Experiment No. 10

Aim of the Experiment: Create a simple calculator application using servlet.

Objective:

To create a simple calculator application using servlet.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required, Apache tomcat 9.0 server

Course Outcome Addressed: CO6

Theory:

Servlets are small programs that execute on the server side. Servlets are pieces of Javasource code that add functionality to a web server.

Servlet provides full support for sessions, a way to keep track of a particular user over time as a website's pages are being viewed. They also can communicate directly with aweb server using a standard interface.

Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java classlibrary that supports large-scale development projects.

Running servlets requires a server that supports the technologies. Several web servers, each of which has its own installation, security and administration procedures, support Servlets. The most popular one is the Tomcat- an open source server developed by the Apache Software Foundation in cooperation with Sun Microsystems version 5.5 of Tomcat supports Java Servlet.

Getting Tomcat

The software is available a free download from Apache's website at the address http://jakarta.apache.org/tomcat. Several versions are available: Linux users should download the **rpm** of Tomcat.

The javax.servlet package

The important interfaces and classes are described in the table below.

Interface	Description
Servlet	A java servlet must implement the Servlet interface. This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods.

ServletConfig	The ServletConfig interface is used by the server to pass configuration information to a servlet. Its methods are used by the servlet to retrieve this information.
ServletRequest	The ServletRequest interface encapsulates a client request for service. It defines a number of methods for obtaining information about the server, requester, and request.
ServletResponse	The ServletResponse interface is used by a servlet to respond to a request by sending information back to the client.
ServletContext	The ServletContext interface defines the environment in which an applet is executed. It provides methods that are used by applets to access environment information.
SingleThreadModel	The SingleThreadModel interface is used to identify servlets that must be thread-safe. If a servlet implements this interface, the Web server will not concurrently execute the service() method of more than one instance of the servlet.
Class	**Description**
GenericServlet	The GenericServlet class implements the Servlet interface. You can subclass this class to define your own servlets.
ServletInputStream	The ServletInputStream class is used to access request information supplied by a Web client. An object of this class is returned by the getInputStream() method of the ServletRequest interface.
ServletOutputStream	The ServletOutputStream class is used to send response information to a Web client. An object of this class is returned by the getOutputStream() method of the ServletResponse interface.

The javax.servlet.http package

Interface	Description
HttpServletRequest	The HttpServletRequest interface extends the ServletRequest interface and adds methods for accessing the details of an HTTP request.
HttpServletResponse	The HttpServletResponse interface extends the ServletResponse interface and adds constants and methods for returning HTTP-specific responses.
HttpSession	This interface is implemented by servlets to enable them to support browser- server sessions that span multiple HTTP request-response pairs. Since HTTP is a stateless protocol, session state is maintained externally using client-side cookies or URL rewriting. This interface provides methods for reading and writing state values and managing sessions.
HttpSessionContext	This interface is used to represent a collection of HttpSession

	objects that areassociated with session IDs.
Class	**Description**
HttpServlet	Used to create HTTP servlets. The HttpServlet class extends the GenericServlet class.
Cookie	This class represents an HTTP cookie. Cookies are used to maintain session state over multiple HTTP requests. They are named data values that are created on the Web server and stored on individual browser clients. The Cookie class provides the method for getting and setting cookie values and attributes.

Servlet Life Cycle

A servlet's life cycle methods function similarly to the life cycle methods of applets.

- The **init(ServletConfig)** method is called automatically when a web server first begins a servlet to handle the user's request. The init() method is called only once. ServletConfig is an interface in the javax.servlet package, containing the methods to find out more about the environment in which a servlet is running.
- The servlet action is in the **service()** method. The service() method checks the HTTP request type (GET, POST, PUT, DELETE etc.) and calls doGet(), doPost(),doPut(), doDelete() etc. methods. A GET request results from normal request for a URL or from an HTML form that has no METHOD specified. The POST requestresults from an HTML form that specifically lists POST as the METHOD.
- The **destroy()** method is called when a web server takes a servlet offline.

Using Servlets

One of the main tasks of a servlet is to collect information from a web user and present something back in response. Collection of information is achieved using form, which is a group of text boxes, radio buttons, text areas, and other input fields on the web page. Each field on a form stores information that can be transmitted to a web server and then sent to a Java servlet. web browsers communicate with servers by using Hypertext Transfer Protocol (HTTP).

- Form data can be sent to a server using two kinds of HTTP requests: get and post. When web page calls a server using **get** or **post**, the name of the program that handles the request must be specified as a web address, also called uniform resource locator (URL). A get request affixes all data on a form to the end of a URL. A post request includes form data as a header and sent separately from the URL. This is generally preferred, and it's required when confidential information is being collected on the form.
- Java servlets handle both of these requests through methods inherited from the HTTPServlet class: **doGet(HttpServletRequest, HttpServletResponse)** and **doPost(HttpServletRequest, HttpServletResponse).** These methods throw two kinds of exceptions:

ServletException, part of javax.servlet package, and IOException, an exception in the java.io package.

