**Assignment 03**

|  |
| --- |
|  |

**Q.1) Draw the diagram of Life Cycle of a Servlet.**

* **The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet.**

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked

There are three states of a servlet: new, ready and end

**Q.2) Enlist the advantages of Servlet.**

* **Advantages of Servlet:** There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:**because it creates a thread for each request, not process.
2. **Portability:**because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

**Q.3) Explain types of servlets and its architecture.**

* **The types of servlets, there are primarily two types, namely:**

1. Generic Servlets, 2. HTTP Servlets

* **Implementing Servlet Interface**

1. Extending Generic Servlet, **2.** Extending HTTP Servlet

* **Components of Servlet Architecture:** Below is the high level architecture diagram of servlet. Let’s see in brief, how does each component add to the working of a servlet.

|  |
| --- |
|  |

* 1. **Client:** The client is primarily working as a medium who is sending out HTTP requests over to the web server and again processing the response it gets back from the server.
  2. **Web Server:** Primary job of a web server is to process the requests and responses that a user sends over time and maintain how a web user would be able to access the files that has been hosted over the server.There are precisely two types of webservers:

1. Static web server 2. Dynamic web server

* 1. **Web Container:** Web container is another typical component in servlet architecture which is responsible for communicating with the servlets. Two prime tasks of a web container are:
  2. Managing the servlet lifecycle, **2.** URL mapping.

**Q.4) Explain Steps to create a servlet with example?**

* **Steps to create a servlet example:** There are given 6 steps to create a servlet example.

|  |
| --- |
|  |

1. **Create a directory structure:** The directory structure defines that where to put the different types of files so that web container may get the information and respond to the client. The Sun Microsystem defines a unique standard to be followed by all the server vendors. Let's see the directory structure that must be followed to create the servlet. As We can see that the servlet class file must be in the classes folder. The web.xml file must be under the WEB-INF folder.
2. **Create a Servlet:** There are three ways to create the servlet.

|  |
| --- |
| import javax.servlet.http.\*;  import javax.servlet.\*;  import java.io.\*;  public class DemoServlet extends HttpServlet{  public void doGet(HttpServletRequest req,HttpServletResponse res)  throws ServletException,IOException  {  res.setContentType("text/html");//setting the content type  PrintWriter pw=res.getWriter();//get the stream to write the data  pw.println("<html><body>");  pw.println("Welcome to servlet");  pw.println("</body></html>");    pw.close();//closing the stream  }} |

1. By implementing the Servlet interface
2. By inheriting the GenericServlet class
3. By inheriting the HttpServlet class

* The HttpServlet class is widely used to create the servlet because it provides methods to handle http requests such as doGet(), doPost, doHead() etc. In this example we are going to create a servlet that extends the HttpServlet class. In this example, we are inheriting the HttpServlet class and providing the implementation of the doGet() method. Notice that get request is the default request.

1. **Compile the Servlet:** For compiling the Servlet, jar file is required to be loaded. Different Servers provide different jar files:

|  |  |
| --- | --- |
| **Jar file** | **Server** |
| 1) servlet-api.jar | Apache Tomcat |
| 2) weblogic.jar | Weblogic |
| 3) javaee.jar | Glassfish |
| 4) javaee.jar | JBoss |

1. **Create a deployment descriptor:** The deployment descriptor is an xml file, from which Web Container gets the information about the servet to be invoked. The web container uses the Parser to get the information from the web.xml file. There are many xml parsers such as SAX, DOM and Pull
2. **Start the server and deploy the project:** To start Apache Tomcat server, double click on the startup.bat file under apache-tomcat/bin directory.
3. **Access the servlet.**

**Q.5) Describe Life Cycle of a Thread in servlet.**

* **Life Cycle of a Thread:** A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.

|  |
| --- |
|  |

* **Following are the stages of the life cycle :**
* **New −** A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.
* **Runnable −** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
* **Waiting −** Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
* **Timed Waiting −** A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
* **Terminated (Dead)**− A runnable thread enters the terminated state when it completes its task or otherwise terminates.

**Q6) Write short note on interfacing Java Servlet Program with JDBC Connection.**

* **To start with interfacing Java Servlet Program with JDBC Connection:**

1. Proper JDBC Environment should set-up along with database creation.
2. To do so, download the mysql-connector.jar file from the internet,
3. As it is downloaded, move the jar file to the apache-tomcat server folder,
4. Place the file in lib folder present in the apache-tomcat directory.
5. To start with the basic concept of interfacing:

* **Step 1: Creation of Database and Table in MySQL**

As soon as jar file is placed in the folder, create a database and table in MySQL,

mysql> create database demoprj;

mysql> use demoprj Database changed

mysql> create table demo(id int(10), string varchar(20));

mysql> desc demo;

