Spring ORM module (Object Relation Mapping)

Spring doesn’t provide any ORM tool. It allow to integrate with existing ORM tool like JPA (Java Persistence API) or Hibernate etc.

Using ORM we can interact with Database like JDBC.

Limitation of JDBC

1. Using JDBC we can’t store as well as can’t retrieve Java Object from database. In DAO layer we need to convert java object into sql query format or vice-versa.
2. JDBC use SQL language. SQL is database dependent language.
3. JDBC throw checked exception. Exception hierarchy is database dependent.
4. JDBC doesn’t support relationship not is a (inheritance) and has a relationship.

ORM : ORM is a concept like OOPs according to ORM we need class in programming language side and that class must be map to table.

Programming language database

Java mysql

Object Relation

Mapping

@Entity

@Table(name=”productdetails”)

class Product { ProductDetails table

@Id

pid PID,name,Price

@Column(name=”name”)

pname columns

price variable

}

1,TV,34000 1,TV,34000

Mapping

Product(class) 🡨-🡪 Product (table)

Pid 🡨--🡪 PID PK

PName🡨🡪 PNAME

Price 🡨🡪PRICE with their data types

Mapping we provide using

1. Xml file (Old version
2. Annotation

If we set come property in xml file or annotation base upon entity class automatically all table will create database.

JPA (Java persistence API)

Hibernate

JPA is specification provided by Java people. JPA is a type of EJB. Hibernate is a framework base upon JPA. Hibernate provide implementation for JPA.

TCL : Transaction control language

commit

rollback

insert, delete and update

commit

or

rollback

throws JDBC by default all DML operation are auto commit.

If we want to do transaction concept using jdbc

con.setAutocommit(false);

after stmt.executeUpdte or pstmt.executeUpdate(“DML Operation”)

con.commit() or con.rollback();

using ORM by default no auto commit. We need to use Transaction.

JPA provide JPQL (Java persistence query language). SQL is database dependent and it retrieve record as string format or query format. JPQL retrieve entity object and database independent.

SQL

Select \* from product; \* means all columns and product is table name. product is not case sensitive.

Select \* from product where pid=101;

Select \* from product where pname like ‘TV’

Select \* from product where price > 50000

Select pid from product; retrieve only pid column

Select pname from product retrieve only pname column

Select pname,price from product retrieve pname and price column

JPQL

select p1 from Product p1; Product is entity class name and case sensitive. p1 is object name. p1 retrieve all variable names.

Select p from Product p where p.pid=100;

p is object and pid and variable name.

select p from Product p where p.pname like ‘TV’;

select p from Product p where p.price > 50000

select p.pid from Product p; partial object retrieve

Select p.pname,p.price from Product p retrieve more than one variable details

Entity class

Account

accno

name

amount

@Entity, @Table, @Id and @Column annotation

create table accountdetails(accountnumber int primary key,customername varchar(30), balance float);