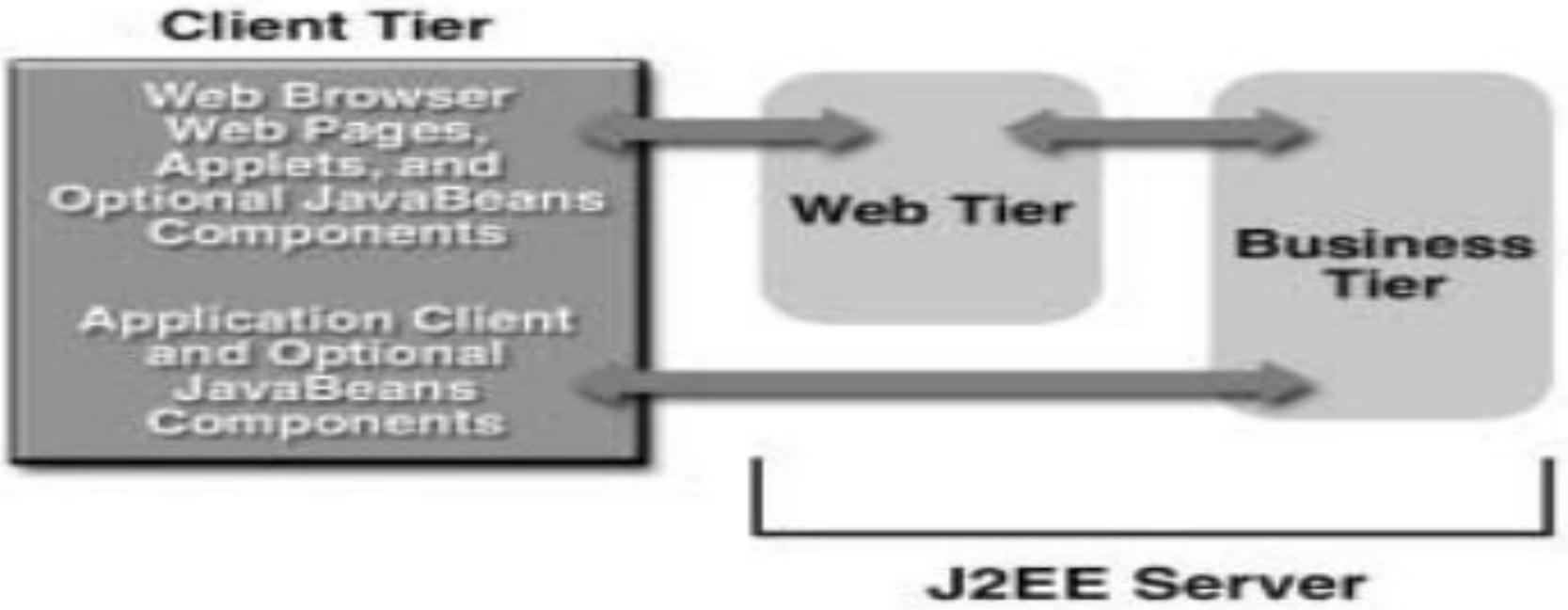
J2EE COMPONENTS AND CONTAINERS

Server Communications

Dr.R.Saranya

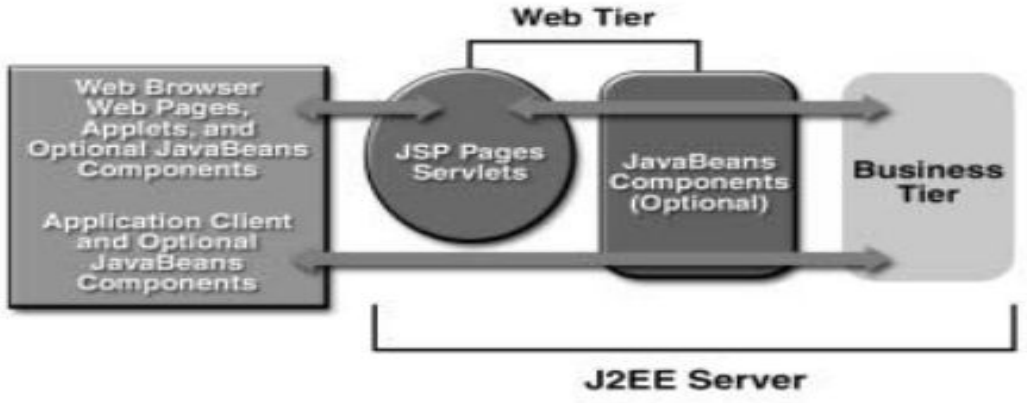
J2EE SERVER COMMUNICATIONS



J2EE SERVER COMMUNICATIONS

Web Components

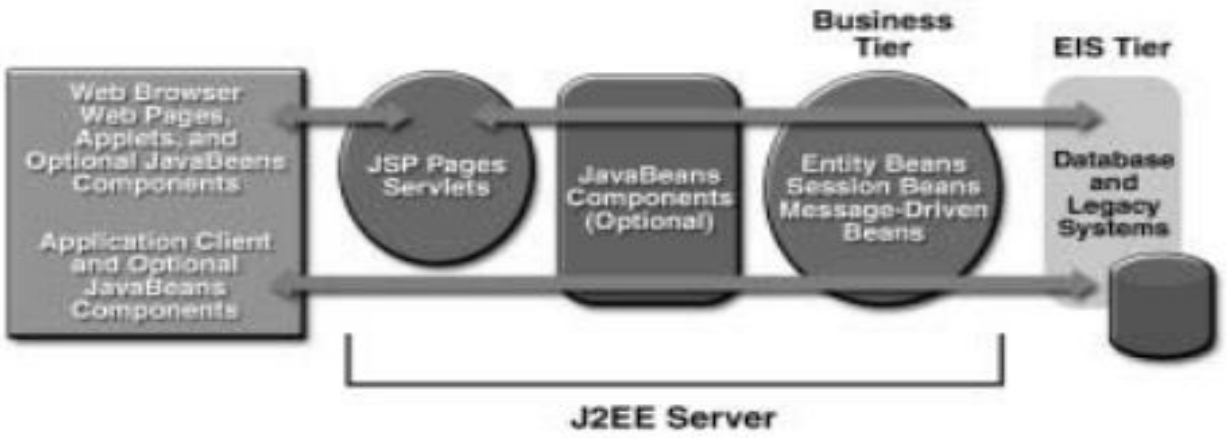
- Can be either Servlets or JSP pages.
- Servlets are Java programming language classes that dynamically process requests and construct responses.
- JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.



J2EE SERVER COMMUNICATIONS

Business Components

- Has business code which are handled by enterprise beans running in the business tier.
- An enterprise bean receives data from client programs, processes it (if necessary), and sends it to the enterprise information system (EIS) tier for storage.
- An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.



J2EE SERVER COMMUNICATIONS

Business Components

Three kinds of enterprise beans: session beans, entity beans, and message-driven beans.

Session bean - represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone.

Entity bean - represents persistent data stored in one row of a database table. If the client terminates or if the server shuts down, the underlying services ensure that the entity bean data is saved.

Message-driven bean - combines features of a session bean and a Java Message Service ("JMS") message listener, allowing a business component to receive JMS messages asynchronously.

