Hibernate 工作原理及为什么要使用 Hibernate？

工作原理：

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 Session
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

为什么要使用 Hibernate（即它的优点）：

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 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。

2， Hibernate 中 get 和 load 方法的区别

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；

hibernate 对于 get 方法，hibernate 一定要获取到真实的数据，否则返回 null。

具体介绍：

1. 对于 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。

2. 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：

(1)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。

(2)若为 false,就跟 get 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。

3，Hibernate 是如何延迟加载?

Hibernate3 提供了属性的延迟加载功能。当 Hibernate 在查询数据的时候，数据并没有存在于内存之中，而是当程序真正对数据的操作时，对象才存在于内存中，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。

4，Hibernate 中怎样实现类之间的关系?

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，它们都是对对象进行操作，我们在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 进行操作。

5，Hibernate 中的 update()和 saveOrUpdate()的区别？

saveOrUpdate()：

1，如果对象已经在本 session 中持久化了，不做任何事

2，如果另一个与本 session 关联的对象拥有相同的持久化标识(identifier)，抛出一个异常

3，如果对象没有持久化标识(identifier)属性，对其调用 save()

4，如果对象的持久标识(identifier)表明其是一个新实例化的对象，对其调用 save()

5，如果对象是附带版本信息的（通过<version>或 <timestamp>）并且版本属性的值表明其是一个新实例化的对象，调用 save()。否则 update() 这个对象。

update() ：

是将一个游离状态的实体对象直接更新。

6，说说 Hibernate 的缓存机制。

1. 一级缓存：内部缓存存在 Hibernate 中，属于应用事物级缓存。

2. 二级缓存：应用级缓存、 分布式缓存。

使用场景：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

3.引入第三方缓存（如 ehcache 等）。

7，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

8，谈谈 hibernate 的延迟加载和 openSessionInView

延迟加载要在 session 范围内，用到的时候再加载；

opensessioninview 是在 web 层写了一个 filter 来打开和关闭 session，这样就表示在一次 request 过程中 session 一直开着，保证了延迟加载在 session 中的这个前提。

9，简要阐述 struts2 的工作流程

1、客户端浏览器发出 HTTP 请求。
2、根据 web.xml 配置，该请求被 FilterDispatcher 接收。
3、根据 struts.xml 配置，找到需要调用的 Action 类和方法，并通过 IoC 方式，将值注入给 Aciton。
4、Action 调用业务逻辑组件处理业务逻辑，这一步包含表单验证。
5、Action 执行完毕，根据 struts.xml 中的配置找到对应的返回结果 result，并跳转到相应页面。
6、返回 HTTP 响应到客户端浏览器。

10，说下 Struts 的设计模式

MVC 模式

1，web 应用程序启动时就会加载并初始化 ActionServlet；

2，用户提交表单时，一个配置好的 ActionForm 对象被创建，并被填入表单相应的数 据；

3，ActionServlet 根据 Struts-config.xml 文件配置好的设置决定是否需要表单验证，如果需要就调用 ActionForm 的 Validate()验证后选择将请求发送到哪个 Action，如果 Action 不存在，ActionServlet 会先创建这个对象，然后调用 Action 的 execute（）方法；

4，Execute（）从 ActionForm 对象中获取数据，完成业务逻辑，返回一个 ActionForward 对 象，ActionServlet 再把客户请求转发给 ActionForward 对象指定的 jsp 组件；

5，ActionForward 对象指定的 jsp 生成动 态的网页，返回给客户。

11，Struts 的优缺点

优点：

1. 实现 MVC 模式，结构清晰,使开发者只关注业务逻辑的实现.

2. 有丰富的 tag 可以用 ,Struts 的标记库(Taglib)，如能灵活动用，则能大大提高开发效率。另外，就目前国内的 JSP 开发者而言，除了使用 JSP 自带的常用标记外，很少开发自己的标记，或许 Struts 是一个很好的起点。

3. 页面导航.页面导航将是今后的一个发展方向，事实上，这样做，使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。

4. 提供 Exception 处理机制 .

5. 数据库链接池管理

6. 支持 I18N

缺点：

1. 转到展示层时，需要配置 forward，每一次转到展示层，相信大多数都是直接转到 jsp，而涉及到转向， 需要配置 forward，如果有十个展示层的 jsp，需要配置十次 struts，而且还不包括有时候目录、文件变更，需要重新修改 forward，注意，每次修改配置之后，要求重新部署整个项目，而 tomcate 这样的服务器，还必须重新启动服务器，如果业务变更复杂频繁的系统，这样的操作简单不可想象。现在就是这样，几十上百个人同时在线使用我们的系统，大家可以想象一下，我的烦恼有多大。

2. Struts 的 Action 必需是 thread - safe 方式，它仅仅允许一个实例去处理所有的请求。所以 action 用到的所有的资源都必需统一同步，这个就引起了线程安全的问题。

3. 测试不方便. Struts 的每个 Action 都同 Web 层耦合在一起，这样它的测试依赖于 Web 容器，单元测试也很难实现。不过有一个 Junit 的扩展工具 Struts TestCase 可以实现它的单元测试。

4. 类型的转换. Struts 的 FormBean 把所有的数据都作为 String 类型，它可以使用工具 Commons-Beanutils 进行类型转化。但它的转化都是在 Class 级别，而且转化的类型是不可配置的。类型转化时的错误信息返回给用户也是非常困难的。

