一、why（为什么要用Hibernate缓存？）

Hibernate是一个持久层框架，经常访问物理数据库。

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。

二、what（Hibernate缓存原理是怎样的？）Hibernate缓存包括两大类：Hibernate一级缓存和Hibernate二级缓存。

1.Hibernate一级缓存又称为“Session的缓存”。

Session内置不能被卸载，Session的缓存是事务范围的缓存（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）。

一级缓存中，持久化类的每个实例都具有唯一的OID。

2.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，默认下SessionFactory不会启用这个插件。

Hibernate提供了org.hibernate.cache.CacheProvider接口,它充当缓存插件与Hibernate之间的适配器。

什么样的数据适合存放到第二级缓存中？ 　　  
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) 常量数据 　　  
不适合存放到第二级缓存的数据？ 　　  
1) 经常被修改的数据 　　  
2) 绝对不允许出现并发访问的数据，如财务数据，绝对不允许出现并发 　　  
3) 与其他应用共享的数据。

3.Session的延迟加载实现要解决两个问题：正常关闭连接和确保请求中访问的是同一个session。

Hibernate session就是java.sql.Connection的一层高级封装，一个session对应了一个Connection。

http请求结束后正确的关闭session（过滤器实现了session的正常关闭）；延迟加载必须保证是同一个session（session绑定在ThreadLocal）。

4.Hibernate查找对象如何应用缓存？  
当Hibernate根据ID访问数据对象的时候，首先从Session一级缓存中查；

查不到，如果配置了二级缓存，那么从二级缓存中查；

如果都查不到，再查询数据库，把结果按照ID放入到缓存删除、更新、增加数据的时候，同时更新缓存。

5.一级缓存与二级缓存的对比图。

|  |  |  |
| --- | --- | --- |
|  | 一级缓存 | 二级缓存 |
| 存放数据的形式 | 相互关联的持久化对象 | 对象的散装数据 |
| 缓存的范围 | 事务范围，每个事务都拥有单独的一级缓存 | 进程范围或集群范围，缓存被同一个进程或集群范围内所有事务共享 |
| 并发访问策略 | 由于每个事务都拥有单独的一级缓存不会出现并发问题，因此无须提供并发访问策略 | 由于多个事务会同时访问二级缓存中的相同数据，因此必须提供适当的并发访问策略，来保证特定的事务隔离级别 |
| 数据过期策略 | 处于一级缓存中的对象永远不会过期，除非应用程序显示清空或者清空特定对象 | 必须提供数据过期策略，如基于内存的缓存中对象的最大数目，允许对象处于缓存中的最长时间，以及允许对象处于缓存中的最长空闲时间 |
| 物理介质 | 内存 | 内存和硬盘，对象的散装数据首先存放到基于内存的缓存中，当内存中对象的数目达到数据过期策略的maxElementsInMemory值，就会把其余的对象写入基于硬盘的缓存中 |
| 缓存软件实现 | 在Hibernate的Session的实现中包含 | 由第三方提供，Hibernate仅提供了缓存适配器，用于把特定的缓存插件集成到Hibernate中 |
| 启用缓存的方式 | 只要通过Session接口来执行保存，更新，删除，加载，查询，Hibernate就会启用一级缓存，对于批量操作，如不希望启用一级缓存，直接通过JDBCAPI来执行 | 用户可以再单个类或类的单个集合的粒度上配置第二级缓存，如果类的实例被经常读，但很少被修改，就可以考虑使用二级缓存，只有为某个类或集合配置了二级缓存，Hibernate在运行时才会把它的实例加入到二级缓存中 |
| 用户管理缓存的方式 | 一级缓存的物理介质为内存，由于内存的容量有限，必须通过恰当的检索策略和检索方式来限制加载对象的数目，Session的evit（）方法可以显示的清空缓存中特定对象，但不推荐 | 二级缓存的物理介质可以使内存和硬盘，因此第二级缓存可以存放大容量的数据，数据过期策略的maxElementsInMemory属性可以控制内存中的对象数目，管理二级缓存主要包括两个方面：选择需要使用第二级缓存的持久化类，设置合适的并发访问策略；选择缓存适配器，设置合适的数据过期策略。SessionFactory的evit（）方法也可以显示的清空缓存中特定对象，但不推荐 |

三、how（Hibernate的缓存机制如何应用？）

1.  一级缓存的管理:

evit(Object obj)  将指定的持久化对象从一级缓存中清除,释放对象所占用的内存资源,指定对象从持久化状态变为脱管状态,从而成为游离对象。

clear()  将一级缓存中的所有持久化对象清除,释放其占用的内存资源。

contains(Object obj) 判断指定的对象是否存在于一级缓存中。

flush() 刷新一级缓存区的内容,使之与数据库数据保持同步。

2.一级缓存应用： save()。当session对象调用save()方法保存一个对象后，该对象会被放入到session的缓存中。 get()和load()。当session对象调用get()或load()方法从数据库取出一个对象后，该对象也会被放入到session的缓存中。 使用HQL和QBC等从数据库中查询数据。

[复制代码](javascript:void(0);)

public class Client

{

public static void main(String[] args)

{

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction tx = null;

try

{

/\*开启一个事务\*/

tx = session.beginTransaction();

/\*从数据库中获取id="402881e534fa5a440134fa5a45340002"的Customer对象\*/

Customer customer1 = (Customer)session.get(Customer.class, "402881e534fa5a440134fa5a45340002");

System.out.println("customer.getUsername is"+customer1.getUsername());

/\*事务提交\*/

tx.commit();

