Java Servlets

Introduction

Web Application is a piece of software with dynamic functionality on the server. Google, Facebook, Twitter are some popular examples of web applications.

Server-side programming languages such as Java Servlets, Java Server Pages, ASP.Net and PHP etc. can be used to create Dynamic Web Applications.

Servlets are java classes that run on the Java-enabled web server and are widely used for web processing as well as are capable of handling complex requests obtained from the web server.

Servlets need some protocol and methods for communicating with the server.

Java Servlet

A servlet is also a java class that is descended from the class *javax.servlet.http.HttpServlet*.

Java Servlets are small, platform-independent java programs that run on the Java-enabled web server.

The Servlet extends either GenericServlet or HttpServlet, overriding the appropriate methods, so it handles requests.

Servlets are executed within the address space of a Web Server.

Java Servlet technology defines HTTP-specific servlet classes. A HTTP Servlet runs under the HTTP protocol.

This HTTP protocol is an asymmetrical request-response protocol where the client sends a request message to the server, and the server returns a response for requested data.

HTTP PROTOCOL

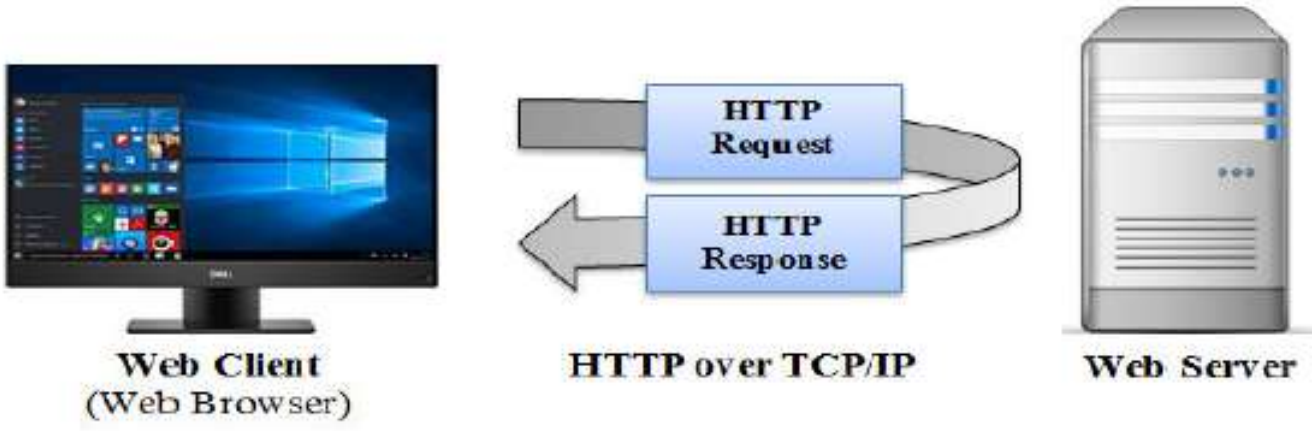
The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.

HTTP is a TCP/IP based communication protocol.

This is the foundation for data communication for the World Wide Web.

HTTP is connectionless, HTTP is media independent and HTTP is stateless.

HTTP works on the basis of Request-Response Model in Client-Server Architectu



HTTP Methods

HTTP methods are HTTP requests that indicates what actions the server should perform. These methods allow for a much richer communication between the client and the server. These are the common HTTP methods: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS

