JAVA SERVLETS

INTRODUCTION

Java servlets are programs that run on web server. Java applets are programs that are embedded into the web pages. When browser loads the web page the applets byte code is downloaded and executed by Java Virtual Machine. In order to execute an applet you should have a java enabled web browser. While the servlet code is executed at the server side the hence java compatibility web browser problem is solved by servlets.

SERVLETS

Servlets are java programs that run on web or application servers, servlets are also called server side programs i.e the code of the program executes at server side, acting as a middle layer between request coming from Web browsers or other HTTP clients and databases or applications on the HTTP server. A servlet container uses a Java Virtual Machine to run servlet code as requested by a web server. A servlet dynamically loaded module that services requests from Web server. It runs entirely inside the Java Virtual Machine.

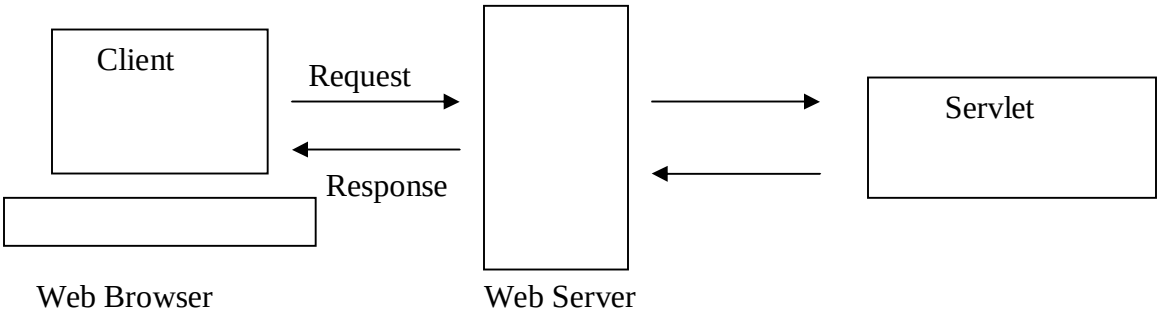


Fig 20.1 Servlet Execution

Servlets are not tied to specific client server protocol but they are most commonly used with HTTP. Since servlets run inside the web server they do not display a graphical user interface.

APPLICATION OF JAVA SERVLETS

Servlets can be used for the following applications:

- Servlets are used for developing on-line shopping store web page. When servlet receives the posted data, it will process it and fullfils the client request .
- Servelets are usefuls for developing distributed applications.

ADVANTAGES OF SERVLETS

Java Servlets are efficient, portable, robust, extensible and secure

EFFICIENT

A Servlet code is loaded into memory once. After the servlet is loaded, handling new request is only matter of calling a service method. In a CGI, if there are N requests to the same CGI program, the code for the CGI program is loaded into memory N times. While in case of servlets there are N thread, but only a single copy the servlet would be loaded. Loading a new executable for every request is expensive technique. The java servlet provides an is efficient technique to it.

PORTABLE

Servlets are java program, hence the servlet code is independent of machine architecture. They can be moved to any machine or any operating system.

ROBUST

Because servlets are developed with access to entire JDK, servlet support several capabilities they are very powerful. It has garbage collector to prevent problems with memory leaks. Servlets can also maintain information from request to request, simplifying techniques like session tracking and caching of previous computations.

EXTENSIBLE

Servlets are developed using java they can be extended into new objects that are better than the previous one. You can implement a base servlet that builds and initializes the search tool and then extend it to display transaction – specific response.

SECURE

Since servlets run on server side they are secure. They inherits the security provided by the Web Server.

The CGI programs are often executed by general – purpose operating shells. So, the CGI programmer must be careful to filterout characters such as backquotes and semicolons that are treated specially by the shell.

Table 201. Server with Buit –in Servlet Support

Product	**Vendor**
Apache Web Server	Apache
Java Web Server	Sun Microsystem
Enterprise Server	Netscape
Internet Connection Server	IBM
Sun Web Server	Sun Microsystem
Domino Go Web Server	Lotus
JRun	MacroMedia

SERVLET ARCHITECTURE

A servlet is a java program. Servlets are created by extending **GenericServlet** or **HttpServlet** class which are available in package javax.servlet. and javax.servlet.htpp respectively. Every time a server receives a request that points to a servlet, it calls servlet's service method. Hence service method be defined to to provide customized functionality. The other two methods are init () and destroy (), the init () method is called only once when servlet is loaded in the memory, while destroy () method is executed only once when servlet is unloaded from the memory.

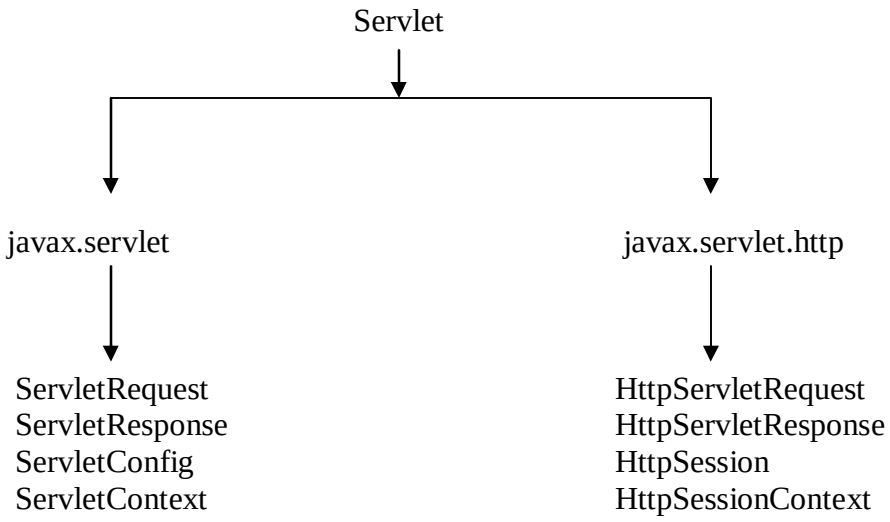


Fig 20.2 Servlet Interfaces

GenericServlet

GenericServlet is the core class in javax.servlet package. Generic servlet may be used to create servlets by extending it. GenericServlet provides methods init() and destroy() and the methods in ServletConfig interface.

HttpServlet

HttpServlet inherits basic Servlet functionality by extending GenericServlet.HttpServlet must override methods such as service(), doGet(), doPost(). Its service method receive two parameters they are HttpServletRequest and HttpServletResponse. The doGet() method is called in response to an HTTP GET request, used to send client data. The doPost() is called in response to an HTTP POST request from HTML form.

```
public class ServletDemo extends HttpServlet
{
        public void init()
        {
                /* used to initialize resources  */
```

```
        }
        public void service()
        {
                /* used to fulfill client request  */

        }

        public void destroy()
        {
                /* Release the resources */
        }
}
```

Structure of servlet program

Program 20.1 HelloWorld.java

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

public class HelloWorld extends GenericServlet
{
        public void service(ServletRequest request,
                                        ServletResponse response )
                                        throws IOException
        {

                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<B> Hello World <B>");
                pw.close();


