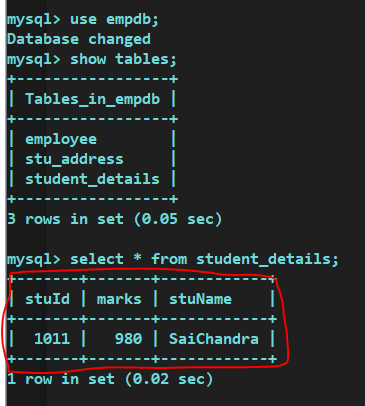
# Fetching the Data

Writing a java program to fetch the following data:



Student.java(pojo class):

**package** com.hibersessions;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

//@Entity: Is an annotation which creates or updates the table in the database. Takes class name as table name by default.

//@Entity(name = "student\_details") : Entity with table name

//@Id: To make that variable as primary key column.

@Entity(name = "student\_details")

**public** **class** Student {

@Id

**private** **int** stuId;

**private** String stuName;

**private** **int** marks;

**public** **int** getStuId() {

**return** stuId;

}

**public** **void** setStuId(**int** stuId) {

**this**.stuId = stuId;

}

**public** String getStuName() {

**return** stuName;

}

**public** **void** setStuName(String stuName) {

**this**.stuName = stuName;

}

**public** **int** getMarks() {

**return** marks;

}

**public** **void** setMarks(**int** marks) {

**this**.marks = marks;

}

**public** String toString() {

**return** stuId + " " + stuName + " " + marks;

}

}

Hibernate.cfg.xml:

<?xml version = *"1.0"* encoding = *"utf-8"*?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/empdb?useSSL=false</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>root</property>

<property name=*"dialect"*>org.hibernate.dialect.MySQL5Dialect</property>

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

<property name=*"show\_sql"*>true</property>

<!-- tag added to create a new table from the class. -->

<mapping class=*"com.hibersessions.Student"* />

<mapping class=*"com.hibersessions.Address"* />

</session-factory>

</hibernate-configuration>

Main class:

**package** com.hibersessions;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**public** **class** FetchData\_3 {

**public** **static** **void** main(String[] args) {

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

System.***out***.println(factory);

// Triggering the session

Session session = factory.openSession();

//Fetch the Student data using 'get' method

Student stu = session.get(Student.**class**, 1011);

System.***out***.println(stu);

session.close();

System.***out***.println("\nCompleted Fetching !!!");

}

}

Console Output:

Oct 10, 2023 12:33:41 PM org.hibernate.Version logVersion

INFO: HHH000412: Hibernate ORM core version 5.6.15.Final

Oct 10, 2023 12:33:42 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>

INFO: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}

Oct 10, 2023 12:33:42 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)

Oct 10, 2023 12:33:42 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/empdb?useSSL=false]

Oct 10, 2023 12:33:42 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001001: Connection properties: {password=\*\*\*\*, user=root}

Oct 10, 2023 12:33:42 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001003: Autocommit mode: false

Oct 10, 2023 12:33:42 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>

INFO: HHH000115: Hibernate connection pool size: 20 (min=1)

Oct 10, 2023 12:33:43 PM org.hibernate.dialect.Dialect <init>

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect

Oct 10, 2023 12:33:44 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection

INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@1dbd580] for (non-JTA) DDL execution was not in auto-commit mode; the Connection 'local transaction' will be committed and the Connection will be set into auto-commit mode.

Oct 10, 2023 12:33:44 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService

INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]

org.hibernate.internal.SessionFactoryImpl@46383a78

Hibernate: select student0\_.stuId as stuid1\_1\_0\_, student0\_.marks as marks2\_1\_0\_, student0\_.stuName as stuname3\_1\_0\_ from student\_details student0\_ where student0\_.stuId=?

1011 SaiChandra 980

Completed Fetching !!!

Using ‘load()’ function:

Hibernate.cfg.xml:

<?xml version = *"1.0"* encoding = *"utf-8"*?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/empdb?useSSL=false</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>root</property>

<property name=*"dialect"*>org.hibernate.dialect.MySQL5Dialect</property>

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

<property name=*"show\_sql"*>true</property>

<!-- tag added to create a new table from the class. -->

<mapping class=*"com.hibersessions.Student"* />

<mapping class=*"com.hibersessions.Address"* />

</session-factory>

</hibernate-configuration>

Address.java(pojo):

**package** com.hibersessions;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Lob;

**import** javax.persistence.Table;

**import** javax.persistence.Temporal;

**import** javax.persistence.TemporalType;

**import** javax.persistence.Transient;

@Entity

@Table(name = "stu\_address")

