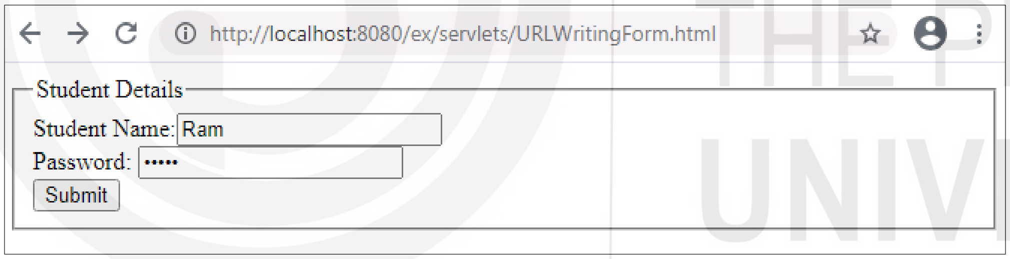
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class CreateURLServlet extends HttpServlet
{
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
     response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String name = request.getParameter("uname");
      String password = request.getParameter("pwd");
```

```
        if(password.equals("socis"))
        {
          response.sendRedirect("FetchURLServlet?username="+ name);
        }
        else
        {out.println("Password is incorrect");}
      }
    }
```

FetchURLServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FetchURLServlet extends HttpServlet
{
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     String name = request.getParameter("username");
     out.println("Welcome in SOCIS,  "+name);
   }
}
```

Now, you can compile both the servlet programs and place class file in 'classes' folder. Also make entry in web.xml file and start web server. Finally, you can run URLWritingForm.html file from your browser. Here, you can submit your data such as student name, password and click on 'Submit' button (see Figure-7)


http://localhost:8080/ex/servlets/URLWritingForm.html

Student Details

Student Name: Ram

Password: •••••

Figure 7: URL Writing HTML Form

After submitting your data, control goes to the CreateURLServlet program. This servlet program fetches your username and password from html form and compares your password statically with given 'socis' password in the program. If a match is done then control is transferred to the FetchURLServlet program else it displays an error message on the server.

Here, you can see parameters value in URL of the FetchURLServlet program in address bar like the following figure-8.

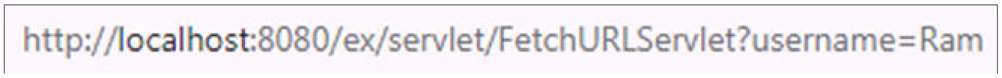
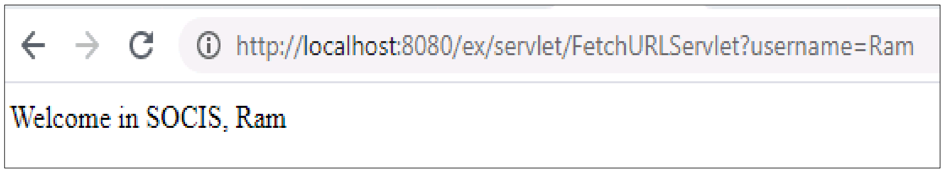

http://localhost:8080/ex/servlet/FetchURLServlet?username=Ram

Figure 8: URL Parameter value in address bar

After fetching the parameter value, the FetchURLServlet program will display output


http://localhost:8080/ex/servlet/FetchURLServlet?username=Ram

Welcome in SOCIS, Ram

Figure 9: Example Output screen for URL Writing technique

as like following Figure-9.

3.2.4 Hidden Fields

Another technique for managing user sessions is by passing a token as the value for an HTML hidden field. You have seen web-form many times on the website. When client submits the form, additional fields will also be sent in the request in the form of hidden fields to keep track of the session. This method gives you the advantage to use this without depending on the browser whether the cookie is disabled or not. It has a disadvantage also as it is not secure because anyone can view the hidden form field value from the HTML file and use it to hack the session. Another disadvantage of using this method is that it needs extra form submission on each page.

You can create a unique hidden filed in the HTML form to keep track of the session like the following format:

```
<input type=”hidden”  name=”userid”  value=”socis”>
```

You can get this hidden field in Java Servlet using the getParameter() method of HttpServletRequest interface.

```
String param1 = request.getParameter("userid");
```

☞ Check Your Progress 1

1. What is session management? What are the different techniques of session management in servlets? Why is session management needed for HTTP protocol? Explain how cookies can be used for session management.

----

2. Write a servlet program by using HttpSession Object of session management. Servlet program will display creation time when user will visit web page for the first time else it displays last access time. Also display session ID in both the conditions.

