**Servlet Tutorial**

[**What is Servlet** [open link](https://www.javatpoint.com/servlet-tutorial)](https://www.javatpoint.com/servlet-tutorial)[Web Terminology](https://www.javatpoint.com/web-terminology)

[Servlet API](https://www.javatpoint.com/servlet-api)

[Servlet Interface](https://www.javatpoint.com/Servlet-interface)

[GenericServlet](https://www.javatpoint.com/GenericServlet-class)

[HttpServlet](https://www.javatpoint.com/HttpServlet-class)

[Servlet Life Cycle](https://www.javatpoint.com/life-cycle-of-a-servlet)

[Servlet Example](https://www.javatpoint.com/steps-to-create-a-servlet-using-tomcat-server)

[How servlet works?](https://www.javatpoint.com/how-servlet-works)

[War File](https://www.javatpoint.com/war-file)

[welcome-file-list](https://www.javatpoint.com/welcome-file-list)[Load on startup](https://www.javatpoint.com/load-on-startup)

[**Servlet with IDE**](https://www.javatpoint.com/creating-servlet-in-eclipse-ide)

[servlet in Eclipse](https://www.javatpoint.com/creating-servlet-in-eclipse-ide)

[servlet in Myeclipse](https://www.javatpoint.com/creating-servlet-in-myeclipse-ide)

[servlet in Netbeans](https://www.javatpoint.com/creating-servlet-in-netbeans-ide)

[**ServletRequest**](https://www.javatpoint.com/servletrequest)

[**Servlet Collaboration**](https://www.javatpoint.com/requestdispatcher-in-servlet)

[RequestDispacher](https://www.javatpoint.com/requestdispatcher-in-servlet)

[sendRedirect](https://www.javatpoint.com/sendRedirect()-method)

[**ServletConfig**](https://www.javatpoint.com/servletconfig)

[**ServletContext**](https://www.javatpoint.com/servletcontext)

[**Attribute in Servlet**](https://www.javatpoint.com/attribute)

**Servlet Advance**

[**Session Tracking**](https://www.javatpoint.com/session-tracking-in-servlets)

[Session Techniques](https://www.javatpoint.com/session-tracking-in-servlets)

[1) Cookies in Servlet](https://www.javatpoint.com/cookies-in-servlet)

[Cookies: Login & Logout](https://www.javatpoint.com/servlet-login-and-logout-example-using-cookies)

[2) Hidden Form Field](https://www.javatpoint.com/hidden-form-field-in-session-tracking)

[3) URL Rewriting](https://www.javatpoint.com/url-rewriting-in-session-tracking)

[4) HttpSession](https://www.javatpoint.com/http-session-in-session-tracking)

[Session: Login & Logout](https://www.javatpoint.com/servlet-http-session-login-and-logout-example)

[**Event and Listener**](https://www.javatpoint.com/Event-and-Listener-in-Servlet)

[**Servlet Filter**](https://www.javatpoint.com/servlet-filter)

[What is Filter](https://www.javatpoint.com/servlet-filter)

[Authentication Filter](https://www.javatpoint.com/authentication-filter)

[FilterConfig](https://www.javatpoint.com/filter-config)

[Useful examples](https://www.javatpoint.com/filter2)

[**Servlet CRUD**](https://www.javatpoint.com/crud-in-servlet)

[CRUD Example](https://www.javatpoint.com/crud-in-servlet)

[**Servlet Pagination**](https://www.javatpoint.com/pagination-in-servlet)

[Pagination Example](https://www.javatpoint.com/pagination-in-servlet)

[**Servlet Miscellaneous**](https://www.javatpoint.com/ServletInputStream-class)

[ServletInputStream](https://www.javatpoint.com/ServletInputStream-class)[ServletOutputStream](https://www.javatpoint.com/ServletOutputStream-class)[Annotation Servlet](https://www.javatpoint.com/servlet-with-annotation)[SingleThreadModel](https://www.javatpoint.com/SingleThreadModel-interface)[SSI](https://www.javatpoint.com/server-side-include)

[**Development**](https://www.javatpoint.com/example-of-registration-form-in-servlet)

