02-DBHANDLE with JPA

JPA Java Persistence API

-
- SP/SQL , OOP Entity Repository

, .

CodeLink : http://git.webnori.com/projects/WEBF/repos/spring_jpa/browse

DB

```
# application.properties
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/spring
spring.datasource.username=test
spring.datasource.password=test1234
```

spring.jpa.hibernate.ddl-auto

- `none:` , DB JPA .
- `update:` JPA , .
- `create:` drop+create
- `create-drop:` drop+create drop
- validate :

> *DDL(Data Definitison Language) (Create),(Alert),(Drop)*
>
> *. JPA*

> / DDL
>
> .
>
> ,, // .

- DML(Data Manipulation Language) : Insert/Update/Select ( )
- DCL(Data Control Language) : Commit,RollBack,SavePoint (DB)

JPA DDL/DML/DCL .

JPA class .

	DB	JPA	
Table(Class)	sample_table	SampleTable	
Field(member)	sample_name	sampleName	

> db .
>
> OS ,
>
> JPA db /
>
> .(? ?)

, Class .

, .

- @Table(name=”SAMPLE_TABLE”)
- @Column(name = “COLUMN_NAME”,nullable=false)

JPA , .

application.properties:

- spring.jpa.hibernate.naming.strategy=org.hibernate.cfg.EJB3NamingStrategy
- spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

---

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
```

> DB SQL , JPA DB
>
> , JPA .

> *MYSQL Limit , Oracle Rownum*
>
> *JPA Pageable .*
>
> *, JPA SQL*
>
> *.*

Database (Dialect)

Database SQL JPA Dialect Class Database Dialect Class .

Hibernate Dialect .

Name	
Cache71 Dialect	Caché 2007.1 dialect.
DataDirectOracle9 Dialect	
DB2390Dialect	An SQL dialect for DB2/390.
DB2400Dialect	An SQL dialect for DB2/400.
DB2Dialect	An SQL dialect for DB2.
DerbyDialect	Hibernate Dialect for Cloudscape 10 - aka Derby.
Dialect	Represents a dialect of SQL implemented by a particular RDBMS.
FirebirdDialect	An SQL dialect for Firebird.
FrontBaseDialect	An SQL Dialect for Frontbase.
H2Dialect	A dialect compatible with the H2 database.
HSQLDialect	An SQL dialect compatible with HSQLDB (HyperSQL). HSQLDialect.ReadUncommittedLockingStrategy
InformixDialect	Informix dialect. Seems to work with Informix Dynamic Server Version 7.31.UD3, Informix JDBC driver version 2.21JC3.
Ingres10Dialect	A SQL dialect for Ingres 10 and later versions.
Ingres9Dialect	A SQL dialect for Ingres 9.3 and later versions.
IngresDialect	An SQL dialect for Ingres 9.2.
InterbaseDialect	An SQL dialect for Interbase.
JDataStoreDialect	A Dialect for JDataStore.
MckoiDialect	An SQL dialect compatible with McKoi SQL.
MimerSQLDialect	An Hibernate 3 SQL dialect for Mimer SQL.
MySQL5Dialect	An SQL dialect for MySQL 5.x specific features.

MySQL5InnoDBDialect MySQLDialect	An SQL dialect for MySQL (prior to 5.x).
MySQLInnoDBDialect	
MySQLMyISAMDialect	
Oracle10gDialect	A dialect specifically for use with Oracle 10g.
Oracle8iDialect	A dialect for Oracle 8i.
Oracle9Dialect	Deprecated. Use either Oracle9iDialect or Oracle10gDialect instead
Oracle9iDialect	A dialect for Oracle 9i databases.
OracleDialect	Deprecated. Use Oracle8iDialect instead.
PointbaseDialect	A Dialect for Pointbase.
PostgresPlusDialect	An SQL dialect for Postgres Plus
PostgreSQLDialect	An SQL dialect for Postgres For discussion of BLOB “support” in postrges, as of 8.4, have a peek at http://jdbc.postgresql.org/documentation/84/binary-data.html.
ProgressDialect	An SQL dialect compatible with Progress 9.1C Connection Parameters required: hibernate.dialect org.hibernate.sql.ProgressDialect hibernate.driver com.progress.sql.jdbc.JdbcProgressDriver hibernate.url jdbc:JdbcProgress:T:host:port:dbname;WorkArounds=536870912 hibernate.username username hibernate.password password The WorkArounds parameter in the URL is required to avoid an error in the Progress 9.1C JDBC driver related to PreparedStatements.
RDMSOS2200Dialect	This is the Hibernate dialect for the Unisys 2200 Relational Database (RDMS).
ResultColumnReferenceStrategy	Defines how we need to reference columns in the group-by, having, and order-by clauses.
SAPDBDialect	An SQL dialect compatible with SAP DB.
SQLServer2008Dialect	A dialect for Microsoft SQL Server 2008 with JDBC Driver 3.0 and above
SQLServerDialect	A dialect for Microsoft SQL Server 2000 and 2005
Sybase11Dialect	A SQL dialect suitable for use with Sybase 11.9.2 (specifically: avoids ANSI JOIN syntax)
SybaseAnywhereDialect	SQL Dialect for Sybase Anywhere extending Sybase (Enterprise) Dialect (Tested on ASA 8.x)
SybaseASE15Dialect	An SQL dialect targetting Sybase Adaptive Server Enterprise (ASE) 15 and higher.
SybaseDialect	Deprecated. use AbstractTransactSQLDialect, SybaseASE15Dialect or SQLServerDialect instead depending on need.
TeradataDialect	A dialect for the Teradata database created by MCR as part of the dialect certification process.

TimesTenDialect	A SQL dialect for TimesTen 5.1.

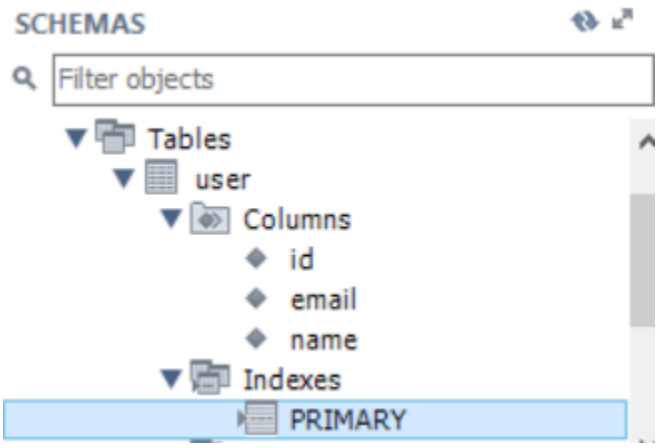
Data Model(Entity)

JPA Id .

.

User Entity