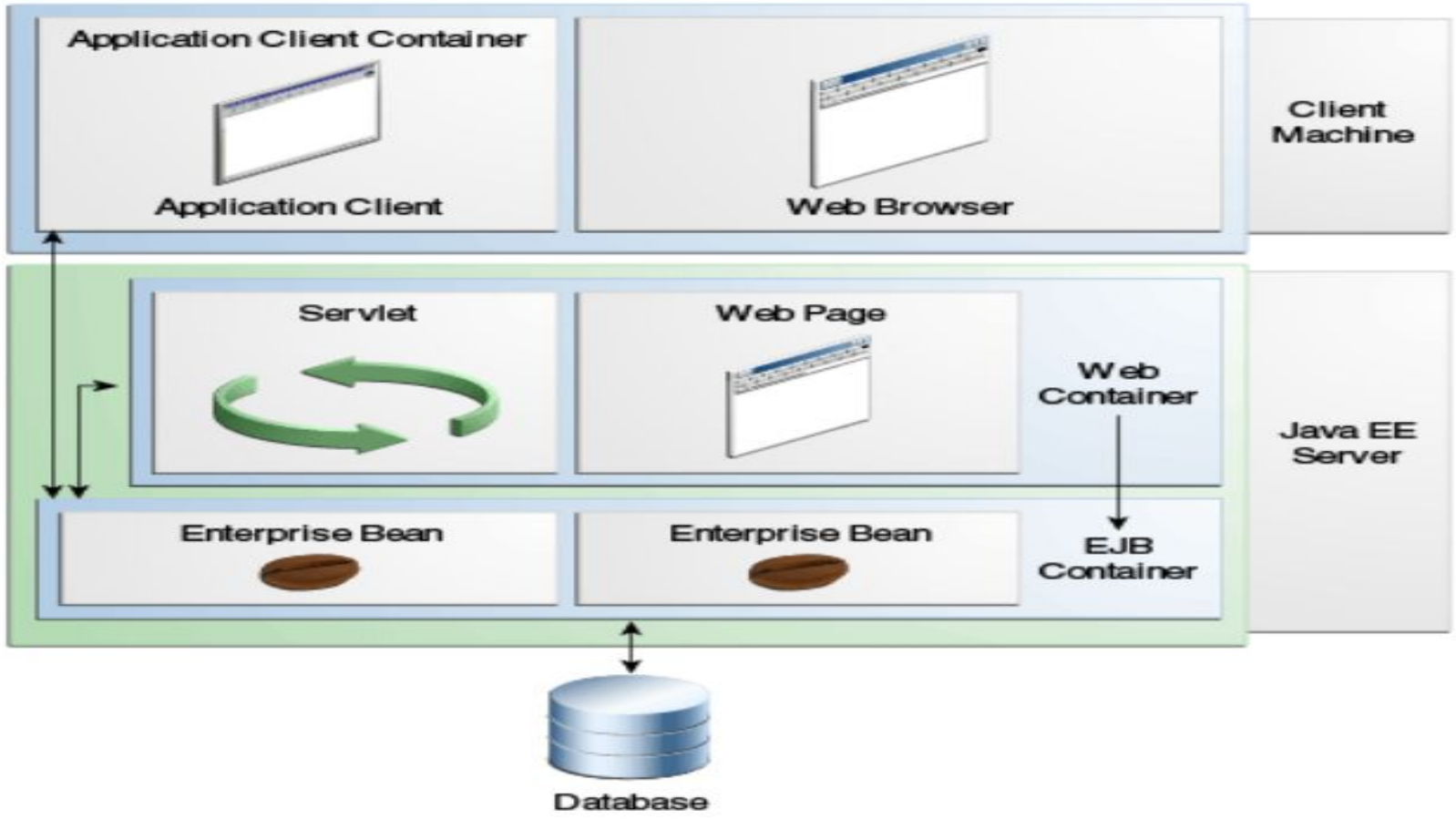
J2EE CONTAINERS

- *Containers* are the interface between a component and the low-level platform-specific functionality that supports the component.
- Before a web component, enterprise bean, or application client component can be executed, it must be assembled into a J2EE module and deployed into its container.
- Container provides an execution environment on top of the JVM.
- Services provided by the container are :
  - Security
  - Transaction
  - invocation
  - Persistence
  - Concurrency
  - Availability
  - Configuration
  - Remote connectivity.