        }
}
```

The service method throws IOException. The statement response.setContent("text/html") is used to set the MIME type for HTTP header it specify the response type of document displayed at the client browser. PrintWriter is a class is used to send content back to the

browser, and println method is used to print content to the browser with the help of handle created with PrintWriter class.

COMPILING AND RUNNING SERVLET PROGRAM

Compiling and running a servlet program requires a Web Server to be installed on your computer. The Web Server contains the necessary classes required to create a servlet program. The following example illustrate the compilation of java servlet program with the help of Java Servlet Development Kit 2.0

- After creating the above program save program with name HelloWorld.java and move to the MS-DOS prompt.
- Move to the foldder where you have java servlet program
- At the MS-DOS prompt type the path and classpath of java servlet
- path = G:\Program Files\Java\jdk1.6.0_01\bin
- set classpath=G:\JSDK2.0\lib\jsdk.jar
- compile java program as : javac HelloWorld.java
- copy HelloWorld.class file and paste it to the example folder of JSDK2.0
- open servlet.properties files in notepad and type mapping to the servlet as: servlet.code.Hello=HelloWorld
- move to the bin folder of JSDK2.0 start servletrunner program by double clicking it.
- Start the web browser by double clicking it, and at its address bar type : http://localhost:8080/servlet/HelloWorld

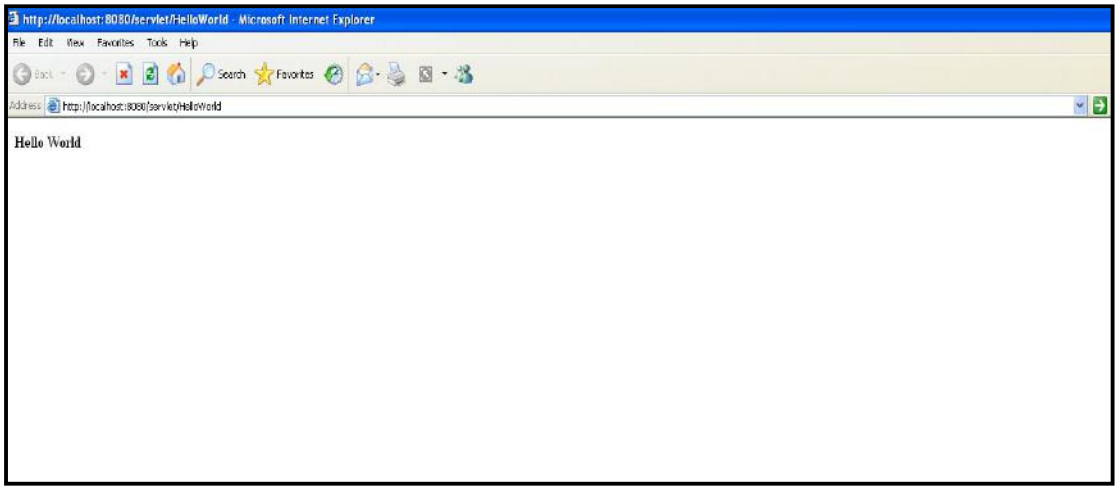


Fig 20.3 Java Servlet

In the above URL the 8080 specifies the port number where web service runs.

Compiling and running the same program with the help of **Apache Tomcat Web Server 4.x** type the above java program with a modification to it. Executing it with the help of invoker servlet.

Program 20.2 HelloWorld1.java

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

public class HelloWorld1 extends GenericServlet
{
        public void service(ServletRequest request,
                                                ServletResponse response )
                                                throws IOException
        {

                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<title> Hello World 1</title>");
                pw.println("<B> Hello World, Executed with the Apache Tomcat Web
                Server<B>");
                pw.close();
```

```
        }
}
```

- Move to the MS-DOS prompt and set classpath and path of java compiler.
- path = \Install_dir\Java\jdk1.6.0_01\bin
- set classpath=C:\Program Files\Apache Group\Tomcat 4.1\common\lib\servlet.jar
- compile java program as javac HelloWorld1.java
- copy class file to the classes folder under web-inf folder
- \install_dir\Apache Group\Tomcat 4.1\webapps\ROOT\WEB-INF\classes
- open web.xml file from install_dir\conf\web.xml
- To enable the invoker servlet remove comments from web.xml file The mapping for the invoker servlet. Uncomment it and save it
- Start tomcat web server
- Open web browser and type the url : http://localhost:8080/servlet/HelloWorld1

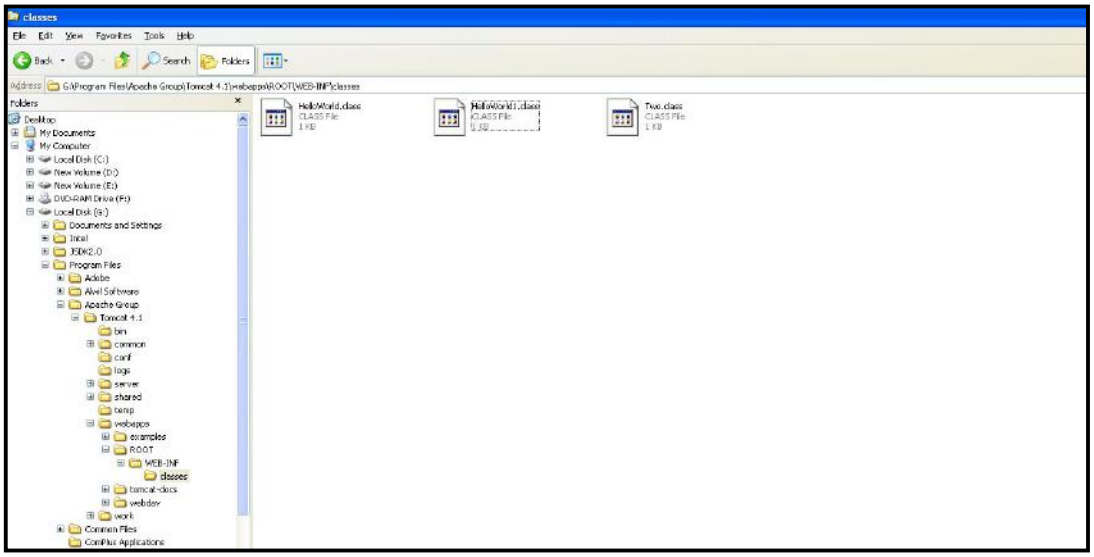


Fig 20.4 paste class file here

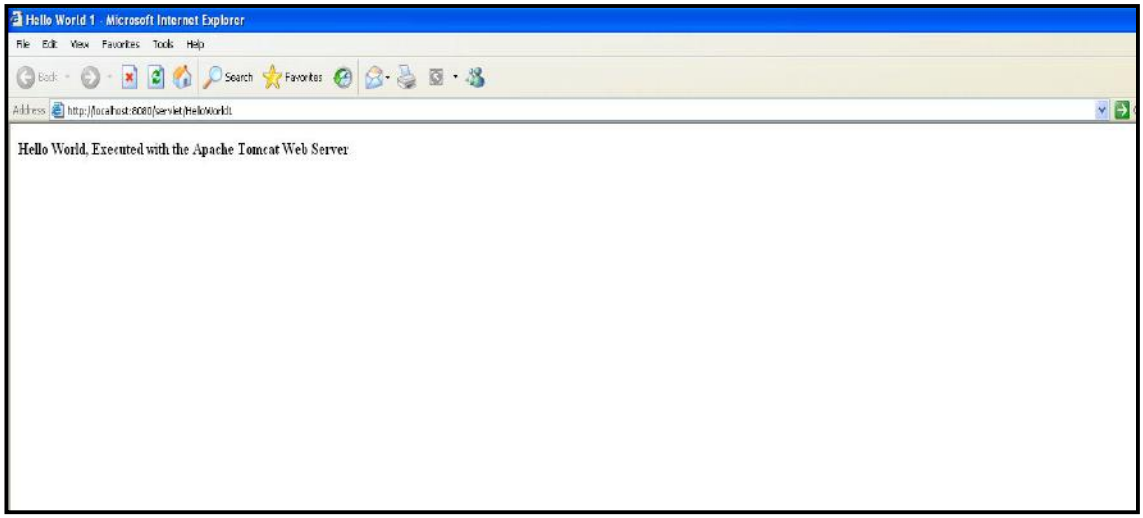


Fig 20.5 Java Servlet output with Tomcat Web Server

The above code is run with the help of tomcat web server. The <title> </title> tag is used to specify title at the top of web browser. The <B> </B> tag is used to print the text in bold size.

DEPLOYING WEB APPLICATION USING TOMCAT

1. Create a sub directory under the \install_directory\webapps move your application to this directory. A context with the directory name will be created automatically.
2. Create a directory WEB-INF under subdirectory created in step 1. The WEB-INF stores files such web.xml which is used to keep application configurations.
3. Create classes subdirectory under the WEB-INF. The classes directory is used to store the java servlets class files.
4. Create a subdirectory lib under WEB-INF. This is used to store the jar files and native libraries.

