Hibernate 是什么?

Hibernate 是一个开放源代码的对象关系映射框架，做数据持久化的工具，它对 JDBC 进行了非常轻量级的对象封装,使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。

Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序实用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

优点:

a.Hibernate 使用 Java 反射机制而不是字节码增强程序来实现透明性。

b.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。

c.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。

Hibernate 工作原理及为什么要用?

1. 读取并解析配置文件，Configuration config = new Configuration().configure();
2. 读取并解析映射信息，创建 SessionFactory factory = config.buildSessionFactory();
3. 打开 Sesssion，Session session = factory.openSession();
4. 创建事务 Transation，Transaction tran = session.beginTransaction();
5. 持久化操作
6. 提交事务
7. 关闭 Session
8. 关闭 SesstionFactory

为什么要用:

* 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。
* 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
* hibernate 使用 Java 反射机制，而不是字节码增强程序来实现透明性。
* 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。

Hibernate 是如何延迟加载?

hibernate 中主要是通过代理（proxy）机制来实现延迟加载。它的具体过程：Hibernate 从数据库获取某一个对象数据时、获取某一个对象的集合属性值时，或获取某一个对象所关联的另一个对象时，由于没有使用该对象的数据，hibernate 并不是数据库加载真正的数据，而只是为该对象创建一个代理对象来代表这个对象，这个对象上的所有属性都是默认值；只有在真正需要使用该对象的数据时才创建这个真实对象，真正从数据库中加载它的数据，这样在某些情况下，就可以提高查询效率。

Hibernate 中默认采用延迟加载的情况主要有以下几种

1 当调用 session 的 load()加载一个实体时，会采用延迟加载，而 get 不会。

2 当 session 加载某个实体时，会对这个实体中的集合属性值采用延迟加载

3 当 session 加载某个实体时，会对这个实体所有单端关联的另一个实体对象采用延迟加载。

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，它们都是对对象进行操作，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

说下 Hibernate 的缓存机制

Hibernate 缓存的作用：

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

Hibernate 缓存分类：

Hibernate 缓存包括两大类：Hibernate 一级缓存和 Hibernate 二级缓存

Hibernate 一级缓存又称为“Session 的缓存”，它是内置的，不能被卸载（不能被卸载的意思就是这种缓存不具有可选性，必须有的功能，不可以取消 session 缓存）。由于 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,因此它的缓存是事务范围的缓存。第一级缓存是必需的，不允许而且事实上也无法卸除。在第一级缓存中，持久化类的每个实例都具有唯一的对象标识符。

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，在默认情况下，SessionFactory 不会启用这个插件。

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。

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

无论何时，我们在管理 Hibernate 缓存时，当你给 save()、update()或 saveOrUpdate()方法传递一个对象时，或使用 load()、 get()、list()、iterate() 或 scroll()方法获得一个对象时，该对象都将被加入到 Session 的内部缓存中。

Hibernate 的查询方式

hql 查询： Query query = session.createQuery("hql 语句");

sql 查询： SQLQuery sqlQuery = session.createSQLQuery("sql 语句");

对象化查询 Criteria 方法。

jdbc、Hibernate、ibatis 的区别

jdbc:手动

手动写 sql

delete、insert、update 要将对象的值一个一个取出传到 sql 中,不能直接传入一个对象。

select:返回的是一个 resultset，要从 ResultSet 中一行一行、一个字段一个字段的取出，然后封装到一个对象中，不直接返回一个对象。

ibatis 的特点:半自动化

sql 要手动写

delete、insert、update:直接传入一个对象

select:直接返回一个对象

hibernate:全自动

不写 sql,自动封装

delete、insert、update:直接传入一个对象

select:直接返回一个对象

如何优化 Hibernate?

* 使用双向一对多关联，不使用单向一对多
* 灵活使用单向一对多关联
* 不用一对一，用多对一取代
* 配置对象缓存，不使用集合缓存
* 一对多集合使用 Bag,多对多集合使用 Set
* 继承类使用显式多态
* 表字段要少，表关联不要怕多，有二级缓存撑腰

在数据库中条件查询速度很慢的时候,如何优化?

1.建索引

2.减少表之间的关联

3.优化 sql，尽量让 sql 很快定位数据，不要让 sql 做全表查询，应该走索引,把数据量大的表排在前面

4.简化查询字段，没用的字段不要，已经对返回结果的控制，尽量返回少量数据

hibernate 的核心类是什么，它们的相互关系是什么?重要的方法是什么?

