Spring

1. [What is Spring Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-overview)

The **Spring Framework** is an application **framework**and inversion of control container for the **Java**platform. The **framework's** core features can be used by any **Java** application, but there are extensions for building web applications on top of the **Java** EE (Enterprise Edition) platform. ... The **Spring Framework** is open source.

1. [What are some of the important features and advantages of Spring Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-advantages)

**Spring** enables the developers to develop the enterprise applications using POJOs (Plain Old **Java**Object). ... **Spring** provides a consistent transaction management interface that can scale down to a local transaction and scale up to global transactions (using JTA).

1. [What do you understand by Dependency Injection?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#dependency-injection)

**dependency injection** is a technique whereby one object supplies the dependencies of another object. A dependency is an object that can be used (a [service](https://en.wikipedia.org/wiki/Service_(systems_architecture))). An injection is the passing of a dependency to a dependent object (a [client](https://en.wikipedia.org/wiki/Client_(computing))) that would use it. The service is made part of the client's [state](https://en.wikipedia.org/wiki/State_(computer_science)).[[1]](https://en.wikipedia.org/wiki/Dependency_injection#cite_note-JamesShore-1) Passing the service to the client, rather than allowing a client to build or [find the service](https://en.wikipedia.org/wiki/Service_locator_pattern), is the fundamental requirement of the pattern.

1. [How do we implement DI in Spring Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-dependency-injection)

Dependency Injection (DI) is a software design pattern that implements inversion of control for resolving dependencies.

An injection is the passing of a dependency to a dependent object that would use it.

DI is a process whereby objects define their dependencies. The other objects they work with—only through constructor arguments or arguments to a factory method or property—are set on the object instance after it is constructed or returned from a factory method.

The container then injects those dependencies, and it creates the bean. This process is named Inversion of Control (IoC) (the bean itself controls the instantiation or location of its dependencies by using direct construction classes or a Service Locator).

DI refers to the process of supplying an external dependency to a software component.

1. [What are the benefits of using Spring Tool Suite?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-tool-suite)

The reason is if we want to create a spring application, we will have to write Xml, adding jars using maven. But if we use SPS, within a minute we can start building our application

1. [Name some of the important Spring Modules?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-modules)

Spring Core Module

Spring Context [ J2EE ]

Spring DAO Module [ Spring JDBC ]

Spring ORM module

Spring AOP [ Aspect Oriented Programming ]

Spring WEB-MVC Module

1. [What do you understand by Aspect Oriented Programming?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#aspect-oriented-programming)

**aspect-oriented programming** (**AOP**) is a [programming paradigm](https://en.wikipedia.org/wiki/Programming_paradigm) that aims to increase [modularity](https://en.wikipedia.org/wiki/Modularity_(programming)) by allowing the [separation of](https://en.wikipedia.org/wiki/Separation_of_concerns) [cross-cutting concerns](https://en.wikipedia.org/wiki/Cross-cutting_concern).

1. [What is Aspect,](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "aspect-advice-pointcut-joinpoint) [Advice, Pointcut, JointPoint and Advice Arguments in AOP?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "aspect-advice-pointcut-joinpoint)
2. **Aspect**: Aspect is a class that implements cross-cutting concerns, such as transaction management. Aspects can be a normal class configured and then configured in Spring Bean configuration file or we can use Spring AspectJ support to declare a class as Aspect using @Aspect annotation.
3. **Advice**: Advice is the action taken for a particular join point. In terms of programming, they are methods that gets executed when a specific join point with matching pointcut is reached in the application. You can think of Advices as [Spring interceptors](https://www.journaldev.com/2676/spring-mvc-interceptor-example-handlerinterceptor-handlerinterceptoradapter) or [Servlet Filters](https://www.journaldev.com/1933/java-servlet-filter-example-tutorial).
4. **Pointcut**: Pointcut are regular expressions that is matched with join points to determine whether advice needs to be executed or not. Pointcut uses different kinds of expressions that are matched with the join points. Spring framework uses the AspectJ pointcut expression language for determining the join points where advice methods will be applied.
5. **Join Point**: A join point is the specific point in the application such as method execution, exception handling, changing object variable values etc. In Spring AOP a join points is always the execution of a method.
6. **Advice Arguments**: We can pass arguments in the advice methods. We can use args() expression in the pointcut to be applied to any method that matches the argument pattern. If we use this, then we need to use the same name in the advice method from where argument type is determined.

1. [What is the difference between Spring AOP and AspectJ AOP?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-vs-aspectj)

Spring-AOP Pros

* It is simpler to use than AspectJ, since you don't have to use LTW ([load-time weaving](http://www.eclipse.org/aspectj/doc/next/devguide/ltw.html)) or the AspectJ compiler.
* It uses the Proxy pattern and the Decorator pattern

Spring-AOP Cons

* This is proxy-based AOP, so basically you can only use method-execution joinpoints.
* Aspects aren't applied when calling another method within the same class.
* There can be a little runtime overhead.
* Spring-AOP cannot add an aspect to anything that is not created by the Spring factory

AspectJ Pros

* This supports all joinpoints. This means you can do anything.
* There is less runtime overhead than that of Spring AOP.

AspectJ Cons

* Be careful. Check if your aspects are weaved to only what you wanted to be weaved.
* You need extra build process with AspectJ Compiler or have to setup LTW (load-time weaving)

1. [What is Spring IoC Container?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-ioc-container)

A bean is an object that is instantiated, assembled, and otherwise managed by a **Spring IoC container**. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a **container**

1. [What is a Spring Bean?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-bean)

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.

