Hibernate:

* Hibernate is Java framework that simplifies the development of Java application to interact with the DB.
* It is ORM tool, Open source and lightweight[ because it mostly based on Pojo’s
* It is a non-invasive [not force] framework, means it won’t forces the programmers to extend/ implement any class/interface.
* Traditional way to save Data (JDBC): Java Object-> JDBC API/JDBC driver -> DB: [ here we need to write query to store data in DB, need to write DAO classes]
* Using Hibernate: Java object -> Hibernate-> DB [ here hibernate will do all operations for us we don’t need to write query manually to save the data with the help of ORM we just need to do mapping and call the function, No need to write DAO classes.]

ORM

* Object relationship mapping: technique to used to map Java Objects to DB tables.
* It allows developers to work with databases using Object-oriented programming concepts, making it easier to interact with relational databases.
* 

JPA: Java persistence API

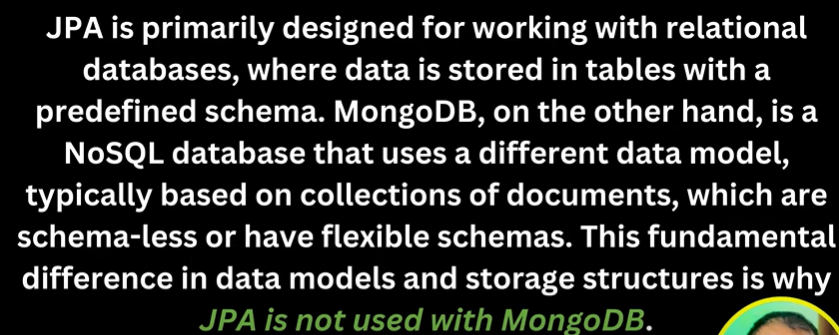
* It is a way to achieve ORM, includes interfaces and annotations that you use in your Java classes, requires a persistence provider (ORM tools) for implementation.

Persistence Provider/ ORM tools:

* To use JPA, we need a persistence provider. A persistence provider is a specific implementation of the JPA specification.
* Example: of JPA persistence provider providers include Hibernate, EclipseLink and OpenJPA. These providers implement the JPA interfaces and provide the underlying functionality to interact with DB.

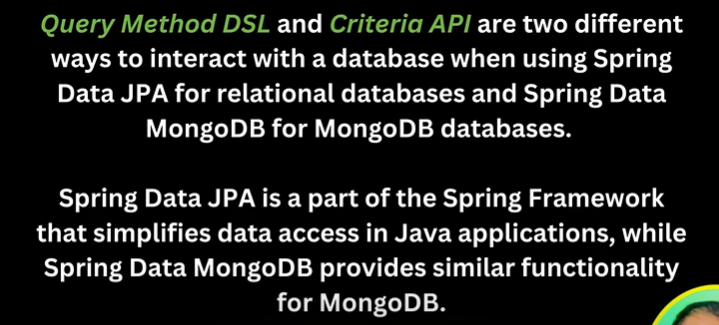
Spring Data JPA:

* It is built on top of the JPA specification, but it is not a JPA implementation itseft. Instead, It simplifies working with JPA by providing higher-level abstractions and utilities. However, to use Spring Data JPA effectively, you still need a JPA implementation, such as Hibernate, EclipseLink, or another JPA-compliant provider, to handle the actual database interactions.
* Query Method DSL : way t create queries based on method naming conventions.
* Criteria API: Offers a more dynamic and programmatic approach for building complex and custom queries.



A black background with white text

AI-generated content may be incorrect.



Hibernate Interview Questions:

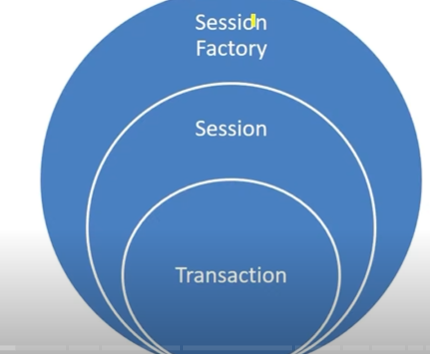
1. What is hibernate and why to use it:

* It is an Object Relational Mapping tool used to map Java Objects and database tables.
* How: It provides JPA implementation hence we can use JPA annotations as well as xml configurations to achieve this mapping.
* JPA Annotations: Hibernate is an implementation of JPA. Just like JPA in an interface and hibernate is class, JPA provides what to do but how to do it will decide hibernate.

1. Why Hibernate:

* Hibernate eliminates all the boiler-plate[code which required to make JDBC connection with DB] code that comes with JDBC.
* Supports HQL with more object oriented.
* It provides transactions management implicitly.
* It throws JDBCException or HibernateException which are the unchecked exceptions, so we don’t need to worry about handling using try and catch.
* It supports caching for better performance.