----

3. What is the difference between a session and cookie?

3.3 SERVLET COLLABORATION

In the previous section, you have gone through the session tracking techniques. Using these techniques, you can track a series of user requests. This section describes to you how servlets share information between two or more servlets.

Servlets running together under the same server, need to communicate with each other for exchanging data information. When two or more servlets can communicate or share common information, you can call it Servlet Communication or Servlet Collaboration.

Servlet collaboration means sharing information among the servlets. This tactic requires each servlet to know the other servlet with which it collaborates. Sometimes, a situation arises in program coding when you may require passing the request from one servlet to another. Also, you may want to include the content from HTML, JSP or Servlet into your servlet. Java provides Servlet-API to accomplish Servlet Collaboration and Servlet-API covers two interfaces namely: javax.servlet.RequestDispatcher and javax.servlet.http.HttpServletResponse. Both interfaces have various methods which are used for sharing information between servlets.

3.3.1 Using RequestDispatcher Interface

RequestDispatcher is an interface which is found in javax.servlet package. There are two methods defined in the RequestDispatcher interface: forward () and include() methods for dispatching requests to/from web resources. Both the methods accept a javax.servlet.ServletRequest and a javax.servlet.ServletResponse object as arguments. The client or browser is not involved in request dispatching.

3.3.1.1 The forward() Method

This method is used to forward a request from a servlet to another web resource like servlet, JSP file or HTML file on the server. When this method is called, control is transferred from the current to the next resource called.

The signature of this method is as follows:

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

3.3.1.2 The include() method

This method is used to include the content of another servlet, JSP page, HTML file in the servlet response. After calling this method, the response of another resource is included in the called resource.

The signature of this method is as follows:

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

For using a servlet forward() or include() method, you first need to obtain a RequestDispatcher object. You can obtain a RequestDispatcher object like the following:

```
RequestDispatcher rd = request.getRequestDispatcher(String resource);
```

Example-4

The following example explains how to use both the forward() and include() method of RequestDispatcher interface to achieve Servlet Collaboration. This example comprises one HTML program (loginForm.html) and two servlets programs (RequestDispatcherServlet.java and WelcomeStudentServlet.java). The LoginForm contains two input fields i.e., name and password. RequestDispatcherServlet program received data from LoginForm and compares statically with given data in the servlet program. If a match is done, control is transferred to the WelcomeStudentServlet program, else control gets back to LoginForm again to re-enter the data.

The source codes of all three programs are listed below:

LoginForm.html

```
<html>
<head></head>
<body>
<form action="../servlet/RequestDispatcherServlet" method="post">
<fieldset>
<legend>Student Details</legend>
Student Name: <input type="text" name="user">
<br>
Password: <input type="password" name="pwd">
<br>
<input type="submit" value="Submit">
 </fieldset>
</form>
</body>
</html>
```

RequestDispatcherServlet.java

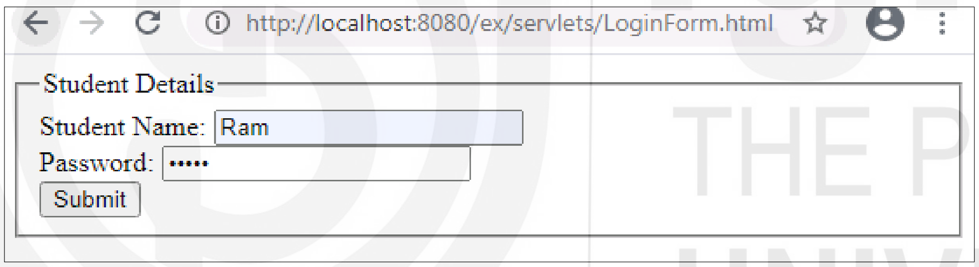
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RequestDispatcherServlet extends HttpServlet
{
   public void doPost(HttpServletRequest req, HttpServletResponse res) throws
   ServletException, IOException
   {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      String name=req.getParameter("user");
      String pass=req.getParameter("pwd");
      if(name.equals("Ram") && pass.equals("socis"))
      {
        RequestDispatcher rd=req.getRequestDispatcher("WelcomeStudentServlet");
        rd.forward(req, res);
      }
      else
      {
        out.print("User name or password is incorrect!");
        RequestDispatcher rd=req.getRequestDispatcher("../servlets/LoginForm.html");
        rd.include(req, res);
      }
    }
}
```

WelcomeStudentServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class WelcomeStudentServlet extends HttpServlet
{
  public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String name=request.getParameter("user");
      out.print("Welcome in IGNOU, "+name+"!");
  }
}
```

You can compile both the servlet programs and make a suitable entry in web.xml file. Now you can start the web server and run HTML form through the browser.

