1. i) How to Handle HTTP Request and Response in a Servlet? Explain with a example.

Ans:

The http servlet class provides speacialized methods that handle the various types of http requests. A servlet developer typically overrides one of these methods. Those methods are doDelete(), doGet(), doHead(), doOptions(), doPost(), doPut() and doTrace().

* Handling HTTP Get Requests here we will develop a servlet that handles a HTTP Get Request. The servlet is invoked when a form on web page is submitted.

The Example contains 2 files.

- A web Page is defined in ColorGetServlet.html and a servlet is defined in ColorGetServlet.java. The HTML source code for ColorGet.html is shown in following listing.
- It defines a form that contains a select element and a submit button.

```
<html>
  <body>
    <center>
      <form name = "form1" action="http://localhost:8080/
                examples/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>
```

* The servlet code for colorGetServlet.java is in listing.
It uses getParameter() method of HttpServlet Request to obtain this selection that was made by user code:

```
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();
  }
}.
```

ii) Construct servlet Program to display a message "Welcome to World" in webPage.

Servlets are java classes which service Http request.

code:-

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    private String message;
    public void init() throws ServletExpeption {
        message = "Hello World";
    }
    public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
{ response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<h1>" + message + "</h1>");
}
public void destroy()
    {
    }
}
```

output:-

http://localhost:8080/HelloWorld

Hello World

CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

9 b) How to access a database from a JSP? Explain with suitable example

Access a database from a JSP typically involves the following step

1. Load the JDBC driver
2. Establish connection to database
3. create SQL query.
4. Execute query and process results
5. Close the connection

Assumptions :

- TOMCAT webserver is installed
- mySQL server is installed
- JDK is installed.

STEPS TO ACCESS DATABASE :-

1. Create Database named "books" statement is :-

   mySQL > Create Database books ;

2. Create a table named "book_details" statement is :-

   mySQL > CREATE TABLE books_details ('id' INT (11) NOT NULL AUTO_INCREMENT, 'book_name' VARCHAR (100) NOT NULL, 'author' VARCHAR (100) NOT NULL, PRIMARY KEY ('id')

)

Table:

ID	BOOK_NAME	AUTHOR
1.	Java I/O	Jim Ritch
2.	Java & XML, 2 Edition	Brett McLaughlin
3	Java Swing	Dave Wood.

iii) copy the Mysql-connector-java-3.1.14-bin.jar to the C:\tomcat-directory\common\lib.

Then set classpath. For that, we have to do following Control Panel → System → System properties. → Environment variables and set classpath

iv) After setting classpath, restart the tomcat web server using the startup.bat file at the Cmd.

Create folder DBDemo using following line:

C:\tomcat-directory\jsp-examples\DBDemo

and now write JSP program which is used to connect the database.

Example

```
<%@ page language="java" import="java.sql.*" %>
<%
    Connection con=null;
    ResultSet: Statement stmnt=null;
String drives="org.gjt.mm.mysql.Drives";
Class.forName(drives).newInstance();
try { String url="jdbc:mysql://localhost/books?
user=<user>&password=<password>";
con = DrivesManager.getConnection(url);
stmt = con.createStatement();
}
catch (Exception e) { system.out.println(e.getMessage());
}.
if (request.getParameter("action")!=null)
{
```