1. [What is the importance of Spring bean configuration file?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-bean-configuration-file)

We use Spring Bean configuration file to define all the beans that will be initialized by Spring Context. When we create the instance of Spring ApplicationContext, it reads the spring bean xml file and initialize all of them. Once the context is initialized, we can use it to get different bean instances.

1. [What are different ways to configure a class as Spring Bean?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-bean-configuration)

**XML - based configuration**

**Java based**

**annotation based**

https://springframework.guru/spring-framework-annotations/

1. [What are different scopes of Spring Bean?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-bean-scopes)
2. **1. singleton(default\*)**
3. Scopes a single bean definition to a single object instance per Spring IoC container.
4. **2. prototype**
5. Scopes a single bean definition to any number of object instances.
6. **3. request**
7. Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.
8. **4. session**
9. Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.
10. **5. global session**
11. Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

1. [What is Spring Bean life cycle?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "spring-bean-life-cycle)

Spring Beans are initialized by Spring Container and all the dependencies are also injected. When context is destroyed, it also destroys all the initialized beans. This works well in most of the cases but sometimes we want to initialize other resources or do some validation before making our beans ready to use. Spring framework provides support for post-initialization and pre-destroy methods in spring beans.

We can do this by two ways – by implementing InitializingBean and DisposableBean

1. [How to get ServletContext and ServletConfig object in a Spring Bean?](https://www.journaldev.com/2696/spring-interview-questions-and-answers" \l "servlet-context-config-spring-bean)

There are two ways to get Container specific objects in the spring bean.

1. Implementing Spring \*Aware interfaces, for these ServletContextAware and ServletConfigAware interfaces, for complete example of these aware interfaces, please read [Spring Aware Interfaces](https://www.journaldev.com/2637/spring-bean-life-cycle)
2. Using @Autowired annotation with bean variable of type ServletContext and ServletConfig. They will work only in servlet container specific environment only though.

@Autowired

ServletContext servletContext;

1. [What is Bean wiring and @Autowired annotation?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#bean-wiring-autowiring)

The process of injection spring bean dependencies while initializing it called Spring Bean Wiring.Usually it’s best practice to do the explicit wiring of all the bean dependencies, but spring framework also supports autowiring. We can use @Autowired annotation with fields or methods for **autowiring byType**. For this annotation to work, we also need to enable annotation based configuration in spring bean configuration file. This can be done by **context:annotation-config** element

1. [What are different types of Spring Bean autowiring?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#bean-autowire-types)

There are four types of autowiring in Spring framework.

1. **autowire byName**
2. **autowire byType**
3. **autowire by constructor**
4. autowiring by **@Autowired** and **@Qualifier** annotations
5. [Does Spring Bean provide thread safety?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-bean-thread-safety)

The default scope of Spring bean is singleton, so there will be only one instance per context. That means that all the having a class level variable that any thread can update will lead to inconsistent data. Hence in default mode spring beans are not thread-safe.However we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It’s a design decision and based on the project requirements.

1. [What is a Controller in Spring MVC?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-controller-bean)

ust like MVC design pattern, Controller is the class that takes care of all the client requests and send them to the configured resources to handle it. In Spring MVC, org.springframework.web.servlet.DispatcherServlet is the front controller class that initializes the context based on the spring beans configurations.

A Controller class is responsible to handle different kind of client requests based on the request mappings. We can create a controller class by using @Controller annotation. Usually it’s used with @RequestMapping annotation to define handler methods for specific URI mapping.

1. [What’s the difference between @Component, @Repository & @Service annotations in Spring?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#component-vs-controller-vs-service-vs-repository)

**@Component** is used to indicate that a class is a component. These classes are used for auto detection and configured as bean, when annotation based configurations are used.

**@Controller** is a specific type of component, used in MVC applications and mostly used with RequestMapping annotation.

**@Repository** annotation is used to indicate that a component is used as repository and a mechanism to store/retrieve/search data. We can apply this annotation with DAO pattern implementation classes.

**@Service** is used to indicate that a class is a Service. Usually the business facade classes that provide some services are annotated with this.

1. [What is DispatcherServlet and ContextLoaderListener?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#DispatcherServlet-ContextLoaderListener)

DispatcherServlet is the front controller in the Spring MVC application and it loads the spring bean configuration file and initialize all the beans that are configured. If annotations are enabled, it also scans the packages and configure any bean annotated with @Component, @Controller, @Repository or @Service annotations.

ContextLoaderListener is the listener to start up and shut down Spring’s root WebApplicationContext. It’s important functions are to tie up the lifecycle of ApplicationContext to the lifecycle of the ServletContext and to automate the creation of ApplicationContext. We can use it to define shared beans that can be used across different spring contexts.

1. [What is ViewResolver in Spring?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-ViewResolver)

ViewResolver implementations are used to resolve the view pages by name. Usually we configure it in the spring bean configuration file

1. [What is a MultipartResolver and when its used?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#MultipartResolver)

MultipartResolver interface is used for uploading files – CommonsMultipartResolver and StandardServletMultipartResolver are two implementations provided by spring framework for file uploading. By default there are no multipart resolvers configured but to use them for uploading files, all we need to define a bean named “multipartResolver” with type as MultipartResolver in spring bean configurations.

Once configured, any multipart request will be resolved by the configured MultipartResolver and pass on a wrapped HttpServletRequest. Then it’s used in the controller class to get the file and process it.

