Q.1 a. What are technologies we use in DHTML? Explain their significance in DHTML application?

Answer:

Speaking in true sense there is nothing dynamic in DHTML but enclosing technologies such as CSS, JavaScript, DOM and the static markup language it becomes dynamic.

JavaScript: Whether we call it JavaScript, Jscript, or ECMAScript, it is the most common language used today for client-side scripting. The main reason for this JavaScript comes with virtually every browser.For example, an onload event could execute a JavaScript function to query the browser's cookies collection to determine whether the user is a first-time visitor to the page.

CSS: It stands for Cascading Style Sheet. This is used for the presentation part of the web page. In simple words it holds the designing of the page. The look & feel of the page completely depends on CSS. In DHTML CSS rules can be modified at both the document and the element level using JavaScript with event handlers, they can add a significant amount of dynamism with very little code.

DOM: It stands for Dynamic Object Model and it is the weakest link in DHTML as many of the browser does not support the DOM functionality. It defines the object and its properties. It is a standard way of accessing and manipulating the static content. The Document Object Model is a platform and language-neutral interface that allows program and scripts to dynamically access the content and update it.

b. Define a socket? How read and write is performed using sockets?

Answer:

An application program inter face specifies the details of how an application program interacts with protocol software. Socket API is a de-facto standard. Once a socket has been established the application can transfer information.
recv( ) and send( ) are used to read and wri te the data.
recv(socket , buffer, length, flags)

The socket is the descriptor of the socket , buffer specifies the address in memory where incoming message should be placed and length specifies the size of the buffer , flags al lows the caller to control details.
send(socket , data, length, f lags)
Here data is the address of data to be sent and other arguments are same.
Sockets also al lows read( ) and wri te( ) to t ransfer data l ike send ( ) and recv( ).
read( ) and wri te() have three arguments: a socket descriptor, the locat ion of the buffer in the memory and the length of the memory buffer.

c. What is trivial file transfer protocol? Explain briefly.

Answer:

Trivial File Transfer Protocol (TFTP) is useful for bootstrapping a hardware device that does not have a disk on which to store system software. Al l the device needs is a network connect ion and a small amount of read only memory (ROM) into which TFTP, UDP and IP are hardwired. Al though TFTP is less powerful than FTP. TFTP does have two advantages. First , TFTP can be used in environments where UDP is available, but TCP is not . Second the code for TFTP requires less memory than the code for FTP.

d. Define DTD (Document Type Definition). Explain DTD and schema.

Answer:

DTD - Document Type Definition defines the legal building blocks of an XML document.

It defines the document structure with a list of legal elements and attributes.

XML DTD is a rule book that an XML document follows. Once DTD is ready, you can create number of XML documents following the same rules specified in the DTD. DTD can be internal or external DTD. The internal DTD is included in the XML document, while external DTD exists outside the content of the documents.

A DTD provides a list of the elements, attributes, comments, notes, and entities contained in an XML or HTML document and indicates their relationship with each other.

The 'DOCTYPE' tells the browser that it is a Document Type Declaration

Some commonly used attribute types are

- CDATA The value is character data
- ID The value is a unique id
- IDREF The value is the id of another element
- IDREFS The value is a list of other ids

Schema means the organization and the structure of a database.

E.g.: An XML schema is a description of XML document. It is expressed in terms of constraints on the structure and content of documents.

e. Which are the two methods used for cross domain Ajax calls? What is JSON in Ajax?

Answer:

There are two methods used to transfer data between the two more more security domains:

- CROS – Cross Origin Resource Sharing and it works with the HTTP web browsers
- JSONP – JSON with Padding which works with the HTTP GET and on legacy browsers JSON is abbreviated as JavaScript Object Notation.

JSON is a safe and reliable data interchange format in JavaScript, which is easy to understand for both users and machines.

f. What is the role of the JMS provider? What are the components of JMS?

Answer:

The JMS provider handles data conversion, security of the messages and the client triggering. It specifies the level of encryption, security level of the message and the best-data type for the non-JMS client.

JMS provider

- JMS client
- Messages
- Administered objects
- Native clients

g. Differentiate 'Stateful Session' from 'Entity Bean'? **(7×4)**

Answer:

While both undergo activation and passivation; EB have ejbStore () callback to save state through passivation and ejbLoad () callback to load state through activation. But in case of SS, this is not needed because S.S.B fields are serialized through objects by containers.