--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| id | int(10) | YES | | NULL | |

| string| varchar(20) | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

* **Step 2: Implementation of required Web-pages:** Create a form in HTML file, where take all the inputs required to insert data into the database. Specify the servlet name in it, with the POST method as security is important aspects in database connectivity.

|  |
| --- |
| <!DOCTYPE html>  <html> <head>  <title>Insert Data</title>  </head> <body>      <!-- Give Servlet reference to the form as an instances      GET and POST services can be according to the problem statement-->      <form action="./InsertData" method="post">          <p>ID:</p>          <!-- Create an element with mandatory name attribute,          so that data can be transfer to the servlet using getParameter() -->          <input type="text" name="id"/>          <br/>          <p>String:</p>          <input type="text" name="string"/>          <br/><br/><br/>          <input type="submit"/>      </form>  </body> </html> |

|  |
| --- |
|  |

* **Step 3: Creation of Java Servlet program with JDBC ConnectionTo create a JDBC Connection steps are:**

1. Import all the packages
2. Register the JDBC Driver
3. Open a connection
4. Execute the query, and retrieve the result
5. Clean up the JDBC Environment.

**Q.7) Short note on DTD with attribute-type.**

* **DTD stands for Document Type Definition:** It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.
* A document type definition (DTD) provides you with the means to validate XML files against a set of rules. When you create a DTD file, you can specify rules that control the structure of any XML files that reference the DTD file.
* A DTD can contain declarations that define elements, attributes, notations, and entities for any XML files that reference the DTD file. It also establishes constraints for how each element, attribute, notation, and entity can be used within any of the XML files that reference the DTD file.
* Its main purpose is to define the structure of an XML document. It contains a list of legal elements and define the structure with the help of them.
* **Attribute-type:**

|  |  |
| --- | --- |
| **Type** | **Description** |
| [CDATA](https://www.quackit.com/xml/tutorial/dtd_attribute_types_cdata.cfm) | Character Data (text that doesn't contain markup) |
| [ENTITY](https://www.quackit.com/xml/tutorial/dtd_attribute_types_entity.cfm) | The name of an entity (which must be declared in the DTD) |
| [ENTITIES](https://www.quackit.com/xml/tutorial/dtd_attribute_types_entities.cfm) | A list of entity names, separated by whitespaces. (All entities must be declared in the DTD) |
| [Enumerated](https://www.quackit.com/xml/tutorial/dtd_attribute_types_enumerated.cfm) | A list of values. The value of the attribute must be one from this list. |
| [ID](https://www.quackit.com/xml/tutorial/dtd_attribute_types_id.cfm) | A unique ID or name. Must be a valid XML name. |
| [IDREF](https://www.quackit.com/xml/tutorial/dtd_attribute_types_idref.cfm) | Represents the value of an ID attribute of another element. |
| [IDREFS](https://www.quackit.com/xml/tutorial/dtd_attribute_types_idrefs.cfm) | Represents multiple IDs of elements, separated by whitespace. |
| [NMTOKEN](https://www.quackit.com/xml/tutorial/dtd_attribute_types_nmtoken.cfm) | A valid XML name. |
| [NMTOKENS](https://www.quackit.com/xml/tutorial/dtd_attribute_types_nmtokens.cfm) | A list of valid XML names, separated by whitespace. |
| [NOTATION](https://www.quackit.com/xml/tutorial/dtd_attribute_types_notation.cfm) | A notation name (which must be declared in the DTD). |

**Q.8) What is servlet? Explain Working of Servlet?**

* **Servlet:** A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. Servlet technology is used to create a web application (resides at server side and generates a dynamic web page. Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. Servlet is a web component that is deployed on the server to create a dynamic web page.
* **Working of servlet:**
  + 1. **Servlet class is loaded:** The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.
    2. **Servlet instance is created:** The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.
    3. **init method is invoked:** The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface.Syntax of the init method is given below: **public void init(ServletConfig config) throws ServletException**
    4. **service method is invoked:** The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below: **public void service(ServletRequest request, ServletResponse response)     throws ServletException, IOException**
    5. **Destroy method is invoked:** The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below: **public void destroy()**

**Q.9) Explain XML and advantages of XML & Parts of an XML document.**

* **Extensible Markup Language:** The Extensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books, transactions, invoices, and much more. XML tags are not predefined. You must define your own tags. XML is platform independent and language independent.
* **Features and Advantages of XML:**
  + 1. **XML separates data from HTML:** If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes. With XML, data can be stored in separate XML files. This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML.
    2. **XML simplifies data sharing:** In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.
    3. **XML simplifies data transport:** One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.
    4. **XML simplifies Platform change:** Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.
    5. **XML increases data availability:** Different applications can access your data, not only in HTML pages, but also from XML data sources. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.
    6. **XML can be used to create new internet languages:** A lot of new Internet languages are created with XML.**Examples:**
    - XHTML
    - WSDL for describing available web services
    - WAP and WML as markup languages for handheld devices
    - RSS languages for news feeds
    - RDF and OWL for describing resources and ontology
    - SMIL for describing multimedia for the web.

