Hibernate\_day04总结

今日内容

* Hibernate的二级缓存
* Hibernate的查询缓存

## 上次课内容回顾:

Hibernate的检索方式:

\* 对象导航方式:

\* Customer customer = ...;

\* customer.getOrders();

\* 根据OID进行检索:

\* get();

\* load();

\* HQL:

\* session.createQuery(“HQL”);

\* 简单查询:

\* from Customer;

\* 排序查询:

\* from Cusomer order by 属性;

\* 条件查询:

\* 位置绑定:

\* from Customer where cname = ?;

\* 名称绑定:

\* from Customer where cname = :aaa;

\* 分页查询:

\* query.setFirstResult(0);

\* query.setMaxResults(10);

\* 聚集函数:

\* select count(\*) from Customer;

\* 多态查询:

\* 别名查询:

\* 构造方法:

\* 投影查询:

\* 多表查询:

\* 交叉连接:

\* 内连接:

\* 迫切内连接:

\* 左外连接:

\* 迫切左外连接:

\* 命名查询:

\* QBC:

\* session.createCriteria(Class clazz);

\* 简单查询:

\* 排序查询:

\* 条件查询:

\* 分页查询:

\* 离线条件查询:

\* SQL:

\* session.createSQLQuery(“SQL”);

\* Hibnerate的抓取策略:

\* 立即检索:

\* 延迟检索:

\* 在<class>设置lazy=”false”

\* 在持久化类上设置final.

\* 在调用方法的时候,初始化代理对象.

\* 延迟的时候:

\* 类级别延迟:

\* <class>标签上lazy

\* 关联级别延迟:

\* <set>/<many-to-one>/<one-to-one>

\* fetch和lazy配置:

\* <set>

\* fetch:

\* select :默认值:

\* join :使用迫切左外连接的形式.

\* subselect:使用子查询查询的关联对象.

\* lazy:

\* true :默认值

\* false :不采用延迟.

\* extra :及其懒惰.

\* <many-to-one>

\* fetch :

\* select :默认值.

\* join :使用迫切左外连接

\* lazy :

\* proxy :是否采用延迟看另一方<class>标签上配置的lazy是什么.

\* false :不采用延迟

\* no-proxy

\* batch-size:批量抓取.

\* Hibernate中事务与并发:

\* 事务管理:

\* 特性:

\* 隔离级别:

\* 丢失更新解决:

\* Hibernate中事务并发管理:

\* 设置隔离级别:

\* 使用悲观锁和乐观锁解决丢失更新:

\* 提供了本地线程绑定session:

\* Hibernate的二级缓存:

\* 二级缓存:SessionFactory级别缓存.可以在多个session之间共享数据的.

\* 二级缓存结构:

\* 类缓存区,集合缓存区,更新时间戳,查询缓冲区.

\* 二级缓存的适合放入的数据:

\* 不经常修改的,允许偶尔出现并发问题.

\* 二级缓存的配置:

\* 在Hibernate中开启二级缓存.

\* 配置二级缓存的提供商:

\* EHCache

\* 配置哪些类使用二级缓存:

\* 在映射文件中配置.

\* 在核心配置文件中配置(推荐).

## Hibernate的二级缓存:

### 类缓存区特点:

缓存的是对象的散装的数据.

### 集合缓存区的特点:

缓存的是对象的id.需要依赖类缓冲区的配置.

### 二级缓存数据到硬盘:

### 更新时间戳区域:

### 查询缓存:

比二级缓存功能更加强大,而且查询缓存必须依赖二级缓存.

二级缓存:对类/对象的缓存.

查询缓存:针对类中属性的缓存.

查询缓存的配置:

\* 配置查询缓存:

\* 前提是二级缓存已经配置完毕.

\* 在核心配置文件中:

<!-- 配置查询缓存 -->

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

\* 编写代码的时候:

query.setCacheable(true);

## Struts2和Hibernate的整合:

### SH的整合

#### 第一步:创建一个web项目:

#### 第二步:导入jar包:

Struts2开发包:

Hibernate开发包:

\* hibernate3.jar

\* lib/jpa/\*.jar

\* lib/required/\*.jar

\* mysql驱动

\* c3p0

引入配置文件:

\* struts.xml

\* hibernate.cfg.xml

\* log4j.properties

#### 第三步:创建包结构:

cn.itcast.vo

...

#### 第四步:创建实体类与映射:

#### 第五步:请求

customer\_findAll.action