**Hibernate**

* Hibernate is there to develop persistence logic.
* Like c , c++, java are the programming lang . jsp ,jdbc , servlet are the technologies in java similar hibernate is framework.

**Persistence**

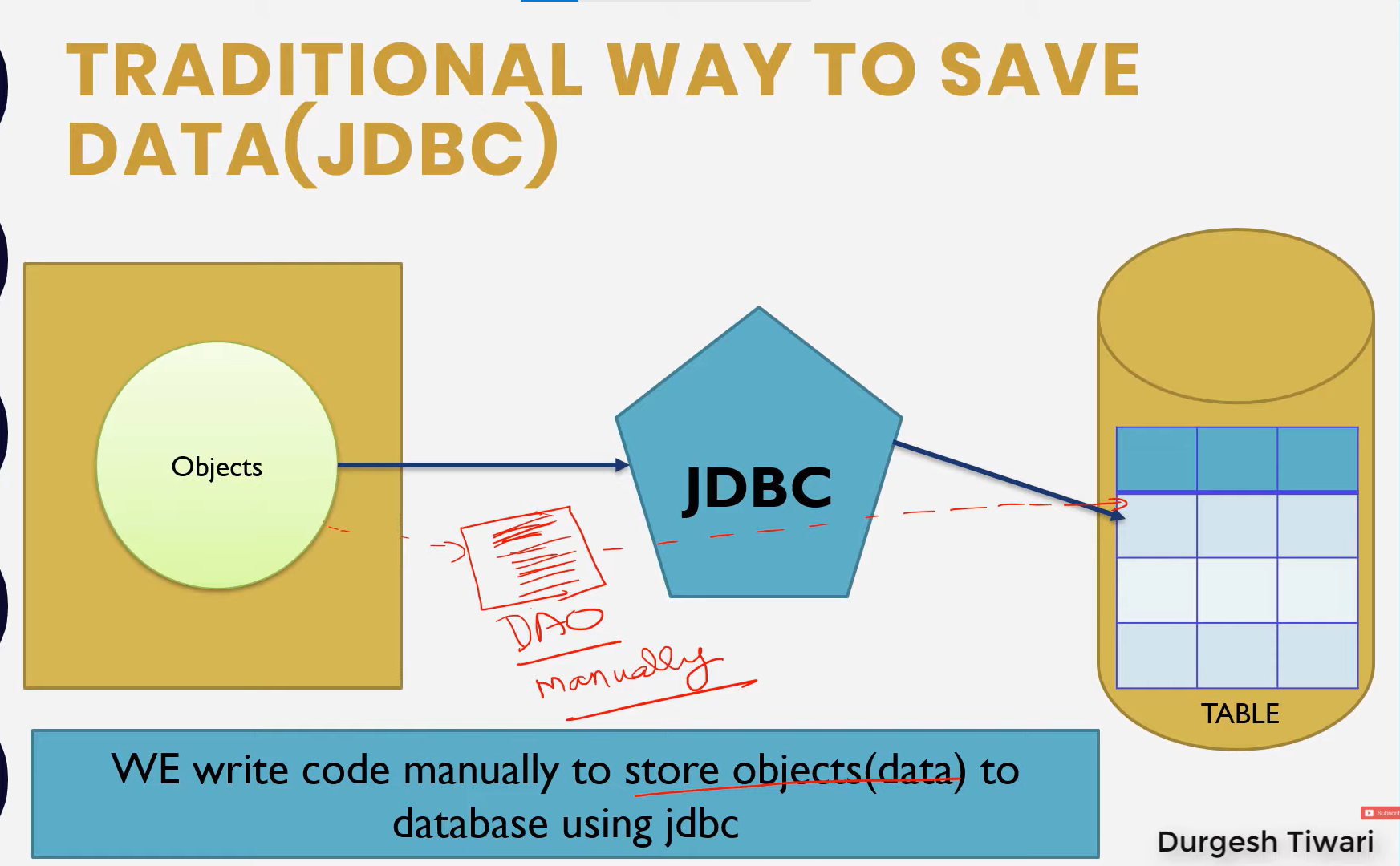
a process of storing & manage data for long time is called as persistence.

**Terminologies of persistence**

* 1. Persistence store: e.g. Files, db s/w where we can store our data
  2. Persistence data: e.g. file data, db table records in which format we can store our data.
  3. Persistence operations: e.g. Insert, update, delete and select operations that we can perform on data that is crud operations.
  4. Persistence logic : e.g. I/O streams , jdbc code, hibernate code
  5. Persistence technologies/framework: e.g. Hibernate , spring , spring Boot