Q.2 a. Write a UDP client server program using Java. **(9)**

Answer:

UDP server

```
import java.io.*;
import java.net.*;

class UDPServer
{
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while(true)
          {
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String( receivePacket.getData());
            System.out.println("RECEIVED: " + sentence);
            InetAddress IPAddress = receivePacket.getAddress();
```

```
                    int port = receivePacket.getPort();
                    String capitalizedSentence = sentence.toUpperCase();
                    sendData = capitalizedSentence.getBytes();
                    DatagramPacket sendPacket =
                    new DatagramPacket(sendData, sendData.length, IPAddress, port);
                    serverSocket.send(sendPacket);
                  }
            }
        }
```

UD[P client

```
import java.io.*;
        import java.net.*;

        class UDPClient
        {
          public static void main(String args[]) throws Exception
          {
            BufferedReader inFromUser =
              new BufferedReader(new InputStreamReader(System.in));
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress IPAddress = InetAddress.getByName("localhost");
            byte[] sendData = new byte[1024];
            byte[] receiveData = new byte[1024];
            String sentence = inFromUser.readLine();
            sendData = sentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
        sendData.length, IPAddress, 9876);
            clientSocket.send(sendPacket);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
        receiveData.length);
            clientSocket.receive(receivePacket);
            String modifiedSentence = new String(receivePacket.getData());
            System.out.println("FROM SERVER:" + modifiedSentence);
            clientSocket.close();
          }
        }
```

b. Write a java program for Session tracking a hit count. (9)

Answer:

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

public class SessionTracker extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    // Get the current session object, create one if necessary
    HttpSession session = req.getSession();

    // Increment the hit count for this page. The value is saved
    // in this client's session under the name "tracker.count".
    Integer count = (Integer)session.getAttribute("tracker.count");
    if (count == null)
      count = new Integer(1);
    else
      count = new Integer(count.intValue() + 1);
    session.setAttribute("tracker.count", count);

    out.println("<HTML><HEAD><TITLE>SessionTracker</TITLE></HEAD>");
    out.println("<BODY><H1>Session Tracking Demo</H1>");

    // Display the hit count for this page
    out.println("You've visited this page " + count +
      ((count.intValue() == 1) ? " time." : " times."));

    out.println("<P>");

    out.println("<H2>Here is your session data:</H2>");
    Enumeration enum = session.getAttributeNames();
    while (enum.hasMoreElements()) {
      String name = (String) enum.nextElement();
      out.println(name + ": " + session.getAttribute(name) + "<BR>");
    }
    out.println("</BODY></HTML>");
  }
}
```

Q.3 a. Write a simple XML file you could use for a slide presentation. (9)

Answer:

```
<?xml version='1.0' encoding='us-ascii'?>

<!--  A SAMPLE set of slides  -->
```

```
<slideshow
   title="Sample Slide Show"
   date="Date of publication"
   author="Yours Truly"
   >

   <!-- TITLE SLIDE -->
   <slide type="all">
     <title>Wake up to WonderWidgets!</title>
   </slide>

   <!-- OVERVIEW -->
   <slide type="all">
      <title>Overview</title>
      <item>Why <em>WonderWidgets</em> are great</item>
      <item/>
      <item>Who <em>buys</em> WonderWidgets</item>
   </slide>

</slideshow>
```

b. What is XSL? Define CSS and XSL. How do you display XML with XSLT? (9)

Answer:
XSL is a language for expressing style sheets. An XSL style sheet is a file that describes the way to display an XML document.
Using XSL stylesheets, we can separate the XML document content and its styling.
An XSL style sheet begins with the XML declaration:
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet> defines that the document is an XSLT style sheet document.
The <xsl:template> element defines a template.

XSL is a language for expressing style sheets. An XSL style sheet is a file that describes the way to display an XML document.
Cascading Style Sheets is an answer to the limitations of HTML, where the structure of documents was defined and not the display. CSS formats documents for display in browsers that support it.

First you need to declare the XSL style sheet:
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
Then,
you create an XSL Style Sheet with a transformation template.
Add the XSL style sheet reference to your XML document to link them

Q.4 a. What are the advantages and disadvantages of Ajax? (9)

Answer:

Following are the advantages of Ajax:

- Bandwidth utilization – It saves memory when the data is fetched from the same page.
- More interactive
- Speeder retrieval of data

Disadvantages of Ajax:

1. AJAX is dependent on Javascript. If there is some Javascript problem with the browser or in the OS, Ajax will not support
2. Ajax can be problematic in Search engines as it uses Javascript for most of its parts.
3. Source code written in AJAX is easily human readable. There will be some security issues in Ajax.
4. Debugging is difficult
5. Increases size of the requests
6. Slow and unreliable network connection.
7. Problem with browser back button when using AJAX enabled pages

b. Write an Ajax code to retrieve header information. (9)

Answer:

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc(url)
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById('p1').innerHTML=xmlhttp.getAllResponseHeaders();
    }
  }
xmlhttp.open("GET",url,true);
xmlhttp.send();
}
</script>
</head>
<body>
<p id="p1">The getAllResponseHeaders() function returns the header information of a
        resource, like length, server-type, content-type, last-modified, etc.</p>
<button onclick="loadXMLDoc('ajax_info.txt')">Get header information</button>
</body>
</html>
```