**Q.10) Describe XML DOM structure.**

* **XML Document Object Model (DOM):** The XML Document Object Model (DOM) class is an in-memory representation of an XML document. The DOM allows you to programmatically read, manipulate, and modify an XML document.

|  |
| --- |
|  |

|  |
| --- |
| <?xml version="1.0"?>  <books>  <book>  <author>Carson</author>  <price format="dollar">31.95</price>  <pubdate>05/01/2001</pubdate>  </book>  <pubinfo>  <publisher>MSPress</publisher>  <state>WA</state>  </pubinfo>  </books> |

* In the XML document structure, each circle in this illustration represents a node, which is called an **XmlNode** object. The **XmlNode** object is the basic object in the DOM tree. Both **XmlNode** and **XmlDocument** have performance and usability enhancements and have methods and properties to:
  1. Access and modify nodes specific to the DOM, such as element nodes, entity reference nodes, and so on.
  2. Retrieve entire nodes, in addition to the information the node contains, such as the text in an element node.
* **Node objects** have a set of **methods** and **properties**, as well as basic and well-defined **characteristics**. Some of these characteristics are:

1. Nodes have a single parent node, a parent node being a node directly above them. The only nodes that do not have a parent is the Document root, as it is the top-level node and contains the document itself and document fragments.
2. Most nodes can have multiple child nodes, which are nodes directly below them.

* The following is a list of node types that can have child nodes: **Document, DocumentFragment, EntityReference, Element, Attribute**.
* The **XmlDeclaration**, **Notation**, **Entity**, **CDATASection**, **Text**, **Comment**, **ProcessingInstruction**, and **DocumentType** nodes do not have child nodes. Nodes that are at the same level, represented in the diagram by the **book** and **pubinfo** nodes, are siblings.
* **Accessing Nodes:** We can access a node in three ways: By using the getElementsByTagName() method. The getElementsByTagName() Method getElementsByTagName() returns all elements with a specified tag name. Syntax: **node.getElementsByTagName("tagname");**

**Q.11) What is Ajax, How does it work with diagram?**

|  |
| --- |
|  |

* **Ajax (Asynchronous Javascript and XML):** It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.
* **How does it work?**
* AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.
* There are two types of requests synchronous as well as asynchronous.
* Synchronous requests are the one which follows sequentially i.e if one process is going on and in the same time another process wants to be executed, it will not be allowed that means the only one process at a time will be executed.
* This is not good because in this type most of the time CPU remains idle such as during I/O operation in the process which are the order of magnitude slower than the CPU processing the instructions.
* Thus to make the full utilization of the CPU and other resources use asynchronous calls.
* **AJAX comprises the following technologies:**
* **XHTML and**[**CSS**](https://www.hostinger.in/tutorials/what-is-css)**–** For presenting the information.
* **The Document Object Model (DOM) –** For the dynamic display data and its interaction.
* **XML,**[**HTML**](https://www.hostinger.in/tutorials/what-is-html)**, and XSLT –** For data interchange and manipulation. However, many developers have replaced XML with JSON since it originated from JavaScript.
* **XMLHttpRequest object –** Allows asynchronous communication with the web server.
* **JavaScript –** The [programming language](https://www.hostinger.in/tutorials/best-programming-languages-to-learn) that links all these web technologies.
* **How AJAX Works:**

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript.
3. The XMLHttpRequest object sends a request to a web server.
4. The server processes the request.
5. The server sends a response back to the web page.
6. The response is read by JavaScript.
7. Proper action (like page update) is performed by JavaScript.

**Q.12) Compare working of AJAX with conventional method?**

|  |  |
| --- | --- |
| **Conventional model** | **AJAX model** |
| The browser sends an HTTP request to the server. | The browser creates a JavaScript call, which then creates a new XMLHttpRequest object. |
| The web server receives and processes the request. | The new XMLHttpRequest object transfers data between the browser and the web server in an XML format. |
| The web server sends the requested data to the browser. | The XMLHttpRequest object sends a request for the updated page data to the web server. Subsequently, the latter processes the request and sends it back to the browser. |
| The browser receives the data from the server and reloads it as an HTML page. Users have to wait until it finishes loading. Therefore, the conventional model increases the load on the server and is more time-consuming. | The browser uses JavaScript to process the response and displays the updated content directly on the HTML page without reloading. |

**(PPT Give you more understanding than PDF)** The material for the PDF has been compiled from various sources such as books, tutorials (offline and online), lecture notes, several resources available on Internet. The information contained in this PDF is for general information and education purpose only. While we endeavor to keep the information up to date and correct, **we make no representation of any kind about the completeness and accuracy of the material.** The information shared through this PDF material should be used for educational purpose only.

**Thank you…**