```
String book1 = request.getParameter("bookname");
String author = request.getParameter("author");
stmt.executeUpdate("insert into books_details(book_name, author)
values('"+bookname+"','"+author+"');");
rst = stmt.executeQuery("select * form books_details");
%>
while (rst.next())
<%>
    <tr> <td> <%= no%> </td>
    <td> <%= rst.getString("book_name") %> </td>
    <td> <%= rst.getString("author") %> </td>
  </tr>
  <%no++;
}
rst.close()
stmt.close()
con.close()
```

ii) List out all implicit objects of JSP.

→ There are 9 JSP implicit objects. These objects are created by the web container that are available to all jsp pages.

→ The available implicit objects are out, request, config, session, application.

① **Response :-**

Response object is an instance of javax.servlet.http.HttpServlet Response object.

Just as server creates request obj, it also creates to represent

an obj for response to client

Eg:- response.sendtodirect("http://www.google.com");

② config:

→ In JSP config is implicit object of type servlet config

→ This obj can be used to get initialization parameter for a particular JSP Page.

→ The config object is created by web container for each jsp page.

Eg: String drivers = config.getInitParameter("dname");

③ out :-

→ The out implicit obj is instance of javax.servlet JSP writer object and is used to send content in responce.

Eg: out.println("welcome");

④ Request :-

→ The request object is instance of javax.servlet.http.HttpServletRequest object

Ex: String name = request.getparameter("uname");

⑤ Application :-

↳ The instance of servlet context is created only once by the web container.

String drivers = application.getInitparameter("dname");

⑥ Session :-

→ The obj can be used to get initialization parameter

→ The config obj is created by the web container for each jsp

Eg:

String driver = config.getInitparameter("dname").

⑦ Page Content:-

The JSP Page content can be used to set get/remove attribute from

- Page
- request
- session
- app

Ex.

PageContext.removeAttribute("attrName", PAGE_SCOPE)

⑧ Page:

→ It can be thought of an object to represent entire JSP page

→ The page obj is really direct synonym for this object

⑨ Exception:-

→ In JSP Exception of type java.lang.Throwable class.

→ It is better to learn it after page direction

(3) i) Explain JavaBeans with Suitable examples?

* The JavaBean is specially constructed Java class and coded to JavaBeans API specification.
* It provides a default, no-argument constructor
* It may have no. of "getter" & "setter" methods for properties.

JavaBeans Properties:-

* A JavaBean property is named attribute accessed by the user, can be of any datatype.

1. getPropertyName():- if property name is firstName your method name would be getFirstName(). This is accessor.
2. setPropertyName():- if property name is firstName,

your method name would be setFirstName().
This is called mutator.

Example :-

```
Public class StudentBean implement Serializable.
{ String Rno;
  String Name;
  public void setRno(String rno)
  { this.Rno = rno; }
  Public void getRno (String)
  { return Rno; }
  public void setName (String name)
  { this.Name = name; }
  Public void getName()
  { return Name; }
}.
```

* There are various scopes using which bean is in JSP.

1) Page Scope :--
   It is default page scope for a bean in jsp

2) Request Scope :-
   the obj remains in existance as long as req. obj is present.

3) Session scope :
   A session is specific period of time

4) Application Scope.
   Broadest scope provided by jsp.
   It gets stored in servletContext.

ii) Explain cookies & session handling in jsp with suitable examples

↳ cookies are usually set in HTTP header.

↳ Various methods are used in handling cookies are

i) create cookies

ii) Read cookies

iii) Delete cookies

i) CREATE COOKIE :-

STEP 1 :- In JSP using constructor cookie.

```
Cookie cookie = new Cookie("name","value");
```

STEP 2 To set validity period. for example to set the cookie alive for 24 hr we will write code as

```
cookie.setMaxAge(60*60*24);
```

STEP 3 Now our cookie is ready to send over. we can add cookie in HTTP response header as follow

② READ COOKIE

STEP 1 :- First cookie is retrieved using getCookies() method

```
Cookie[] cookies = request.getCookies();
```

STEP 2 :- then using getName() & getValue() methods the cookies are read

③ DELETE COOKIE

STEP 1 :- Read the already created Cookie and store it in object

```
Cookie cookie = new Cookie("name","");
```

STEP 2 :- then set its period of existence as 0.

```
cookie.setMaxAge(0);
```

STEP 3 :- Add this cookie back to response.

