Behind the scence, the Hibernate methods automatically create SQL query

HIBERNATE

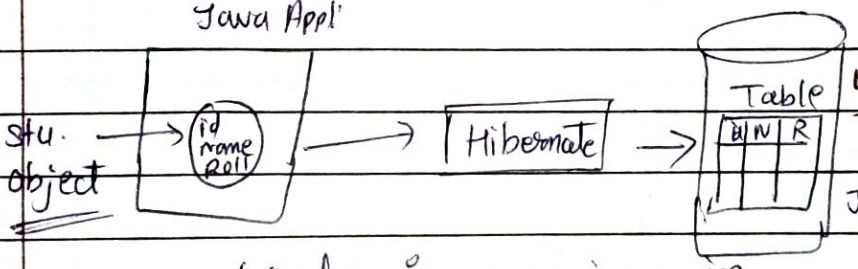
PAGE NO.:

Hibernate :-

What :-

- It is a framework of java, simplify the development with database of java application to interact with the database.

- It is open-source and light-weight [source code open to all]
- It is a ORM (object Relational Mapping) Tool.



what Hibernate does :-
it directly map the JAVA object with the database Table

- object is mapping in Relation(Table) with the help of Hibernate

In Hibernate :-

Entity class (model class)

- It make your application loosely Coupled.
- It is developed by Jevin king in 2001.

Cache hit → mil jana.
Cache miss → nhi milna
(Frequently using data stored in cache memory)

PAGE NO.:

"JDBC"

1. You must have a knowledge of SQL.
2. # make your application database depend.

(Dialect) (Hibernate)
mysql. oracle.
Dialect ko bata do ke mysql database ~

3. Lots of boiler plate code.
4. Mannual Transcation Handling.
5. Mannual Exception handling, lots of Exce.. handl... code.
6. No Caching Support

Hibernate

The Solution of all those problem, is Hibernate

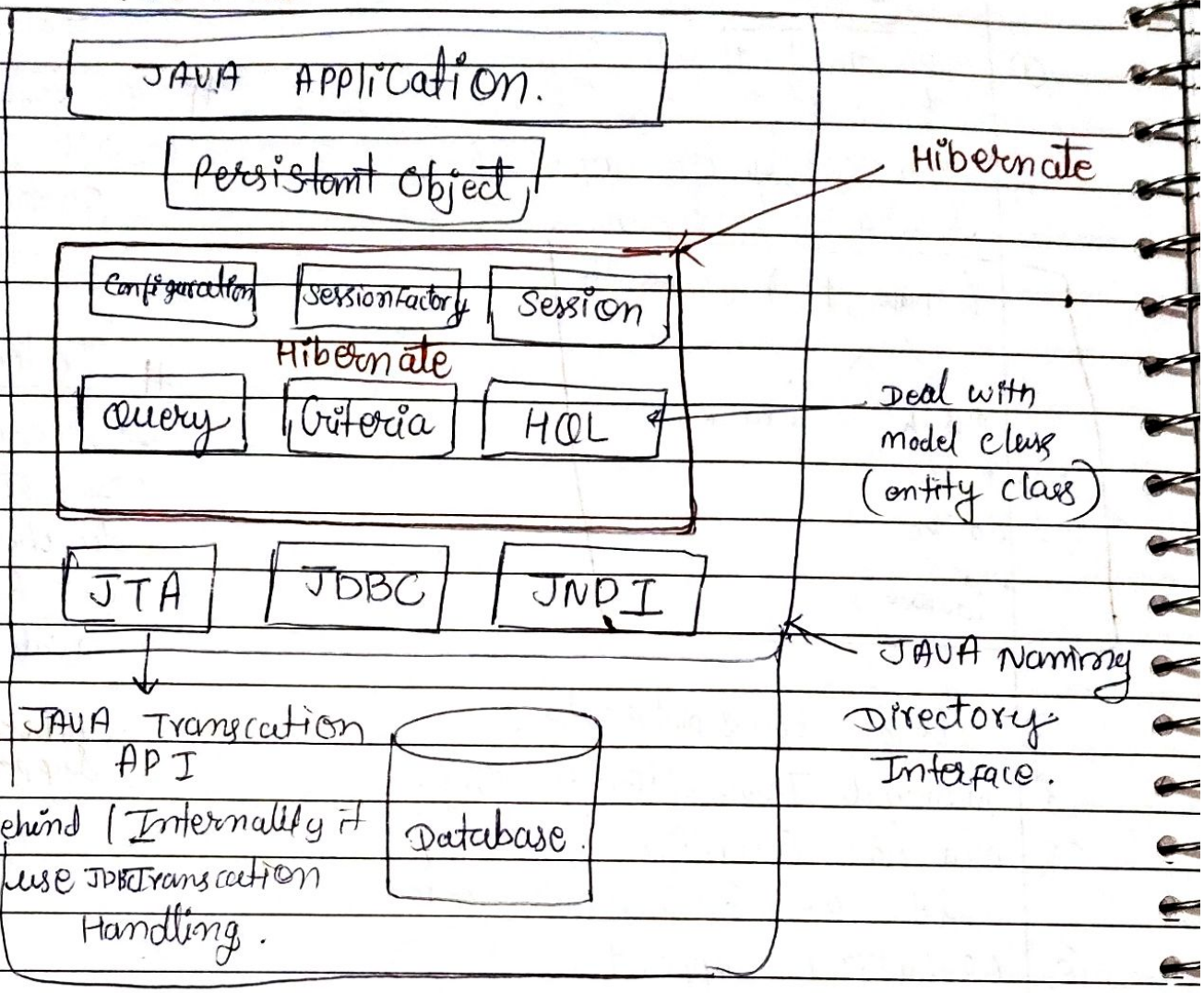
Independency, Just change the Dialect name of database, and get know to Dialect

