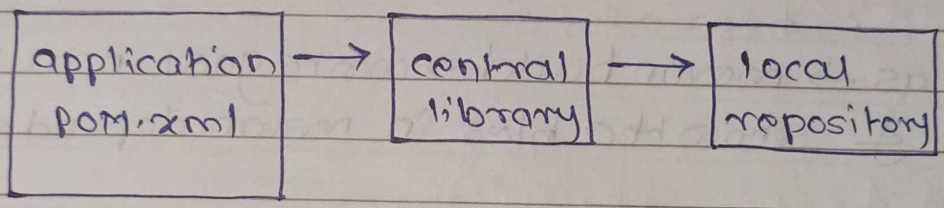
Tuesday 09-01-24

Web Based Java Programming

* POM.xml
(project object model)

```
<dependancies>
  <dependency>
      <affected id> (<artifact id>)
      <group id>
  </dependency>
</dependancies>
```



<packaging></packaging> jar packaging
<packaging>war</packaging> war packaging

src
- main/java
- main/resources
- [illegible]/test/resources
- [illegible]/webapp
- test/java

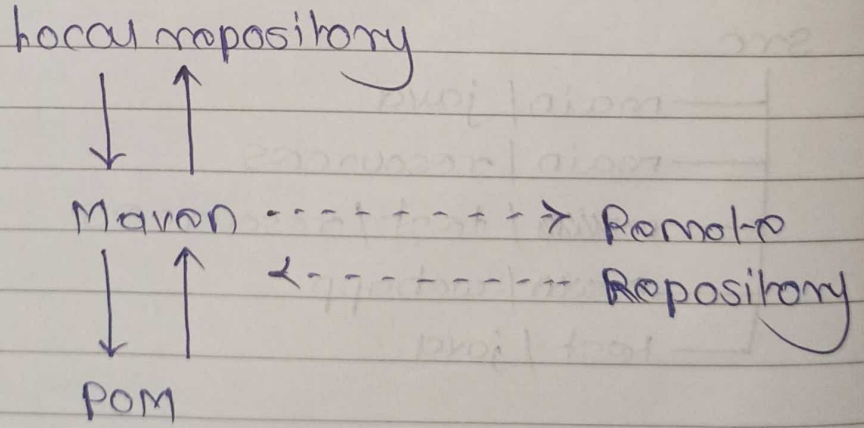
1) new → maven project → create → next → give groupid → artifact id → description → select package jar & create.

2) update pom.xml file & add dependancies

3) window → preferences → maven → user settings & see the location of file

4) run as → maven build → write goal install → apply → run & show the files in maven dependency folder.

at the time of testing
at the time of development
} add scope in the dependency file.



ORM : Object Relation Mapping

* Hybernet

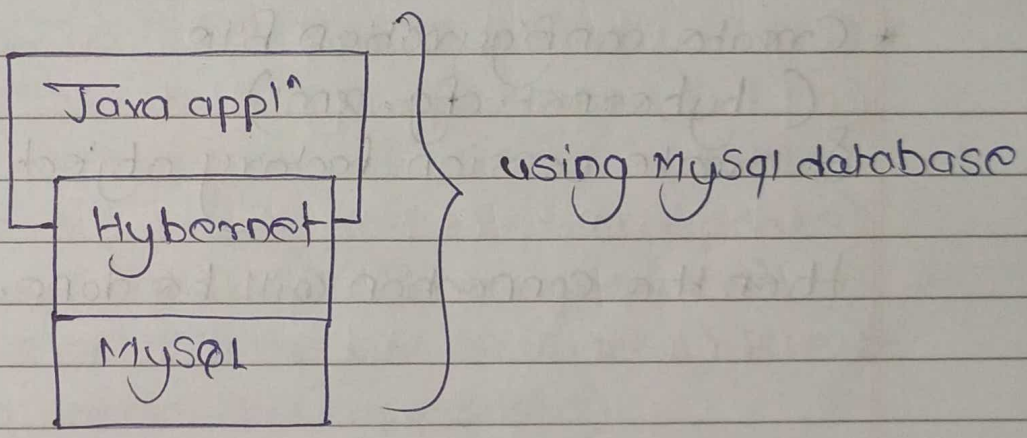
```
class Employee{
    private int empid;
    private String name;
```