System.out.println("-------------------------------------");

/\*开启一个新事务\*/

tx = session.beginTransaction();

/\*从数据库中获取id="402881e534fa5a440134fa5a45340002"的Customer对象\*/

Customer customer2 = (Customer)session.get(Customer.class, "402881e534fa5a440134fa5a45340002");

System.out.println("customer2.getUsername is"+customer2.getUsername());

/\*事务提交\*/

tx.commit();

System.out.println("-------------------------------------");

/\*比较两个get()方法获取的对象是否是同一个对象\*/

System.out.println("customer1 == customer2 result is "+(customer1==customer2));

}

catch (Exception e)

{

if(tx!=null)

{

tx.rollback();

}

}

finally

{

session.close();

}

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

结果

Hibernate:

select

customer0\_.id as id0\_0\_,

customer0\_.username as username0\_0\_,

customer0\_.balance as balance0\_0\_

from

customer customer0\_

where

customer0\_.id=?

customer.getUsername islisi

-------------------------------------

customer2.getUsername islisi

-------------------------------------

customer1 == customer2 result is true

[复制代码](javascript:void(0);)

输出结果中只包含了一条SELECT SQL语句，而且customer1 == customer2 result is true说明两个取出来的对象是同一个对象。其原理是：第一次调用get()方法， Hibernate先检索缓存中是否有该查找对象，发现没有，Hibernate发送SELECT语句到数据库中取出相应的对象，然后将该对象放入缓存中，以便下次使用，第二次调用get()方法，Hibernate先检索缓存中是否有该查找对象，发现正好有该查找对象，就从缓存中取出来，不再去数据库中检索。

3.二级缓存的管理：

evict(Class arg0, Serializable arg1)将某个类的指定ID的持久化对象从二级缓存中清除,释放对象所占用的资源。

sessionFactory.evict(Customer.class, new Integer(1));

evict(Class arg0)  将指定类的所有持久化对象从二级缓存中清除,释放其占用的内存资源。

sessionFactory.evict(Customer.class);

evictCollection(String arg0)  将指定类的所有持久化对象的指定集合从二级缓存中清除,释放其占用的内存资源。

sessionFactory.evictCollection("Customer.orders");

4.二级缓存的配置

常用的二级缓存插件

EHCache  org.hibernate.cache.EhCacheProvider

OSCache  org.hibernate.cache.OSCacheProvider

SwarmCahe  org.hibernate.cache.SwarmCacheProvider

JBossCache  org.hibernate.cache.TreeCacheProvider

[复制代码](javascript:void(0);)

<!-- EHCache的配置,hibernate.cfg.xml -->

<hibernate-configuration>

<session-factory>

<!-- 设置二级缓存插件EHCache的Provider类-->

<property name="hibernate.cache.provider\_class">

org.hibernate.cache.EhCacheProvider

</property>

<!-- 启动"查询缓存" -->

<property name="hibernate.cache.use\_query\_cache">

true

</property>

</session-factory>

</hibernate-configuration>

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

<!-- ehcache.xml -->

<?xml version="1.0" encoding="UTF-8"?>  
<ehcache>  
    <!--  
        缓存到硬盘的路径  
    -->  
    <diskStore path="d:/ehcache"></diskStore>  
    <!--  
        默认设置  
        maxElementsInMemory : 在內存中最大緩存的对象数量。  
        eternal : 缓存的对象是否永远不变。  
        timeToIdleSeconds ：可以操作对象的时间。  
        timeToLiveSeconds ：缓存中对象的生命周期，时间到后查询数据会从数据库中读取。  
        overflowToDisk ：内存满了，是否要缓存到硬盘。  
    -->  
    <defaultCache maxElementsInMemory="200" eternal="false"   
        timeToIdleSeconds="50" timeToLiveSeconds="60" overflowToDisk="true"></defaultCache>  
    <!--  
        指定缓存的对象。  
        下面出现的的属性覆盖上面出现的，没出现的继承上面的。  
    -->  
    <cache name="com.suxiaolei.hibernate.pojos.Order" maxElementsInMemory="200" eternal="false"   
        timeToIdleSeconds="50" timeToLiveSeconds="60" overflowToDisk="true"></cache>  
</ehcache>

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

<!-- \*.hbm.xml -->

<?xml version="1.0" encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >

<hibernate-mapping>

<class>

<!-- 设置该持久化类的二级缓存并发访问策略 read-only read-write nonstrict-read-write transactional-->

<cache usage="read-write"/>

</class>

</hibernate-mapping>

[复制代码](javascript:void(0);)

若存在一对多的关系，想要在在获取一方的时候将关联的多方缓存起来，需要在集合属性下添加<cache>子标签，这里需要将关联的对象的hbm文件中必须在存在<class>标签下也添加<cache>标签，不然Hibernate只会缓存OID。

[复制代码](javascript:void(0);)

<hibernate-mapping>

<class name="com.suxiaolei.hibernate.pojos.Customer" table="customer">

<!-- 主键设置 -->

<id name="id" type="string">

<column name="id"></column>

<generator class="uuid"></generator>

</id>

<!-- 属性设置 -->

<property name="username" column="username" type="string"></property>

<property name="balance" column="balance" type="integer"></property>

<set name="orders" inverse="true" cascade="all" lazy="false" fetch="join">

<cache usage="read-only"/>

<key column="customer\_id" ></key>

<one-to-many class="com.suxiaolei.hibernate.pojos.Order"/>

</set>

</class>

</hibernate-mapping>

[复制代码](javascript:void(0);)