Hibernate 6 Supports JPA

SessionFactory :- Singleton object

PAGE NO.:

Hibernate - Architecture



① **Configuration object** :- It is the first Hibernate object,

- only once creation during initialization
- represents Configuration,
- Property file required by Hibernate

Database Connection :- { class Mapping setup } :- It create Connection between the java classes and database Tables.

② **Session factory** :- It is responsible to provide session object.

one database → one session factory

session factory objects configure Hibernate for the application using the supplied Configuration file and allows for session object to be instantiated

③ **Session** :- jitni bar bhi database operation perform karna, utni bar session object create karenge.

Transcation object :- a unit of work with database

POM.XML → Project object model.

PAGE NO.:

- Persistent object :- want to save the object, Saved and retrieved through a session object.

A session is used to get a physical connection with a database.

④ Query object :- if use SQL or HQL
to execute HQL, we use Query object
if we deal with HQL, then use Query object
used to fetch record.

⑤ Criteria object :- used to execute & create object oriented criteria queries,

Configuration work :-

① file :-

Hibernate.Cfg.XML (configuration)
(xtensible markup langa)
(Database related code information)

Tag :- in file

```
< hibernate-configuration >
  < session-factory >
    < property name = "hibernate.Connection.driver-class" >
        Com.mysql.cj.Jdbc.Driver </property>
    < property name = "hibernate.connection.url" > jdbc:mysql://
        localhost:3306/hdb </property>
    < property name = "hibernate.connection.username" > root </property>
    < property name = "hibernate.connection.password" > </property>
    < property name = "hibernate.connection.hbm2ddl.auto" >
        update </property>
```

: create
: update
validate

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

<property name = "show_sql"> true </property>

<property name = "format_sql"> true </property>

</session factory>

</hibernate_configuration>

"Resource folder setup"

Project → Build Path → Configure build Path, Right click.

Source → Add folder → main select (create new folder)

Name → Resources.

ok, ok, Apply, Apply close.

Resource folder → others → XML → Next (Hibernate.Cfg.XML)

Annotation

@Entity

@Table (kis nam se Table banana. Hai vo batasakte)

@Column (management Constraints of column name etc)

@Id (primarey key banana hai Id ko)

@JoinTable (third Table se Related Infoormation dena)

@Transient (when apn ko kol data, database me save nhi karana hai)

@ 1:1

@ 1:M

@ M:1

@ M:M

@GeneratedValue (auto Incrament)

Application Shutdown hone par.
Session factory Close karenge
SF.close();

JRE system library [JA... SE-11]

PAGE NO.:

Configuration Class
↓
Method.
Configure(); // Read Hibernate.cfg.xml file.
(
buildSessionFactory(); It build the sessionFactory

Session :—
OpenSession();

Transcation :—
beingTranscation();

⇒ fetchUserById()
(User user = session.get(user.class, id);

try(SessionFactory sessionFactory = HibernateUtil.getFactory();
Session Session = sessionFactory.openSession();){

}

Update() :-
// Load ---> update ---> Save
Delete() :-
load ---> delete --->
Dependency.
Maven repository. → Spring security crypto → version 5(5.7.1)

original data → Ciphertext (encrepted encrypted data)
[Plan text

encrypted password with 16character / salt key
2.....

