**Q1 LifeCycle of jsp**

When jsp is run on web server, it is converted into Servlet.

When jsp is requested for the first time :

1) jsp is converted into .java (servlet)

2) servlet is compiled by a web container.

3) .class is loaded

4) Instantiated using “public no-arg constructor”.

5) Initialized by calling “init() “ method.

6) Thread is created or retrieved from a thread pool.

7) request and response are created

8) service() method is called.

For subsequent requests to same jsp , provided jsp source code is not modified :

1) Thread is created or retrieved from a thread pool.

2) request and response are created

3) service() method is called.

LifeCycle methods of Jsp

public void jspInit(){}

public void \_jspService(HttpServletRequest request, HttpServletResponse response) {}

public void jspDestroy(){}

out of above methods , if you want to override, you can override only “jspInit()” and “jspDestroy()”.

You can not override “\_jspService()”. But you can write a code in your jsp, which can go to generated “\_jspService()” method of servlet.

**Q3) What are implicit EL Objects in jsp**

EL provides various implicit objects

pageScope- to read page attributes

requestScope - to read request attributes

sessionScope- to read session attributes

applicationScope- to read context attributes

param- to read request parameters

paramValues- to read request parameters returning more than one value

cookie- to read cookies

initParam- to read context parameters

All the EL implicit objects are implicitly maps.

**Q4) Difference between redirect and forward (include ka nai milra)**

redirect

can go beyond web container.

it is slower

client clearly knows who gives the final response.

different requests.

forward

within the web container.

it is faster than redirect.

client is under the impression that the first servlet/jsp has sent the response.

same requests.

**6)** **Difference between GenericServlet and HttpServlet**

GenericServlet is derived from Servlet interface

HttpServlet is derived from GenericServlet

GenericServlet has not defined all methods of Servlet so it's an Abstract class.

HttpServlet doesn't have any abstract method but still its Abstract because they doesn't want to allow programmer to instantiate it, but allow inheritance.

We extends GenericServlet to handle any request eg. ftp,http etc.

We extends HttpServlet to handle only http request

22) **get vs load**

Here are few differences between get and load method in Hibernate.

**1. Behavior when Object is not found in Session Cache**

Apart from performance this is another difference between get and load which is worth remembering. get method of Hibernate Session class returns null if object is not found in cache as well as on database while load() method throws ObjectNotFoundException if object is not found on cache as well as on database but never return null.

**2. Database hit**

Get method always hit database while load() method may not always hit the database.

**3. Proxy**

Get method never returns a proxy, it either returns null or fully initialized Object, while load() method may return proxy, which is the object with ID but without initializing other properties, which is lazily initialized. If you are just using returned object for creating relationship and only need Id then load() is the way to go.

**4. Performance**

get method will return a completely initialized object if Object is not on the cache but exists on Database, which may involve multiple round-trips to database based upon object relational mappings while load() method of Hibernate can return a **proxy** which can be initialized on demand (lazy initialization) when a non identifier method is accessed. Due to above reason use of load method will result in slightly **better performance**, but there is a possibility that proxy object will throw ObjectNotFoundException later if corresponding row doesn’t exists in database.

**When to use Session get() and load() in Hibernate**

1. Use get method to determine if an instance exists or not because it can return null if instance doesn’t exists in cache and database and use load method to retrieve instance only if you think that instance should exists and non availability is an error condition.

2. get() method could suffer performance penalty if only identifier method like getId() is accessed. So consider using load method if your code doesn't access any method other than identifier or you are OK with lazy initialization of object, if persistent object is not in Session Cache because load() can return proxy.

**Q5 Explain LifeCycle of Servlet**

servlet life cycle :- since servlet is going to run inside web container, it has life cycle. Following is the life cycle of servlet

**public void init()**

it is called only once.

it can be used to

create database connection

lookup remote object or any ejb component.

