1 Explain in brief various technologies provided by Java EE Platform

An enterprise application is a business application. Java EE is set of coordinated technologies and practices that enable developing, deploying and managing multi tier server centric enterprise applications.
It's a collection of standardized components; containers and services for creating and deploying distributed applications within well defined distributed computing architecture.

Following are the technologies provided by the Java EE platform:

1. **Web application technologies:**
   - ***Java Servlet:***
     A servlet is a program that extends the functionality of a web server. servlet receives a request from a client, dynamically generate the response and then the response containing an HTML or XML document.
   - ***JavaServer Pages:***
     JSP technology help us to generate dynamic content for a web client. A JSP page is text based document that contains two types of text:
     Static Data: can be expresses in any text based format such as HTML,WML and XML JSP elements: informs how the page constructs dynamoc content
   - ***JavaServer pages standard Tag Library[JSTL]:***
     It encapsulates core functionality of JSP applications. thie technology allows creating a standard set of tags,instead of mixing tags from numerous vendors in a JSP application. This standardization allows deploying application on any JSP server that supports JSTL.
   - ***JavaServer Faces-JSF:***
     It's a user interface framework for building web applications.
   - ***Facelets:***
     Facelets refers to JavaServer Faces ViewDefinition framework, which is a page declaration language developed for use with JavaServer Faces.it allows declaration of UI componenets in different presentation technologies.
2. **Enterprise Application Technologies:**
   - **Enterprise Java Beans-EJB**:
     An Enterprise JavavBean[EJB] componenet is an enterprise bean having methods to implement modules of business logic.
     they are scalable, transactional, secure and are building blocks of the enterprise applications running on JavaEE Server.
     EJB can be accessed locally and remotely through RMI/HTTP for SOAP and RESTful web services.
   - **Java Persistent API-JPA:**
     It's a Java standards-based API for Object-Relational Mapping[ORM], a solution for persistence.
     Java persistence consists of:
     - ✓ The Java Persistence API
     - ✓ The query language
   - **Java Transaction API**

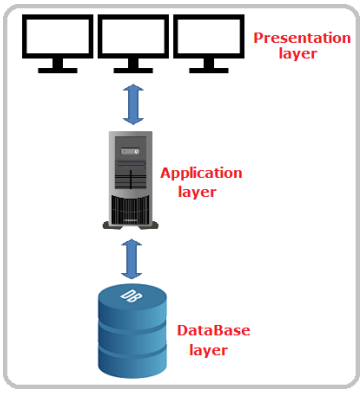
It provides a standard interface for democrating transactions. It provides default auto commit and rollback transactions.

- **Java Message Services[JMS]:**
  This is an API for messaging standard that allow Java EE application components to create,send,receive and read messages. It enables distributed coomunications that is loosely coupled, reliable and asynchronous.
- **Java Mail:**
  Java EE includes Java Mail with a service provider interface that allow application components to send emails.

3. **Web Services technologies:**
   - **SOAP and RESTful:**
     JAX-WS is the Java API for SOAP web services. JAX-RS is the java API for RESTFUL web services.
   - **Java API for XML Registries (JAXR) :**
     JAXR allows accessing business and general-purpose registries over the web.
4. **Security Services:**
   - **JACC:**
     Java Authorization Service Provider Contract for containers (JACC) specification defines a contract between a java EE application server and an authporization policy provider. JACC enables services to authenticate and enforce access controls upon users. All java EE containers support this contract.
   - **JASPIC:**
     The java Authentication Services Provider Interface for containers(JASPIC) specification defines a service provider interface(SPI) by which authenctication providers that implement message authentication mechanism may be integrated in client or server message processing containers or runtimes.

2 What are different types of System Architecture?

1. **Single Tier Architecture –**
   This is the type of architecture where the data and the application reside in the same machine. This is done by organizations which are small and are not geographically spread.