J2EE CONTAINERS – Services

- Security model - lets you configure a web component or enterprise bean so that system resources are accessed only by **authorized users**.
- Transaction model - lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
- Remote connectivity - model manages low-level communications between clients and enterprise beans.
- Configurable services - components within the same application can behave differently based on where they are deployed.
- The container also manages non-configurable services such as
  - enterprise bean and servlet lifecycles,
  - database connection resource pooling,
  - data persistence,
  - access to the Java EE platform APIs

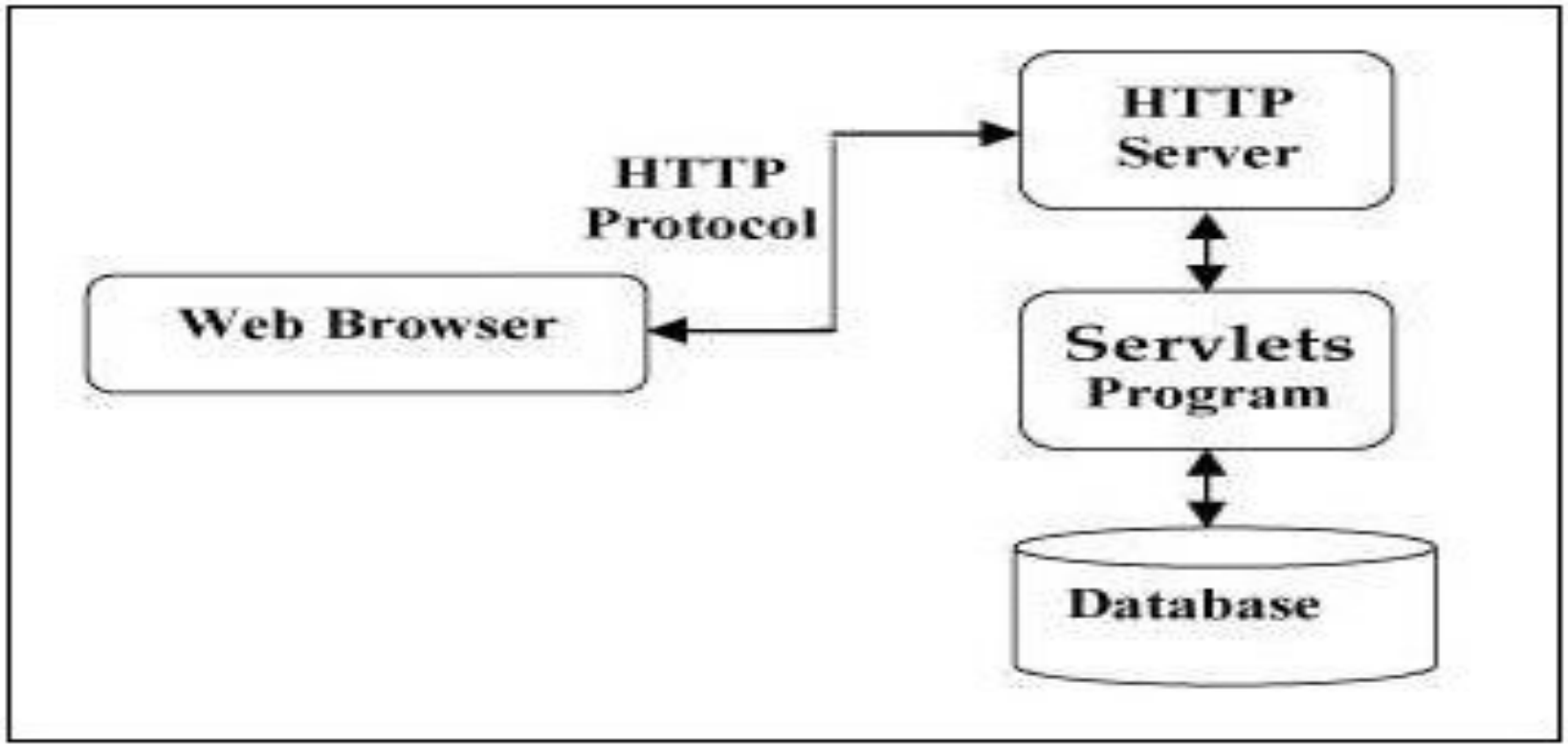
J2EE CONTAINERS – Types



J2EE CONTAINERS - Servlet

- A servlet is a small Java program that runs within a Web or application server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol (act as middle layer)
- Servlets offer several advantages in comparison with the CGI(Common Gateway Interface (CGI) :
  - Performance is significantly better.
  - Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
  - Servlets are platform-independent because they are written in Java.
  - Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
  - The full functionality of the Java class libraries is available to a servlet