- The **getparameter(String)** method is used to retrieve the fields in a servlet withthe name of the field as an argument. Using an HTML document a servlet communicates with the user.
- While preparing the response you have to define the kind of content the servlet issending to a browser. The **setContentType(String)** method is used to decide the type of response servlet is communicating. Most common form of response is written using an HTML as: **setContentType("text/html").**
- To send data to the browser, you create a servlet output stream associated with the browser and then call the **println(String)** method on that stream. The **getWriter()** method of HttpServletResponse object returns a stream. which can be used to send a response back to the client.

Example

```
import   java.io.*; import
javax.servlet.* ;
import javax.servlet.http.*;
public class MyHttpServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException, IOException
{
     // Use "req" to read incoming request
     // Use "res" to specify the HTTP response status
     //Use req.getParameter(String) or getParameterValues(String) to obtain
parameters
           PrintWriter out = res.getWriter();//stream for output
     // Use "out" to send content to browser
  }
}
```

Request and Response methods

ServletRequest methods	
String getParameter(String name)	Obtains the value of a parameter sent to the servlet as part of a get or post request. The name argument represents the parameter name.
Enumeration getParameterNames()	Returns the names of all the parameters sent to the servlet as part of a postrequest.
String[]getParameterValues(String name)	For a parameter with multiple values, this method Returns an array of strings containing the values for a specified servlet parameter.
String getProtocol()	Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.
String getRemoteAddr()	Returns the Internet Protocol (IP) address of the client that sent the request.
String getRemoteHost()	Returns the fully qualified name of the client that sent the request.
String getServerName()	Returns the host name of the server that received therequest.
int getServerPort()	Returns the port number on which this request was received.

HttpServletRequest methods	
Cookie[] getCookies()	Returns an array of Cookie objects stored on the client by the server.
HttpSession getSession(boolean create)	Returns an HttpSession object associated with the client's current browsing session. This method can create an HttpSession object (True argument) if one does not already exist for the client.
String getServletPath()	Returns the part of this request's URL that calls the servlet.
String getMethod()	Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
String getQueryString()	Returns the query string that is contained in the request URL after the path.
String getPathInfo()	Returns any extra path information associated with the URL the client sent when it made this request.
String getRemoteUser()	Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.

ServletResponse methods	
ServletOutputStream getOutputStream()	Obtains a byte-based output stream for sending binary data to the client.
PrintWriter getWriter()	Obtains a character-based output stream for sending text data (usually HTML formatted text) to the client.
void setContentType(String type)	Specifies the content type of the response to the browser. The content type is also known as MIME (Multipurpose Internet Mail Extension) type of the data. For examples, "text/html", "image/gif" etc.
String setContentLength(intlen)	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
HttpServletResponse methods	
void addCookie(Cookiecookie)	Used to add a Cookie to the header of the response to the client.
void sendError(int ec)	Sends an error response to the client using the specified status.
void sendError(int ec, String messg)	Sends an error response to the client using the specified status code and descriptive message.
void sendRedirect(Stirng url)	Sends a temporary redirect response to the client using the specified redirect location URL.
void setHeader(String name, String value)	Sets a response header with the given name and value.

Writing, Compiling and Running Servlet

Type the first sample program of the self-activity section. After saving this servlet,compile it with the Java compiler as: javac SimpleServlet.java. After compilation a class file with name SimpleServlet.class is created.

To make the servlet available, you have to publish this class file in a folder on your web server that has been designated for Java servlets. Tomcat provides the classes sub-folder to deploy this servlet's class file. Copy this class file in this classes sub-folder, which is available on the path: tomcat/webapps/ WEB-INF/classes. Now edit the web.xml file available under WEB-INF sub-folder with the following lines:

```
<servlet>
        <servlet-name>SimpleServlet</servlet-name>
        <servlet-class>SimpleServlet</servlet-class>

</servlet>
```

```
<servlet-mapping>

        <servlet-name>SimpleServlet</servlet-name>

        <url-pattern>/SimpleServlet</url-pattern>

</servlet-mapping>
```

Repeat the above sequence of line to run every newly created servlet. Remember, these line lines must be placed somewhere after the <web-app> tag and before the closing

</web-app> tag.

After adding these lines, save web.xml file. Restart the Tomcat service and run the servlet by loading its address with a web browser as: http://localhost:8080/FirstServlet.