1. [How to handle exceptions in Spring MVC Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-exceptions)

Spring MVC Framework provides following ways to help us achieving robust exception handling.

1. **Controller Based** – We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.
2. **Global Exception Handler** – Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.
3. **HandlerExceptionResolver implementation** – For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.
4. [How to create ApplicationContext in a Java Program?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#java-ApplicationContext)

There are following ways to create spring context in a standalone java program.

1. **AnnotationConfigApplicationContext**: If we are using Spring in standalone java applications and using annotations for Configuration, then we can use this to initialize the container and get the bean objects.
2. **ClassPathXmlApplicationContext**: If we have spring bean configuration xml file in standalone application, then we can use this class to load the file and get the container object.
3. **FileSystemXmlApplicationContext**: This is similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from anywhere in the file system.
4. [Can we have multiple Spring configuration files?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#multiple-context-files)

For Spring MVC applications, we can define multiple spring context configuration files through contextConfigLocation. This location string can consist of multiple locations separated by any number of commas and spaces.

1. [What is ContextLoaderListener?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#ContextLoaderListener)

ContextLoaderListener is the listener class used to load root context and define spring bean configurations that will be visible to all other contexts.

1. [What are the minimum configurations needed to create Spring MVC application?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-hello-world)
2. For creating a simple Spring MVC application, we would need to do following tasks.
   * Add spring-context and spring-webmvc dependencies in the project.
   * Configure DispatcherServlet in the web.xml file to handle requests through spring container.
   * Spring bean configuration file to define beans, if using annotations then it has to be configured here. Also we need to configure view resolver for view pages.
   * Controller class with request mappings defined to handle the client requests.

Above steps should be enough to create a simple Spring MVC Hello World application.

1. [How would you relate Spring MVC Framework to MVC architecture?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-architecture)

As the name suggests Spring MVC is built on top of **Model-View-Controller** architecture. DispatcherServlet is the Front Controller in the Spring MVC application that takes care of all the incoming requests and delegate it to different controller handler methods.

Model can be any Java Bean in the Spring Framework, just like any other MVC framework Spring provides automatic binding of form data to java beans. We can set model beans as attributes to be used in the view pages.

View Pages can be JSP, static HTMLs etc. and view resolvers are responsible for finding the correct view page. Once the view page is identified, control is given back to the DispatcherServlet controller. DispatcherServlet is responsible for rendering the view and returning the final response to the client.

1. [How to achieve localization in Spring MVC applications?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-localization-i18n)

Spring provides excellent support for localization or i18n through resource bundles. Basis steps needed to make our application localized are:

1. Creating message resource bundles for different locales, such as messages\_en.properties, messages\_fr.properties etc.
2. Defining messageSource bean in the spring bean configuration file of type ResourceBundleMessageSource or ReloadableResourceBundleMessageSource.
3. For change of locale support, define localeResolver bean of type CookieLocaleResolver and configure LocaleChangeInterceptor interceptor
4. [How can we use Spring to create Restful Web Service returning JSON response?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-restful-json)

We can use Spring Framework to create Restful web services that returns JSON data. Spring provides integration with [Jackson JSON](https://www.journaldev.com/2324/jackson-json-java-parser-api-example-tutorial) API that we can use to send JSON response in restful web service.

1. [What are some of the important Spring annotations you have used?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-annotations)

Some of the [Spring annotations](https://www.journaldev.com/16966/spring-annotations) that I have used in my project are:

* **@Controller** – for controller classes in Spring MVC project.
* **@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through [Spring MVC RequestMapping Annotation Examples](https://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example)
* **@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.
* **@PathVariable** – for mapping dynamic values from the URI to handler method arguments.
* **@Autowired** – for autowiring dependencies in spring beans.
* **@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.
* **@Service** – for service classes.
* **@Scope** – for configuring scope of the spring bean.
* **@Configuration**, **@ComponentScan** and **@Bean** – for java based configurations.
* AspectJ annotations for configuring aspects and advices, **@Aspect**, **@Before**, **@After**, **@Around**, **@Pointcut** etc.

1. [Can we send an Object as the response of Controller handler method?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-object-response).

* **@Controller** – for controller classes in Spring MVC project.
* **@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through [Spring MVC RequestMapping Annotation Examples](https://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example)
* **@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.
* **@PathVariable** – for mapping dynamic values from the URI to handler method arguments.
* **@Autowired** – for autowiring dependencies in spring beans.
* **@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.
* **@Service** – for service classes.
* **@Scope** – for configuring scope of the spring bean.
* **@Configuration**, **@ComponentScan** and **@Bean** – for java based configurations.
* AspectJ annotations for configuring aspects and advices, **@Aspect**, **@Before**, **@After**, **@Around**, **@Pointcut** etc.

1. [How to upload file in Spring MVC Application?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-file-upload)

Spring provides built-in support for uploading files through **MultipartResolver** interface implementations. It’s very easy to use and requires only configuration changes to get it working. Obviously we would need to write controller handler method to handle the incoming file and process it. For a complete example

1. [How to validate form data in Spring Web MVC Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-form-validation)

Spring supports JSR-303 annotation based validations as well as provide Validator interface that we can implement to create our own custom validator. For using JSR-303 based validation, we need to annotate bean variables with the required validations.

1. [What is Spring MVC Interceptor and how to use it?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-mvc-interceptors)

Spring MVC Interceptors are like Servlet Filters and allow us to intercept client request and process it. We can intercept client request at three places – **preHandle**, **postHandle** and **afterCompletion**.