Servlet Architecture



Servlets Tasks

Major tasks are :

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML)

Servlet – Environment setup

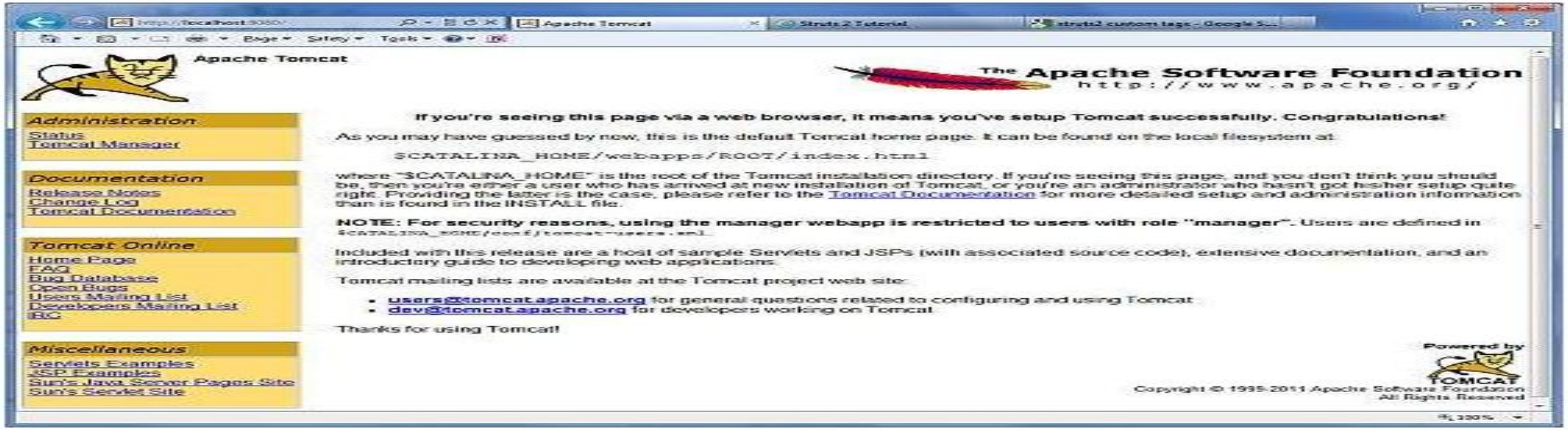
- Servlets Packages - **javax.servlet** and **javax.servlet.http**
- Download SDK from Oracle's Java site – Java SE Downloads

```
set PATH = C:\jdk1.8.0_65\bin;%PATH%
set JAVA_HOME = C:\jdk1.8.0_65
```

- Setting up Web Server – Tomcat
- Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies and can act as a standalone server for testing servlets and can be integrated with the Apache Web Server.

Servlet – Environment setup

After startup, the default web applications included with Tomcat will b available by visiting **http://localhost:8080/**. If everything is fine then i should display following result −



Tomcat can be stopped by executing the following commands on windows machine −

```
C:\apache-tomcat-8.0.28\bin\shutdown
```

Servlet – Life Cycle

Three essential methods :

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.