```
package com.psmon.springdb;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity // This tells Hibernate to make a table out of this class
public class User {
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        private Integer id;

        private String name;

        private String email;


    @Column(columnDefinition="char(1)")
    private String categorycode1;

}
```



DataBase , .

DB (ex> char(1), columnDefinition .)

DateTime :

http://www.developerscrappad.com/228/java/java-ee/ejb3-jpa-dealing-with-date-time-and-timestamp/

CRUD

DataBase , SQL , SP Table .

JPA CrudRepository Database .

User , UserRepository .

// , .

CrudRepository Create/Read/Update/Delete .

UserRepository

```
package com.psmon.springdb;

import org.springframework.data.repository.CrudRepository;

//This will be AUTO IMPLEMENTED by Spring into a Bean called userRepository
//CRUD refers Create, Read, Update, Delete

public interface UserRepository extends CrudRepository<User, Long> {

}
```

CRUD

CURD .

, .

Use Case

```
package com.psmon.springdb;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class JparestdemoApplicationTests {

	@Autowired
	private UserRepository userRepository;

	@Test
	public void contextLoads() {
		jpaTest1();
	}

	public void jpaTest1() {
		//
		User addUser = new User();
		addUser.setName("minsu");
		addUser.setEmail("test@x.com");
		userRepository.save(addUser);

		//
		Iterable<User> userList = userRepository.findAll();
		userList.forEach(item->System.out.println(item.getName() ));
	}
}
```

SQL .