Password

@ PrePersist
@ PreUpdate
@ Converter (To Convert the data into encrypted way)

To encryption of data, there are several algorithms -

JAVAX.Crypto ⇒ Package

Cipher ⇒ class.

Algorithm = AES;

Cipher cipher = Cipher.getInstance(ALGORITHM);

↓ method in Cipher class

↓ Algorithm name

Cipher se encryption or decryption data karsakte

SecretKeySpec (class)

↑ isme hum denge (certificate)

HibernateUtil ⇒

1) SessionFactory sessionFactory = HibernateUtil.getFactory();

2) SessionFactory sessionFactory = sessionFactory.openSession();

1 line :- Ye line hibernate ka connection factory ready karti hai jisse hum aage sessions (database connections) bana sakein

2 line :- Ye line actual me database se ek connection open karti hai taaki hum query execute kar sakein.

// No Such Padding Exception //

PAGE NO.:

DataConvertor [code to encrypt the data]

```
Public class DataConvertor implements AttributeConverter
    < string , string >{
private static final string ALGORITHM = " AES";
private static final string KEY = "MySuperSecretKey";
@ override
Public String ConvertToDatabaseColumn ( string attribute){
 try {
     IF (attribute == null)
         return null ;
     Cipher cipher = Cipher.getInstance (ALGORITHM);
 SecretkeySpec keySpecification = new SecretKeySpec (
                 key KEY . getBytes (), ALGORITHM);
 byte [] encryptedBytes = Cipher. doFinal ( attribute -getBytes ()).
 String encryptedString = Base64. getEncoder ( ). encodeToString
                                              ( encryptedBytes);
System. out. println (encryptedString);
return encryptedString ;
}
catch ( Exception e ){
 e. printStackTrace ();
 throw new RuntimeException ( e.getMessage ());
  }
 }
```

"Decoded the data"

```
public static String convertToEntityAttribute(String dbData)
{
try {
    if(dbData == null)
        return null;
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    SecretKeySpec keySpecification = new SecretKeySpec
                                    (key.getBytes(), ALGORITHM);
    cipher.init(cipher.DECRYPT_MODE, keySpecification);
    byte[] decodedBytes = Base64.getDecoder().decode
                                    (dbData);
    byte[] decryptedBytes = cipher.doFinal(decodedBytes);

    return new String(decryptedBytes);
    }
    catch (Exception e){
      e.printStackTrace();
      throw new RuntimeException(e.getMessage());
    }
    }.
```

connection ~~String~~ Stream.com

PAGE NO.:

"PasswordHashUTIL"

⇒ **working :-**

Package :- import org.springframework.security.crypto.bcrypt.BCrypt;

[In this we import BCrypt Class of Spring Security library
BCrypt ek password hashing algorithm hai —
ye passwords ko encrypt kar deta hai taaki wo
secure ho jaaye].

```
Public Static String hashPassword(String password){
String hashedPassword = BCrypt.haspw(plainPassword, BCrypt.gensalt(12));
System.out.println(hashedPassword);
return hashedPassword;
}
```

- plainPassword :— user's normal Password.
- BCrypt.haspw(plainPassword, BCrypt.gensalt(12))
  - hashpw() :— This method do password Hashing (encryption)
  - gensalt(12) :— it generate random salt, and this salt increase the security.
  - (12) ⇒ cost factor, jitna bda number, utna zyada secure (lekin thoda slow)

Example :— "admin123" plainPassword

↓

$2a$12$DgL3cYrZbMTQ2zUQqDJMUGFMgkZC4oht1SG1/Jm

16 character salt key — The 32 chars hash password

CipherText (encrypted data)

ye string har baar alag hoti hai kyuki salt random hota hai

important function	Purpose
① BCrypt.hashpw(), BCrypt.gensalt()	→ Password ko Hash Karna
② BCrypt.checkpw()	→ Entered password ko stored hash se verify karna

⇒ These are the static methods of BCrypt class.
~~Hash~~ hashpw(), gensalt(), checkpw()

Fetching Strategy :-
used when association implementation

Difference between get() and load() method in Hibernate.

