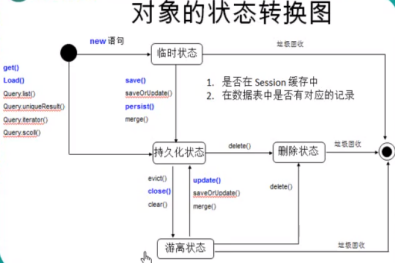
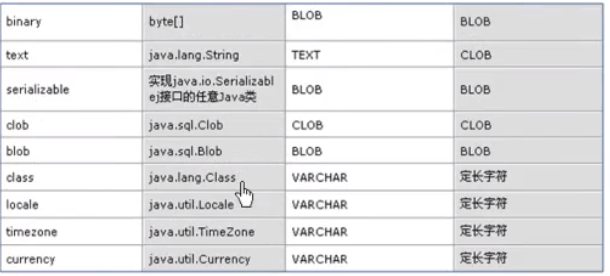
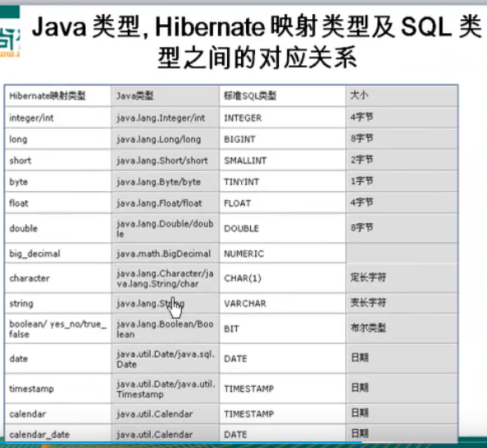
### Hibernate