Advantages:

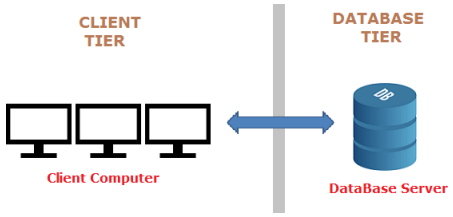
1) High security as the data and application exist in a single system.
2) The simplest and least expensive architecture
3) Less equipment to purchase and maintain.

Disadvantage:

1) Communication with other systems is not present.
2) These types of implementation are lower security and the lack of scalability.
3) Running all the site's components on a single computer also limits expansion and optimization possibilities

2. Two-tier Architecture:

In a modern two-tier architecture, the server holds both the application and the data. The application resides on the server rather than the client, probably because the server will have more processing power and disk space than the PC.



Advantages :

1) Communication is faster

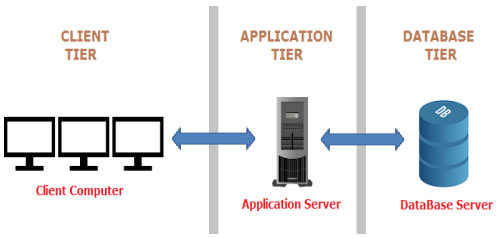
2) Removes data redundancy and helps different department to communicate via data

Disadvantage

1) 2-Tier architecture application performance will be degraded upon increasing the users beyond the 50-100 limit.

3. Three tier architecture:

In a three-tier architecture, the data and applications are split onto separate systems, with the server-side distributed between a database server and an application server. The client is a front end, simply requesting and displaying data. Reason being that each server will be dedicated to processing either data or application requests, hence a more manageable system and less contention for resources will occur.



Advantage:

1) High degree of flexibility in deployment platform and configuration

2) Improved Security – Client is not direct access to database

3) Enhanced scalability – Each tier can scale horizontally

Disadvantage : Increases complexity

4. Multi tier architecture:

Java EE architecture is based on the notion of multitier applications. In this application logic is divided by its functionalities:

- User interface tier:
  It handles user interface with user's interaction with the application. It could be a web browser run through firewall, a heavier desktop or even wireless device
- Presentation tier:
  Defines what user interface displays and how a users requests are handled depends upon what user interfaces are supported. This layer provides an interface for the end user to the application data manipulation code spec.

- Business Tier:
  It contains business logic of the application. This layer doesn't know anything neither about HTML nor about its output. It should not have any code to access database/tables or SQL. These tasks are assigned to corresponding layers wither above or below the business tier.
- Infrastructure services:
  Provides additional functionality which are required by the application components such as messaging, transactional support and so on
- Integration tier/data access tier:
  It is responsible for providing access to backend resources including powerful relational databases and external systems. This layer doesn't contain any business rules or data manipulation or transformation logic. It is reusable interface to a database engine and through the engine business data stored in related tables.
- Data tier:
  Data tier layer could be a DBMS such a as SQL Server, M.S.Access, Oracle, MySQL, Plain text files[binary files]and so on. This layer is only used to deal with storage and retrieval of business data.

5. Enterprise Architecture:

Enterprise architecture is the extension of the concept of multi tier architecture and available technologies resulting in a custom made platform that can run across hardware and software frameworks found across an enterprise, often spanning geographical boundaries.

Application logic is divided into components according to functions and the various application components put together make up the enterprise application. These components are physically installed on different computers, at different physical locations, depending on the tier in the multi tiered environment to which the application component belongs. All components work together across enterprise.

Enterprise architecture is divided into the following tiers:

- **Client Tier**: Services run on the Client Machine
- **Web Tier**: Services run on the Java EE Server
- **Business Tier:** Services run on the Java EE Server
- **Enterprise Information System [EIS]:** software run on the Database Server Machine.

