Hibernate is a tool that implement ORM concept. Hibernate persistence api provide traditional jpa annotation as well as xml based

JPA – specification, standard for persisiting data into the database is the pre defined annotation,

JPA advantage is that so if you change different (TopLink)

.save() is hibernate is hibernate specific method. Return type is Serializable Object(primary key of the entity)

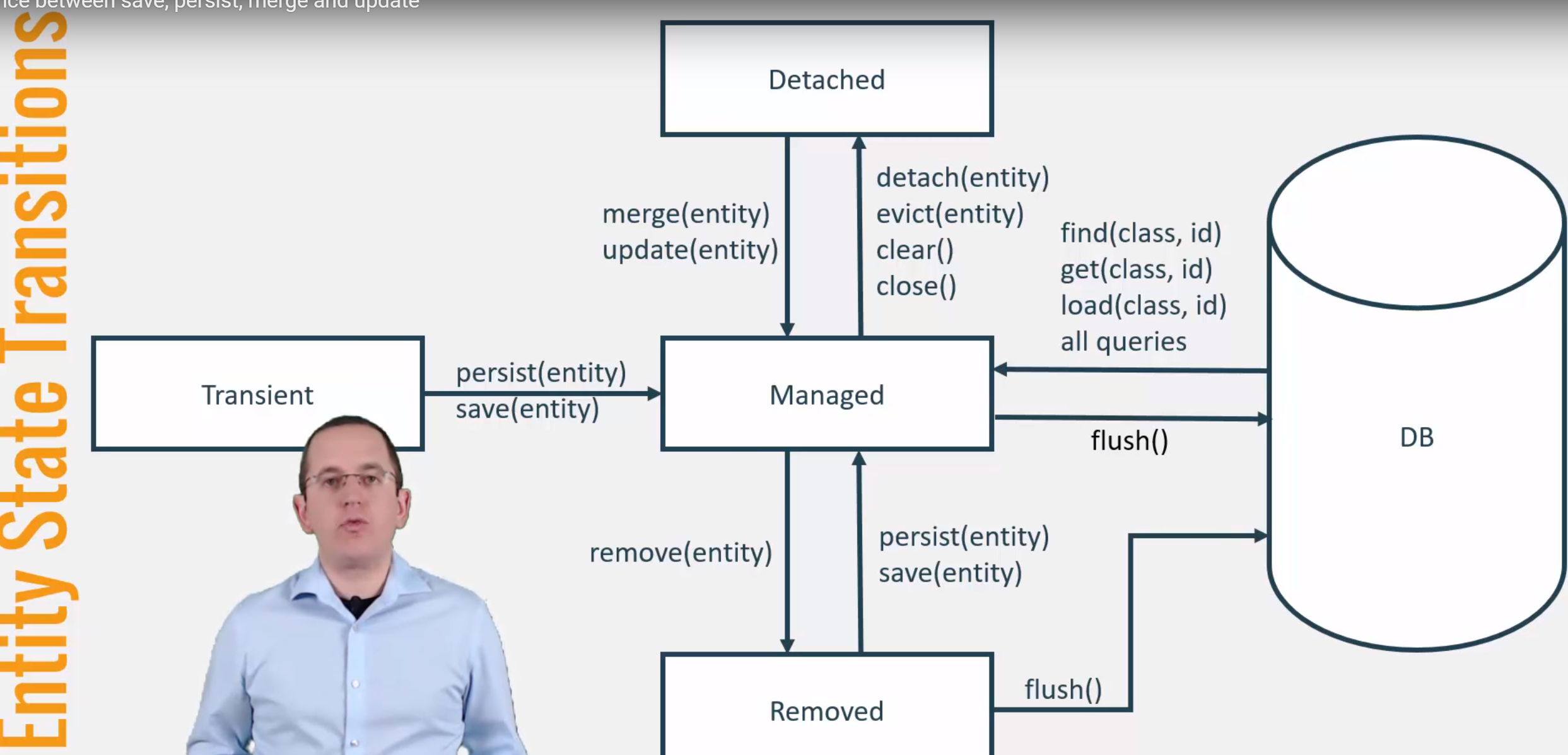
.persist() is JPA standard method. Return type is void