1. 什么是hibernate
   1. 一个低侵入式框架
   2. 一个java领域的持久化框架
   3. 一个ORM框架
2. ORM（Object/Relation Mapping）:对象关系映射
   1. 思想：程序员可以吧对数据库的操作转化为对对象的操作
   2. 采用元数据来描述对象-关系映射细节
3. 创建持久化java类的要求
   1. 提供一个无参的构造器
   2. 提供一个标识属性
   3. 为类的持久化类字段声明访问方法
   4. 使用非final类
   5. 重写equals和hashCode方法
4. Configuration类负责管理hibernate的配置信息
   1. Hibernate运行的底层信息：数据库的url，用户名，密码，jdbc驱动类，数据库连接池等
   2. 持久化类与数据库表的映射关系（\*.hbm.xml文件）
5. 创建Configuration的两种方式
   1. 属性文件（hibernate.properties）：Configuration cfg = new Configuration();
   2. Xml文件（hibernate.cfg.xml）：Configuration cft = new Configuration().configure();
   3. Configuration的configure还支持带参数的访问：
      1. File file = new File(“simple.xml”);
      2. Configuration cft = new Configuration().configure(file);
6. Hbm2ddl.auto的几个取值
   1. Create：每次都会生成新的表
   2. Create-drop:sessionFactory一关闭就会把表删除
   3. Update(常用值)：不会删除原有的数据
   4. Validate：会跟数据库中的表进行比较，如果hbm.xml中的列不存在则抛出异常
7. Session：
   1. 提供了基本的保存，更新，删除和加载java对象的方法
   2. Session具有一个缓存，位于缓存中的对象成为持久化对象，它和数据库中的相关记录对应
   3. 站在持久化的角度，hibernate将对象分为4种状态：持久化状态，临时状态，游离状态，删除状态
      1. 临时状态（transient）：
         1. Oid通常为null
         2. 不处在session的缓存中
         3. 在数据库中没有对应的记录
      2. 持久化对象（persist）：
         1. Oid不为空
         2. 位于session缓存中
         3. 若在数据库中已经有和其相对应的记录，持久化对象和数据库中的相关记录对应
         4. Session在flush缓存时，会根据持久化对象的属性状态，来同步更新数据库
         5. 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
      3. 删除对象（removed）：
         1. 在数据库中没有和其oid对应的记录
         2. 不再处于session缓存中
         3. 一般情况下，应用程序不应该再使用被删除的对象
      4. 游离对象（detached）：
         1. Oid不为null
         2. 不再处于session缓存中
         3. 一般情况下，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其对应的数据
8. Flush:使数据库表中的记录和Session缓存中的对象的状态保存一致，为了保持一致则可能发送对应的sql语句
   1. 在tracsaction的commit() 方法中：先调用session的flush方法在提交事务
   2. Flush方法可能会发送sql语句，但不会提交事务，如果session缓存中的数据跟数据库中的数据是相同的，则不会发送sql语句
   3. 注意：在未进行提交事务的操作和显示调用flush时也可能会触发flush操作
      1. 执行HQL 或QBC查询，会先进行flush操作，以得到数据库的最新数据
      2. 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请求，因为save方法后，必须保证对象的id是存在的
9. Session按照缓存中对象的属性变化来同步更新数据库
10. Refresh方法：会强制发送select语句，以使session缓存中对象的状态和数据库表中对应的数据保持一致
11. Clear:清除缓存
12. Save方法
    1. 使一个对象变为持久化对象
    2. 为对象分配id
    3. 在flush缓存时会发送一条insert语句
    4. 在save方法之前的id是无效的
    5. 持久化对象的id是不能修改的
13. Persist方法
    1. 在调用persist方法之前若对象已经有id了，则不会执行insert，而抛出异常
14. Get 和 load 方法
    1. 执行get方法会立即加载对象
    2. 执行load方法，若不使用该对象，则不会立即执行操作，而返回一个代理对象
    3. Get是立即检索 load是延迟检索
    4. 若数据库表中没有对应数据，get返回null，load抛出异常
    5. Load方法可能会抛出懒加载异常 ，可能是因为在加载之前关闭了而session
15. update方法：
    1. 使一个游离对象转变为持久化对象，并且计划执行一条update语句
    2. 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，然后提交事务
    3. 更新一个游离对象，需要显示的调用session的update方法，可以把一个游离对象变为持久化对象
    4. 需要注意
       1. 无论更新的游离对象和数据库的记录是否一致，都会发送update语句，可以设置.hbm.xml 的class节点添加select-before-update=true,让跟新前往数据库中查询是否需要更新
       2. 若数据库中没有对应的记录，但还调用update方法，会抛出异常
       3. 当update方法关联一个游离对象时，如果session已存在一个oid相同的持久化对象，会抛异常，因为在session缓存中不能有两个oid相同的持久化对象



1. Delete方法执行完成后并不会立即删除，而是在事务提交后进行删除
2. Evict方法将一个持久化对象从session中移除
3. 调用存储过程使用work方法，以匿名实现类的方式调用
4. 每个hibernate配置文件对应一个Configuration对象
5. Native标识生成器
   1. Native标识生成器根据底层数据库对自动生成标识的支持能力，来选择使用identity，sequence，或hilo标识生成器，适合跨数据库平台开发
   2. Oid必须为long，int或short类型，不然会抛异常
6. Java类型，hibernate映射类型即sql类型之间的对应关系