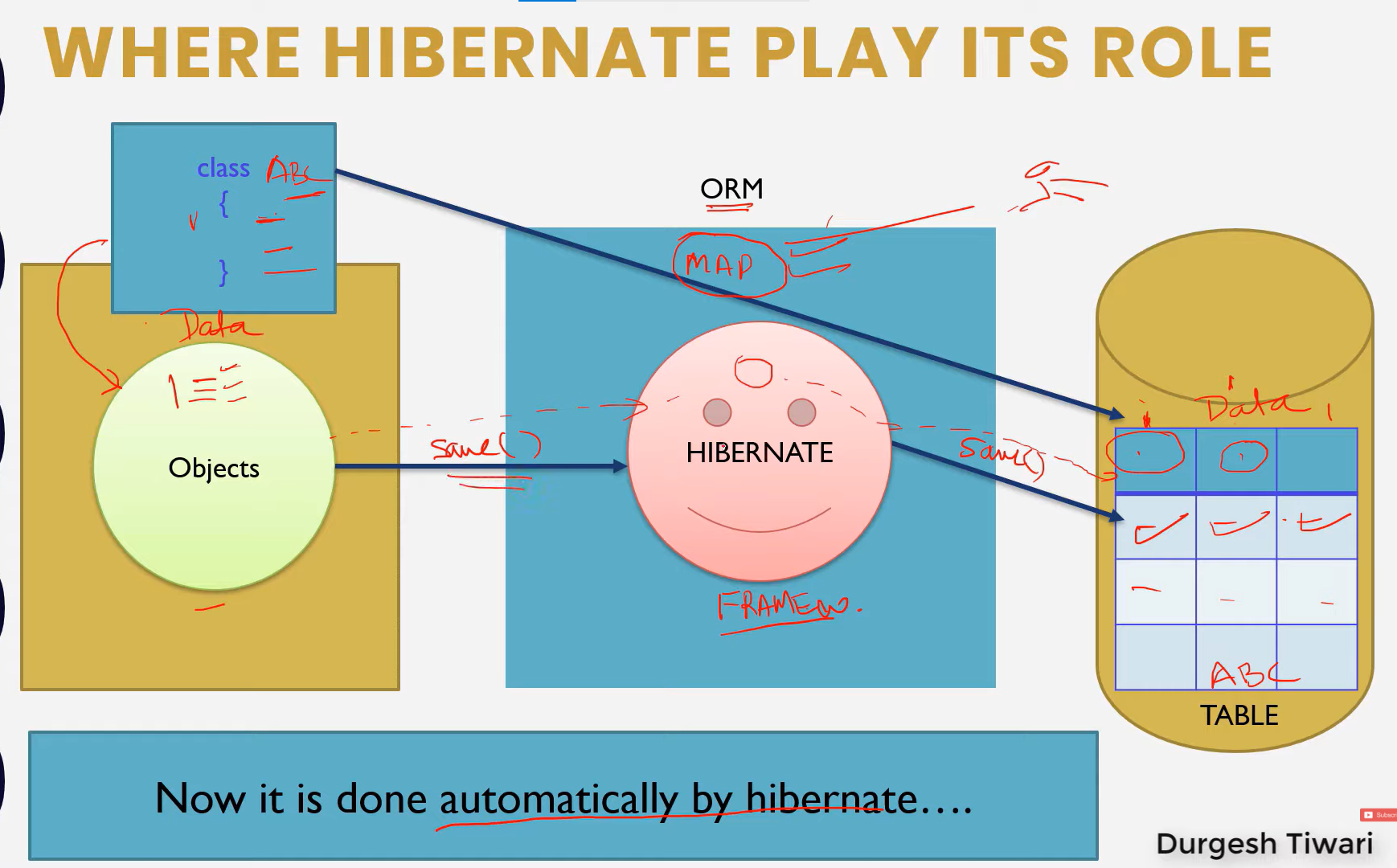


What we do in JDBC



That means we create DAO (Data Access Object) class & write code manually to store data.

Now what we do in Hibernate i.e. only mapping



Now how we can perform mapping.

Internally we map our java class with the respective table and that data members that we create in our class which instance variable which are the fields or columns of that table.

Hibernate internally used JDBC API to perform persistence logic.

Now there are two ways of mapping by using annotations and XML.

**How it works?**

Database

Object

Object

Hibernate

Object

Values

**How to build hibernate.cfg.xml file?**

Step 1: download hibernate dtd configuration file online.

Step 2: Add first tag which is <hibernate-configuration></hibernate-configuration>

Step 3: In <hibernate-configuration></hibernate-configuration> add <session-factory></session-factory>

Step 4: in <session-factory> there is one tag which is <property></property> and here we have to specify the properties of database

Here is the code of hibernate.cfg.xml file

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

<!DOCTYPE hibernate-configuration SYSTEM "../../../hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/myhiber</property>

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

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

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

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

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

</session-factory>

</hibernate-configuration>

**What is Session Factory?**

* Session factory is one type of connection in hibernate which is like connection interface in JDBC there we will get statements , prepareStatement.
* Seam it works like connections but this is factory where we can get sessions.
* It is thread safe object which we build one time for only one project.
* We cant create an object of session factory we will because session factory is an interface and we cant create an object of session factory will get object from SessionFactoryImpl.class which is implement by session factory interface.

**How to build session factory object ?**

**package** com.projectwithmaven;

**import** org.hibernate.SessionFactory;

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

**public** **class** App

{

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

{

System.***out***.println( "Project Started !" );

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

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

}

}

**How to store a student data using model class ?**

Step 1 : create a Student model class

**package** com.projectwithmaven;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

@Entity

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

@Id

**private** **int** id;

@Column(name="name")

**private** String name;

@Column(name="city")

**private** String city;

**public** Student(**int** id, String name, String city) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.city = city;

}

**public** Student() {

**super**();

// **TODO** Auto-generated constructor stub

}

@Override

**public** String toString() {

**return** "Student [id=" + id + ", name=" + name + ", city=" + city + "]";

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

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

**this**.city = city;

}

}

Step 2 : to store data in database In App.java

**package** com.projectwithmaven;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

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

**public** **class** App

{

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

{

System.***out***.println( "Project Started !" );

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

Student st=**new** Student();

st.setId(01);

st.setName("Aatish");

st.setCity("Jalna");

Session session=factory.openSession();

Transaction tx=session.beginTransaction();

session.save(st);

tx.commit();

session.close();

}

}

Step 3 : Map a class in hibernate.cfg.xml file

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

<!DOCTYPE hibernate-configuration SYSTEM "../../../hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/myhiber</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>

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

</session-factory>

</hibernate-configuration>

**Steps to add data in database using hibernate**

* In App.java create a object of configuration call the configure method with the object of configuration build a object for session factory which can hold sessions.

