Servlet

Servlets are server-side Java program modules that process and answer client requests and implement the servlet interface directly or indirectly.

A servlet is a small program that runs on a server.

Since a servlet is integrated with the Java language, it also possesses all the Java features such as high portability, platform independence, security and Java database connectivity.

A servlet acts as an intermediary between the client and the server

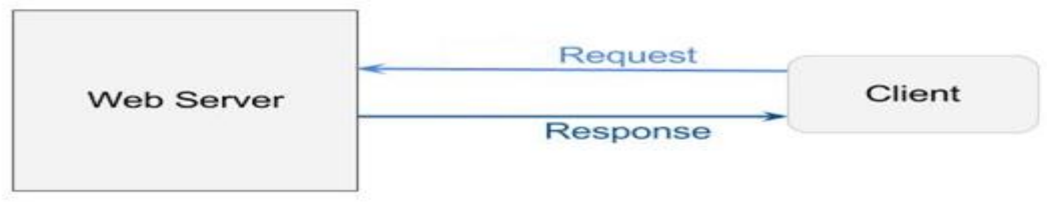
A servlet is capable of receiving the request from the client(browser) and processes it and also build the response for the client.

A servlet cannot be executed by JRE because it does not have a main method. The Web-server provides a run time environment to run the servlet program.

Servlet is an interface defined in **javax.servlet** package

Web Server

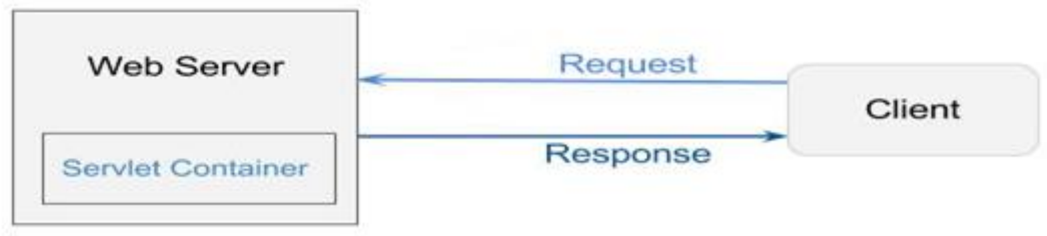
To know what is a Servlet container, we need to know what is a Web Server first.



A *web server* uses HTTP protocol to transfer data. When, a user type in a URL (e.g. www.jspider.com/static.html) in browser (a client), he get a web page to read. So what the server does is sending a web page to the client. The transformation is in HTTP protocol which specifies the format of request and response message. If the web server could not find the resource requested from the user than the server rises the 404 (file not found) error. Web servers are good in serving static pages.

Servlet Container

As we see here, the user/client can only request static webpage from the server. This is not good enough, if the user wants to read the web page based on his input or if he needs dynamic pages. The basic idea of Servlet container is using Java to dynamically generate the web page on the server side. So servlet container is essentially a part of a web server that interacts with the servlets.



Servlet container is the container for Servlets.

Web container

Web container (also known as a Servlet container) is the component of a web server that interacts with Java servlets. A web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

Advantages of servlet container

Communication Support servlet container helps server and servlet to communicate with each other.

Multithreading support container automatically creates new thread for every request.

Declarative support with the web.xml which is used by the servlet container we can configure & modify security & also other parameters without changing anything in the source code.

Life cycle management The container controls life and death of servlets

JSP support container takes care of translating JSP into a servlet.

Different between web server and application server?

A **web server** responsibility is to handler HTTP requests from client browsers and respond with HTML response. A web server understands HTTP language and runs on HTTP protocol.
Apache Web Server is kind of a web server and then we have specific containers that can execute servlets and JSPs known as servlet container, for example Tomcat.

Application Servers provide additional features such as Enterprise JavaBeans support, JMS Messaging support, Transaction Management etc. So we can say that Application server is a web server with additional functionalities to help developers with enterprise applications.

URL

Uniform Resource Locator (URL) it is the global address of documents and other resources on the World Wide Web.

<Protocol> :// <domain or IP> : port / path ? query_String (data) .

Protocol : if one application need to communicate with other (in our case browser & server) their needs to be a common language which both application should understand and that language should have set of rules. In software this language is a protocol, where protocol is a set of rules.

- The most commonly used protocols are HTTP and HTTPS
- Some other protocols are FTP, SFTP, SMTP , POP, TCP and ect...
- HTTPS is a secured protocol because hear the data is encrypted.