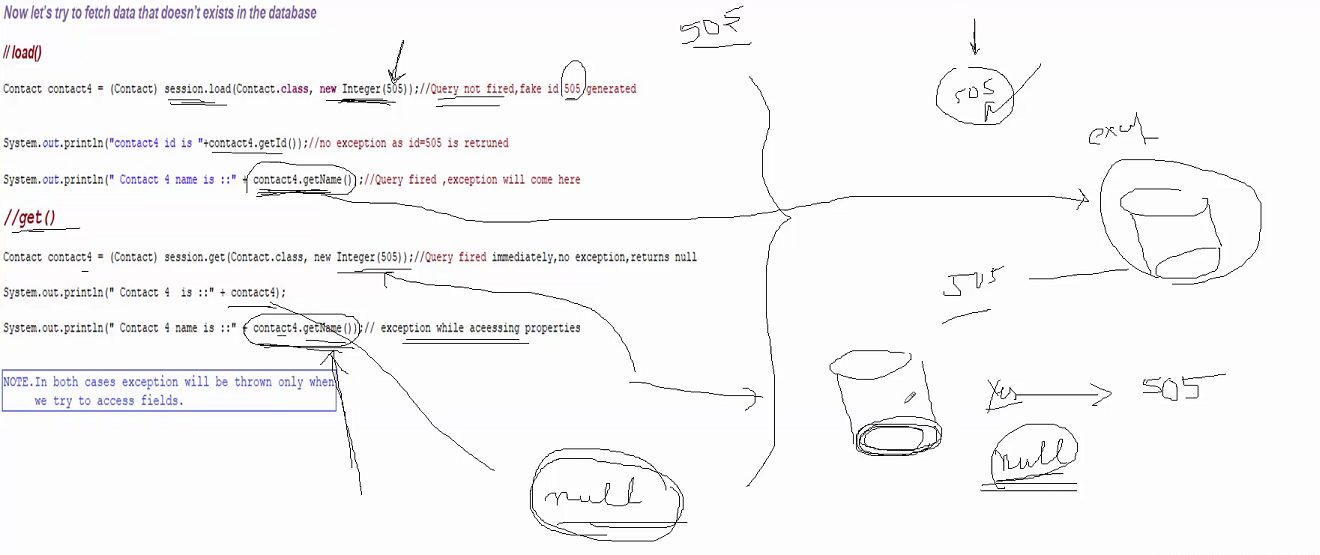
.load() returns proxy object. Only use load if you sure that the object is exist. It throws exeption.

.get() the actual object. Only use if you not sure if the object exist in the database. It returns null

.update() is hibernate

.merge()





from Synergisticit Synergisticit to everyone:

What is ORM?

from Synergisticit Synergisticit to everyone:

Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables.

It also provides various API’s to perform different types of operations on the data tables.

from Synergisticit Synergisticit to everyone:

What is Hibernate Framework?

from Synergisticit Synergisticit to everyone:

Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

After java objects mapping to database tables, database is used and handled using Java objects without writing complex database queries.

We can use Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

from Synergisticit Synergisticit to everyone:

What is Java Persistence API (JPA)?

from Synergisticit Synergisticit to everyone:

Java Persistence API (JPA) provides specification for managing the relational data in applications. Current JPA version 2.1 was started in July 2011 as JSR 338. JPA 2.1 was approved as final on 22 May 2013.

JPA specifications is defined with annotations in javax.persistence package. Using JPA annotation helps us in writing implementation independent code.

from Synergisticit Synergisticit to everyone:

Name some important annotations used for Hibernate mapping?

from Synergisticit Synergisticit to everyone:

Hibernate supports JPA annotations and it has some other annotations in org.hibernate.annotations package. Some of the important JPA and hibernate annotations used are:

1. javax.persistence.Entity: Used with model classes to specify that they are entity beans.

2. javax.persistence.Table: Used with entity beans to define the corresponding table name in database.

3. javax.persistence.Access: Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.

4. javax.persistence.Id: Used to define the primary key in the entity bean.

5. javax.persistence.EmbeddedId: Used to define composite primary key in the entity bean.

from Synergisticit Synergisticit to everyone:

6.javax.persistence.Column: Used to define the column name in database table.

7. javax.persistence.GeneratedValue: Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.

