# Hibernate知识点

1. Hibernate是轻量级JavaEE应用的持久层解决方案，是当前流行的ORM（对象/关系数据库映射）框架之一。
2. Hibernate倡导低侵入式的设计，直接采用了POJO（普通Java对象）作为PO（Persistent Object，持久化对象），PO = POJO + 映射文件。
3. PO可有如下三种状态：瞬态（如果PO实例从未与Session关联过，该PO实例出于瞬态状态）；持久化（如果PO实例与Session关联起来，且该实例对应到数据库记录，则該实例出于持久化状态）；脱管（如果PO实例曾经与Session关联过，但因为Session的关闭等原因，PO实例脱离了Session的管理，这种状态被称为脱管状态）。
4. 可以用过Session的load和get方法来加载一个持久化实例；Session的save、persist和saveOrUpdate方法可以将一个瞬态对象变为持久化对象；Session的update、merge和saveOrUpdate方法可以将一个脱管状态的对象重新变为持久化对象；而Session的delete方法可以将一个持久化对象变为脱管状态。
5. 如果没有匹配的数据库记录，load方法可能抛出HibernateException异常，如果我们在类映射文件中指定了延迟加载，则load方法会返回一个未初始化的代理对象（可以理解为持久化对象的替身），这个代理对象并没有装载数据记录，知道程序调用代理对象的某方法时，Hibernate才会去访问数据库。而get方法会立刻访问数据库，如果没有对应揭露，则返回null，而不是一个代理对象。
6. Hibernate之所以提供与save()功能几乎完全类似的persist()方法，一方面是为了照顾JPA的用法习惯，另一方面，使用save()方法报错持久化对象时，方法会返回该持久化对象的标志属性值-对应记录的主键值，而persist()方法没有任何返回值。
7. merge()和update()之间最大的区别是merge()不会持久化给定对象。当程序使用merge来保存程序对脱管对象所做的修改时，如果Session中存在相同持久化标识的持久化对象，merge方法里提供对象的姿态将覆盖掉原有持久化实例的状态。值得一提的是，merge方法返回的是持久化的对象。
8. 事务具备4个特性：原子性、一致性、隔离性和持续性。Hibernate的事务（Transaction对象）通过Session的begintransaction()方法显示打开，Hibernate自身并不提供事务控制行为（没有添加任何附加锁定行为），Hibernate底层直接使用JDBC连接、JTA资源或其他资源的事务。Hibernate只是对底层事务进行了抽象，让应用程序可以直接面向Hibernate事务编程。从编程角度来看，Hibernate事务由Session对象开启；从底层实现来看，Hibernate事务由TransactionFactory的实例来产生。
9. Hibernate的缓存包括Session的缓存和SessionFactory的缓存，其中SessionFactory的缓存又可以分为两类：内置缓存和外置缓存。Session的缓存是内置的，不能被[卸载](http://baike.baidu.com/view/386432.htm)，也被称为Hibernate的第[一级缓存](http://baike.baidu.com/view/16882.htm)。SessionFactory的内置缓存和Session的缓存在实现方式上比较相似，前者是SessionFactory对象的一些集合属性包含的数据，后者是指Session的一些集合属性包含的数据。SessionFactory的内置缓存中存放了映射元数据和预定义SQL语句，映射元数据是映射文件中数据的拷贝，而预定义SQL语句是在Hibernate初始化阶段根据映射元数据推导出来，SessionFactory的内置缓存是只读的，[应用程序](http://baike.baidu.com/view/330120.htm)不能修改缓存中的映射元数据和预定义SQL语句，因此SessionFactory不需要进行内置缓存与映射文件的同步。SessionFactory的外置缓存是一个可配置的[插件](http://baike.baidu.com/view/18979.htm)。在默认情况下，SessionFactory不会启用这个[插件](http://baike.baidu.com/view/18979.htm)。外置缓存的数据是数据库数据的拷贝，外置缓存的介质可以是内存或者硬盘。SessionFactory的外置缓存也被称为Hibernate的第二级缓存。
10. 当多个并发的事务同时访问持久化层的缓存的相同数据时，会引起并发问题，必须采用必要的事务隔离措施。在进程范围或集群范围的缓存，即第二级缓存，会出现并发问题。因此可以设定以下四种类型的并发访问策略，每一种策略对应一种事务隔离级别。**事务型**：仅仅在托管环境中适用。它提供了Repeatable Read事务隔离级别。对于经常被读但很少修改的数据，可以采用这种隔离类型，因为它可以防止脏读和不可重复读这类的并发问题。**读写型**：提供了Read Committed事务隔离级别。仅仅在非集群的环境中适用。对于经常被读但很少修改的数据，可以采用这种隔离类型，因为它可以防止脏读这类的并发问题。**非严格读写型**：不保证缓存与数据库中数据的一致性。如果存在两个事务同时访问缓存中相同数据的可能，必须为该数据配置一个很短的数据过期时间，从而尽量避免脏读。对于极少被修改，并且允许偶尔脏读的数据，可以采用这种并发访问策略。**只读型**：对于从来不会修改的数据，如参考数据，可以使用这种并发访问策略。事务型并发访问策略是事务隔离级别最高，只读型的隔离级别最低。事务隔离级别越高，并发性能就越低。
11. 适合存放到第二级缓存中的数据：1.很少被修改的数据；2.不是很重要的数据，允许出现偶尔并发的数据；3.不会被并发访问的数据；4.参考数据；

不适合存放到第二级缓存的数据：1.经常被修改的数据；2.财务数据，绝对不允许出现并发；3.与其他应用共享的数据。