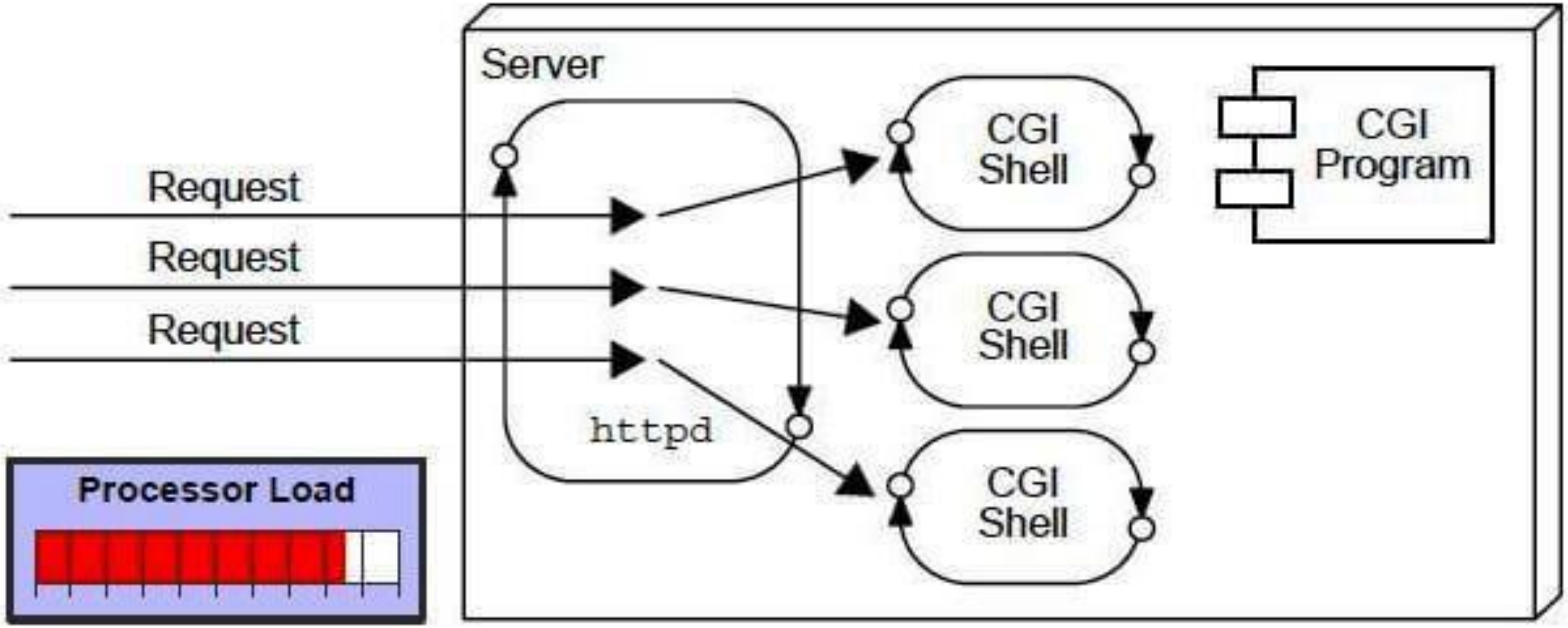
JAVA SERVLET TECHNOLOGY

PREPARED BY: MINAL MANIAR, ASSISTANT PROFESSOR, CSPIT, CHARUSAT

INTRODUCTION

- Shortly after the Web began to be used for delivering services, service providers recognized the need for dynamic content.
- Applets, one of the earliest attempts toward this goal.
- At the same time, developers also investigated using the server platform for the same purpose. Initially, Common Gateway Interface (CGI) server-side scripts were the main technology used to generate dynamic content.
- Although widely used, CGI scripting technology had many shortcomings.

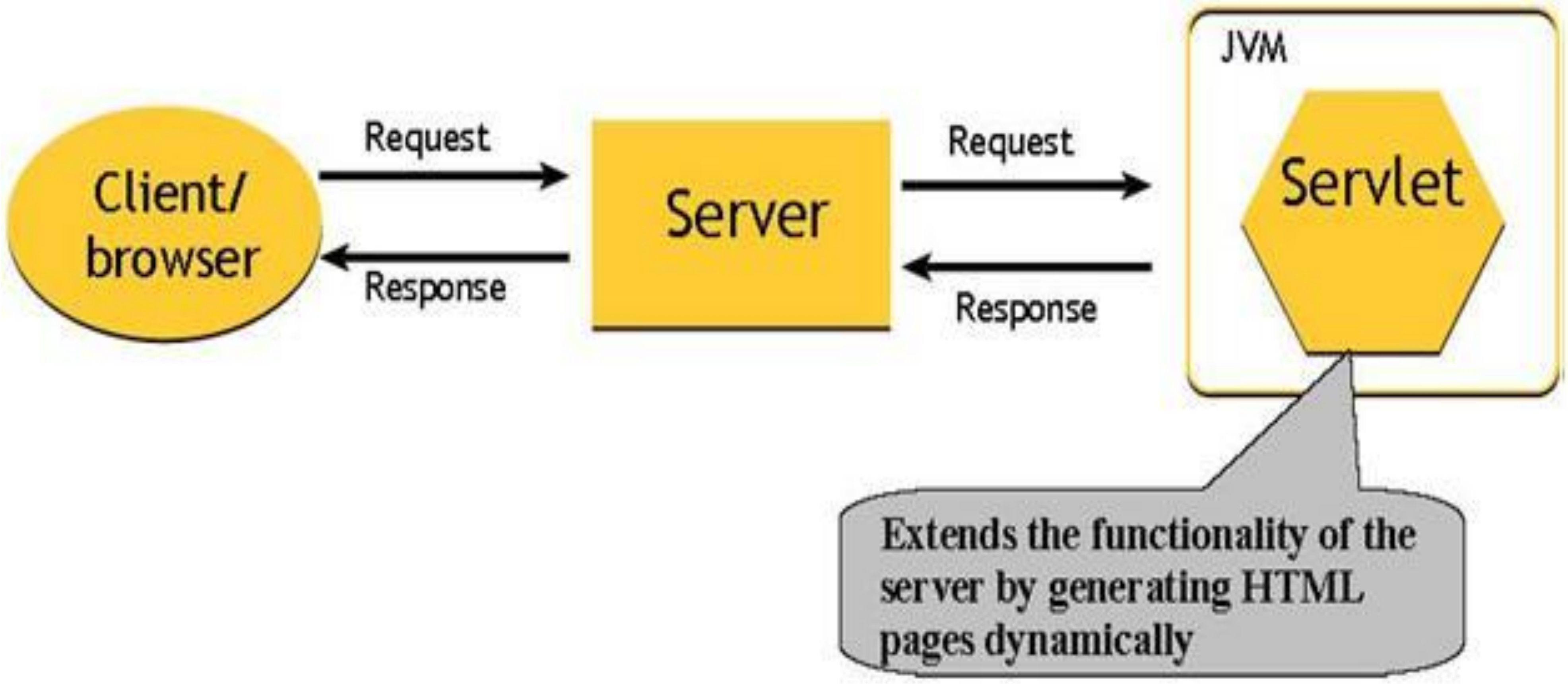
CGI(COMMMON GATEWAY INTERFACE)



DISADVANTAGES OF CGI TECHNOLOGY

- Platform Dependent
- If number of clients increases, it takes more time for sending response. – Lack of Scalability
- For each request, it starts a process and Web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.
- To address these limitations, **Java Servlet technology** was created as a portable way to provide dynamic, user-oriented content.