1. The Client Tier:

The Client tier consists of a client program that makes requests to the Web Tier. The client can be:

- ✓ A web client [Dynamic HTML pages] It consists of :
  - Dynamic web pages containing various types of markup language such as HTML,XML and so on, which are generated by wen components running in the web tier
  - A web browser, which renders the pages received from the server
  - Web client do not query databases, do not execute complex business rules or connect to legacy applications.
- ✓ An Application Client :
  - An application client runs on a client machine and provides a way for

users to handle tasks that require a richer user interface than cannot be provided by a markup language.

- It typically has GUI created from swing or Abstract Windows Toolkit API but a command line interface can also be used here.
- Application clients directly access enterprise beans running in the business tier. An application client can open an HTTP connection to establish communication with a servlet running in the web tier. Application clients written in language other than Java can interact with Java EE Servers, enabling the Java EE platform to interoperate with legacy systems clients and non java language.

2. **The Web Tier:**

The web tier consists of components that handle the interaction between clients and the business tier. The web tier usually performs the following tasks:

- ✓ Dynamically generate content in various formats for the client.
- ✓ Collect input from the client and return appropriate results from the components in the Business tier
- ✓ Control the flow of screens or pages on the client
- ✓ Maintain the state of data for a users session
- ✓ Perform some basic logic and hold some data temporarily in JavaBeans

components We can use are servlets,JSP,JSF,JSF facelets,Expression Language,JSTL and Java Bean Components.

3. **The Business Tier:**

It consists of components that provide the business logic for an application.
It also provides functionality to a particular business domain such as banking, retail, finance or an e-commerce site. In java EE, the business logic is handles by Enterprise Beans. Enterprise bean receive data from the client program,processes it, if necessary, sends it to the Enterprise Information System Tier for Storage, retrieves data from storage and send it back to client program.
Following java EE technologies are used in the business tier:

- **Enterprise Java Beans[EJB]**

  JavaPersistence API [JPA is a standard API for Object Relational Mapping[ORM]. with its Java Persistence Query Language[JPQL], objects stores in the underlying database can be queries.
- **The Enterprise Information Systems [EIS] Tier:**
  This layer consists of database servers, enterprise resource planning systems, Mainframe transaction processing and other legacy information systems. These resources are typically are located on a separate machine than the Java EE server and are accessed by components on the business tier.
  The following java EE technologies are used in the EIS tier: Java database connectivity, JPA, JPQA, Java EE Container Architecture, JTA.

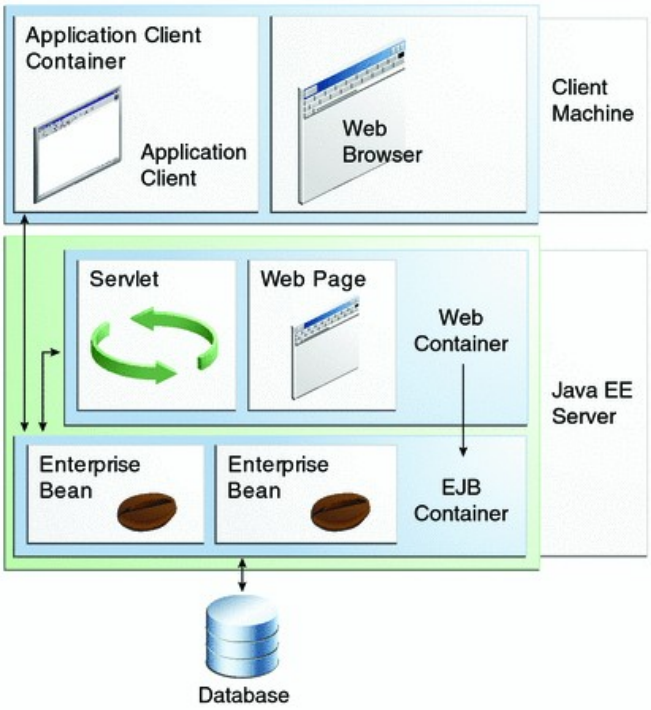
4. What are various containers available in Java EE6? Describe features.