8. javax.persistence.OneToOne: Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany

9. org.hibernate.annotations.Cascade: Used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType

from Synergisticit Synergisticit to everyone:

Difference between save and persist method in Hibernate

from Synergisticit Synergisticit to everyone:

Difference between save and persist method in Hibernate?

from Synergisticit Synergisticit to everyone:

1)First difference between save and persist is there return type. Similar to save method persist also INSERT records into database but return type of persist is void while return type of save is Serializable object.

2) Another difference between persist and save is that both methods make a transient instance persistent. However, persist() method doesn't guarantee that the identifier value will be assigned to the persistent instance immediately, the assignment might happen at flush time(just before commit).

3) Save() takes comparatively more time to execute

4) Save() is hibernate specific while persist() is JPA standard

from Synergisticit Synergisticit to everyone:

get() vs load():????????

from Synergisticit Synergisticit to everyone:

get() returns the object by fetching it from database or from hibernate cache whereas

load() just returns the reference of an object that might not actually exists, it loads the data from database or cache only when you access other properties of the object.

Based on the above explanations we have following differences between get() vs load():

1. get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading.

2. Since load() throws exception when data is not found, we should use it only when we know data exists.

3. We should use get() when we are not sure data exists in the database.

from Synergisticit Synergisticit to everyone:

=========public class Contact {

private int id;

private String name;

private String email;

private String address;

private String telephone;

public Contact() {

}

public Contact(int id, String name, String email, String address,

String telephone) {

this.id = id;

this.name = name;

this.email = email;

this.address = address;

this.telephone = telephone;

}

public Contact(String name, String email, String address, String telephone) {

this.name = name;

this.email = email;

this.address = address;

this.telephone = telephone;

}

from Synergisticit Synergisticit to everyone:

=====ContactManager==========

from Synergisticit Synergisticit to everyone:

package com.mycompany;

import java.io.Serializable;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.service.ServiceRegistryBuilder;

//to show that get() does not throw exception

public class ContactManager {

public static void main(String[] args) {

// loads configuration and creates a session factory

Configuration configuration = new Configuration().configure();

ServiceRegistryBuilder registry = new ServiceRegistryBuilder();

registry.applySettings(configuration.getProperties());

ServiceRegistry serviceRegistry = registry.buildServiceRegistry();

SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);

// opens a new session from the session factory

Session session = sessionFactory.openSession();

session.beginTransaction();

// persists one new Contact objects

Contact contact1 =

from Synergisticit Synergisticit to everyone:

Contact contact1 = new Contact(987,"Tonmoy", "bill@gmail.com", "USA", "18001900");

Serializable id = session.save(contact1);//returns type Serializable

System.out.println("created id is :"+id);

Contact contact2 = new Contact(777,"Alex", "bill@gmail.com", "USA", "18001900");

session.persist(contact2);

Contact contact3= new Contact(888,"prabhat","pkfag@gmail.com", "USA", "1781900");

session.persist(contact3);

//I want to load an object

Contact contact4 = (Contact) session.load(Contact.class, new Integer(1505));

System.out.println("contact4 id is "+contact4.getId());//fake id 1505 will be generated

////

/////

System.out.println(" Contact 4 name is ::" + contact4.getName());//exception will come here

from Synergisticit Synergisticit to everyone:

session.getTransaction().commit();

session.close();

}

}

from Synergisticit Synergisticit to everyone:

==contact.hbm.xml

from Synergisticit Synergisticit to everyone:

<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.mycompany">

<class name="Contact" table="CONTACT">

<id name="id" column="CONTACT\_ID">

<!-- <generator class="increment"/> -->

</id>

<property name="name" column="NAME"/>

<property name="email"/>

<property name="address"/>

<property name="telephone"/>

</class>

</hibernate-mapping>

from Synergisticit Synergisticit to everyone:

==hibernate.cfg.xml

from Synergisticit Synergisticit to everyone:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/contactdb</property>

<property name="connection.username">root</property>

<property name="connection.password">root</property>

<property name="show\_sql">true</property>

<mapping resource="com/mycompany/Contact.hbm.xml"/>

</session-factory>

</hibernate-configuration>

from Synergisticit Synergisticit to everyone:

<property name="hbm2ddl.auto">create</property>

from Synergisticit Synergisticit to everyone:

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>