When you run your LoginForm.html, it will display like the following Figure-10 with two input fields. When you enter student name and Password and click on submit button then action control goes to the RequestDispatcherServlet program.



http://localhost:8080/ex/servlets/LoginForm.html

Student Details

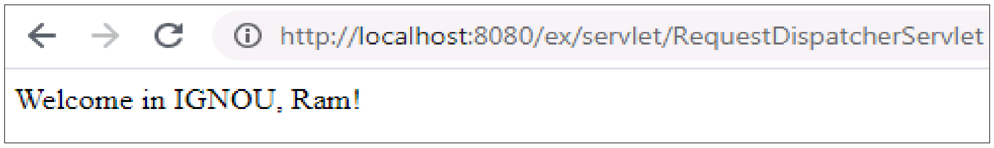
Student Name: Ram

Password: •••••

Submit

Figure10: Login Form for RequestDispatcher servlet example

When you input the Student Name as Ram and Password as socis then it will welcome you (see figure-11); otherwise, it will ask to re-enter values (see figure-12)

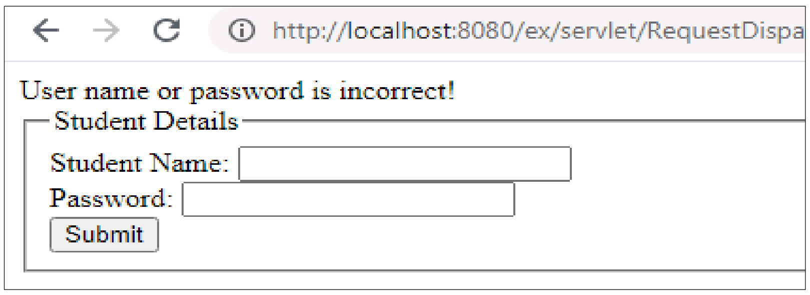


http://localhost:8080/ex/servlet/RequestDispatcherServlet

Welcome in IGNOU, Ram!

Figure 11: Request Dispatcher example, Welcome Screen

When condition is false, it will display a message "User name or password is incorrect" and LoginForm will display again for re-enter input values (see figure-12).



http://localhost:8080/ex/servlet/RequestDispa

User name or password is incorrect!

Student Details

Student Name:

Password:

Submit

Figure 12 Request Dispatcher example, Login Form Screen

3.3.2 Using HttpServletResponse Interface

The HttpServletResponse Interface captures the functionality of a response object that is returned to the client from an HTTP servlet. The interface HttpServletResponse offers many protocol-specific methods which are not available in ServletResponse interface. The interface HttpServletResponse extends the javax.servlet.ServletResponse interface. You have already explored two of the methods in HttpServletResponse at the time of servlet writing in this unit as well as Unit 2. These two methods setContentType() and getWriter() are used when sending output to the browser. The setContentType(java.lang.String type) method is used to set the type of the response data e.g. text, HTML etc. The getWriter() method returns the PrintWriter object for sending text data to the client.

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

Another method of HttpServletResponse interface is addCookie() method which you have already studied in section 3.2.2 of this Unit. Another most important method in HttpServletResponse interface is sendRedirect() method. In the next section, you will learn about this method. You can use this method to redirect the user to another web page.

3.3.2.1 The sendRedirect() Method

The sendRedirect() method of HttpServletResponse interface is used to redirect response to another web resource such as HTML, servlet or jsp file. The sendRedirect() method works at the client side. This method always sends a new request. It can be used within and outside the application. For some applications, you want to send the request outside your web application and then it becomes most useful because it takes more time to fetch resources.

The syntax of sendRedirect() method is as under:

```
public void sendRedirect(String URL) throws IOException;
```

The following example illustrates the simplicity of sendRedirect() method:

```
response.sendRedirect("http://www.ignou.ac.in");
```

In the previous section, you have studied forward() method. Function of both the methods forward() method of RequestDispatcher interface and sendRedirect() of RequestDispatcher interface are almost similar but they differ also.