```
--
INSERT INTO `spring`.`user`
(`id`,
`email`,
`name`)
VALUES
(<{id:}>,
<{email: >,
<{name:}>);

--
SELECT * FROM user
```

JPA RelationShip

DB ,

.

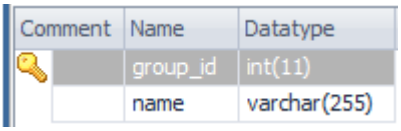
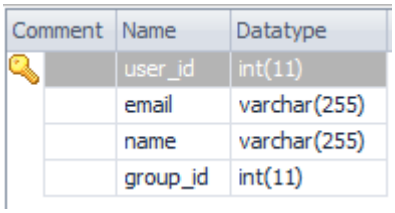
JOIN .

JPA SQL JOIN , / OOP

. .

Class VS Table

Class	Table

Class GroupInfo{
string name;
}
 	
Class User{
GroupInfo groupInfo;
string name;
string email;
}
 	
Class GroupInfoNew{
string name;
List<User> userList;
}
	GroupInfo
	 
	Comment / Name / Datatype
	group_id / int(11)
	name / varchar(255)
	
	User
	 
	Comment / Name / Datatype
	user_id / int(11)
	email / varchar(255)
	name / varchar(255)
	group_id / int(11)
	
	select * from user u join groupinfo g on g.group_id=u.user_id

- Class : , , . (User GroupInfo)
- Table : . (User GroupInfo)
- Class : .
- Table : , .

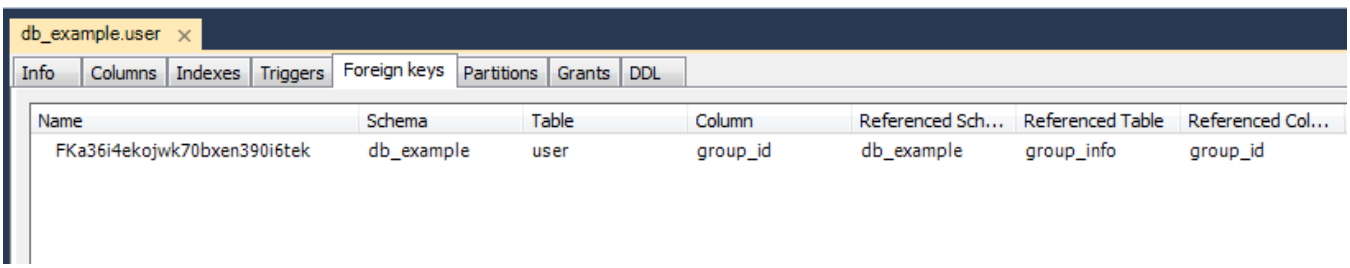
, Table

Class(OOP) (Relation) .

OOP , DB .

- ManyToOne :
- JoinColumn :
- mappedBy :
- OneToMany :
- OneToOne : ,
- ManyToMany : ,
- :

/



JPA , OOP Entity(Class) .

DDL .

, Entity CLASS

.

(DDL) ,

.

DDL OFF, .

DBA //

DDL .

VS

- . SQL .
- 2 .
- . .
- . .
- .

, .

, JPA .

ManyToOne

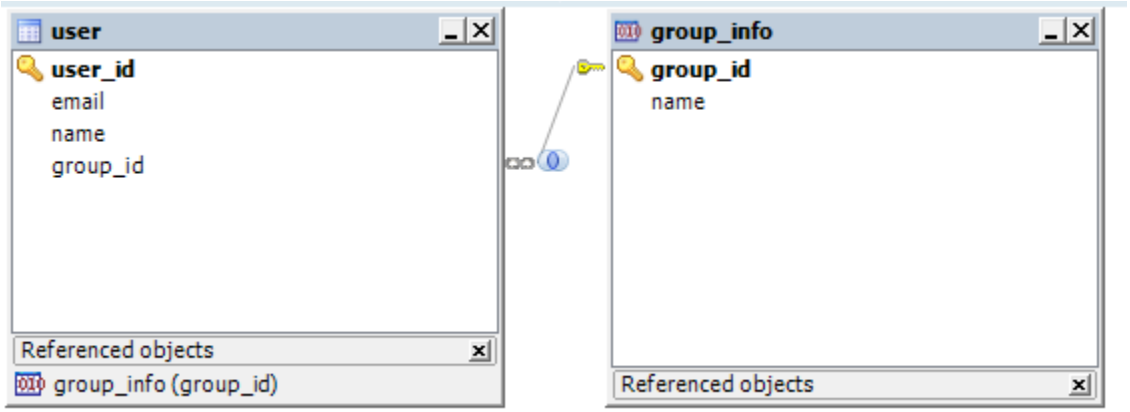
(User) , .

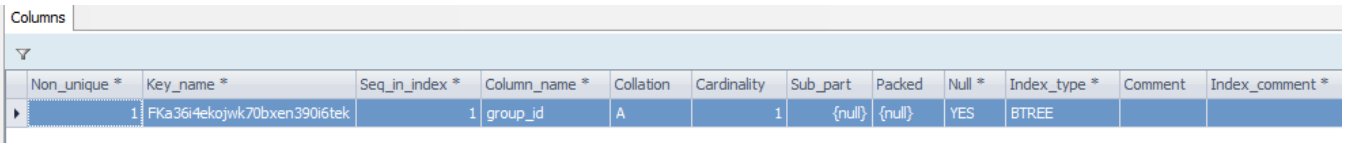
, .

.., .

- ( ) , () .

SQL MODE





Columns

Non_unique *	Key_name *	Seq_in_index *	Column_name *	Collation	Cardinality	Sub_part	Packed	Null *	Index_type *	Comment	Index_comment *
1	FKa36i4ekojwk70bxen390i6tek	1	group_id	A	1	{null}	{null}	YES	BTREE		

, user group_id .

,

, JPA

Relation , .

, DB

OOP .

SQL

SQL