We can create spring interceptor by implementing HandlerInterceptor interface or by extending abstract class **HandlerInterceptorAdapter**.

We need to configure interceptors in the spring bean configuration file. We can define an interceptor to intercept all the client requests or we can configure it for specific URI mapping too.

1. [What is Spring JdbcTemplate class and how to use it?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-jdbc-JdbcTemplate)

Spring Framework provides excellent integration with JDBC API and provides JdbcTemplate utility class that we can use to avoid bolier-plate code from our database operations logic such as Opening/Closing Connection, ResultSet, PreparedStatement etc.

1. [How to use Tomcat JNDI DataSource in Spring Web Application?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-tomcat-jndi-DataSource)

for using servlet container configured JNDI DataSource, we need to configure it in the spring bean configuration file and then inject it to spring beans as dependencies. Then we can use it with JdbcTemplate to perform database operations

1. [How would you achieve Transaction Management in Spring?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-transaction-management)

Spring framework provides transaction management support through Declarative Transaction Management as well as programmatic transaction management. Declarative transaction management is most widely used because it’s easy to use and works in most of the cases.

We use annotate a method with @Transactional annotation for Declarative transaction management. We need to configure transaction manager for the DataSource in the spring bean configuration file.

1. [What is Spring DAO?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-DAO)

Spring DAO support is provided to work with data access technologies like JDBC, Hibernate in a consistent and easy way. For example we have JdbcDaoSupport, HibernateDaoSupport, JdoDaoSupport and JpaDaoSupport for respective technologies.

Spring DAO also provides consistency in exception hierarchy and we don’t need to catch specific exceptions.

1. [How to integrate Spring and Hibernate Frameworks?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-hibernate-integration)

We can use Spring ORM module to integrate Spring and Hibernate frameworks, if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.

1. [What is Spring Security?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-security).

Spring security framework focuses on providing both authentication and authorization in java applications. It also takes care of most of the common security vulnerabilities such as CSRF attack.

It’s very beneficial and easy to use Spring security in web applications, through the use of annotations such as @EnableWebSecurity. You should go through following posts to learn how to use Spring Security framework.

1. [How to inject a java.util.Properties into a Spring Bean?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-properties-inject)

We need to define propertyConfigurer bean that will load the properties from the given property file. Then we can use Spring EL support to inject properties into other bean dependencies.

1. [Name some of the design patterns used in Spring Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-design-patterns)
2. Singleton Pattern: Creating beans with default scope.
3. [Factory Pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java): Bean Factory classes
4. [Prototype Pattern](https://www.journaldev.com/1440/prototype-design-pattern-in-java): Bean scopes
5. [Adapter Pattern](https://www.journaldev.com/1487/adapter-design-pattern-java): Spring Web and Spring MVC
6. [Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern): Spring Aspect Oriented Programming support
7. [Template Method Pattern](https://www.journaldev.com/1763/template-method-design-pattern-in-java): JdbcTemplate, HibernateTemplate etc
8. Front Controller: Spring MVC DispatcherServlet
9. Data Access Object: Spring DAO support
10. Dependency Injection and Aspect Oriented Programming
11. [What are some of the best practices for Spring Framework?](https://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-best-practices)
12. Avoid version numbers in schema reference, to make sure we have the latest configs.
13. Divide spring bean configurations based on their concerns such as spring-jdbc.xml, spring-security.xml.
14. For spring beans that are used in multiple contexts in Spring MVC, create them in the root context and initialize with listener.
15. Configure bean dependencies as much as possible, try to avoid autowiring as much as possible.
16. For application level properties, best approach is to create a property file and read it in the spring bean configuration file.
17. For smaller applications, annotations are useful but for larger applications annotations can become a pain. If we have all the configuration in xml files, maintaining it will be easier.
18. Use correct annotations for components for understanding the purpose easily. For services use @Service and for DAO beans use @Repository.
19. Spring framework has a lot of modules, use what you need. Remove all the extra dependencies that gets usually added when you create projects through Spring Tool Suite templates.
20. If you are using Aspects, make sure to keep the join pint as narrow as possible to avoid advice on unwanted methods. Consider custom annotations that are easier to use and avoid any issues.
21. Use dependency injection when there is actual benefit, just for the sake of loose-coupling don’t use it because it’s harder to maintain.

Hibernate

1. [What is Hibernate Framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-overview)

Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

1. [What is Java Persistence API (JPA)?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#jpa-overview)

Java Persistence API (JPA) provides specification for managing the relational data in applications. Current JPA version 2.1 was started in July 2011 as JSR 338. JPA 2.1 was approved as final on 22 May 2013.

JPA specifications is defined with annotations in javax.persistence package. Using JPA annotation helps us in writing implementation independent code.