Servlet – Life Cycle

init() :

- The init method is called only once.
- It is called only when the servlet is created, and not called for any user requests afterwards.
- The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.
- The init() method simply creates or loads some data that will be used throughout the life of the servlet.

```
public void init() throws ServletException {
    // Initialization code...
}
```

Servlet – Life Cycle

service()

- The servlet container (i.e. web server) calls the service() method to handle **requests coming from the client(browsers) and to write the formatted response back to the client.**
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service.
- The service () method is called by the container and service method invokes doGe, doPost, doPut, doDelete, etc. methods as appropriate.

```
public void service(ServletRequest request, ServletResponse response)
   throws ServletException, IOException {
}
```

Servlet – Life Cycle

doGet() & doPost() Methods

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

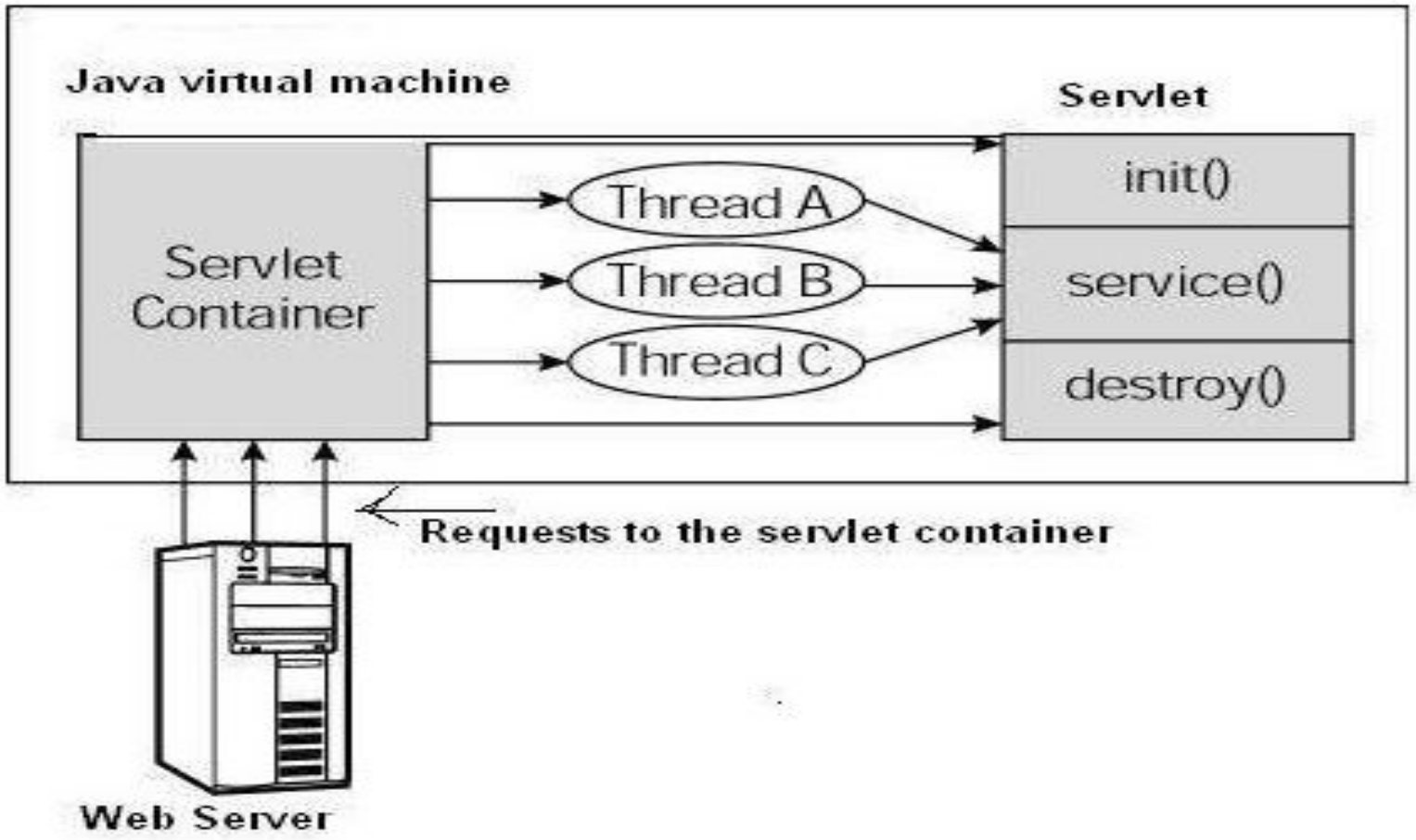
Servlet – Life Cycle

destroy()