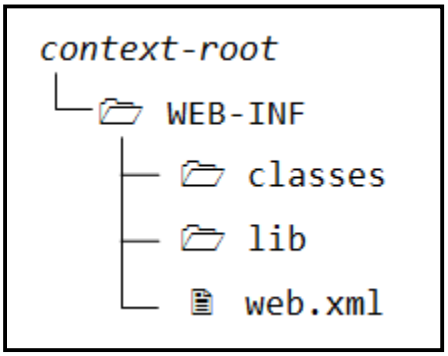


Fig 20.6 Context Root Structure

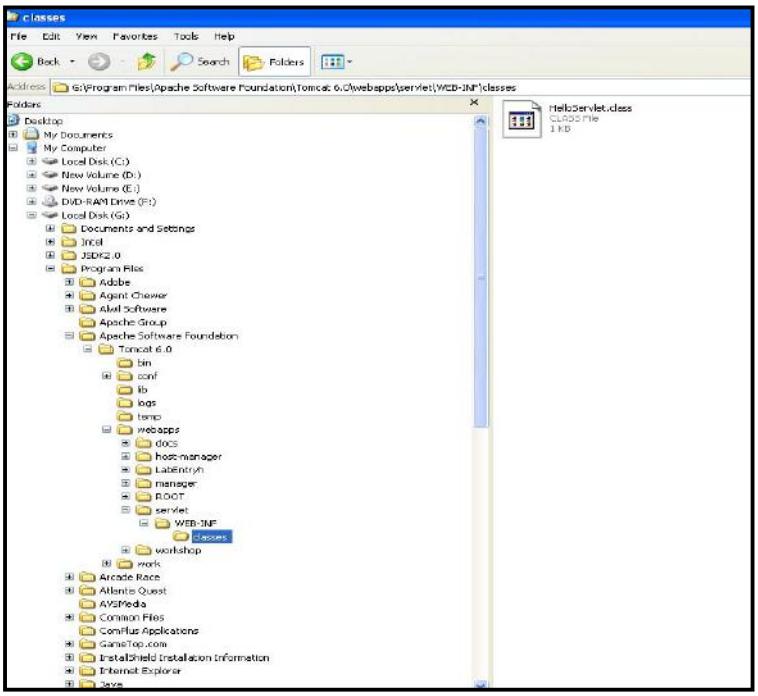


Fig 20.7 The Context – root servlet directory structure

5. Create web.xml file under the WEB-INF folder and add following mapping to it.

web.xml

```
<web-app>
  <servlet>
    <servlet-name>HelloServlet</servlet-name>
```

```
  <servlet-class>HelloServlet</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>HelloServlet</servlet-name>
  <url-pattern>/*</url-pattern>
 </servlet-mapping>
</web-app>
```

6. Start Tomcat Web Server
7. Type url at address bar http://localhost:8080/servlet/HelloServlet
8. create a different context **workshop** i.e directory under webapps directory of Tomact directory structure.
9. Repeat steps 2 to 6 and invoke the web application from web browser with the following address.
   http://localhost:8080/workshop/HelloServlet

The above statement executes same java servlet application i.e HelloServlet from a context **workshop**.

CREATING MORE THAN ONE MAPPING

The above created servlet can be invoked more than one url. This can be accomplished by servlet-mapping tag in web.xml file. After creating more than one mapping the contents of the file web.xml will be as follows:

```
<web-app>
 <servlet>
  <servlet-name>Hello</servlet-name>
  <servlet-class>HelloServlet</servlet-class>
 </servlet>

 <servlet-mapping>
```

```
    <servlet-name>Hello</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>

 <servlet-mapping>
    <servlet-name>Hello</servlet-name>
    <url-pattern>/home</url-pattern>
  </servlet-mapping>
</web-app>
```

LIFE CYCLE OF JAVA SERVLET

The servlet cycle has three methods these init () for initialization of the resources, service() handles zero or more requests and the destroy() method is invoked at end of servlet life it releases all resources which were allocated previously.

init()

The init() method is where the servlets life begins. It is called by the server immediately after the servlet is instantiated. The database connection, opening a file, or a network connection may be established through it. The init method is not called again for each request. The servlet is normally created when a user first invokes a URL corresponding to servlet.

public void init (ServletConfig config) throws ServletException

The following are the task performed while implementing init() mehtod

- Reading initializing parameters using ServletConfig object
- Initializing a database driver
- Opening a database connection
- Initialization of objects

service()

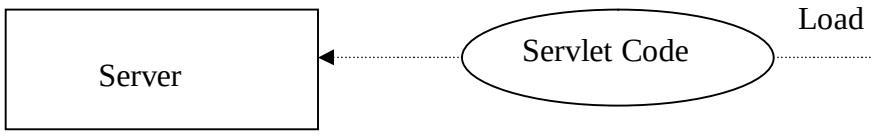
The service method handles all requests sent by a client. Each time the server receive a request for a servlet, the server spawns a new thread and calls service method. The service method checks the request type GET, POST, PUT, DELETE etc. For, every request to the servlet, its service method is called. Two objects ServletRequest and ServletResponse are passed as argument to it.

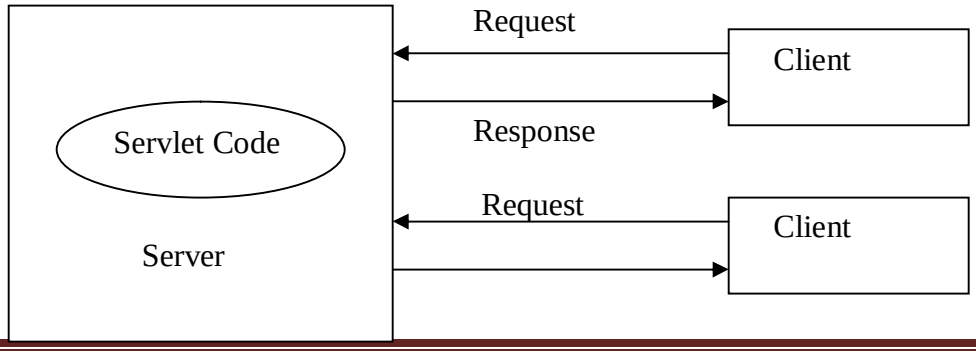
public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException

destroy()

This method signifies the end of a servlet life. The resources that are previously allocated are destroyed. The method is invoked by server administrator or programmatically by servlet itself.

public void destroy()





Response

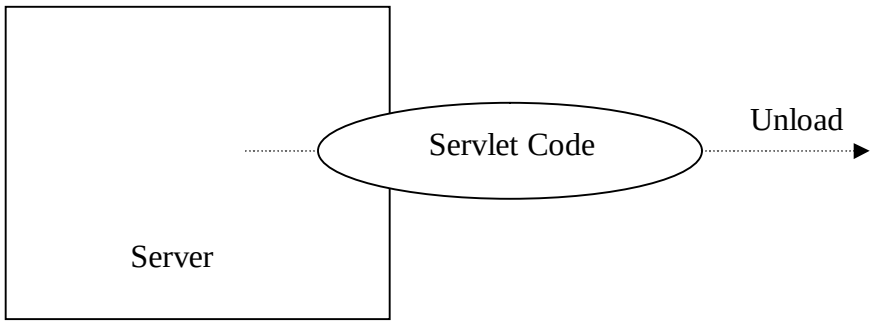


Fig 20.5 Servlet Life Cycle

- A server loads and initializes the servlet
- The servlet handles zero or more client request
- The server removes the servlet

READING HTML FORM PARAMETERS

Reading parameter requires a html form with some text fields which are used to input values. The HTML form reads the values and send it to the servlet by using the action URL. The input values are read by the java servlet program and send back to browser for the display.

Two.html

```
<html>
<head>
<title> Reading Parameters </title>
<body bgcolor="#FDF5E6">
<h1 align ="center"> Reading Parameters </h1>
```

```
<form action ="http://localhost:8080/servlet/Two">
First Parameter : <input type = "text" name = "param1"><br>
Second Parameter: <input type = "text" name = "param2"><br>
<center>  <input type = "submit"></center>
</form>
</body>
</html>
```

Program 20.3 Two.java

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

public class Two extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res )
    throws ServletException, IOException
    {

      res.setContentType("text/html");
      PrintWriter pw = res.getWriter();
      pw.println(req.getParameter("param1"));
      pw.println(req.getParameter("param2"));

