1. What are the different kind of parsers used in XML? Explain briefly Differentiate CDATA and PCDATA in XML. A book has exactly one title, one or more chapters, one or more authors, a price and an ISBN number. An author has a first name, an optional middle name and a last name. A chapter consists of one or more paragraphs. Write a DTD for this scenario. [What are empty elements and how an empty element is declared in a DTD?]

Ans.

The different kinds of parsers used in XML are:

i)DOM (Document Object Model) Parser
- Loads the entire XML document into memory and creates a tree structure.
- Allows traversal, modification, and random access to any part of the XML.
- Best for small to medium-sized XML files.

ii)SAX (Simple API for XML) Parser
- Event-driven, parses XML document sequentially from start to end without loading it into memory.
- Suitable for large XML files as it is memory-efficient.
- Read-only, forward-only access, without modifying the XML structure.

iii)StAX (Streaming API for XML) Parser
- Combines the best of DOM and SAX, providing a pull-based approach where the application controls parsing.
- Efficient and memory-friendly, allowing both reading and writing of XML documents.

iv)XPath Parser
- Used to navigate and query XML documents.
- Allows selection of specific parts of an XML document using XPath expressions.

Each parser has its advantages depending on the size of the XML file and the specific use case (read, write, modify).

Ans.

Criteria	CDATA (Character Data)	PCDATA (Parsed Character Data)
Parsing	CDATA is not parsed by the XML parser.	PCDATA is parsed by the XML parser.
Special Characters	Special characters like `<`, `>`, and `&` are allowed and treated as literal text.	Special characters need to be escaped (e.g., `<` for `<`).

Usage	Used when you want to include raw, unparsed data within an XML element.	Used when the content needs to be parsed and interpreted.
Syntax	CDATA section is enclosed within `<![CDATA[ ... ]]>`.	No special syntax, it is just plain text within the element tags.
Modification	Contents within CDATA are treated as literal text, meaning they won't be altered by the parser.	PCDATA is parsed and processed, so special characters may be interpreted or converted.
Example	<example><![CDATA[<this is unparsed>]]></example>	<example>This is parsed data <example></example>

Ans.
DTD for the Book Scenario -

<!ELEMENT book (title, author+, chapter+, price, isbn)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (firstname, middlename?, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT middlename (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT chapter (paragraph+)>
<!ELEMENT paragraph (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>

Ans.
Empty elements are elements that do not contain any content (i.e., no text or nested elements).
They are used to represent tags that do not have any value or data between their opening and closing tags.

Ans.
An empty element in DTD is declared using `EMPTY`. For example:
<!ELEMENT emptytag EMPTY>

This indicates that `<emptytag />` is a valid usage in the XML document.

2.What is web container ? Write a simple servlet program which will return the current data and time. Discuss the lifecycle of a servlet

Ans.

A **web container** (also known as a **servlet container**) is a part of a web server that manages the execution of Java-based web applications. It provides the environment in which Java servlets, JavaServer Pages (JSP), and other web components run.

Ans.

```
import java.io.*;
import javax.servlet.*;
public class DateSrv extends GenericServlet
{
//implement service()
public void service(ServletRequest req, ServletResponse res) throws IOException,
ServletException
{
//set response content type
res.setContentType("text/html");
//get stream obj
PrintWriter pw = res.getWriter();
//write req processing logic
java.util.Date date = new java.util.Date();
pw.println("<h2>"+"Current Date & Time: " +date.toString()+"</h2>");
//close stream object

pw.close();
}
}
```

Output:

Current Date & Time: Wed Sep 18 10:23:47 IST 2024

Ans.

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.

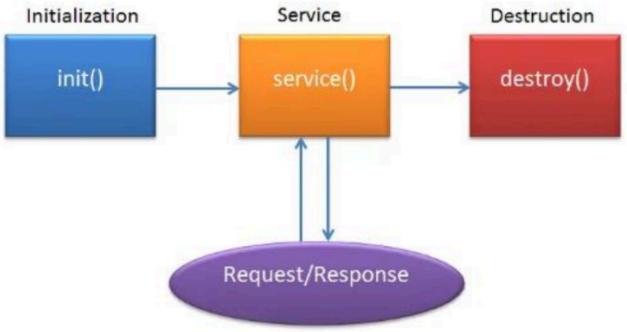
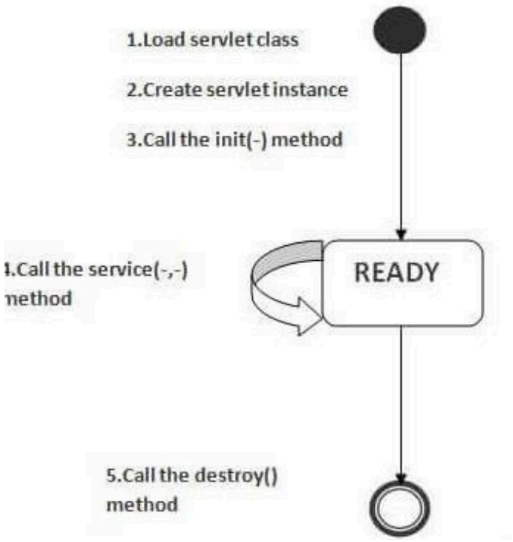


Fig: Life Cycle of a Servlet



In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it
shifts to the end state.