INTRODUCTION TO SERVLET



Extends the functionality of the server by generating HTML pages dynamically

JAVA SERVLET TECHNOLOGY

- **Servlet** technology is used to create web application (resides at server side and generates dynamic web page).
- Servlet can be described in many ways, depending on the context.
- Servlet is a technology i.e. used to create web application.
- Servlet is an API that provides many interfaces and classes including documentations.
- 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 type of requests.
- Servlet is a web component that is deployed on the server to create dynamic web page.

WHAT IS A SERVLET ?

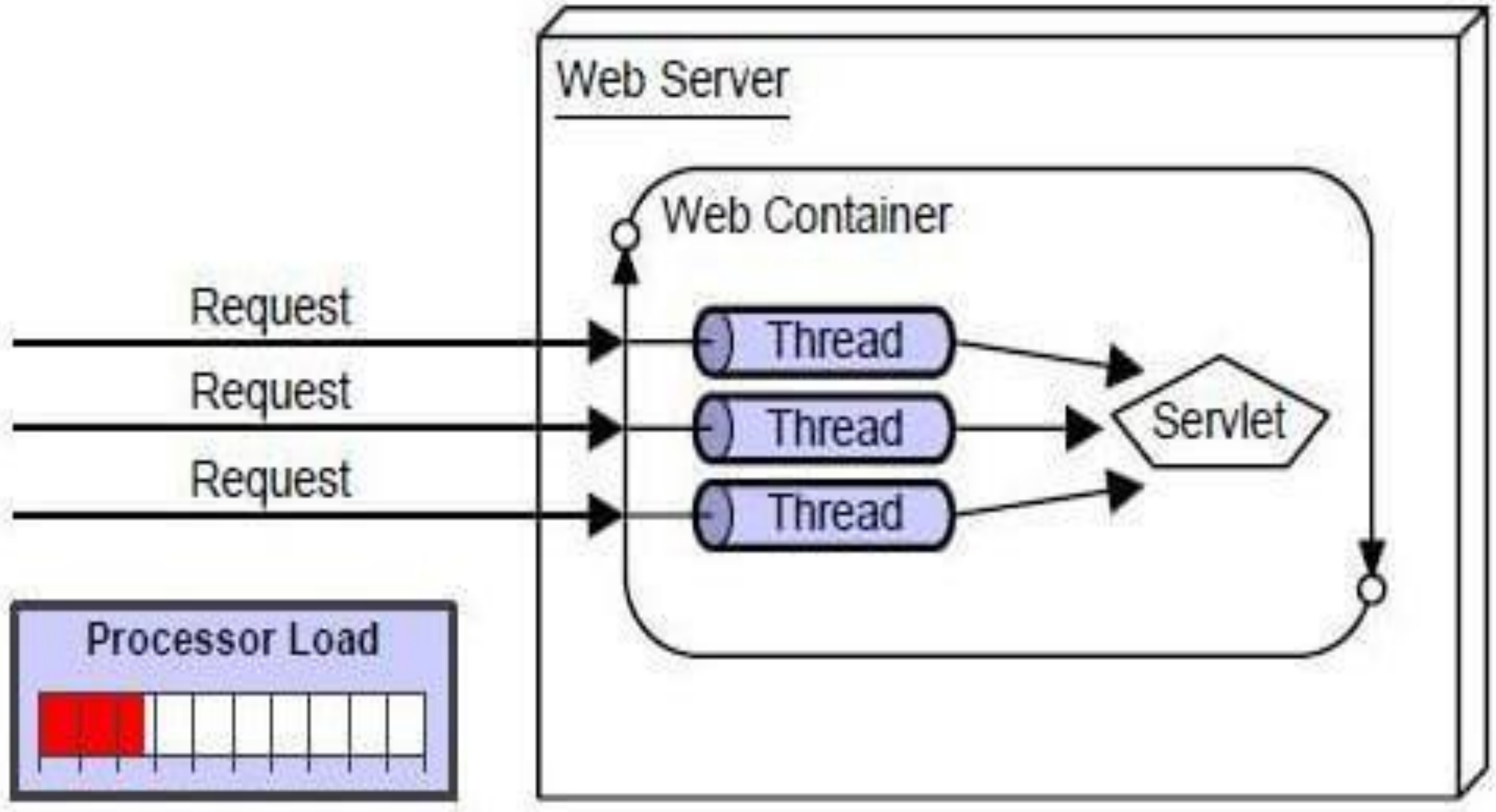
- “A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.”
- Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

WHAT IS A SERVLET ?

- **javax .servlet** and **javax.servlet.http** packages provide interfaces and classes for writing Servlet
- All Servlets must implement the Servlet interface, which defines lifecycle methods.
- Javax.servlet package contains many interfaces and classes that are used by the servlet or web container.
- **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

ADVANTAGES OF SERVLET OVER CGI

- Better Performance
- Portability
- Robust
- Secure

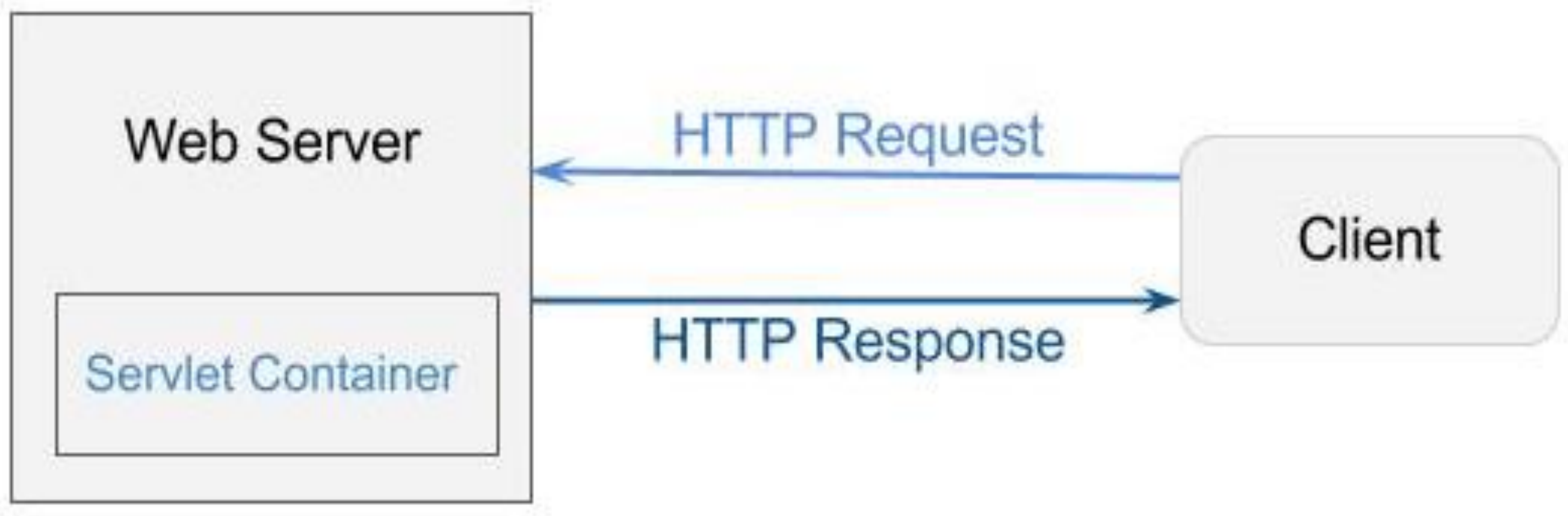


WEB TERMINOLOGY

- Website – Static and Dynamic
- HTTP, HTTP Request
- GET, POST
- Servlet Container
- Web Server and Application Server

SERVLET CONTAINER

- **The Servlet Container performs many operations that are given below:**
- Life Cycle Management
- Multithreaded support
- Object Pooling
- Security



WEB SERVER

- Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
- It is a computer where the web content can be stored.
- web server can be used to host the web sites but there also used some other web servers also such as FTP, email, storage, gaming etc.
- Examples of Web Servers are: **Apache Tomcat, IIS**
- It can respond to the client request in either of the following two possible ways:
  - Generating response by using the script and communicating with database.
  - Sending file to the client associated with the requested URL.

