**What is a Servlet?**

A servlet is a Java program that runs within a Web server and it is used to create a web application. Servlets receive requests from Web clients and respond them.

**How servlet works?**

Servlets handle client requests by first receiving the HTTP request, which the web server forwards to the servlet container. If the servlet isn't already loaded, the container initializes it. The servlet processes the request, determines the request type, and dispatches it to the appropriate method like doGet() or doPost(). It generates the response content based on the request, sets the response properties such as content type and status code, and writes the content to the HttpServletResponse output stream. The servlet container then sends this response back to the client, completing the request-response cycle. The servlet remains ready for further requests and may be destroyed if no longer needed, allowing resource cleanup.

**What is web server?**

A web server is a software application that serves web content to clients over the internet or an intranet. It hosts websites and web applications, responding to client requests made through web browsers. When a client sends an HTTP request, the web server processes the request, retrieves the appropriate content (such as HTML files, images, or data from a database), and sends it back to the client in the form of an HTTP response.

### What is a web application?

A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript. The web components typically execute in Web Server and respond to the HTTP request.

### CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.



### Disadvantages of CGI

There are many problems in CGI technology:

1. If the number of clients increases, it takes more time for sending the response.
2. For each request, it starts a process, and the web server is limited to start processes.
3. It uses platform dependent language e.g. [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), [perl](https://www.javatpoint.com/perl-tutorial).

**What are the advantages of Servlet over CGI?**

**Performance**:

* **Efficiency**: Servlets are Java programs that run within a web server's process, meaning they do not require a new process to be created for each request. This reduces the overhead associated with creating and destroying processes, which is a common performance bottleneck in CGI.
* **Multithreading**: Servlets handle multiple requests using threads rather than separate processes. Threads are less resource-intensive than processes, which leads to better performance and scalability.

**Portability**:

* **Platform Independence**: Servlets are written in Java, which is platform-independent. They can run on any system that has a Java-compatible web server, whereas CGI scripts, often written in languages like Perl or C, may require platform-specific adjustments.

**Integration and Maintenance**:

* **Integration with Java Libraries**: Servlets can easily use Java's extensive set of libraries, including APIs for database access (JDBC), networking, and more. This integration simplifies development and enables more powerful applications.
* **Maintainability**: Servlets benefit from Java's object-oriented features, such as encapsulation, inheritance, and polymorphism, which can make the code more modular, reusable, and maintainable compared to CGI scripts.

**Security**:

* **Java Security Features**: Servlets can leverage Java's security model, which includes features like sandboxing, secure class loading, and the ability to restrict file system access, making it easier to create secure applications.

**Session Management**:

* **Built-in Support**: Servlets provide built-in support for session management, which simplifies the handling of user sessions and state between HTTP requests. This feature is less standardized and more cumbersome to implement in CGI.

**Ease of Development**:

* **API Consistency**: The Java Servlet API provides a consistent interface for web development, which can streamline development and reduce the learning curve for new developers.
* **Rich Ecosystem**: Servlets are part of the Java EE (Enterprise Edition) ecosystem, which includes a wide range of tools and frameworks for building enterprise-level applications.

**How is a Servlet implemented in code?**

**Answer:** Servlet can be implemented in code by simply extending the Httpservlet or generic servlet class.

**What is the difference between the Http Servlet and Generic Servlet?**

**Answer:** Generic Servlet can handle all types of requests. As it has a service () method, it is independent, whereas Http Servlet extends the generic servlet and supports the HTTP methods such as doGet (), doPost (), doHead (), doTrace (), etc.

**What do you mean by deployment descriptor?**

**Answer:** WEB.XML is said to be the deployment descriptor in a servlet.

It is the entry point for any application and possesses the welcome file list. It d

**Explain load on start-up and its importance?**

**Answer:**Load on start-up is an element defined in web.xml (deployment descriptor) which helps the servlet to load at the time of deployment while the server is restarting.

The reason to use load on start-up is as the servlet is loaded on the first request received so initially it takes more time to load resulting decreased efficiency if we define loan on start-up is **loads** the servlet while server restarting which increases efficiency.

**Load on start-up is also working on two values:**

* **Positive (0,1,2,3….):** The lowest positive value will be loaded first.
* **Negative:** The servlet will be loaded when the first request is received.

**Is servlet synchronized?**

**Answer:** No, the servlets are not synchronized. If we want to make the servlet synchronized, we must implement SingleThreadInterface.

**Explain is servlet mapping?**

**Ans:** Servlet mapping is a process of defining an association between a URL pattern and a servlet. The mapping is used to map requests to Servlets.

**What are the annotations used in Servlet 3?**

**Ans:** The important 3 annotations used in the servlets are.

* **@WebServlet** : for servlet class.
* **@WebListener** : for listener class.
* **@WebFilter** : for filter class.