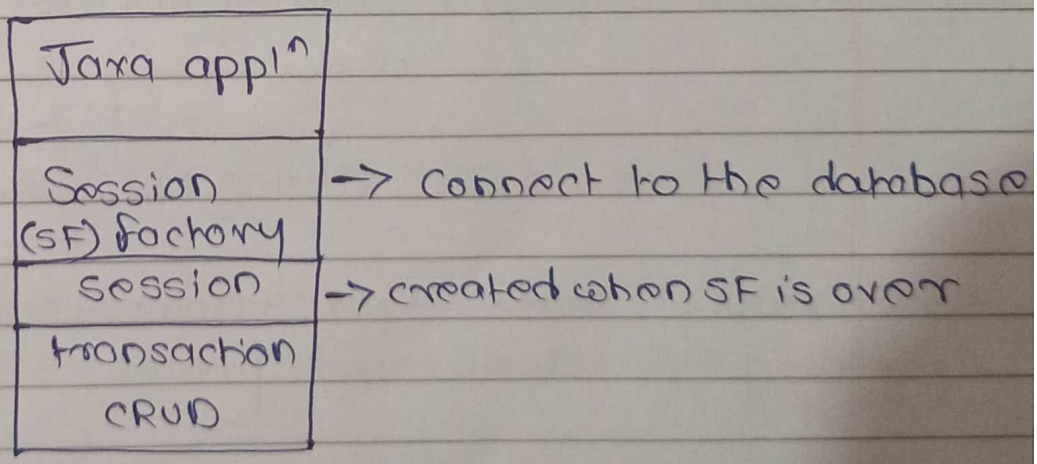
The table will automatic create with the help of hybernet.

Employee

empid	name

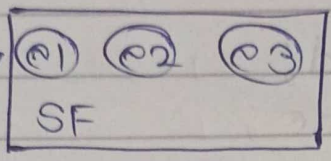
Hybernet configuration is take care of it.





1) Sessionfactory
2) Session
3) Transaction
4) Transaction commit
5) CRUD operations.

} steps

Transient → 

Java application

Session
Transaction
Session.save(e1)
Session.save(e2)

From transient its state change to persistent state. (By using save function)

* Create configuration file
(hybernet.cfg.xml)
& create session factory object

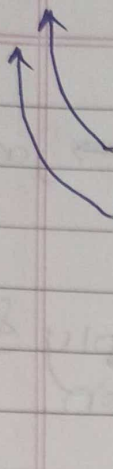
then the connection will be done.

1) Create new folder (JEE-Hybernet)
2) file → maven project → MyFirstHybernet → jar finish.
3) pom.xml and add dependencies in it.
4) run as → maven build → Goal: install & apply & save (The maven dependancies folder has been created).
5) Inside src/main/resources create (hybernet.xml) file & copy-paste data.
6) 1) If dialect is wrong then the file will not work properly. (MySQL8Dialect)
2) <"show-sql">true for seeing query
<"format-sql">true for formatting
<"hbm2ddl.auto">update for table (if table is not their then add & already their then update)
7) In model create new class (Employee.java)
8) In resources create (employee.hbm.xml) file & write mapping inside it. (copy-paste)

```
<class name="com.demo.model.Employee" table="empdb"
>
<id name="empid">
<generator class="assigned"></generator>
</id>
<property name="ename" type="string"></property>
<property name="salary" type="double"></property>
</class>
```

9) Inside test create new class (EmployeeTest) write main method. & write objects.

```
SessionFactory sf = new Configuration().configure().
                    buildSessionFactory();
Session session = sf.openSession();
Transaction tr = session.beginTransaction();
Employee e1 = new Employee(12, "sanjay", 3456);
```



```
Employee e2=new Employee(13,"Archana",5555);
Employee e3=new Employee(14,"Mugdha",6666);
Session.save(e1);
Session.save(e2);
tr.commit();
session.close();
```

