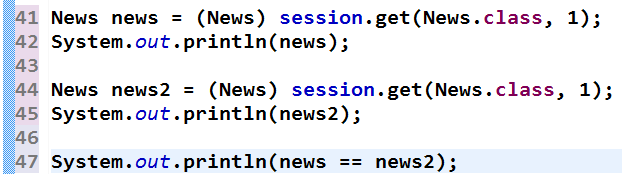
通过session操纵对象

1. session缓存机制
   1. session缓存机制及其优点

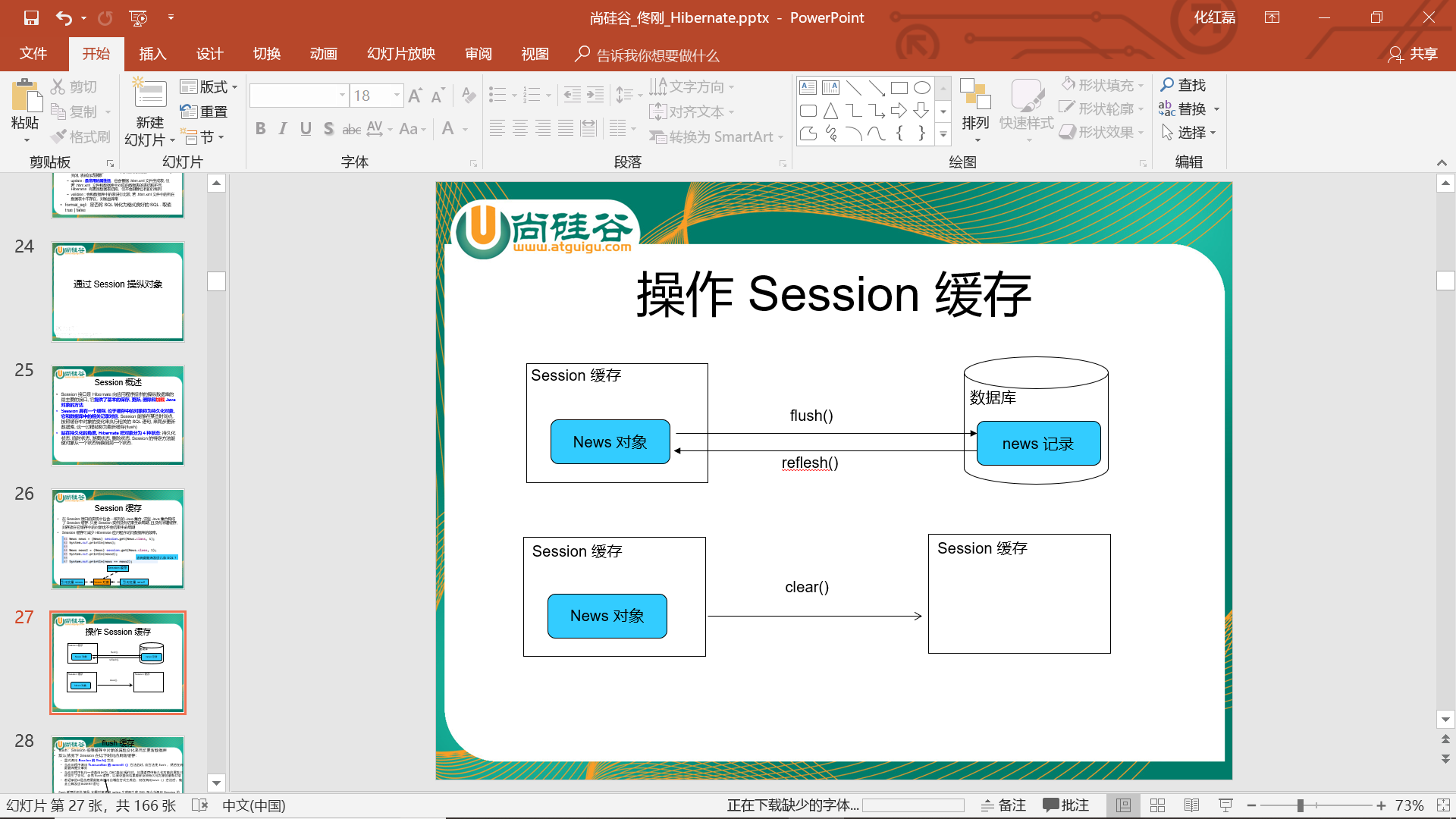
-在 Session 接口的实现中包含一系列的 Java 集合, 这些 Java 集合构成了 Session 缓存. 只要 Session 实例没有结束生命周期, 且没有清理缓存，则存放在它缓存中的对象也不会结束生命周期