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()

3.Write a javascript code to calculate factorial of a number.What is the use of cookies in Javascript? Show with example how to get and set cookies in Javascript.(Note: The full code is not needed Only the code excerpts showing how to get and set cookies.) Compare DTD and XML schema.

Ans.

```
// program to find the factorial of a number
// take input from the user
const number = parseInt(prompt(&#39;Enter a positive integer: &#39;));
// checking if number is negative
if (number &lt; 0) {
console.log(&#39;Error! Factorial for negative number does not exist.&#39;);
}
// if number is 0

else if (number === 0) {
console.log(`The factorial of ${number} is 1.`);
}
// if number is positive
else {
let fact = 1;
for (i = 1; i &lt;= number; i++) {
fact *= i;
}
console.log(`The factorial of ${number} is ${fact}.`);
}
```

Output
Enter a positive integer: 5
The factorial of 5 is 120.
Ans.
Cookies in JavaScript are used to store small pieces of data on the client's machine. They can be used to remember user preferences, login sessions, or track user behavior across pages.
Ans.

Set a Cookie:

```
function setCookie(cname, cvalue, exdays) {
    let d = new Date();
    d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
    let expires = "expires=" + d.toUTCString();
    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

// Example: Setting a cookie
setCookie("username", "JohnDoe", 7);  // Set cookie with expiry of 7 days
```

Get a Cookie:

```
function getCookie(cname) {
    let name = cname + "=";
    let decodedCookie = decodeURIComponent(document.cookie);
    let ca = decodedCookie.split(';');
    for(let i = 0; i < ca.length; i++) {
        let c = ca[i].trim();
        if (c.indexOf(name) === 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}

// Example: Getting a cookie
let user = getCookie("username");
```

Ans.
Comparison: DTD vs XML Schema:

Aspect	DTD (Document Type Definition)	XML Schema

Type	Simple, older specification for XML validation	More powerful, newer specification for XML validation
Data Types	Limited support (only basic types like `#PCDATA`, etc.)	Supports many data types (e.g., `string`, `int`, `date`)
Namespace Support	No support for XML namespaces	Fully supports namespaces
Syntax	Not XML-based, uses its own syntax	XML-based, more readable and extensible
Extensibility	Limited flexibility and extensibility	Highly extensible, supports user-defined types

4.Create XML document which has a root element student. Root element contains more than one student element. Each student has sub-element student_name, student_id, gender,course_name and marks. Write XSLT code to display the names and gender of the students Write XSLT code which will print the names of the students in sorted order Write XSLT code which will print the marks and names of the students who has got marks>75 .What is the need for Namespace in XML?.

Ans.

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
  <student>
    <student_name>John Doe</student_name>
    <student_id>101</student_id>
    <gender>Male</gender>
    <course_name>Mathematics</course_name>
    <marks>85</marks>
  </student>
  <student>
    <student_name>Jane Smith</student_name>
    <student_id>102</student_id>
    <gender>Female</gender>
    <course_name>Physics</course_name>
    <marks>72</marks>
  </student>
  <student>
    <student_name>Emily Johnson</student_name>
    <student_id>103</student_id>
    <gender>Female</gender>
    <course_name>Chemistry</course_name>
```

```
    <marks>90</marks>
  </student>
</students>
```

Ans.
XSLT Code to Display Names and Gender of Students

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!-- Root template -->
  <xsl:template match="/">
    <html>
      <body>
        <h2>Student Names and Gender</h2>
        <table border="1">
          <tr>
            <th>Name</th>
            <th>Gender</th>
          </tr>
          <xsl:for-each select="students/student">
            <tr>
              <td><xsl:value-of select="student_name"/></td>
              <td><xsl:value-of select="gender"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

Ans.
XSLT Code to Display Names of Students in Sorted Order

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!-- Root template -->
  <xsl:template match="/">
    <html>
      <body>
        <h2>Student Names in Sorted Order</h2>
```

```
      <ul>
        <xsl:for-each select="students/student">
          <!-- Sorting the student names alphabetically -->
          <xsl:sort select="student_name" order="ascending"/>
          <li><xsl:value-of select="student_name"/></li>
        </xsl:for-each>
      </ul>
    </body>
  </html>
 </xsl:template>

</xsl:stylesheet>
```