**What is the difference between a Generic Servlet and HTTP Servlet?**

**Ans:**A **common feature** between Generic Servlet and HTTP Servlet is both these Classes are [**Abstract Classes.**](https://www.edureka.co/blog/abstract-classes-in-java/) But, they do have differences between them which discussed as follows

|  |  |
| --- | --- |
| **Generic Servlet** | **HTTP Servlet** |
| Protocol Independent | Protocol Specific |
| Belongs to javax.servlet package | Belongs to javax.servlet.http package |
| supports only service() method | supports doGet(), doPost(), doHead() methods |

**Life Cycle of a Servlet (Servlet Life Cycle)**



**Servlet Class Loading**: 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.

**Servlet Instance Creation**: AThe 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.

**init() Method Invocation**: The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet.

public void init(ServletConfig config) throws ServletException

**service() Method Invocation**: The service() method handles client requests, determining the request type (GET, POST, etc.) and dispatching it to the appropriate method (e.g., doGet(), doPost()). The web container calls the service method each time when request for the servlet is received.

public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException

**destroy() Method Invocation:** 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.

public void destroy()

**6) What is difference between Get and Post method?**

| **Feature** | **GET Method** | **POST Method** |
| --- | --- | --- |
| **Purpose** | Retrieve data from the server | Send data to the server, potentially modifying server state |
| **Data Transmission** | Data is sent in the URL as a query string | Data is sent in the body of the request |
| **Data Visibility** | Data is visible in the URL | Data is not visible in the URL |
| **Security** | Less secure for sensitive data | More secure for sensitive data |
| **Data Length** | Limited by URL length restrictions | No strict limit on data length |
| **Caching** | Can be cached by browsers and servers | Typically not cached |
| **Bookmarking** | URLs can be bookmarked | URLs cannot be bookmarked |
| **Idempotence** | Idempotent (multiple identical requests have the same effect) | Not necessarily idempotent (can change server state) |

**7) What is difference between PrintWriter and ServletOutputStream?**

PrintWriter is a character-stream class where as ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas ServletOutputStream class can be used to write primitive values as well as character-based information.

**8) What is difference between GenericServlet and HttpServlet?**

The GenericServlet is protocol independent whereas HttpServlet is HTTP protocol specific. HttpServlet provides additional functionalities such as state management etc.

**9) What is servlet collaboration?**

When one servlet communicates to another servlet, it is known as servlet collaboration.

### What is the purpose of RequestDispatcher Interface?

The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of antoher resource.

RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");

Rd.forward(request,response);

**What is ServletConfig?**

ServletConfig is an interface in the Java Servlet API used to provide configuration information to servlets. It is used to pass initialization parameters to a servlet at the time of its creation. Here’s a brief overview:

**Key Points About ServletConfig**

* **Purpose**: ServletConfig allows a servlet to access initialization parameters defined in the web.xml deployment descriptor or annotations. These parameters can be used to configure the servlet when it is first loaded.
* **Access**: ServletConfig is provided to the servlet by the servlet container and is available through the servlet’s init() method. It can be accessed using getServletConfig() within the servlet.
* **Methods**:
  + getServletName(): Returns the name of the servlet.
  + getInitParameter(String name): Returns the value of a specific initialization parameter.
  + getInitParameterNames(): Returns an Enumeration of all initialization parameter names.
  + getServletContext(): Returns the ServletContext object, which provides access to application-wide parameters and attributes.
* **Usage**: It is commonly used for retrieving initialization parameters configured in the deployment descriptor (web.xml) or through servlet annotations. For example, you might use ServletConfig to get database connection details or other configuration values needed for the servlet’s operation.

**Example Usage**

public class MyServlet extends HttpServlet {

private String dbUrl;

@Override

public void init(ServletConfig config) throws ServletException {

super.init(config);

dbUrl = config.getInitParameter("databaseUrl");

}

// Other servlet methods

}

In this example, the init() method retrieves an initialization parameter named "databaseUrl" from the ServletConfig object, which can then be used by the servlet for its operations.

**what is ServletContext**

ServletContext is an interface in the Java Servlet API that provides a way for servlets to interact with their environment and access application-wide parameters and resources. It represents the context in which a servlet operates within a web application.

**Key Points About ServletContext**

* **Purpose**: ServletContext allows servlets to obtain information about the web application and access resources that are common across all servlets within the application.
* **Access**: ServletContext can be accessed from a servlet using the getServletContext() method from within the servlet's init() method or through ServletConfig. It can also be accessed using the ServletContext object directly from the servlet container.
* **Methods**:
  + **getContext**(String uri): Returns a ServletContext object for the specified URI, allowing access to another web application context.
  + **getAttribute**(String name): Retrieves an attribute set in the context.
  + **setAttribute**(String name, Object object): Sets an attribute in the context.
  + **removeAttribute**(String name): Removes an attribute from the context.
  + **getResource**(String path): Retrieves a URL to a resource in the web application.
  + **getResourceAsStream**(String path): Retrieves an InputStream to a resource in the web application.
* **Usage**:
  + **Sharing Data**: ServletContext is often used to share data and objects between different servlets and JSPs within the same web application.
  + **Accessing Resources**: It allows servlets to access resources like files and images located within the web application.
  + **Logging**: Servlets can use ServletContext to log messages to the server's log file.