- This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.
- After the destroy() method is called, the **servlet object is marked for garbage collection.** The destroy method definition looks like this –

```
public void destroy() {
   // Finalization code...
}
```

Servlet – Life Cycle - Architecture



Servlet – Life Cycle - Example

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

   private String message;

   public void init() throws ServletException {
      // Do required initialization
      message = "Hello World";
   }

   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      // Set response content type
      response.setContentType("text/html");

      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
   }

   public void destroy() {
      // do nothing.
   }
}
```

Servlet – Life Cycle - Example

Compiling a Servlet

```
$ javac HelloWorld.java
```

Servlet Deployment

Copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-I

```
<servlet>
   <servlet-name>HelloWorld</servlet-name>
   <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>HelloWorld</servlet-name>
   <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

Java Server Pages(JSP)

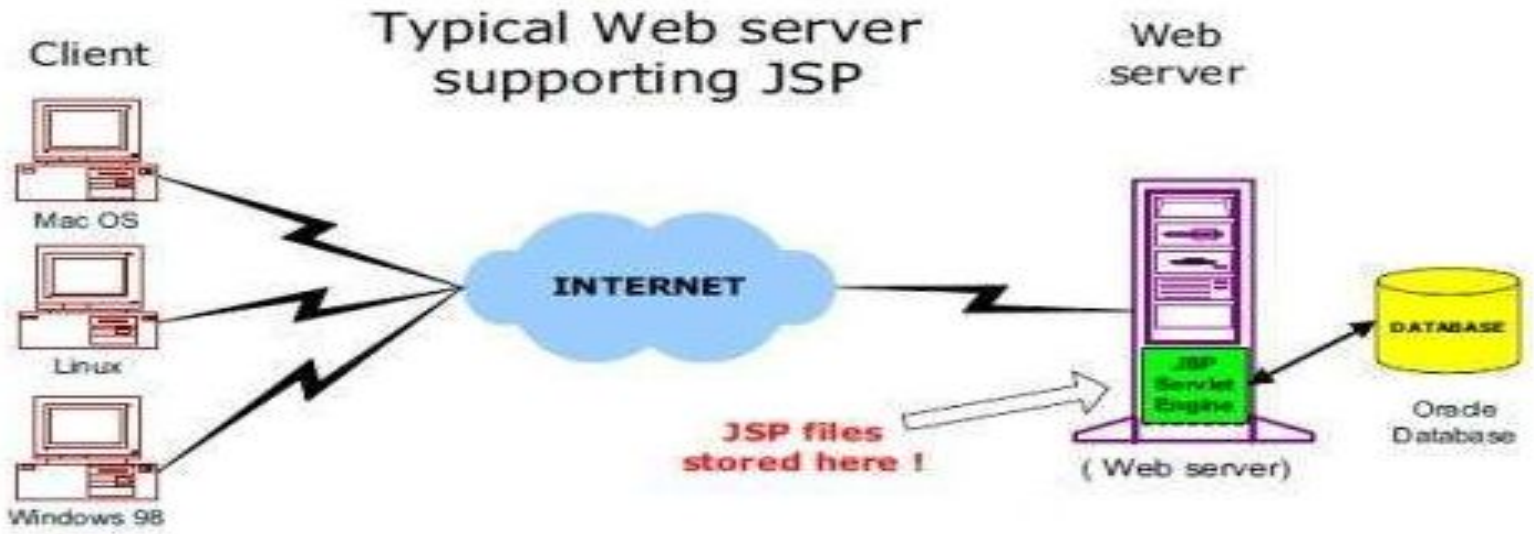
- Text based documents describe how to process a request and create a response that supports dynamic content.
- Contains HTML or XML and other JSP elements defined by JSP specification.
- JSP tags - start with **<%** and end with **%>.**