Source Code:

Index.html

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Calculator Application Using Servlet</title>
</head>
<body>
<form method=get action="CalculatorServlet" >
Enter First Number <input type="text" name="t1"><br>
Enter Second Number <input type="text" name="t2" ><br>
Select an Operation<input type="radio" name="opr" value="+">
ADDTION <input type="radio" name="opr" value="-">
SUBSTRACTION <input type="radio" name="opr" value="*">
MULTIPLY <input type="radio" name="opr" value="/">
DIVIDE <br><input type="reset">
<input type="submit" value="Calculate" >
</form>
</body>
</html>
```

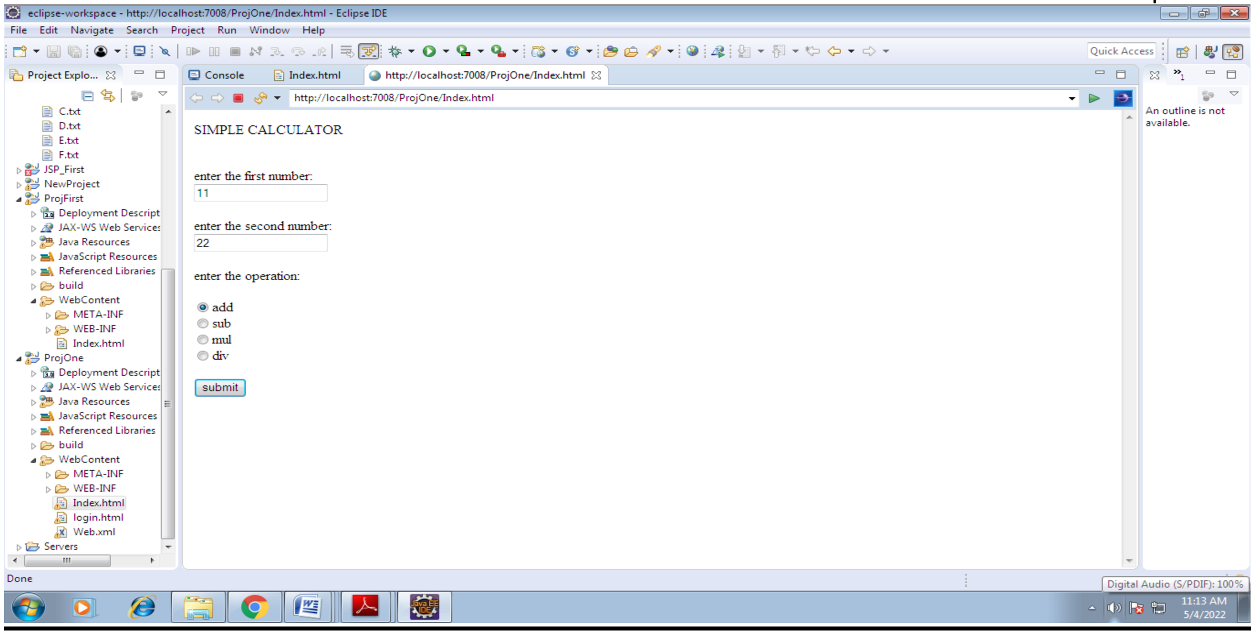
CalculatorServle.java

```
package AjpPr10;

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

public class CalculatorServlet extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException {
int result = 0;
try {
String number1 = req.getParameter("num1");
String number2 = req.getParameter("num2");
String operator = req.getParameter("op");
int x = Integer.parseInt(number1);
int y = Integer.parseInt(number2);
if(operator == "+") {
result = x + y;
}
else if(operator == "-") {
result = x - y;
}
else if(operator == "*") {
result = x * y;
}
else if(operator == "/") {
result = x/y;
}
PrintWriter p = res.getWriter();
p.println(result);
}
catch(Exception e) {}
}
}
```

OUTPUT:


eclipse-workspace - http://localhost:7008/ProjOne/Index.html - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explo...
Console
Index.html
http://localhost:7008/ProjOne/Index.html
An outline is not available.
Done
Digital Audio (S/PDIF): 100%
11:13 AM
5/4/2022

SIMPLE CALCULATOR

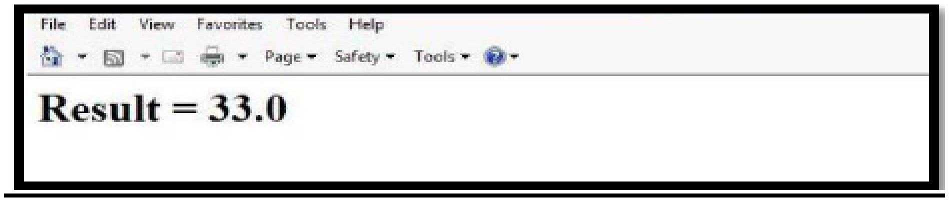
enter the first number: 11

enter the second number: 22

enter the operation:

- (●) add
- ( ) sub
- ( ) mul
- ( ) div

submit



Result = 33.0

Conclusion:

.

References:

Herbert Schildt , “Java : The Complete Reference” Tata McGraw-Hill (7th Edition).

Questions: What are Servlets?

1. What are the major tasks of servlets?
2. Explain servlet life cycle.
3. What is difference between Get and Post method?
4. What is HTTPServletRequest class?