1. [What are the important benefits of using Hibernate Framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-benefits)
2. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
3. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
4. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association.
5. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentations and help is easily available in forums.
6. Hibernate is easy to integrate with other Java EE frameworks, it’s so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.
7. Hibernate supports lazy initialization using proxy objects and perform actual database queries only when it’s required.
8. Hibernate cache helps us in getting better performance.
9. For database vendor specific feature, hibernate is suitable because we can also execute native sql queries.
10. [What are the advantages of Hibernate over JDBC?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-vs-jdbc)
11. Hibernate removes a lot of boiler-plate code that comes with JDBC API, the code looks more cleaner and readable.
12. Hibernate supports inheritance, associations and collections. These features are not present with JDBC API.
13. Hibernate implicitly provides transaction management, in fact most of the queries can’t be executed outside transaction. In JDBC API, we need to write code for transaction management using commit and rollback. Read more at [JDBC Transaction Management](https://www.journaldev.com/2483/java-jdbc-transaction-management-savepoint).
14. JDBC API throws SQLException that is a checked exception, so we need to write a lot of try-catch block code. Most of the times it’s redundant in every JDBC call and used for transaction management. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception, so we don’t need to write code to handle it. Hibernate built-in transaction management removes the usage of try-catch blocks.
15. Hibernate Query Language (HQL) is more object oriented and close to java programming language. For JDBC, we need to write native sql queries.
16. Hibernate supports caching that is better for performance, JDBC queries are not cached hence performance is low.
17. Hibernate provide option through which we can create database tables too, for JDBC tables must exist in the database.
18. Hibernate configuration helps us in using JDBC like connection as well as JNDI DataSource for connection pool. This is very important feature in enterprise application and completely missing in JDBC API.
19. Hibernate supports JPA annotations, so code is independent of implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.
20. [Name some important interfaces of Hibernate framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-interfaces)
21. SessionFactory (org.hibernate.SessionFactory): SessionFactory is an immutable thread-safe cache of compiled mappings for a single database. We need to initialize SessionFactory once and then we can cache and reuse it. SessionFactory instance is used to get the Session objects for database operations.
22. Session (org.hibernate.Session): Session is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It wraps JDBC java.sql.Connection and works as a factory for org.hibernate.Transaction. We should open session only when it’s required and close it as soon as we are done using it. Session object is the interface between java application code and hibernate framework and provide methods for CRUD operations.
23. Transaction (org.hibernate.Transaction): Transaction is a single-threaded, short-lived object used by the application to specify atomic units of work. It abstracts the application from the underlying JDBC or JTA transaction. A org.hibernate.Session might span multiple org.hibernate.Transaction in some cases.
24. [What is hibernate configuration file?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-configuration-file)

Hibernate configuration file contains database specific configurations and used to initialize SessionFactory. We provide database credentials or JNDI resource information in the hibernate configuration xml file. Some other important parts of hibernate configuration file is Dialect information, so that hibernate knows the database type and mapping file or class details.

1. [What is hibernate mapping file?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-mapping-file)

Hibernate mapping file is used to define the entity bean fields and database table column mappings. We know that JPA annotations can be used for mapping but sometimes XML mapping file comes handy when we are using third party classes and we can’t use annotations.

1. [Name some important annotations used for Hibernate mapping?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-annotations)
2. javax.persistence.Entity: Used with model classes to specify that they are entity beans.
3. javax.persistence.Table: Used with entity beans to define the corresponding table name in database.
4. javax.persistence.Access: Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.
5. javax.persistence.Id: Used to define the primary key in the entity bean.
6. javax.persistence.EmbeddedId: Used to define composite primary key in the entity bean.
7. javax.persistence.Column: Used to define the column name in database table.
8. javax.persistence.GeneratedValue: Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.
9. javax.persistence.OneToOne: Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany
10. org.hibernate.annotations.Cascade: Used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType
11. javax.persistence.PrimaryKeyJoinColumn: Used to define the property for foreign key. Used with org.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter
12. [What is Hibernate SessionFactory and how to configure it?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-session-factory)

SessionFactory is the factory class used to get the Session objects. SessionFactory is responsible to read the hibernate configuration parameters and connect to the database and provide Session objects. Usually an application has a single SessionFactory instance and threads servicing client requests obtain Session instances from this factory.

The internal state of a SessionFactory is immutable. Once it is created this internal state is set. This internal state includes all of the metadata about Object/Relational Mapping.

SessionFactory also provide methods to get the Class metadata and Statistics instance to get the stats of query executions, second level cache details etc.

1. [Hibernate SessionFactory is thread safe?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#session-factory-thread-safe)

Internal state of SessionFactory is immutable, so it’s thread safe. Multiple threads can access it simultaneously to get Session instances.

1. [What is Hibernate Session and how to get it?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-session)

Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.

Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object

1. [Hibernate Session is thread safe?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#session-thread-safe)

Hibernate Session object is not thread safe, every thread should get it’s own session instance and close it after it’s work is finished.

1. [What is difference between openSession and getCurrentSession?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#openSession-vs-getCurrentSession)

[Hibernate SessionFactory](https://www.journaldev.com/3522/hibernate-sessionfactory) getCurrentSession() method returns the session bound to the context. But for this to work, we need to configure it in hibernate configuration file. Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed.

Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.

1. [What is difference between Hibernate Session get() and load() method?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-get-vs-load)

Hibernate session comes with different methods to load data from database. get and load are most used methods, at first look they seems similar but there are some differences between them.

get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading.

Since load() throws exception when data is not found, we should use it only when we know data exists.

We should use get() when we want to make sure data exists in the database.

1. [What is hibernate caching? Explain Hibernate first level cache?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-caching)

As the name suggests, hibernate caches query data to make our application faster. Hibernate Cache can be very useful in gaining fast application performance if used correctly. The idea behind cache is to reduce the number of database queries, hence reducing the throughput time of the application.

Hibernate first level cache is associated with the Session object. Hibernate first level cache is enabled by default and there is no way to disable it. However hibernate provides methods through which we can delete selected objects from the cache or clear the cache completely.  
Any object cached in a session will not be visible to other sessions and when the session is closed, all the cached objects will also be lost.