**public void service(HttpServletRequest,HttpServletResponse)**

it is called whenever a new request comes to the servlet and then in turn it invokes either "doGet" or "doPost" methods depending upon the type of request.

**get**

public void doGet(HttpServletRequest,HttpServletResponse)

**post**

public void doPost(HttpServletRequest,HttpServletResponse)

**public void destroy()**

it is called only when

u shutdown the server

or

u undeploy the application.

**23) Advantages OF Spring**

In java we do:

Pojo based programming

jdbc

ORM tools

Enterprise Applications

[RMI,EJB]

MVC based Applications

Spring framework was released by Rod Johnson in 2005 which provides a great support for all the above mentioned Applications.

Spring is light weight.

If Developers do not need to use all of its services, they should be allowed to choose the ones they desire to use and implement.

Spring achieves the loose coupling through dependency injection and interface based programming.

Declarative programming through aspects.

Boilerplate reduction through templates.

**25) What is maven dependency?**

In Maven, dependency is another archive—JAR, ZIP, and so on—which your current project needs in order to compile, build, test, and/or to run. The dependencies are gathered in the pom.xml file, inside of a <dependencies> tag.

When you run a build or execute a maven goal, these dependencies are resolved, and are then loaded from the local repository. If they are not present there, then Maven will download them from a remote repository and store them in the local repository. You are allowed to manually install the dependencies as well.

**Q15) Hibernate entity lifecycle**

In Hibernate, either we create an object of an entity and save it into the database, or we fetch the data of an entity from the database. Here, each entity is associated with the lifecycle. The entity object passes through the different stages of the lifecycle.

The Hibernate lifecycle contains the following states: -

● Transient state

● Persistent state

● Detached state

**Q16) What is IOC (or Dependency Injection)? What are advantages? What are Types of DI?**

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled. To understand the DI better, Let's understand the Dependency Lookup (DL) first:

There are mainly two problems of dependency lookup.

● **tight coupling** The dependency lookup approach makes the code tightly coupled. If resource is changed, we need to perform a lot of modification in the code.

● **Not easy for testing** This approach creates a lot of problems while testing the application especially in black box testing.

### **Two ways to perform Dependency Injection in Spring framework**

Spring framework provides two ways to inject dependency

● By Constructor

● By Setter method

*BeanFactory*

*ApplicationContext*

ClassPathXMLApplicationContext

**Q17) Difference between BeanFactory and the ApplicationContext**

**BeanFactory** and **ApplicationContext** interfaces acts as the **IoC container**. The ApplicationContext interface is built on top of the BeanFactory interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application. So it is better to use ApplicationContext than BeanFactory.

**Q18) Explain Spring MVC architecture?**

A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

## **Spring Web Model-View-Controller**

● **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.

● **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.

● **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.

● **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

**30) What is DAO?**

**DAO ( Data Access Object) design pattern**

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. Following are the participants in Data Access Object Pattern.

**Data Access Object Interface** - This interface defines the standard operations to be performed on a model object(s).

**Data Access Object concrete class** - This class implements above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.

**Model Object or Value Object** - This object is simple POJO containing get/set methods to store data retrieved using DAO class.

In Short, DAO decouples business layer from a particular persistence mechanism

[ file,database etc]. So business layer can concentrate only on business logic.

**31) What is MVC?**

**MODEL**

In the MVC architecture, model components provide an interface to the data and / or services used by an application. This way, controller components don’t unnecessarily embed code for manipulating an application’s data. Instead , they communicate with the model components that perform the data access and manipulation. Thus the model component provides the business logic. Model components come in many different forms and can be as simple as a basic Java bean or as complex as Enterprise JavaBeans (EJBs) or Web Services.

**VIEW COMPONENTS**

View components are used in the MVC architecture to generate the response to the browser. Thus, a view component provides what the user sees. Often times the view components are simple JSPs or HTML pages.

