1．Hibernate 的检索方式

①导航对象图检索 ②OID 检索 ③HQL 检索 ④QBC 检索 ⑤本地 SQL 检索

2． Forward 与 Global-Forward 的区别

答：Forward 是根据 Action return 的值找到对应的 JSP 页。当多个 Action 共同 return 同一个值时，可将这个 Forward 元素写在 Global-Forward 中。

3、在 Hibernate 应用中 Java 对象的状态有哪些？

1 临时状态(transient)：刚刚用 new 语句创建，还没有被持久化，不处于 Session 的缓存中。处于临时状态的 Java 对象被称为临时对象。

2 持久化状态（persistent）：已经被持久化，加入到 Session 的缓存中。处于持久化状态的 Java 对象被称为持久化对象。

3 游离状态（detached）：已经被持久化，但不再处于 Session 的缓存中。处于游离状态的 Java 对象被称为游离对象。

4、三种检索策略是什么，分别适用于哪种场合？

立即检索–

优点：对应用程序完全透明，缺点：select 语句数目多。适用：类级别。

延迟检索–

优点： 由应用程序决定加载哪些对象，可以避免执行多余的 select 语句以及避免加载不需要访问的对象，节省内存空间，提高检索效率。

缺点： 应用程序如果要访问游离态的代理类实例，必须保证它在持久化时已经被初始化。

适用： 一对多或多对多关联。应用程序不需要立即访问或者根本不会访问的对象。

迫切左外连接检索：

优点：对应用程序完全透明，不管对象处于持久化状态还是游离状态，应用程序都可以方便的从一个对象导航到另一个与它相关联的对象。使用了外连接，select 语句数目少。

缺点：可能会加载程序不许要访问的对象。复杂的数据库表连接形象检索性能。

适用：一对一或多对一关联。应用程序需要立即访问的对象。数据库系统具有良好的表连接性能。

5、ORM 解决的不匹配问题（域模型与关系模型之间存在的不匹配）

域模型是面向对象的，关系模型是面向关系的。

域模型中有继承关系，关系模型中不能直接表示继承关系。

域模型中有多对多关联关系，关系模型中通过连接表来表示多对多关联关系。

域模型中有双向关联关系，关系模型中只有单向参照关系，而且总是 many 参照 one 方。

域模型提倡精粒度模型，关系模型提倡粗粒度模型。

6、映射继承关系的三种方式？

（1）继承关系树的每个具体类对应一张表：在具体类对应的表中，不仅包含和具体类属性对应的字段，还包括与具体类的父类属性对应的字段。

（2）继承关系树的根类对应一张表：在根类对应的表中，不仅包括根类属性对应的字段，

还包括根类的所有子类属性对应的字段。

（3）继承关系树中的每个类对应一张表，每个表中只包括和这个类本身属性对应的字段，子类的表参照父类对应的表。

7、Session 的 find()方法以及 Query 接口的区别。

答案 Session 类的 find()方法以及 Query 接口都支持 HQL 检索方式。这两者的区别在于，前者只是执行一些简单 HQL 查询语句的便捷方法，它不具 有动态绑定参数的功能，而且在 Hibernate3.x 版本中，已经淘汰了 find()方法；而 Query 接口才是真正的 HQL 查询接口，它提供了以上列 出的各种查询功能。

8．hibernate 的配置文件(hibernate.properties)中 hibernate.show_sql=true/flase

在开发阶和测试段应设置为（）在发布阶段应设置为（）；true\false

9．映射一对多双向关联关系中设置 SET 元素:

请写出级联保存和更新、级联删除应在 set 元素中需要设置那些子元素？(标明关系、避面重复执行多余 SQL 语句)

