WEBSERVICES

1. What is Web Service?

In short, Web Service is a software system for communicating two devices over the network.

2. How does web services work?

A web service is used to communicate among various applications by using open standards such as HTML, XML, WSDL, and SOAP. You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

3. What are the advantages of web services?

Interoperability: By the help of web services, an application can communicate with other application developed in any language.

Reusability: We can expose the web service so that other applications can use it.

Modularity: By the help of web service, we can create a service for a specific task such as tax calculation etc.

4. SOAP

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.

5. Advantages of SOAP

WS Security: SOAP defines its own security known as WS Security.

Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.

6. Disadvantages of SOAP

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So, it is slow and consumes more bandwidth and resource.
WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

7. WSDL

WSDL is a xml document containing information about web services such as method name, method parameter, something like that.

8. WSDL usage

WSDL is used in web service to describe the availability of service.

9. REST

REST is an architectural style, not a protocol like SOAP.

10. Advantages of REST

Fast: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

Language and Platform independent: RESTful web services can be written in any programming language and executed in any platform.

Can use SOAP: RESTful web services can use SOAP web services as the implementation.

Permits different data format: REST permits different data format such as Plain Text, HTML, XML and JSON.

11. Difference between SOAP and REST

1. SOAP is a **protocol**. REST is an **architectural style**.
2. SOAP **defines standards** to be strictly followed. REST does not define too much standards like SOAP.
3. SOAP **requires more bandwidth** and resource than REST. REST **requires less bandwidth** and resource than SOAP.
4. SOAP **defines its own security**. RESTful web services **inherit security measures** from the underlying transport.
5. SOAP **permits XML** data format only. REST **permits different data format** such as Plain text, HTML, XML, JSON etc.
6. Rest - Stateless, Cacheable. Soap: Stateful

7. SOAP **can't use REST** because it is a protocol. REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
8. SOAP **uses services interfaces to expose the business logic**. REST **uses URI to expose business logic**.
9. In Java **JAX-WS** is the java API for SOAP web services. In Java **JAX-RS** is the java API for RESTful web services.
10. SOAP is **less preferred** than REST. REST **more preferred** than SOAP.

11. What is SOA?

SOA is a design pattern to provide services to other application through protocol.

SPRING FRAMEWORK

1. What is SPRING?

Spring is now widely used framework to develop enterprise application in java.

2. What are the advantages of Spring Framework?

There are many advantages of Spring Framework.

1. **Predefined templates:** Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So, there is no need to write too much code. It hides the basic steps of these technologies.
2. **Loose coupling:** The Spring applications are loosely coupled because of dependency injection. The objects give their dependencies instead of creating or looking for dependent objects.
3. **Lightweight:** Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.
4. **Fast development:** The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.
5. **Aspect Oriented Programming (AOP):** Spring supports Aspect oriented programming and enables cohesive development by separating application business logic from system services.
6. **Exception handling:** Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.

3. Spring modules

The Spring comprises of many modules such as **core**, **beans**, **context**, **expression language** (**CORE CONTAINER**), **AOP**, **Aspects**, **Instrumentation**, **JDBC, ORM, OXM, JMS, Transaction (DATA ACCESS / INTEGRATION), Web, Servlet (WEB),** etc. *These modules are grouped into Core Container, AOP, Aspects, Instrumentation, Data Access / Integration, Web (MVC / Remoting)*

4. Difference between IOC and DI

Inversion of Control (IoC) is a general concept, and it can be expressed in many different ways and Dependency Injection is merely one concrete example of Inversion of Control.

5. Inversion of Control (IOC) and Dependency Injection (DI)

DI concept says that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. **The IOC container** is then responsible for hooking it all up.

6. Types of IOC container

There are 2 types of IoC container. **BeanFactory** and **ApplicationContext.**

7. Difference between BeanFactory and ApplicationContext

The **BeanFactory** and the **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. So, it is better to use **ApplicationContext** than **BeanFactory**.

8. Example of BeanFactory implementation

The most commonly used BeanFactory implementation is the **XmlBeanFactory** class. This container reads the configuration metadata from an XML file and uses it to create a fully configured system or application.

9. Common implementations of ApplicationContext

There are three commonly used implementations of Application Context.

FileSystemXmlApplicationContext: container loads the definitions of the beans from an XML file. Here you need to provide the full path of the XML bean configuration file to the constructor.

ClassPathXmlApplicationContext: container loads the definitions of the beans from an XML file. Here you do not need to provide the full path of the XML file, but you need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH.

WebXmlApplicationContext: container loads the XML file with definitions of all beans from within a web application.

10. Types of IOC (DI)

Constructor-based dependency injection: is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.