1. What are the interfaces you have used from Hibernate:

* SessionFactory (org.hibernate.SessionFactory): Instance of this used to retrieve Session objects for database operations. We need to initialize that once and can cache it to reuse it again and again. Its like one SessionFactory object per Database connection. Like 1 for mysql, different 1 for oracle.
* Session (org.hibernate.Session) : Its factory for transaction, it’s used for connecting application with persistent store like hibernate framework/DB. It is used to get a physical connection with the database. It also provides methods for CRUD operations.
* Transaction (org.hibernate.Transaction) : This specifies single/ atomic units of work.
* 
* A computer screen shot of a code

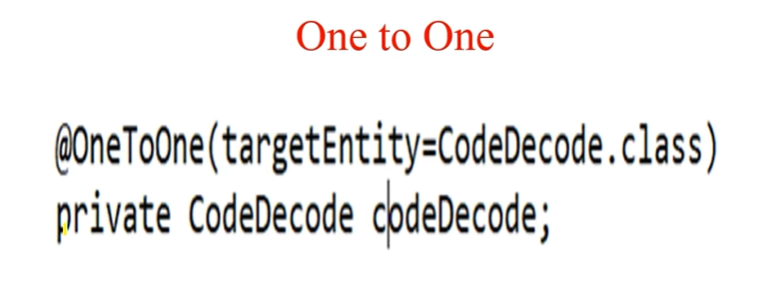
  AI-generated content may be incorrect.

1. What are Annotations used in Hibernate: all annotations present in javax.persistence package.

* @Entity: Used with model/pojo classes to specify that they are entity beans.
* @Table: Used with entity beans to define the corresponding table name in Database.
* @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. E.g. @Access(value=AccessType.PROPERTY)
* @Id: used to define the primary key in the entity bean.
* @EmbeddedId: used to define composite primary key in the entity bean.
* @Column: used to define the column name in database table.
* @GeneratedValue: Used to define the strategy to be used for generating of primary key. Used in conjunction with javax.persistence.GenerationType enum eg: GenerationType.IDENTITY.
* @OneToOne: used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany.
* @Cascade: used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType.
* @PrimaryKeyJoinColumn: Used to define the property for foreign key Used with org.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter

1. Mappings in Hibernate: 3 types

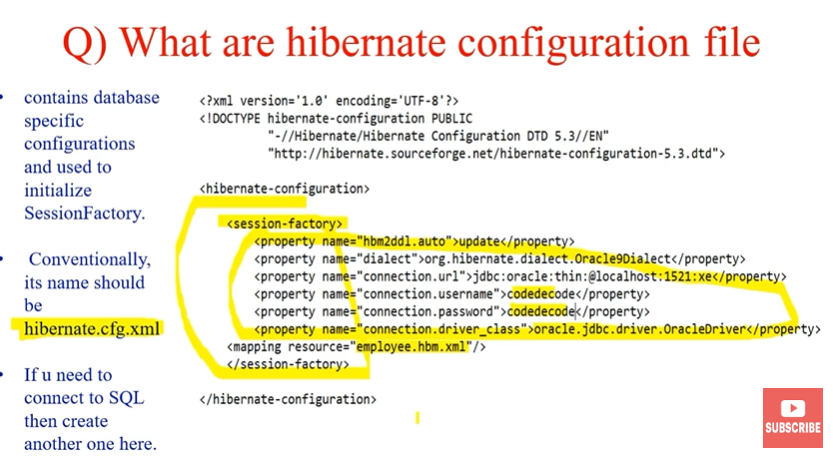
* OneToOne
* ManyToOne
* ManyToMany
* A computer code with red text

  AI-generated content may be incorrect.

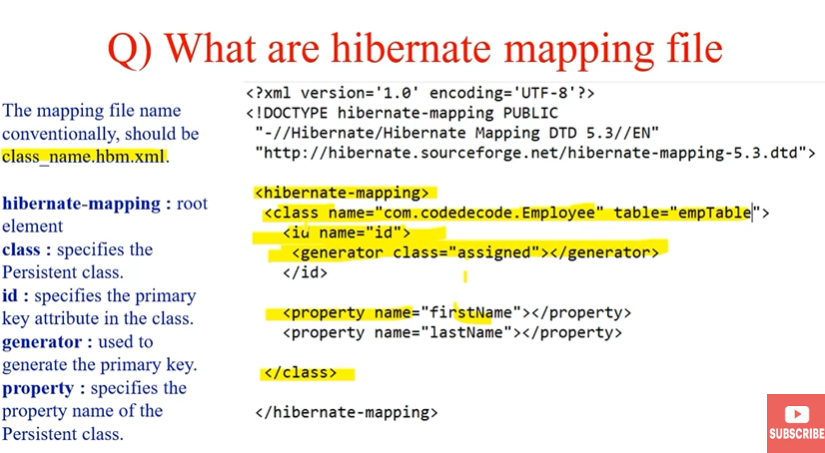
A computer code with text

AI-generated content may be incorrect.