```
CREATE TABLE `group_info` (
  `group_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (`group_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;


CREATE TABLE `user` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `email` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `name` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `group_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  KEY `FKa36i4ekojwk70bxen390i6tek` (`group_id`),
  CONSTRAINT `FKa36i4ekojwk70bxen390i6tek` FOREIGN KEY (`group_id`) REFERENCES `group_info` (`group_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

JPA MODE

GroupInfo Entity

```
@Entity
public class GroupInfo {
        @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
        @Column(name = "GROUP_ID")
    private Integer id;

    private String name;
}
```

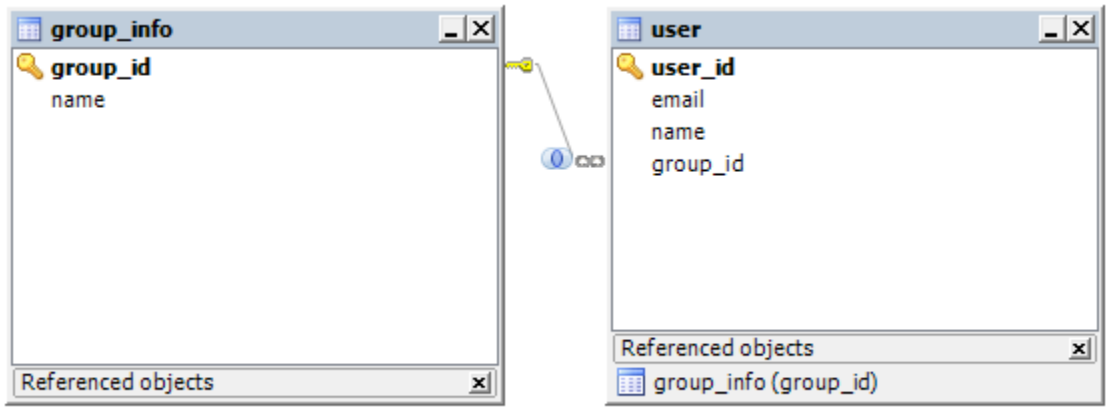
User Entity

```
@Entity
public class User {
        @Id
    @GeneratedValue
    @Column(name = "USER_ID")
    private Integer id;

        private String name;

        private String email;

        @ManyToOne
        @JoinColumn(name = "GROUP_ID", nullable=true )
    private GroupInfo groupInfo;
}
```

```
@Autowired
private GroupRepository groupRepository;


@Autowired
private UserRepository userRepository;

GroupInfo newGroup = new GroupInfo();
newGroup.setName("");
groupRepository.save(newGroup);

//
User addUser = new User();
addUser.setName("minsu");
addUser.setEmail("test@x.com");
addUser.setGroupInfo(newGroup);

userRepository.save(addUser);

// select * from user join group_info   ,     .
Iterable<User> userList = userRepository.findAll();
userList.forEach(item->System.out.println( String.format("Name:%s  GroupName:%s", item.getName(),item.
getGroupInfo().getName() )  ));
```

```
   Insert   ,
      .(    <-> SQL <-> DataSet <-> Object <-> Json )
Json      5      .

JPA       SQL
    .   (  )
    .
```