Ans.
XSLT Code to Print Names and Marks of Students with Marks > 75

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <!-- Root template -->
 <xsl:template match="/">
  <html>
   <body>
    <h2>Students with Marks Greater than 75</h2>
    <table border="1">
     <tr>
      <th>Name</th>
      <th>Marks</th>
     </tr>
     <!-- Iterating through student elements -->
     <xsl:for-each select="students/student">
      <!-- Filter: students with marks greater than 75 -->
      <xsl:if test="marks &gt; 75">
       <tr>
        <td><xsl:value-of select="student_name"/></td>
        <td><xsl:value-of select="marks"/></td>
       </tr>
      </xsl:if>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
```

</xsl:stylesheet>

Ans.

Need for Namespace in XML

Namespaces in XML are used to avoid name conflicts when combining documents or elements from different XML vocabularies. They allow differentiation between elements and attributes that may have the same name but different meanings in different contexts.

5.What are Servlets? What are the advantages and disadvantages of Servlets? Briefly explain the Servlet API and Servlet Packages .
Ans.

Servlets are Java programs that run on a web server or application server and handle requests from web clients (usually browsers), process the requests, and generate dynamic responses (often in HTML or XML). Servlets act as a middle layer between client requests and server responses.

Ans.
Advantages of Servlets:

1. Platform-Independent: Written in Java, servlets are portable across different operating systems and servers.
2. Performance: Servlets are efficient, as they run in the server's JVM (Java Virtual Machine) and are loaded once, processing multiple requests.
3. Scalability: Support for handling multiple concurrent requests efficiently using multithreading.
4. Reusability: Servlets can be reused across different applications, improving modularity.
5. Security: Servlets benefit from Java's robust security model and can be combined with SSL, access control, and session tracking.
6. Integration: Easily integrates with other Java technologies like JDBC, JSP, and EJB.

Disadvantages of Servlets:

1. Complexity: Writing servlets can sometimes be more complex than simpler web technologies like JSP.
2. Requires Java Knowledge: Developers must be proficient in Java to work with servlets.
3. Maintenance Overhead: Maintaining servlets for large-scale applications may require additional effort compared to higher-level frameworks like Spring MVC.
4. No Separation of Concerns: Mixing business logic with HTML output can lead to less maintainable code, but this can be mitigated using JSP.

Ans.
Servlet API and Servlet Packages:

1. Servlet API

The Servlet API provides classes and interfaces for writing servlets. The core interfaces and classes in the API are essential for handling client requests, processing them, and generating responses.

Key Interfaces in the Servlet API:
- Servlet Interface: Defines the basic methods every servlet must implement, such as init(), service(), and destroy().
- GenericServlet Class: An abstract class implementing the Servlet interface, providing generic methods for protocol-independent services.
- HttpServlet Class: A subclass of GenericServlet, specifically designed to handle HTTP requests (GET, POST, etc.).
- ServletRequest Interface: Encapsulates client request information, including parameters, attributes, and input streams.
- ServletResponse Interface: Encapsulates server response information, including output streams and headers.
- ServletContext Interface: Provides information about the web application context and is used to share data between servlets.

2. Servlet Packages
Servlet-related classes and interfaces are part of the following packages:

- javax.servlet: Contains the basic classes and interfaces required to develop servlets. Core classes include GenericServlet, ServletRequest, and ServletResponse.

- javax.servlet.http: Contains HTTP-specific servlet classes like HttpServlet, HttpSession, and Cookie. This package allows servlets to interact with the HTTP protocol.

6.What is CGI ? What are the advantages of Servlet over CGI ? Compare DOM and SAX in XML processing List and explain any two JavaScript built in objects.
Ans.

CGI (Common Gateway Interface) is a standard protocol used to enable web servers to interact with external programs (often scripts) to generate dynamic content. It allows web servers to execute scripts or applications written in languages like Python, Perl, or C and send the generated response (HTML or other) back to the client.

Ans.

Advantages of Servlet over CGI:

1. Performance: Servlets are faster since they run in the server's JVM, whereas CGI spawns a new process for every request.

2. Memory Usage: Servlets are loaded once and handle multiple requests, while CGI creates a separate process for each request, leading to high memory usage.
3. Scalability: Servlets use multithreading, handling multiple requests efficiently, while CGI processes requests one by one.
4. Portability: Servlets are Java-based and thus platform-independent, while CGI scripts depend on the server's operating system.

Comparison: DOM vs SAX in XML Processing

Aspect	DOM (Document Object Model)	SAX (Simple API for XML)
Type	Tree-based (loads entire XML document in memory)	Event-based (parses XML as a stream of events)
Memory	Requires more memory (loads entire XML file)	Requires less memory (processes line-by-line)
Speed	Slower for large XML files	Faster, especially for large files
Read/Write	Allows both reading and modifying XML data	Read-only, no direct modification of XML

Ans.

Two JavaScript Built-in Objects:

1. `Date` Object:
    - Provides methods to work with dates and times.
    - Example: `let today = new Date();` allows access to the current date and time.
    - Methods: `getDate()`, `getMonth()`, `getFullYear()`, etc.
2. `Math` Object:
    - Provides mathematical constants and functions.
    - Example: `let randomNum = Math.random();` generates a random number between 0 and 1.
    - Methods: `Math.max()`, `Math.min()`, `Math.pow()`, `Math.sqrt()`.

7.Write a program in JavaScript to validate IP address (in IPV4 address format) in a form using regular expression. Flight numbers are typically of the format AI 123 or 6E 2415. First two letters correspond to the flight company followed by a space, and then followed by 3 or 4 digits. Write a

regular expression that will validate the above format. Write a JavaScript program to remove the duplicate items from an array (ignore case sensitivity).

Ans.

JavaScript Program to Validate IP Address (IPv4 Format) Using Regular -