Difference between forward() and sendRedirect() method

Both methods bring the user to a new web resource. Both are similar, but knowing the fundamental difference between the two can help you write a servlet more efficiently. The difference between the two methods is as follows:

- The forward method can be used to redirect the request without any help of client browser, and control transfer remains within-applications resources only, whereas sendRedirect method can redirect the request to client browser and control transfer goes outside the current domain.
- In forward method, HttpServletRequest object and HttpServletResponse object are passed to the new resource, whereas in sendRedirect method, previous HttpServletRequest object is lost. For passing information between the original and new request, you can pass the information as a query string appended to the destination URL.

3.3.3 Using ServletContext Interface

In servlet programming, servlet context is the environment where the servlet runs. The ServletContext's object is created by the web container at the time of deploying the project. Using this you can use to access all the resources available within the application and to store attributes which other servlets with the same context can use. You can use one context per "web application" per Java Virtual Machine. The ServletContext object is used to get configuration information from deployment descriptor file (web.xml), which will be available to any servlet or JSPs that are component of the 'webapps'.

The ServletContext Interface defines various methods which are used by servlets to communicate with its servlet container. For example, dispatch requests or writes to a log file to get the MIME type of a file. The object of ServletContext can be added and retrieved from the context using the setAttribute and getAttribute methods.

How to get the Object of ServletContext

```
ServletContext app = getServletContext();
//This is convenient way to get the ServletContext object
OR
ServletContext app = getServletConfig().getServletContext();
//You can get the ServletContext object from ServletConfig object
```

When you have defined ServletContext object, you can set attributes of ServletContext object by using the setAttribute() method. From now, ServletContext object is available to all the servlets of the web application. By using the getAttribute() method, other servlets can get the attribute from the ServletContext object.

3.3.3.1 setAttribute() Method

This method stores an object and binds the object with the given attribute name in the application scope. It means that the said object is accessible from any servlet within the same web application. If attribute name already exists in the ServletContext, the old bound object will be replaced by the object passed to this method.

```
void setAttribute(java.lang.String name, java.lang.Object obj)
```

This method has two parameters:

String name: By providing the value of this argument, you can specify the name of attribute.

Object obj: By providing the value of this argument, you can specify the value of the named attribute.

3.3.3.2 getAttribute() Method

This method is used to get attribute with the given name from context. It returns the value of named attribute as an Object or NULL if there is no attribute by that name.

```
Object getAttribute(String name)
```

Example- 5

The following example explains to you how setAttribute() and getAttribute() method works with the ServletContext.

The example uses two servlets: ServletDemo1 and ServletDemo2 and one HTML Form (Login.html). This form comprises two input fields such as name and percentage of student. A student must have a percentage of more than 80. When the user submits the form, control will transfer to the ServletDemo1 servlet.

The ServletDemo1 servlet received two input values i.e. name and percentage by using getParameter() method from html form. This servlet uses setAttribute() method to bind name attribute and does this by first obtaining the ServletContext object. Here, this servlet checks percentage if percentage is more than 80, then the control will pass to the ServletDemo2 servlet, otherwise control will transfer to Login form to fill data again.

In ServletDemo2 servlet, after getting the ServletContext object, you can use getAttribute() method to get name attribute.

The code source of the three programs is listed below:

Login.html

```
<form action="../servlet/ServletDemo1" method="post">

<fieldset>
<legend>Student Details</legend>

Student Name: <input type="text" name="username">
<br>
Percentage: <input type="text" name="percent">
<br>

<input type="submit" value="Submit">
 </fieldset>
</form>
```

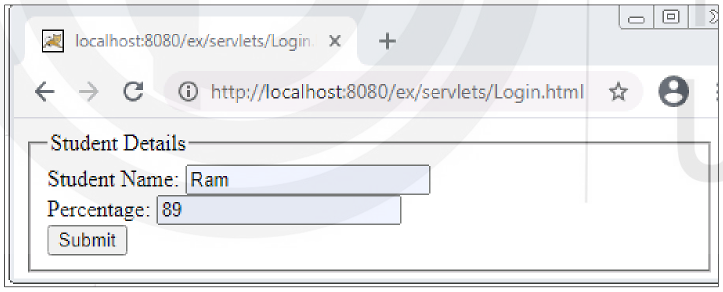
ServletDemo1.java

```
import java.io.*;
import javax.servlet.*;
public class ServletDemo1 extends GenericServlet
{
  public void service(ServletRequest request, ServletResponse response) throws
  ServletException, IOException
  {
    //Creating ServletContext object
    ServletContext sc = getServletContext();
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("Using ServletContext object to set and read attributes");
    out.println("<br/>");
    String uname = request.getParameter("username");
    int Per = Integer.parseInt(request.getParameter("percent"));
    //Setting name attribute to be shared between multiple servlets
    sc.setAttribute("Name", uname);
    if( Per > 80)
    {
            RequestDispatcher rd = sc.getRequestDispatcher("/servlet/ServletDemo2");
            rd.forward(request,response);
        }
        else
        {
            out.print("Data is incorrect!");
            out.println("<br/>");
            out.print("Fill your data again");
            RequestDispatcher rd = sc.getRequestDispatcher("/servlets/Login.html");
            rd.include(request,response);
        }
    }
}
```