1. [How to configure Hibernate Second Level Cache using EHCache?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-EHCache)

EHCache is the best choice for utilizing hibernate second level cache. Following steps are required to enable EHCache in hibernate application.

* Add hibernate-ehcache dependency in your maven project, if it’s not maven then add corresponding jars.
* Add below properties in hibernate configuration file
* Create EHCache configuration file, a sample file myehcache.xml would look like below.
* Annotate entity beans with @Cache annotation and caching strategy to use.

1. [What are different states of an entity bean?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#entity-bean-states)

Transient: When an object is never persisted or associated with any session, it’s in transient state. Transient instances may be made persistent by calling save(), persist() or saveOrUpdate(). Persistent instances may be made transient by calling delete().

Persistent: When an object is associated with a unique session, it’s in persistent state. Any instance returned by a get() or load() method is persistent.

Detached: When an object is previously persistent but not associated with any session, it’s in detached state. 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().

1. [What is use of Hibernate Session merge() call?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-merge)

Hibernate merge can be used to update existing values, however this method create a copy from the passed entity object and return it. The returned object is part of persistent context and tracked for any changes, passed object is not tracked.

1. [What is difference between Hibernate save(), saveOrUpdate() and persist() methods?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-save-saveOrUpdate-persist)

Hibernate save can be used to save entity to database. Problem with save() is that it can be invoked without a transaction and if we have mapping entities, then only the primary object gets saved causing data inconsistencies. Also save returns the generated id immediately.

Hibernate persist is similar to save with transaction. I feel it’s better than save because we can’t use it outside the boundary of transaction, so all the object mappings are preserved. Also persist doesn’t return the generated id immediately, so data persistence happens when needed.

Hibernate saveOrUpdate results into insert or update queries based on the provided data. If the data is present in the database, update query is executed. We can use saveOrUpdate() without transaction also, but again you will face the issues with mapped objects not getting saved if session is not flushed.

1. [What will happen if we don’t have no-args constructor in Entity bean?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#entity-bean-constructor)

Hibernate uses [Reflection API](https://www.journaldev.com/1789/java-reflection-example-tutorial) to create instance of Entity beans, usually when you call get() or load() methods. The method Class.newInstance() is used for this and it requires no-args constructor. So if you won’t have no-args constructor in entity beans, hibernate will fail to instantiate it and you will get HibernateException.

1. [What is difference between sorted collection and ordered collection, which one is better?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#sorted-vs-ordered-collection)

When we use Collection API sorting algorithms to sort a collection, it’s called sorted list. For small collections, it’s not much of an overhead but for larger collections it can lead to slow performance and OutOfMemory errors. Also the entity beans should implement Comparable or Comparator interface for it to work, read more at [java object list sorting](https://www.journaldev.com/780/comparable-and-comparator-in-java-example).

If we are using Hibernate framework to load collection data from database, we can use it’s Criteria API to use “order by” clause to get ordered list. Below code snippet shows you how to get it

Ordered list is better than sorted list because the actual sorting is done at database level, that is fast and doesn’t cause memory issues.

1. [What are the collection types in Hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-collection-types)
2. Bag
3. Set
4. List
5. Array
6. Map
7. [How to implement Joins in Hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-joins)

here are various ways to implement joins in hibernate.

Using associations such as one-to-one, one-to-many etc.

Using JOIN in the HQL query. There is another form “join fetch” to load associated data simultaneously, no lazy loading.

We can fire native sql query and use join keyword.

1. [Why we should not make Entity Class final?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#entity-final-class)

Hibernate use proxy classes for lazy loading of data, only when it’s needed. This is done by extending the entity bean, if the entity bean will be final then lazy loading will not be possible, hence low performance.

1. [What is HQL and what are it’s benefits?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hql-benefits)

Hibernate Framework comes with a powerful object-oriented query language – Hibernate Query Language (HQL). It’s very similar to SQL except that we use Objects instead of table names, that makes it more close to object oriented programming.

Hibernate query language is case-insensitive except for java class and variable names. So SeLeCT is the same as sELEct is the same as SELECT, but com.journaldev.model.Employee is not same as com.journaldev.model.EMPLOYEE.

The HQL queries are cached but we should avoid it as much as possible, otherwise we will have to take care of associations. However it’s a better choice than native sql query because of Object-Oriented approach.

1. [What is Query Cache in Hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#query-cache)

Hibernate implements a cache region for queries resultset that integrates closely with the hibernate second-level cache.

This is an optional feature and requires additional steps in code. This is only useful for queries that are run frequently with the same parameters. First of all we need to configure below property in hibernate configuration file.

<property name="hibernate.cache.use\_query\_cache">true</property>

1. [Can we execute native sql query in hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#native-sql-query)

hibernate provide option to execute native SQL queries through the use of SQLQuery object.

For normal scenarios, it is however not the recommended approach because we loose benefits related to hibernate association and hibernate first level caching

1. [What is the benefit of native sql query support in hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#native-sql-benefits)

Native SQL Query comes handy when we want to execute database specific queries that are not supported by Hibernate API such as query hints or the CONNECT keyword in Oracle Database

1. [What is Named SQL Query?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#named-sql-query)

Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.

Hibernate Named Queries can be defined in Hibernate mapping files or through the use of JPA annotations @NamedQuery and @NamedNativeQuery.

1. [What are the benefits of Named SQL Query?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#named-sql-query-benefits).

Hibernate Named Query helps us in grouping queries at a central location rather than letting them scattered all over the code.  
Hibernate Named Query syntax is checked when the hibernate session factory is created, thus making the application fail fast in case of any error in the named queries.  
Hibernate Named Query is global, means once defined it can be used throughout the application.

However one of the major disadvantage of Named query is that it’s hard to debug, because we need to find out the location where it’s defined.

1. [What is the benefit of Hibernate Criteria API?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-criteria-benefits)

Hibernate provides Criteria API that is more object oriented for querying the database and getting results. We can’t use Criteria to run update or delete queries or any DDL statements. It’s only used to fetch the results from the database using more object oriented approach.

Some of the common usage of Criteria API are:

* Criteria API provides Projection that we can use for aggregate functions such as sum(), min(), max() etc.
* Criteria API can be used with ProjectionList to fetch selected columns only.
* Criteria API can be used for join queries by joining multiple tables, useful methods are createAlias(), setFetchMode() and setProjection()
* Criteria API can be used for fetching results with conditions, useful methods are add() where we can add Restrictions.
* Criteria API provides addOrder() method that we can use for ordering the results.

1. [How to log hibernate generated sql queries in log files?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-show-sql)

We can set below property for hibernate configuration to log SQL queries.

<property name="hibernate.show\_sql">true</property>

However we should use it only in Development or Testing environment and turn it off in production environment.

1. [What is Hibernate Proxy and how it helps in lazy loading?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-proxy)

Hibernate uses proxy object to support lazy loading. Basically when you load data from tables, hibernate doesn’t load all the mapped objects. As soon as you reference a child or lookup object via getter methods, if the linked entity is not in the session cache, then the proxy code will go to the database and load the linked object. It uses javassist to effectively and dynamically generate sub-classed implementations of your entity objects.

1. [How to implement relationships in hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-mappings)

We can easily implement one-to-one, one-to-many and many-to-many relationships in hibernate. It can be done using JPA annotations as well as XML based configurations.

1. [How transaction management works in Hibernate?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-transaction-management)
2. Transaction management is very easy in hibernate because most of the operations are not permitted outside of a transaction. So after getting the session from SessionFactory, we can call session beginTransaction() to start the transaction. This method returns the Transaction reference that we can use later on to either commit or rollback the transaction.
3. Overall hibernate transaction management is better than JDBC transaction management because we don’t need to rely on exceptions for rollback. Any exception thrown by session methods automatically rollback the transaction.
4. [What is cascading and what are different types of cascading?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-cascading)
5. None: No Cascading, it’s not a type but when we don’t define any cascading then no operations in parent affects the child.
6. ALL: Cascades save, delete, update, evict, lock, replicate, merge, persist. Basically everything
7. SAVE\_UPDATE: Cascades save and update, available only in hibernate.
8. DELETE: Corresponds to the Hibernate native DELETE action, only in hibernate.
9. DETATCH, MERGE, PERSIST, REFRESH and REMOVE – for similar operations
10. LOCK: Corresponds to the Hibernate native LOCK action.
11. REPLICATE: Corresponds to the Hibernate native REPLICATE action.
12. [How to integrate log4j logging in hibernate application?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-log4j)

Hibernate 4 uses JBoss logging rather than slf4j used in earlier versions. For log4j configuration, we need to follow below steps.

* Add log4j dependencies for maven project, if not maven then add corresponding jar files.
* Create log4j.xml configuration file or log4j.properties file and keep it in the classpath. You can keep file name whatever you want because we will load it in next step.
* For standalone projects, use static block to configure log4j using DOMConfigurator or PropertyConfigurator. For web applications, you can use ServletContextListener to configure it.

1. [How to use application server JNDI DataSource with Hibernate framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-jndi)

For web applications, it’s always best to allow servlet container to manage the connection pool. That’s why we define JNDI resource for DataSource and we can use it in the web application. It’s very easy to use in Hibernate, all we need is to remove all the database specific properties and use below property to provide the JNDI DataSource name.

<property name="hibernate.connection.datasource">java:comp/env/jdbc/MyLocalDB</property>

1. [How to integrate Hibernate and Spring frameworks?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#spring-hibernate)

Spring is one of the most used Java EE Framework and Hibernate is the most popular ORM framework. That’s why Spring Hibernate combination is used a lot in enterprise applications. The best part with using Spring is that it provides out-of-box integration support for Hibernate with Spring ORM module. Following steps are required to integrate Spring and Hibernate frameworks together.

Add hibernate-entitymanager, hibernate-core and spring-orm dependencies.

Create Model classes and corresponding DAO implementations for database operations. Note that DAO classes will use SessionFactory that will be injected by Spring Bean configuration.

If you are using Hibernate 3, you need to configure org.springframework.orm.hibernate3.LocalSessionFactoryBean or org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean in Spring Bean configuration file. For Hibernate 4, there is single class org.springframework.orm.hibernate4.LocalSessionFactoryBean that should be configured.

Note that we don’t need to use Hibernate Transaction Management, we can leave it to Spring declarative transaction management using @Transactional annotation.

1. [What is HibernateTemplate class?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernatetemplate-class)

When Spring and Hibernate integration started, Spring ORM provided two helper classes – HibernateDaoSupport and HibernateTemplate. The reason to use them was to get the Session from Hibernate and get the benefit of Spring transaction management. However from Hibernate 3.0.1, we can use SessionFactory getCurrentSession() method to get the current session and use it to get the spring transaction management benefits. If you go through above examples, you will see how easy it is and that’s why we should not use these classes anymore.