[Registration Example](https://www.javatpoint.com/example-of-registration-form-in-servlet)

[Fetching records](https://www.javatpoint.com/example-of-fetching-result-for-the-given-rollno-in-servlet)

[Improving Performance](https://www.javatpoint.com/improving-servlet-performance-to-fetch-records)

[Uploading file](https://www.javatpoint.com/example-of-uploading-file-to-the-server-in-servlet)

[Downloading file](https://www.javatpoint.com/example-of-downloading-file-from-the-server-in-servlet)

[Servlet Sending Email](https://www.javatpoint.com/sending-email-through-java-mail-api-in-servlet)

[Write data to PDF](https://www.javatpoint.com/how-to-write-data-into-PDF-using-servlet)

[Login Example](https://www.javatpoint.com/example-of-login-form-in-servlet)

[Writing Image](https://www.javatpoint.com/example-to-display-image-using-servlet)

# Servlets | Servlet Tutorial

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

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

## What is a Servlet?

Servlet can be described in many ways, depending on the context.

* Servlet is a technology which is used to create a web application.
* Servlet is an API that provides many interfaces and classes including documentation.
* Servlet is an interface that must be implemented for creating any Servlet.
* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.



Do You Know?

* What is the web application and what is the difference between Get and Post request?
* What information is received by the web server if we request for a Servlet?
* How to run servlet in Eclipse, MyEclipse and Netbeans IDE?
* What are the ways for servlet collaboration and what is the difference between RequestDispatcher and sendRedirect() method?
* What is the difference between ServletConfig and ServletContext interface?
* How many ways can we maintain the state of a user? Which approach is mostly used in web development?
* How to count the total number of visitors and whole response time for a request using Filter?
* How to run servlet with annotation?
* How to create registration form using Servlet and Oracle database?
* How can we upload and download the file from the server?

### 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).

### Advantages of Servlet



There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) manages Servlets, so we don't need to worry about the memory leak, [garbage collection](https://www.javatpoint.com/Garbage-Collection), etc.
4. **Secure:** because it uses java language.

# Class javax.servlet.http.HttpServlet

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of HttpServlet must override at least one method, usually one of these:

1. doGet, if the servlet supports HTTP GET requests
2. doPost, for HTTP POST requests
3. doPut, for HTTP PUT requests
4. doDelete, for HTTP DELETE requests
5. init and destroy, to manage resources that are held for the life of the servlet
6. getServletInfo, which the servlet uses to provide information about itself

# Content Type

Content Type is also known as **MIME (Multipurpose internet Mail Extension)**Type. It is a **HTTP header** that provides the description about what are you sending to the browser.

MIME is an internet standard that is used for extending the limited capabilities of email by allowing the insertion of sounds, images and text in a message.

The features provided by MIME to the email services are as given below:

List of Content Types

There are many content types. The commonly used content types are given below:

* text/html
* text/plain
* application/msword
* application/vnd.ms-excel
* application/jar
* application/pdf
* application/octet-stream
* application/x-zip
* images/jpeg
* images/png
* images/gif
* audio/mp3
* video/mp4

[**next →**](https://www.javatpoint.com/get-vs-post)[**← prev**](https://www.javatpoint.com/http)

HTTP Requests

The request sent by the computer to a web server, contains all sorts of potentially interesting information; it is known as HTTP requests.

The HTTP client sends the request to the server in the form of request message which includes following information:

* The Request-line
* The analysis of source IP address, proxy and port
* The analysis of destination IP address, protocol, port and host
* The Requested URI (Uniform Resource Identifier)
* The Request method and Content
* The User-Agent header
* The Connection control header
* The Cache control header

# Servlet API

1. [Servlet API](https://www.javatpoint.com/servlet-api)
2. [Interfaces in javax.servlet package](https://www.javatpoint.com/servlet-api#servletapi1)
3. [Classes in javax.servlet package](https://www.javatpoint.com/servlet-api#servletapi2)
4. [Interfaces in javax.servlet.http package](https://www.javatpoint.com/servlet-api#servletapi3)
5. [Classes in javax.servlet.http package](https://www.javatpoint.com/servlet-api#servletapi4)

The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.