1. 在标准sql中，date类型表示日期，time表示时间，timestamp类型表示时间戳，同时包含日期和时间信息
2. 映射继承关系
   1. 采用subclass元素的继承映射
      1. 父类和子类使用同一张表
      2. 需要在表中添加一列辨别者列
      3. 使用subclass来映射子类，使用class或subclass的discrimination-value属性指定辨别者列的值
      4. 所有子类定义的字段都不能有非空约束
   2. Joined-subclass
      1. 每个子类一张表
      2. 子类实例由父类表和子类表共同存储
      3. 每个子类使用key元素映射共有主键
      4. 子类增加的元素可以添加非空约束
   3. Union
3. 类级别的检索策略
   1. 立即检索：立即加载检索方法指定的对象
   2. 延迟检索：延迟加载检索方法指定的对象，在使用具体的属性时，再进行加载，注意出现懒加载异常
   3. 使用：如果程序加载一个对象的目的是为了访问它的属性，可以采取立即检索，如果只是为了获取它的引用，则使用延迟检索
   4. 只是针对load方法有效
4. Get方法和Query的list方法在类别总是立即检索
5. 检索方式
   1. 导航对象图检索方式
   2. OID检索方式
   3. HQL检索方式
   4. QBC检索方式
   5. 本地SQL检索方式
6. Schema:的值是连接数据库时的用户名
7. Oracle 插入之后记得commit提交事务
8. HQL迫切左外连接 ，不管是迫切左外连接还是迫切内连接加fetch会直接初始化集合，不会懒加载
   1. LEFT JOIN FETCH 关键字表示迫切左外连接检索策略
   2. List方法返回的集合中存放实体对象的引用
9. Hibernate:左外连接
   1. LEFT JOIN 关键字表示左外连接查询
   2. List() 方法返回的集合中存放的是对象数组类型
   3. 根据配置文件来决定Employee集合的检索策略
10. 左连接：取左边表的全部，右边的表按条件，符合的显示，不符合的显示null
11. 右连接：取右边表的全部，左边的表按条件，符合的显示，不符合的显示null
12. 内连接：取两张表条件都满足的部分
13. 关联级运行时的检索策略
    1. 如果在HQL语句中没有指定显示的检索策略，将使用映射文件配置的检索策略
    2. 如果希望HQL采用迫切左外连接策略，就必须在HQL语句中显示的指定他，
    3. HQL代码中的检索策略会覆盖映射文件中的检索策略
14. 缓存：介于应用程序和永久性数据存储源（硬盘上的文件或数据库），其作用是降低应用程序直接读写永久性数据源的频率，从而提高应用的效率
15. Hibernate中的两个级别的缓存
    1. 第一级别是session级别的缓存，它是属于事务范围内的缓存，这一级别是有hibernate管理的
    2. 第二级别是SessionFactory级别的缓存，它是属于进程范围内的缓存
16. 适合放入二级缓存中的数据
    1. 很少被修改
    2. 不是很重要的数据，允许出现偶尔的并发问题
17. 不适合放入二级缓存
    1. 经常被修改
    2. 财务数据，不允许出现并发问题
    3. 与其他应用程序共享的数据
18. 二级缓存的并发访问策略



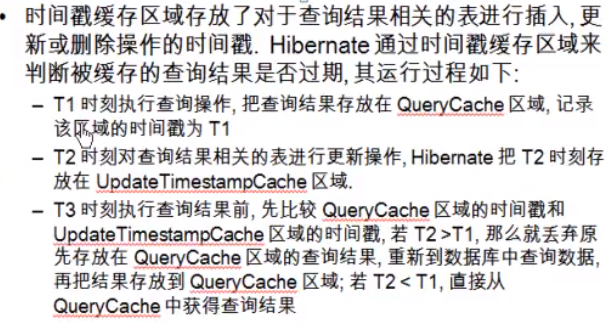
1. Hibernate下载

|  |
| --- |
| <https://jaist.dl.sourceforge.net/project/hibernate/hibernate-orm/5.4.23.Final/hibernate-release-5.4.23.Final.zip> |

1. 这个需要配置对

|  |
| --- |
| *<!--配置使用二级缓存的产品-->* <property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.internal.EhcacheRegionFactory</property> |
| class-cache：必须放在最后一行 |
| *<!--配置那个类使用二级缓存-->* <class-cache class="com.hibernate.hql.Employee" usage="read-write"></class-cache> |

1. 时间戳缓存区域



1. 管理session提供了三种管理session的方法
   1. Session对象的声明周期与本地线程绑定
   2. Session对象的声明周期与JTA事务绑定
   3. Hibrenate委托管理session对象的声明周期
2. 如何在应用层进行批量操作
   1. 通过session
   2. 通过hql
   3. 通过StatelessSession
   4. 通过JDBC API