OneToMany



ManytoOne OneToMany

. Group User .

SQL MODE

DB , List .

/ .. Join

.

```
--  .
INSERT INTO test.group_info(
   group_id
  ,name
) VALUES (
   NULL -- group_id - IN int(11)
  ,''  -- name - IN varchar(255)
)


--    ,  . group id  null
INSERT INTO test.user(
   user_id
  ,email
  ,name
  ,group_id
) VALUES (
   NULL -- user_id - IN int(11)
  ,''  -- email - IN varchar(255)
  ,''  -- name - IN varchar(255)
  ,0   -- group_id - IN int(11)
)


--     .


--  ..Join
select * from user left outter join group_info gi on gi.name=''
```

JPA MODE

JPA , SQL .

Join () .

Join ,

JPA , , Join SQL

FindByname .

Join(InnerJoin,LeftOutterJoin,Right....) JPA Relation .

> JPA ? left outer join
>
> .
>
> ```
> public class User {
> /...
>
>         @ManyToOne
>         @JoinColumn(name = "GROUP_ID", nullable=true )
>     private GroupInfo groupInfo;
> ```
>
> (ManyToOne) ID null .
>
> , InnerJoin
>
> .
>
> InnerJoin , InnerJoin
>
> nuuable=false , JPA User<->Group
>
> , .
>
> JPA / ,
>
> .

SQL Join ,JPA .

Join ? ?

, Join .

? ? JPA .

GroupInfo 1:N

```
@Entity
public class GroupInfo {
  @OneToMany(mappedBy = "groupInfo", cascade = {CascadeType.PERSIST},fetch=FetchType.EAGER)
  private Set<User> users;

        @Override
    public String toString() {
        String result = String.format(
                "GroupInfo[id=%d, name='%s']%n",
                id, name);
        if (users != null) {
            for(User user : users) {
                result += String.format(
                        "User[id=%d, name='%s']%n",
                        user.getId(), user.getName());
            }
        }
        return result;
    }
}


public interface GroupRepository extends CrudRepository<GroupInfo, Long> {
        public GroupInfo findByName(String name);
}
```

- cascade : CascadeType enum enum ALL, PERSIST, MERGE, REMOVE, REFRESH, DETACH .
- targetEntity : Entity Class .
- fetch : FetchType.EAGER, FetchType.LAZY . EAGER Entity LAZY .
- mappedBy : .
- orphanRemoval : Entity DB . cascade cascade JPA DB . false.

Test

```
GroupInfo newGroupA = new GroupInfo("");
Set usersA = new HashSet<User>() {{
        add(new User("minsu2","min2@x.com",newGroupA));
        add(new User("minsu3","min3@x.com",newGroupA));
}};
newGroupA.setUsers(usersA);

GroupInfo newGroupB = new GroupInfo("");
Set usersB = new HashSet<User>() {{
        add(new User("tom1","tom1@x.com",newGroupB));
        add(new User("tom2","tom2@x.com",newGroupB));
}};
newGroupB.setUsers(usersB);

    groupRepository.save(new HashSet<GroupInfo>() {{
            add(newGroupA);
            add(newGroupB);
    }});


GroupInfo groupInfo = groupRepository.findByName("");
    System.out.println( String.format(": %s", groupInfo.toString() ) );
```

, Group

.

save , .

,

. (, Join .)

Paging

JPA MODE

```
public interface UserPageRepo extends Repository<User, Long>{

        Optional<User> findOne(Long id);

        //
        Page<User> findAll(Pageable pageRequest);

        //  -
        Page<User> findByGroupInfoName(String groupName,Pageable pageRequest);

        // Update
        void delete(User deleted);

        User save(User persisted);

        void flush();
}
```

SQL , / .