ServletDemo2.java

```
import java.io.*;
import javax.servlet.*;
public class ServletDemo2 extends GenericServlet
{
    public void service(ServletRequest request, ServletResponse response) throws
    ServletException, IOException
    {
        ServletContext sc = getServletContext();
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<b>" + "Congratulations!  " + sc.getAttribute("Name") + "</b>");
    }
}
```

Now, you can compile both servlets and create appropriate entry in web.xml file. Finally, you can start server and run the Login form from your browser. The output of the html program is displayed like the following figure13.


localhost:8080/ex/servlets/Login

http://localhost:8080/ex/servlets/Login.html

Student Details

Student Name: Ram

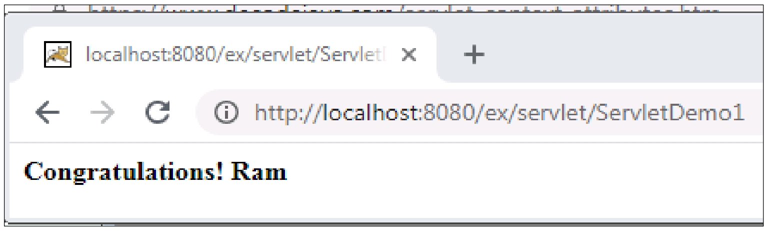
Percentage: 89

Submit

Figure 13: Login form

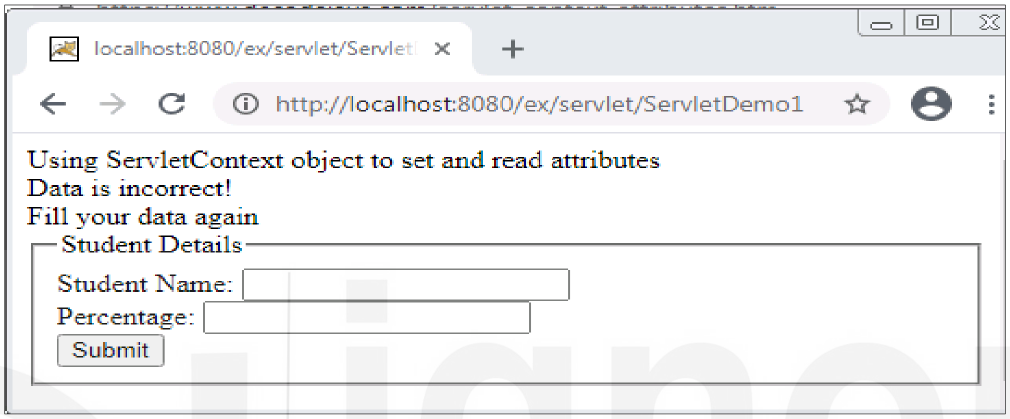
Here you can submit your data. The flow controls of servlets depends on your percentage. If you submitted more than 80, you will get congratulations (see figure-14), this procedure is defined in the ServletDemo2 servlet otherwise you will get back to login again to re-enter data (see figure- 15)

When condition will be true, you will get following figure-14:


localhost:8080/ex/servlet/ServletDemo1

```
Congratulations! Ram
```

Figure 14: Output of setting and getting attribute example


localhost:8080/ex/servlet/ServletDemo1

```
Using ServletContext object to set and read attributes
Data is incorrect!
Fill your data again
Student Details
Student Name:
Percentage:
```

Figure 15: Output Screen for re-enter data

When the condition is false, you will get the following to re-enter data:

☞ Check Your Progress 2

1. What is servlet collaboration? How many ways can servlet communicate with each other? What is the purpose of RequestDispatcher Interface?

   ----

2. What is the difference between forward() and sendRedirect() method?

   ----

3. Write a servlet program for opening the IGNOU website by using sendRedirect() method.

3.4 DATABASE CONNECTIVITY

In the previous section, you have studied more about the advanced features of servlets with examples, but all this work has been done statically without any database connectivity. This section will provide you in-depth knowledge of data access, specifically insertion and retrieval of data to/from a database.