The component-based and platform-independent Java EE architecture makes Java EE applications easy to write because business logic is organized into reusable components. In addition, the Java EE server provides underlying services in the form of a container for every component type. Because you do not have to develop these services yourself, you are free to concentrate on solving the business problem at hand.

Containers are the interface between a component and the low-level platform-specific functionality that supports the component. Before it can be executed, a web, enterprise bean, or application client component must be assembled into a Java EE module and deployed into its container.

The assembly process involves specifying container settings for each component in the Java EE application and for the Java EE application itself. Container settings customize the underlying support provided by the Java EE server, including such services as security, transaction management, Java Naming and Directory Interface (JNDI) API lookups, and remote connectivity. Here are some of the highlights.

- The Java EE security model lets you configure a web component or enterprise bean so that system resources are accessed only by authorized users.
- The Java EE 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.
- JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access these services.
- The Java EE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.
- Because the Java EE architecture provides configurable services, application components within the same Java EE application can behave differently based on where they are deployed. For example, an enterprise bean can have security settings that allow it a certain level of access to database data in one production environment and another level of database access in another production environment.
- The container also manages non-configurable services, such as enterprise bean and servlet lifecycles, database connection resource pooling, data persistence, and access to the Java EE platform APIs



The **deployment** process installs Java EE application components in the Java EE containers as illustrated above:

- **Java EE server**: The runtime portion of a Java EE product. A Java EE server provides EJB and web containers.
- **Enterprise JavaBeans (EJB) container**: Manages the execution of enterprise beans for Java EE applications. Enterprise beans and their container run on the Java EE server.
- **Web container**: Manages the execution of web pages, servlets, and some EJB components for Java EE applications. Web components and their container run on the Java EE server.
- **Application client container**: Manages the execution of application client components. Application clients and their container run on the client.
- **Applet container**: Manages the execution of applets. Consists of a web browser and Java Plug- in running on the client together.

5 Write a note on Glass Fish Server.

- GlassFish is a Java application server project created by Sun Microsystems that allows many developers to generate enterprise technologies that are convenient and scalable, as well as additional services that can be installed based on preference. It is a free, dual-licensed software under the GNU General Public License (GPL) and the Common Development and Distribution License (CDDL). GlassFish was acquired by Oracle in 2010.
- GlassFish is the reference implementation of Java EE and as such supports Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, servlets, etc. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. Optional components can also be installed for additional services.

6. What are advantages of servlet over CGI?

Advantages of Servlets over CGI:

- **Platform independence:**
  Servlets run on top of Java Virtual Machine (JVM) so they are platform independent components. On the other hand CGIs run on the operating system itself so they are platform dependent.
- **Performance**
  CGIs run on separate processes, so it takes more time to start a new process for each and every requests. Servlets are accessed through threads and those threads also being kept in a thread pool. It increases the performance.
- **Security**
  Servlets are running inside the sand box of JVM, so it is hard to damage the server side modules by malfunctioning the Servlets. Since CGIs are native applications, using CGIs a hacker can damage the server side components easily compared to Servlets.
- **ErrorHandling**
  Servlets provides easiest error handling mechanism jointly with web container. CGIs do not have such a powerful error handling support.
- **Extensibility**
  Servlets are just Java classes so it is easy maintain and extend their functionality. All the object oriented concepts can be directly applied to Servlets as well. CGIs are mostly written in scripting languages like Perl so the extensibility is very poor in case of CGIs.