    }
}
```

The getParameter method is used to read the HTML form parameters and displayed to the web browser with the help of PrintWriter stream. The fig 20.7 shows the result after reading the input parameter.

Res.getWriter() returns a PrintWriter object that can send character text to the client.

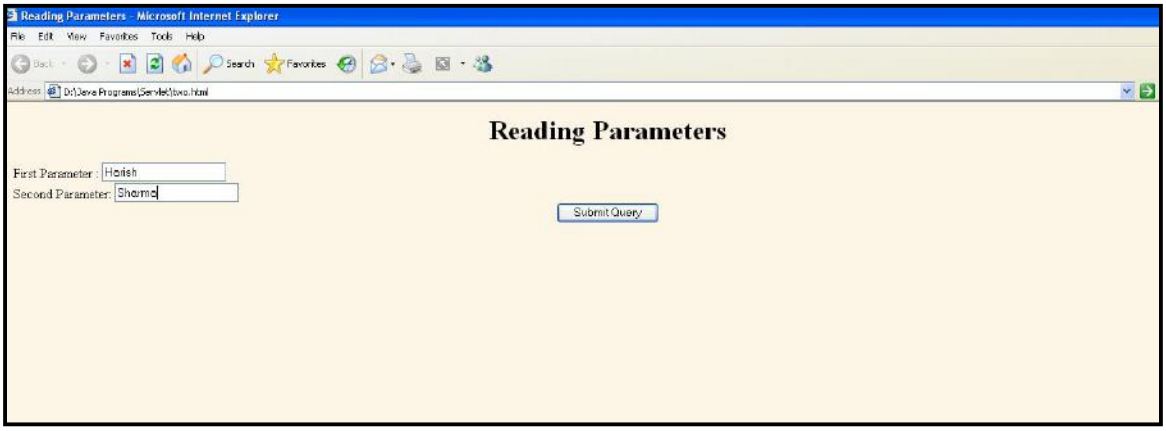


Fig 20.6 HTML Form Reading two parameters

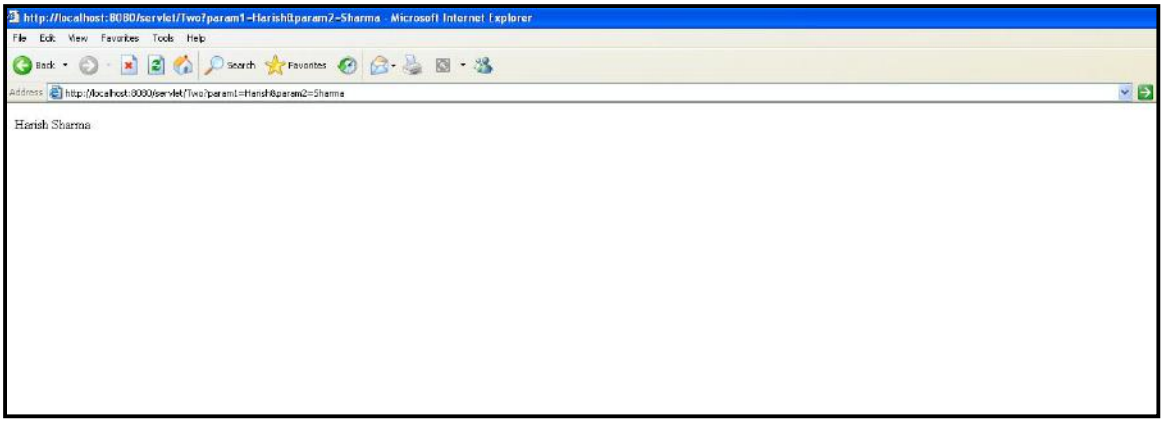


Fig 20.7 Output produced submiting above form

READING THREE PARAMETER USING POST METHOD

threepost.html

```
<html>
<head>
```

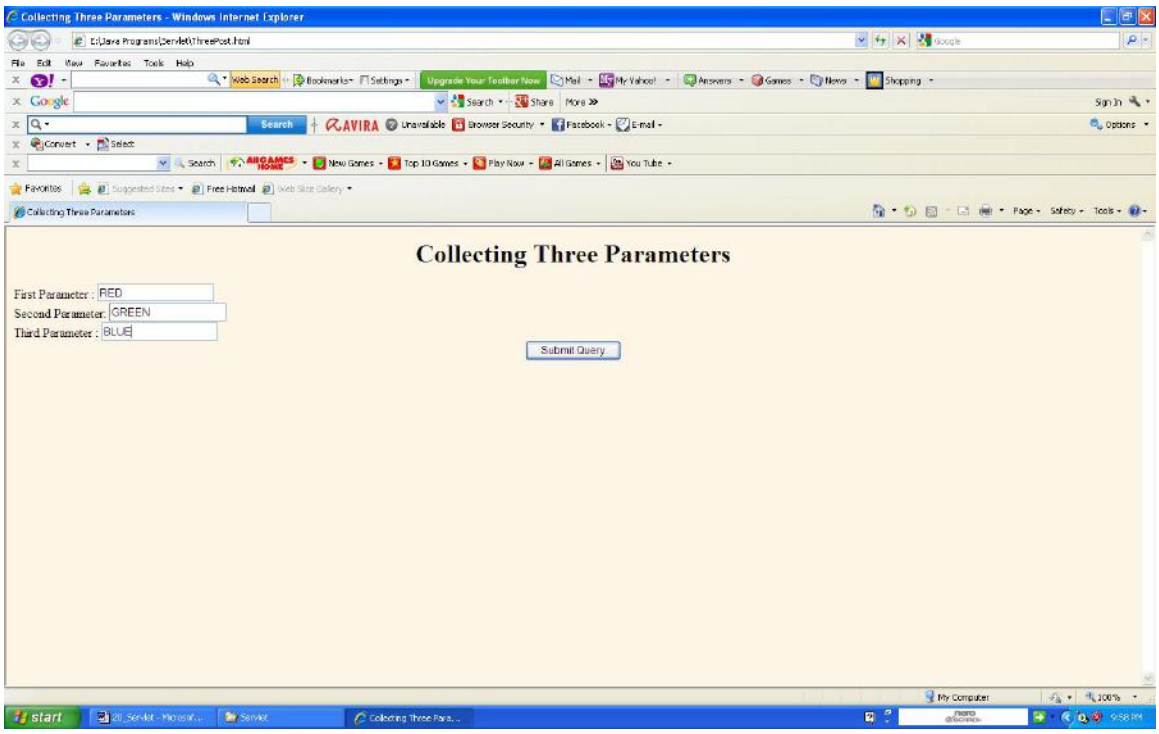
```
<title> Collecting Three Parameters </title>
<body bgcolor="#FDF5E6">
<h1 align ="center"> Collecting Three Parameters </h1>
<form action ="http://localhost:8080/ThreePost" method="post">

First Parameter : <input type = "text" name = "param1"><br>
Second Parameter: <input type = "text" name = "param2"><br>
Third Parameter :  <input type = "text" name = "param3"><br>
<center>  <input type = "submit"></center>
</form>
</body>
</html>
```

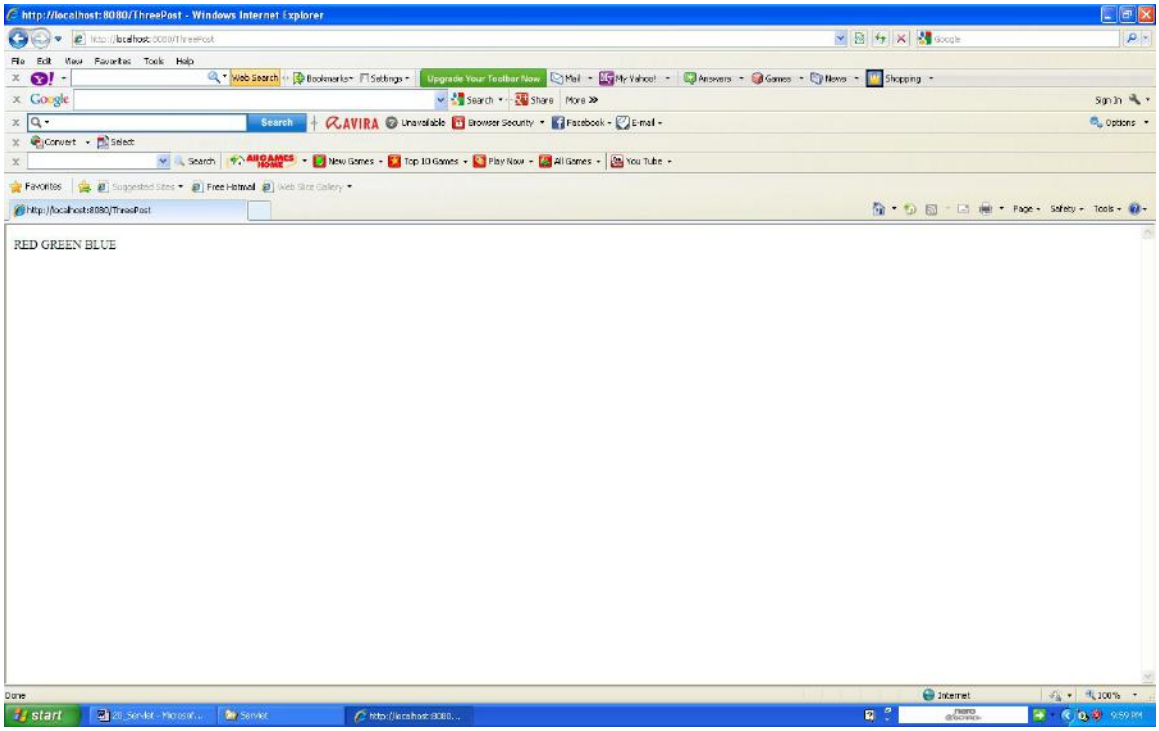
ThreePost.java

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

public class ThreePost extends HttpServlet
{
        public void doPost(HttpServletRequest req, HttpServletResponse res )
        throws ServletException, IOException
        {
          res.setContentType("text/html");
          PrintWriter pw = res.getWriter();
          pw.println(req.getParameter("param1"));
          pw.println(req.getParameter("param2"));
          pw.println(req.getParameter("param3"));
        }
}
```



Html form for collecting three parameter



Output of Post Method

COOKIES

Cookies are small text files that are used to maintain the different session when a user online for shopping different items from same web store. Cookies are send by the web server to the client browser. The browser later return unchanged when visit the same web page again. Once the web server receives cookie information the servlet program processes the cookie information and recognizes it as a new user or already visited user.