ENVIRONMENT SETUP - ECLIPSE

- Configure dynamic Web Project in Eclipse IDE
- Set up Apache Tomcat Server instance in Eclipse IDE
- Set up build path of project; add all Servlet and JSP Libraries (Add Library – servlet-api.jar)
- Add JDBC driver JAR for MySQL – Paste Under Lib folder of your workspace and Add JAR from Properties
  - **https://dev.mysql.com/downloads/connector/j/5.1.html**
- Deploy and run the Project on the Server – Right click Run As Server
  - WAR file - tomcat understands only WAR
- Deploy WAR external to Eclipse IDE

DEPLOY EXTERNAL TO ECLIPSE

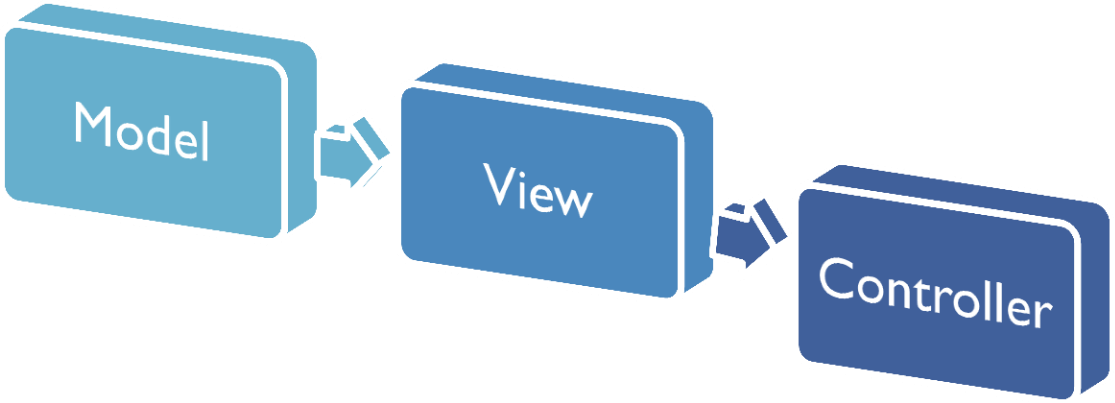
- Right click project and export it to generate WAR file
- Put WAR file in Tomcat – webapps folder to deploy your project
- Go to <Tomcat_Home>\bin in command prompt
- Start Server : startup.bat

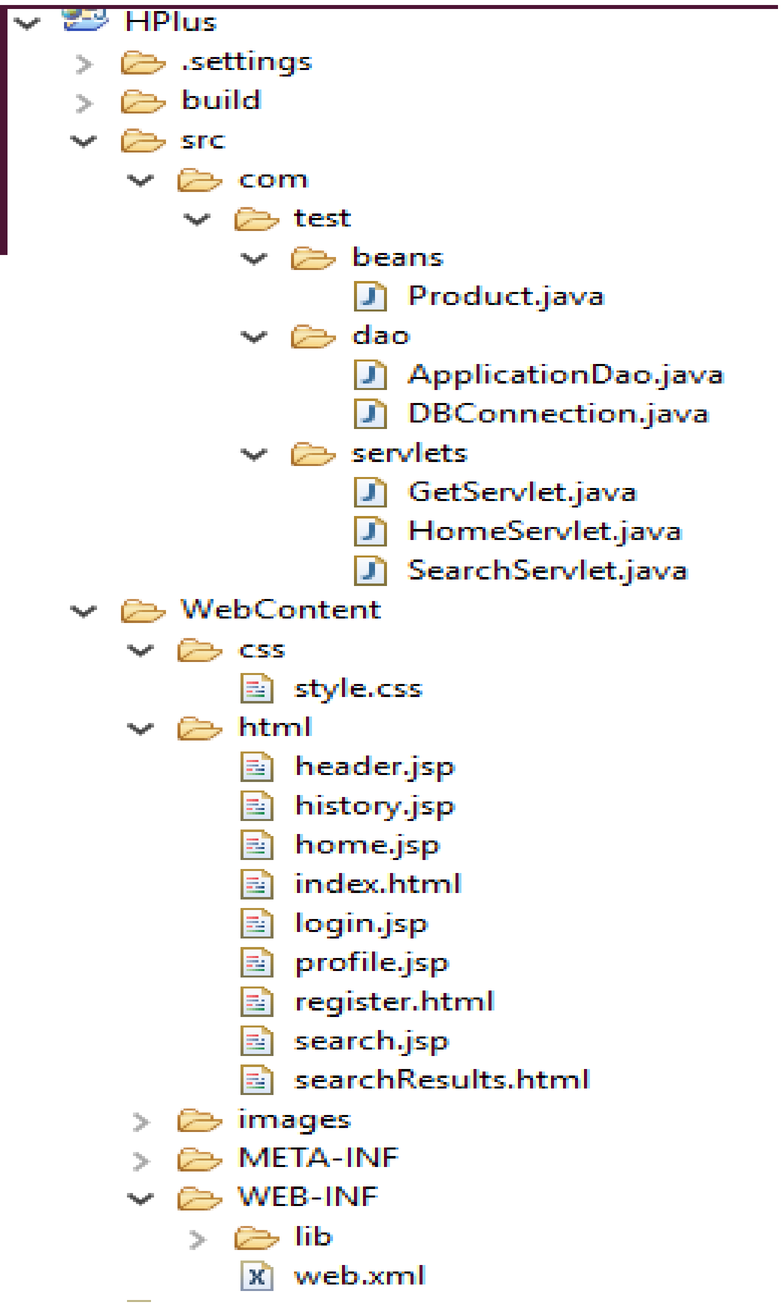
ALTERNATIVES TO SETUP APPLICATION

- Maven or Gradle – Set up the entire project structure and configure build path with all necessary Library
- JBoss, WebLogic, GlassFish or ant Java EE – Compliant contianers/servers for hosting the application
- Ask build tool itself to deploy and run the project on server
- NetBeans or IntelliJ IDE for development
- Any other relational database

PROJECT APPLICATION SETUP

- Implements Model-> View-> Controller(MVC) architecture
  - View – JSP or HTML files
  - Controller – Servlet classes that intercept request and prepare response
  - Model – data access object (DAO) classes that talk to the database
  - Copy css , html and images folder under Web Content in your project





```
HPlus
  .settings
  build
  src
    com
      test
        beans
          Product.java
        dao
          ApplicationDao.java
          DBConnection.java
        servlets
          GetServlet.java
          HomeServlet.java
          SearchServlet.java
  WebContent
    css
      style.css
    html
      header.jsp
      history.jsp
      home.jsp
      index.html
      login.jsp
      profile.jsp
      register.html
      search.jsp
      searchResults.html
    images
    META-INF
    WEB-INF
      lib
      web.xml
```