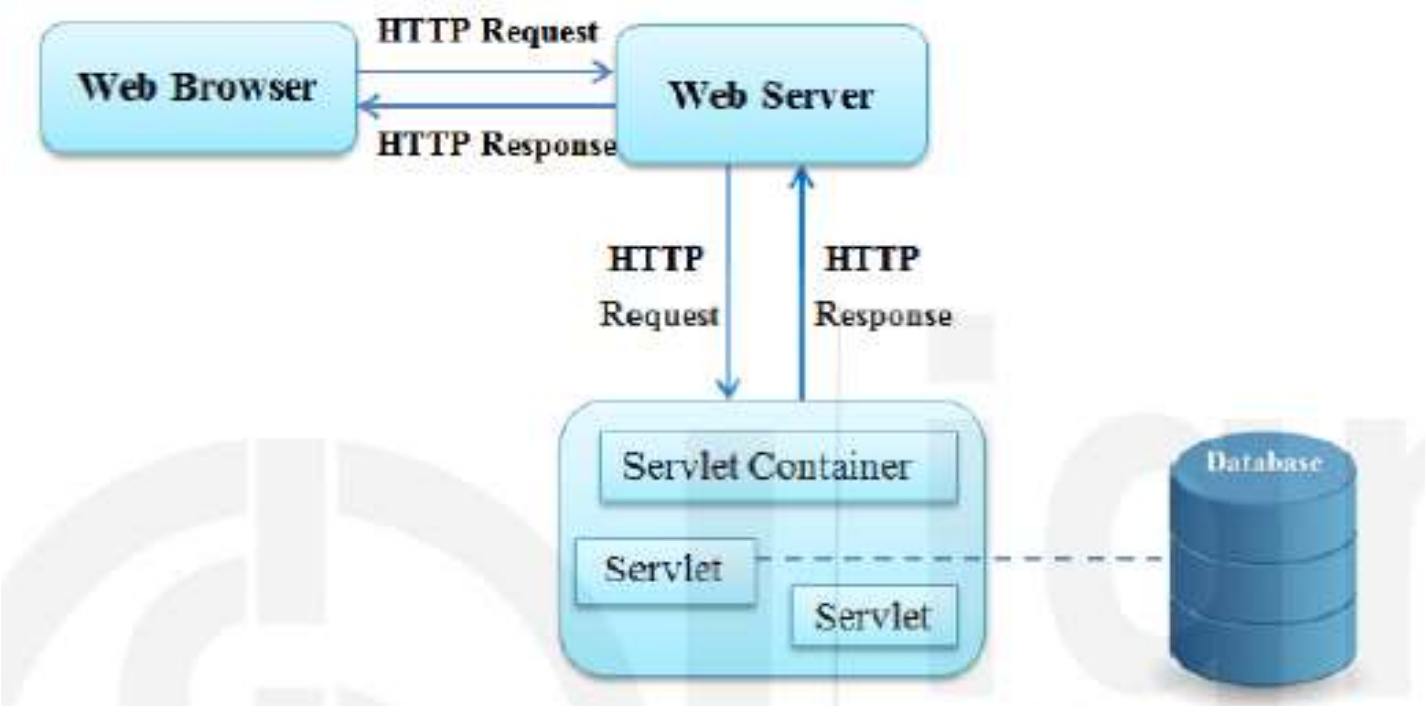
Whenever a client sends a message to a server it is an HTTP request. When the client sends a request to the server, the server returns a response to the client. Web Client sends a request specifying one of the seven HTTP request methods, the location of the resource to be invoked, protocol version, a set of optional headers and an optional message body.

GET /studentzone/hallticket.jsp HTTP/1.1

Get Vs. Post

QUESTION	GET	POST
Can request be bookmarked?	Yes	No
Can request be cached?	Yes	No
Does request stay in browser history?	Yes	No
Is the data visible to the user?	Yes, parameters are visible in the URL.	No
What happens when a browser request is refreshed?	Harmless	Data will be re-submitted and browser will prompt confirmation for re-submission.
What are the encoding types?	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data (for binary data)
Any restrictions on data length?	Yes. The data is added to URL as name and value pairs. The max URL length is 2048 characters	No length restrictions
Any restrictions on data type?	Only ASCII characters are allowed	No restrictions.
How secure are these requests?	GET is less secure then POST since the data is part of the URL. GET should not be used for sending sensitive data (e.g. passwords, credit card information).	POST is safer than GET because the data is not visible to the user. Also, it is not stored in browser history or web server logs.

