*Hands-On-4*

**Difference between JPA, Hibernate and Spring Data-JPA**

**Solution:**

**Java Persistence API (JPA)** is a standard specification (JSR 338) that defines a set of interfaces and rules for persisting, retrieving, and managing data between Java objects and a relational database. It acts as a contract or guideline for ORM (Object Relational Mapping) but does not provide any actual implementation by itself. To work with JPA, a provider like Hibernate or EclipseLink is needed, which implements the defined interfaces like EntityManager, EntityTransaction, and Query.

**Hibernate** is a widely-used ORM framework that implements the JPA specification. It not only fulfills the requirements of JPA but also extends its capabilities with additional features such as lazy loading, caching, and more powerful query options. Hibernate manages database operations by converting Java objects into SQL queries and vice versa. In traditional usage, developers are required to handle session management and transactions explicitly, making the code more verbose but also more flexible when fine-tuned control is needed.

**Spring Data JPA** builds on top of JPA and provides a higher-level abstraction that eliminates much of the boilerplate code required when using plain JPA or Hibernate. It does not provide a JPA implementation itself but uses a JPA provider like Hibernate under the hood. Through interfaces like JpaRepository, Spring Data JPA enables automatic implementation of common operations such as save, delete, and find, and handles transaction management transparently. This makes development faster and cleaner by reducing the amount of repetitive code developers need to write.

**How can Hibernate be used ?**

1. To get started let's just create a table to demonstrate the same.

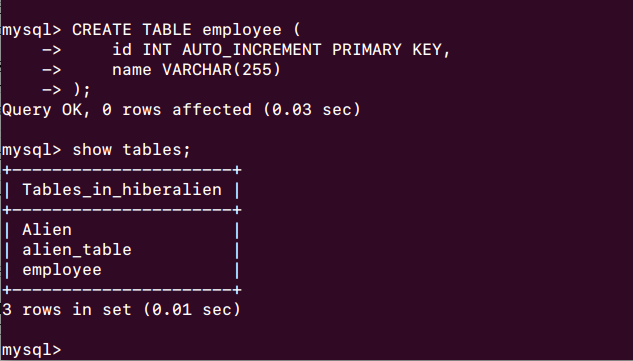
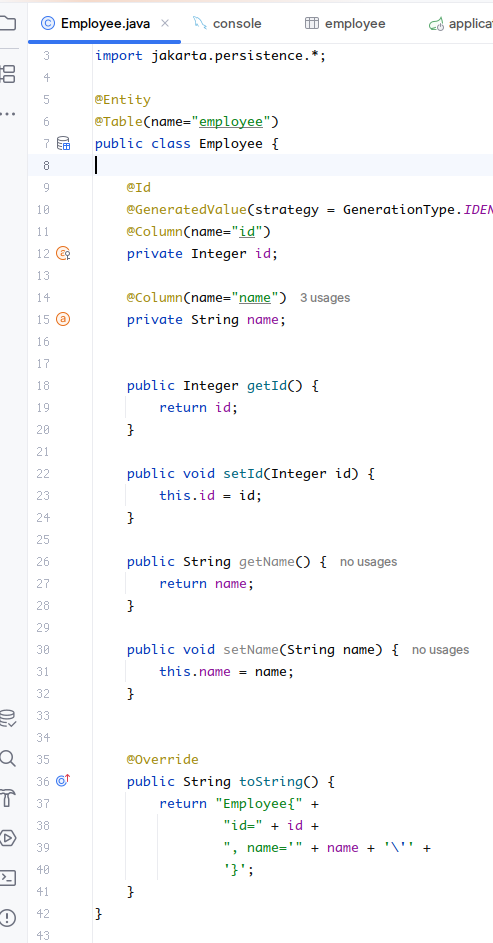
CREATE TABLE employee (