PROJECT APPLICATION SETUP : DATABASE SETUP FOR APPLICATION

- Schema name of your choice – hplus
- MySQL database server
- Following tables have been used:
  - Users – to store all user information
  - Products – to store all product related information
  - Orders - to store order history of a particular user
- hplus.sql :You can import – run this script in MySQL server

TYPES OF HTTP REQUEST

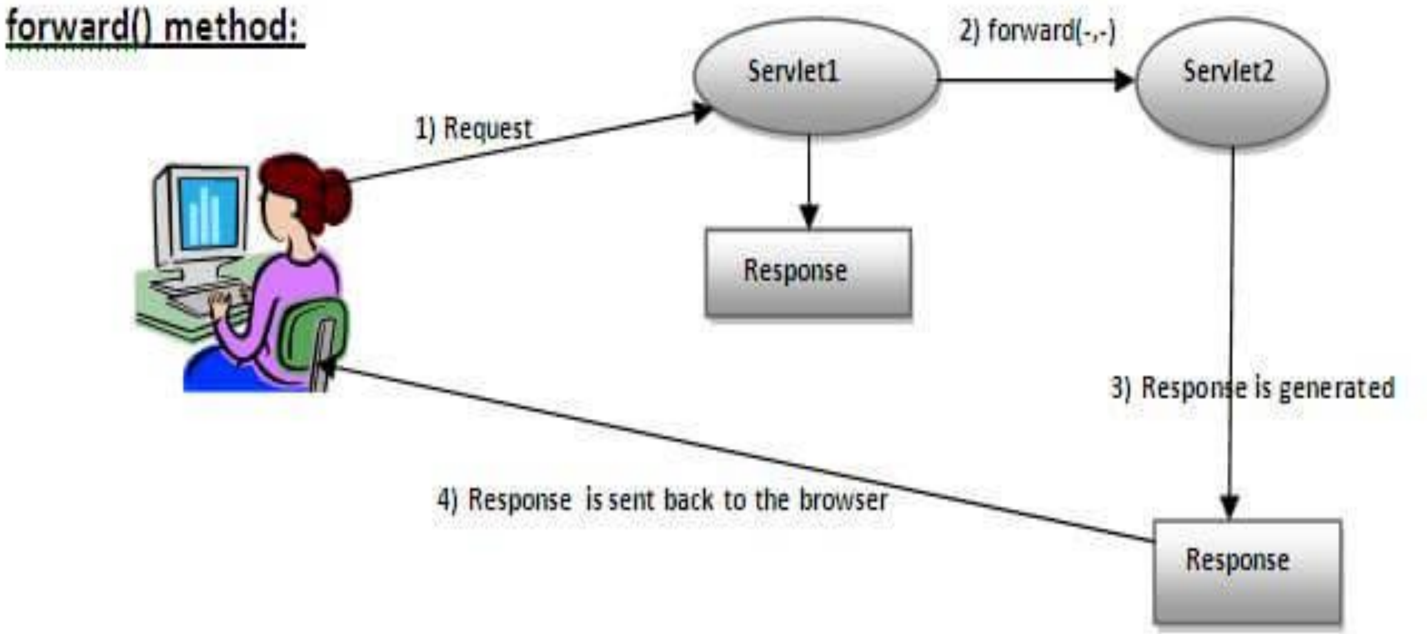
- GET – gets information from server:Idempotent – Wouldn’t change anything on server side if request is sent out multiple times
- POST – processes information on server
- PUT – Uploads a resource on server
- DELETE - deletes a resource on server
- HEAD – same as GET , but returns only the headers
- OPTIONS – helps trace what HTTP methods work on server

FORWARDING IN SERVLET

- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.
- This interface can also be used to include the content of another resource also.
- There are two methods defined in the RequestDispatcher interface.

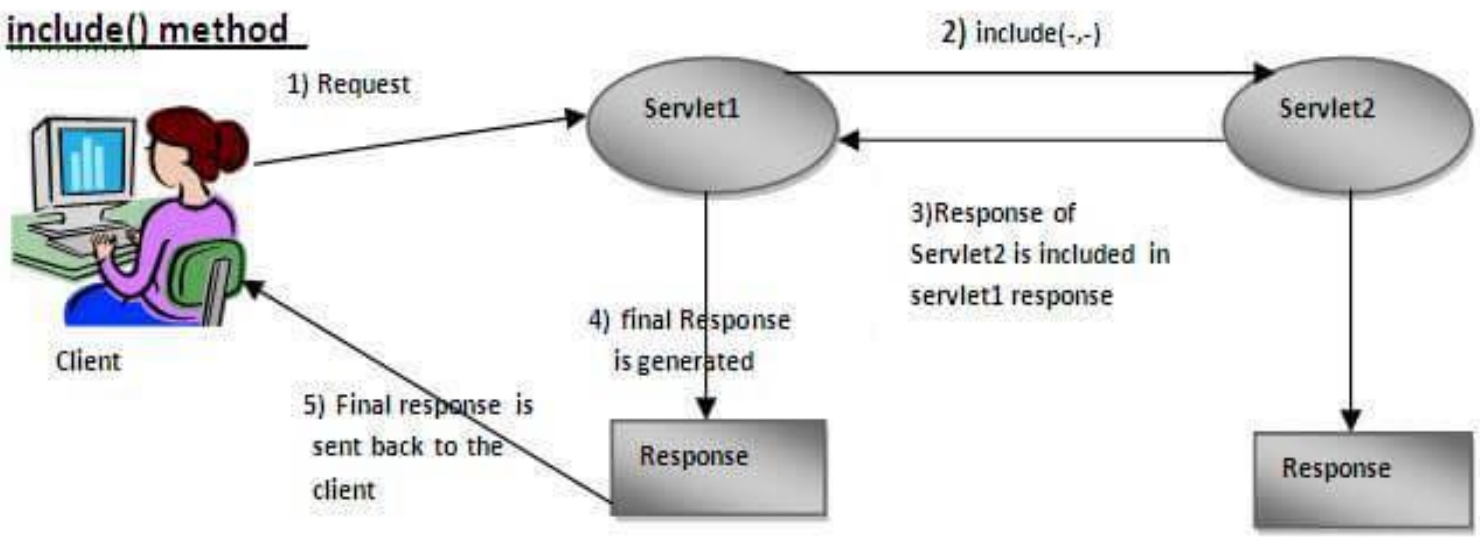
FORWARD() METHOD

- Response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.



FORWARD() METHOD

- Response of second servlet is included in the response of the first servlet that is being sent to the client.


include() method
1) Request
Client
Servlet1
2) include(-,-)
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
5) Final response is sent back to the client
Response
Response

SEND REDIRECT IN SERVLET

- This method is used to redirect response to another resource,
  - It may be servlet
  - JSP or
  - html file.
- This interface can also be used to include the content of another resource.