-Session 缓存可减少 Hibernate 应用程序访问数据库的频率。

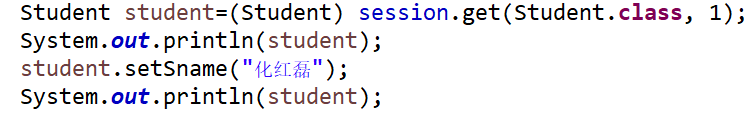
-比如，以下代码只向数据库发送一次sql



* 1. 操纵session的缓存，主要有三个方法，flush、reflesh、clear三个方法

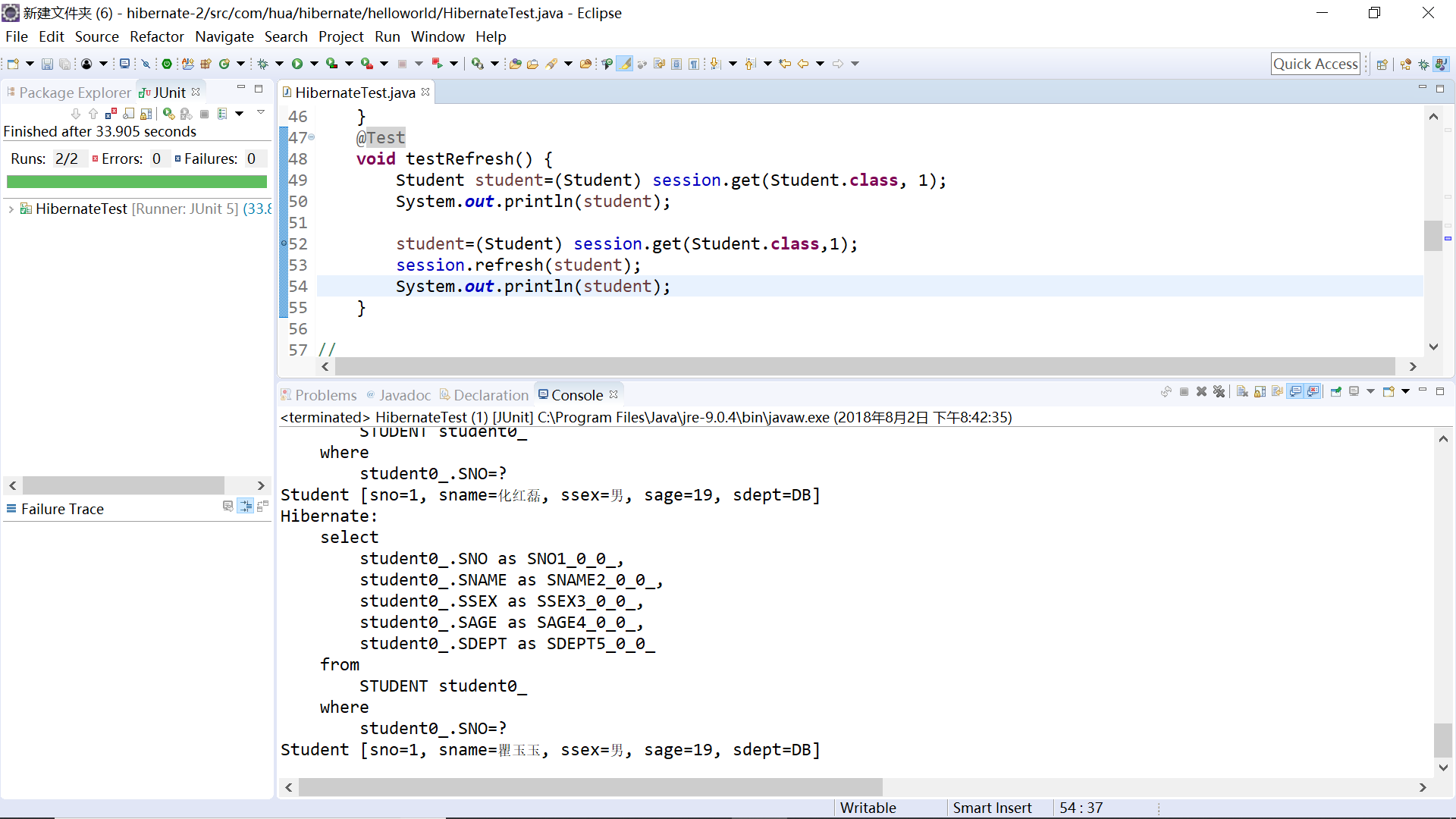


A）.flush方法

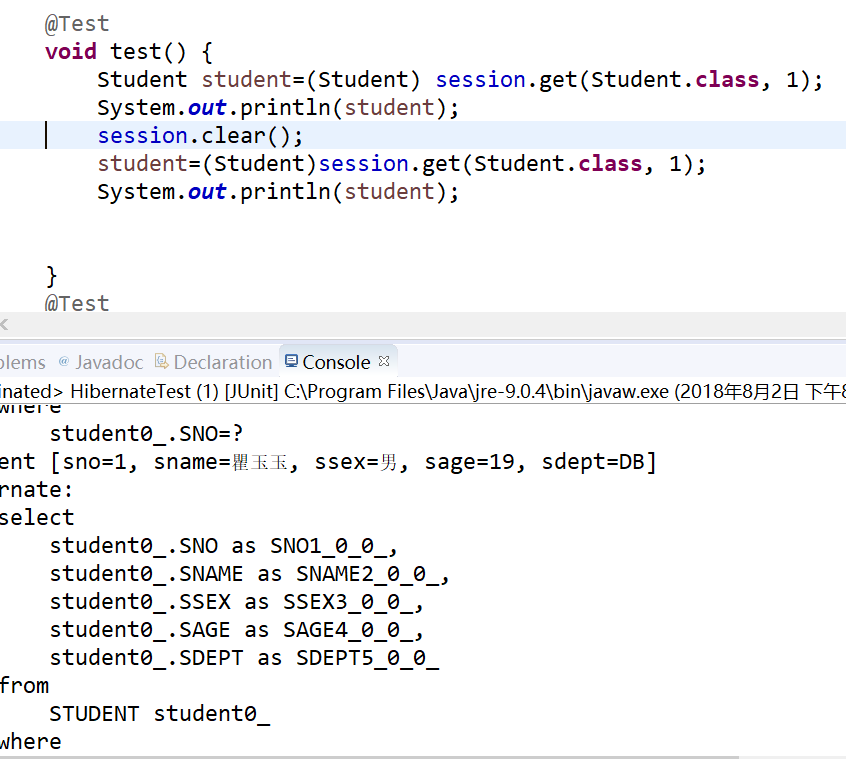
* flush：Session 按照缓存中对象的属性变化来同步更新数据库
* 默认情况下 Session 在以下时间点刷新缓存：
  + 显式调用 **Session 的 flush()** 方法
  + 当应用程序调用 **Transaction 的 commit（）**方法的时, 该方法先 flush ，然后在向数据库提交事务
  + 当应用程序执行一些查询(HQL,QBC查询)操作时，如果缓存中持久化对象的属性已经发生了变化，会先 flush 缓存，以保证查询结果能够反映持久化对象的最新状态
  + 若记录的id是由底层数据库使用自增的方式生成的，则在调用save（）方法后，就会立即发送INSERT语句
  + flush 缓存的例外情况: 如果对象使用 native 生成器生成 OID, 那么当调用 Session 的 save() 方法保存对象时, 会立即执行向数据库插入该实体的 insert 语句.
* commit() 和 flush() 方法的区别：flush 发送一系列 sql 语句，但不提交事务；commit 方法先调用flush() 方法，然后提交事务. 意味着提交事务意味着对数据库操作永久保存下来。
* 

B).refresh方法

例如在前后两次读取数据库信息过程中，第一次读取数据库信息之后，数据库中信息被修改了，如果在其中调用了refresh方法，会强制发送SELECT语句，以使Session缓存中的状态和数据表中对应的记录保持一致（注意mysql数据的话会出现隔离级别的问题，因为mysql数据库默认是读已提交）