id INT AUTO\_INCREMENT PRIMARY KEY,

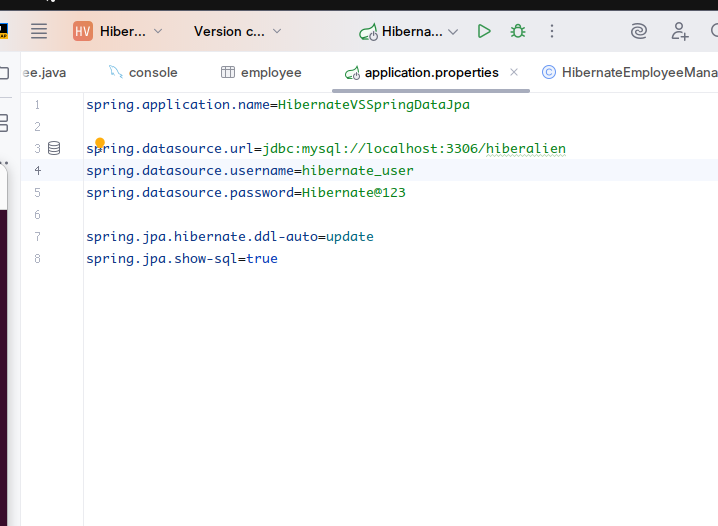
name VARCHAR(255)

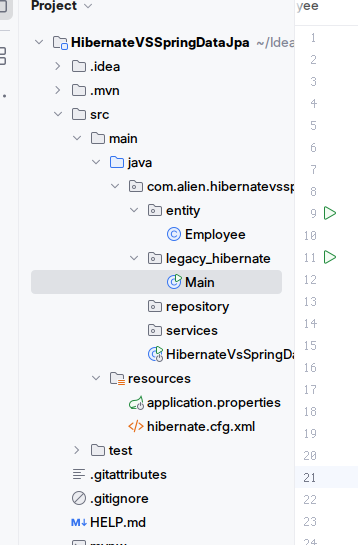
);

1. The entity class can be shown as :



1. Add data source database, url and credentials to the application.properties file





1. Create legacy\_file package to implement driver code to be used with hibernate

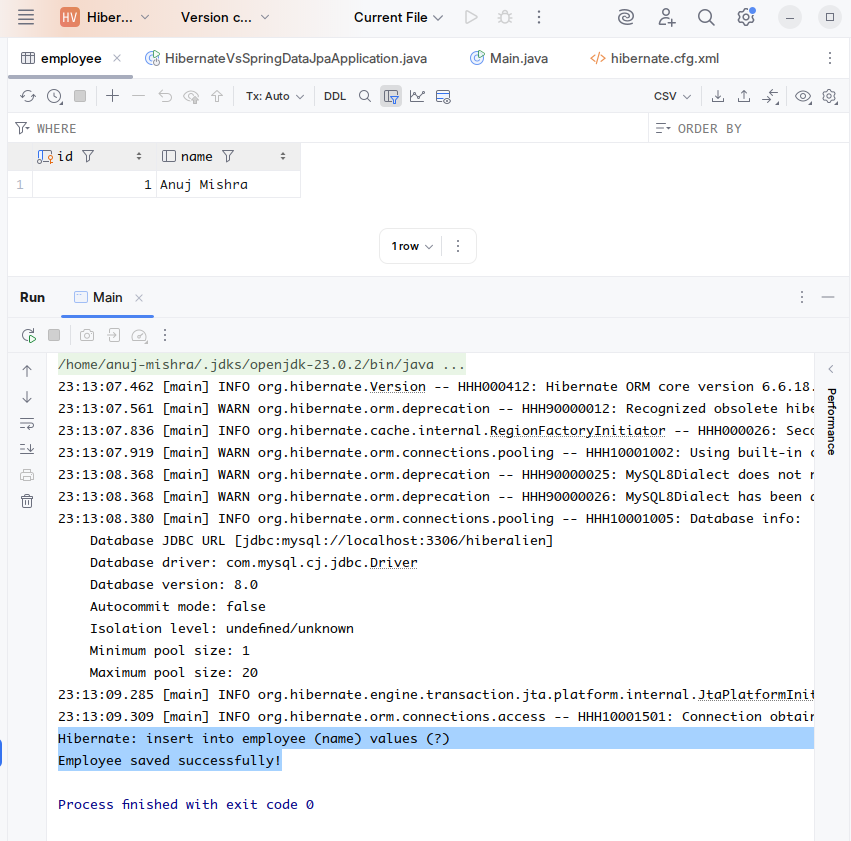
Implement file Main.java to use hibernate, at the same time consider the entity created already.



1. Create hibernate.cfg.xml at src/resources ->



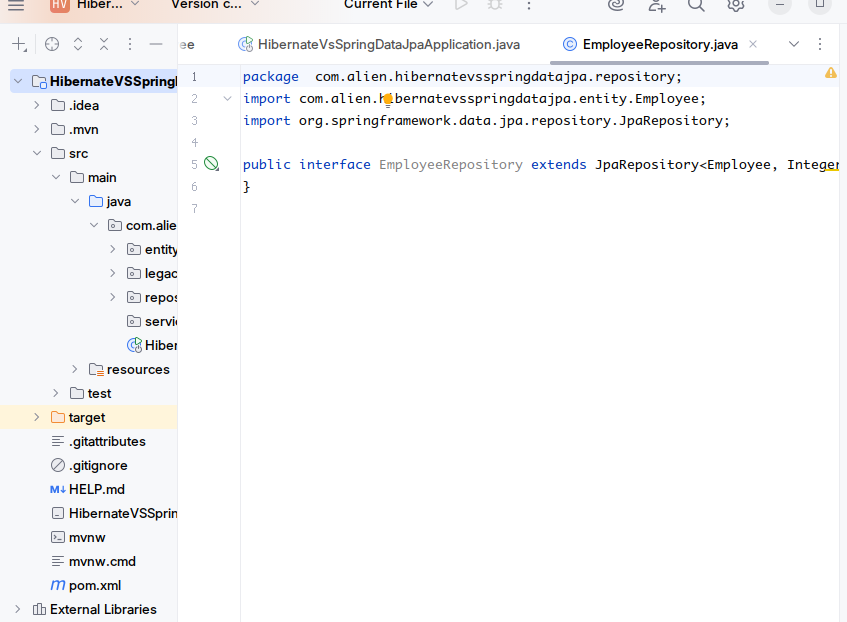
1. Test the hibernate by saving the tuple in the database using implemented code :

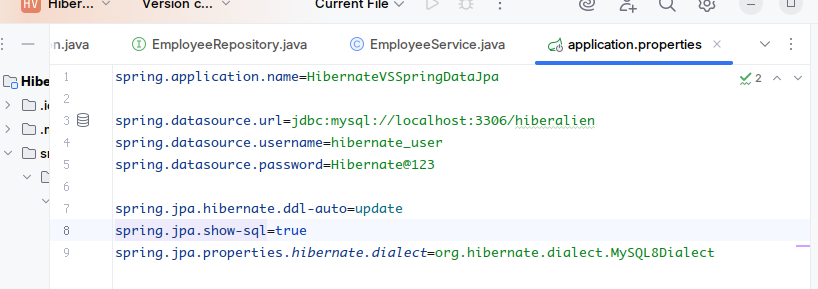
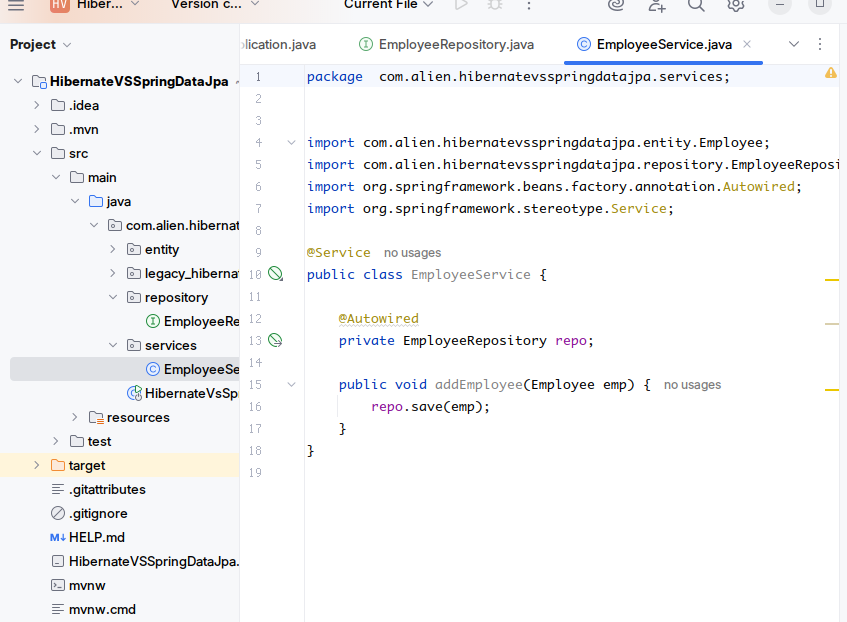


1. The code runs successfully and adds the employee as Anuj Mishra with automatically generated value as 1.

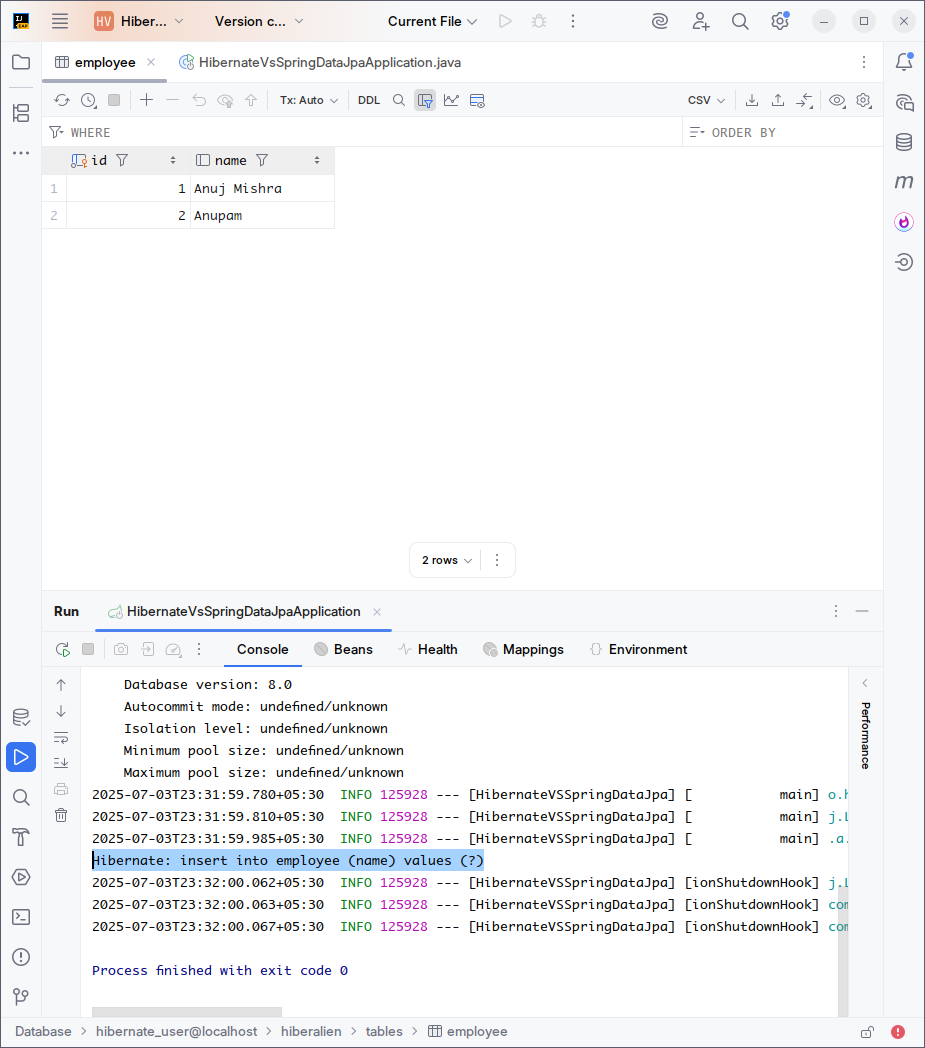
**How Spring DataJPA can be used ?**

1. Create repository interface **EmployeeRepository**



1. Create Service layer for employee, as employee Service and implement method named as addEmployee(...) calling the JPA method .save(...)
2. Edit the application.properties to use the hibernate dialect, (previously we used it on hibernate.cfg.xml but now it would be used here )
3. Test the implementation by non-production code in the main class itself.





It can be clearly seen that the program already added the new Employee “Anupam” with auto generated kay as 2.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*