Introduction to Java Servlets

Today we all are aware of the need of creating dynamic web pages i.e the ones which have the capability to change the site contents according to the time or are able to generate the contents according to the request received by the client. If you like coding in Java, then you will be happy to know that using Java there also exists a way to generate dynamic web pages and that way is Java Servlet. But before we move forward with our topic let's first understand the need for server-side extensions.

Servlets are the Java programs that runs 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, then send response back to the web server.

Properties of Servlets :

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from web server.

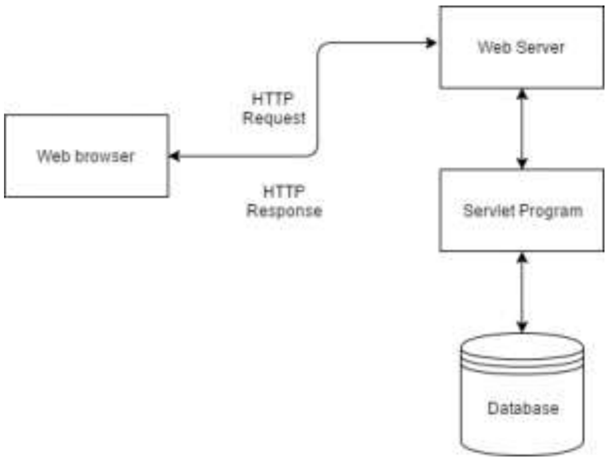
Execution of Servlets :

Execution of Servlets involves six basic steps:

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 web server.
6. The web server sends the response back to the client and the client browser displays it on the screen.

Servlet Architecture

The following diagram shows the servlet architecture:



Servlets API's:
Servlets are build from two packages:

- javax.servlet(Basic)
- javax.servlet.http(Advance)

Various classes and interfaces present in these packages are:

Component	**Type**	**Package**
Servlet	Interface	javax.servlet.*
ServletRequest	Interface	javax.servlet.*
ServletResponse	Interface	javax.servlet.*
GenericServlet	Class	javax.servlet.*
HttpServlet	Class	javax.servlet.http.*
HttpServletRequest	Interface	javax.servlet.http.*
HttpServletResponse	Interface	javax.servlet.http.*
Filter	Interface	javax.servlet.*
ServletConfig	Interface	javax.servlet.*

Advantages of a Java Servlet

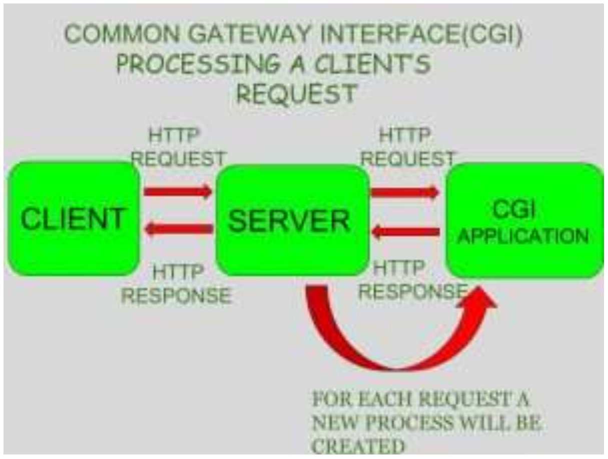
- Servlet is **faster** than CGI(**Common Gateway Interface**) as it doesn't involve the creation of a new process for every new request received.
- Servlets as written in Java are **platform independent**.
- Removes the overhead of creating a **new process** for each request as Servlet doesn't run in a separate process. There is only a single instance which handles all requests concurrently. This also saves the memory and allows a Servlet to easily manage client state.
- It is a server-side component, so Servlet inherits the **security** provided by the Web server.

- The **API** designed for Java Servlet automatically acquires the advantages of Java platform such as platform independent and portability. In addition, it obviously can use the wide range of APIs created on Java platform such as **JDBC** to access the database.

What is CGI

CGI is actually an external application which is written by using any of the programming languages like **C** or **C++** and this is responsible for processing client requests and generating dynamic content.
In CGI application, when a client makes a request to access dynamic Web pages, the Web server performs the following operations :

- It first locates the requested web page *i.e* the required CGI application using URL.
- It then creates a new process to service the client's request.
- Invokes the CGI application within the process and passes the request information to the server.
- Collects the response from CGI application.
- Destroys the process, prepares the HTTP response and sends it to the client.



So, in **CGI** server has to create and destroy the process for every request. It's easy to understand that this approach is applicable for handling few clients but as the number of clients increases, the workload on the server increases and so the time taken to process requests increases.

Difference between Servlet and CGI

Servlet	**CGI(Common Gateway Interface)**
Servlets are portable and efficient.	CGI is not portable
In Servlets, sharing of data is possible.	In CGI, sharing of data is not possible.
Servlets can directly communicate with the web server.	CGI cannot directly communicate with the web server.
Servlets are less expensive than CGI.	CGI are more expensive than Servlets.
Servlets can handle the cookies.	CGI cannot handle the cookies.

The Servlet Container

Servlet container, also known as **Servlet engine** is an integrated set of objects that provide run time environment for Java Servlet components.
In simple words, it is a system that manages Java Servlet components on top of the Web server to handle the Web client requests.

Services provided by the Servlet container :

- **Network Services :** Loads a Servlet class. The loading may be from a local file system, a remote file system or other network services. The Servlet container provides the network services over which the request and response are sent.
- **Decode and Encode MIME based messages :** Provides the service of decoding and encoding MIME-based messages.
- **Manage Servlet container :** Manages the lifecycle of a Servlet.
- **Resource management :** Manages the static and dynamic resources, such as HTML files, Servlets and JSP pages.
- **Security Service :** Handles authorization and authentication of resource access.
- **Session Management :** Maintains a session by appending a **session ID** to the URL path.

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.

File: First.java

```java
import java.io.*;
import javax.servlet.*;

public class First extends GenericServlet{
public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{

res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello generic servlet</b>");
out.print("</body></html>");

}
}
```

HttpServlet class

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

Methods of HttpServlet class

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

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.
2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.
6. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
7. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
8. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.
9. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
10. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.