**Introduction to Servlets Quiz**

1. **What is the difference between a web server and a web container?**

A Web server refers to an execution infrastructure that handles HTTP requests and responses; a servlet container refers to a component that handles the lifecycle for servlets; an application server refers to a

framework (servlet container, EJB container, JSP engine, MQ container, etc.) for handling Web applications. A Web container as the part

of an application server that manages servlets, Java Server Pages (JSP)

files, and other Web-tier components. Some refer to a Web container as the infrastructure for managing the lifecycle for Web services.

1. **What is a servlet?**

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

1. **How do web servers and web containers interact with servlets?**

A web container (also known as a servlet container; and compare "web container") is the component of a web server that interacts with Jakarta 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.

A web container handles requests to servlets, Jakarta Server Pages (JSP) files, and other types of files that include server-side code. The Web container creates servlet instances, loads, and unloads servlets, creates, and manages request and response objects, and performs other servlet-management tasks.

1. **Who creates request objects?**

Servlet container.

1. **What are the states in the servlet lifecycle?**

States of a servlet are new, ready and end.

1. **Who calls init and when?**

Init is called by servlet container. The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards.

1. **Which of init, service, and doGet should you override?**

doGet.

1. **In what sense are servlets multi-threaded?**

Servlets are intrinsically multithreaded. This means a single instance can be accessed by more than one thread. If the container receives multiple requests for one servlet simultaneously, the service () method of that servlet will be executed concurrently in multiple threads. If a servlet implements the Single Thread Model interface, the container will not execute the service () method in more than one thread simultaneously.

The servlet container may synchronize access to a single instance of the servlet. However, servicing requests sequentially seriously hurts performance. To avoid the performance problem, a servlet container may create multiple instances of the servlet class.

1. **What are the implications of this for servlet instance variables?**

Instance variables are not thread safe because all threads share the same heap, and the heap is where instance variables are stored. This means that multiple threads can attempt to use the same instance variables concurrently and so thread safety cannot be guaranteed. There is also no certainty that a client accessing the same URL will even get the same instance variables. It is generally better to use session attributes in place of instance variables, but if for whatever reason you really have to use an instance variable then you will need to add synchronization to guarantee thread safety, which is of course a hit to performance.