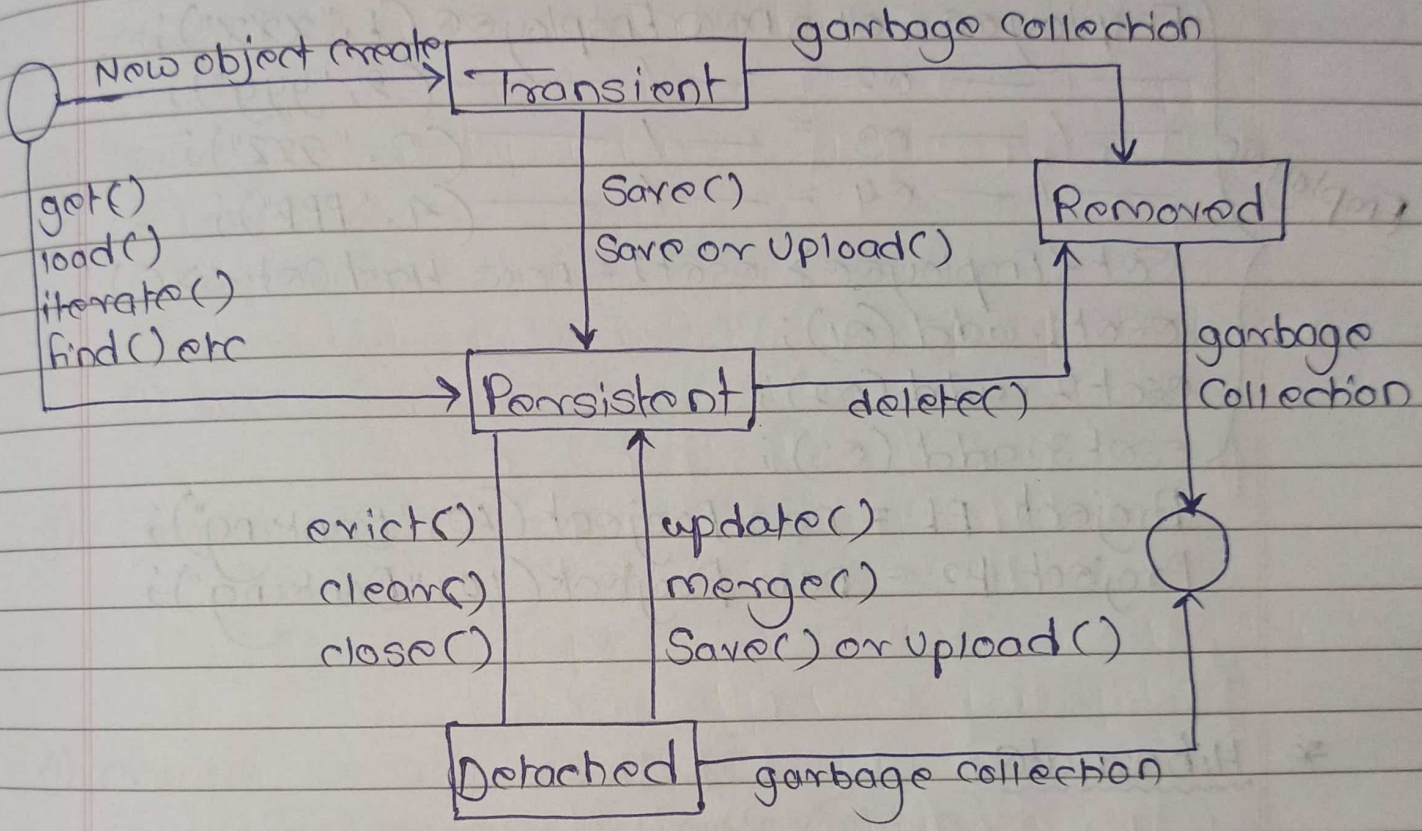
10) run that test.java file

JPA : Java persistance Annotation.

@Entity : used as annotation.
@Table(name="myprodtab") : used for changing table name.
@Id : autogenerate value
@GeneratedValue(Strategy=GenerationType.IDENTITY)
@column(name="id") : changing column name.