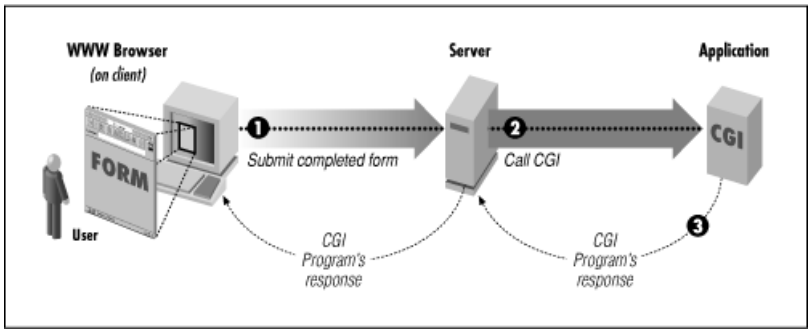
7 Differentiate CGI and Servlet

CGI	Servlet
It is a process based. i.e., for every request, a new process will be created and that process is responsible to generate required response.	It is thread based. i.e., for every request new thread will be created and that thread is responsible to generate required response.
Creation and destruction of new process for every request is costly, if the number of requests increases then the performance of the system goes down. Hence CGI technology fails to deliver scalable applications.	Creation and destruction of a new thread for every request is not costly, hence if the number of requests increases there is no change in response time. Due to this, servlet technology succeeds to deliver scalable applications.
Two processes never share common address space. Hence there is no chance of occurring concurrence problems in CGI	All the threads shares the same address space, Hence concurrence problem is very common in servlets.
We can write CGI program in variety of languages, But most popular language is perl.	We can write servlets only in java.
Most of the CGI languages are not object oriented. Hence we miss the benefits of oops.	Java language itself is object oriented. Hence we can get all the key benefits of oops.
Most of CGI languages are platform dependent.	Java language is platform independent.

8 What is Servlet? Write Short note on Java Servlet Technology.

Servlet: It's a module written using Java that runs in a server application to answer client requests.

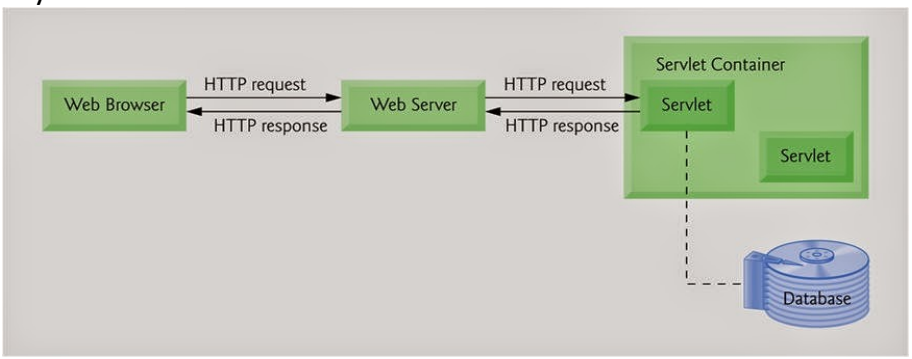
- It's most commonly used with HTTP and the word servlet is often used in the meaning of HTTP servlet.
- It's supported by virtually all web servers and application servers
- Can easily share resources



9 What is Servlet? Write a short note on Java Servlet Technology. Describe Servlet Architecture.

Servlet:

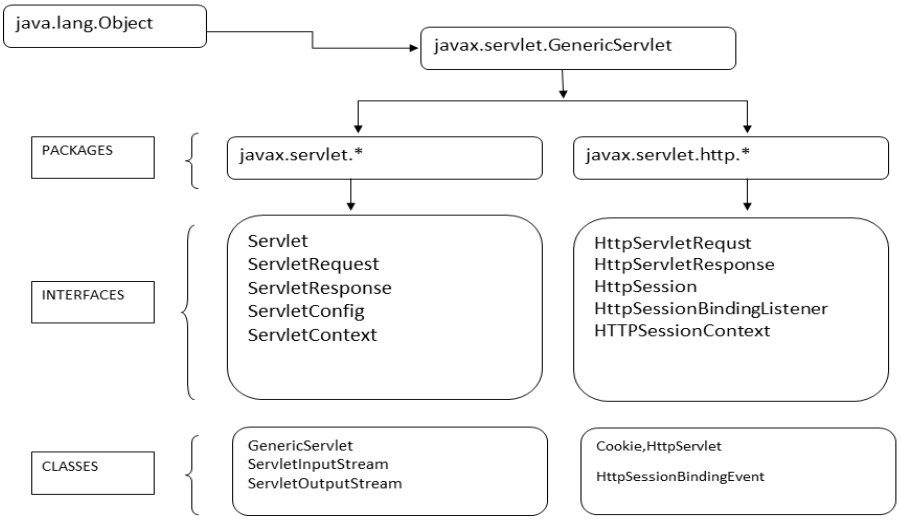
- It's a module written using Java that runs in a server application to answer client requests.
- It's most commonly used with HTTP and the word servlet is often used in the meaning of HTTP servlet
- It's supported by virtually all web servers and application servers
- Can easily share resources