**CONTROLLER COMPONENTS**

At the core of the MVC architecture are the controller components. The Controller is typically a servlet that receives requests for the application and manages the flow of data bet’n the Model layer and the View layer. Thus, it controls the way that the Model and View layers interact. The Controller often uses helper classes for delegating control over specific requests or processes

**32) What is ORM?**

The object-oriented model use **classes** whereas the relational databases use **tables**. Getting the data and associations from objects into relational table structure and vice-versa requires a lot of tedious programming due to the difference between the two. This difference is called The **Impedance Mismatch**. Developers need something simple to covert from one to the other automatically. Bridging the gap between the object m`kodel and the relational model is known as **Object –Relational Mapping. (ORM)**

**34) How does servlet work?**

when u say

http://localhost:8080/myapp/FirstServ , for the first time,

what exactly happens

1) web container opens ur DD (web.xml)

2) it searches the url-pattern by the name "/FirstServ"

3) from url-pattern it finds out servlet-name .

4) from servlet-name, it finds out servlet-class.

5) once it gets the name of servlet-class, it tries to find out that .class file in "classes" folder of "myapp"

6)now it loads "FirstServ.class"

7) instantiate "FirstServ" by invoking its "public no-arg constructor".

8) it will invoke "init" method.

9) thread is created or retrieved from thread pool.

10) HttpServletRequest and HttpServletResponse are created.

11) service() method is called.

service() method checks whether request is "get" or "post". if it is "get" service() method invokes "doGet()" and if it is "post" it invokes "doPost()"

since we are running servlet directly it is "get" request , so service() method will call "doGet()" which we have overridden.

Imp. note:- when u refresh above link, i.e. for subsequent requests ( without makings changes in source code)

**9) 10) and 11) steps will be performed.**

**Q11) What is hibernate dialect**

hibernate.dialect

This property makes Hibernate generate the appropriate SQL for the chosen database.

**Q18.Explain spring MVC architecture with diagram.**

A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

## **Spring Web Model-View-Controller**

● **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.

● **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.

● **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.

● **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

## **Understanding the flow of Spring Web MVC**

● As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller.

● The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller.

● The controller returns an object of ModelAndView.

● The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

**Q10 What is Hibernate? Explain hibernate spring architecture with diagram**

Hibernate is java-based middleware designed to complete the Object Relational (O/R) mapping model.

One of the most complex parts of many O/R mapping mechanisms- writing SQL code- can be simplified significantly with Hibernate.

Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBCs

Hibernate makes use of persistent objects commonly called as POJO (POJO = "Plain Old Java Object".) along with XML mapping documents for persisting objects to the database layer. The term POJO refers to a normal Java objects that does not serve any other special role or implement any special interfaces of any of the Java frameworks (EJB, JDBC etc...).

Rather than utilize byte code processing or code generation, Hibernate uses runtime reflection to determine the persistent properties of a class. The objects to be persisted are defined in a mapping document, which serves to describe the persistent fields of the persistent object. The mapping documents are compiled at application startup time and provide the framework with necessary information for a class.

**Q 23. What are the advantages of spring framework?**

## **Spring Pros**

There are the following advantages of the Spring framework:

1. **Light Weight:** Spring is a lightweight framework because of its POJO implementation. It does not force the programmer to inherit any class and implement any interface. With the help of Spring, we can enable powerful, scalable applications using POJOs (Plain Old Java Object).
2. **Flexible:** It provides flexible libraries trusted by developers all over the world. The developer can choose either XML or Java-based annotations for configuration options. The IoC and DI features provide the foundation for a wide-ranging set of features and functionality. It makes the job simpler.
3. **Loose Coupling:** Spring applications are loosely coupled because of dependency injection. It handles injecting dependent components without a component knowing where they came from.
4. **Powerful Abstraction:** It provides a powerful abstraction to JEE specifications such as JMS, JDBC, JPA, and JTA.
5. **Declarative Support:** It provides declarative support for caching, validation, transaction, and formatting.
6. **Portable:** We can use server-side in web/EJB app, client-side in swing app business logic is completely portable.
7. **Cross-cutting behavior:** Resource management is a cross-cutting concern, easy to copy and paste everywhere.
8. **Configuration:** It provides a consistent way of configuring everything, separate configuration from application logic, varying configuration.
9. **Lifecycle:** Responsible for managing all your application components, particularly those in middle-tier container sees components through well-defined lifecycle: init(), destroy().
10. **Dependency Injection:** The use of dependency injection makes the easy development of JavaEE.
11. **Easier Testing:** The use of dependency injection makes the testing easy. The spring framework does not require a server while the EJB and Struts application requires a server.
12. **Fast:** The team of Spring engineers deeply cares about the performance. Its fast startup, fast shutdown, and optimized execution maintain performance make it fast. Even, we can start a new Spring project in seconds by using Spring Initializr.
13. **Secure:** It monitors third-party dependencies closely. The regular update is issued that make our data and application secure. We can make our application secure by using the Spring Security framework. It provides industry-standard security schemes and delivers a trustworthy solution that is secure by default.
14. **Supportive:** The Spring community provides support and resources to get you to the next level QuickStart guides, tutorials, videos, and meetup helps a lot.
15. **Productive:** It is more productive because the spring application can integrate with other Spring-based applications. For example, we can combine the Spring Boot application with Spring Cloud.

**Q13 What is advantage of hibernate over jdbc?**

Advantages of hibernates:

* Hibernate supports Inheritance, Associations, Collections
* In hibernate if we save the derived class object, then its base class object will also be stored into the database, it means hibernate supporting inheritance
* Hibernate supports relationships like One-To-Many, One-To-One, Many-To-Many-to-Many, Many-To-One
* This will also supports collections like List,Set,Map (Only new collections)
* In jdbc all exceptions are checked exceptions, so we must write code in try, catch and throws, but in hibernate we only have Un-checked exceptions, so no need to write try, catch, or no need to write throws. Actually, in hibernate we have the translator which converts checked to Un-checked
* Hibernate has capability to generate primary keys automatically while we are storing the records into database
* Hibernate has its own query language, i.e hibernate query language which is database independent So if we change the database, then also our application will works as HQL is database independent
* HQL contains database independent commands
* While we are inserting any record, if we don’t have any particular table in the database, JDBC will rises an error like “View not exist”, and throws exception, but in case of hibernate, if it not found any table in the database this will create the table for us
* Hibernate supports caching mechanism by this, the number of round trips between an application and the database will be reduced, by using this caching technique an application performance will be increased automatically.
* Hibernate supports annotations, apart from XML
* Hibernate provided Dialect classes, so we no need to write sql queries in hibernate, instead we use the methods provided by that API.
* Getting pagination in hibernate is quite simple.

**Q17 What is difference between BeanFactory and ApplicationContext?**

**BeanFactory** and **ApplicationContext** interfaces acts as the **IoC container**. The ApplicationContext interface is built on top of the BeanFactory interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application. So it is better to use ApplicationContext than BeanFactory.

**Q16 What is IOC(or dependency injection(DI)) What are advantages? Types of DI?**

It is one of the best ways to implement loosely coupled applications. Using DI, the framework acts as an object factory to build service objects and injects those service objects to application POJOs, based on some runtime configuration. From the application developer’s point of view, the client POJO automatically obtains the correct service object, when it is required

DI makes the code significantly simpler, easier to understand and easier to test. The **IOC container** manages the POJOs and its dependencies in the application. The application defines the dependency and these dependencies are then satisfied at runtime by the IOC container as follows:

oA caller asks the container for an object with a specific name or of a specific type.

oThe container injects these objects by other objects, via either **constructors**, or **properties**.

Objects are not expected to create or obtain their dependencies. Dependencies are injected into the objects that need them.