Wednesday
10-01-2024

* Object lifecycle



Q. Design one-to-one relation b/w course & faculty

course class has
- cid
- cname
- faculty

faculty class has
- fid
- fname
- course

Thursday
11-01-2024

* ManyToMany relation (Test)

Employee

```
Employee e1 = new Employee(1, "xxx");
—"— e2 = —"— (2, "yyy");
—"— e3 = —"— (3, "zzz");
—"— e4 = —"— (4, "ppp");
Set<Employee> eset1 = new HashSet<>();
eset1.add(e1);
eset2.add(e2);
eset3.add(e3);
Project p1 = new Project(10, "Banking");
Project p2 = new Project(10, "Banking");
```

* Hibernate

Relations :- ① One to One → Lazy Eager
② One to many → Lazy Eager
③ Many to many → Lazy Eager
④ Many to one → Lazy Eager

HQL : Hibernate Query language

* Hibernate CRUD Demo (HibernateCRUDDemo)

1) First (pom.xml) file & build for create Maven dependancies folder. (add dependancies)

2) Make packages
   A) com.demo.model
   B) com.demo.dao
   C) com.demo.service
   D) com.demo.test
   } Create Inside src/main/java

3) Create classes inside (A)
   1) Address.java
   2) MyUser.java

4) create classes inside (D)
   1) TestCrudDemo.java (main)
      write cases inside that test class.

5) create classes inside (B)
   1) UserDao (Interface)
   2) UserDaoImpl (class)

6) create classes inside (C)
   1) UserService (Interface)
   2) UserServiceImpl (class)

7) create one more utility class inside (B)
   1) HibernateUtil.java

* session.createQuery("From user"); {used for create or generate query}

a) First create cases inside test class
b) Then write it inside service class & service Impl class
c) Then write it inside dao class & daoImpl class
d) Then finally that case will be worked....

* CriteriaQueryDemo

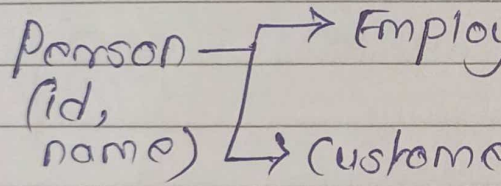
[Restrictions] is the function used for measuring the different conditions like gt (greater than), lt (less than), between, etc.

[logicalExpression] is used for getting the criterion object both or one at a time using (and) & (or) attributes inside Restrictions function.

** Inheritance

Types:

* 1) Single Table: (It takes more space)
1) writing both the classes inside a single table is known as single table, (ex. Employee, customer & [person]) are the classes & parameters of those classes are come inside single table)



ex. Table ↓

id	name	dept	desg	email	fees	descriminator
1	x	a	c	e	500	Emp
2	y	b	d	f	600	custo.

2) hirarchy

@Entity
@Inheritance
@DiscriminatorColumn
} Base class

@Entity
@DiscriminatorValue("...")
} child class

Friday
12-01-2024

* Spring (It is a framework)