Servlet Architecture



Servlets are the Java program that runs on the Java-enabled web server. The client or web browser sends the HTTP request to the web server. The web server receives the request and passes the request to the servlet container and subsequently the servlet container forwards this request to the corresponding servlet. The servlet processes the request and generates the response in the form of output. This process may require communication to a database and may also invoke a web service, or computing the response directly. After processing, the servlet builds the response object and sends it back to the Web Container.

Writting Servlets

You need to use Servlet API to create servlets. These APIs allow developer to build programs that can run with a web server. The Servlet APIs (Application Programming Interface) contains two packages: javax.servlet and javax.servlet.http. These two packages make up the servlet architecture. The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. The javax.servlet package contains generic interfaces and classes that are implemented and extended by all the servlet. The javax.servlet.http package contains a number of classes and interfaces that are used to create HTTP protocol-specific Servlets.

A Servlet can be created by three ways:

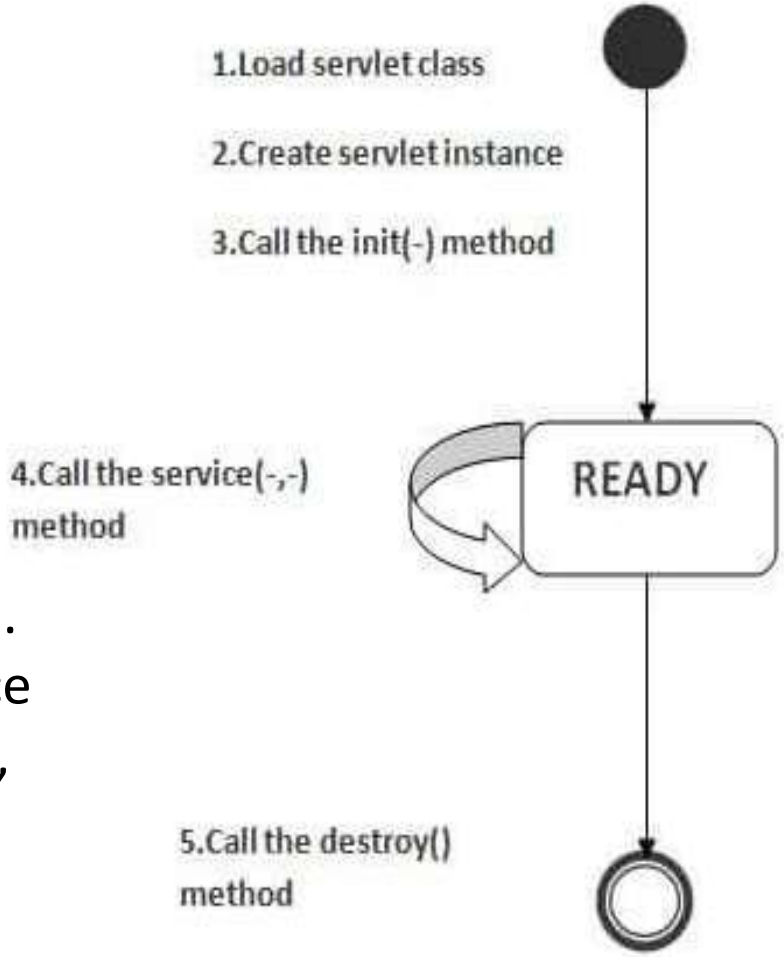
* By implementing Servlet interface,
* By inheriting GenericServlet class, (or)
* By inheriting HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead(), etc.

Servlet Life Cycle

The web container maintains the life cycle of a servlet instance.

The life cycle has following stages:
* Servlet class is loaded.
* Servlet instance is created.
* init() method is invoked.
* service() method is invoked.
* destroy() method is invoked.



There are three states: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

An Example Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class WelcomeServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,  HttpServletResponse response)
  throws ServletException, IOException
  {
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>Servlet Testing</TITLE>");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println("Welcome to the IGNOU Family!!");
    out.println("</BODY></HTML>");
  }
}
```

Thank You….