Classes are no longer required to be coupled to any specific implementation, which is the key benefit of DI , Loose Coupling. Since an object only knows about its dependencies by their interface, the dependency can thus be swapped out with a different implementation without the depending object knowing the difference.

How does DI work ?

In spring , there are many ways to wire components together , but a common approach has always been **via XML**. Typically , the container is configured by loading XML files with some runtime configuration containing bean definitions which provide the information required to create the beans.

Types of DI

**Constructor-arg for Constructor Injection**

<constructor-arg> is used to give Spring additional information to use when constructing a bean. If no <constructor-arg>s are given , the default constructor is used.

This process indicates how the constructor argument is injected into the bean and is widely known as **Constructor Injection**.

**Property injection**

Usually properties of the javabeans are private. Every property has a pair of accessor methods in the form of setXXX() and getXXX().

Values can be injected into these bean properties using the Setter Injection with the help of <property>. <property> is similar to <constructor-arg> in many ways, except that instead of injecting values through a constructor argument, <property> is injected by calling a property’s setter method.

**Q19) How is session management done in servlet-JSP?**

**Session** simply means a particular interval of time.

**Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.

Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:

### **Why use Session Tracking?**

**To recognize the user** It is used to recognize the particular user.

### **Session Tracking Techniques**

There are four techniques used in Session tracking:

Triggers in SQL (Hindi)

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

**What is JDBC**

java database Connectivity. using jdbc, java application can connect to any database such as MySQL,Oracle,SQLServer etc.

jdbc provides two things:

a) jdbc API [classes and interfaces] which remains same no matter which database you use.

b) jdbc drivers

driver is a software to connect to database

There are 4 types of drivers in jdbc:

Type 1 - which requires ODBC to be installed on a machine. It has almost become obsolete.

Type 2 - which requires database client library to be installed on client machine. cannot be used for internet purpose.

Type 3 - which requires middleware (Application server) to get connected to database.

Type 4 - which is the fastest driver as it directly connects to the database.

Steps required for jdbc application:

a) load and register the driver with DriverManager.

b) get connection with database

c) communicate with database with the help of JDBC API.

**Scrollable and Unscrollable statement?**

By default ResultSet is readonly and forward only. It means while traversing through the ResultSet we cannot modify records and we have only one method for traversing i.e. "next".

But we can make ResultSet updatable ( while traversing through the ResultSet we can modify records) and scrollable ( not only "next" we can invoke "previous" or "absolute" method also )

**Syntax difference between Statememt and Prepared Statement**

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from dept");

vs

PreparedStatement pst=con.prepareStatement("select \* from dept");

ResultSet rs=pst.executeQuery();

**About Prepared Statement**

PreparedStatement has 2 advantages:

a) u can use placeholder so that values can be passed dynamically

b) PreparedStatement contains precompiled sql statements which enable faster execution.

suppose we want to issue same query again and again:

Statement st=con.createStatement();

st.executeQuery("select \* from dept");

a) query goes to database

b) it gets parsed,compiled

c) execution plan is made

d) query gets executed.

st.executeQuery("select \* from dept");

a) query goes to database

b) it gets parsed,compiled

c) execution plan is made

d) query gets executed.

st.executeQuery("select \* from dept");

a) query goes to database

b) it gets parsed,compiled

c) execution plan is made

d) query gets executed.

if we use PrepareStatement,

PrepareStatement pst=con.prepareStatement("select \* from dept");

a) query goes to database

b) it gets parsed,compiled

c) execution plan is made

pst.executeQuery();

query gets executed

pst.executeQuery();

query gets executed

pst.executeQuery();

query gets executed

**Disconnected view of data**

A disconnected environment is one in which a user is not necessarily connected with a database. Connection is required only at the time of retrieval

after that connection is close and if data needs to be updated again connection will be re established.The main advantage of disconnected environment is no need to have a permanent connection with a Data Source which is advantageous especially when there are thousands of records in database which u want to traverse. Another advantage is difference between

ResultSet and CachedRowSetImpl.