JPA , SQL .

- JPA : findBy{A}**Or**findBy{B}**And**findBy{C}(....);
- SQL : where A=A or B=B and C

.:

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.query-methods.query-lookup-strategies

UseCase

```
                //Test  100
                GroupInfo newGroupA = new GroupInfo("");
                Set usersA = new HashSet<User>() {{
                        for(int i=0; i<100 ; i++) {
                                String userName = String.format("minsu%d", i);
                                String email = String.format("min%d@x.com", i);
                                add(new User(userName,email,newGroupA));
                        }
                }};
                newGroupA.setUsers(usersA);
        groupRepository.save(new HashSet<GroupInfo>() {{
                add(newGroupA);
        }});


                //  .(  ,  )
        PageRequest  pageRequest =  new PageRequest(1,10);
        Page<User> sPage = userPageRepo.findAll(pageRequest);
        System.out.println( String.format(" %d:Contents %d:Page", sPage.getNumberOfElements(),sPage.getNumber()
) );

        Page<User> sPage2 = userPageRepo.findByGroupInfoName("", pageRequest);
        System.out.println( String.format(" %d:Contents %d:Page", sPage2.getNumberOfElements(),sPage2.
getNumber() ) );
```

```
pageRequest =  new PageRequest(0,10, Sort.Direction.DESC , "a","b","c"  );
pageRequest =  new PageRequest(0,10, Sort.Direction.DESC , "a",Sort.Direction.ASC,"b"  );
```

```
Sort.Direction sortDir = Sort.Direction.DESC;
if( sortdir.equals("asc") ) sortDir = Sort.Direction.ASC;

Sort sortOpt = new Sort( sortDir, sort )
                                                .and( new Sort(Sort.Direction.DESC, "customervaluation") );
```

JPQL MODE

```
public interface UserPageRepo extends Repository<User, Long>{

        // QueryMode
        @Query(value="select t from User t "
                        + "where  t.name =:name  "
                        + "order by t.id " , nativeQuery=false )
        List<User> findBySomeName( @Param("name") String name, Pageable pageable);

}
```

JPA , SQL .

, SQL , SQL .

JPQL VS SQL(Native)

JPQL JPA DB .

JPQL ,

DB .

SQL(Native) , DB .

Mysql Limit

DB SQL , .

nativeQuery , true/flase ,

JPQL .

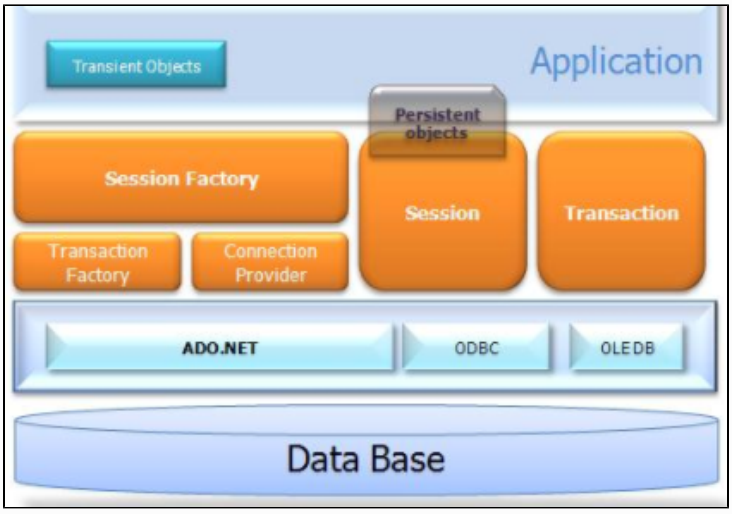
:

JPA : http://blog.woniper.net/255

(.net) JPA .

Entity Framework : https://docs.microsoft.com/en-us/ef/core/

NHibernate : https://en.wikipedia.org/wiki/NHibernate /



http://www.sqler.com/401779

?

.

MVC

.

	JAVA(JPA)	.NET(EntityFrame)
	Entity-	
Persitent	Repository	DbContext
	QueryDSL	Linq