Name cascade key column one-to-many inverse

10. list 要的实现类有那些？ 并按照存储结构、机制简单的说一下。

LinkedList、ArryList、Vector。

LinkedList 采用链表数据结构、ArryList 代表大小可变的数组。

Vector 与 ArryList 功能比较相似，区别在于 Vector 采用同步、ArryList 没有采用。

11：以下哪个不是 Hibernate 的检索方式：

A、导航对象图检索 B、OID 检索 C、ORM 检索

D、QBC 检索 E、本地 SQL 检索 F、HQL 检索

12.持久化类的类名是 Customer.java，写出相应的映射文件名 。

答案：Customer.hbm.xml

13.继承的三种方式

（1）继承关系树的每个具体类对应一张表：在具体类对应的表中，不仅包含和具体类属性对应的字段，还包括与具体类的父类属性对应的字段。

（2）继承关系树的根类对应一张表：在根类对应的表中，不仅包括根类属性对应的字段，还包括根类的所有子类属性对应的字段。

（3）继承关系树中的每个类对应一张表，每个表中只包括和这个类本身属性对应的字段，子类的表参照父类对应的表。

14. Session 接口是 Hibernate 应用使用最广泛的接口。Session 也被称为持久化管理器，它提供了和持久化相关的操作，如添加、更新、删除、加载和查询对象。

15.请简述 Session 的特点有哪些？

(1)不是线程安全的，因此在设计软件架构时，应该避免多个线程共享同一个 Session 实例。

(2)Session 实例是轻量级的，所谓轻量级是指它的创建和销毁不需要消耗太多的资源。这意味着在程序中可以经常创建或销毁 Session 对象，例如为每个客户请求分配单独的 Session 实例，或者为每个工作单元分配单独的 Session 实例。

(3)在 Session 中，每个数据库操作都是在一个事务(transaction)中进行的，这样就可以隔离开不同的操作（甚至包括只读操作）。

16、Hibernate 中采用 XML 文件来配置对象-关系映射的优点有那些？

答 案：Hibernate 既不会渗透到上层域模型中，也不会渗透到下层数据模型中。软件开发人员可以独立设计域模型，不必强迫遵守任何规范。数据库设计人员 可以独立设计数据模型，不必强迫遵守任何规范。对象-关系映射不依赖于任何程序代码，如果需要修改对象-关系映射，只需修改 XML 文件，不需要修改任何程 序，提高了软件的灵活性，并且使维护更加方便。

17、 叙述 Session 的缓存的作用

（1）减少访问数据库的频率。应用程序从内存中读取持久化对象的速度显然比到数据库中查询数据的速度快多了，因此 Session 的缓存可以提高数据访问的性能。

（2）保证缓存中的对象与数据库中的相关记录保持同步。当缓存中持久化对象的状态发生了变换，Session 并不会立即执行相关的 SQL 语句，这使得 Session 能够把几条相关的 SQL 语句合并为一条 SQL 语句，以便减少访问数据库的次数，从而提高应用程序的性能。

18、 多个事务并发运行时的并发问题有哪些？

第一类丢失更新；脏读；虚读；不可重复读；第二类丢失更新；

19、ORM 解决的不匹配问题（域模型与关系模型之间存在的不匹配）

答： 域模型是面向对象的，关系模型是面向关系的。域模型中有继承关系，关系模型中不能直接表示继承关系。域模型中有多对多关联关系，关系模型中通过连接表来表 示多对多关联关系。域模型中有双向关联关系，关系模型中只有单向参照关系，而且总是 many 参照 one 方。域模型提倡精粒度模型，关系模型提倡粗粒度模 型。

20、session 的清理和清空有什么区别？

session 清理缓存是指按照缓存中对象的状态的变化来同步更新数据库；清空是 session 关闭；

21 Hibernate 工作原理及为什么要用？
原理：
1.读取并解析配置文件
2.读取并解析映射信息，创建 SessionFactory
3.打开 Sesssion
4.创建事务 Transation

5.持久化操作
6.提交事务
7.关闭 Session
8.关闭 SesstionFactory
为什么要用：
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，是一个优秀的 ORM 实现。他很大程度的简化 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。

22． Hibernate 是如何延迟加载?
1. Hibernate2 延迟加载实现：a)实体对象 b)集合（Collection）
2. Hibernate3 提供了属性的延迟加载功能
当 Hibernate 在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

23．Hibernate 中怎样实现类之间的关系?(如：一对多、多对多的关系)
类与类之间的关系主要体现在表与表之间的关系进行操作，它们都市对对象进行操作，我们程序中把所有的表与类都映射在一起，它们通过配置文件中的 many-to-one、one-to-many、many-to-many、