```
response.addCookie(cookie);
```

CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

Session Handling in JSP

* If we use a request scope and try to access the data over multiple pages, then same data can be shared over multiple pages. But sometimes we need to use same data for multiple requests. For Example in Hospital Management system, the patient info is entered initially only. that patient may undergo through various tests or operations.
In such a case the session scope
* HTTP is a request-response protocol.
* But at the same time its also called stateless protocol. That means when browser sends a request to server, server processes it and sends response & doesn't remember about this request. To solve this there are methods used :-
is Use of cookies
ii) Embedded hidden fields in HTML form
iii) sending URL string in response body.

Session-ID :- It is used to send info. to and fro b/w server & browser.

Session-tracking :- keep a track of all info b/w server & browser

(4) Illustrate the following with reference to
a) Functions
b) Form validation.

Ans (a) Functions :-
A function consists of function keyword followed

by the name of function and set of open and close paranthesis enclosing an optional.

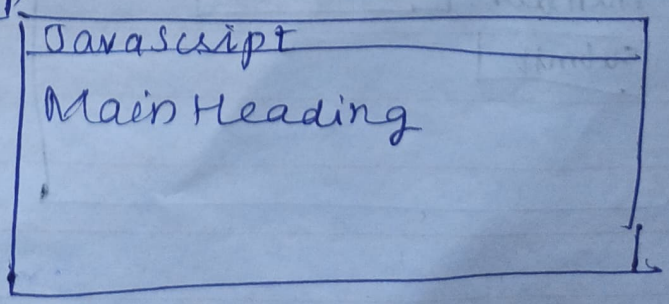
syntax:

```
function functionname (parameterlist)
{  //body
}
```

* A function uses return keyword to return a value from function.

```
<html>
  <head>
    <title> Javascript </title>
    <script type = "text/javascript" >
    function getHeader()
    {   return "<h1> Main Heading </h1>"
    }
    </script>
  </head>
 <body>
    <script type = "text/javascript" >
        document.write(getHeader());
    </script>
 </body>
</html>
```

O/P



CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

⑥ Form validation.

→

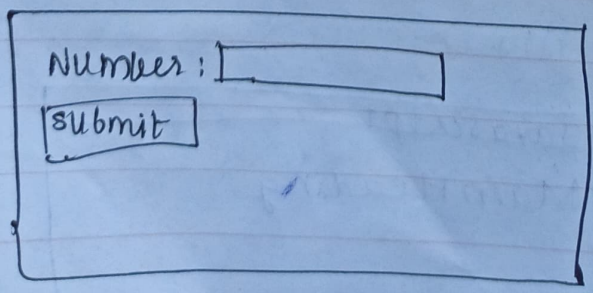
Example

Javascript Number validation

code :-

```
<script>
function validate() {
var num = document.myform.num.val
if (isNaN(num)) {
        document.getElementById("numloc");
        return false
  }
  else
        return true
}
</script>
<form name="myform" onsubmit="return validate()
  Number: <input type="text" name="n"
   <span id="numloc"> </span> </br>
  <input type="submit" value="submit">
</form>
```

O/P



(5) i) Discuss about scope of variable in Javascript

Ans:- The scope of variable determines where variable is accessible within code. We have 3 types of scope in JavaScript.

1. Global Scope :-
   - → variable declared outside of function.
   - → declared using 'var', 'let', or 'const'

```
Ex:- var globalVar = "Im global";
     function showGlobal()
     {   console.log(globalVar);
     }
     showGlobal();
```

2. Function scope.

```
Ex: function showLocal() {
      var localVar = "Im local";
      console.log(localVar);
    }
    showLocal();
```

3) Block scope :-

```
Ex:- If (true) {
       let blockLet = "Im - BlockScoped"
       console.log(blockLet)   // (Im BlockScoped)
     }
     console.log(blockLet)   //(Error)
```

(ii) How Event Handling takes place, explain with Example.

* Event represents change in enviroment.
* For Ex, mouse clicks, such events are intrinsic events

* EventHandler is that gets executed in response to these events. Thus event handler enables web document to respond the user activities through browser window.

* The process of connecting event handler to an event is event registration. Then event handler registration can be done using two methods.
  - Assigning tag attributes
  - Assigning handler address to object properties

Commonly used events & tag attributes:

Event	Intrinsic Event	Meaning	Tag
change	onChange	on occurance of some change	<input> <textarea> <select>
click	onclick	when user clicks the mouse button	<a> <input>
load	onload	when after getting doc loaded	<body>