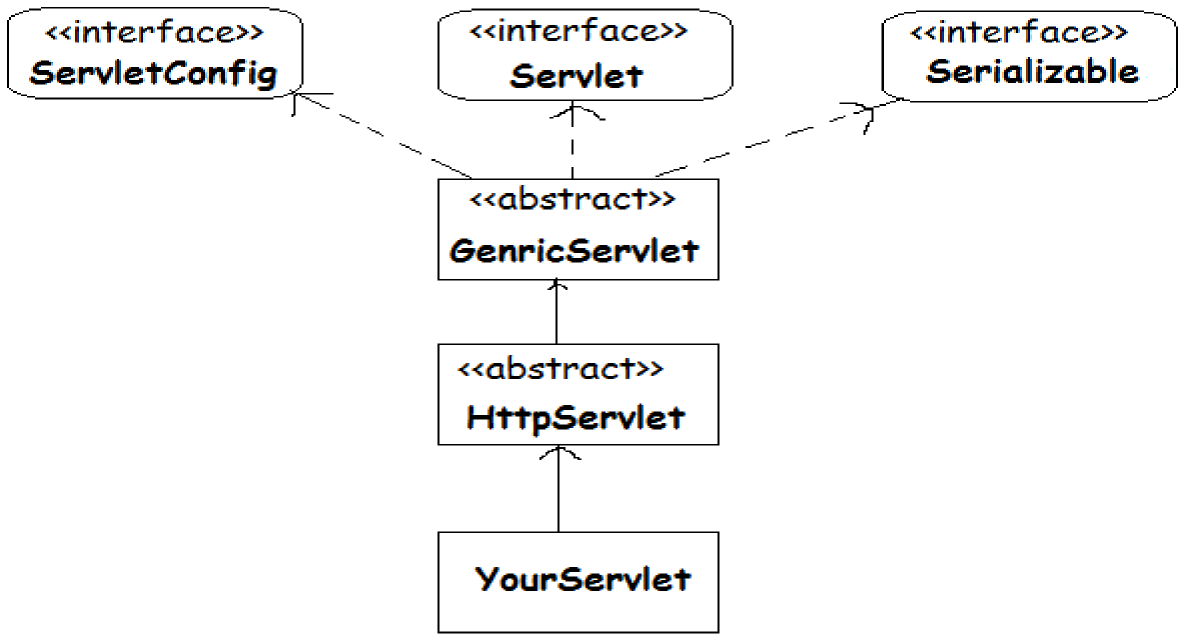
The HTTP protocol always run on the top of TCP/IP

TCP (transmition control protocol) is responsible making sure that data sent from one n/w to another n/w node ends has a complete data at destination , even though the data is split into multiple parts it is sent.

IP (internet protocol) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. Its role has been characterized as follows: "A name indicates what we seek. An address indicates where it is. A route indicates how to get there."

Servlet Hierarchy

A servlet is an object that extends either the javax.servlet.GenericServlet class or the javax.servlet.http.HttpServlet class. If the class extends GenericServlet class, than it is called has protocol independent servlet (such that it can make use of any protocol). If the class extends HttpServlet class, than it is called has protocol dependent servlet (such that it can make use of only HTTP & HTTPS protocol).



The **Servlet interface** is the root interface of the servlet class hierarchy. All Servlets need to either directly or indirectly implement the Servlet interface. The **GenericServlet** class of the **Servlet API** implements the Servlet interface. In addition to the Servlet interface, the GenericServlet class implements the **ServletConfig interface of the Servlet API** and the Serializable interface of the standard java.io. package

The servlet interface of the **javax.servlet** package defines methods that the Web container calls to manage the servlet life cycle. The following table describes various methods of the **javax.servlet.Servlet** interface

Method	**Description**
public void init(ServletConfig config) throws ServletException	The Web container calls this method after creating a servlet instance.
public void service(ServletRequest req , ServletResponse resp)	Called by the servlet container to allow the servlet to respond to a request
public void destroy()	The Web container calls the destroy() method just before removing the servlet instance from service.
public String getServletInfo()	Returns a string that contains information about the servlet, such as author, version, and copyright.
public ServletConfig getServletConfig()	Returns a ServletConfig object that contains configuration information, such as initialization parameters, to initialize a servlet.

The **javax.servlet.ServletConfig** interface is implemented by a Web container to pass configuration information to a servlet, during initialization of a servlet. A servlet is initialized by passing an object of ServletConfig to its init() method by the Web container. The ServletConfig object contains initialization information and provides access to the ServletContext object.

Initialization parameters are name value pairs that we can use to pass information to a servlet. The following table lists some of the methods of the javax.servlet.ServletConfig interface:

Method	**Description**
public String getInitParameter(String param)	Returns a String containing the value of the specified initialization parameters or null if the parameter does not exist.

public Enumeration getInitParameterNames()	Returns the names of all initialization parameters as an Enumeration of String objects. If no initialization parameters have been defined, an empty Enumeration is returned.
public ServletContext getServletContext()	Returns the ServletContext object for the servlet, which allows interaction with the Web container.