24． 说下 Hibernate 的缓存机制
1. 内部缓存存在 Hibernate 中又叫一级缓存，属于应用事物级缓存
2. 二级缓存：
a) 应用及缓存
b) 分布式缓存
条件：数据不会被第三方修改、数据大小在可接受范围、数据更新频率低、同一数据被系统频繁使用、非
关键数据
c) 第三方缓存的实现

25

． Hibernate 的查询方式
Sql、Criteria,object comptosition
Hql：
1、
属性查询

2、
参数查询、命名参数查询
3、
关联查询
4、
分页查询
5、
统计函数

26.
如何优化 Hibernate?
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

补充：

Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式，如图 1 所示。

组成 Spring 框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。每个模块的功能如下：

☆ 核心容器：核心容器提供 Spring 框架的基本功能。核心容器的主要组件是 BeanFactory，它是工厂模式的实现。BeanFactory 使用控制反转 （IOC） 模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。

☆ Spring 上下文：Spring 上下文是一个配置文件，向 Spring 框架提供上下文信息。Spring 上下文包括企业服务，例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。

☆ Spring AOP：通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。所以，可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务

管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

☆ Spring DAO：JDBC DAO 抽象层提供了有意义的异常层次结构，可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理，并且极大地降低了需要编写 的异常代码数量(例如打开和关闭连接)。Spring DAO 的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。

☆ Spring ORM：Spring 框架插入了若干个 ORM 框架，从而提供了 ORM 的对象关系工具，其中包括 JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

☆ Spring Web 模块：Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文。所以，Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

☆ Spring MVC 框架：MVC 框架是一个全功能的构建 Web 应用程序的 MVC 实现。通过策略接口，MVC 框架变成为高度可配置的，MVC 容纳了大量视图技术，其中包括 JSP、Velocity、Tiles、iText 和 POI。

Spring 框架的功能可以用在任何 J2EE 服务器中，大多数功能也适用于不受管理的环境。Spring 的核心要点是：支持不绑定到特定 J2EE 服务的可重用业务和数据访问对象。毫无疑问，这样的对象可以在不同 J2EE 环境 （Web 或 EJB）、独立应用程序、测试环境之间重用。

IOC 和 AOP

控制反转模式（也称作依赖性介入）的基本概念是：不创建对象，但是描述创建它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器 （在 Spring 框架中是 IOC 容器） 负责将这些联系在一起。

在典型的 IOC 场景中，容器创建了所有对象，并设置必要的属性将它们连接在一起，决定什么时间调用方法。下表列出了 IOC 的一个实现模式。

Spring 框架的 IOC 容器采用类型 2 和类型 3 实现。

面向方面的编程

面向方面的编程，即 AOP，是一种编程技术，它允许程序员对横切关注点或横切典型的职责分界线的行为（例如日志和事务管理）进行模块化。AOP 的核心构造是方面，它将那些影响多个类的行为封装到可重用的模块中。

AOP 和 IOC 是补充性的技术，它们都运用模块化方式解决企业应用程序开发中的复杂问题。在典型的面向对象开发方式中，可能要将日志记录语句放在所有方法和 Java 类中才能实现日志功能。在 AOP 方式中，可以反过来将日志服务模块化，并以声明的方式将它们应用到需要日志的组件上。当然，优势就是 Java 类不需要知道日志服务的存在，也不需要考虑相关的代码。所以，用 Spring AOP 编写的应用程序代码是松散耦合的。

AOP 的功能完全集成到了 Spring 事务管理、日志和其他各种特性的上下文中。

IOC 容器

Spring 设计的核心是 org.springframework.beans 包，它的设计目标是与 JavaBean 组件一起使用。这个包通常不是由用户直接使用，而是由服务器将其用作其他多数功能的底层中介。下一个最高级抽象是 BeanFactory 接口，它是工厂设计模式的实现，允许通过名称创建和检索对象。BeanFactory 也可以管理对象之间的关系。