EJB : Enterprise Java Bean
JWT : Jason Web Token

* Spring Version 5
* JDK 1.8 or 1.11 version
* Tomcat 9 version

Gradle → use yml file
Maven → use pom.xml file

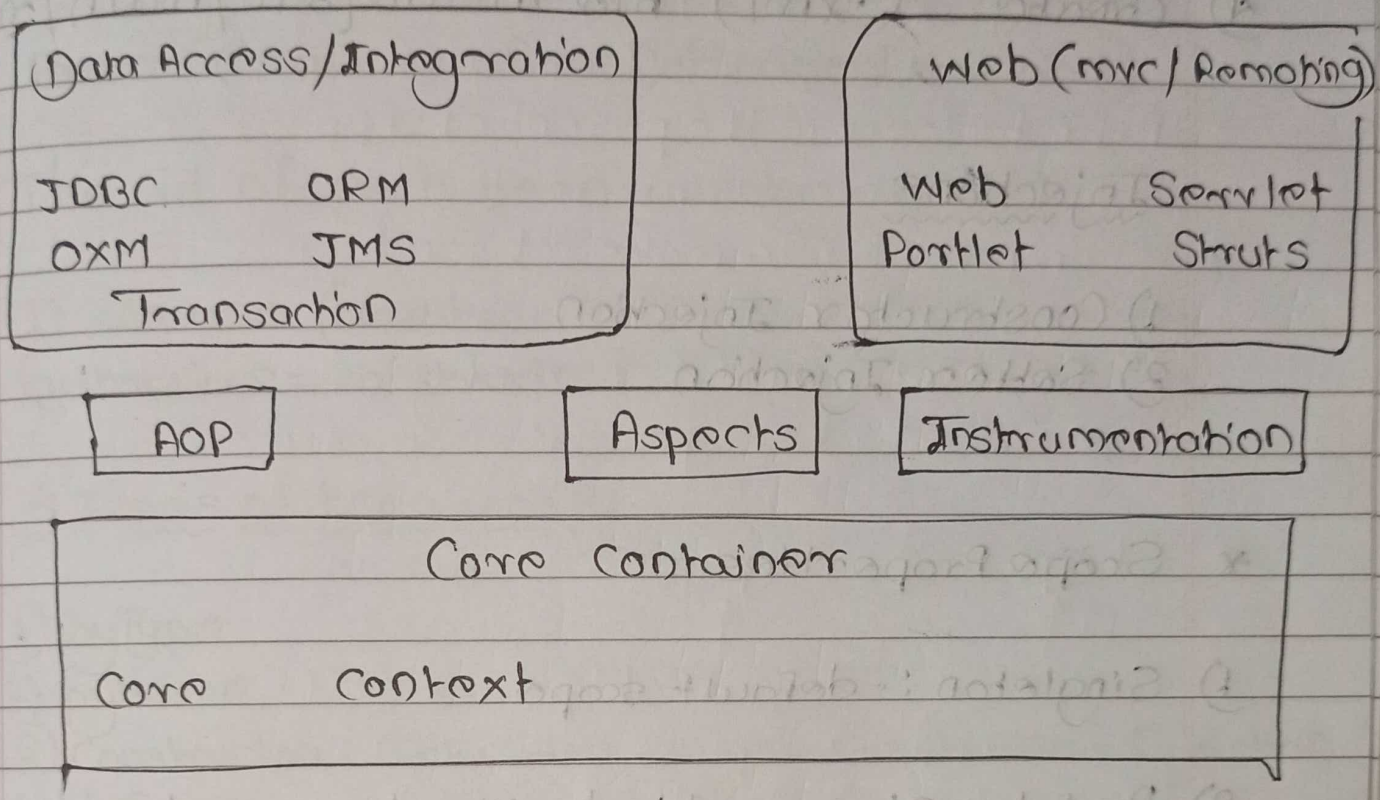
Spring advance

1) library loosely coupled
2) Provides IOC feature (Inversion of Control)
3) lightweight
4) It uses dependency Injection
5) It is bean Container → It manage our control.

AOP : Aspect-Oriented Programming

complexity of application ↓ complexity of program (sol?)

Spring Framework Runtime



Spring Project

1) file → new → maven project → create simple project → SpringHelloWorld (name of the folder)

2) copy paste spring dependancies inside pom.xml.
Spring Framework (5.3.24) version
run as → maven build → install → run

3) Create packages inside src/main/java
1) com.demo.model
HelloWorld (class)
2) com.demo.test
TestHello (class)

[use BeanContainer or BeanFactory for calling appropriate destruction method)

4) Create (.xml) file inside [src/main/resources]

Injections

1) Constructor Injection
2) Setter Injection

* Scope Properties

1) Singleton :- default scope

2) Prototype :- new object will get created every time when getBean or ref is used.

3) Request :- new object will be created for every new request object

4) Session :- new object will be created for every new session.

5) Global-session :- new object will be created for every new context.

* Bean Wiring

When we are using HAS A relation with beans, it means that one bean class contain member of other bean it is called bean wiring.

Q. If autowire & 2 types of same Address

→ Error (NoUniqueBeanDefinationException)

Q. The id of each bean will be unique

Q. If we use (constructor Injection) then use (autowire = constructor).

Imp 4 Types of bean wiring

1) byType
2) byName
3) constructor: Only when we use constructor Injection
4) default

Imp. What is lifecycle of Beans in Spring factory container?

Saturday
13-01-2024

* Internationalization → Resource / Message bundling
* Bean factory is the parent of Application Context
* @Component : To make a bean from class

```
<context : annotation-config>
<context : base-package="com.demo">
```

* Employee employee
  ↑ class ↑ variable (ctx.getBean("employee");
* [Annotation is used after version 3]
* @Component : used for model
  @Service : used for service
  @repository : used for dao
  } all annotations works same.

Scope of all annotations is Singleton.

