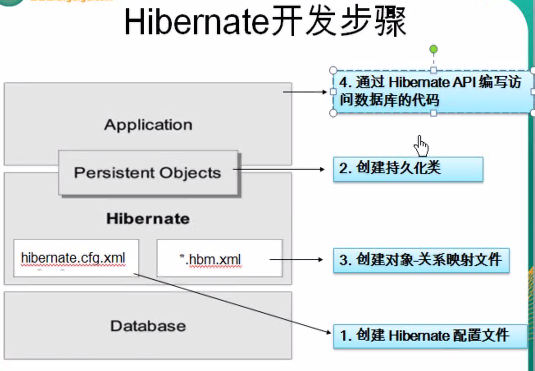
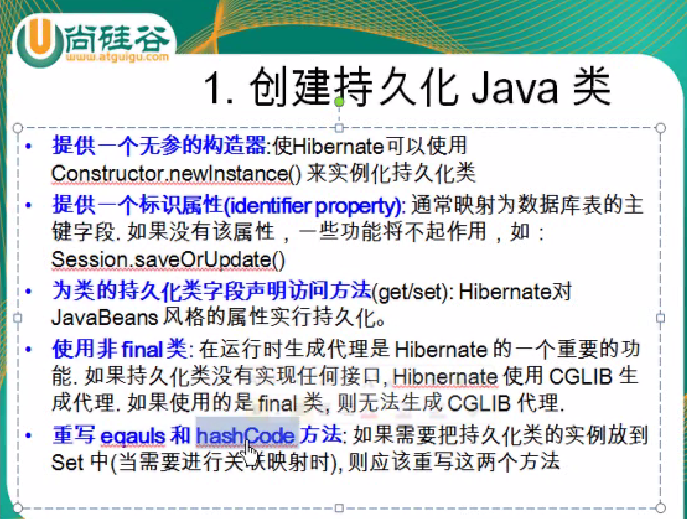
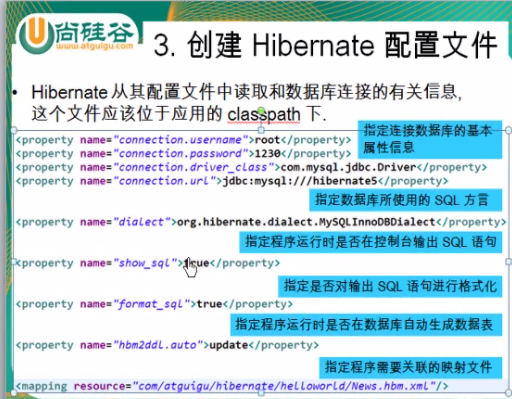
ideo by “atguigu”…

Video 1-4:

1. jar包 = hibernate-release-4.2.4.Final/lib/required + mysqlDriver
2. Hibernate四步曲



1. 注意

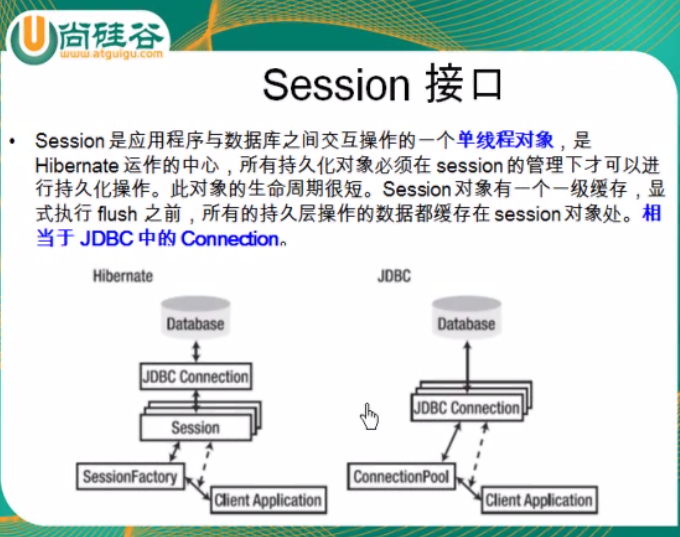


4.

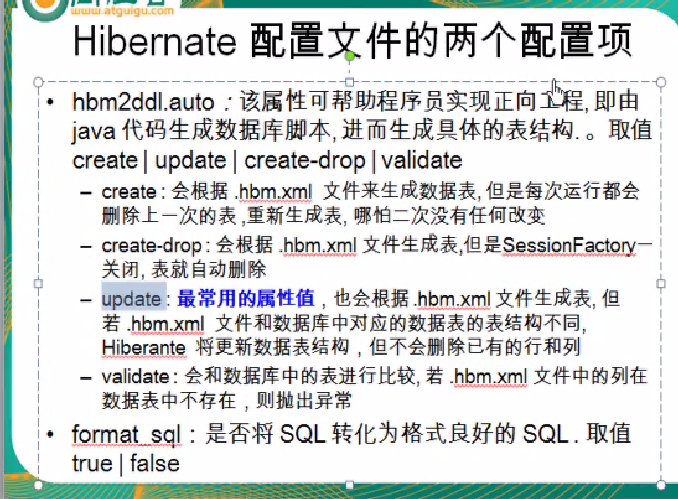
1）Configeration类管理Hibernate配置信息



2）线程安全

3）相当于对Connection封装

4）任何事务都要在Transaction下执行

5）

6）Session接口

7）用junit单元测试类完成之后的工作

**package** com.atguigu.hibernate.entities;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** org.hibernate.service.ServiceRegistry;