Q.5 a. What are the different types of EJB? (9)

Answer:

Entity Bean: it represents an entity which is mapped with database or we can say it makes OR object Relational mapping with Database. Entity bean typically represent table in RDBMS and each instance represent row in the table.

Two types of entity bean:

- **CMP Entity bean**: Container managed entity bean its responsibility of container to manage the bean persistence behavior.
- **BMP Entity bean**: Programmer manage the bean persistence behavior.

Session bean: Session bean is responsible for developing business logic it makes the client server relationship so session beans exist till there is a session exist between client and server, it doesn’t contain persistent business concept.

Types of session bean

Stateless session bean: when there is not need to maintain state of a particular client stateless session bean is used .They alive for short period of time.

For example if we are validating the credit card we can use stateless session bean.

Stateful session bean: stateful session bean maintain the conversational state of client over the series of method call before the bean instance goes to passive state conversational state is saved to persistence area like Hard disk and again when same client send a request and bean instance come into the active state it will come out from hard disk to main memory.

For Example when we do online banking transaction ,online reservation we use stateful session bean

Message Driven Beans: these beans are work as a listener for messaging services like JMS .

b. What are the main classes which are used in struts application? What is the difference between DispatchAction and LookupDispatchAction in Struts Framework? **(9)**

Answer:

This is another *beginner's level Struts interview question* which is used to check how familiar candidate is with Struts framework and API. Main classes in Struts Framework are:

Action servlet: it's a back-bone of web application it's a controller class responsible for handling the entire request.

Action class: using Action classes all the business logic is developed us call model of the application also.

Action Form: it's a java bean which represents our forms and associated with action mapping. And it also maintains the session state its object is automatically populated on the server side with data entered from a form on the client side.

Action Mapping: using this class we do the mapping between object and Action.

ActionForward: this class in Struts is used to forward the result from controller to destination.

Dispatch Action	**LookupDispatchAction**
It's a parent class of LookupDispatchAction	Subclass of Dispatch Action
DispatchAction provides a mechanism for grouping a set of related functions into a single action, thus eliminating the need to create separate actions for each function.	An abstract **Action** that dispatches to the subclass mapped executes method. This is useful in cases where an HTML form has multiple submit buttons with the same name. The button name is specified by the parameter property of the corresponding ActionMapping.
If not using Internalization functionality then dispatch action is more useful.	Lookup Dispatch Action is useful when we are using Internalization functionality
DispatchAction selects the method to execute depending on the request parameter value which is configured in the xml file.	**LookupDispatchAction** looks into the resource bundle file and find out the corresponding key name. We can map this key name to a method name by overriding the getKeyMethodMap() method.

DispatchAction is not useful for I18N	LookupDispatchAction is used for I18N

Q.6 a. Explain the Role of the JMS Provider? What is the difference between JMS and RPC? Explain the use of Message object. (9)

Answer:

JMS Provider is an implementation of the JMS interface for a Message Oriented Middleware (MOM). Providers are implemented as either a Java JMS implementation or an adapter to a non-Java MOM.
Their functions are:
- Handling security of messages
- Data Conversion
- Client triggering
- Specifying encryption level, security level, best data type for JMS Client.

Answer

The basic difference between RPC and JMS lies in the way they message. RPC uses synchronous messaging while JMS uses asynchronous messaging approach. In RPC the method invoker waits for the method to finish execution and return the control back to the invoker.
In JMS the message sender just sends the message to the destination and continues it's own processing.