GenericServlet

GenericServlet is a abstract class. Hear the service method is the abstract method. Any class that extends the GenericServlet, that child class should override the service method. The GenericServlet class defines a protocol-independent(HTTP-less) servlet.

As you can see it is an abstract class and it extends Object class & implements Servlet, ServletConfig and Serializable interfaces.

Methods of GenericServlet class:

- **void destroy()**: It is called by servlet container to indicate that the servlet life cycle is finished.
- **java.lang.String getInitParameter(java.lang.String name)**: It returns the value of the specified initialization parameter. If the parameter does not exist then it returns null.
- **java.util.Enumeration getInitParameterNames()**: This method returns all the initialization parameters in form of a String enumeration. By using enumeration methods we can fetch the individual parameter names.
- **ServletConfig getServletConfig()**: It returns this servlet's ServletConfig object.
- **ServletContext getServletContext()**: It returns a reference to the ServletContext in which this servlet is running.
- **java.lang.String getServletInfo()**: It returns information about the servlet, such as author, version, and copyright.
- **java.lang.String getServletName()**: This method returns the name of this servlet instance.
- **void init()**: This is the first method of servlet life cycle and it is used for initializing a servlet.
- **void init(ServletConfig config)**: Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
- **abstract void service(ServletRequest req, ServletResponse res)**: It is called by the servlet container to allow the servlet to respond to the client's request.

Servlet life cycle

The life cycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the **following steps**

- Loading a servlet and Instantiation
- Initialising the servlet
- Request handling
- Destroying the servlet

Loading a servlet

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. **Instantiation** 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.

Initialising

The web container calls the init method only once after creating the servlet instance. The **init method** is used to initialize the servlet.

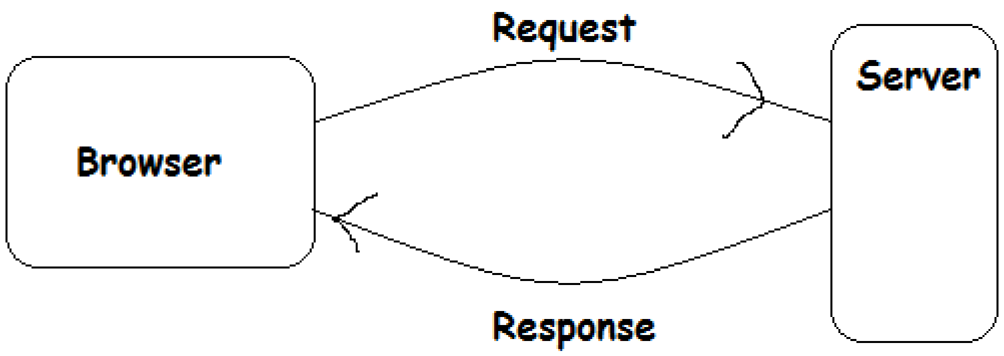
Request handling

The web container calls the service method each time when request for the servlet is received. The web container creates the object of Request and Response and calls the service method. 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**. the servlet is initialized only once.

Destroying the servlet

The web container calls the **destroy method** before removing the servlet instance from the service.

Interaction of browser and server



The browser(client) send the request to the server, the server is going to respond back for the browser(client) for the specified request.

Key elements of request are

URL
Form data
Method Type

Key elements of response are

Status code
Content Type
Actual Content

Status codes HTTP status codes are standard response codes given by web site servers on the Internet. The codes help identify the cause of the problem when a web page or other resource does not load properly. If the status code is in the range of 4XX than it is a client side error. If the status code is int the range of 5XX than it is a server side error.

Eg:

200	Standard response for successful HTTP requests
404	The requested resource could not be found
405	request method not supported by that resource
500	Internal Server Error

Content Type Content type or sometime referred has MIME type(multipurpose internet mailed extension) tells the browser that what type of content it is going to send so that the browser can be prepared itself to handle the data.

Eg:

text/html
text/pdf
image/jpeg

Actual content this information will be provided by either server or developer depending on the scenario. Its a mandatory information present in the body on http responce.

URL every resource(static/dynamic) in the web has unique address in the form of URL. Its a mandatory information present in http request header.

Form data in the html the html forms are used to capture information from user. What ever the information user provides via html form is called has form data. Its a optional information present it will be present in the either body or in the header of the http request depending on the http method in the request.

Method type its a mandatory information present on the header of request. It indicates which method need to be invoked to satisfy the resource **. 99.99%** of times its **get and post**

Deployment Descriptors

A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests. When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request.