**import** org.hibernate.service.ServiceRegistryBuilder;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** HibernateTest {

**private** SessionFactory sessionFactory;

**private** Session session;

**private** Transaction transaction;

@Before

**public** **void** init(){

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

ServiceRegistry serviceRegistry = **new** ServiceRegistryBuilder().

applySettings(configuration.getProperties())

.buildServiceRegistry();

sessionFactory = configuration.buildSessionFactory(serviceRegistry);

session = sessionFactory.openSession();

transaction = session.beginTransaction();

}

@After

**public** **void** destroy() {

transaction.commit();

session.close();

sessionFactory.close();

}

@Test

**public** **void** test() {

}

}

Video 5-6

1. Session的缓存
   1. Flush()：使得数据库中的记录和Session中一致，为了保持一致在Transaction的commit()方法中：先调用session的flush方法，再提交事务

Flush()方法可能会发送SQL语句(如果session里数据和数据库数据不一致才发)，不会提交事务

注意：在未提交事务或者显示调用session.flush()之前，也有可能会进行flush操作。。

1).执行HQL或者QBC查询，会进行flush操作,以得到数据表最新数据

2).若记录的ID是由底层数据库采用自增的方式生成的，则在调用save()方法之后，会立即发送Insert语句。 因为save方法之后，必须报账对象id存在。

@Test

**public** **void** testSessionFlush2(){

News news = **new** News("java", "SUN", **new** Date());

session.save(news);

}

@Test

**public** **void** testSessionFlush(){

News news = (News) session.get(News.**class**, 1);

news.setAuthor("SUN");

session.flush();

System.***out***.println("flush..");

}



b) refresh()方法 ：refresh会强制发送SQL语句

//刷新缓存 51行断点处手动修改mysql数据库记录

//1,若发现两个sysout语句结果一样，则是数据库隔离级别的问题，mysql的隔离级别默认是repeatable read，改成啥去了。。

@Test

//refresh会强制发送SQL语句

**public** **void** testRefresh(){

News news = (News) session.get(News.**class**, 1);

System.***out***.println(news);

session.refresh(news);

System.***out***.println(news);

}

* 1. clear()清理缓存

//清理缓存

@Test

**public** **void** testClear(){

News news = (News) session.get(News.**class**, 1);

session.clear();

News news2 = (News) session.get(News.**class**, 1);

}

Video 6：

1. 持久化对象的状态（脑海里形成一个公司与一个员工的关系）

临时对象：new出来的

持久化对象：save之后，位于Session缓存中，托管对象；持久化状态时，不允许程序随意修改其OID

删除对象：delete 曾经被session监管过，曾经在数据库中有记录

游离对象：save之后，close之后脱管对象，员工“请假”

1. save方法：

将临时对象变为持久化对象，为对象分配ID

在flush缓存时会发送一条insert语句

在save方法之前的id是无效的

1. persist方法：也会执行insert操作

和save的区别：在调用 persist方法之前，若对象已经有ID了，则不会执行insert，会抛出异常。



Video7- 依然是讲session核心方法

1. update方法
2. /\*
3. \* update
4. \* 1.若更新一个持久化对象，不需要显示调用update，调用setXXX即可。
5. \* 2.更新一个游离对象，需要显示敲update方法,
6. \* 可以把游离对象变为持久化对象
7. \* 需要注意的是：
8. \* 1.如果把news.setAuthor("DB4")；注释，还是会发update语句。。。
9. \* 如何能够让update方法不盲目发送update语句呢？
10. \* 在.hbm.xml文件class节点设置 select-before-update = true
11. \* 但通常不需要设置。
12. \* 2.news.setId(1000);行 对于数据表中不存在的对象，但还调用了update方法，会StaleStateException
13. \* 3.当update方法关联一个有游离对戏那个
14. \*如果在session缓存中已经存在OID相同的，会异常，因为在session缓存中不能存在两个IOD相同的对象。
15. \*
16. \*/

@Test

**public** **void** testUpdate(){

News news = (News) session.get(News.**class**, 1);

transaction.commit();

session.close();

session = sessionFactory.openSession();

transaction = session.beginTransaction();

news.setAuthor("SUN");

session.update(news);

}

1. saveOrUpdate()方法

判断对象为临时对象的方法：

OID是否为null（一般情况下）

极端情况下，。。。<id> 设置了unsaved-value属性

/\*

\* 注意：

\* 1.若OID不为空，但数据表中还没有和它对应的记录，会异常

\* 2.了解：若OID=id的unsaved-value 属性值的对象，也认为是游离对象

\*/

@Test

**public** **void** testSaveOrUpdate(){

News news = **new** News("FF", "fffggadsg", **new** Date());

news.setId(1);

session.saveOrUpdate(news);

}

1. delete方法：只要在数据库中有记录和session中OID对应，就删除；

没有的话就异常

4. evict从缓存中移除对象