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.

Let's see what are the interfaces of javax.servlet package.

### Interfaces in javax.servlet package

There are many interfaces in javax.servlet package. They are as follows:

1. Servlet
2. ServletRequest
3. ServletResponse
4. RequestDispatcher
5. ServletConfig
6. ServletContext
7. SingleThreadModel
8. Filter
9. FilterConfig
10. FilterChain
11. ServletRequestListener
12. ServletRequestAttributeListener
13. ServletContextListener
14. ServletContextAttributeListener

### Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletRequestWrapper
5. ServletResponseWrapper
6. ServletRequestEvent
7. ServletContextEvent
8. ServletRequestAttributeEvent
9. ServletContextAttributeEvent
10. ServletException
11. UnavailableException

### Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. They are as follows:

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession
4. HttpSessionListener
5. HttpSessionAttributeListener
6. HttpSessionBindingListener
7. HttpSessionActivationListener
8. HttpSessionContext (deprecated now)

### Classes in javax.servlet.http package

There are many classes in javax.servlet.http package. They are as follows:

1. HttpServlet
2. Cookie
3. HttpServletRequestWrapper
4. HttpServletResponseWrapper
5. HttpSessionEvent
6. HttpSessionBindingEvent
7. HttpUtils (deprecated now)

# Servlet Interface

1. [Servlet Interface](https://www.javatpoint.com/Servlet-interface)
2. [Methods of Servlet interface](https://www.javatpoint.com/Servlet-interface#servletmethods)

**Servlet interface provides** commonbehaviorto all the servlets.Servlet interface defines methods that all servlets must implement.

Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

### Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public void init(ServletConfig config)** | initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once. |
| **public void service(ServletRequest request,ServletResponse response)** | provides response for the incoming request. It is invoked at each request by the web container. |
| **public void destroy()** | is invoked only once and indicates that servlet is being destroyed. |
| **public ServletConfig getServletConfig()** | returns the object of ServletConfig. |
| **public String getServletInfo()** | returns information about servlet such as writer, copyright, version etc. |

### Servlet Example by implementing Servlet interface

Let's see the simple example of servlet by implementing the servlet interface.

### It will be better if you learn it after visiting steps to create a servlet.

*File: First.java*

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
4. **public** **class** First **implements** Servlet{
5. ServletConfig config=**null**;
7. **public** **void** init(ServletConfig config){
8. **this**.config=config;
9. System.out.println("servlet is initialized");
10. }
12. **public** **void** service(ServletRequest req,ServletResponse res)
13. **throws** IOException,ServletException{
15. res.setContentType("text/html");
17. PrintWriter out=res.getWriter();
18. out.print("<html><body>");
19. out.print("<b>hello simple servlet</b>");
20. out.print("</body></html>");
22. }
23. **public** **void** destroy(){System.out.println("servlet is destroyed");}
24. **public** ServletConfig getServletConfig(){**return** config;}
25. **public** String getServletInfo(){**return** "copyright 2007-1010";}
27. }

# GenericServlet class

1. [GenericServlet class](https://www.javatpoint.com/GenericServlet-class)
2. [Methods of GenericServlet class](https://www.javatpoint.com/GenericServlet-class#genericmethods)
3. [Example of GenericServlet class](https://www.javatpoint.com/GenericServlet-class)

**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.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

### Methods of GenericServlet class

There are many methods in GenericServlet class. They are as follows:

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9. **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

### Servlet Example by inheriting the GenericServlet class

Let's see the simple example of servlet by inheriting the GenericServlet class.

### It will be better if you learn it after visiting steps to create a servlet.

*File: First.java*

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
4. **public** **class** First **extends** GenericServlet{
5. **public** **void** service(ServletRequest req,ServletResponse res)
6. **throws** IOException,ServletException{
8. res.setContentType("text/html");
10. PrintWriter out=res.getWriter();
11. out.print("<html><body>");
12. out.print("<b>hello generic servlet</b>");
13. out.print("</body></html>");
14. }
15. }
16. }