Feature	get()	load()
(1) When data is fetched	Immediately hits the database and fetches the data as soon as it's called (eager loading.)	Returns a proxy object first - data is fetched only when needed (lazy loading)
(2) If record not found.	Returns null	Throws ObjectNotFound Exception. when you try to access the object and it doesn't exist.
(3) Use case	when you are not sure if the record exists (safe to handle null).	when you are sure that the record exists in the database.
(4) Performance	Slightly slower because it always hits the database immediately.	more efficient in some cases because it delays the database call until necessary.
(5) Proxy behaviour	Doesn't use proxy - directly returns the actual object.	Returns a proxy (dummy) object that represents the entity.

Example :-

① `Student S1 = session.get(Student.Class, 1);`

// --> Immediately fetches student with ID 1 from DB.

// ---> Returns null if not found.

② `Student S2 = session.load(Student.Class, 1);`

// --> Returns a proxy object first

// ---> only when you call S2.getName(), hibernate fetches data from DB

// -> If not found, throws Exception

Hibernate Object States :-

Four States

① **Transient State** :- the object just created using new keyword. It is not associated with any Hibernate session. It does not exist in the database yet, Hibernate doesn't track any changes made to it.

Ex:-
```
Student S = new Student (); // new object → transient.
S.setId(1);
S.setName("Neha");
```

② **Persistent State** :- The object is associated with a hibernate session. Hibernate tracks changes to this object - any modification will be automatically sync with the database when the session is flushed or closed. The object represents a record in the database.

Ex:-
```
Session session = factory.openSession();
Transcation tx = session.beginTranscation();
session.save(S); // now 'S' becomes persistent
S.setName("Priya");
```
⇒ If we change S.setName("Priya") before commit, Hibernate will update it automatically in the DB.
```
tx.commit();
```

③ **Detached State** :- • After the session is closed, the persistent object • becomes detached. It still has a database identity, but it no longer tracked by Hibernate.

whenwe want to fetch all the records, use HQL

PAGE NO.:

Ex:-
```
session.close(); // session is closed
s.setName("mita"); // change won't be reflected in DB.
```

(4) Remove (Deleted) State :- The object is marked for deletion using session.delete(). It will be deleted from database after the transaction commits. After deletion, it becomes transient again

Ex:- session.delete(s); // object is now removed state.

HQL (Hibernate query language)

Interview Q.

Q. HQL vs SQL

	HQL	SQL
①	Hibernate query lang.	① Structure query lang.
②	Database independent	② Database dependent.
③	work with entity class / Java class/ object rather than table	③ work with table / row / column
④	Select * from User ↓ class.Name	Select * from user ↓ Table
⑤	To fetch the data (use → session.get() - case) session.load() → Id basis par, fetch hoga. session.remove() session.update()	data

SessionFactory sessionFactory = HibernateUtil.getFactory();
try(Session session = HibernateUtil.getFactory().openSession();)

PAGE NO.:

{ TypedQuery<User> query = session.createQuery("from User", User.class);
List<User> userList = query.getResultList();
for(User user : userlist)
System.out.println(user.getId()+" "+
user.getName()+" "+
user.getEmail());
}

↑

To fetch all the record from User entity (Table.

// Hql query, when fetchById()=>

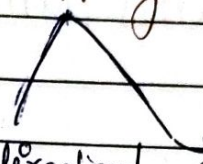
Public static void fetchById(){
SessionFactory sessionFactory = HibernateUtil.getFactory();
try(
TypedQuery<User> query = session.createQuery("from
User where Id =: UserId",
User.Class);
query.setParameter("Id", UserId);
User user = query.getSingleResult();
System.out.println("Login Success");

PAGE NO.:

* Association / Mapping

1. One To One
2. One To many
3. Many To One
4. Many To Many

Mapping



Annotation.

One To one (@ oneTo one)
one To Many (@ one To many
Many To one @ Many To one
Many To Many. @ Many To many

} @ JoinColumn
@ Join Table
mapped by

1. One To one

E_1 ——— E_2

owing side ~~Foreignkey~~ Hogi.
Inverse-side → mappedby Hega.