ResultSet is not serializable but CachedRowSetImpl is. Hence we can always pass CachedRowSetImpl over the network.

**SERVLET INFO**

Request-Response Model

browser sends a request and Web Server sends the response. i.e. it will return some static page to client. all clients can see the same page i.e. static content.

**Dynamic Content AKA Request-Process-Response Model**

when client sends the request, it will be processed by a software component (server side technology) inside Web Server and accordingly client will get the response. Here response is based on type of request. Hence every client will see different type of response.

servlet/jsp

since these technologies can be written only in java, they get all the advantages of java language.

servlet/jsp vs CGI

in CGI for each client request, a separate process is created. in servlet/jsp for each client request, a lightweight thread is created.

**Get vs Post**

1) in case of "get" we can see parameters passed on the address bar.

in "post" we don't see them.

2) in case of "get" limited amount of data can be passed to the server.

in "post" we don't have such limit.

3) Get request should be used to get information from the server.

Post request should be used to add or update information on the server

**What is Connection Pool?**

Connection pool is a cache of database connections maintained so that the connections can be reused when future requests to the database are required. Connection pools are used to enhance the performance of executing commands on a database.

**HttpServlet\_init\_Container Concept**

1) There are two init() methods with HttpServlet

public void init()

public void init(ServletConfig config)

2) Web Container before invoking init method , creates ServletConfig and stores any init parameter set inside it.

3) Web Container invokes "init(ServletConfig config)" method of HttpServlet by passing ServletConfig implementation.

4) HttpServlet's "init(ServletConfig config)" method invokes "init()" method of its own.

**What is JNDI?**

The Java Naming and Directory InterfaceTM (JNDI) is an application programming interface (API) that provides common way of accessing variety of directories(new, emerging, and already deployed). i.e. client can access any directory service/s using JNDI without worrying about their internal implementation.

Context and InitialContext

Here Context is an interface and InitialContext is an implementation. These two are the part of JNDI API.

Let's take a look at Context's methods:

· **void bind(String stringName, Object object):** Binds a name to an object. The name must not be bound to another object. All intermediate contexts must already exist.

· **void rebind(String stringName, Object object):** Binds a name to an object. All intermediate contexts must already exist.

· **Object lookup(String stringName):** Returns the specified object.

·  **void unbind(String stringName):** Unbinds the specified object.

The Context interface also provides methods for renaming and listing bindings.

· void rename(String stringOldName, String stringNewName):Changes the name to which an object is bound.

**Servlet Config and Context Info**

getServletConfig().getInitParameter("file");

getServletConfig() returns "ServletConfig".

ServletConfig is one per servlet. it is used to read "init" or "config" parameter value/s.

String database=getServletContext().getInitParameter("database");

pw.println(database);

getServletContext() returns "ServletContext".

ServletContext is one per web app. it is used to read "context" parameter value/s.

**Parameters**

parameters are used to send some kind of information to the servlet.

types of parameters passed to servlet

1) request

2) init or config

3) context

request:- these parameters are limited to only a particular request

init or config :- these parameters are shared by all the requests to a particular servlet

context :- these parameters are shared by all the servlets in that particular context (e.g. myapp)

request parameters are automatically set in the request.

init and context parameters need to be explicitly set in the DD.

request parameter can be read by the method "getParameter()".

init or config and context parameters can be read by the method "getInitParameter()".

in order to read request parameter, you need to use "HttpServletRequest".

in order to read init or config parameter, you need to use "ServletConfig" which is one per servlet.

in order to read context parameter, you need to use "ServletContext" which is one per context.