Cookies are therefore, a accepted method used by the web server for session tracking. Since the enable web servers to store and retrieve data from the client side. Cookies let a user to log in automatically. Cookies can be used to remember user preferennces.

CONTENTS OF A COOKIE

A cookie contains the following contents

- The name of the cookie
- The value of the cookie
- The expiration date of the cookie
- The valid path of cookie
- The domain of cookie
- The secure connection

PERMANENT COOKIES

Permanent cookies are stored in client machine, they are not deleted when browsing session is closed. Permanent cookies can be used to identifify individual users. The persist after the user restart the computer.

SESSION COOKIE

Session cookies are also called transient cookies. These cookies exists till the browsing session continues and automatically gets deleted as soon as the user closes the web browser. These cookies usually store a session ID that is not permanently bound to the user, allowing the user to move from page to page without login each time.

CREATING COOKIE

Cookies are created by the web server and stored in client computer. A Cookie can be created with the help of **Cookie** class which is then placed in HTTP response header with the help of method addCookie(cookie_name)

Cookie c = new Cookie("user", "1234");

Syntax: public Cookie (String <name>, String <value>);

Creates a new cookie with name and value. The following are some methods used with Cookie class.

1. public void setPath (String uri)

   It specifies the path for the cookie. A cookie path must be such that it includes servlet that creates the cookie. e.g. /servlet/cookies
2. public String getPath()

   This method returns path of the cookie
3. public void setMaxAge(int limit)

   This method is used to set life time of a cookie. The life time is specified in seconds a negative value indicate that cookie will expire when session closes.

   c.setMaxAge(60 * 60 * 24 * 365);

   This cookie will expire after one year

   c.setMaxAge(60 * 60 * 24 * 7);

   The cookie expires after one week
4. public int getMaxAge()

   This method returns a integer value which indicates the life of the existing cookie. A negative value indicates that cookie will be deleted as soon as the browsing session is closed.

   int x = c.getMaxAge();

   x contains the maximum age limit of the cookie **c**.
5. public void setComment(String str)

   The setComment(String str) used to set the comment field in the cookie

6. public String getComment()

Return the comment field from the cookie. The getComment return null value if c ookie has no comment

Following program creates a cookie and display its name and maximum life of the cookie.

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

public class DemoCookie extends HttpServlet
{

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

    {

        response.setContentType("text/html");
            PrintWriter pw = response.getWriter();
        Cookie c = new Cookie("DemoCookie","123456");
            /* Creates the cookie */
            c.setMaxAge(7*24*60*60);
            /* set life of cookie to one week */
            response.addCookie(c);
            /* add cookie to the HTTP response header */
            pw.println("<HTML><HEAD><TITLE>");
            pw.println("Demo Cookie");
            pw.println("</TITLE></HEAD><BODY>");
            pw.println("<H3>");
            pw.println("The Cookie name is : "+ c.getName());
```

```
            pw.println("<BR>");
            pw.println("The Value of Cookie : " + c.getValue());
            pw.println("<BR>");
            pw.println("Life of Cookie is  : " + c.getMaxAge() + " seconds");
            pw.println("<BR>");
            pw.println("</H3>");
            pw.println("</BODY></HTML>");
            pw.close();

        }

}
```

Program 20.4 DemoCookie.java

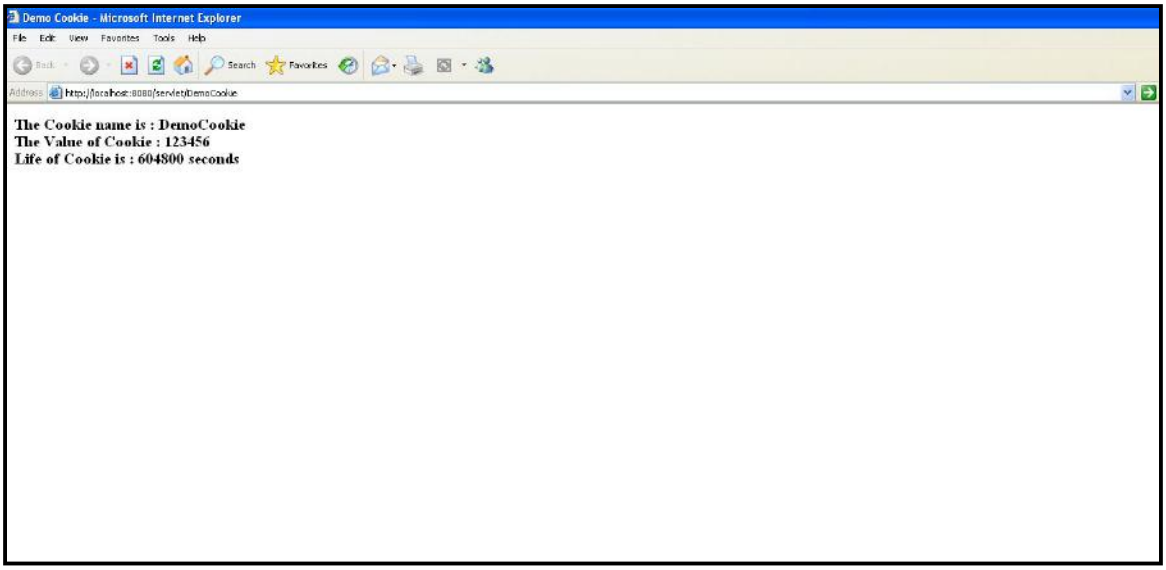


Fig. 20.8 Creating a Cookie

SEARCHING COOKIES

Cookies can be used to check whether the user is the first time visitor or he / she visiting again. To search a specific cookie, all the available cookies need to be checked to find the particular cookie using the getCookies() method of HttpServletRequest.

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

public class FindCookie extends HttpServlet
{
        public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
        {
                boolean find = true;
                Cookie [] cookies = request.getCookies();
                if(cookies != null )
                {
                        for(int i =0; i < cookies.length; i++ )
                        {
                                Cookie c = cookies[i];
                        if (( c.getName(). equals("DemoCookie")) && (c.getValue(). equals("123456")))
                        {
                                        find = false;
                                        break;
                        }
                }
        }
```

```
        String title;
        if (find)
        {
                Cookie ck = new Cookie("DemoCookie", "123456");
                ck.setMaxAge(60*60*24*7);
                response.addCookie(ck);
                title = "The New User ";
        }
        else
        {
          title = "The Old User ";
        }

        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<html>");
        pw.println("<h1>"+ title + "</h1>");
        pw.println("</html>");
  }
}
```

Program 20.5 Searching a specific cookie

The program 20.5 search for the DemoCookie in the computer hard disk if it does not find one then it creates a new cookie and displays the message The New User otherwise if it finds the DemoCookie then it display the message The Old User.

