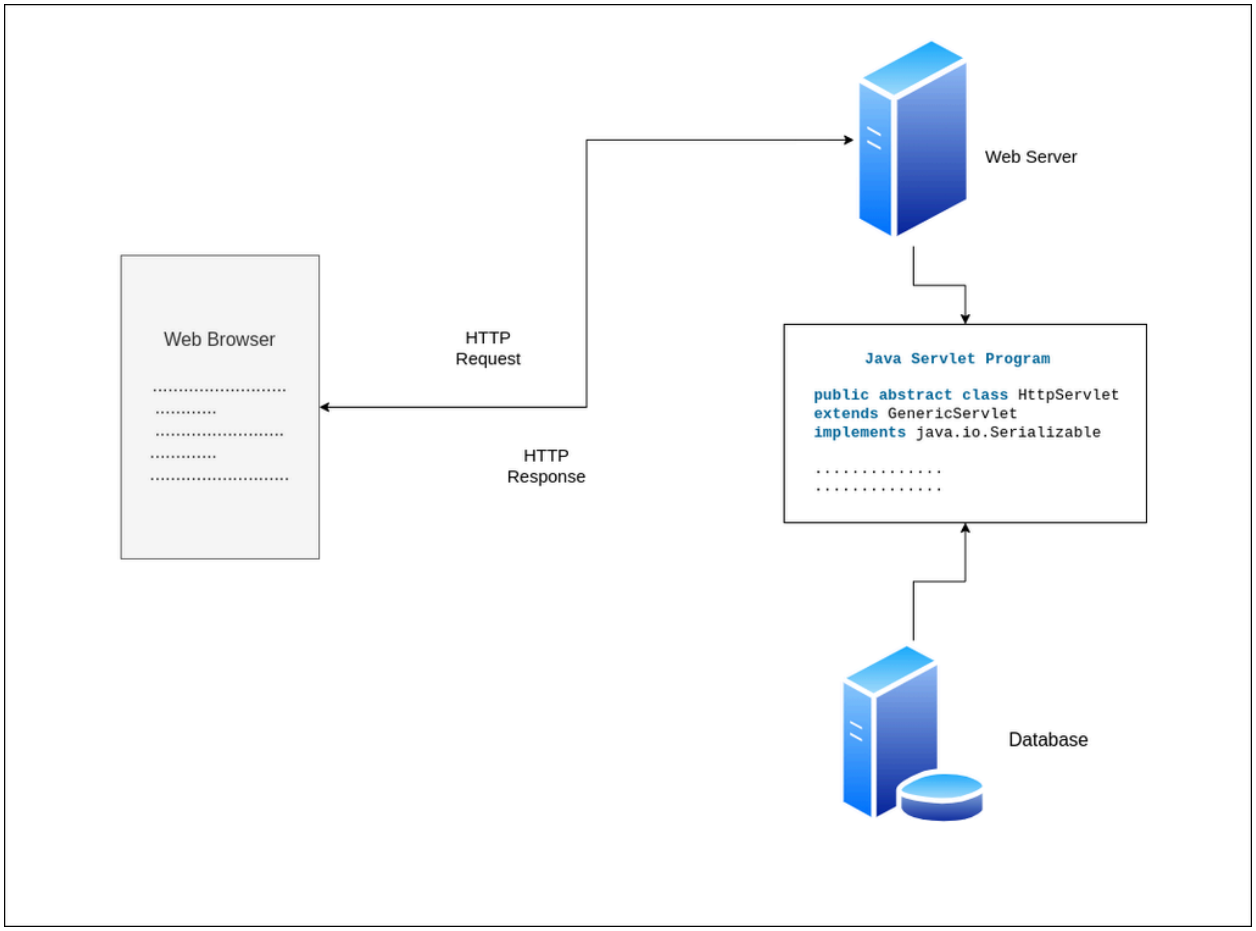
Develop a web application using Servlets e.g. Create a database table ebookshop (book_id, book_title, book_author, book_price, quantity) using database like Oracle/MySQL etc. and display (use SQL select query) the table content using servlet.

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

Java Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.

Servlets work on the server side.



1. The Clients send the request to the Web Server.

2. The Web Server receives the request.
3. The Web Server passes the request to the corresponding servlet.
4. The Servlet processes the request and generates the response in the form of output.
5. The Servlet sends the response back to the webserver.
6. The Web Server sends the response back to the client and the client browser displays it on the screen.

Life Cycle of a Servlet (Servlet Life Cycle):
1. **Servlet class is loaded**: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.
4. **service method is invoked:**The web container calls the service method each time when a request for the servlet is received.
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.

Http:data communication protocol used to establish communication between client and server.HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80

Http request :it is request send by computer to web server that contains all sort of information

Container:it is used in java for dynamically creating web pages on server side

Web server:It is a computer where the web content can be stored.

- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource

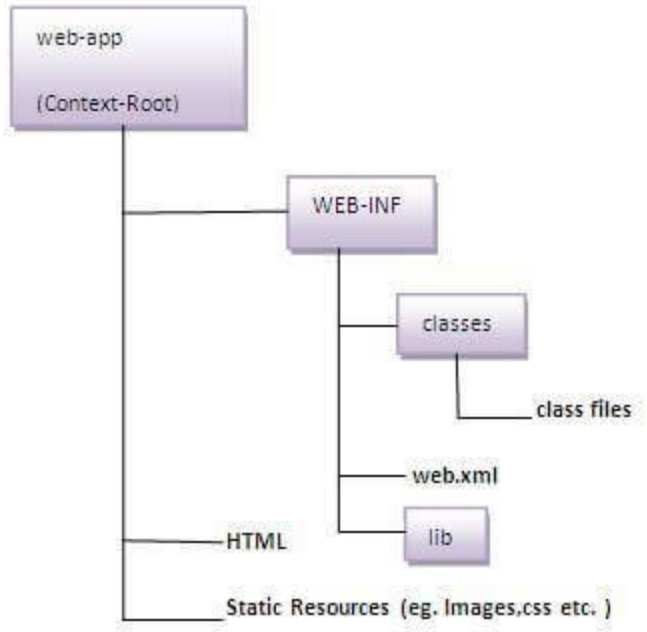
The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

GenericServlet class implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.GenericServlet class can handle any type of request so it is protocol-independent.

The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Directory structure:



How Servlet works?

The server checks if the servlet is requested **for the first time**.

If yes, web container does the following tasks:

- loads the servlet class.
- instantiates the servlet class.
- calls the init method passing the ServletConfig object

Else : calls the service method passing request and response objects

The web container calls the destroy method when it needs to remove the servlet such as at time of stopping the server or undeploying the project.

The public service method converts the ServletRequest object into the HttpServletRequest type and ServletResponse object into the HttpServletResponse type. Then, calls the service method passing these objects.

The protected service method checks the type of request, if request type is get, it calls doGet method, if request type is post, it calls doPost method

Session simply means a particular interval of time.

Session Tracking is a way to maintain the state (data) of an user. It is also known as **session management** in servlet.

Each time a user requests to the server, the server treats the request as the new request. So we need to maintain the state of a user to recognize a particular user.

There are four techniques used in Session tracking:

1. **Cookies**

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

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.

2. **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.

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

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

3. **URL Rewriting**

   In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource.. When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. We can send parameter name/value pairs using the following format:

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

4. **HttpSession**

   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.
3. **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
4. **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.