BeanFactory 支持两个对象模型。

□ 单态 模型提供了具有特定名称的对象的共享实例，可以在查询时对其进行检索。Singleton 是默认的也是最常用的对象模型。对于无状态服务对象很理想。

□ 原型 模型确保每次检索都会创建单独的对象。在每个用户都需要自己的对象时，原型模型最适合。

bean 工厂的概念是 Spring 作为 IOC 容器的基础。IOC 将处理事情的责任从应用程序代码转移到框架。正如我将在下一个示例中演示的那样，Spring 框架使用 JavaBean 属性和配置数据来指出必须设置的依赖关系。

BeanFactory 接口

因为 org.springframework.beans.factory.BeanFactory 是一个简单接口，所以可以针对各种底层存储方法实现。最常用的 BeanFactory 定义是 XmlBeanFactory，它根据 XML 文件中的定义装入 bean，如清单 1 所示。

清单 1. XmlBeanFactory

```
BeanFactory factory = new XMLBeanFactory(new
FileInputSteam(“mybean.xml”));
```

在 XML 文件中定义的 Bean 是被消极加载的，这意味在需要 bean 之前，bean 本身不会被初始化。要从 BeanFactory 检索 bean，只需调用 getBean() 方法，传入将要检索的 bean 的名称即可，如清单 2 所示。

清单 2. getBean()

```
MyBean mybean = (MyBean) factory.getBean(“mybean”);
```

每个 bean 的定义都可以是 POJO （用类名和 JavaBean 初始化属性定义） 或 FactoryBean。FactoryBean 接口为使用 Spring 框架构建的应用程序添加了一个间接的级别。

IOC 示例

理解控制反转最简单的方式就是看它的实际应用。在对由三部分组成的 Spring 系列 的第 1 部分进行总结时，我使用了一个示例，演示了如何通过 Spring IOC 容器注入应用程序的依赖关系（而不是将它们构建进来）。

我用开启在线信用帐户的用例作为起点。对于该实现，开启信用帐户要求用户与以下服务进行交互：

☆ 信用级别评定服务，查询用户的信用历史信息。

☆ 远程信息链接服务，插入客户信息，将客户信息与信用卡和银行信息连接起来，以进行自动借记（如果需要的话）。

☆ 电子邮件服务，向用户发送有关信用卡状态的电子邮件。

三个接口

对于这个示例，我假设服务已经存在，理想的情况是用松散耦合的方式把它们集成在一起。以下清单显示了三个服务的应用程序接口。

清单 3. CreditRatingInterface

```
public interface CreditRatingInterface {

public boolean getUserCreditHistoryInformation(ICustomer iCustomer);
```

```
}
```

清单 3 所示的信用级别评定接口提供了信用历史信息。它需要一个包含客户信息的 Customer 对象。该接口的实现是由 CreditRating 类提供的。

清单 4. CreditLinkingInterface

```
public interface CreditLinkingInterface {

public String getUrl();

public void setUrl(String url);

public void linkCreditBankAccount() throws Exception ;

}
```

信用链接接口将信用历史信息与银行信息（如果需要的话）连接在一起，并插入用户的信用卡信息。信用链接接口是一个远程服务，它的查询是通过 getUrl() 方法进行的。URL 由 Spring 框架的 bean 配置机制设置，我稍后会讨论它。该接口的实现是由 CreditLinking 类提供的。

清单 5. EmailInterface

```
public interface EmailInterface {

public void sendEmail(ICustomer iCustomer);
```

```
public String getFromEmail();

public void setFromEmail(String fromEmail) ;

public String getPassword();

public void setPassword(String password) ;

public String getSmtpHost() ;

public void setSmtpHost(String smtpHost);

public String getUserId() ;

public void setUserId(String userId);
```

==========

Hibernate 方面：

A 不会涉及到 2 级缓存以及 hql
1. Configuration
读取 hibernate.cfg.xml.并把.hbm.xml 文件交给 HbmBinder 做第一次处理,
HbmBinder 根据.hbm.xml 解析出 PersistentClass, Collection,然后在创建
SessionFactory 的时候,会对 Collection 做第 2 次处理塞入关联