Servlet Technology :

- Servlets 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), setting cookies and caching parameters, and other such tasks.

Servlet Architecture:



Servlet architecture made by two packages:

1. javax.servlet.Servlet
2. javax.servlet.http

The **javax.servlet.Servlet** package contains the generic interfaces and Classes that are implemented and extended by all **Servlets**. The **javax.servlet.http** package contains the Classes that are extended when creating **HTTP-specific Servlets**.

The main root of the **Servlet Architecture** is the **javax.servlet.Servlet**. it has following interfaces:

- Servlet
- ServletRequest
- ServletResponse
- RequestDispatcher
- ServletConfig
- ServletContext
- SingleThreadModel
- Filter
- FilterConfig

Classes included in javax.servlet package are as follows:

- GenericServlet
- ServletInputStream
- ServletOutputStream
- ServletRequestWrapper
- ServletResponseWrapper
- ServletRequestEvent
- ServletContextEvent
- ServletRequestAttributeEvent
- ServletContextAttributeEvent
- ServletException

Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. They are as follows:

- HttpServletRequest
- HttpServletResponse
- HttpSession
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSessionBindingListener
- HttpSessionActivationListener
- HttpSessionContext (deprecated now)
- Classes in javax.servlet.http package

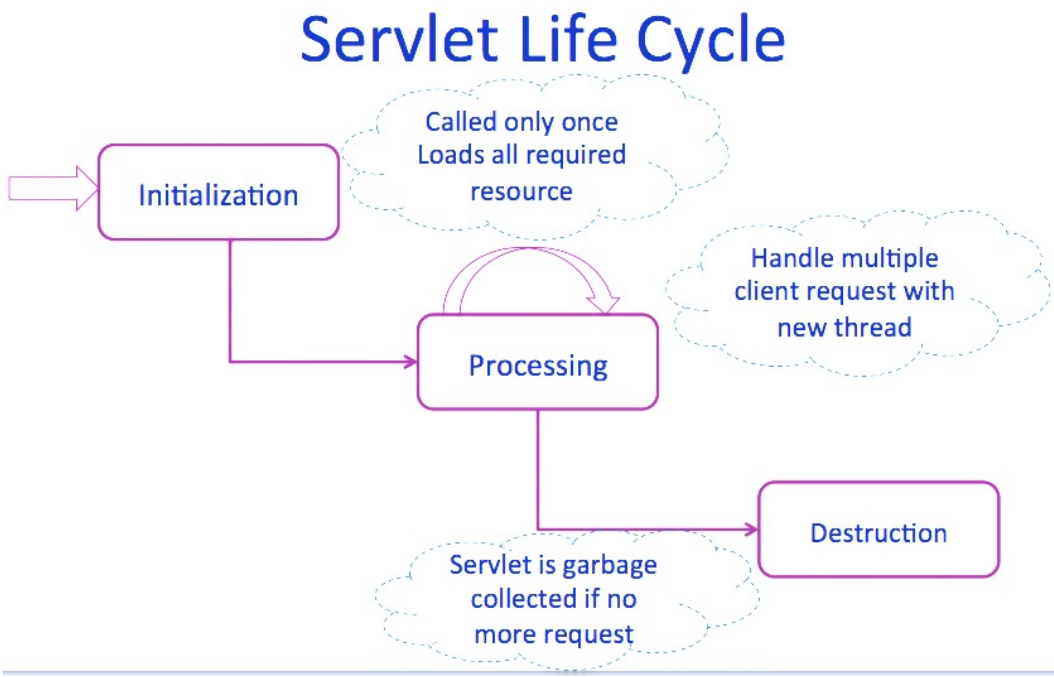
There are many classes in javax.servlet.http package. They are as follows:

- HttpServlet
- Cookie
- HttpServletRequestWrapper
- HttpServletResponseWrapper
- HttpSessionEvent
- HttpSessionBindingEvent
- HttpUtils (deprecated now)

10. Describe servlet life cycle.

A servlet life cycle can be defined as the entire process from its creation till the destruction. The lifecycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the following steps:

- If an instance of the servlet does not exist, the web container
  - Loads the servlet class.
  - Creates an instance of the servlet class.
  - Initializes the servlet instance by calling the init method.
- Invokes the service method, passing request and response objects. If it needs to remove the servlet, the container finalizes the servlet by calling the servlet's destroy method.



1. **Loading a Servlet**:
   The first stage of the Servlet lifecycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages. The Servlet container performs two operations in this stage :
   - Loading : Loads the Servlet class.
   - Instantiation : Creates an instance of the Servlet. To create a new instance of the Servlet, the container uses the no-argument constructor.
2. **Initializing a Servlet**:
   - After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking

     the Servlet.init(ServletConfig) method which accepts ServletConfig object reference as parameter.
   - The Servlet container invokes the Servlet.init(ServletConfig) method only once, immediately after the Servlet.init(ServletConfig) object is instantiated successfully. This method is used to initialize the resources, such as JDBC datasource.
   - Now, if the Servlet fails to initialize, then it informs the Servlet container by throwing the ServletException or UnavailableException.

   .
3. **Handling request:**
   - After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :
   - It creates the ServletRequest and ServletResponse objects. In this case, if this is a HTTP request then the Web container creates HttpServletRequest and HttpServletResponse objects which are subtypes of the ServletRequest and ServletResponse objects respectively.
   - After creating the request and response objects it invoke the Servlet.service(ServletRequest, ServletResponse) method by passing the request and response objects.
   - The service() method while processing the request may throw the ServletException or UnavailableException or IOException.
4. **Destroying a Servlet:**
   - When a Servlet container decides to destroy the Servlet, it performs the following operations,
   - It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
   - After currently running threads have completed their jobs, the Servlet container calls the destroy() method on the Servlet instance.
   - After the destroy() method is executed, the Servlet conatiner releases all the references of this Servlet instance so that it becomes eligible for garbage collection.

11 Write a note on RequestDispatcher

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. It is one of the way of servlet collaboration.

To obtain the object of RequestDispatcher:-

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher. Syntax:

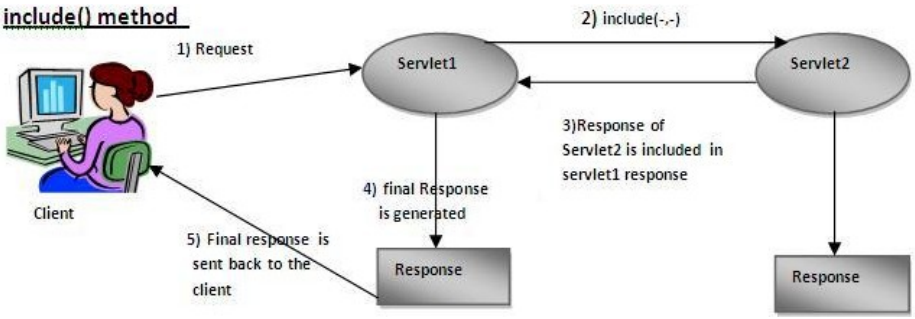
- public RequestDispatcher getRequestDispatcher(String resource);