⇒ one record of E_1 is associated with one record of E_2

Entity class

⇒ Agr jis Entity me foreign ki hogi ~~honi~~ chahiye, usse opposite wali Entity me mapped by lagaye.

```
@ Entity
class user{
Private int id;
Private String name;
@ one To one (mappedby = "user")
Private Passport passport;
}
```

Bi-directional Hai
Kyki user se password nikal lenge, or Password se user.

```
@ Entity
class passport{
Private int id;
private String passport;
@ oneTo one
Private User user;
}
```

In hibernate, first level cache is enable

DML ke sath transaction
~~insert, update, delete.~~

PAGE NO.:

owing-side

Jidhar foreign key likha hai

Inverse-side

Jidhar mappedby likha hai

@JoinColumn(name = "passport_id") (it represents ~~enti~~ owing side)

@OneToOne(mappedby = "user")

[Cascade = CascadeTYPE. ALL]

related → If i delete the data from parent entity, so the child entity data also ~~of~~ delete
→ (ONE TO ONE),

~~fetch~~ fetch technique → EAGER (By default)

~~CascadeTYPE. All~~ [~~Owing- Side~~] X
~~fetchTYPE. lazy~~ →

CascadeTYPE- → (Inverse-side)
fetch TYPE (foreign-key wali entity me)

Default fetching TYPE

Relationship TYPE	Default fetch TYPE
① @ oneToone	EAGER
② @ ManyToone	EAGER
③ @ OneToMany	LAZY
④ @ many ToMany	LAZY

PAGE NO.:

to Supporting table

@JoinTable(name = "student_course", joinColumns = @JoinColumn(name = "course_id"), inverseJoinColumns = @joinColumn (name = "student_id"))

@Entity
@Table(name = "courses")
Public class Course{
@Id
@GeneratedValue(strategy = GenerationTYPE.IDENTITY)
private int id;

@Column(name = "Course_name")
private String courseName;

@ManyToMany

private List<student> studentlist;

Agar Id primary key hai toh,
get ( ) method se direct call kar lenge.

meta - NAFN

PAGE NO.:

Merge() :- It inserts the record, when the table is null, and update the dat record if it is exists.

↓

[IF not that detached problem arrive]

(different Session) → Merge()

(Same Session) → persist()

Jakarta Persistance API

JPA :- To make Applications, ORM independent

↓

Interface (certain rules & regulation) Top vendor → Hibernate

ORM vendors provide JPA Implementation

Hibernate		JPA
SessionFactory	→	EntityManagerFactory
Session	→	EntityManager
Transaction	→	EntityTransaction

one to many

one to one

Query language ⇒ JPAQL

DIF

JDBC API / JPA / Hibernate

Sample persistance xml

JPA (Java Persistence API)

PAGE NO.:

JPA (Java Persistence API) is a java Application Specification that makes it easier to work with relational databases. It lets you map java classes (Entities) to database tables and manage data using simple APIs instead of writing complex SQL. In short, JPA acts as a bridge between Java objects and database records.

- It make Application ORM independent.
  Top vendor Hibernate (ORM vendors provide JPA implementation)
- JPA is not a framework or tool. It is just set of rules (interfaces).

Key features of JPA in java :—

① **Data Integrity** :— Ensures the data is accurate and consistent across the database.

② **Allows Direct mapping** :— Maps java classes and fields directly to database tables & columns.

③ **portable** :— works across different databases without changing the code.

④ **improves productivity** :— Reduces boilerplate code and simplifies database operations

⑤ **Easy to Implement** :— Provides a standard and simple API for CRUD operations & queries.

PAGE NO.:

JPA	Hibernate
① JAVA is described in javax.persistence Package.	① Hibernate is described in org.hibernate package
② It describes the handling of relational data in java Application	② Hibernate is an object-Relational mapping (ORM) tol that is used to save java objects in the relational database system.
③ It is not an implementation. It is only a java specification (Interface).	③ Hibernate is an implementation of JPA. Hence, the commen standard which is given by JPA is followed by Hibernate.
④ It is a standard API that permits to perform database operation.	④ It is used in mapping java data types with SQL data types and database tables.
⑤ It uses (JPQL) java persistence query language to execute database operations.	⑤ it uses HQL (Hibernate query language to execute database operation
⑥ To interconnect with the entity manager factory for the persistence unit, it uses the EntityManaggerFactory Interface, Thus, it gives entityManager	⑥ To create Session instances, it uses the SessionFactory interface.