FORWARD() VS SENDREDIRECT()

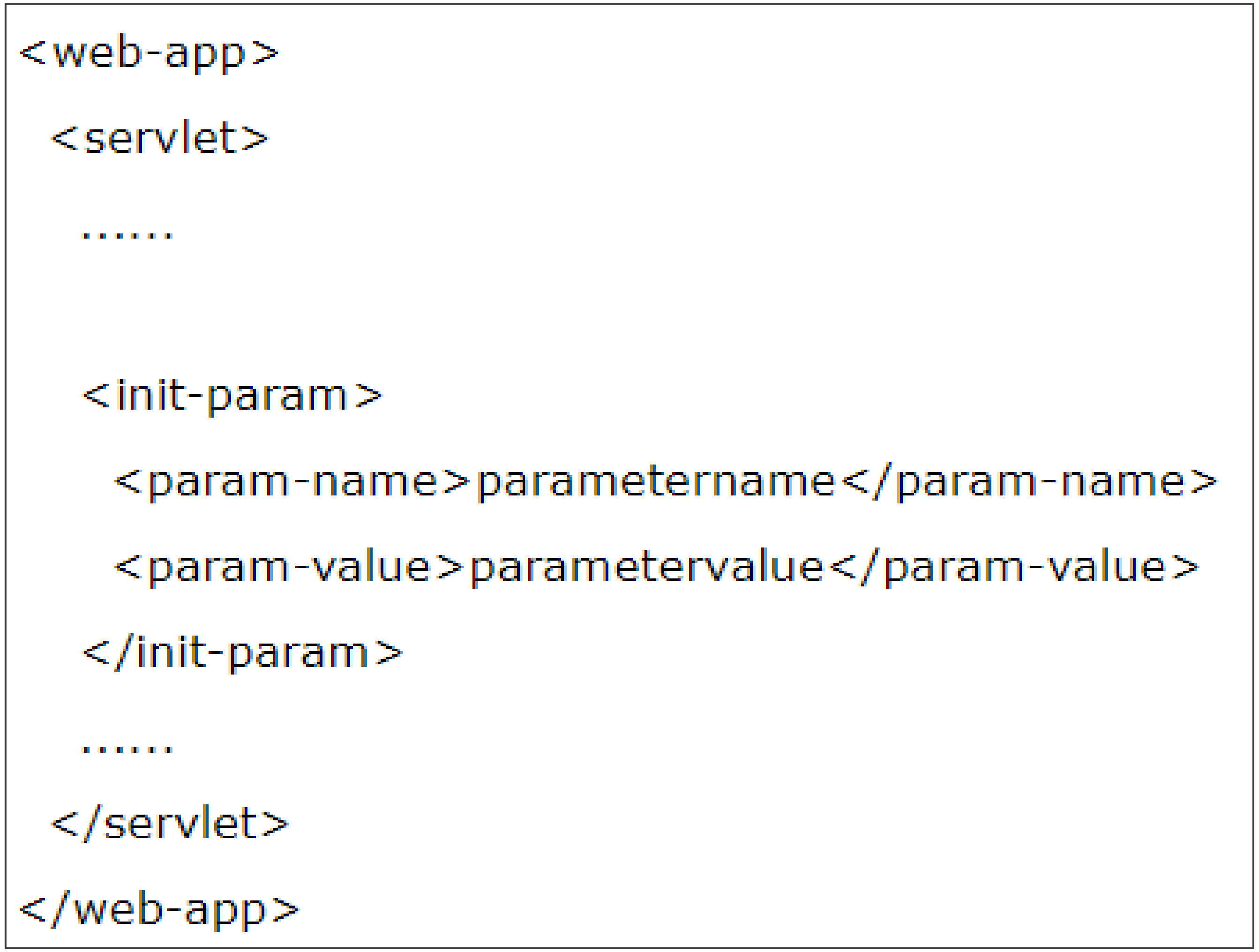
- comparison

SERVLETCONFIG INTERFACE

- An object of ServletConfig is created by the web container for each servlet.
- This object can be used to get configuration information from web.xml file.
- The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.
- **public** ServletConfig getServletConfig(); //syntax

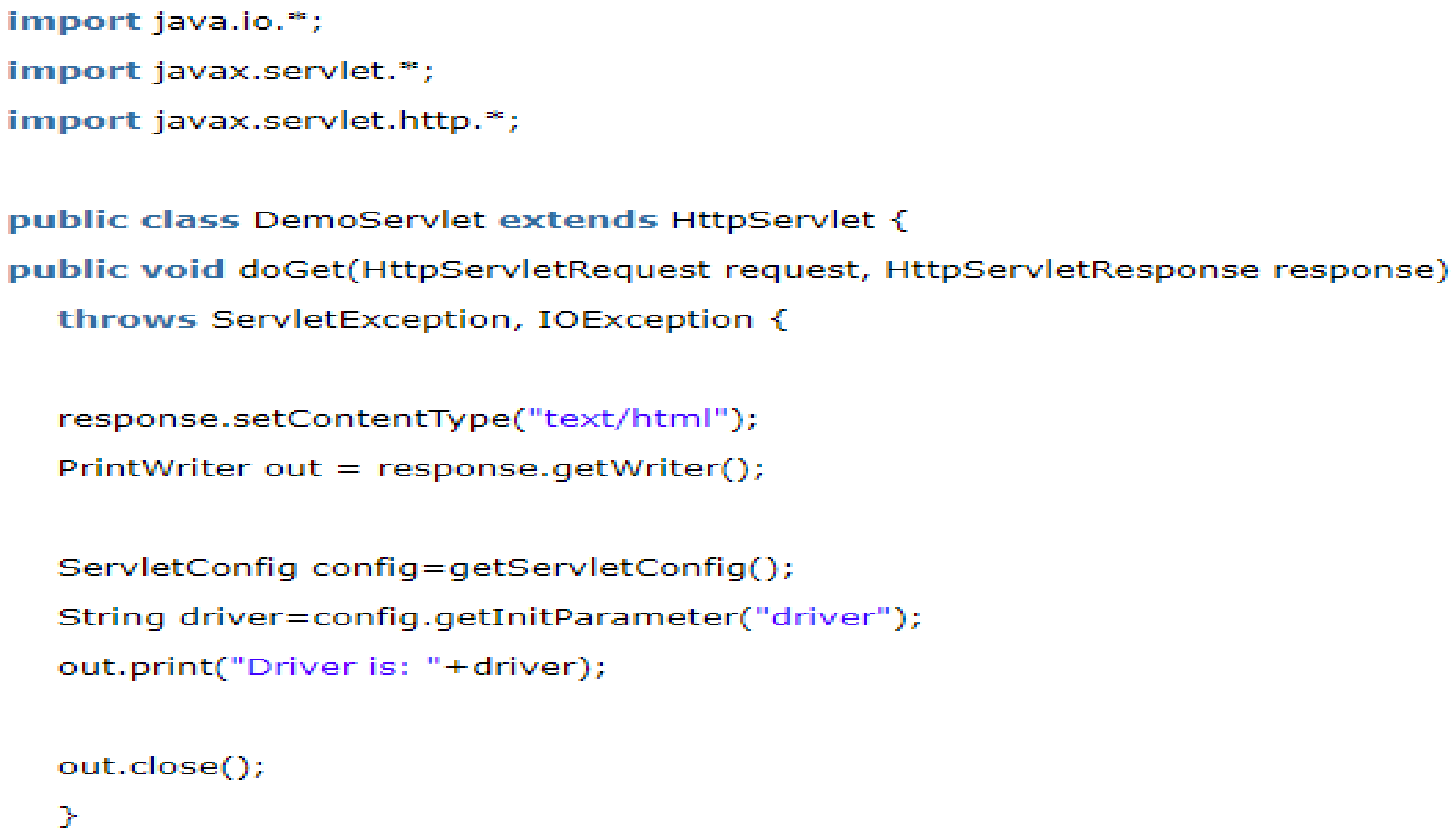
SERVLETCONFIG INTERFACE

- ServletConfig config=getServletConfig();