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

OR

Directly you can use one line to this all of the operations

SessionFactory factory = **new** Configuration().configure().buildSessionFactory()

Create a Object of Student Class

Student st=**new** Student();

Call the setters

st.setId(01);

st.setName("Aatish");

st.setCity("Jalna");

* retrieve a session from session factory which is store in factory and open session

Session session=factory.openSession();

* Begin a transaction by using method beginTransaction() provided by session where you can store the value in Transaction variable

Transaction tx=session.beginTransaction();

* Save The session

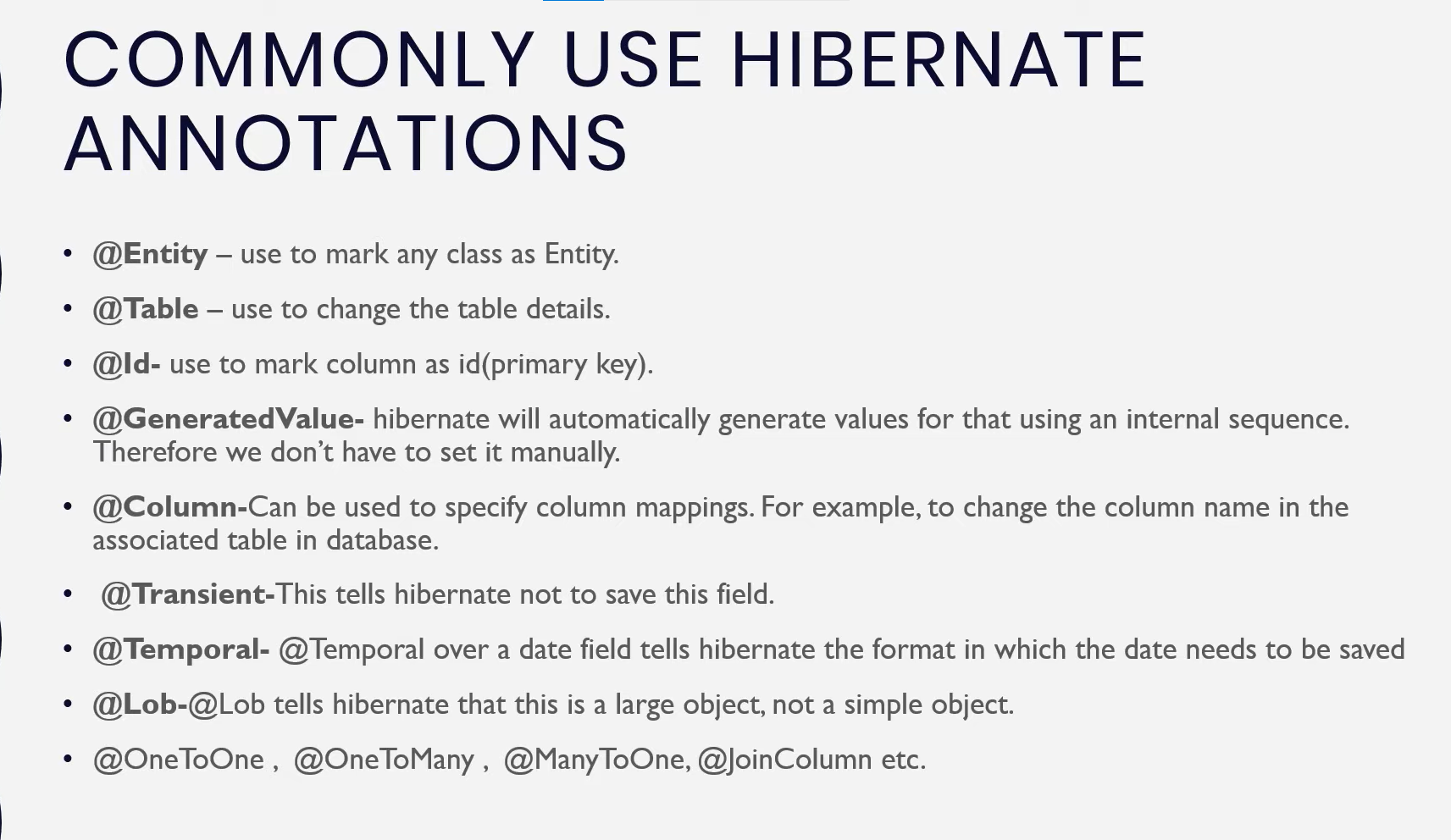
session.save(st);

* Commit Transaction

tx.commit();

Close Session: session.close();

**What the commonly used annotations are in hibernate?**

****

**How To Read an image and store in database?**

//Reading Image

FileInputStream fis=**new** FileInputStream("src/main/java/img.jpeg");

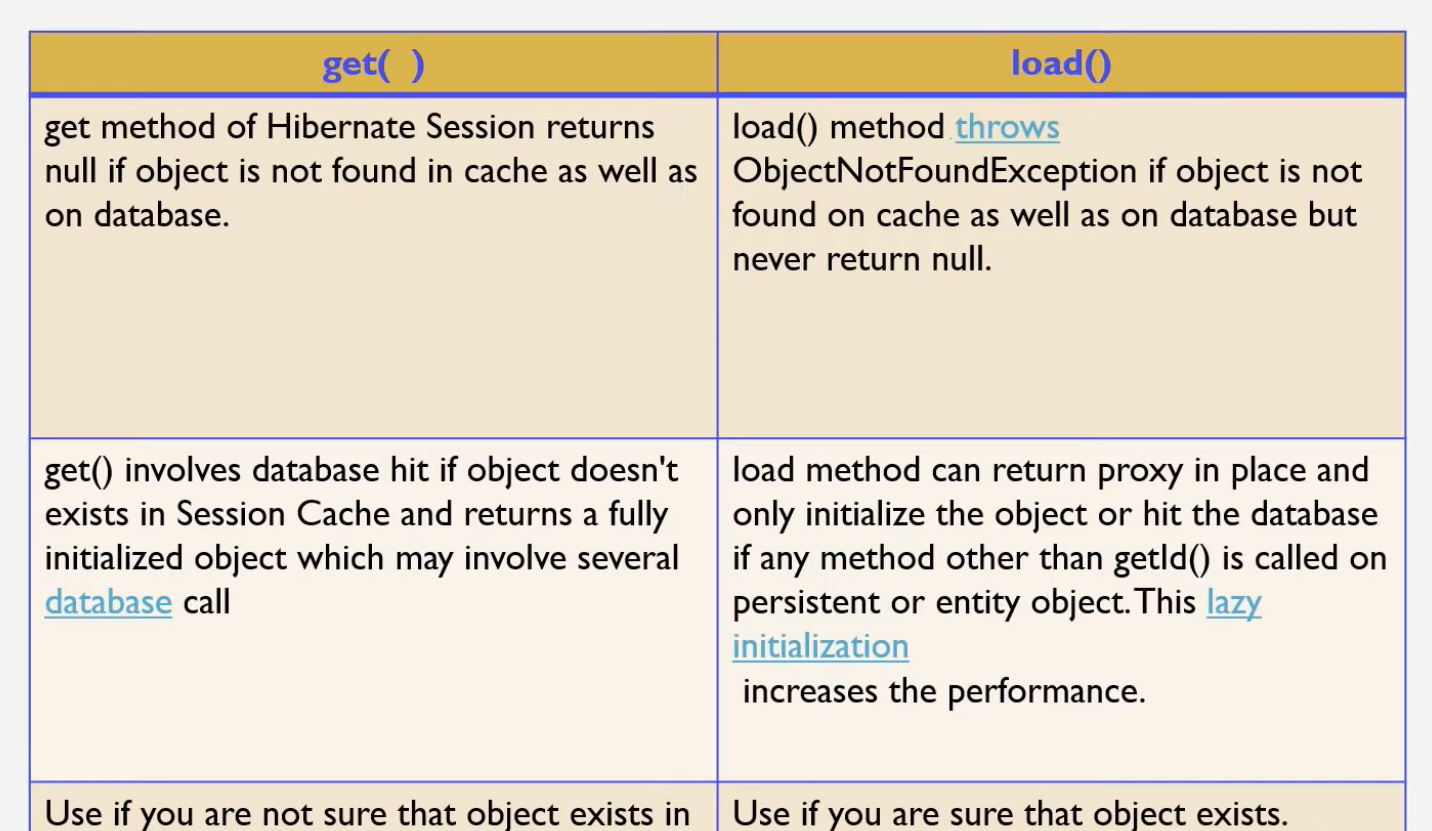
**byte**[] data=**new** **byte**[fis.available()];

fis.read(data);

ad.setImage(data);

**Q. How To Fetch Data from Database?**

* There are two methods which is used to fetch data from database

****

**Eg.**

**package** com.projectwithmaven;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**public** **class** FetchDataUsingGet\_Load {

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

//Building a object of session factory

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory=cfg.buildSessionFactory();

//retrieving session from session factory & opening session

//here you dont want To Transaction obj becuase transaction is only used while you storing data in DB.

Session session= factory.openSession();

//Get Method get the data from DB and store it into session cache if need second time that object then it not going to hit DB it simply searches on session cache if its found then return data from chache otherwise hit DB.

Student st=(Student)session.get(Student.**class**, 2);//here it hit DB directly

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

//Load Method here we are building proxy object because it don’t hit directly to database it hits when its necessary to fetch.

Address ad=session.load(Address.**class**, 2);//here it doesn’t hit DB

System.***out***.println(ad);//here we are fetching data from Address obj now it hit DB

}

}

**What is Embeddable Annotations in Hibernate?**

If we want to embed to class properties with each other without defining as an entity to second class then we can used @Embeddable object.

See in the given example we are creating one class named as certificates without creating a table of certificate table in DB we just embed values of certificate object to Student object.

**package** com.projectwithmaven;

**import** javax.persistence.Embeddable;

@Embeddable

**public** **class** Certificates {

**private** String course;

**private** String duration;

**public** Certificates(String course, String duration) {

**super**();

**this**.course = course;

**this**.duration = duration;

}

**public** String getCourse() {

**return** course;

}

**public** **void** setCourse(String course) {

**this**.course = course;

}

**public** String getDuration() {

**return** duration;

}

**public** **void** setDuration(String duration) {

**this**.duration = duration;

}

**public** Certificates() {

**super**();

// **TODO** Auto-generated constructor stub

}

@Override

**public** String toString() {

**return** "Certificates [course=" + course + ", duration=" + duration + "]";

}

}

//

**package** com.projectwithmaven;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

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

**public** **class** EmbeddableDemo {

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

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory=cfg.buildSessionFactory();

//creating object of student and store values in it

Student st=**new** Student();

st.setName("Aatish");

st.setCity("Jalna");

//creating object of certificates class & store values in it

Certificates certificate=**new** Certificates();

certificate.setCourse("Java Development");

certificate.setDuration("6 Months");

//embedd certifucate object values to student object

st.setCertificate(certificate);

Session session=factory.openSession();

Transaction tx=session.beginTransaction();

session.save(st);

tx.commit();

factory.close();

session.close();

}

}

Finally, we'll use @Embedded annotation to embed a specific type. We have used @Embedded annotation here to denote that these objects will be embedded in our entity. In other words, all of these objects will together be mapped to a single person database table.

The Student class has an certificate field which is marked with the @Embedded annotation. This tells Hibernate that the certificate object is an embeddable object and its properties should be mapped to columns in the same table as the Student class.

**Mappings in Java**

1. **One To One**

One to one represents that a single entity is associated with a single instance of the other entity. An instance of a source entity can be at most mapped to one instance of the target entity.

E.g. we have two entities (Questions & Answers) that mean one questions can have one answers only.

package com.QusAns;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

@Entity

public class Questions {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int questionId;

private String question;

@OneToOne

@JoinColumn(name = "a\_id")

private Answers answerId;

public Questions() {

// TODO Auto-generated constructor stub

}

@Override

public String toString() {

return "Questions [questionId=" + questionId + ", question=" + question + ", answerId=" + answerId + "]";

}

public Questions(int questionId, String question, Answers answerId) {

super();

this.questionId = questionId;

this.question = question;

this.answerId = answerId;

}

public Answers getAnswerId() {

return answerId;

}

public void setAnswerId(Answers answerId) {

this.answerId = answerId;

}

public int getQuestionId() {

return questionId;

}

public void setQuestionId(int questionId) {

this.questionId = questionId;

}

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

}

------------------------------------------------------------------------

package com.QusAns;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity

public class Answers {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int answerId;

private String answer;

@OneToOne(mappedBy = "answerId")// that means answerId is taking care of all the things.

private Questions questionId;

public Answers() {

// TODO Auto-generated constructor stub

}

public Answers(int answerId, String answer, Questions questionId) {

super();

this.answerId = answerId;

this.answer = answer;

this.questionId = questionId;

}

@Override

public String toString() {

return "Answers [answerId=" + answerId + ", answer=" + answer + ", questionId=" + questionId + "]";

}

public Questions getQuestionId() {

return questionId;

}

public void setQuestionId(Questions questionId) {

this.questionId = questionId;

}

public int getAnswerId() {

return answerId;

}

public void setAnswerId(int answerId) {

this.answerId = answerId;

}

public String getAnswer() {

return answer;

}

public void setAnswer(String answer) {

this.answer = answer;

}

}

package com.QusAns;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class App

{

public static void main( String[] args )

{

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

Questions qs = new Questions();

Questions question=new Questions();

question.setQuestion("What is collections ?");

Answers answer=new Answers();

answer.setAnswer("collections is a utility class available in java.utils package");

answer.setQuestionId(question);

question.setAnswerId(answer);

Session session=factory.openSession();

Transaction tx= session.beginTransaction();

session.save(question);

session.save(answer);

tx.commit();

}

}

1. **One To Many & Many To One**

One to Many represents that a single entity is associated with a Multiple instance of the other entity. An instance of a source entity can be at most mapped to one instance of the target entity.

E.g. we have two entities (Questions & Answers) that mean one questions can have multiple answers and multiple answers can have one questions that is many to one mapping.

package com.QusAns;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

import javax.persistence.Table;

@Entity

@Table(name = "Questions\_OTM")

public class Questions {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int questionId;

private String question;

@OneToMany(mappedBy = "question")

private List<Answers> answers;

public Questions() {

// TODO Auto-generated constructor stub

}

public Questions(int questionId, String question, List<Answers> answers) {

super();

this.questionId = questionId;

this.question = question;

this.answers = answers;

}

@Override

public String toString() {

return "Questions [questionId=" + questionId + ", question=" + question + ", answers=" + answers + "]";

}

public int getQuestionId() {

return questionId;

}

public void setQuestionId(int questionId) {

this.questionId = questionId;

}

public String getQuestion() {

return question;

}

public void setQuestion(String question) {

this.question = question;

}

public List<Answers> getAnswers() {

return answers;

}

public void setAnswers(List<Answers> answers) {

this.answers = answers;

}

}

package com.QusAns;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

@Entity

@Table(name = "Answers\_MTO")

public class Answers {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int answerId;

private String answer;

@ManyToOne

@JoinColumn(name = "question\_id")

private Questions question;

public Answers() {

// TODO Auto-generated constructor stub

}

public Answers(int answerId, String answer, Questions question) {

super();

this.answerId = answerId;

this.answer = answer;

this.question = question;

}

@Override

public String toString() {

return "Answers [answerId=" + answerId + ", answer=" + answer + ", question=" + question + "]";

}

public int getAnswerId() {

return answerId;

}

public void setAnswerId(int answerId) {

this.answerId = answerId;

}

public String getAnswer() {

return answer;

}

public void setAnswer(String answer) {

this.answer = answer;

}

public Questions getQuestion() {

return question;

}

public void setQuestion(Questions question) {

this.question = question;

}

}

**package com.QusAns;**

import java.util.ArrayList;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class App

{

public static void main( String[] args )

{

System.out.println( "Project Started !" );

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory=cfg.buildSessionFactory();

Questions question=new Questions();

question.setQuestion("What is Java ? ");

Answers answer1=new Answers();

answer1.setAnswer("Java is a programming Langauge");

answer1.setQuestion(question);

Answers answer2=new Answers();

answer2.setAnswer("Java is Secure, safe & robust");

answer2.setQuestion(question);

Answers answer3=new Answers();

answer3.setAnswer("It is completely based on object oriented Prgramming");

answer3.setQuestion(question);

List<Answers> list=new ArrayList<Answers>();

list.add(answer1);

list.add(answer2);

list.add(answer3);

question.setAnswers(list);

Session session=factory.openSession();

Transaction tx=session.beginTransaction();

session.save(question);

session.save(answer1);

session.save(answer2);

session.save(answer3);

tx.commit();

session.close();

factory.close();

}

}

//fetching data

**package** com.QusAns;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**public** **class** fetchData {

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

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

Session session=factory.openSession();

//for getting question from DB

Questions question=(Questions)session.get(Questions.**class**,1);

System.***out***.println(question.getQuestion());

//for getting answers for particular questions

**for**(Answers asnwer:question.getAnswers())

{

System.***out***.println(asnwer.getAnswer());

}

}

}

1. **Many to Many**

Many-to-Many mapping is usually implemented in database using a Join Table.

E.g. we have twp entities which is Employees & Project in many to many mapping one employee can work on multiple projects and one projects can assign to multiple employees.

**package** com.MTM;

**import** java.util.List;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

@Entity

**public** **class** Employee {

@Id

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

**private** **int** eid;

@Column(name="employee\_name")

**private** String name;

@ManyToMany

@JoinTable(name="emp\_pro", joinColumns = {@JoinColumn(name="eid")},inverseJoinColumns = {@JoinColumn(name="pid")})

**private** List<Project> project;

**public** Employee() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Employee(**int** eid, String name, List<Project> project) {

**super**();

**this**.eid = eid;

**this**.name = name;

**this**.project = project;

}

@Override

**public** String toString() {

**return** "Employee [eid=" + eid + ", name=" + name + ", project=" + project + "]";

}

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** List<Project> getProject() {

**return** project;

}

**public** **void** setProject(List<Project> project) {

**this**.project = project;

}

}

**package** com.MTM;

**import** java.util.List;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.ManyToMany;

@Entity

**public** **class** Project {

@Id

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

**private** **int** pid;

@Column(name="project\_name")

**private** String pname;

@ManyToMany(mappedBy = "project")

**private** List<Employee> employee;

**public** Project() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Project(**int** pid, String pname, List<Employee> employee) {

**super**();

**this**.pid = pid;

**this**.pname = pname;

**this**.employee = employee;

}

@Override

**public** String toString() {

**return** "Project [pid=" + pid + ", pname=" + pname + ", employee=" + employee + "]";

}

**public** **int** getPid() {

**return** pid;

}

**public** **void** setPid(**int** pid) {

**this**.pid = pid;

}

**public** String getPname() {

**return** pname;

}

**public** **void** setPname(String pname) {

**this**.pname = pname;

}

**public** List<Employee> getEmployee() {

**return** employee;

}

**public** **void** setEmployee(List<Employee> employee) {

**this**.employee = employee;

}

}

**package** com.MTM;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

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

**public** **class** App

{

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

{

System.***out***.println( "Many To Many Mapping Started!" );

Configuration cfg=**new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory=cfg.buildSessionFactory();

Employee emp1=**new** Employee();

Employee emp2=**new** Employee();

emp1.setName("Aatish Pawar");

emp2.setName("Anish Jha");

Project p1=**new** Project();

Project p2=**new** Project();

p1.setPname("Library Management System");

p2.setPname("CHATBOT");

List<Employee> employees=**new** ArrayList<Employee>();

List<Project> projects=**new** ArrayList<Project>();

employees.add(emp1);

employees.add(emp2);

projects.add(p1);

projects.add(p2);

emp1.setProject(projects);

p1.setEmployee(employees);

Session session=factory.openSession();

Transaction tx=session.beginTransaction();

session.save(emp1);

session.save(emp2);

session.save(p1);

session.save(p2);

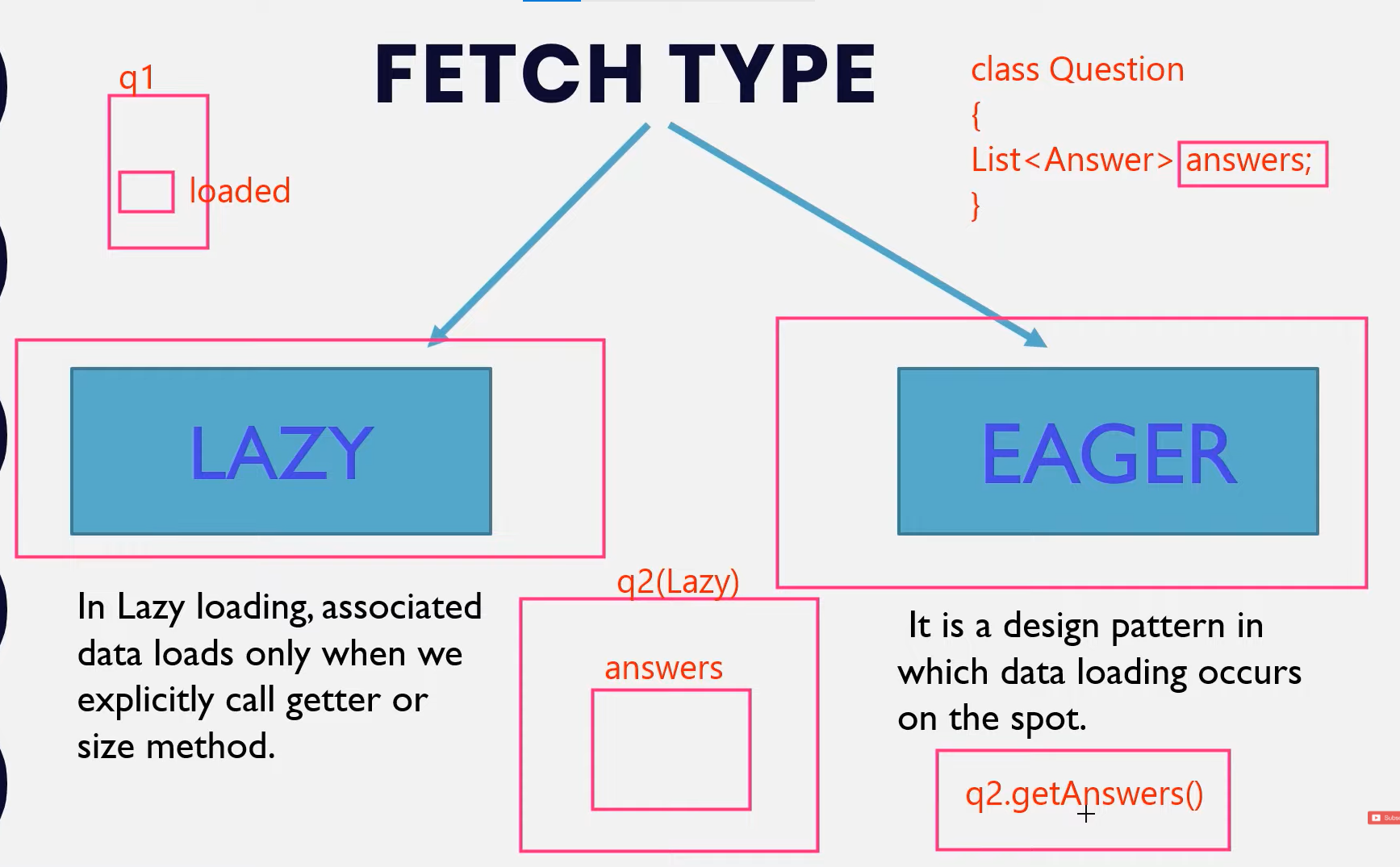
tx.commit();

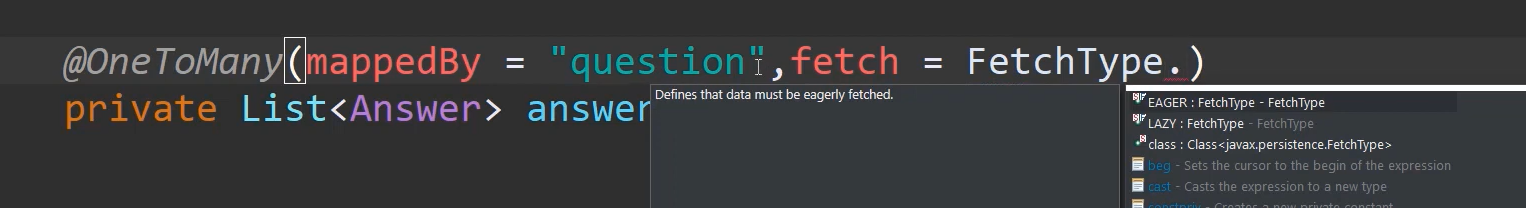
factory.close();

}

}

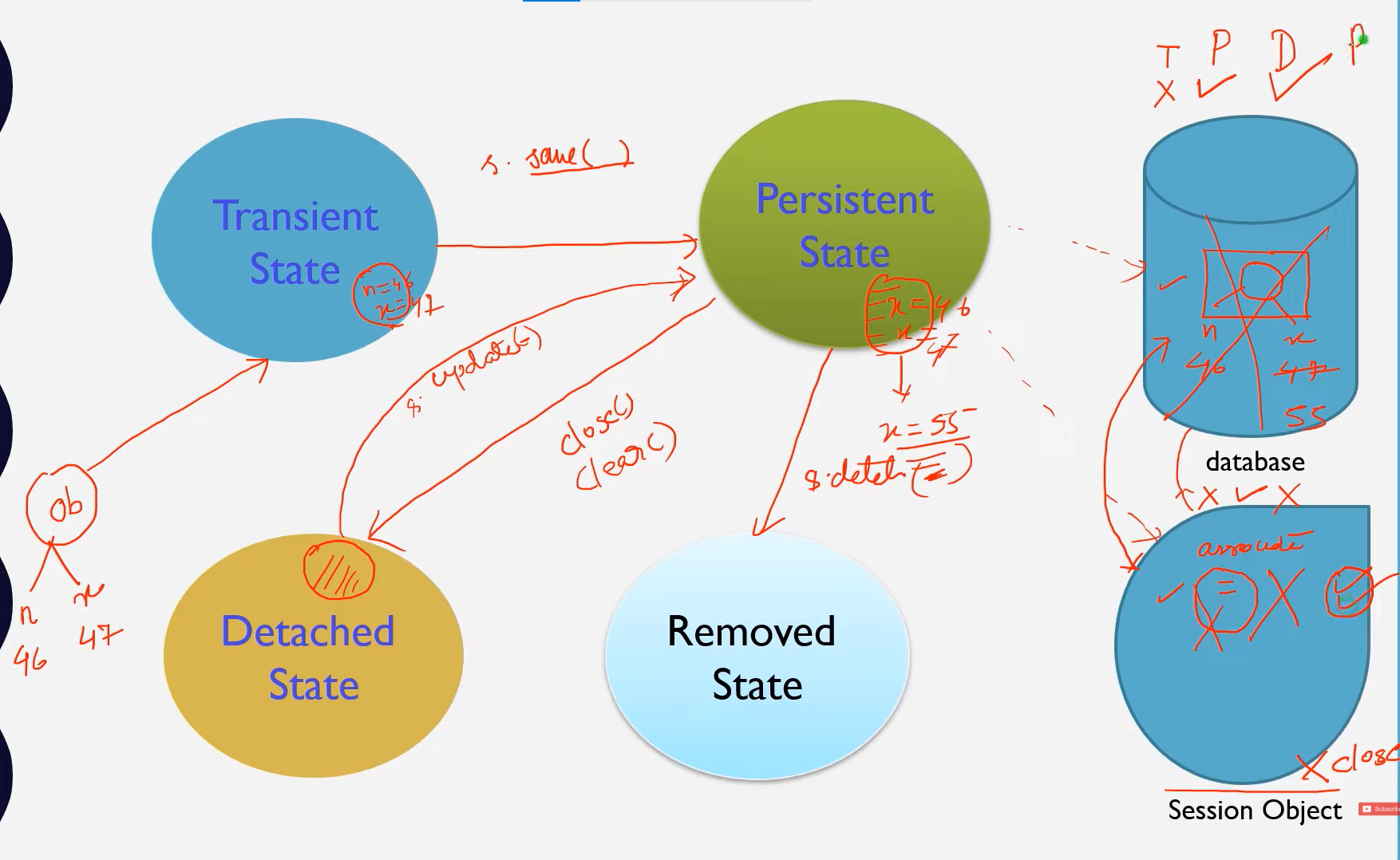
* **Fetch Technique in Hibernate**

****

****

**By default lazy loading is applied if we have set fetch type has eager then we set in annotation it does not related to get and load method it’s another topic that we can use on get method.**

* **Hibernate/Persistence Life Cycle States**

****

1. **Transient :**

When we create a new object and we set the values using setters’ method then this object is in transient state.

That means it does not associated with DB & Session

1. **Persistent :**

When we called save() or persist() & saveorupdate() methods that means we are passing that object to session & it stores values in Database.

That means session and database both has a values of object.

If we changed value of an object while its in persistent state it synchronized with database that means it automatically save the values we don’t explicitly call save or update method.

1. **Detached State :**

If we close the session or clear the session then object goes in detached state

That means we remove the object from session but not from the database but that object cant associated with session

1. **Removed state :**

If we called delete method from persistent state it deletes the values which is stored in database but that object associated with session

**package** com.states;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

@Entity

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

@Id

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

**private** **int** sid;

@Column(name = "Student\_Name")

**private** String sname;

@Column(name = "Student\_City")

**private** String scity;

**private** Certificate certi;

**public** Student() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Student(**int** sid, String sname, String scity, Certificate certi) {

**super**();

**this**.sid = sid;

**this**.sname = sname;

**this**.scity = scity;

**this**.certi = certi;

}

@Override

**public** String toString() {

**return** "Student [sid=" + sid + ", sname=" + sname + ", scity=" + scity + ", certi=" + certi + "]";

}

**public** **int** getSid() {

**return** sid;

}

**public** **void** setSid(**int** sid) {

**this**.sid = sid;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

**public** String getScity() {

**return** scity;

}

**public** **void** setScity(String scity) {

**this**.scity = scity;

}

**public** Certificate getCerti() {

**return** certi;

}

**public** **void** setCerti(Certificate certi) {

**this**.certi = certi;

}

}

// Certificate class which is embeddable

**package** com.states;

**import** javax.persistence.Embeddable;

@Embeddable

**public** **class** Certificate {

**private** String course;

**private** String duration;

**public** Certificate() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** String getCourse() {

**return** course;

}

**public** **void** setCourse(String course) {

**this**.course = course;

}

**public** String getDuration() {

**return** duration;

}

**public** **void** setDuration(String duration) {

**this**.duration = duration;

}

**public** Certificate(String course, String duration) {

**super**();

**this**.course = course;

**this**.duration = duration;

}

@Override

**public** String toString() {

**return** "Certificate [course=" + course + ", duration=" + duration + "]";

}

}

// Main Class : States.java

**package** com.states;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

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

**public** **class** States

{

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

{

System.***out***.println( "Project States Started !" );

//buliding session factory

SessionFactory factory=**new** Configuration().configure().buildSessionFactory();

Student student=**new** Student();

student.setSname("Anish Jha");

student.setScity("Darbhanga");

student.setCerti(**new** Certificate("Java","6 Months"));

//Student : Is in Transient State because we set the values now

Session session=factory.openSession();

Transaction tx=session.beginTransaction();

session.save(student);

//Student : is in Persistent State because it now associated with Session and Database

//if we change value here it will affect on Db lets see

student.setScity("Bangalore");

//here "Darbhanga " value change to "Bangalore"

//now if we see database it fire update command to change values

tx.commit();

session.close();

//Student: Now in a detached State because we close the session not it not associated with DB

student.setSname("Aatish Pawar");

//it will print session values

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

System.***out***.println("Stated Ended !");

}

}