**public** **class** Address {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "address\_id")

**private** **int** addressId;

@Column(length = 50, name = "stu\_street")

**private** String street;

@Column(length = 30, name = "CityName")

**private** String city;

**private** **boolean** isOpen;

@Transient

**private** **double** tollCharges;

@Column(name = "street\_renvo\_dt")

@Temporal(TemporalType.***DATE***) // displays only date but not time

**private** Date renovationDate;

@Lob

**private** **byte**[] image;

**public** Address() {

**super**();

}

**public** Address(**int** addressId, String street, String city, **boolean** isOpen, **double** tollCharges, Date renovationDate,

**byte**[] image) {

**super**();

**this**.addressId = addressId;

**this**.street = street;

**this**.city = city;

**this**.isOpen = isOpen;

**this**.tollCharges = tollCharges;

**this**.renovationDate = renovationDate;

**this**.image = image;

}

**public** **int** getAddressId() {

**return** addressId;

}

**public** **void** setAddressId(**int** addressId) {

**this**.addressId = addressId;

}

**public** String getStreet() {

**return** street;

}

**public** **void** setStreet(String street) {

**this**.street = street;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** **boolean** isOpen() {

**return** isOpen;

}

**public** **void** setOpen(**boolean** isOpen) {

**this**.isOpen = isOpen;

}

**public** **double** getTollCharges() {

**return** tollCharges;

}

**public** **void** setTollCharges(**double** tollCharges) {

**this**.tollCharges = tollCharges;

}

**public** Date getRenovationDate() {

**return** renovationDate;

}

**public** **void** setRenovationDate(Date renovationDate) {

**this**.renovationDate = renovationDate;

}

**public** **byte**[] getImage() {

**return** image;

}

**public** **void** setImage(**byte**[] image) {

**this**.image = image;

}

**public** String toString() {

**return** addressId + " " + street + " " + city + " " + isOpen + " " + renovationDate;

}

}

Main class:

**package** com.hibersessions;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**public** **class** FetchData\_3 {

**public** **static** **void** main(String[] args) {

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

System.***out***.println(factory);

// Triggering the session

Session session = factory.openSession();

//Fetch the Student data using 'get' method

Student stu = session.get(Student.**class**, 1011);

System.***out***.println(stu);

System.***out***.println("\n");

//We can also use 'load()' function just like 'get()'

Address add = session.load(Address.**class**, 2);

System.***out***.println(add);

session.close();

System.***out***.println("\nCompleted Fetching !!!");

}

}

Console output:

Oct 10, 2023 1:01:01 PM org.hibernate.Version logVersion

INFO: HHH000412: Hibernate ORM core version 5.6.15.Final

Oct 10, 2023 1:01:02 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>

INFO: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}

Oct 10, 2023 1:01:02 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)

Oct 10, 2023 1:01:02 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/empdb?useSSL=false]

Oct 10, 2023 1:01:02 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001001: Connection properties: {password=\*\*\*\*, user=root}

Oct 10, 2023 1:01:02 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001003: Autocommit mode: false

Oct 10, 2023 1:01:02 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>

INFO: HHH000115: Hibernate connection pool size: 20 (min=1)

Oct 10, 2023 1:01:03 PM org.hibernate.dialect.Dialect <init>

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect

Oct 10, 2023 1:01:04 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection

INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@1dbd580] for (non-JTA) DDL execution was not in auto-commit mode; the Connection 'local transaction' will be committed and the Connection will be set into auto-commit mode.

Oct 10, 2023 1:01:04 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService

INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]

org.hibernate.internal.SessionFactoryImpl@46383a78

Hibernate: select student0\_.stuId as stuid1\_1\_0\_, student0\_.marks as marks2\_1\_0\_, student0\_.stuName as stuname3\_1\_0\_ from student\_details student0\_ where student0\_.stuId=?

1011 SaiChandra 980

Hibernate: select address0\_.address\_id as address\_1\_0\_0\_, address0\_.CityName as cityname2\_0\_0\_, address0\_.image as image3\_0\_0\_, address0\_.isOpen as isopen4\_0\_0\_, address0\_.street\_renvo\_dt as street\_r5\_0\_0\_, address0\_.stu\_street as stu\_stre6\_0\_0\_ from stu\_address address0\_ where address0\_.address\_id=?

2 Dabagardens null true 2023-10-09

Completed Fetching !!!

Database Data:

