缓存

缓存（Cache）简介

- ✓ 缓存(Cache)是计算机领域非常通用的概念。它介于应用程序和永久性数据存储源(如硬盘上的文件或者数据库)之间，其作用是降低应用程序直接读写永久性数据存储源的频率，从而提高应用的运行性能。缓存中的数据是数据存储源中数据的拷贝，应用程序在运行时直接读写缓存中的数据，只在某些特定时刻按照缓存中的数据来同步更新数据存储源。
- ✓ 缓存的物理介质通常是内存，而永久性数据存储源的物理介质通常是硬盘或磁盘，应用程序读写内在的速度显然比读写硬盘的速度快，如果缓存中存放的数据量非常大，也会用硬盘作为缓存的物理介质。
- ✓ Hibernate在查询数据时，首先到缓存中去查找，如果找到就直接使用，找不到的时候就会从物理数据源中检索，所以，把频繁使用的数据加载到缓存区后，就可以大大减少应用程序对物理数据源的访问，使得程序的运行性能明显的提升.

缓存的分类

- ✓ Hibernate的缓存分为：
  - ➢ 一级缓存：在Session级别的，在Session关闭的时候，一级缓存就失效了。
  - ➢ 二级缓存：在SessionFactory级别的，它可以使用不同的缓存实现，如EhCache、JBossCache、OsCache等。

一级缓存

- ✓ 一级缓存在Hibernate中是不可配置的部分
- ✓ save,update,saveOrUpdate,load,get,list,iterate, 这些方法都会将对象放在一级缓存中，一级缓存不能控制缓存的数量，所以要注意大批量操作数据时可能造成内存溢出；可以用方法清除缓存中的内容。

一级缓存的管理

- ✓ flush() 刷新一级缓存区的内容,使之与数据库数据保持同步
- ✓ 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

二级缓存

- ✓ 二级缓存在Hibernate中对应的即为 SessionFactory范围的缓存，通常来讲SessionFactory的生命周期和应用的生命周期相同
- ✓ 和一级缓存一样，二级缓存只缓存实体对象，不会缓存普通属性

配置二级缓存

- ✓ 二级缓存默认是开启的，修改hibernate.cfg.xml文件

```
<property name=" hibernate.cache.use_second_level_cache ">
        true
</property>
```

- ✓ 可以自己指定缓存产品提供商，很多缓存产品都是免费的，例如EhCache，修改hibernate.cfg.xml文件

```
<property name=" hibernate.cache.provider_class ">
                org.hibernate.cache.EhCacheProvider
</property>
```

ehcache.xml

- ✓ 如果使用ehcache，需要有一个配置文件ehcache.xml，这个文件放置在src根目录下

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ehcache.xsd">
        <defaultCache
            maxElementsInMemory="10000"
            eternal="false"
            timeToIdleSeconds="120"
            timeToLiveSeconds="120"
            overflowToDisk=“false"
            />
</ehcache>
```

配置属性解释

- ✓ maxElementsInMemory=“10000”
  - ➢ 缓存中最大存储的对象个数10000个
- ✓ eternal="false“
  - ➢ 设置为true表示永远不失效，false表示可以失效
- ✓ timeToLiveSeconds="120“
  - ➢ eternal=“false“的前提下设置缓存的对象的生命周期是120秒
- ✓ timeToIdleSeconds="120"
  - ➢ eternal=“false“的前提下设置间隔多少秒没有访问这个缓存对象就失效
- ✓ overflowToDisk=“false“
  - ➢ 如果设置为true，表示缓存对象超过最大限制就保存到磁盘上

缓存策略

- ✓ Read-write
  - 读写型策略
  - 用在读写比较频繁的场合
- ✓ Nostrict-read-write
  - 非严格读写型缓存
  - 用在偶尔进行读写的场合
- ✓ Read-only策略
  - 用在只读场合
- ✓ transactional （事务型）
  - 在Hibernate中，事务型缓存必须运行在JTA事务环境中。

配置实体支持二级缓存方法一

✓ 实体的映射文件中做显式的设置

```
<hibernate-mapping>
   <class name="com.Dept" table="DEPT" schema="HB2">

    <!-- 设置实体支持二级缓存，并且采取只读策略-->
    <cache usage="read-only"/>

      <id name="deptno" type="java.lang.Integer">
         <generator class="assigned" />
      </id>
          ……
   </class>
</hibernate-mapping>
```

配置实体支持二级缓存方法二

- ✓ 在hibernate.cfg.xml文件中配置哪些实体对象使用二级缓存，需要写在<mapping>标签后面，推荐

```
<hibernate-configuration>
    <session-factory>
        ……
        <mapping resource="com/Cource.hbm.xml" />

        <class-cache class="com.Dept" usage="read-only"/>

    </session-factory>
</hibernate-configuration>
```

释放二级缓存的方法

- ✓ SessionFacatoyr.evict(xxx) ：
  将某个特定的对象从内部缓存中清除，上述的XXX 为对象的Class类型，此举是清除所有此类型的对象
- ✓ SessionFacatoyr.evict(xxx,id)
  将某个特定的对象从内部缓存中清除，上述的XXX 为对象的Class类型，id是清除对象的oid，此举是清除单个对象
- ✓ 示例

```
//获得Session工厂对象
SessionFactory factory =
   HibernateSessionFactory.getSessionFactory();
//清除某一个Dept对象
factory.evict(Dept.class,40);
//清除所有Dept对象
factory.evict(Dept.class);
```

一级缓存和二级缓存的交互

- ✓ 可以设置当前session和二级缓存之间的交互，使用session.setCacheMode(模式);
- ✓ 模式有三种：
  - ➢ CacheMode.NORMAL - 从二级缓存中读、写数据（默认）
  - ➢ CacheMode.GET - 从二级缓存中读取数据，仅在数据更新时对二级缓存写数据。
  - ➢ CacheMode.PUT - 仅向二级缓存写数据，但不从二级缓存中读数据。
- ✓ 示例
  - ➢ session.setCacheMode(CacheMode.GET);

查询缓存

- ✓ 查询缓存是针对普通属性结果集的缓存，对实体对象的结果集只缓存id，查询缓存的生命周期，当前关联的表发生修改或是查询条件改变时，那么查询缓存生命周期结束，它不受一级缓存 和二级缓存 的生命周期的影响。

配置、使用查询缓存

- ✓ 默认情况下关闭，需要打开。查询缓存，对list/iterator这样的操作会起作用。。。。
- ✓ 可以使用：

```
<property name="hibernate.cache.use_query_cache">
true
</property>
```

- ✓ 来打开查询缓存，默认的情况下是关闭的。
- ✓ 所谓查询缓存，即让hibernate缓存list、iterator、createQuery等方法的查询结果集。如果没有打开查询缓存，hibernate将只缓存load方法获得的单个持久化对象。
- ✓ 在打开了查询缓存之后，需要注意，调用query.list()操作之前，必须显式调用query.setCachable(true)来标识某个查询使用缓存。

查询缓存和二级缓存

- ✓ 当只是用**Hibernate**查询缓存 而关闭 二级缓存的时候：
  第一：如果查询的是部分属性结果集： 那么当第二次查询的时候 就不会发出SQL 直接从Hibernate查询缓存中取数据
  第二：如果查询的是 实体结果集eg(from Student) 这个HQL 那么 查询出来的实体，首先Hibernate查询缓存 存放实体的ID ， 第二次查询，的时候 就到Hibernate查询缓存中取出ID 一条一条的到数据库查询 这样 将发出N 条SQL造成了SQL泛滥

- ✓ 当都开启**Hibernate**查询缓存和二级缓存的时候
  第一：如果查询的是部分属性结果集： 这个和上面只是用Hibernate查询缓存 而关闭 二级缓存的时候，一致 因为不涉及实体 不会用到二级缓存
  第二：如果查询的是 实体结果集， 这个HQL 那么 查询出来的实体，首先Hibernate查询缓存 存放实体的ID ， 第二次查询，的时候 就到Hibernate查询缓存中取出ID,拿到二级缓存区找数据，如果有数据 就不会发出SQL 如果都有 一条SQL 都不会发出 直接从二级缓存中取数据。