Setter-based dependency injection: is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

11. Difference between CONSTRUCTOR and SETTER injection

Constructor injection is no-partial, **Setter** is partial.

Constructor doesn’t override the setter property. **Setter** overrides the constructor property if both are defined.

Constructor creates new instance if any modification occurs. **Setter** doesn’t create new instance if you change the property value. I think **Constructor** is better for many properties, and **Setter** is better for few properties.

12. Autowiring, autowiring modes

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic. **There are 4 modes.**

no: is the default mode, it means autowiring is not wiring.

byName: injects the bean based on the property name. It uses setter method.

byType: injects the bean based on the property type. It uses setter method.

constructor: injects the bean using constructor.

13. Spring Beans

The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container, for example, in the form of XML <bean/> definitions.

14. **How do you provide configuration metadata to the Spring container?** There are three important methods to provide configuration metadata to the Spring container. **XML-based configuration file**, **Annotation-based configuration**, **Java-based configuration**.

15. Bean scopes in Spring?

There are 5 bean scopes in Spring framework.
singleton: The bean instance will be only once, and same instance will be returned by the IoC container. **(Default)**
prototype: The bean instance will be created each time when requested.
request: The bean instance will be created per HTTP request.
session: The bean instance will be created per HTTP session.
global-session: The bean instance will be created per HTTP global session. It can be used in portlet context only.

16. In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

17. Transaction managements that Spring supports

Programmatic Transaction Management: should be used for few transaction operations.
Declarative Transaction Management: should be used for many transaction operations.

18. Which transaction management do you prefer?

Declarative transaction management is preferable even though it is less flexible than programmatic transaction management. Because, programmatic transaction management is very difficult to maintain even though it is extreme flexible.

19. Advantages of JDBC template in Spring

Less code: By using the JdbcTemplate class, you don't need to create connection, statement, start transaction, commit transaction and close connection to execute different queries. You can execute the query directly.

20. Classes in Spring JDBC API

JdbcTemplate, SimpleJdbcTemplate, NamedParameterJdbcTemplate, SimpleJdbcInsert, SimpleJdbcCall

21. **How to fetch records by Spring JdbcTemplate?** You can fetch records from the database by the **query method of JdbcTemplate**. There are two interfaces to do this **ResultSetExtractor** and **RowMapper.**

22. What is AOP?

AOP is all about managing the common functionality within the application such that it is not embedded within the business logic.
Examples to such cross-cutting concerns are logging, managing security, transaction management.

23. Benefits of AOP

AOP enables you to dynamically add or remove concern before or after the business logic. It is **pluggable** and **easy to maintain**.

24. AOP concepts and terminologies

JOIN POINT: represents a point in your application where you can plug-in AOP aspect. *It is the actual place in the application where an action will be taken using Spring AOP framework.*

ADVICE: represents action taken by aspect.
POINTCUT: It is an expression language of AOP that matches join points.
INTRODUCTION: represents introduction of new fields and methods for a type.
TARGET OBJECT: is a proxy object that is advised by one or more aspects.
ASPECT: is a class in spring AOP that contains advices and joinpoints.
INTERCEPTOR: s a class like aspect that contains one advice only.
AOP Proxy:
WEAVING: is a process of linking aspect with other application.

25. Does spring framework support all JoinPoints?

No, spring framework supports **method execution joinpoint only**.

26. What are the AOP implementations?

There are 3 AOP implementations such as **Spring AOP**, **Apache AspectJ** and **JBoss AOP**.

27. What are the types of advice in AOP?

There are 5 types of advices.
Before Advice: executes before a join point.
After Returning Advice: executes after a joint point completes normally.
After Throwing Advice: executes if method exits by throwing an exception.
After Advice: executes after a join point regardless of join point exit whether normally or exceptional return.
Around Advice: It executes before and after a join point.

28. What is the front controller class of Spring MVC?

The **DispatcherServlet** class works as the front controller in Spring MVC.

29. What is DispatcherServlet?

The **DispatcherServlet** is a class that handles all the HTTP requests and responses.

30. @Controller and @RequestMapping annotations

The **@Controller** annotation marks the class as controller class. It is applied on the class.
The **@RequestMapping** annotation maps the request with the method. It is applied on the method.

31. MODEL, VIEW, CONTROLLER?

The Model encapsulates the application data and in general they will consist of POJO.
The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

32. What does ViewResolver class?

The View Resolver class resolves the view component to be invoked for the request. It defines prefix and suffix properties to resolve the view component.

33. Which ViewResolver class is widely used?

The **InternalResourceViewResolver** class is widely used.

34. Does spring MVC provide validation support?

Yes. It does.

35. Ways to access Hibernate by using SPRING?

I know 2 ways to access Hibernate in Spring.