Answer

Message Object is a light weight entity that comprises of only header and properties. It does not comprise of payload. Thus, no data needs to be transferred when the receivers just need to be notified. Due to this using message becomes a very efficient way.

b. Write a simple JMS class that Creates and then reads a StreamMessage and a BytesMessage. (9)

Answer:

```
import javax.jms.*;

    public class MessageConversion {

      /**
       * Main method.  Takes no arguments.
       */
      public static void main(String[] args) {
        ConnectionFactory    connectionFactory = null;
        Connection           connection = null;
        Session              session = null;
        BytesMessage         bytesMessage = null;
        StreamMessage        streamMessage = null;
        int                  exitResult = 0;

        try {
         connectionFactory =
            SampleUtilities.getConnectionFactory();
         connection =
            connectionFactory.createConnection();
         session = connection.createSession(false,
            Session.AUTO_ACKNOWLEDGE);
      } catch (Exception e) {
         System.out.println("Connection problem: " + e.toString());
         if (connection != null) {
            try {
               connection.close();
            } catch (JMSException ee) {}
         }
        System.exit(1);
      }

       try {

         streamMessage = session.createStreamMessage();
        streamMessage.writeBoolean(false);
        streamMessage.writeDouble(123.456789e222);
        streamMessage.writeInt(223344);
        streamMessage.writeChar('q');
         streamMessage.reset();
         System.out.println("Reading StreamMessage items of various data"
            + " types as String:");
         System.out.println(" Boolean: " + streamMessage.readString());
         System.out.println(" Double: " + streamMessage.readString());
         System.out.println(" Int: " + streamMessage.readString());
         System.out.println(" Char: " + streamMessage.readString());

               streamMessage.clearBody();
         streamMessage.writeString("true");
         streamMessage.writeString("123.456789e111");
         streamMessage.writeString("556677");
         // Not char:  String to char conversion isn't valid
         streamMessage.reset();
         System.out.println("Reading StreamMessage String items as other"
            + " data types:");
         System.out.println(" Boolean: " + streamMessage.readBoolean());
         System.out.println(" Double: " + streamMessage.readDouble());
         System.out.println(" Int: " + streamMessage.readInt());

               bytesMessage = session.createBytesMessage();
        bytesMessage.writeBoolean(false);
        bytesMessage.writeDouble(123.456789e22);
```

```
                    bytesMessage.writeInt(778899);
                    bytesMessage.writeInt(0x7f800000);
                    bytesMessage.writeChar('z');
                          bytesMessage.reset();
                     System.out.println("Reading BytesMessages of various types:");
                     System.out.println(" Boolean: " + bytesMessage.readBoolean());
                     System.out.println(" Double: " + bytesMessage.readDouble());
                     System.out.println(" Int: " + bytesMessage.readInt());
                     System.out.println(" Float: " + bytesMessage.readFloat());
                     System.out.println(" Char: " + bytesMessage.readChar());
                   } catch (JMSException e) {
                     System.out.println("JMS Exception occurred: " + e.toString());
                     exitResult = 1;
                   } finally {
                     if (connection != null) {
                       try {
                         connection.close();
                       } catch (JMSException e) {
                         exitResult = 1;
                       }
                     }
                   }
                   SampleUtilities.exit(exitResult);
                 }
               }
```

Q.7 a. How do you set cookies in the JSP? How to read cookies with JSP? How to delete cookies with JSP? (9)

Answer:

Setting cookies with JSP involves three steps:

- **Creating a Cookie object:** You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.
- **Setting the maximum age:** You use setMaxAge to specify how long (in seconds) the cookie should be valid.
- **Sending the Cookie into the HTTP response headers:** You use response.addCookie to add cookies in the HTTP response header

To read cookies, you need to create an array of javax.servlet.http.Cookie objects by calling the getCookies( ) method of HttpServletRequest. Then cycle through the array, and use getName() and getValue() methods to access each cookie and associated value.

A:To delete cookies is very simple. If you want to delete a cookie then you simply need to follow up following three steps:

- Read an already existing cookie and store it in Cookie object.

- Set cookie age as zero using **setMaxAge()** method to delete an existing cookie.
- Add this cookie back into response header

b. Write a JSP Filter that would print the clients IP address and current date time each time it would access any JSP file. (9)

Answer:

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

// Implements Filter class
public class LogFilter implements Filter  {
  public void  init(FilterConfig config)
                   throws ServletException{
    // Get init parameter
    String testParam = config.getInitParameter("test-param");

    //Print the init parameter
    System.out.println("Test Param: " + testParam);
  }
  public void  doFilter(ServletRequest request,
             ServletResponse response,
             FilterChain chain)
             throws java.io.IOException, ServletException {

    // Get the IP address of client machine.
    String ipAddress = request.getRemoteAddr();

    // Log the IP address and current timestamp.
    System.out.println("IP "+ ipAddress + ", Time "
                        + new Date().toString());

    // Pass request back down the filter chain
    chain.doFilter(request,response);
  }
  public void destroy( ){
    /* Called before the Filter instance is removed
    from service by the web container*/
  }
}
```

TEXT BOOK

I. Java Network Programming, Merlin Hughes, Michael Shoffner, Derek Hamner, 2nd Edition
II. Professional AJAX by Nicholas Zakas et alia, Wrox Press