Database access is one of the important features of Web application. Java has its own technology for database connectivity called JDBC (Java Database Connectivity). JDBC provides a standard library for accessing a wide range of relational databases like MS-Access, Oracle and Microsoft SQL Server. Using JDBC API, you can access a wide variety of different SQL databases. The core functionality of JDBC is found in java.sql package.

Consider a table named as Student created in Microsoft SQL Server database with Roll No, Name and Program name. This table is used in both the following sections. This section defines example(s) for storing/retrieving data into/from a Microsoft SQL Server using type-4 JDBC driver. You can run these programs on any web server such as Tomcat. For running these programs, you need a JDBC driver in .jar form and placing them in lib directory under the web server.

3.4.1 Insert data in Database

For inserting data into a database, you can use the following example.

Example - 6

The following example will demonstrate to you how to store data in a database using servlet. For this, create a HTML Student Form and one Servlet for storing data into a database.

The student form contains student's details like student enrolment no., student name and Programme name. You will have to create a student table with related fields of student form. When you submit these details, access 'DataStoreServlet' to store data into the database.

The DataStoreServlet program is written similar to the above servlet programs with database query and 'try' and 'catch' clause of standard Java mechanism.

The source codes of both the files are listed below:

StudentForm.html

```
<html>
<body>
<form action="../servlet/DataStoreServlet" method="post">
<h3>Indira Gandhi Open University </h3>

<fieldset>
<legend><b>Student Details </b></legend>

Enrolment No: <input name="sid" type="text"/><br>
Student Name : <input name="sname" type="text" value="" ><br>
Programme     : <input type="text" name="prg" value=""> <br>
<input type="submit" value="Submit" />

</fieldset>
</form>
```

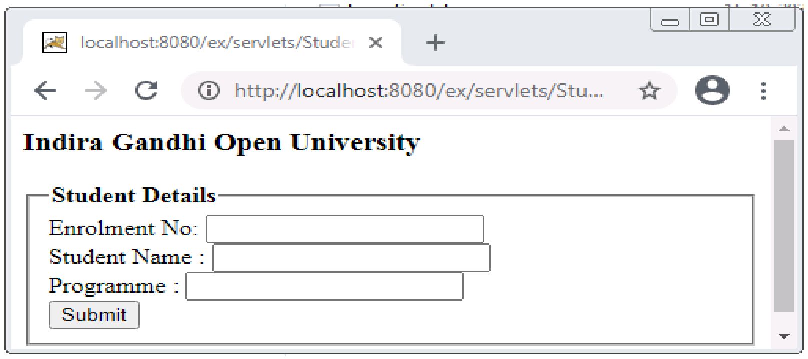
```
</body>
</html>
```

In the following servlet program, the first step is to get the data from 'StudentForm.html' program using request.getParameter() method. After connecting to database, an insert query is executed using executeUpdate() method.

DataStoreServlet.java

```
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DataStoreServlet extends HttpServlet
{
  public void doPost(HttpServletRequest req, HttpServletResponse res) throws
  ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String rollNo = req.getParameter("sid");
    String StuName = req.getParameter("sname");
    String prgName = req.getParameter("prg");
    //create connection object
    Connection con = null;
     // create statement object
    Statement stmt = null;
    // connection string using Type-4 Microsoft SQL Server driver
    // you can also change the next line with your own environment
    String url=
    "jdbc:microsoft:sqlserver://127.0.0.1:1433;user=sa;password=poonam;DatabaseN
    ame=SOCIS";
    try
    {
      // load JDBC type-4 SQL Server driver
      Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
      con = DriverManager.getConnection(url);
      if (con != null)
      {
        stmt = con.createStatement();
        //insert query
        String rsql ="insert into student
        values("+rollNo+",'"+StuName+"','"+prgName+"'"+")";
        //execute query
        stmt.executeUpdate(rsql);
        out.println("Your data is successfully stored in database");
      }
      if(con == null)
      { con.close();  // release connection }
    }
     // end of try clause catch(SQLException se)
    { out.print("SQL:"+se.getMessage());}
    catch(Exception e)
    { out.print("Exception:"+e.getMessage());}
  }
}
```

As earlier, you can compile the above servlet and make related entry in deployment descriptor file (web.xml) then start your web server and run your StudentForm.html file from your browser. When you run your form, it will display output like the following figure-16:


localhost:8080/ex/servlets/Stude
http://localhost:8080/ex/servlets/Stu...