2 PersistentClass
根据.hbm.xml 产生的描述要持久化的类的信息的类.主要的实例变量包括
List<Property> properties
Property 对象里的有个 Value 属性 value,通过 value 来描述该 property 和
数据库里的哪些列对应以及获得该 property 对应的 type

3 Value
主要分为
SimpleValue,Collection,Component,ToOne
SimpleValue 主要包括 Table 和 Columns 属性,用于描述简单属性或单主键
Collection 主要属性包括
collectionTable 表示 Collection 里面 element 对象所对应的 Table
key 表示 CollectionTable 里的哪几列和 Collection owner 所对应的表的主键做关联
element,描述了主表(referencingTable),从表的 EnityName,以及从表对应的
PersistentClass
Component 可以用来描述多主键,通过属性 properties 来表示

ToOne 包括被引用的属性名,被引用的实体名,columns,(被引用的属性名不能和 columns 同时设置),用于 OneToOne,ManyToOne

4 SessionFactory

在创建 SessionFactory 的时候,会根据 Configuration 里 Classes 和 Collections,创建 EntityPersister 和 CollectionPersister.
SessionFactory 会缓存这些 persisters.
EnityPersistenter 的 key 是 EntityName,
CollectionPersister 的 key 是 entityName+propertyName

5 EntityPersister

分为
SingleTableEntityPersister(一个实体一个表/一个类继承结构一个表)
通过一个字段做标识
JoinedSubclassEntityPersister(每个子类一个表)
UnionSubclassEntityPersister(每个具体类一个表)

6 CollectionPersister

封装对一个 Collection 的 crud 操作.
不过做 insert,update,delete,recreate 的时候,会判 Inverse 是否为 false.如果为 false 才会执行相应的操作,表示是由 Collection 维护关系.
如果 Inverse==true,表示关联关系是由多端来维护(即直接通过操作 Collection 里的 element 来维护,而不是通过操作 Collection 来维护)
则该 CollectionPersister 不会做任何操作

7 Type

主要包括对 SqlTypes 的封装,以及 CollectionType,EntityType (ManyToOneType,OneToOneType),主要接口有 nullSafeGet(从 ResultSet 拼装出对象),nullSafeSet(给 PreparedStatement setParameter).
可以通过 Value.getType()获得 Type
对于 EntityType,CollectionType,就是通过 resolve 方法从 ResultSet 中拼装出对象
EntityType 的 nullSafeSet,就是获取 One 端对象的主键所对应的 Type 进行 nullSafeSet
CollectionType 没有实现 nullSafeSet,通过保存时的 Cascade 或者 CollectionPersister,将 Collection 的 Element 一个一个的 set
也可以自定义 Type,实现 UserType 接口

具体见

http://docs.huihoo.com/framework/hibernate/reference-v3_zh-cn/inheritance.html

封装对一个 entity 的 crud 操作,在创建 EntityPersister 实例过程中,会产生 crud 的 sql,可以在以后的操作提高效率。不过如果是 DynamicInsert, DynamicUpdate,则会根据对象修改的属性动态的生成 sql
DAS 不支持类继承的映射,因此只会用到 SingleTableEntityPersister

8 StatefulPersistenceContext
StatefulPersistenceContext 和 SessionImpl 是一一对应的，会缓存通过 SessionImpl 操作过的对象,包括 entity 和 collection.主要属性有
EntitiesByKey key=EntityKey, value=entity.
(EntityKey=id+EntityPersister+EntityMode entityMode))
entityEntries key=entity, value=EntityEntry.
EntityEntry 用于描述一个对象的持久化状态,如 DELETED,MANAGED 等等

CollectionsByKey
key=CollectionKey, value=PersistentCollection
CollectionKey=CollectionPersister+key+EntityMode, 这个 Key 是通过 Collection.value.getType.nullSafeGet()得到的
PersistentCollection 是 hibernate 对于 Collection 的封装，主要用于实现延迟加载
collectionEntries key=PersistentCollection,value=CollectionEntry
StatefulPersistenceContext 的主要用途可以实现对象之间的关联关系的设置,动态的更新,以及对缓存的数据无需显示调用 save,update,delete 的方法就可以实现这些操作,是因为在 Transaction.commit()的时候会调用 session.flush(),会保证内存对象状态和数据库的一致性

