**Hibernate** is an **Object-Relational Mapping (ORM) framework** for Java that simplifies database interactions by mapping Java objects to database tables. It allows developers to work with databases using Java objects instead of writing complex SQL queries.

**🔹 Key Features of Hibernate**

1. **ORM (Object-Relational Mapping)** – Maps Java classes to database tables.
2. **Eliminates JDBC Boilerplate Code** – Reduces the need for writing raw SQL queries.
3. **Automatic Table Creation** – Can generate database tables based on entity classes.
4. **Supports HQL (Hibernate Query Language)** – A database-independent query language.
5. **Caching Mechanism** – Improves performance by reducing database access.
6. **Transaction Management** – Works seamlessly with Java transactions.
7. **Lazy and Eager Loading** – Controls how data is fetched from the database.

**🔹 How Hibernate Works?**

1. **Define Entity Classes** – Create Java classes and annotate them with @Entity, @Table, etc.
2. **Configure Hibernate** – Use hibernate.cfg.xml to set up database connection.
3. **Create SessionFactory** – Establishes a connection with the database.
4. **Perform CRUD Operations** – Use Hibernate's API to insert, update, delete, and retrieve records.

**🔹 Basic Example of Hibernate in Java**

@Entity

@Table(name = "students")

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String email;

// Getters and Setters

}

SessionFactory factory = new Configuration().configure("hibernate.cfg.xml")

.addAnnotatedClass(Student.class)

.buildSessionFactory();

Session session = factory.getCurrentSession();

try {

Student student = new Student("John", "john@example.com");

session.beginTransaction();

session.save(student);

session.getTransaction().commit();

} finally {

factory.close();

}

**🔹 Why Use Hibernate?**

✔ Reduces development time  
✔ Eliminates SQL dependency  
✔ Supports multiple databases  
✔ Handles transactions automatically

Since you are working on **Java Full Stack projects**, Hibernate might be useful for database handling. However, since you mentioned avoiding Hibernate for your **Online Quiz System**, you can use **JDBC** instead.

The **hibernate.cfg.xml** file is the main configuration file in Hibernate, where you define the database connection details and Hibernate properties.

**📌 Basic Structure of hibernate.cfg.xml**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database Connection Settings -->

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/your\_database</property>

<property name="hibernate.connection.username">your\_username</property>

<property name="hibernate.connection.password">your\_password</property>

<!-- Hibernate Dialect (for MySQL) -->

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

<!-- Show SQL Queries in Console -->

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

<!-- Automatically Create or Update Tables -->

<property name="hibernate.hbm2ddl.auto">update</property>

<!-- Mapping Classes (Entities) -->

<mapping class="com.example.model.Student"/>

<mapping class="com.example.model.Course"/>

</session-factory>

</hibernate-configuration>

**📌 Explanation of Properties**

| **Property** | **Description** |
| --- | --- |
| hibernate.connection.driver\_class | Specifies the JDBC driver for the database (MySQL in this case). |
| hibernate.connection.url | Database URL (jdbc:mysql://localhost:3306/db\_name). |
| hibernate.connection.username | Username for the database. |
| hibernate.connection.password | Password for the database. |
| hibernate.dialect | Specifies SQL dialect (MySQL8Dialect, PostgreSQLDialect, etc.). |
| hibernate.show\_sql | Prints SQL queries to the console (true/false). |
| hibernate.hbm2ddl.auto | Automatically updates or creates tables (update, create, validate, none). |
| <mapping class="..."> | Specifies entity classes to be mapped. |

**📌 Steps to Use hibernate.cfg.xml**

1. **Place the hibernate.cfg.xml file in src/main/resources/** (for Maven projects).
2. **Load the configuration in Java:**

java

CopyEdit

SessionFactory factory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

1. **Perform CRUD operations using Hibernate.**