Server-side Scripting

Server-side Scripting is responsible for creating the scripts on a web server that handles the responses made for a client’s request to the website. These scripts are written and used by back-end web developers to create the back-end software of a website that is also in control of the main usability and functionality of the website.

Server-side scripts run on a server and is written to interact with and process information from databases that the server can access. It is basically a direct line from the user to the database, playing a big role in how a database is built and managed. Scripts also manages the transfer of data from server to browser that can be usually observed through user requests and submission of forms. Scripts are also responsible for the functionality of dynamic web applications such as user validation, saving and retrieving data, and navigating through other pages. Lastly, it could also be used in building Application Programming Interfaces.

**Java Servlets**

Java Servlets are basically Java objects that are intended to play the role of server components in client-server communications. It handles client requests by generating responses to such requests. It has access to the entire family of Java APIS and is part of the Java EE Specification. Servlets are hosted in a “servlet container” that provides the environment in which the servlet runs, as well as controls the “servlet lifecycle”. Servlets are able to collect input from users through forms, retrieve and display data from a database and other sources, and create web pages dynamically.

Java Servlets often fulfill the same purpose of programs that implement the Common Gateway Interface (CGI), but offers more advantages compared to CGI by being significantly better in performance. With servlets, it is not necessary to allocate a separate process in handling each client request because it executes within the address space of the Web Server. By being written in Java, servlets are platform-independent and protected by the Java security manager. It can also communicate with applets, databases, or other software because the full functionality of the Java class libraries are available to servlets like sockets and RMI mechanisms.

*Servlet Tasks*

* Read and interpret explicit data, coming from an HTML form or web page, or applet, sent by the clients.
* Read and interpret implicit HTTP request data sent by the client like cookies and compression schemes.
* Process data gotten from databases and generating the results by executing RMI or CORBA, invoking a Web service, or computing the response directly.
* Send the explicit data

*Servlet Life Cycle*

This refers to the process that includes the creation of a servlet up to its destruction.

* To initialize the servlet, the **init()** method is invoked.
* To process a client’s request, the servlet calls the **service()** method.
* To terminate the servlet, the **destroy()** method is called.
* Lastly, servlet is garbage collected by the garbage collector of the JVM.

The **init()** Method

This method is only invoked once- when the servlet is created. When a user first invokes a URL corresponding to the servlet, the servlet is normally created. But, it could also be specified that the servlet is created once the server is first started.

The **service()** Method

This method is assigned to perform actual tasks like handling requests from the client (browsers) and to write the formatted response back to the client when the servlet container calls the service() method. This method is also responsible for checking the HTTP request type and calls its corresponding appropriate methods.

The **destroy()** Method

This method is invoked only once- at the end of the life cycle of a servlet. By calling this method, the servlet can close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other cleanup activities.

https://www.upwork.com/hiring/development/server-side-scripting-back-end-web-development-technology/