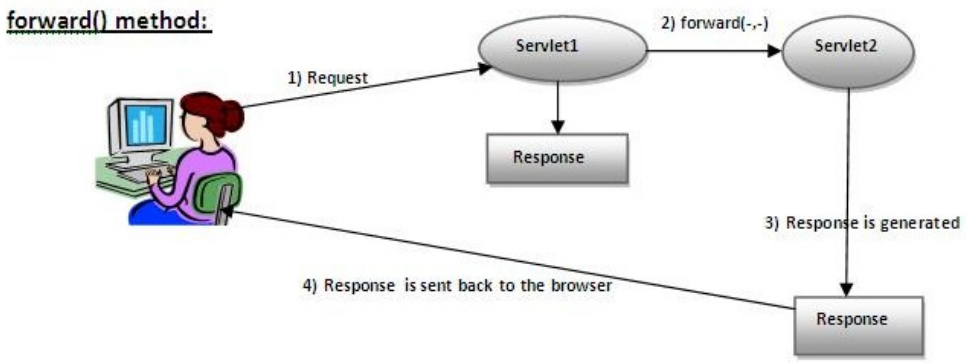
Example of using getRequestDispatcher method

- RequestDispatcher rd=request.getRequestDispatcher("servlet2");
- //servlet2 is the url-pattern of the second servlet

There are two methods defined in the RequestDispatcher interface.

- **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:** Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:** Includes the content of a resource (servlet, JSP page, or HTML file) in the response.





index.html

```
<form action="servlet1" method="post">
Name:<input type="text"
name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
```

Login.java

```
import
java.io.*;
import
javax.servlet.*;
import
javax.servlet.http.*;
public class Login extends HttpServlet {
 public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletExcep tion, IOException {
   response.setContentType("text/htm
   l"); PrintWriter out =
   response.getWriter(); String
   n=request.getParameter("userNam
   e"); String
   p=request.getParameter("userPass"
   );

   if(p.equals("servlet"){
      RequestDispatcher rd=request.getRequestDispatcher("servlet2");
      rd.forward(request, response);
```

```
    }
    else{
      out.print("Sorry UserName or Password Error!");
      RequestDispatcher rd=request.getRequestDispatcher("/index.html");
      rd.include(request, response);
     }
    }
}
```

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

```
import javax.servlet.http.*;
public class WelcomeServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletExc eption, IOException {
   response.setContentType("text/htm
   l"); PrintWriter out =
   response.getWriter(); String
   n=request.getParameter("userNam
   e"); out.print("Welcome "+n);
   }
}
```

12 Differentiate ServletConfig and Servlet Context

ServletConfig	ServletContext
• 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 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

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked**	Post request **cannot be** bookmarked
4) Get request is **idempotent**. It means second request will be ignored until response of first request is delivered.	Post request is **non-idempotent**
5) Get request is **more efficient** and used more than Post	Post request is **less efficient** and used less than get.

	GenericServlet	HTTPServlet
1	Generic Servlet class is super class/parent class of HTTP servlet in servlet architecture.	HTTP servlet class is sub class/child class of Generic Servlet class in servlet architecture.
2	The javax.servlet package contains the generic interfaces and classes that are implemented &extended by the servlets.	The javax.servlet.http package contains the classes that are extended when creating http specific servlets.
3	This class implements Servlet interface.	This class extends Generic Servlet class.
4	This class implements Servlet Config which handles initialization parameter & Servlet Context.	This class implements service() of generic class by calling method specific implementation to http request method(i.e. doGet() and doPost()).
5	Servlet that extends this class are protocol independent. They do not contain any support for HTTP or other protocol.	Servlet that extends this class have built-in support for HTTP protocol.
6	The GenericServlet does not include protocol-specific methods for handling request parameters, cookies, sessions and setting response headers.	The HttpServlet subclass passes generic service method requests to the relevant doGet() or doPost() method.
7	GenericServlet is not specific to any protocol.	HttpServlet only supports HTTP and HTTPS protocol.
8	javax.servlet.GenericServlet	javax.servlet.http.HttpServlet
9	GenericServlet defines a generic, protocol-independent servlet.	HttpServlet defines a HTTP protocol specific servlet.