The deployment descriptor is a file named web.xml. It resides in the app's WAR under the folder by name WEB-INF/ directory. The file is an XML file whose root element is <web-app>.

web.xml

This is the configuration file of web applications in java. It instructs the servlet container (tomcat for ex.) which classes to load, what parameters to set in the context & config, and how to intercept requests coming from browsers. The web.xml file is the deployment descriptor for a Servlet-based Java web application. The web server uses this configuration to identify the servlet to handle a given request and call the class method that corresponds to the request method.

Has soon has the server is started it is going to parse the web.xml files, present in the application. If any thing is wrong in the deployment of any server it troughs a parseException.

Note

- Any class which extends GenericServlet or HttpServlet is called as servlet
- A servlet can have its own methods, block of code(static/non-static) inner-class etc, but shouldnot have abstract method
- Servlet should not have private constructor
- If we have a class that extends GenericServlet or HttpServlet, than the sub class of that class is also called has servlet
- At any point of view only one instance of the servlet exists. Hence servlet is singleton in nature
- Every request to a servlet runs in separate thread
- HttpServlet class can be empty we don't get any compile time error or runtime exception
- Servlet class can contain main method but its of no use

HttpServlet

HTTPServlet is an abstract class with all implemented methods. The HttpServlet class extends GenericServlet class. HttpServlet is a class present in the **javax.servlet.http** package. The HttpServlet class have the implementation to all the methods listed below. And these are the HttpServlet class methods.

protected void	**doDelete**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a DELETE request. This allow the client to delete the file from server
protected void	**doGet**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a GET request. The get request don't have body hence data sent using this request will be present in the header of Http request.
protected void	**doHead**(HttpServletRequest req, HttpServletResponse resp) Receives an HTTP HEAD request from the protected service method and handles the request. It allows the client to see only headers
protected void	**doOptions**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a OPTIONS

	request. This method determines which http method that server supports.
protected void	**doPost**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a POST request. The post request have a body hence the data sent using the post will be present inside the body of http request.
protected void	**doPut**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a PUT request. This method allows the client to place the file in a server and the operation is similar to sending the file using FTP.
protected void	**doTrace**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a TRACE request. In the sence this method echos back the recived request so that a client can see changes(if any) happened to the request
protected long	**getLastModified**(HttpServletRequest req) Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT.
protected void	**service**(HttpServletRequest req, HttpServletResponse resp) Receives standard HTTP requests from the public service method and dispatches them to the do*XXX* methods defined in this class.
void	**service**(ServletRequest req, ServletResponse res) Dispatches client requests to the protected service method.

Depending on the type of method specified in the http request the corresponding http method is going to be invoked. 99.99% of time we are going to override only **doGet( )** and **doPost( )** methods. To call the doPost( ) method of the HttpServelt class you need to specify **method=post** in the html form tag. Similarly to call doGet( ) method=get, but by default it is going to invoke doGet( ) method always.

- A sub class of HttpServlet can override any of the following methods
  service(ServletRequest req, ServletResponse res) **OR**
  service(HttpServletRequest req, HttpServletResponse resp) **OR**
  doXXX(HttpServletRequest req, HttpServletResponse resp)

- There is no need to override the service method if we extend the HttpServlet class, it has been already overridden inside HttpServlet class. The service method present in HttpServlet class handles http request and transfer the control to the respective doxxx( ) method.
- If we don't over ride any methods in the child class of HttpServlet there is no compile time error or run time error.
- HttpServlet class is a abstract class but there are no abstract methods inside it, so we cant use this class has over servlet class. So we have to create a class which extends Httpservlet class and we should make use of this child class has our servlet.

Difference between GenericServlet and HttpServlet

GernericServlet	**HttpServlet**
Is protocol independent	Is protocol dependent
Is a abstract class where service( ) method is abstract	Is also abstract class but non of the methods are abstract methods.
If we extend GenricServlet than we need to override the service method.	If we extend HttpServlet there no force or restriction to override any methods.
GenericServlet class don't extend any class	HttpServlet class extends Generic servlet
It belongs to **javax.servlet** package	It belongs to **javax.servlet.http** package

Difference between get & post methods

GET / doGet( )	**POST / doPost( )**
Get request don't have a body	Post request has a body
Its a default method	Its not a default method, we have to explicitly define method=post in the form tag
Data sent using get will be present in the header of http request in the form of query string	Data will be present in the body of http request
Its not secure	Its secure
The amount of data sent using the get is restricted(only 1024 characters)	There is no restricting on the amount of data sent using post
We cant send the file using get	We can send the file using post
Get request are idempotent in nature, i.e we can perform same operation without any side effect	Not idempotent
We can book mark the get request	We cant book mark the post request