1st one. Inversion of Control with a Hibernate Template and Callback.

2nd one. Extending **HibernateDAOSupport** and Applying an AOP Interceptor node.

36. What are the ORM's Spring supports?

Spring supports many ORM's. But I currently know **hibernate** and **JPA**

HIBERNATE

1. What is ORM? - Object Mapping Tool

ORM is a programming technique for converting data between relational databases and object-oriented programming languages such as Java, C# etc.

2. Hibernate?

Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and It provides a framework for mapping an object-oriented domain model to a relational database.

3. What is JDBC?

JDBC provides a set of Java API for accessing the relational databases from Java program. These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database.

4. What are the core interfaces of Hibernate?

What are the key components/objects of hibernate?

Configuration: Represents a configuration or properties file required by the Hibernate.

SessionFactory: is a factory of Session. It provides the instance of Session. It holds the data of second level cache that is not enabled by default.

Session: Used to get a physical connection with a database, and it provides methods to store, update, delete, or fetch data from database.

Query: uses SQL or HQL string to retrieve data from the database and create objects.

Criteria: is used to create and execute object-oriented criteria queries to retrieve objects.

Transaction: represents a unit of work with the database and most of the RDBMS supports transaction functionality.

5. Key components of Configuration object

Database Connection: is handled through one or more configuration files supported by Hibernate. These files are hibernate.properties and hibernate.cfg.xml as I remember.

Class Mapping Setup: creates the connection between the Java classes and database tables.

6. What are the advantages of Hibernate over JDBC?

Hibernate has many benefits. For example:

Database independent, same code works for all databases like Oracle, MySQL

Caching: Hibernate supports two level of cache, First level and 2nd level

Lazy loading: Hibernate loads the child objects on demand with lazy loading. It improves the performance.

Fast development: because developer doesn't need to write queries.

7. Difference between First and Second level cache

First Level Cache is associated with **Session**. Second Level Cache is associated with **SessionFactory**.

First Level Cache is **enabled by default**, and Second Level Cache **is not enabled by default**.

8. Difference between get() and load()

If object with id is not found, **get() method returns null**, and **load() throws an exception**.
Get() method always **hit the database**, load() method **doesn't.**
Get() returns **real object**, not proxy. Load() returns **proxy object**.
I use get() method **when I am not sure about the existence of instance** and use load() method **when I am sure that instance exists**.

9. Difference between save() and persist()?

Main difference between save() and persist() method is that, save returns a Serializable object. And persist doesn't return anything because of its return type void.

10. Difference between save() and saveOrUpdate()?

The key difference between them is that save can only INSERT records, and saveOrUpdate() can either INSERT or UPDATE records.

11. Difference between update() and merge()?

update() method should be used when the Hibernate session does not contain an already persistent instance with the same id, and **merge()** should be used if you want to merge your modifications at any time without considering the state of the session.

12. Difference between Session and SessionFactory?

Session is a single-threaded, short-lived object. **SessionFactory** is Immutable and shared by all **Session**. It also lives until the Hibernate is running.
Session provides first level cache, **SessionFactory** provides the Second level cache. In-short SessionFactory is a factory of Session. **Session** is used to get a physical connection with a database.

13. Is Session thread-safe?

No, Session is not a thread-safe object. I sometimes share it between threads.

14. SessionFactory is thread-safe in Hibernate?

Yes. SessionFactory is a concept of a single datastore and it is thread-safe so that many threads can access it concurrently and request for sessions and immutable cache of compiled mappings for a single database. A SessionFactory is usually only built once at startup.

15. States of object/"persistent entity" in Hibernate?

Transient − A new instance of a persistent class which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.

Persistent − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.

Detached − Once we close the Hibernate Session, the persistent instance will become a detached instance.

16. What is the Hibernate Query Language (HQL)?

Hibernate query language, HQL is an object-oriented extension to SQL. It allows you to query, store, update, and retrieve objects from a database without using SQL.

17. Difference between sorted and ordered collection?

Sorted collection: sorts the data in JVM's heap memory using Java's collection framework sorting methods.

Ordered collection: is sorted using order by clause in the database itself.

18. Types of association mapping

One to One: association is similar to many-to-one association with a difference that the column will be set as unique.

One to Many: An Object can be associated with multiple objects

Many to One: An Object can be associated with multiple objects

Many to Many: mapping can be implemented using a Set java collection that does not contain any duplicate element.

19. Fetching types/strategies

There are 2 fetching types in Hibernate. **Lazy** and **Eager**.

Lazy loading: Hibernate loads the child objects on demand with lazy loading. It improves the performance

Eager loading: Hibernate loads the child object immediately with eager.