**Attributes vs Parameters**

parameters vs attributes

a) parameters are strings , whereas attributes can be java objects.

b) parameters can be set in DD only (except request parameter which is set implicitly). attributes are set programmatically only.

c) parameters are of

request, config and context

attributes in servlet

there are 3 types of attributes in servlet

a) request

b) session

c) context

request attributes can be accessible only till the request is available.

session attributes are accessible till the session is alive.

context attributes can be accessible to all the servlets/jsps of the context (e.g. myapp).

parameters are used to send some kind of information to the servlet.

types of parameters passed to servlet

1) request

2) init or config

3) context

request:- these parameters are limited to only a particular request

init or config :- these parameters are shared by all the requests to a particular servlet

context :- these parameters are shared by all the servlets in that particular context (e.g. myapp)

request parameters are automatically set in the request.

init and context parameters need to be explicitly set in the DD.

request parameter can be read by the method "getParameter()".

init or config and context parameters can be read by the method "getInitParameter()".

in order to read request parameter, you need to use "HttpServletRequest".

in order to read init or config parameter, you need to use "ServletConfig" which is one per servlet.

in order to read context parameter, you need to use "ServletContext" which is one per context.

**Get vs Post?  
By default html method is "get"**

1) in case of "get" we can see parameters passed on the address bar.

in "post" we don't see them

2) in case of "get" limited amount of data can be passed to the server.

in "post" we don't have such limit.

3) Get request should be used to get information from the server.

Post request should be used to add or update information on the server.

**What if programmer does not know any of these? will he be able to display records?**

Yes, with the help of "ResultSetMetaData" interface which gives complete information about ResultSet such as no.of columns , their names , types etc.

**CACHED ROWSET IMPL**

A disconnected environment is one in which a user is not necessarily connected with a database. Connection is required only at the time of retrieval after that connection is close and if data needs to be updated again connection will be re established.The main advantage of disconnected environment is no need to have a permanent connection with a Data Source which is

advantageous especially when there are thousands of records in database which u want to traverse. Another adavantage is difference between ResultSet and CachedRowSetImpl.

ResultSet is not serializable but CachedRowSetImpl is. Hence we can always pass CachedRowSetImpl over the network.

**Q12 What is difference between load and get(repeat Q22)? Also what is difference in persist merge save or saveOrupdate?**

The main runtime interface between a Java application and Hibernate. This is the central API class abstracting the notion of a persistence service.

The lifecycle of a Session is bounded by the beginning and end of a logical transaction. (Long transactions might span several database transactions.)

The main function of the Session is to offer create, read and delete operations for instances of mapped entity classes. Instances may exist in one of three states:

transient: never persistent, not associated with any Session

persistent: associated with a unique Session

detached: previously persistent, not associated with any Session

Transient instances may be made persistent by calling save(), persist() or saveOrUpdate(). Persistent instances may be made transient by calling delete(). Any instance returned by a get() or load() method is persistent. Detached instances may be made persistent by calling update(), saveOrUpdate(), lock() or replicate(). The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().

save() and persist() result in an SQL INSERT, delete() in an SQL DELETE and update() or merge() in an SQL UPDATE. Changes to persistent instances are detected at flush time and also result in an SQL UPDATE. saveOrUpdate() and replicate() result in either an INSERT or an UPDATE.

It is not intended that implementors be threadsafe. Instead each thread/transaction should obtain its own instance from a SessionFactory.

**Q14(repeat of Q23)**

**Q20(repeat of Q3)**

**Q24 hibernate entity lifecycle**

Transient object

Transient objects are those that are instantiated by the application but not yet made persistent by calling the save() method of the session object.

Persistent object

Persistent objects are those that are instantiated by the application and then made persistent by calling the save() method of the session object. Persistent objects are always associated with a Session and are transactional. Persistent objects participate in transactions and their state is synchronized with the database at the end of the transaction. This means when a transaction is committed, the state held in memory is propagated to the database by the execution of the appropriate SQL Insert, Update and Delete statements.

Detached object

After a transaction completes, all the associated persistent objects still exists in memory but they lose their association with the session on encountering a session.close(). Such objects are called Detached objects.