C).clear方法，清理缓存的方法，字面意思，清理缓存，比如下面例子，就执行了两条select语句



1. 如何在hibernate里设置数据库的隔离级别
   1. JDBC 数据库连接使用数据库系统默认的隔离级别. 在 Hibernate 的配置文件中可以显式的设置隔离级别. 每一个隔离级别都对应一个整数（分别为1248）:

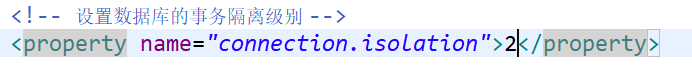
1. READ UNCOMMITED

2. READ COMMITED

4. REPEATABLE READ

. 8SERIALIZEABLE

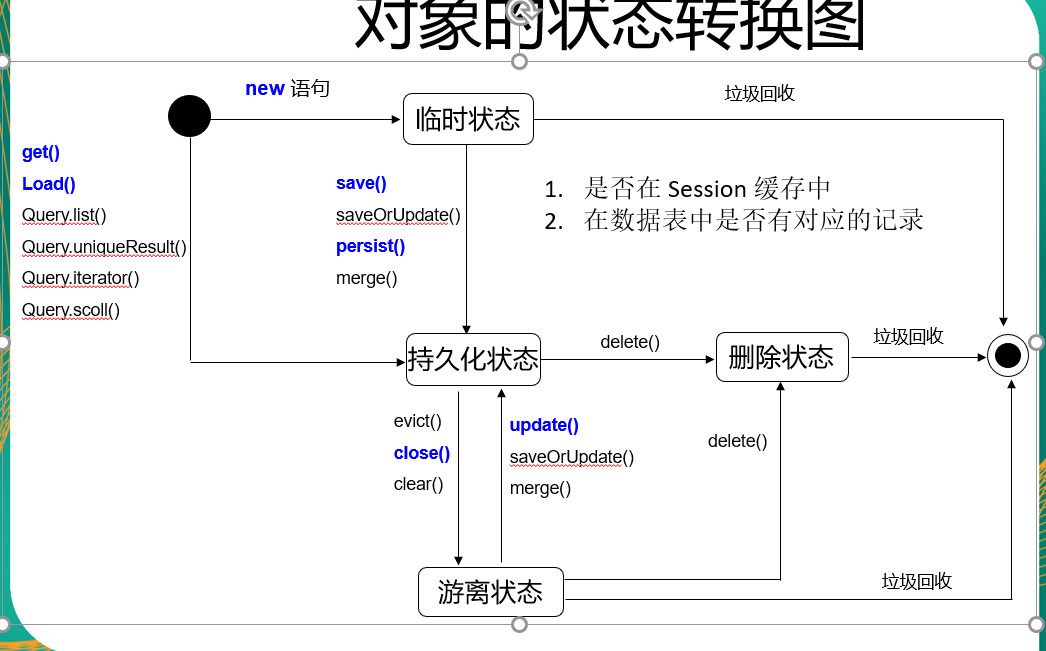
* 1. Hibernate 通过为 Hibernate 映射文件指定 hibernate.connection.isolation 属性来设置事务的隔离级别



1. 持久化对象的状态
   1. 概述

**站在持久化的角度, Hibernate 把对象分为 4 种状态**: 持久化状态, 临时状态, 游离状态, 删除状态. Session 的特定方法能使对象从一个状态转换到另一个状态.

* 临时对象（Transient）:
  + 在使用代理主键的情况下, **OID 通常为 null**
  + **不处于 Session 的缓存中**
  + **在数据库中没有对应的记录**
* 持久化对象(也叫”托管”)（Persist）：
  + **OID 不为 null**
  + **位于 Session 缓存中**
  + 若在数据库中已经有和其对应的记录, **持久化对象和数据库中的相关记录对应**
  + **Session 在 flush 缓存时, 会根据持久化对象的属性变化, 来同步更新数据库**
  + **在同一个 Session 实例的缓存中, 数据库表中的每条记录只对应唯一的持久化对象**
* 删除对象(Removed)
  + 在数据库中没有和其 OID 对应的记录
  + 不再处于 Session 缓存中
  + 一般情况下, 应用程序不该再使用被删除的对象
* 游离对象(也叫”脱管”) （Detached）：
  + **OID 不为 null**
  + **不再处于 Session 缓存中**
  + 一般情况需下, 游离对象是由持久化对象转变过来的, 因此在数据库中可能还存在与它对应的记录
  1. 对象的状态转化图



