

**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.
5. Web Terminology

|  |  |
| --- | --- |
| **Servlet Terminology** | **Description** |
| [Website: static vs dynamic](https://www.javatpoint.com/website-static-vs-dynamic) | It is a collection of related web pages that may contain text, images, audio and video. |
| [HTTP](https://www.javatpoint.com/http) | It is the data communication protocol used to establish communication between client and server. |
| [HTTP Requests](https://www.javatpoint.com/http-requests) | It is the request send by the computer to a web server that contains all sorts of potentially interesting information. |
| [Get vs Post](https://www.javatpoint.com/get-vs-post) | It gives the difference between GET and POST request. |
| [Container](https://www.javatpoint.com/container) | It is used in java for dynamically generating the web pages on the server side. |
| [Server: Web vs Application](https://www.javatpoint.com/server-web-vs-application) | It is used to manage the network resources and for running the program or software that provides services. |
| [Content Type](https://www.javatpoint.com/content-type) | It is HTTP header that provides the description about what are you sending to the browser. |

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. |

1. **import** java.io.IOException;
2. **import** java.io.PrintWriter;
3. **import** javax.servlet.ServletException;
4. **import** javax.servlet.SingleThreadModel;
5. **import** javax.servlet.http.HttpServlet;
6. **import** javax.servlet.http.HttpServletRequest;
7. **import** javax.servlet.http.HttpServletResponse;
9. **public** **class** MyServlet **extends** HttpServlet **implements** SingleThreadModel{
10. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
11. **throws** ServletException, IOException {
12. response.setContentType("text/html");
13. PrintWriter out = response.getWriter();
15. out.print("welcome");
16. **try**{Thread.sleep(10000);}**catch**(Exception e){e.printStackTrace();}
17. out.print(" to servlet");
18. out.close();
19. }
20. }

Annotations: @WebServlet,@WebInitParam,@WebFilter

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(value = "/Simple")

public class Simple extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("<html><body>");

out.print("<h3>Hello Servlet</h3>");

out.print("</body></html>");

}

}

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(value = "/Simple", initParams = {

@WebInitParam(name = "foo", value = "Hello "),

@WebInitParam(name = "bar", value = " World!")

})

public class Simple extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("<html><body>");

out.print("<h3>Hello Servlet</h3>");

out.println(getInitParameter("foo"));

out.println(getInitParameter("bar"));

out.print("</body></html>");

}

}

import java.io.IOException;

import javax.servlet.annotation.WebFilter;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.\*;

import java.util.\*;

// Implements Filter class

@WebFilter(urlPatterns = {"/\*"}, initParams = {

@WebInitParam(name = "test-param", value = "Initialization Paramter")})

public class LogFilter implements Filter {

public void init(FilterConfig config) throws ServletException {

// Get init parameter

String testParam = config.getInitParameter("test-param");

//Print the init parameter

System.out.println("Test Param: " + testParam);

}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

// Log the current timestamp.

System.out.println("Time " + new Date().toString());

// Pass request back down the filter chain

chain.doFilter(request,response);

}

public void destroy( ) {

/\* Called before the Filter instance is removed

from service by the web container\*/

}

}