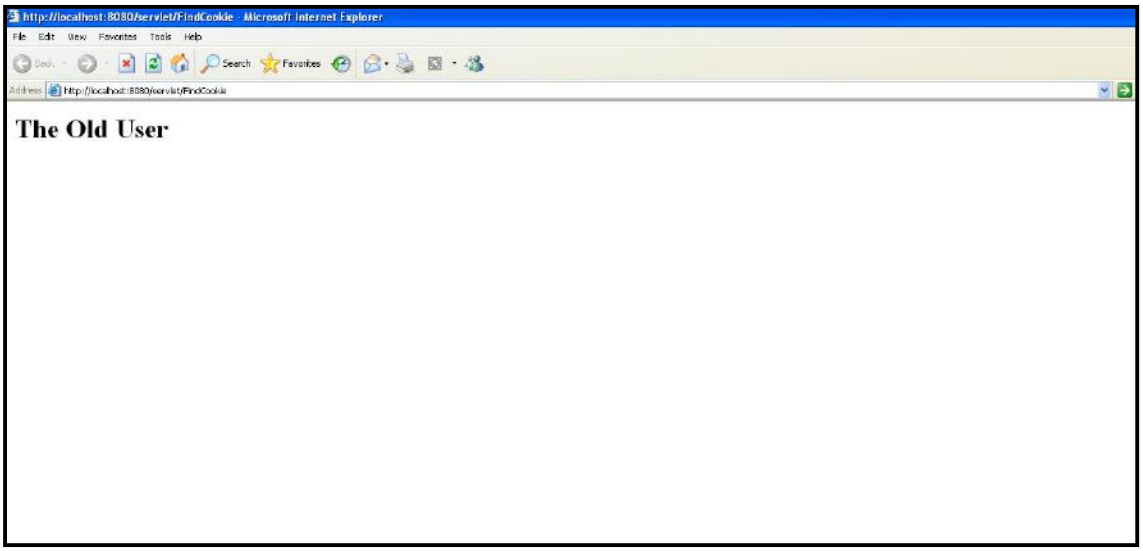


Fig 20. 9 Searching Specific Cookie

DELETING COOKIE

To delete cookies from local disk start Microsoft Internet Explorer select internet option from Tools menu and press Delete Cookies button of General Tab. A cookie can also be deleted with following step.

1. Set the maximum age of cookie to zero.
2. Assign null value to cookie

```
Cookie c = new Cookie("JavaCookie");
c.setMaxAge(0);
response.addCookie(c);
```

First create the cookie of same as to be deleted and set its age to zero and add it to the response header of the HTTP response.

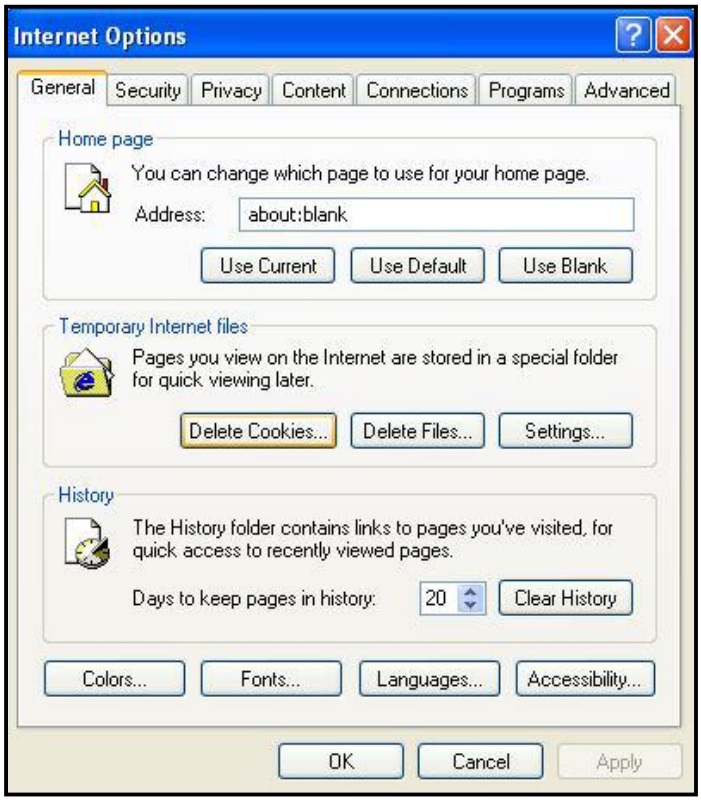


Fig 20.10 Deleting Cookies from Internet Explorer

SERVLET AND DATABASE CONNECTION

The following steps are needed to establish a database connection. The example in this section establishes a database connection with Microsoft Access Northwind database the sample database that comes with Microsoft Access. The example uses the HTML form which accept the SQL queries. The query is submitted to the servlet and servlet reads the input query and generates a output which displayed back web browser.

1. From the windows control panel ⟶ Administrative Tools
2. From Administrative Tools ⟶ ODBC Data Source
3. After double clicking ODBC the following figure comes up.

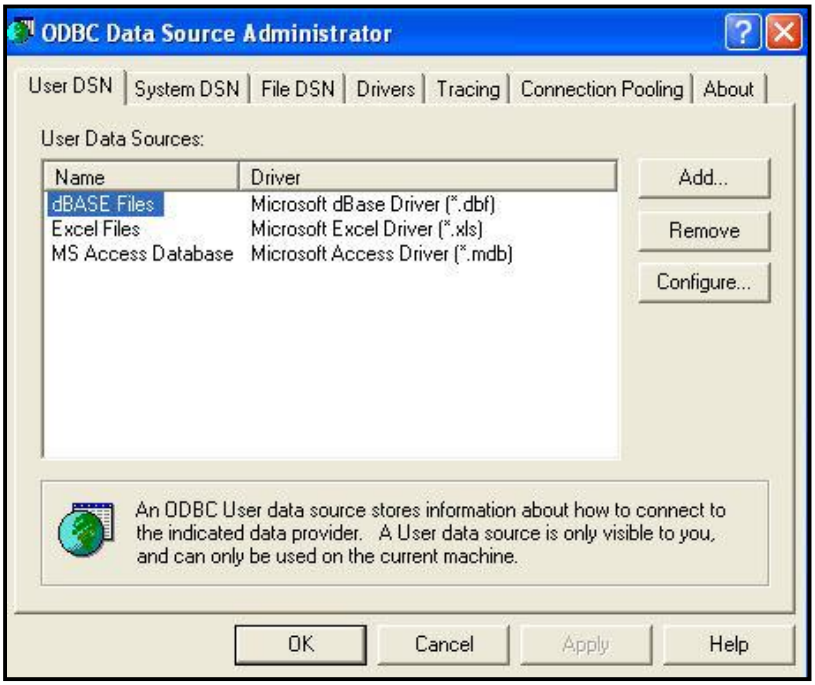


Fig 20.11 Creating Data Source

4. Select Add button and then select Microsoft Access Driver

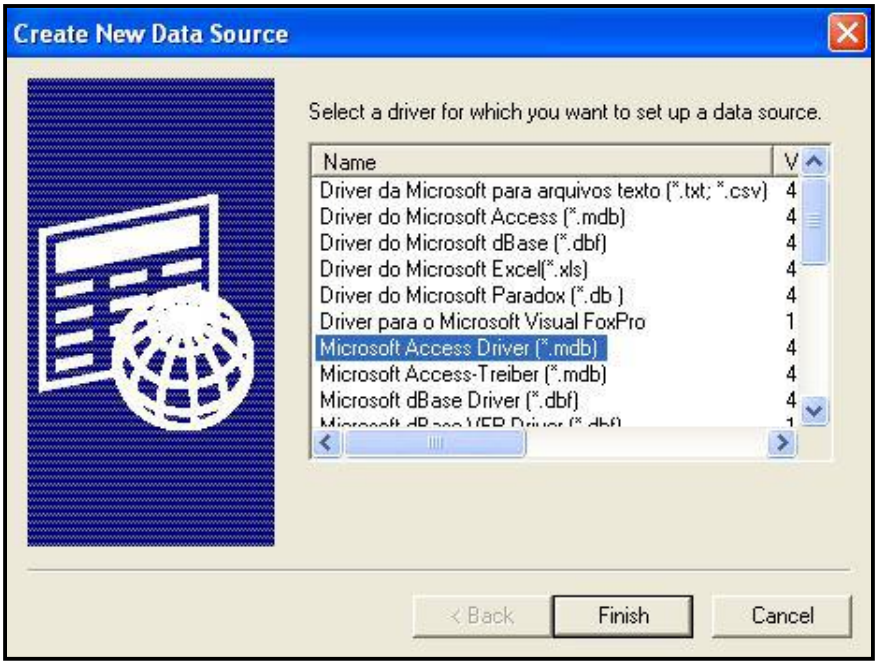


Fig 20.12 Select the Access Driver

The following HTML form is used to submit the query.