1. [How to integrate Hibernate with Servlet or Struts2 web applications?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-servlet)

Hibernate integration with Servlet or Struts2 needs to be done using ServletContextListener

1. [Which design patterns are used in Hibernate framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-design-patterns)

Some of the design patterns used in Hibernate Framework are:

Domain Model Pattern – An object model of the domain that incorporates both behavior and data.

Data Mapper – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.

[Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern) for lazy loading

[Factory pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java) in SessionFactory

1. [What are best practices to follow with Hibernate framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-best-practices)

Some of the best practices to follow in Hibernate are:

* Always check the primary key field access, if it’s generated at the database layer then you should not have a setter for this.
* By default hibernate set the field values directly, without using setters. So if you want hibernate to use setters, then make sure proper access is defined as @Access(value=AccessType.PROPERTY).
* If access type is property, make sure annotations are used with getter methods and not setter methods. Avoid mixing of using annotations on both filed and getter methods.

Use native sql query only when it can’t be done using HQL, such as using database specific feature.

If you have to sort the collection, use ordered list rather than sorting it using Collection API.

Use named queries wisely, keep it at a single place for easy debugging. Use them for commonly used queries only. For entity specific query, you can keep them in the entity bean itself.

For web applications, always try to use JNDI DataSource rather than configuring to create connection in hibernate.

Avoid Many-to-Many relationships, it can be easily implemented using bidirectional One-to-Many and Many-to-One relationships.

For collections, try to use Lists, maps and sets. Avoid array because you don’t get benefit of lazy loading.

1. [What is Hibernate Validator Framework?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-validator)

Data validation is integral part of any application. You will find data validation at presentation layer with the use of Javascript, then at the server side code before processing it. Also data validation occurs before persisting it, to make sure it follows the correct format.

Validation is a cross cutting task, so we should try to keep it apart from our business logic. That’s why JSR303 and JSR349 provides specification for validating a bean by using annotations. Hibernate Validator provides the reference implementation of both these bean validation specs.

1. [What is the benefit of Hibernate Tools Eclipse plugin?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-tools-eclipse-plugin)

Hibernate Tools plugin helps us in writing hibernate configuration and mapping files easily. The major benefit is the content assist to help us with properties or xml tags to use. It also validates them against the Hibernate DTD files, so we know any mistakes before hand

Maven

1. **Explain what is Maven? How does it work?**

Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.

1. **List out what are the aspects does Maven Manages?**

Maven handles following activities of a developer

• Build  
• Documentation  
• Reporting  
• Dependencies  
• SCMs  
• Releases  
• Distribution  
• Mailing list

1. **Mention the three build lifecycle of Maven?**

• Clean: Cleans up artifacts that are created by prior builds  
• Default (build): Used to create the application  
• Site: For the project generates site documentatio

1. **Explain what is POM?**

POM (Project Object Model) is the fundamental unit of work. It is an XML file which holds the information about the project and configuration details used to build a project by Maven.

1. **Explain what is Maven artifact?**

Usually an artifact is a JAR file which gets arrayed to a Maven repository. One or more artifacts a maven build produces such as compiled JAR and a sources JAR.Each artifact includes a group ID, an artifact ID and a version string.

1. **Explain what is Maven Repository? What are their types?**

A Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven

1. **Why Maven Plugins are used?**

Maven plugins are used to  
• Create a jar file  
• Create war file  
• Compile code files  
• Unit testing of code  
• Documenting projects  
• Reporting

1. **List out the dependency scope in Maven?**

The various dependency scope used in Maven are:

• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project  
• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime  
• Runtime: This tells that the dependency is not needed for compilation but is required during execution  
• Test: It says dependency is available only for the test compilation and execution phases  
• System: It indicates you have to provide the system path  
• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM’s section

1. **Mention how profiles are specified in Maven?**

Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

1. **Explain how you can exclude dependency?**

By using the exclusion element, dependency can be excluded

1. **Mention the difference between Apache Ant and Maven?**

Apache Ant Maven  
• Ant is a toolbox – Maven is a framework  
• Ant does not have formal conventions like project directory structure – Maven has conventions  
• Ant is procedural; you have to tell to compile, copy and compress – Maven is declarative ( information on what to make & how to build)  
• Ant does not have lifecycle; you have to add sequence of tasks manually – Maven has a lifecycle  
• Ant scripts are not reusable – Maven plugins are reusable

1. **In Maven what are the two setting files called and what are their location?**

In Maven, the setting files are called settings.xml, and the two setting files are located at

• Maven installation directory: $M2\_Home/conf/settings.xml  
 • User’s home directory: ${ user.home }/ .m2 / settings.xml

1. **List out what are the build phases in Maven?**

Build phases in Maven are

• Validate  
 • Compile  
 • Test  
 • Package  
 • Install  
 • Deploy

1. **List out the build, source and test source directory for POM in Maven?**

• Build = Target  
• Source = src/main/java  
• Test = src/main/test

1. **Where do you find the class files when you compile a Maven project?**

You will find the class files ${basedir}/target/classes/.

1. **Explain what would the “jar: jar” goal do?**

jar: jar will not recompile sources; it will imply just create a JAR from the target/classes directory considering that everything else has been done

1. **List out what are the Maven’s order of inheritance?**

The maven’s order of inheritance is

• Parent Pom  
 • Project Pom  
 • Settings  
 • CLI parameters

1. **For POM what are the minimum required elements?**

The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version

1. **Explain how you can produce execution debug output or error messages?**

To produce execution debug output you could call Maven with X parameter or e parameter

1. **Explain how to run test classes in Maven?**

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”