1. What are hibernate configuration file:

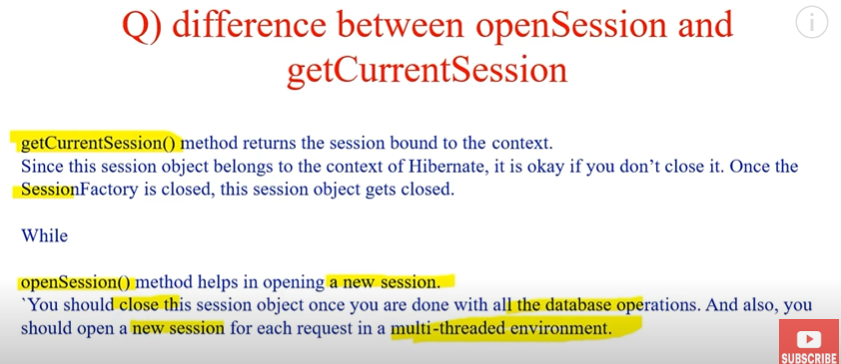
* 

1. What are hibernate Mapping file:

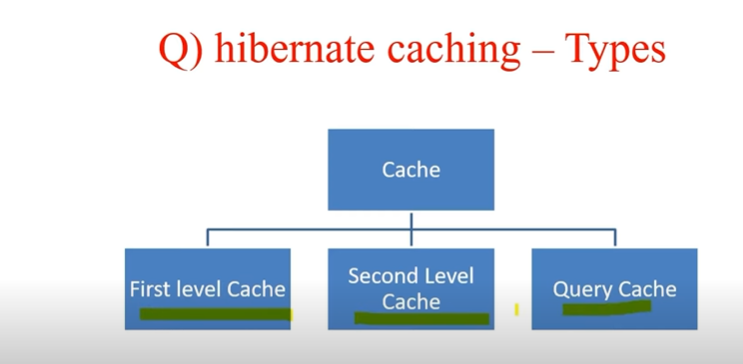


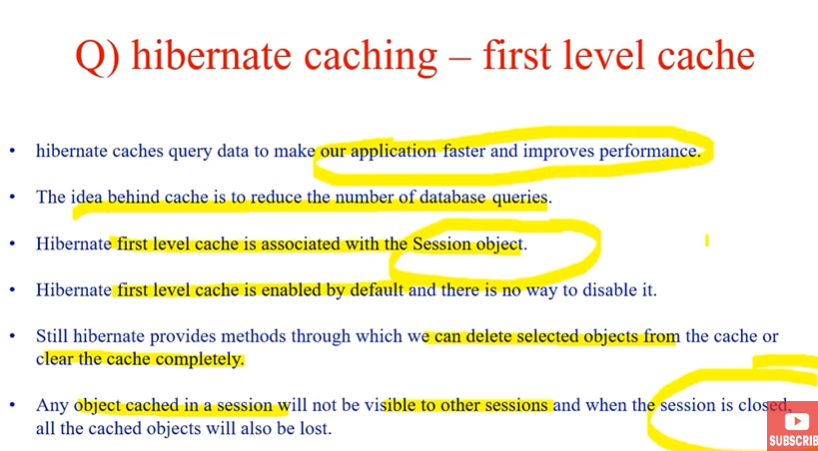
1. Steps to create app of hibernateA diagram of a application

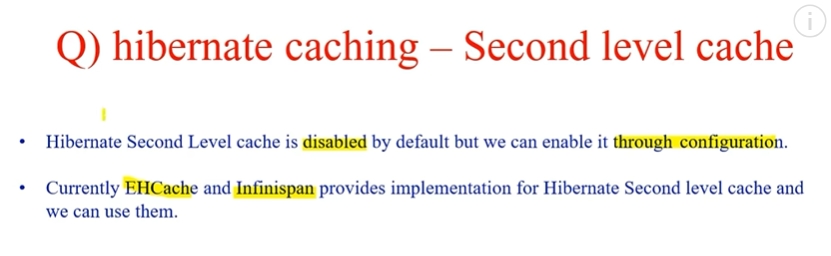
   AI-generated content may be incorrect.

A white background with blue text

AI-generated content may be incorrect.







A screen shot of a computer

AI-generated content may be incorrect.

A close-up of a computer code

AI-generated content may be incorrect.