```
function validateIPAddress(ip) {
   const ipRegex =
/^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9
][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/;
   return ipRegex.test(ip);
}

// Example usage:
let ip = "192.168.1.1";
console.log(validateIPAddress(ip) ? "Valid IP" : "Invalid IP");
```

Ans.

Regular Expression for Validating Flight Numbers (AI 123 or 6E 2415):

```
const flightNumberRegex = /^[A-Z]{2} \d{3,4}$/;

// Example usage:
let flightNumber = "AI 123";
console.log(flightNumberRegex.test(flightNumber) ? "Valid Flight Number" : "Invalid Flight Number");
```

Ans.

JavaScript Program to Remove Duplicate Items from an Array (Ignoring Case Sensitivity):

```
function removeDuplicates(arr) {
   let lowerCaseArr = arr.map(item => item.toLowerCase());
   let uniqueArr = [...new Set(lowerCaseArr)];
   return uniqueArr;
}

// Example usage:
let items = ["Apple", "Banana", "apple", "ORANGE", "banana", "Orange"];
let uniqueItems = removeDuplicates(items);
console.log(uniqueItems);  // Output: ["apple", "banana", "orange"]
```

8.Write a JavaScript program to do a global, case-insensitive search for the pattern "is" in a string.Explain how local and global functions can be written using JavaScript? Write a program to illustrate Popup boxes in JavaScript.

Ans.
JavaScript Program for Global, Case-Insensitive Search for "is" in a String:

```
function searchPattern(str) {
   const pattern = /is/gi; // Global (g) and case-insensitive (i) flags
   const matches = str.match(pattern);
   return matches ? matches.length : 0;
}

// Example usage:
let text = "This is a simple string. IS it clear?";
console.log("Occurrences of 'is':", searchPattern(text));  // Output: 3
```

Ans.
Global Function: A global function is declared outside any function or block. It is accessible from any part of the script.

```
function globalFunction() {
   console.log("This is a global function");
}

globalFunction(); // This can be called from anywhere in the script.
```

Local Function: A local function is defined inside another function (or block). It is accessible only within the scope where it is defined.

```
function outerFunction() {
   function localFunction() {
      console.log("This is a local function");
   }
   localFunction(); // Can only be called within outerFunction
}

outerFunction();  // Works
// localFunction(); // This will cause an error if called here, as it's not accessible globally
```

Ans.
JavaScript provides three types of popup boxes:

Alert box: Displays a simple message.
Confirm box: Asks for a user confirmation.
Prompt box: Asks for user input.

```
// Alert box
function showAlert() {
   alert("This is an alert box!");  // Just a notification to the user
```

```
}

// Confirm box
function showConfirm() {
    let confirmResult = confirm("Do you want to proceed?");
    if (confirmResult) {
        console.log("User clicked OK.");
    } else {
        console.log("User clicked Cancel.");
    }
}

// Prompt box
function showPrompt() {
    let userInput = prompt("Please enter your name:");
    if (userInput != null) {
        console.log("Hello, " + userInput + "!");
    }
}

// Example usage:
showAlert();    // Opens alert popup
showConfirm();  // Opens confirm popup
showPrompt();   // Opens prompt popup
```