5. 对 Servlet 的依赖性过强. Struts 处理 Action 时必需要依赖 ServletRequest 和 ServletResponse，所有它摆脱不了 Servlet 容器。

6. 前端表达式语言方面.Struts 集成了 JSTL，所以它主要使用 JSTL 的表达式语言来获取数据。可是 JSTL 的表达式语言在 Collection 和索引属性方面处理显得很弱。

7. 对 Action 执行的控制困难. Struts 创建一个 Action，如果想控制它的执行顺序将会非常困难。甚至你要重新去写 Servlet 来实现你的这个功能需求。

8. 对 Action 执行前和后的处理. Struts 处理 Action 的时候是基于 class 的 hierarchies，很难在 action 处理前和后进行操作。

9. 对事件支持不够. 在 struts 中，实际是一个表单 Form 对应一个 Action 类(或 DispatchAction)，换一句话说：在 Struts 中实际是一个表单只能 对应一个事件，struts 这种事件方式称为 applicationevent，application event 和 component event 相比是一种粗粒度的事件。

12，为什么要使用 spring（即优点）？

1. Spring 能有效地组织你的中间层对象，不管你是否选择使用了 EJB。如果你仅仅使用了 Struts 或其他为 J2EE 的 API 特制的 framework，Spring 致力于解决剩下的问题。

2. Spring 能消除在许多工程中常见的对 Singleton 的过多使用。过多使用 Singleton 降低了系统的可测试性和面向对象的程度。

3. 通过一种在不同应用程序和项目间一致的方法来处理配置文件，Spring 能消除各种各样自定义格式的属性文件的需要。曾经对某个类要寻找的是哪个魔法般的 属性项或系统属性感到不解，为此不得不去读 Javadoc 甚至源编码？有了 Spring，你仅仅需要看看类的 JavaBean 属性。Inversion ofControl 的使用（在下面讨论）帮助完成了这种简化。

4.通过把对接口编程而不是对类编程的代价几乎减少到没有，Spring 能够促进养成好的编程习惯。

5. Spring 被设计为让使用它创建的应用尽可能少的依赖于他的 APIs。在 Spring 应用中的大多数业务对象没有依赖于 Spring。

6. 使用 Spring 构建的应用程序易于单元测试。

7.Spring 能使 EJB 的使用成为一个实现选择,而不是应用架构的必然选择。你能选择用 POJOs 或 local EJBs 来实现业务接口，却不会影响调用代码。

8. Spring 帮助你解决许多问题而无需使用 EJB。Spring 能提供一种 EJB 的替换物，它们适用于许多 web 应用。例如，Spring 能使用 AOP 提供声明性事务管理而不通过 EJB 容器，如果你仅仅需要与单个数据库打交道，甚至不需要一个 JTA 实现。

9. Spring 为数据存取提供了一个一致的框架,不论是使用的是 JDBC 还是 O/R mapping 产品（如 Hibernate）。

13，列举一下你知道的实现 spring 事务的几种方式

(1)，编程式事务管理：需要手动编写代码，在实际开发中很少使用,

(2)，基于 TransactionProxyFactoryBean 的声明式事务管理，需要为每个进行事务管理的类做相应配置

(3)，基于 AspectJ 的 XML 的声明式事务管理，不需要改动类，在 XML 文件中配置好即可

(4)，基于注解的声明式事务管理，配置简单，需要在业务层类中添加注解

14，谈谈 spring 事务的隔离级别和传播行为

隔离级别：

- DEFAULT 使用数据库默认的隔离级别
- READ_UNCOMMITTED 会出现脏读，不可重复读和幻影读问题
- READ_COMMITTED 会出现重复读和幻影读
- REPEATABLE_READ 会出现幻影读

- SERIALIZABLE 最安全，但是代价最大，性能影响极其严重

和传播行：

- REQUIRED 存在事务就融入该事务，不存在就创建事务
- SUPPORTS 存在事务就融入事务，不存在则不创建事务
- MANDATORY 存在事务则融入该事务，不存在，抛异常
- REQUIRES_NEW 总是创建新事务
- NOT_SUPPORTED 存在事务则挂起，一直执行非事务操作
- NEVER 总是执行非事务，如果当前存在事务则抛异常
- NESTED 嵌入式事务

15，什么是 DI 机制？

依赖注入（Dependecy Injection）和控制反转（Inversion of Control）是同一个概念，具体的讲：当某个角色需要另外一个角色协助的时候，在传统的程序设计过程中，通常由调用者来创建被调用者的实例。

但在 spring 中 创建被调用者的工作不再由调用者来完成，因此称为控制反转。创建被调用者的工作由 spring 来完成，然后注入调用者

因此也称为依赖注入。

spring 以动态灵活的方式来管理对象 ， 注入的两种方式，设置注入和构造注入。

设置注入的优点：直观，自然

构造注入的优点：可以在构造器中决定依赖关系的顺序。

16，什么是 AOP？

面向切面编程（AOP）完善 spring 的依赖注入（DI），

面向切面编程在 spring 中主要表现为两个方面 ：

1.面向切面编程提供声明式事务管理

2.spring 支持用户自定义的切面