PAGE NO.:

Steps of JPA Implementation :-

Step :- ① :- Add required Dependencies (Hibernate + JPA + JDBC)

Step :- 2 :- Create persistence.xml

Inside
src/main/resource/META-INF/persistence.xml

<persistence></persistence>

Step :- 3 :- Create an Entity Class.

```
import jakarta.persistence.*;
@Entity
public class Student{
    @Id
    private int id;
    private String name;
    private String city;
    // getter, setters
}
```

@Entity ⇒ maps class to table
@Id ⇒ primary key.

Step :- 4 :- Create EntityManagerFactory

```
EntityManggerFactory Factory = Persistence.CreateEntityManagerFactory("my-persistence-UNIT");
EntityManager manager = factory.CreateEntityManager();
```

Step:-5 Perform CRUD using EntityManager()

INSERT

```
manager.getTransaction().begin();
Student S = new Student();
S.setId(1);
S.setName("Neha");
S.setCity("Indore");
manager.persist(S);
Transaction.commit();
```

DELETE →

```
manager.getTransaction().begin();
Student S = manger.find
        (Student.class, 1);
manger.Remove(S);
transaction.commit();
```

fetch

```
Student S = manager.
        find(Student.class, 1);
so.p(S.getName());
```

Update →

```
manager.getTransaction.
  begin();
Student S = manager.find
        (Student.class, 1);
S.setCity("Bhopal");
transaction.commit();
```

PAGE NO.:

Cache management

① EntityManager level-1 cache :- Enable By default

frequently operation perform karna hota hai tab direct cache se lekar aa jate hai data.

Manager level par

② level-2 cache EntityManagerFactory level par Hota.

Need two dependency Ehcache, ~~normal~~ hibernate-jcache.

provider JCacheRegionFactory of second level cache

iska provider hai (Ehcache) factory level par.

Second-level-Caching.

<property name="cache.use_second_level_cache"
value="true" />

<properety name="cache.region.factory_class" value="org.hibernate.cache.jcache.." />

<properety name="hibernate.javax.cache.provider"
value"---" />

for Second-level-Caching :-

@ Cacheable

@ Cache(usage=CacheConcurrencyStrategy.READ_ONLY)

PAGE NO.:

Composite key

StudentCourse

student_id	student_Name	course_id	course_name
100	Afeel	200	Java fullstack
100	Afeel	201	MERN
101	Ankita	200	Java
101	Ankita	201	MERN.

Domain → unity IP address ← default port Number of Apache Tomcat

HHps:// localhost :8084 / testapp ← Application name

↑ represents your machine.

PAGE NO.:

JSP / Servlet

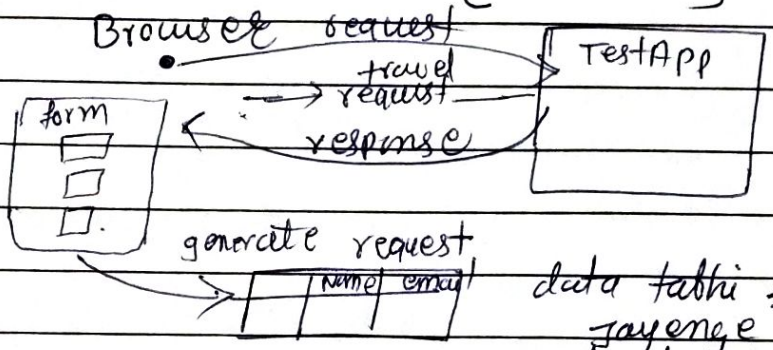
Web-Application — collection of web-pages.

browser ← Client (who send the request)

Server (software) जो किसी न किसी machine पर run होता.

"Apache Tomcat" [web-server]

Request object ke andar data encapsulate karenge.



data tabhi server par jayenge jab apan input field ko name attribute assign karenge like name email

Servlet program :- It is Java program

a program which process the dynamic data. request and give response to the client,

It process the dynamic data.

Server side execution of Java Application.