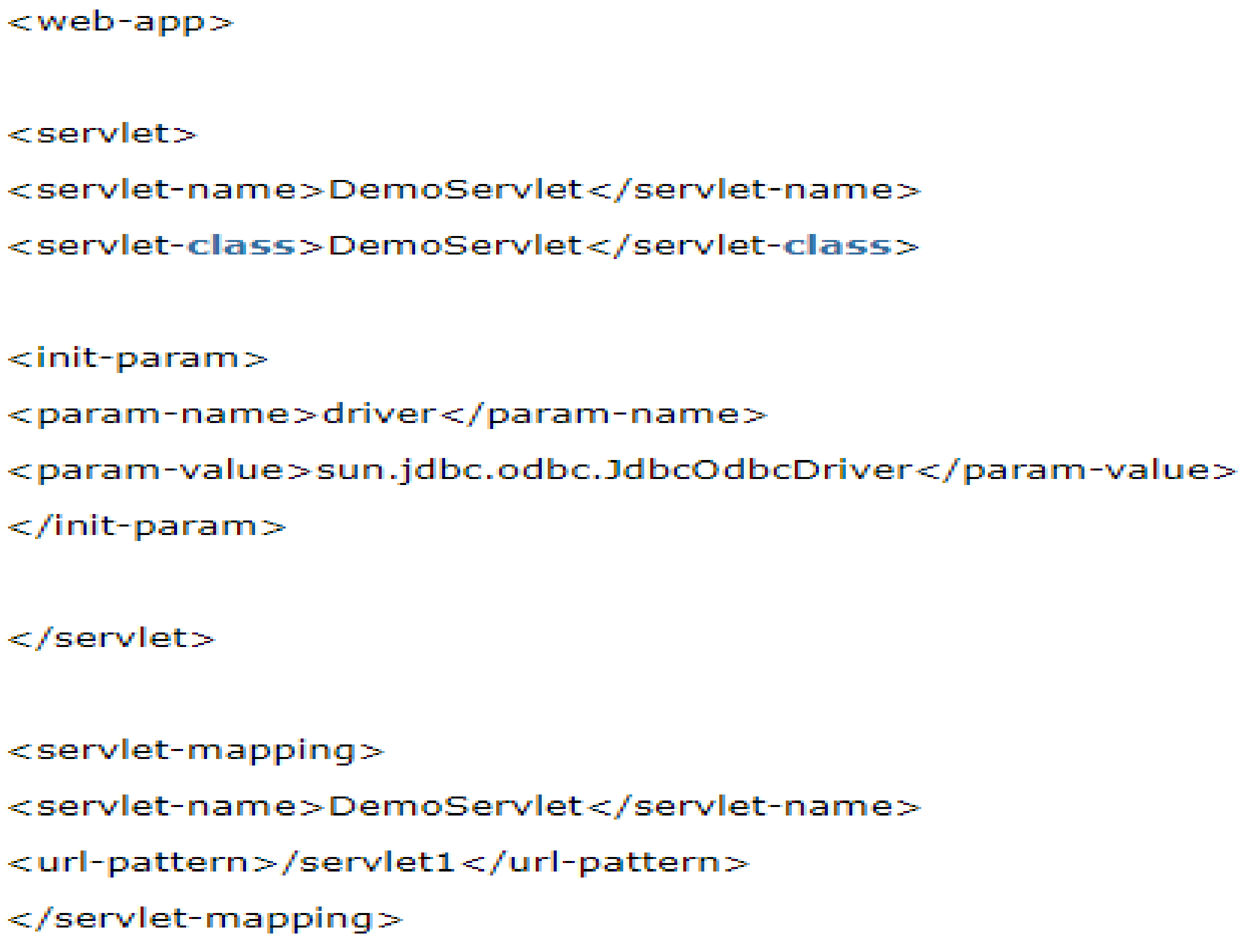
```
<web-app>
 <servlet>
  ......

  <init-param>
   <param-name>parametername</param-name>
   <param-value>parametervalue</param-value>
  </init-param>
  ......
 </servlet>
</web-app>
```

EXAMPLE OF SERVLETCONFIG TO GET INITIALIZATION PARAMETER



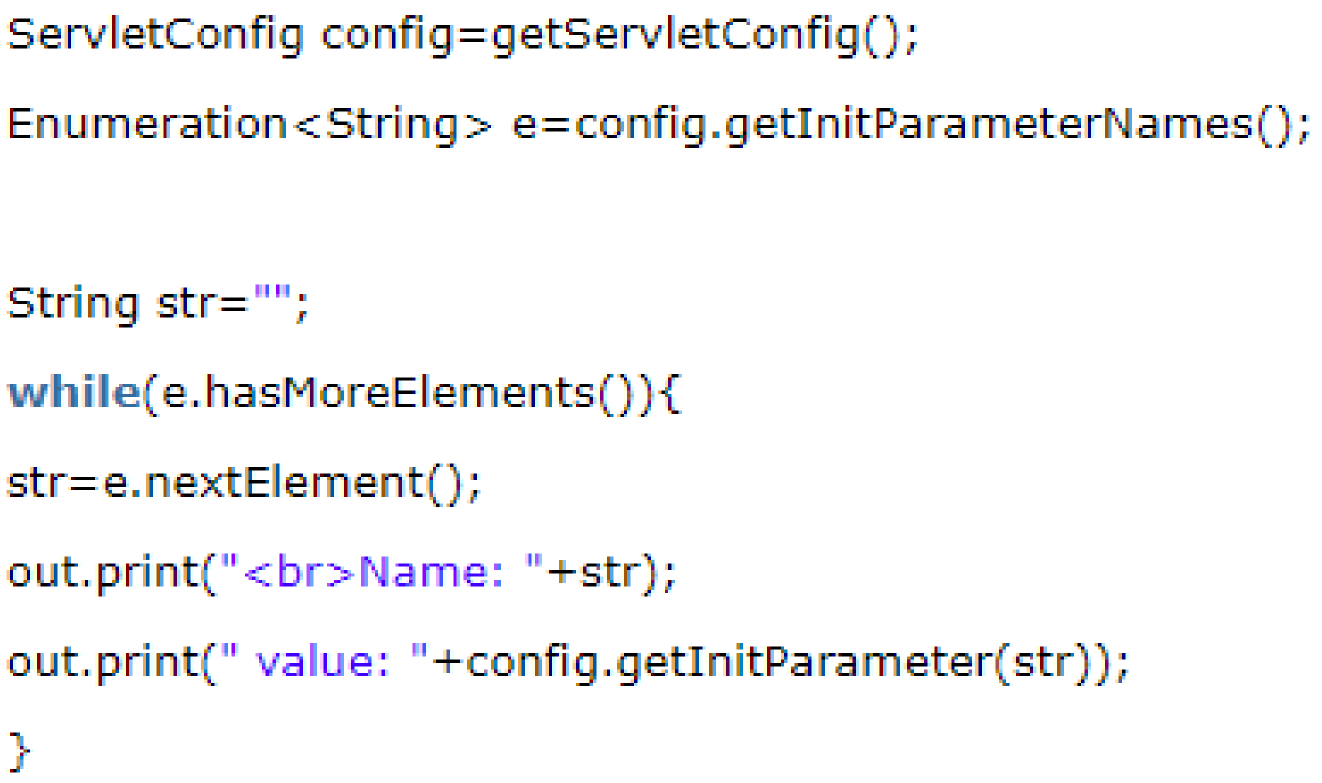
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DemoServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {

  response.setContentType("text/html");
  PrintWriter out = response.getWriter();

  ServletConfig config=getServletConfig();
  String driver=config.getInitParameter("driver");
  out.print("Driver is: "+driver);

  out.close();
  }
```

WEB.XML



```
<web-app>

<servlet>
<servlet-name>DemoServlet</servlet-name>
<servlet-class>DemoServlet</servlet-class>

<init-param>
<param-name>driver</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>DemoServlet</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>
```

EXAMPLE OF SERVLETCONFIG TO GET ALL THE INITIALIZATION PARAMETERS



```
ServletConfig config=getServletConfig();
Enumeration<String> e=config.getInitParameterNames();


String str="";
while(e.hasMoreElements()){
str=e.nextElement();
out.print("<br>Name: "+str);
out.print(" value: "+config.getInitParameter(str));
}
```

SERVLETCONTEXT INTERFACE

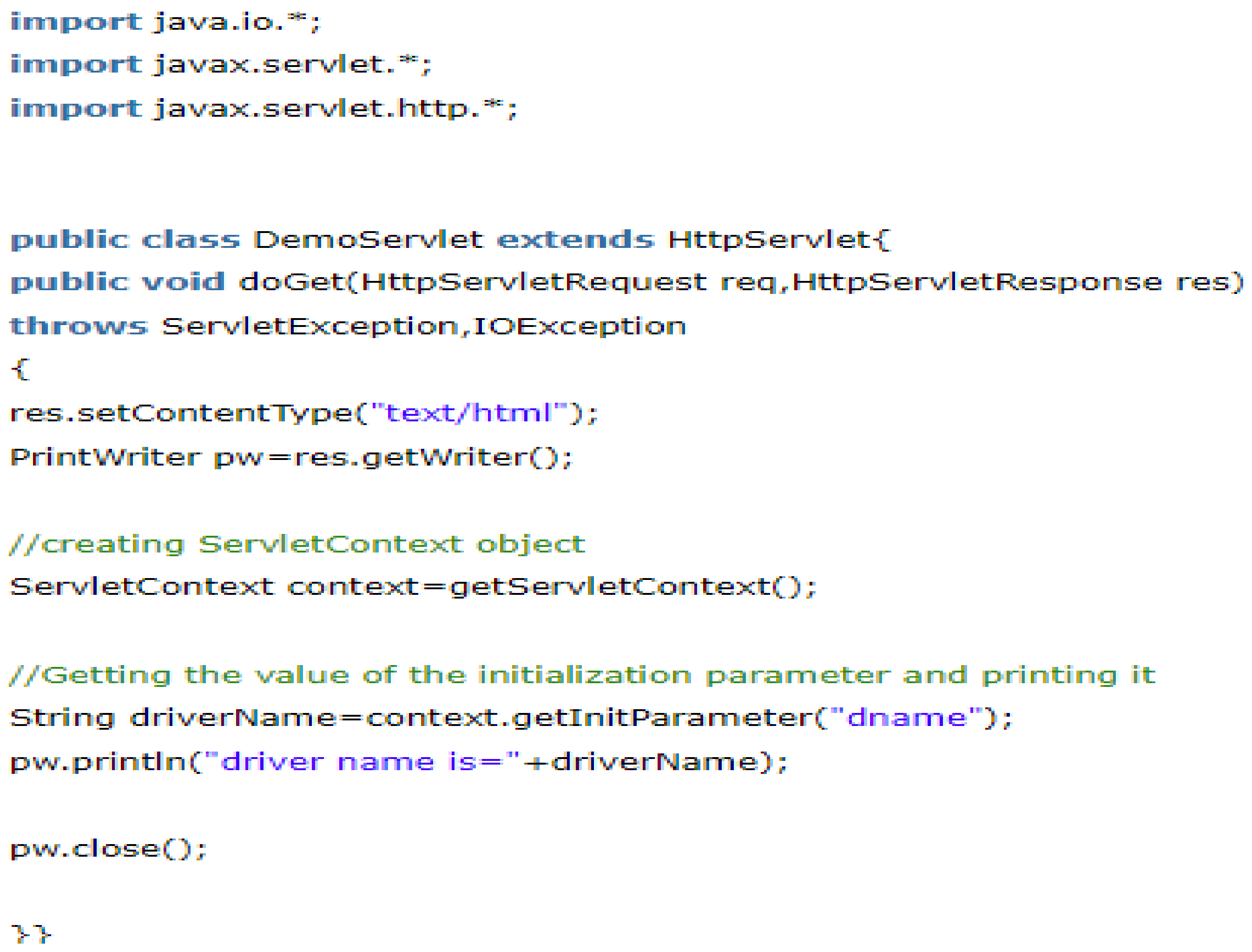
- An object of ServletContext is created by the web container at time of deploying the project.
- This object can be used to get configuration information from web.xml file.
- There is only one ServletContext object per web application.
- If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.
  - The object of ServletContext provides an interface between the container and servlet.
  - The ServletContext object can be used to get configuration information from the web.xml file.
  - The ServletContext object can be used to set, get or remove attribute from the web.xml file.
  - The ServletContext object can be used to provide inter-application communication.

TO GET THE OBJECT OF SERVLETCONTEXT INTERFACE

- **getServletContext() method** of ServletConfig interface returns the object of ServletContext.
- **getServletContext() method** of GenericServlet class returns the object of ServletContext.
- Examples
  - //We can get the ServletContext object from ServletConfig object
  - ServletContext application=getServletConfig().getServletContext();

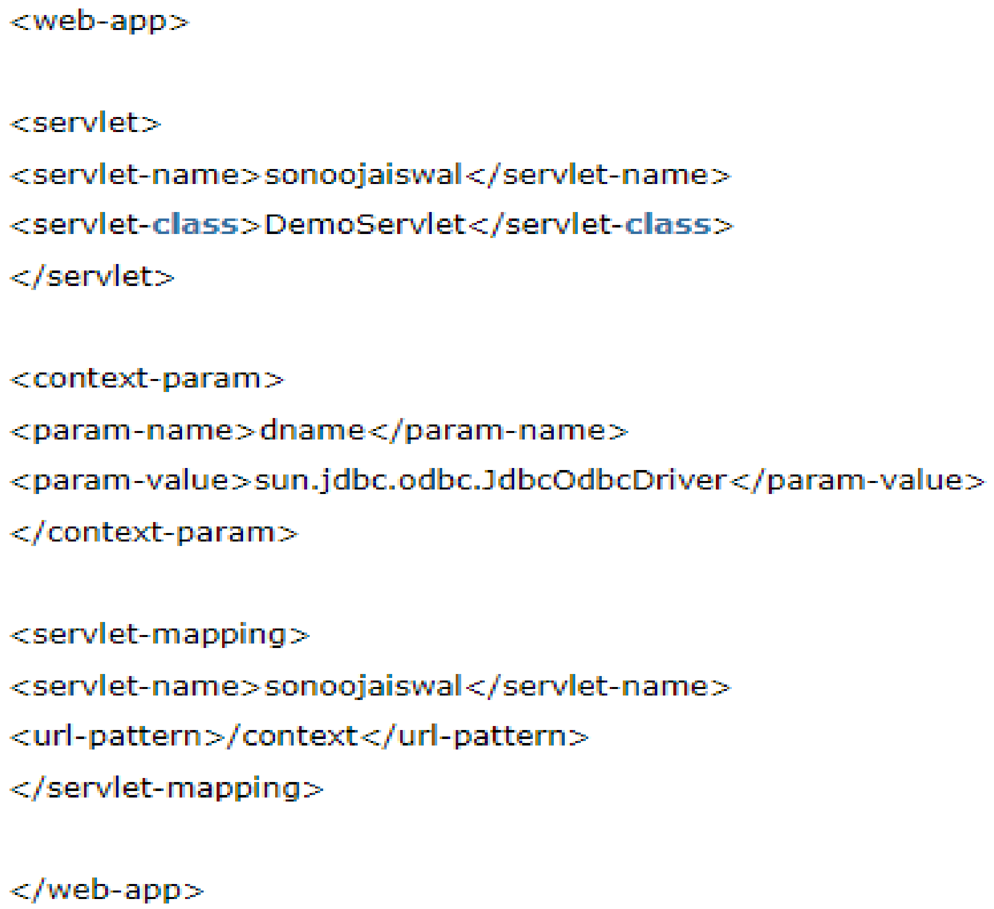
  - //Another convenient way to get the ServletContext object
  - ServletContext application=getServletContext();

EXAMPLE OF SERVLETCONTEXT TO GET THE INITIALIZATION PARAMETER



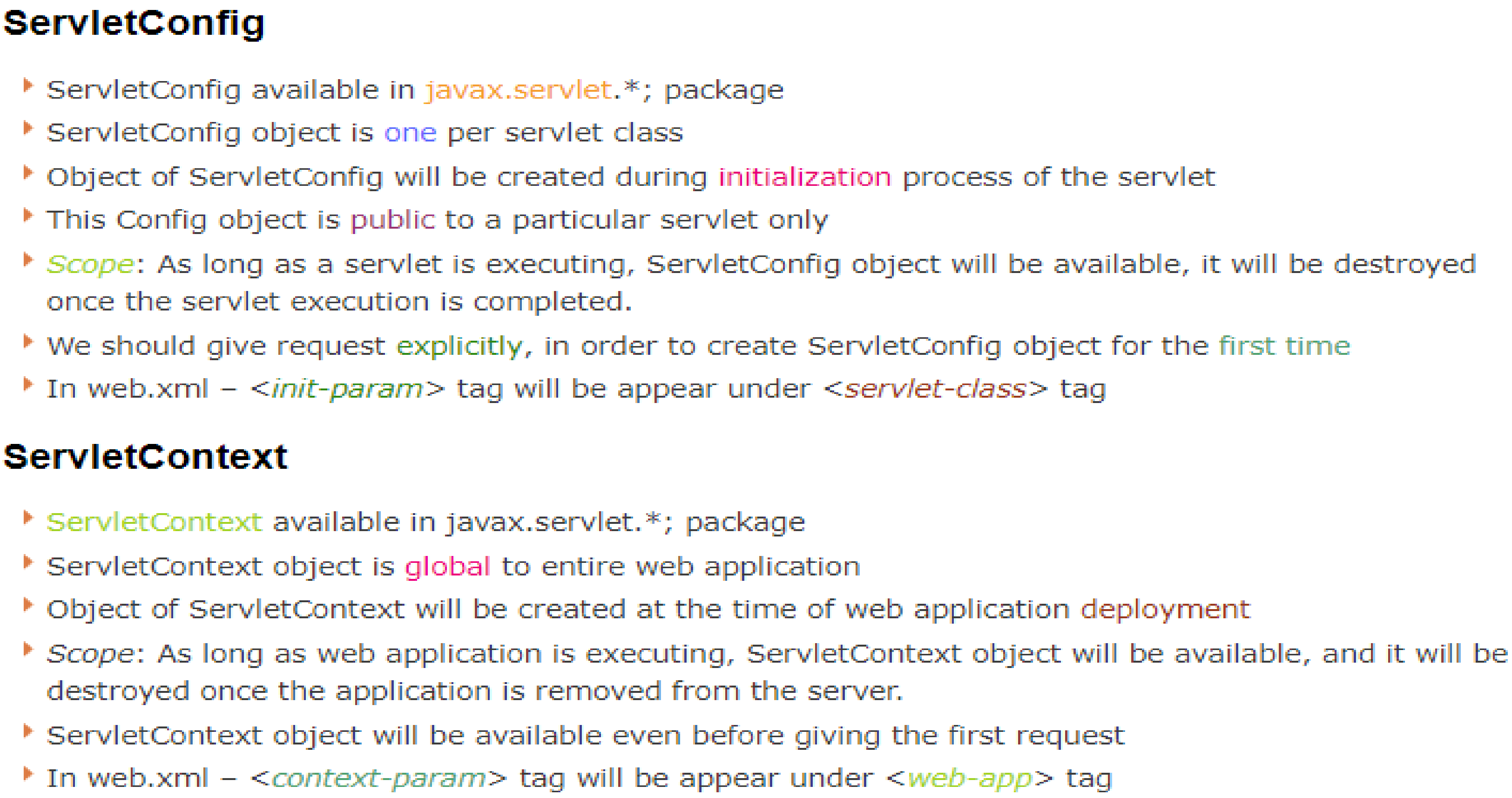
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter pw=res.getWriter();

//creating ServletContext object
ServletContext context=getServletContext();

//Getting the value of the initialization parameter and printing it
String driverName=context.getInitParameter("dname");
pw.println("driver name is="+driverName);

pw.close();

}}
```