**Example Usage**

public class MyServlet extends HttpServlet {

@Override

public void init() throws ServletException {

ServletContext context = getServletContext();

String dbConfig = context.getInitParameter("databaseConfig");

context.log("Database configuration: " + dbConfig);

}

// Other servlet methods

}

In this example, the init() method retrieves a context-wide initialization parameter named "databaseConfig" and logs its value. This demonstrates how ServletContext can be used to access and share configuration information within a web application.

### What is Session Tracking?

**Session** simply means a particular interval of time.

Session Tracking is a way to maintain state of an user. Http protocol is a stateless protocol. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

### Session Tracking Techniques

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

# **Session tracking Cookies**

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

### How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



### Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

### Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

### Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

### Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

### Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

### Cookie class

**javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

### Constructor of Cookie class

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| Cookie() | constructs a cookie. |
| Cookie(String name, String value) | constructs a cookie with a specified name and value. |

### Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void setMaxAge(int expiry) | Sets the maximum age of the cookie in seconds. |
| public String getName() | Returns the name of the cookie. The name cannot be changed after creation. |
| public String getValue() | Returns the value of the cookie. |
| public void setName(String name) | changes the name of the cookie. |
| public void setValue(String value) | changes the value of the cookie. |

### Other methods required for using Cookies

|  |
| --- |
| For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:   1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object. 2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser. |

### How to create Cookie?

Let's see the simple code to create cookie.

Cookie ck=**new** Cookie("user","sonoo jaiswal");//creating cookie object

response.addCookie(ck);//adding cookie in the response

### How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

Cookie ck=**new** Cookie("user","");//deleting value of cookie

ck.setMaxAge(0);//changing the maximum age to 0 seconds

response.addCookie(ck);//adding cookie in the response

### How to get Cookies?

Let's see the simple code to get all the cookies.

Cookie ck[]=request.getCookies();

**for**(**int** i=0;i<ck.length;i++){

 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie

}

# **Session tracking using Hidden Form Field**

In case of Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.

In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

Let's see the code to store value in hidden field.

<input type="hidden" name="uname" value="Vimal Jaiswal">

Here, uname is the hidden field name and Vimal Jaiswal is the hidden field value.

### Real application of hidden form field

It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

### Advantage of Hidden Form Field

1. It will always work whether cookie is disabled or not.

### Disadvantage of Hidden Form Field:

1. It is maintained at server side.
2. Extra form submission is required on each pages.
3. Only textual information can be used.

### Example of using Hidden Form Field

In this example, we are storing the name of the user in a hidden textfield and getting that value from another servlet.



**Example :**

<form action="processForm" method="post">

<input type="hidden" name="sessionId" value="ABC123XYZ">

<!-- Other form fields -->

<input type="submit" value="Submit">

</form>

# **Session tracking using URL Rewriting**

In URL rewriting, we append (attaches) a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.



### Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

### Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

**Example:**

<a href="profile.jsp?userId=<%= userId %>">Go to Profile</a>

# **Session tracking using HttpSession interface**

In such case, container creates a session id for each user.The container uses this id to identify the particular user.An object of HttpSession can be used to perform two tasks:

1. bind objects
2. view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



### How to get the HttpSession object ?

The HttpServletRequest interface provides two methods to get the object of HttpSession:

1. **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
2. **public HttpSession getSession(boolean create):**Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

### Commonly used methods of HttpSession interface

1. **public String getId():**Returns a string containing the unique identifier value.
2. **public long getCreationTime():**Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
3. **public long getLastAccessedTime():**Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
4. **public void invalidate():**Invalidates this session then unbinds any objects bound to it.

**Storing Data in Session**:

* You can store user-specific data in the HttpSession object using setAttribute(). This data will be available across multiple requests from the same client.

HttpSession session = request.getSession();

session.setAttribute("username", "JohnDoe");

**Retrieving Data from Session**:

* To retrieve data stored in the session, use getAttribute().

HttpSession session = request.getSession();

String username = (String) session.getAttribute("username");

**Invalidating a Session** (for logout)

* You can invalidate a session to remove all its data and terminate it using invalidate().

HttpSession session = request.getSession();

session.invalidate();

### What is filter?

A filter is an object that is invoked either at the preprocessing or postprocessing of a request. It is pluggable.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

* data compression
* encryption and decryption
* input validation etc.

### How can we upload the file to the server using servlet?

One of the way is by MultipartRequest class provided by third party.

### What is war file?

A war (web archive) file specifies the web elements. A servlet or jsp project can be converted into a war file. Moving one servlet project from one place to another will be fast as it is combined into a single file.

[more details...](https://www.javatpoint.com/war-file)

### 24) How to create war file?

The war file can be created using jar tool found in jdk/bin directory. If you are using Eclipse or Netbeans IDE, you can export your project as a war file.

To create war file from console, you can write following code.

jar -cvf abc.war \*