9 Cascade, CascadeStyle, CascadingAction
在 Cascade 执行级联操作的时候,会通过 CascadeStyle.doCascade(CascadingAction)来判断是否可以执行 cascade,并且当要保存的对象有外键约束的关联对象时候会通过 ForeignKeyDirection 来判断是应该在保存该对象之前要保存关联对象还是在保存该对象之后再保存关联对象

10 Loader, CriteriaQueryTranslator, Criteria, QueryParameters, CriteriaJoinWalker
用于 Criteria api 对实体的查询
Criteria 是一个查询 entity 的 api。可以设置类似 where 条件的表达式, Select 字段,order 等等
当使用 Criteria 查询时,首先会创建 CriteriaLoader,CriteriaLoader 会通过 CriteriaQueryTranslator 从 Criteria 中得到查询参数 QueryParameters,通过 CriteriaJoinWalker 把 Criteria 变成 sql,然后执行查询

11 ActionQueue
当调用 session 对实体进行 insert,update,delete 的时候，只是会创建相应的 action 放入 ActionQueue,然后在 session.flush()的时候才会真正操作数据库

ActionQueue 的执行顺序:
executeActions(insertions);executeActions(updates);executeActions(collectionRemovals);executeActions(collectionUpdates);executeActions(collectionCreations);executeActions(deletions);

12 Tuplizer
用于根据 entityname 实例化出对象,以及 set/get property
hibernate 内置的有 PojoEntityTuplizer，Dom4jEntityTuplizer

DAS 是在 hibernate 基础之上实现了 DataObject 的持久化,支持 DataObject 对象之间的关联,延迟加载,级联,控制反转,不支持 DataObject 的继承

1. 新实现了一个 SDOEntityTuplizer
支持创建 DataObject,以及 set/get DataObject 的属性
SDOEntityTuplizer 实现抽象类中规定的如下方法

SDOEntityTuplizer 类的 buildInstantiator 方法用来根据 mapping 信息来为相应的实体建立不同的构造器，我们用 SDOEntityInstantiator 来实现 SDO 的构造器：

2 其中 SDOComponentInstantiator 是用来当实体有复合主键用于实例化表示主键的类

3PropertyAccessor 是 Hibernate 中定义的一个接口，用来表达在访问 mapping 的实体对象的属性的时候使用的”属性访问器”。它有两个相关的接口：Getter 接口和 Setter 接口。
DASPropertyAccessor 的内部结构如下图所示：
其中 getGetter 方法和 getSetter 方法是 PropertyAccessor 接口定的规格。DASGetter 实现 Getter 接口，DASSetter 类实现 Setter 接口。

4 自定义 type,支持 blob,clob 的存储
BlobFileType:从一个文件路径读取文件存入数据库 blob 字段.从数据库读取 blob 字段,生成文件放在临时目录,返回路径
BlobByteArrayType:把一个 byte[]数组存入 blob 字段.从数据库读取 blob 字段放入 byte[]
ClobFileType:从一个文件路径读取文件存入数据库 clob 字段.从数据库读取 clob 字段,生成文件放在临时目录,返回路径
ClobStringType:把一个 String 存入 blob 字段.从数据库读取 blob 字段放入 String

5 对查询实体的支持
增加一个查询实体定义文件.dbquery. 因为查询实体没有唯一标识，所以默认的 id 的是“$queryEntityId$”.用户也可以自己指定一个列做唯一标识.
在 Loader.prepareQueryStatement()里增加转换带查询实体的 Sql.
SelectFragment.addColumn() 做了判断，如果列名是“$queryEntityId$”,查询的 column 就替换成‘queryEntityId’常量.

在 Loader.getRow() 增加判断如果 key.getIdentifier()是
queryEntityId 的话,则不做检查,看内存是否存在.
AbstractEntityPersister.getDatabaseSnapshot()里修改生成的查询
语句,如果有 QueryEntity,则用定义的 sql 替换 QueryEntity