**JDBC**

Works with mysql

Cannot with oracle

So there is need to change code

**ORM**

Object realtational mapping

Pgm technique for converting data btwn relational db and oop language

Hibernate is ORM Servce

**Architecture**

Java appln =>objects => ORM Mapping => db

**Setup**

Spring intialiser

Spring Web

Spring Data JPA

Mysql

Hibernate Plugin install

New hibernate cfg.xml

**Connection establish**

Like connection,statement in mysql jdbc here

Configuration con = new Configuration(); //class

SessionFactory sf = con.buildSessionFactory(); //interface

Session session= sf.openSession(); //interface

session.save(book);

@Entity / @Entity(name=”lib”)

@Id

To create tables when there is an entry

<property name=*"hbm2ddl.auto0"*>update</property>

@Table @Entity

@Column(name=firstname)

Private string firstname;

@Transient

Will not store the property

**Properties**

Show-sql =true

**Query**

Session.get(Book.class,id)

Session.update(book);

Session.save(book);

List<Book> bookList = session.createQuery("FROM Book").list();

**@Embeddable**

Embeds the object inside another object

Private string id;

Private Name name; => Name class with fname,lname

@Embedabble// will embed in other class

Class Name{}

**Relationship**

**OnetoOne**

Class student{

@OneToOne //will create laptop id in student table

Private laptop lap;

}

Class laptop{

}

**OneToMany and ManyToOne**

Class student{

@OneToMany //will create one new table student\_laptop table

Private list<laptop> lap;

}

Class laptop{

}

*To avoid student\_laptop table*

Class student{

@OneToMany // This class will create one new table student\_laptop table

Private list<laptop> lap;

}

Class laptop{

@ManyToOne

Private Student student // will create student id in laptop table but student\_laptop table created by student remains

}

*To avoid student\_laptop table*

Use mapping in student table

@OneToMany (mapping=”student”)

**ManytoMany**

Use @manytomany in both classes with mapping attribute mentioned in one of the classes

**Lazy and eager fetch**

If student object has laptop property while printing laptop property is not printed

To print it use eager fetch

@OnetoMany(mappedBy=””,fetch=FetchType.eager)

Private list<laptop> laptop

**Caching Types**

**Level1 Cache**

When only one session is used, using session.get(laptop.class) multiple times will not perform select query in db multiple time

But doesn’t work for 2 sessions

Session1.get(laptop.class)

Session2.get(laptop .class)

The above will perform 2 queries to avoid use ehcache which is level 2 cache

**Level2 Cache**

Annotations = @cacheable, @cache config in xml

Hibernate ecache config in hbg.xml

**HQL same as SQL**

Hibernate query language

Can write sql in hibernate

|  |  |
| --- | --- |
| Select roll from student | Select roll from Student |
| Roll->column, student->table | Roll->property,Student->Entity |
|  | Query q= session.createQuery(“from student”)  List<stude>=q.list() |
|  | Query q= session.createQuery(“from student where marks >50”)  List<stude>=q.list() |
|  | Query q= session.createQuery(“from student where roll = 32”)  List<stude>=q.uniqueResult(); |
|  | Query q= session.createQuery(“select roll, name from student where marks > 50”)  List<object>=q.list(); |
|  | Query q= session.createQuery(“select roll, name from student where roll =7 ”)  Object[] stu= (object[])q.uniqueResult(); |
|  | Query q= session.createQuery(“select sum(marks) from student ”) |
|  | Let b=60;  Query q= session.createQuery(“select \* from student where marks >= :b ”)  q.setParameter(“b”,b) |
|  |  |
|  |  |
|  |  |

**Lifecycle of objects:**

New = created obj

Tranisent = initialised values

Persistent => stored in db and then attain remove state

Detach =>detached from db

Garbage

**Load and get**

Load = > gets the proxy object (doest use create statement)

Get => gets actual object

JPA

Java persistence API

Switching from hibernate to Top link is not easy

Common standard for hibernate,toplink,ibatis => Easy to switch

JPA cannot be used without hibernate or any frameworks

//=========================================

// Create a User instance

User user = new User("Rajeev", "Singh","MY\_SUPER\_SECRET\_PASSWORD");

Calendar dateOfBirth = Calendar.*getInstance*();

dateOfBirth.set(1992, 7, 21);

// Create a UserProfile instance

UserProfile userProfile = new UserProfile("+91-8197882053", *Gender*.***MALE***, dateOfBirth.getTime(),

"747", "2nd Cross", "Golf View Road, Kodihalli", "Bangalore",

"Karnataka", "India", "560008");

// Set child reference(userProfile) in parent entity(user)

user.setUserProfile(userProfile);

// Set parent reference(user) in child entity(userProfile)

userProfile.setUser(user);

// Save Parent Reference (which will save the child as well)

userRepository.save(user);

//=========================================