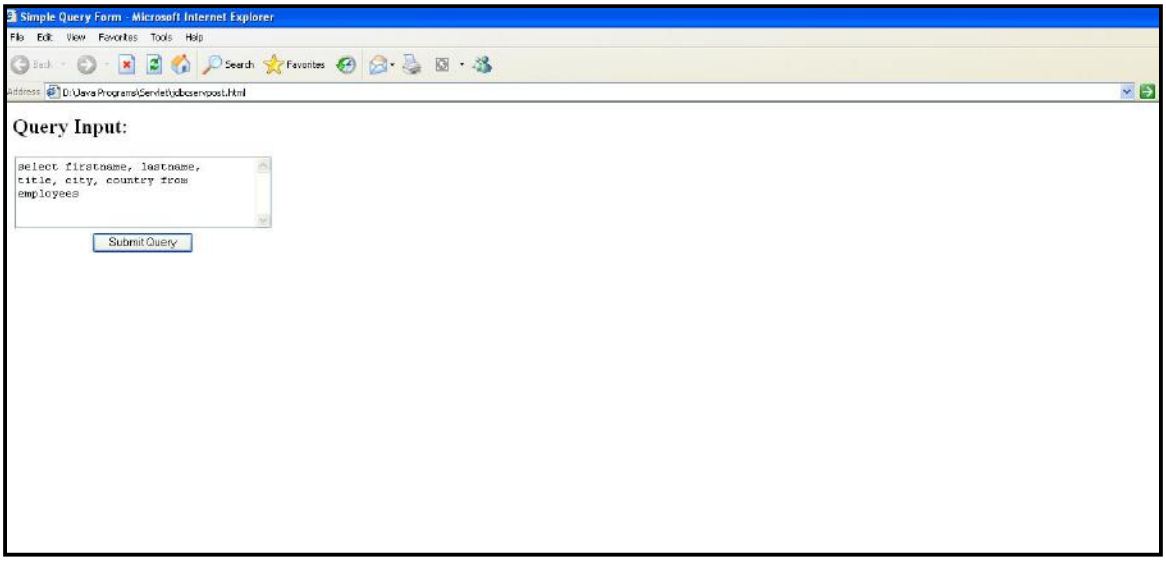
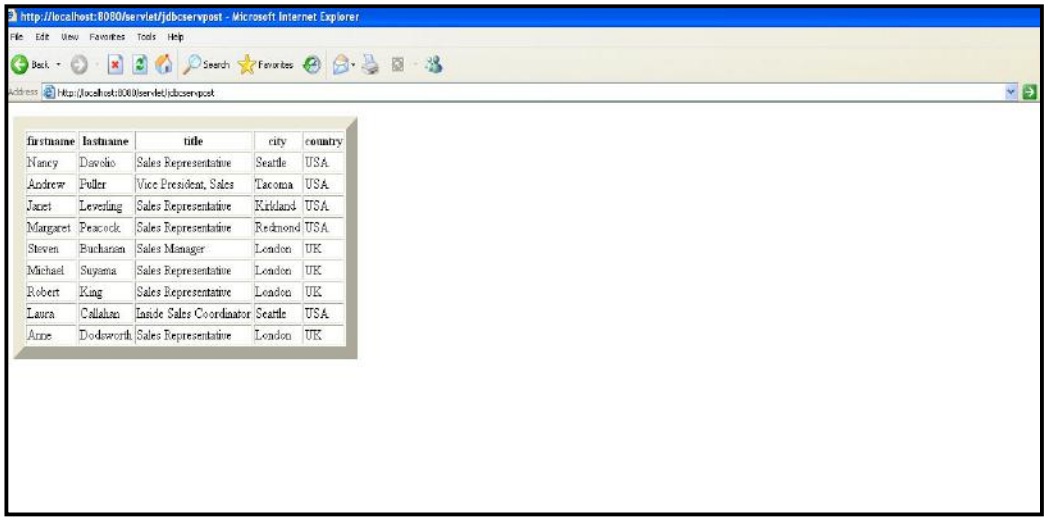


Fig 20.13 HTML form to read SQL Query

```
<html>
<head>
<title> Simple Query Form </title>
</head>
<body>
<h2> Query Input:</h2>
<form action ="http://localhost:8080/servlet/jdbcservpost"
method = post>
<table>
<td> <textarea rows="5" cols ="35" name = "query"></textarea>
<tr><td colspan="2" align = "center"><input type = "submit">
</table>
</form>
</body>
</html>
```

Program 20.6 The HTML cocde of the above SQL Query form

On clicking the submit button the java servlet produces the following output



firstname	lastname	title	city	country
Nancy	Davolio	Sales Representative	Seattle	USA
Andrew	Fuller	Vice President, Sales	Tacoma	USA
Janet	Leverling	Sales Representative	Kirkland	USA
Margaret	Peacock	Sales Representative	Redmond	USA
Steven	Buchanan	Sales Manager	London	UK
Michael	Suyama	Sales Representative	London	UK
Robert	King	Sales Representative	London	UK
Laura	Callahan	Inside Sales Coordinator	Seattle	USA
Anne	Dodsworth	Sales Representative	London	UK

Fig 20.14 Output of Database

The java servlet code for generating the above output.

jdbcservpost.java

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

public class jdbcservpost extends HttpServlet
{
        public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
        {

                response.setContentType("text/html");
                String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
                String url = "jdbc:odbc:TestDb";
```

```
try
{
        Class.forName(driver);
        Connection con = DriverManager.getConnection(url);
        Statement stmt = con.createStatement();
        String query = request.getParameter("query");
        ResultSet result = stmt.executeQuery(query);
        ResultSetMetaData resultSetMetaData= result.getMetaData();
        PrintWriter pw = response.getWriter();
        int columnCount = resultSetMetaData.getColumnCount();
        System.out.println(columnCount);
        pw.println("<TABLE BORDER=15>");
        pw.print("<TR>");
        for(int i=1; i<=columnCount; i++)
        {
                pw.print("<TH>"+resultSetMetaData.getColumnName(i));
        }
        pw.println();
        while(result.next())
        {
          pw.println("<TR>");
          for(int i=1; i<=columnCount;i++)
          {
                pw.print("<TD>"+result.getString(i));
          }
         }
        pw.println("</table>");
        con.close();
}
catch(ClassNotFoundException e)
{
```

```
                }
                catch(SQLException e1)
                {
                }
        }
}
```

Program 20.7 Servlet Code

Explanation

The above java overrides doPost method the Jdbc – Odbc dirver is defined by the string variable.

String driver = "sun.jdbc.odbc.JdbcOdbcDriver";

The data source is define as

String url = "jdbc:odbc:TestDb";

The Jdbc – Odbc driver is loaded by the statement

Class.forName(driver);

The ResultSetMetaData provides some extra information about the ResultSet object.

1. Number of column in the result set
2. Data type of the column
3. Name of the column
4. Is the column case sensitive

int columnCount = resultSetMetaData.getColumnCount(); prints number of columns of the input query.

pw.println("<TABLE BORDER=15>"); creates table

pw.print("<TH>"+resultSetMetaData.getColumnName(i)); prints the column name of each column which are inputed through the query.

pw.print("<TD>"+result.getString(i)); prints the actual data value form the table

pw.println("</table>"); closes the table

con.close(); closes the database connection

HTTP RESPONSE HEADER FROM SERVLET

Response headers can be used to specify cookies, to supply the page modification date to instruct browser to reload the page after a designated interval.

HttpServletResponse also supplies a number of convenience methods for specifying common headers.

SetContentType(String mime type)

COMMON MIME TYPES

Application / msword	Microsoft word
Application/pdf	acrobat pdf
Application /vnd.ms-excel	Excel spreadsheet
Image/gif	GIF image
Image/jpeg	jpeg image
Text/html	html document

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ApplesWord extends HttpServlet
{
        public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
        {
                response.setContentType("application/vnd.ms-excel");
                PrintWriter out = response.getWriter();
                out.println("This header indicates the number of bytes ");
                out.println("Apples\t78\t87\t92\t29\t=SUM(B2:E2)");
                out.println("Oranges\t77\t86\t93\t30\t=SUM(B3:E3)");
```

```
        }
}
```

ColorGetServlet.java

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


public class ColorGetServlet extends HttpServlet
{
        public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
        {
                String color = request.getParameter("color");
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<B> The selected color is :");
                pw.println(color);
                pw.close();
        }
}
```

color.html

```
<html>
<body>
<center>
<form name="Form1"
action = "http://localhost:8080/servlet/ColorGetServlet">
<B>: Color: </B>
```

