Hibernate

1. Hibernate实现了面向对象的数据库编程。

比起jdbc来，在代码上书写简单了。Hibernate提供了缓存机制，这样可以使访问数据的效率提高很大。

jdbc的缺点

1、代码太繁琐了

2、不是面向对象的数据库操作

3、资源关闭的代码也很繁琐，每次都得打开、关闭

4、没有做到数据缓存

5、移植性比较差

优点：

因为是最低层的操作，所以效率比较高

hibernate

1、代码比较精简了

2、是面向对象的数据库操作

3、只需要关闭一个对象就可以了session

4、数据缓存 一级缓存 二级缓存 查询缓存

5、移植性比较好

缺点：

1、程序员不能控制sql语句的生成

hibernate中有一个hql

2、如果一个项目对sql语句的优化要求特别高，不适合用hibernate

3、如果一张表的数据量特别大，不适合用hibernate

2.hibernate的三大要素：

1、持久化类

对应数据库中的一张表

2、映射文件

把表与类建立关联

3、配置文件

链接数据库的信息

客户端：hibernate API

1. hibernate的基本流程:基本api操作流程
2. session的update时候快照区。
3. 主键生成机制：哪几种；increment,identity,assigned,uuid,sequence
4. Hibernate中对象的状态：临时状态（new出来的对象）、持久化状态（session的get、save、update）、脱管状态（session的clear、evict、close）。
5. 持久化类，最好实现对象的序列化，必须有一个默认的构造函数，主键的类型不用基本类型。
6. Hibernate会根据持久化状态的对象的id值发出相应的sql语句。
7. 级联操作，在配置文件中指定casecade=”” save-update,all,delete，主要用于在删除主表的记录时候，是否把从表的记录也相应的删掉。
8. Inverse通过对inverse属性的值决定是由双向关联的哪一方来维护表和表之间的关系。Inverse=false为主动方，负责维护关联关系。是否发出sql语句。只和外键有关联，如果让一的一方维护关系，要发出update语句，让多的一方维护不会发出。
9. Hql的迫切连接from Classes c inner join fetch c.students s

From Classes c left outer join fetch c.students s

1. Hibernate提高性能
2. 延迟加载lazy=true，extra在使用聚合函数时候，只加载函数数据；no-proxy/proxy相当于true。通过改变sql语句的发出时间来解决性能问题。
3. 抓取策略set的fetch属性select为默认值，会先加载classes再加载student，会按照id进行加载，n+1条sql语句；join左外连接，只发出一条sql语句；subselect子查询。
4. 一对多，让多的一方来维护关系
5. 一级缓存，持久化对象就位于session的缓存中，session级别的缓存，是属于事务范围的缓存。这一级别的缓存由 hibernate 管理的，一般情况下无需进行干预。利用session.get,load,save,update,getStudents都可以把相应的数据放入到session的缓存中，清空缓存用session.clear,session.evict。session.flush把缓存的数据同步到数据库中，session.refresh方法把数据库中的数据刷到缓存中。
6. 二级缓存，sessionfactory级别的缓存，是数据进程范围的缓存，hibernate启动的时候要启动二级缓存，关闭的时候二级缓存关闭，存放的是公有的一般不会改变的数据，hibernate本身不提供二级缓存的实现，需要引入第三方插件，把数据放入到二级缓存中的方法，session.get和session.load方法，二级缓存的应用范围是类和集合，如果想开启集合的二级缓存，需要开启针对集合的类的二级缓存。当二级缓存的数据量特别大时候，可采用一定的机制，缓存到硬盘上
7. 查询缓存是建立在二级缓存之上的缓存，默认是关闭的，需要开启查询缓存，<property name="cache.use\_query\_cache">true</property>

Query query = session.createQuery("");,利用query对象可以使用查询缓存

query.setCacheable(true)

query对象提供了把数据放入到查询缓存和提取数据的一个接口

1. ThreadLocal对象session缓存中存放私有数据，因为session从threadlocal中产生，所以session中的数据是线程安全的