4对象相互转换的方法，即session的核心方法

1. save()方法和persist()方法

A)save方法

\*使一个临时对象变为一个持久化对象

\*为对象分配ID

\* 在flush缓存时会发送一条SQL语句

\* 在save方法之前的ID是无效的

\*在save方法之后无法设置ID，因为持久化对象的ID是不能被修改的

B）persist方法

\* persist方法也发送INSERT语句，基本和save语句

C）persist() 和 save() 区别：

当对一个 OID 不为 Null 的对象执行 save() 方法时, 会把该对象以一个新的 oid 保存到数据库中; 但执行 persist() 方法时会抛出一个异常

1. load和get

都可以根据跟定的 OID 从数据库中加载一个持久化对象

Load方法是代理一个对象，当使用时，再从数据库中获取

区别

1. 执行get方法，会立即加载对象，而执行load方法，则不会立即执行查询操作，而返回一个代理对象（get是立即检索，load是延迟检索）
2. Load方法可能会抛出LazyInitializationException异常：在需要初始化代理对象之前已经关闭了Session
3. 若数据表中没有对应的记录，Session也没有被关闭

get返回null

load若不使用该对象的任何属性，没问题；若需要初始化，抛出异常

1. update方法

\* 1.若更新一个持久化对象，不需要显示的调用update方法，因为在调用Transaction的commit方法时，会先执行session的flush方法

\* 2.若更新一个游离对象，需要显示调用session的update方法。可以把一个游离对象变为持久化对象

\*

\* 需要注意的：

\* 1.无论要更新的游离对象和数据表的记录是否一致都会发送UPDATE语句

\* 让update方法不再盲目护法UPDATE语句的方法：在hdm.xml文件的class节点设置一个属性，select-before-update=true

\* 一般不设置该属性，除非是和触发器协同工作

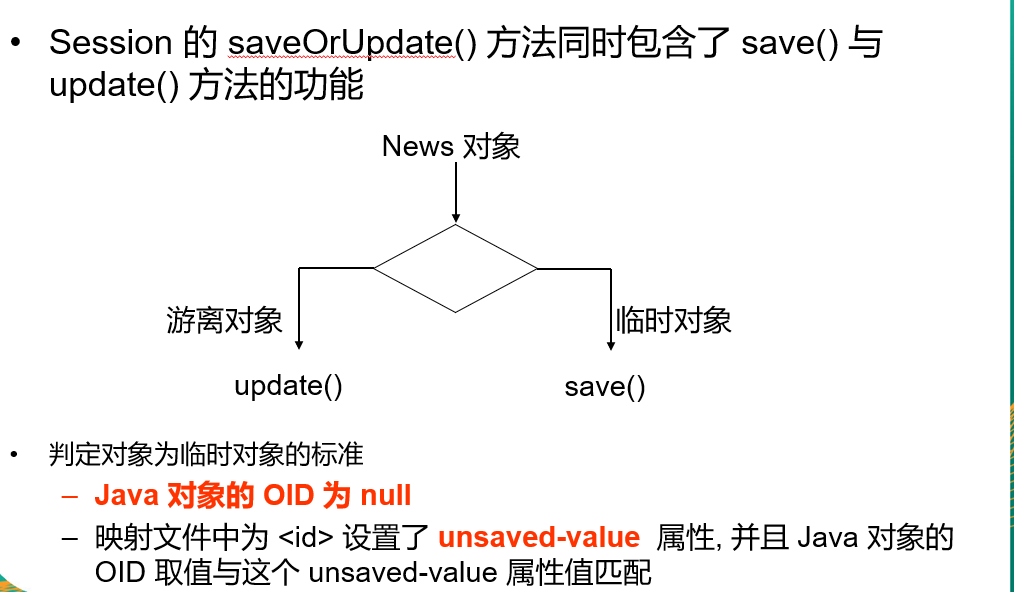
\* 2.当数据表中没有对应的记录，但还是调用了update方法，会抛出异常

\*

\* 3.当update（）方法关联一个游离对象时，如果在Session的缓存中已经存在相同的OID的持久化对象，会抛出异常

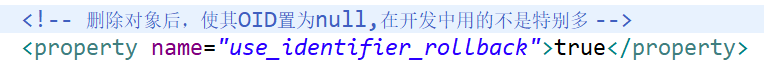
\* 因为在Session缓存中，不能有两个OID相同的持久化对象

