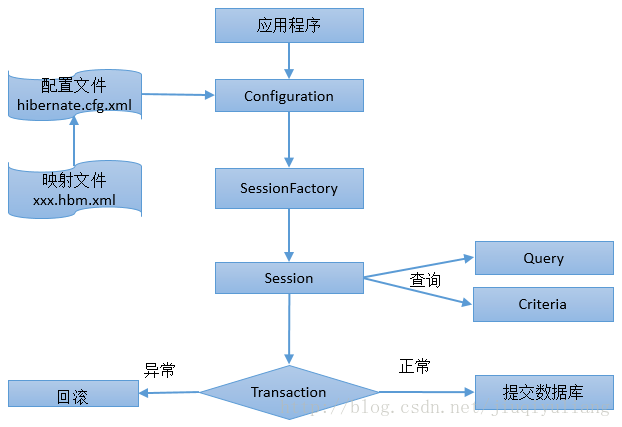
1. Hibernate工作原理
2. 工作原理图



1. Hibernate六大接口，两大配置文件
2. Configuration接口 ：负责配置并启动Hibernate
3. SessionFactory接口 ：负责初始化Hibernate
4. Session接口 ：负责持久化对象的CRUD操作
5. Transaction接口 ：负责事务
6. Query接口和Criteria接口 ：负责执行各种数据库查询
7. 实例代码

//1.创建一个SessionFactory对象

//读取并解析hibernate.cfg.xml配置文件

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

//hibernate.cfg.xml中的<mapping resource="com/xx/User.hbm.xml"/>读取并解析映射信息

configuration.addClass(Customer.class);

//创建SessionFactory

ServiceRegistry registry = new

StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

SessionFactory sessionFactory = configuration.buildSessionFactory(registry);

//2.创建一个Session对象

Session session = sessionFactory.openSession();

//3.开启一个事务

Transaction transaction = session.beginTransaction();

//4.执行操作

Customer customer = new Customer();

customer.setId(1);

customer.setName("John");

customer.setAddress("Beijing");

customer.setAge(13);

session.save(customer);

//5.提交事务

transaction.commit();

//6.关闭session

session.close();

sessionFactory.close();

1. Hibernate的优势

(1) 对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

(2) Hibernate是一个基于JDBC的主流持久化框架，很大程度地简化DAO层的编码工作

(3) hibernate使用Java反射机制，而不是字节码增强程序来实现透明性。

(4) hibernate的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

1. Hibernate的延迟加载 - get和load的区别
2. get方法

对于Hibernate get方法，Hibernate会确认一下该id对应的数据是否存在，首先在session缓存中查找，然后在二级缓存中查找，还没有就查询数据库，数据库中没有就返回null。主要要说明的一点就是在这个版本(hibernate3.2以上)中get方法也会查找二级缓存！

1. Hibernate load方法加载实体对象的时候，根据映射文件上类级别的lazy属性的配置(默认为true)，分情况讨论:
2. 若为true,则首先在Session缓存中查找，看看该id对应的对象是否存在，不存在则使用延迟加载，返回实体的代理类对象(该代理类为实体类的子类，由CGLIB动态生成)。等到具体使用该对象(除获取OID以外)的时候，再查询二级缓存和数据库，若仍没发现符合条件的记录，则会抛出一个ObjectNotFoundException。
3. 若为false,就跟Hibernateget方法查找顺序一样，只是最终若没发现符合条件的记录，则会抛出一个ObjectNotFoundException。
4. get和load的区别
5. 如果未能发现符合条件的记录，Hibernate get方法返回null，而load方法会抛出一个ObjectNotFoundException。
6. load方法可返回没有加载实体数据的代 理类实例，而get方法永远返回有实体数据的对象。
7. hibernate对于 load方法认为该数据在数据库中一定存在，可以放心的使用代理来延迟加载，如果在使用过程中发现了问题，只能抛异常；而对于get方 法，hibernate一定要获取到真实的数据，否则返回null。
8. Hibernate的缓存机制
9. 缓存机制的作用

Hibernate是一个持久层框架，为了降低应用程序对物理数据源访问的频次，从而提高应用程序的运行性能。缓存内的数据是对物理数据源中的数据的复制，应用程序在运行时从缓存读写数据，在特定的时刻或事件会同步缓存和物理数据源的数据。

1. 缓存的分类
2. 一级缓存

Hibernate一级缓存又称为“Session的缓存”，它是内置的，意思就是说，只要你使用hibernate就必须使用session缓存。由于Session对象的生命周期通常对应一个数据库事务或者一个应用事务，因此它的缓存是事务范围的缓存。在第一级缓存中，持久化类的每个实例都具有唯一的OID。

1. 二级缓存

Hibernate二级缓存又称为“SessionFactory的缓存”，由于SessionFactory对象的生命周期和应用程序的整个过程对应，因此Hibernate二级缓存是进程范围或者集群范围的缓存，有可能出现并发问题，因此需要采用适当的并发访问策略，该策略为被缓存的数据提供了事务隔离级别。第二级缓存是可选的，是一个可配置的插件，在默认情况下，SessionFactory不会启用这个插件。

1. 适合在二级缓存存放的数据：少修改，不重要，不会被并发访问

不适合存放的数据：经常被修改的数据，共享的数据，绝对不允许并发访问的数据

1. 管理缓存实例
2. 无论何时，我们在管理Hibernate缓存（Managing the caches）时，当你给save()、update()或saveOrUpdate()方法传递一个对象时，或使用load()、 get()、list()、iterate() 或scroll()方法获得一个对象时, 该对象都将被加入到Session的内部缓存中。
3. 当随后flush()方法被调用时，对象的状态会和数据库取得同步。 如果你不希望此同步操作发生，或者你正处理大量对象、需要对有效管理内存时，你可以调用evict() 方法，从一级缓存中去掉这些对象及其集合。
4. Hibernate的优化

(1) 使用双向一对多关联，不使用单向一对多

(2) 灵活使用单向一对多关联

(3) 不用一对一，用多对一取代

(4) 配置对象缓存，不使用集合缓存

(5) 一对多集合使用Bag,多对多集合使用Set

(6) 继承类使用显式多态

(7) 表字段要少，表关联不要怕多，有二级缓存撑腰

1. Hibernate和Mybatis缓存机制的比较
2. 相同点：Hibernate和Mybatis的二级缓存除了采用系统默认的缓存机制外，都可以通过实现你自己的缓存或为其他第三方缓存方案，创建适配器来完全覆盖缓存行为。
3. 不同点：
4. Hibernate的二级缓存配置**在SessionFactory生成的配置文件中进行详细配置**，然后在具体的表-对象映射中配置是那种缓存。
5. MyBatis的二级缓存配置都是**在每个具体的表-对象映射中进行详细配置**，这样针对不同的表可以自定义不同的缓存机制。并且Mybatis可以在命名空间中共享相同的缓存配置和实例，通过Cache-ref来实现。
6. 两者比较：
7. 因为Hibernate对查询对象有着良好的管理机制，用户无需关心SQL。所以在使用二级缓存时如果出现脏数据，系统会报出错误并提示。
8. MyBatis在这一方面，使用二级缓存时需要特别小心。如果不能完全确定数据更新操作的波及范围，避免Cache的盲目使用。否则，脏数据的出现会给系统的正常运行带来很大的隐患。