Indira Gandhi Open University

Student Details

Enrolment No:

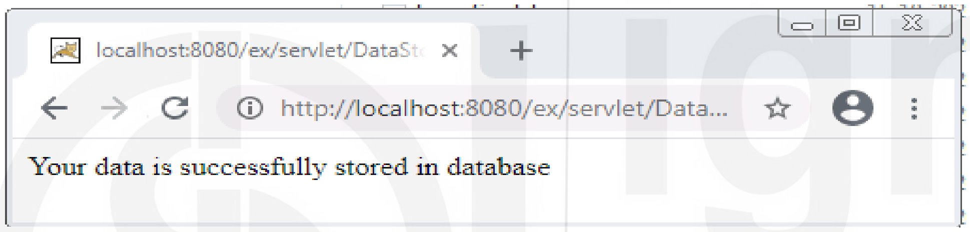
Student Name :

Programme :

Submit

Figure 16: Student data Form for storing data into database

In the above screen, when you will enter values then the following screen (figure-17) will show you a message for successful data storage.


localhost:8080/ex/servlet/DataSt
http://localhost:8080/ex/servlet/Data...

Your data is successfully stored in database

Figure 17: Data stored in persistent storage

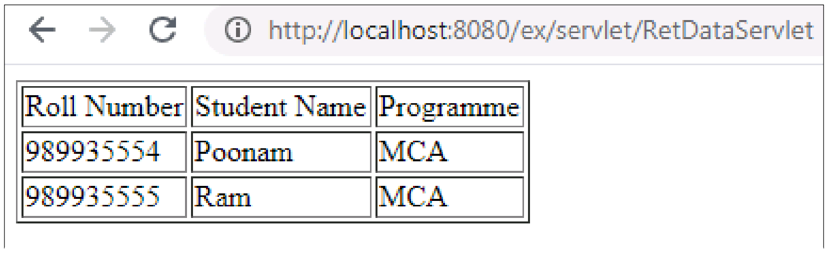
Example-7

The following example gives you an illustration about how to retrieve data from the database. After execution of the above DataStoreServlet program, you have stored sufficient data into the database. Now, you will execute the following code for retrieving the data from the database. In this program, one additional ResultSet object is used for retrieving data from the select query. The data is retrieved from ResultSet object by using getXXX() method such as getInt() and getString(). Note that if the column name is an integer type, then you should use getInt() method instead getString() method. The following RetDataServlet program is listed for retrieving data from database.

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RetDataServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws
  ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    Connection con = null;                    //create connection object
    Statement stmt = null;                    // create statement object
    ResultSet rs = null;                     // create ResultSet object
    // connection string using Type-4 Microsoft SQL Server driver
    // you can also change the next line with your own environment
    String url=
    "jdbc:microsoft:sqlserver://127.0.0.1:1433;user=sa;password=poonam;DatabaseN
    ame=SOCIS";
    try
    {
      // load JDBC type-4 SQL Server driver
      Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
      con = DriverManager.getConnection(url);
      if (con != null)
      {
        stmt = con.createStatement();       // select SQL statement
        String rsql ="select * from Student";
        rs = stmt.executeQuery(rsql);              //execute query
        out.println("<table border=1><tr><td>Roll Number</td><td>Student
        Name</td><td>Programme</td></tr>");
        while( rs.next() )
        {
          out.println("<tr><td>" + rs.getInt("RollNo") + "</td>");
          out.println("<td>" + rs.getString("Student_Name") + "</td>");
          out.println("<td>" + rs.getString("Programme") + "</td></tr>");
        }
        out.println("</table>");
      }
      if(con == null) {con.close();}
    }
    catch(SQLException se){ out.println("SQL:"+se.getMessage());}
    catch(Exception e){ out.println("Exception:"+e.getMessage());}
  }
}
```

Now, you can compile the above servlet and make entry in deployment descriptor file (web.xml), start your web server and run your servlet program from your browser. After running the RetDataServlet program, the data will be displayed on your output screen like following figure-18:

If you want to build your project using Oracle or other database as back end then you can change only port no. and relevant JDBC driver name in above servlet source code.


http://localhost:8080/ex/servlet/RetDataServlet

Roll Number	Student Name	Programme
989935554	Poonam	MCA
989935555	Ram	MCA

Figure 18: Display data from database

3.5 SUMMARY

In this unit, you have learned how to use session management using four techniques: HttpSession object, Cookie, URL Writing, and Hidden form field. You have been shown a number of examples to demonstrate the use of session management in the servlets. In addition, this Unit introduced you to Servlet collaboration which is all about sharing information among the servlets. The Servlet-API provides the RequestDispatcher interface to achieve Servlet Collaboration. This interface is useful for forwarding the request to another web resource and including the resource in the current servlet. Apart from these, this unit also discussed database access using Java Database Connectivity (JDBC) technology.

3.6 SOLUTIONS/ANSWERS TO CHECK YOUR PROGRESS

☞ Check Your Progress 1

1) Session management allows servlets to maintain information about a series of requests from the same user. Session in Java Servlet is managed through four techniques such as HttpSession API, Cookies, URL rewriting and Hidden Field.

   HTTP is a stateless protocol. It means a HTTP server does not remember any state information. So one needs to maintain state using session management techniques.

   For this cookie can be used in the following way for managing sessions in java servlets:

   - You can create a cookie by calling Cookie class like the following:

     Cookie c = new Cookie(“userid”, “socis”);

   - You can send the cookie to the client browser by using addCookie() method of the HttpServletResponse interface like the following:

     response.addCookie(c);

   - You can retrieve cookies from the request by using getCookie() of the HttpServletRequest interface like the following:

     request.getCookie(c);

2) The source code of servlet program by using HttpSession object is as under:

```
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;
public class sessionServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws
  ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    HttpSession session = req.getSession(true);
    if ( session.isNew())
    {
      out.println("New Session : " + session.isNew() );
      out.println("<br/>");
      out.println("Session ID : " + session.getId());
      out.println("<br/>");
```

```
        out.println("creation Time : " + new Date(session.getCreationTime()));
      }
      else
      {
        out.println("New Session : " + session.isNew() );
        out.println("<br/>");
        out.println("Session ID : " + session.getId());
        out.println("<br/>");
        out.println("Last Access Time : " + new
        Date(session.getLastAccessedTime()));
       }
    }
  }
```

3) Both the Session and Cookie are used to store information. The session is stored in server-side machine, whereas Cookies are stored on the client-side machine. Session should work regardless of the settings on the client browser. Session data will be available to all pages on the site during the particular session of visit. Session can store objects, and cookies can store only strings. Cookie expires depending on the lifetime you set for it whereas a Session ends when a user closes the browser or after leaving the site.

☞ Check Your Progress 2

1) Servlet collaboration means sharing information among the servlets. This tactic requires each servlet needs to know the other servlet with which it collaborates. The Servlet API provides the following interfaces and their methods which are responsible for sharing information between servlets:

   - RequestDispatcher Interface : include() and forward() method;
   - HTTPServletResponse Interface : sendRedirect() method;
   - ServletContext Interface : setAttribute() and getAttribute() method;

   RequestDispatcher interface is found in javax.servlet package. There are two methods defined in the RequestDispatcher interface: forward () and include() methods for dispatching requests to/from web resources.

2) Both the methods bring the user to a new web resource. Both are similar but there is a fundamental difference between the two. The difference between two methods is as follows:

   - The forward method can be used to redirect the request without the help of the client browser and control transfer remains within-applications resources only. The sendRedirect method can redirect the request to the client browser and control transfer goes outside the current domain.
   - In forward method, HttpServletRequest object and HttpServletResponse object are passed to the new resource whereas in sendRedirect method, previous HttpServletRequest object is lost. For passing information between the original and new request, you can pass the information as a query string appended to the destination URL.

3) The source code of servlet by using sendRedirect() method is given below:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class sendRedirectServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws
  ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    res.sendRedirect("http://www.ignou.ac.in/");
  }
}
```

3.7 REFERENCES/FURTHER READING

- …Jason Hunter, and William Crawford "Java Servlet Programming", O'Reilly Media, Inc., 2001.
- …Kathy Sierra, Bryan Basham, anddd Bert Bates ,"Head First Servlets and JSP", O'Reilly Media, Inc., 2008.
- … Budi Kurniawan , "Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions: A Developer's Guide to Scalable Solutions", *Techmedia* , 2002.
- …Pratik Patel and Karl Moss, "Java Database Programming with JDBC: Discover the Essentials for Developing Databases for Internet or Intranet Applications", Coriolis Group, 1996.
- … https://www.w3adda.com/servlet-tutorial
- … https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletContext.html
- … https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpSession.html
- … list of drivers: http://www.devx.com/tips/Tip/28818