④ saveOrUpdate方法



⑤ delete方法

* Session 的 delete() 方法既可以删除一个游离对象, 也可以删除一个持久化对象
* 若没有OID对应的对象，则抛出异常
* Session 的 delete() 方法处理过程
  + 计划执行一条 delete 语句
  + 把对象从 Session 缓存中删除, 该对象进入删除状态.
* Hibernate 的 cfg.xml 配置文件中有一个 **hibernate.use\_identifier\_rollback** 属性, 其默认值为 false, 若把它设为 true, 将改变 delete() 方法的运行行为: delete() 方法会把持久化对象或游离对象的 OID 设置为 null, 使它们变为临时对象



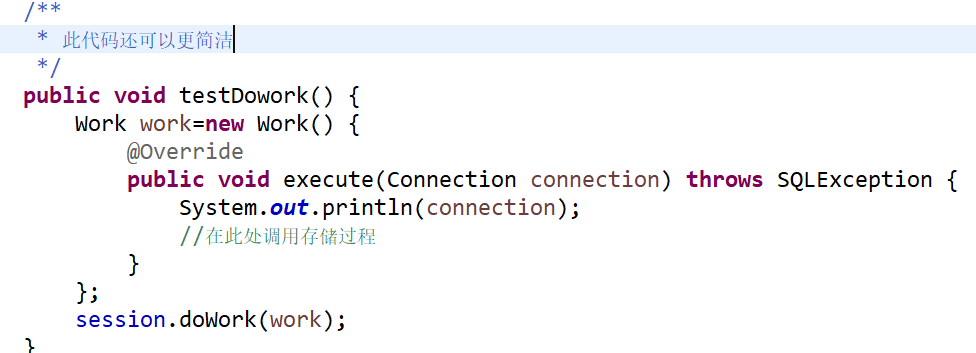
⑥evict方法

从session缓存中把指定的持久化对象移除

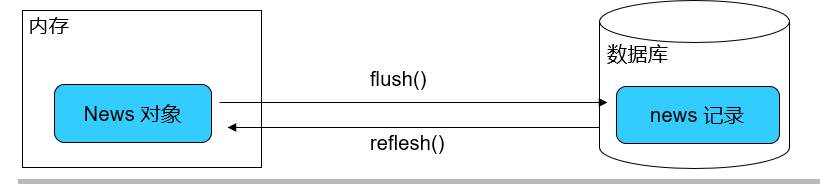
5.hibernate中调用存储过程（dowork方法）

* Session 的 doWork(Work) 方法用于执行 Work 对象指定的操作, 即调用 Work 对象的 execute() 方法. Session 会把当前使用的数据库连接传递给 execute() 方法.

使用方法：



5. Hibernate 与触发器协同工作

* Hibernate 与数据库中的触发器协同工作时, 会造成两类问题
  + 触发器使 Session 的缓存中的持久化对象与数据库中对应的数据不一致:触发器运行在数据库中, 它执行的操作对 Session 是透明的
  + Session 的 update() 方法盲目地激发触发器: 无论游离对象的属性是否发生变化, 都会执行 update 语句, 而 update 语句会激发数据库中相应的触发器
* 解决方案:
  + 在执行完 Session 的相关操作后, 立即调用 Session 的 flush() 和 refresh() 方法, 迫使 Session 的缓存与数据库同步(refresh() 方法重新从数据库中加载对象)
* 
  + 在映射文件的的 <class> 元素中设置 select-before-update 属性: 当 Session 的 update 或 saveOrUpdate() 方法更新一个游离对象时, 会先执行 Select 语句, 获得当前游离对象在数据库中的最新数据, 只有在不一致的情况下才会执行 update 语句