```
<select name = "color" size="1">
  <option value ="Red"> RED </option>
  <option value ="green"> Green </option>
  <option value ="blue"> Blue </option>
</select>
<br><br>
<input type=submit value="submit">
</form>
</body>
</html>
```

servlet mapping for the above code in *web.xml* is as follows:

```
<servlet>
    <servlet-name>ColorGetServlet</servlet-name>
    <servlet-class>ColorGetServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ColorGetServlet</servlet-name>
    <url-pattern>/servlet/ColorGetServlet</url-pattern>
</servlet-mapping>
```

SERVLET DEBUGGING

Servlet run at the server side hence debugging a servlet is difficult task. The Servlet API gives you two ways of to deal with errors: you can manually send an error message back to the client or you can throw a ServletException.

1. Use print statements. Insert a print statement, restart the server, and see if the print statement is displayed in the standard output window.
2. Use the log file. The log contains the error messages. The HttpServlet class has a method log that lets you write information into a logging file on the server. The

log file is an option even when running on the remote server in such situation print statements are rarely useful.

3. Write separate classes. The methods that are frequently used put them in a separate class. Avoid repetition of a method.
4. Look at the HTML source. If the result you see in the browser look odd, then look att the the HTML code for possible error.
5. Look at the request data separately. Servlets read data from HTTP request, construct a response, and send send it back to the client. If something in the process goes worong, you want to discover if the cause is that the client is sending the wrong data or that the servlet is processing it in incorrectly. The EchoServer class let you submit HTML forms and get a result that shows you exactly how the data arrived at the server.
6. Stop and restart the server. Servers are supposed to keep servlets in memory between requests, not reload them each time they are executed. Howerver, most server support a development mode in which servlets are supposed to be automatically reloaded whenever their associated class file changes.

THREAD SAFE SERVLET

Threads are light weight process. When a servlet receives request from multiple clients then it creates a inconsistency in the servlet. Since there is only one copy of servlet in the memory and different clients are trying to update or modify servlet data structure. To overcome this problem a servlet needs to be a thread safe, i.e one client at a time can access to servlet and modify it. The second user has to wait till the first releases all its resources this can be achieved by surrounding section of code with synchronized blocks. While a particular synchronized block is executing, no other sections of code that are synchronized on the same object can execute.

Imagine if a servlet maintains a bank balance using an integer in memory. If two servlets try to access the balance at the same time we might get this sequence of events.

- User1 connects to the servlet to make a Rs. 5000 withdrawal
- The servlet checks the balance for user1 finding Rs. 10,000
- User2 connects to servlet to make as Rs.7000 withdrwal
- The servelt checks the balance for user2 finding Rs.10,000
- The servelt debit Rs.5000 for user1 leaving Rs.5000
- The servlet debit Rs.7000 for user2, leavinig Rs. –2000

The above problem can be solved by making a thread safe servlet.

HTTP

HTTP stands for Hypertext Transfer Protocol. HTTP is used in internet for accessing the world wide web. HTTP is used to access image, text or video over the internet A browser is works as an HTTP client because it sends requests to an HTTP server which is called Web server. The Web Server then sends responses back to the client. The standard and default port for HTTP servers to listen on is 80. HTTP uses client server technology.

- HTTP is connectionless

  HTTP disconnects from the server after it request and wait for the response.

- HTTP is Stateless

  The server and client forget each other after each request. The does not know whether the request is coming from the same client or from the different client.

- HTTP is independent of media

  HTTP is used to send any type of data it can be image file, text file, HTML file, video file or any other type file. The content type is handled by the MIME type.

The most methods of HTTP are GET and POST. The GET method reads information identified by request URI. The entire form submission can be encapsulated in one URL. While in POST method there's a block of data sent with the request, in the message body.

The whole file can sent with the help of POST method. There are usually extra headers to describe this message body, like Content-Type: and Content-Length

HTTP CODES

The response header field must be included in response messages. . The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the Request-URI.

1xx: Information

Message:	Description:
100 Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request
101 Switching Protocols	The server switches protocol

2xx: Successful

Message:	Description:
200 OK	The request is OK
201 Created	The request is complete, and a new resource is created
202 Accepted	The request is accepted for processing, but the processing is not complete
203 Non-authoritative Information	
204 No Content	
205 Reset Content	
206 Partial Content	

3xx: Redirection

Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that

	location. Maximum five addresses
301 Moved Permanently	The requested page has moved to a new url
302 Found	The requested page has moved temporarily to a new url
303 See Other	The requested page can be found under a different url
304 Not Modified	
305 Use Proxy	
306 *Unused*	This code was used in a previous version. It is no longer used, but the code is reserved
307 Temporary Redirect	The requested page has moved temporarily to a new url

HTTP SERVLET REQUEST INTERFACE

The interface HttpServletRequest encapsulates the functionality for a request object that is passed to an HTTP Servlet. It provides access to an input stream and so allows the servlet to read data from the client. The interface also provides methods for parsing the incoming HTTP FORM data and storing the individual data values - in particular *getParameterNames( )* returns the names of all the FORM's control/value pairs (request parameters). These control/value pairs are usually stored in an Enumeration object - such objects are often used for working with an ordered collection of objects.

Every call to doGet or doPost for an HTTPServlet receives an object that implements interface HTTPServletRequest. The web server that executes the servlet creates an HTTPRequest object and passes this to servlet's service method, hence this object contains the request from the client. A variety of methods are provided to enable the servlet to process the client's request. Some of these methods are listed below:

a) getCookies

public Cookie[] getCookies();

It returns an array containing all the cookies present in this request. Cookies can be used to uniquely identify clients to servlet. If there are no cookies in the request, then an empty array is returned.

b) getQueryString

public String getQueryString();
It returns query string present in the request URL if any. A query string is defined as any information following a ? character in the URL. If there is no query string, this method returns null.

c) getSession
public HttpSession getSession();

public HttpSession getSession(boolean create);

Returns the current valid session associated with this request. If this method is called with no arguments, a session will be created for the request if there is not already a session associated with the request. If this method is called with a Boolean argument, then the session will be created only if the argument is true.

To ensure the session is properly maintained, the servlet developer must call this method before the response is committed.

If the create flag is set to false and no session is associated with this request, then this method will return null.

HTTP SERVLET RESPONSE INTERFACE

The interface HttpServletResponse encapsulates the functionality for a response object that is returned to the client from an HTTP Servlet. It provides access to an output stream and so allows the servlet to send data to the client. A key method is getWriter ( ) which obtains a reference to a PrintWriter object for it is this PrintWriter object that is used to send the text of the HTML document to the client.

Every call to doGet or doPost for an HTTPServlet receives an object that implements interface HTTPServletResponse. The web server that executes the servlet creates an HTTPRequest object and passes this to servlet's service method, hence this object contains the response to the client. A variety of methods are provided to formulate the response to client. Some of these methods are listed below:

a) addCookie
public void addCookie(Cookie cookie);

It is used to add the specified cookie to the header of response. This method can be called multiple times to set more than one cookie. This method must be called before the response is committed so that the appropriate headers can be set. The cookies maximum age and whether the client allows Cookies to be saved determine whether or not Cookies will be stored on the client.

b) sendError

public void sendError(int statusCode) throws IOException;

public void sendError(int statusCode, String message) throws IOException;

It sends an error response to the client using the specified status code. If a message is provided to this method, it is emitted as the response body, otherwise the server should return a standard message body for the error code given. This is a convenience method that immediately commits the response. No further output should be made by the servlet after calling this method.

c) getWriter

public Printwriter getWriter()

It obtains a character-based output stream that enables text data to be sent to the client.

d) getOutputStream()

public ServletOutputStream getOutputStream()

It obtains a byte-based output stream that enables binary data to sent to the client.

e) sendRedirect

public void sendRedirect(String location) throws IOException;

It sends a temporary redirect response to the client (SC_MOVED_TEMPORARILY) using the specified location. The given location must be an absolute URL. Relative URLs are not permitted and throw an IllegalArgumentException.

This method must be called before the response is committed. This is a convenience method that immediately commits the response. No further output is be made by the servlet after calling this method.

REFERENCES

1. Core Servlets and JavaServer Pages, Second Edition, Marty Hall, Larry Brown,
2. The Java Complete Reference 5th Edition, Herbet Schildt, Tata McGraw Hill
3. Java Server Programming for Professionals, 2nd Edition, Ivan Bayross, Sharanam Shah, Cynthia Bayross, Vaishali Shah , Shroff Publisher