* @Bean : If we want to use this annotation inside any class then we use,
  @Configuration

ex.
```
@Configuration
class TestData {
    main() {
    }

    @Bean
    JdbcTemplate getjdbcTemplate() {
    }
```

import org.springframework.bean....

we can use

@Component
@Qualifier

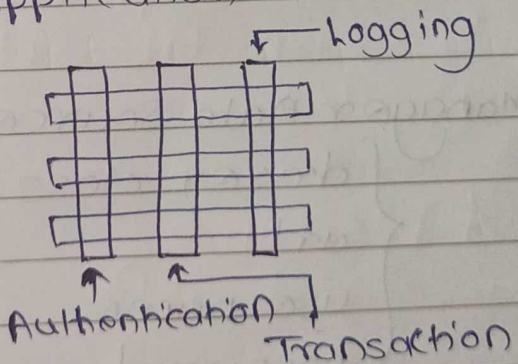
* AOP (Aspect Oriented Programming) {imp for project}

Add two dependancies inside pom.xml
1) Spring aop

Aspect will automatically called by the spring inside AOP.

ex. for banking application

investments
loan's
Recurring depo.



1) Authentication
2) Transaction
3) logging

} are the aspects mostly used. (common aspects)

1) Aspect : cross-cutting
2) Advice : a) Before
b) Return
c) After
d) After Returning
e) After Throwing

3) JoinPoints : All the classes in spring.
4) PointCut : The method which applying advice.
5) Proxy

@Before(execution com.demo.*.*.*.(..))

parameter

package — method or class — class & its funct

* JDBC

JDBC Template → Connection

→ Data Source → DriverManager DataSource

Driver Manager Data Source
- driver class
- url
- username
- password

- Service
- Dao
- JDBC Template
- data source
- Property

Inside ctx

Tuesday
14-01-2024

classmate
Date
Page

Properties (Internationalization)

message.properties → default type
message-GB-EN ..... languages
message-LN-MA ..... location & language } Locale

Group of properties files = Resource Bundling
(Internationalization)

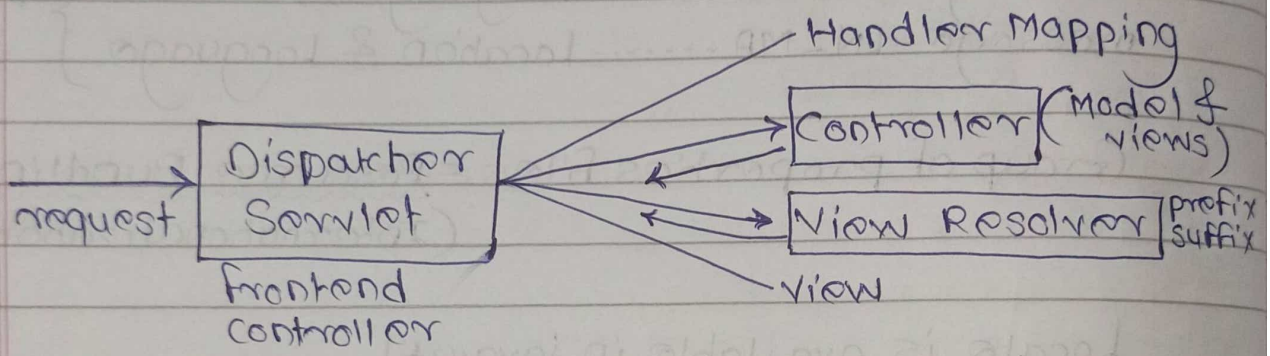
Locale is available in java.util

ex. Locale = new Locale("Marathi", "IN");
↑ language ↑ location
↓ getLanguage() ↓ getCountry()

Locale.US
Locale.UK } Some Inbuild locale

Q. What is the lifecycle of mvc ?
Imp

* MVC (Model view Controller) (war package)



Logical view Converted into Physical view in view resolver.

In Controller :-> Multiple or single
we have to return different services & dao's inside controller.

Standard file name
(spring-servlet.xml)

Package Selection for mvc
(war)

If web.xml is missing then add another file instead that file.

Src → main → webapp → Web-INF → web.xml
(for adding servelet & servelet mapping)
Servelets are present inside web.xml.

Error 404 → (url error) or (Wrong url)
Error 500 → (Programming Error)

: uses in nodejs
{} uses in java