Advantages of JSP

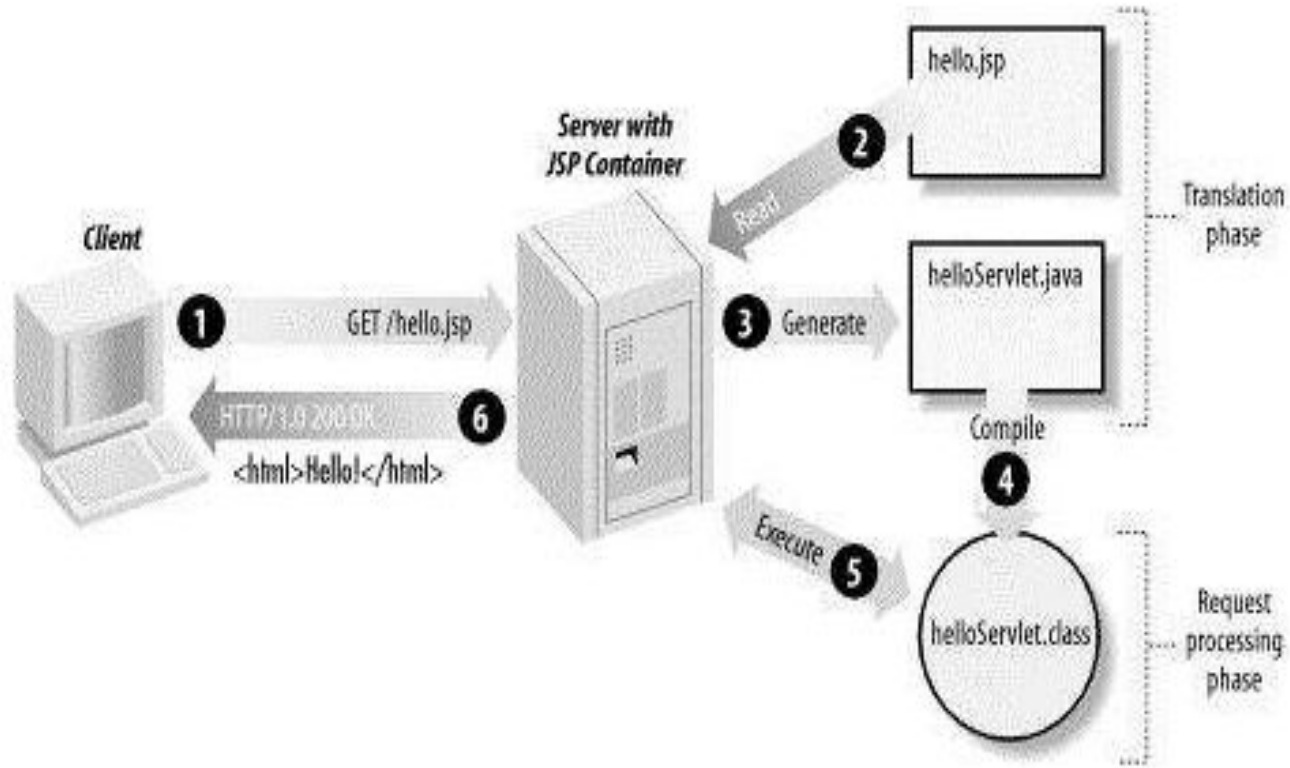
- The advantages of JSP compared to ASP are twofold.
  - First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use.
  - Second, it is portable to other operating systems and non-Microsoft Web servers.
- The advantages of JSP compared to Pure Servlets - It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
- The advantages of JSP compared to JavaScript - JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

JSP - Architecture

- The web server needs a JSP engine, i.e, a container to process JSP pages.
- The JSP container is responsible for intercepting requests for JSP pages.



JSP Processing



JSP Processing

- s with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is furthur passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.