Web.xml



```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</servlet-name>
<servlet-class>DemoServlet</servlet-class>
</servlet>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/context</url-pattern>
</servlet-mapping>

</web-app>
```

COMPARISON



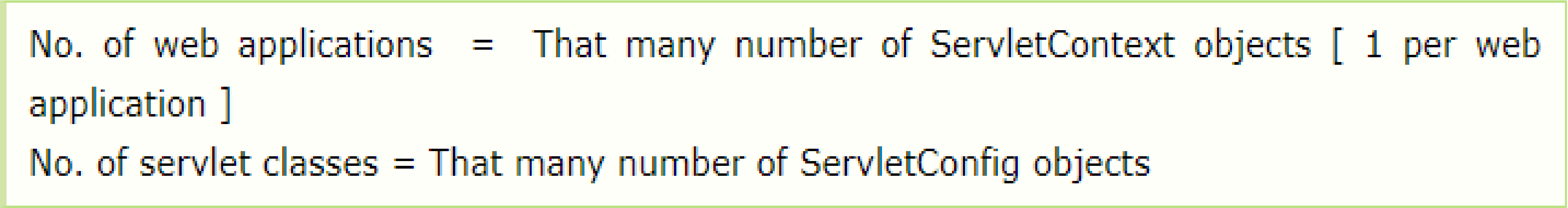
ServletConfig

- ServletConfig available in javax.servlet.*; package
- ServletConfig object is one per servlet class
- Object of ServletConfig will be created during initialization process of the servlet
- This Config object is public to a particular servlet only
- *Scope*: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.
- We should give request explicitly, in order to create ServletConfig object for the first time
- In web.xml – <*init-param*> tag will be appear under <*servlet-class*> tag

ServletContext

- ServletContext available in javax.servlet.*; package
- ServletContext object is global to entire web application
- Object of ServletContext will be created at the time of web application deployment
- *Scope*: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.
- ServletContext object will be available even before giving the first request
- In web.xml – <*context-param*> tag will be appear under <*web-app*> tag

COMPARISON



No. of web applications = That many number of ServletContext objects [1 per web application]

No. of servlet classes = That many number of ServletConfig objects

REFERENCES

- https://docs.oracle.com/javaee/7/tutorial/
- https://www.java4s.com
- https://www.lynda.com
- https://javarevisited.blogspot.com
- https://www.javatpoint.com/