Configuration

SessionFactory

Session 如下方法

Save

load

Update

Delete

Query q=CreateQuery(“from Customer where customerName=:customerName”)

beginTransaction

close

Transaction

Commit()

load()和 get()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，load 默认支持延迟加载，在用到对象中的其他属性数据时才查询数据库，但是万一数据库中不存在该记录，只能抛异常 ObjectNotFoundEcception；所说的 load 方法抛异常是指在使用该对象的数据时，数据库中不存在该数据时抛异常，而不是在创建这个对象时。由于 session 中的缓存对于 hibernate 来说是个相当廉价的资源，所以在 load 时会先查一下 session 缓存看看该 id 对应的对象是否存在，不存在则创建代理。get() 先在一级缓存找，没有就去二级缓存找，没有就去数据库找没有就返回 null；而对于 get 方法，hibernate 一定要获取到真实的数据，否则返回 null。

get()采用立即加载方式,而 load()采用延迟加载;

get()方法执行的时候,会立即向数据库发出查询语句，而 load()方法返回的是一个代理(此代理中只有一个 id 属性),只有等真正使用该对象属性的时候,才会发出 sql 语句

如果数据库中没有对应的记录,get()方法返回的是 null.而 load()方法出现异 ObjectNotFoundException

Hibernate 中的 update()和 saveOrUpdate()的区别.

saveOrUpdate()方法可以实现 update()的功能，但会多些步骤，具体如下:

如果对象已经在本 session 中持久化了，不做任何事

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

如果对象没有持久化标识(identifier)属性，对其调用 save()

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()

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，save()它。

以上皆不符合则调用 update()更新之。

Hibernate 如何分页？

Query 接口的 setFirstResult(int i) 和 setMaxResults(int j)方法，i 是从第几行开始读，j 是读多少行。

Hibernate 的主键生成机制

1) assigned

主键由外部程序负责生成，无需 Hibernate 参与。

2) hilo

通过 hi/lo 算法实现的主键生成机制，需要额外的数据库表保存主键生成历史状态。

3) seqhilo

与 hilo 类似，通过 hi/lo 算法实现的主键生成机制，只是主键历史状态保存在 Sequence 中，适用于支持 Sequence 的数据库，如 Oracle。

4) increment

主键按数值顺序递增。此方式的实现机制为在当前应用实例中维持一个变量，以保存着当前的最大值，之后每次需要生成主键的时候将此值加 1 作为主键。这种方式可能产生的问题是：如果当前有多个实例访问同一个数据库，那么由于各个实例各自维护主键状态，不同实例可能生成同样的主键，从而造成主键重复异常。因此，如果同一数据库有多个实例访问，此方式必须避免使用。

5) identity

采用数据库提供的主键生成机制。如 DB2、SQL Server、MySQL 中的主键生成机制。

6) sequence

采用数据库提供的 sequence 机制生成主键。如 Oralce 中的 Sequence。

7) native

由 Hibernate 根据底层数据库自行判断采用 identity、hilo、sequence 其中一种作为主键生成方式。

8) uuid.hex

由 Hibernate 基于 128 位唯一值产生算法生成 16 进制数值(编码后以长度 32 的字符串表示)作为主键。

9) uuid.string

与 uuid.hex 类似，只是生成的主键未进行编码（长度 16）。在某些数据库中可能出现问题（如 PostgreSQL）。

10) foreign

使用外部表的字段作为主键。一般而言，利用 uuid.hex 方式生成主键将提供最好的性能和数据库平台适应性。

这10中生成OID标识符的方法,increment 比较常用,把标识符生成的权力交给Hibernate处理.但是当同时多个 Hibernate 应用操作同一个数据库,甚至同一张表的时候.就推荐使用 identity 依赖底层数据库实现,但是数据库必须支持自动增长,当然针对不同的数据库选择不同的方法.如果你不能确定你使用的数据库具体支持什么的情况下.可以选择用 native 让Hibernate 来帮选择 identity,sequence,或 hilo.

另外由于常用的数据库，如 Oracle、DB2、SQLServer、MySql 等，都提供了易用的主键生成机制（Auto-Increase 字段或者 Sequence）。我们可以在数据库提供的主键生成机制上，采用 generator-class=native 的主键生成方式。

不过值得注意的是，一些数据库提供的主键生成机制在效率上未必最佳，大量并发 insert 数据时可能会引起表之间的互锁。数据库提供的主键生成机制，往往是通过在一个内部表中保存当前主键状态（如对于自增型主键而言，此内部表中就维护着当前的最大值和递增量），之后每次插入数据会读取这个最大值，然后加上递增量作为新记录的主键，之后再把这个新的最大值更新回内部表中，这样，一次 Insert 操作可能导致数据库内部多次表读写操作，同时伴随的还有数据的加锁解锁操作，这对性能产生了较大影响。因此，对于并发 Insert 要求较高的系统，推荐采用 uuid.hex 作为主键生成机制

Hibernate 的三种状态

瞬时态(Transient)、 持久态(Persistent)、脱管态(Detached)。处于持久态的对象也称为 PO(Persistence Object)，瞬时对象和脱管对象也称为 VO（Value Object）。

瞬时态

由 new 命令开辟内存空间的 java 对象，

eg. Person person = new Person(“amigo”，“女”);

如果没有变量对该对象进行引用，它将被 java 虚拟机回收。

瞬时对象在内存孤立存在，它是携带信息的载体，不和数据库的数据有任何关联关系，在 Hibernate 中，可通过 session 的 save()或 saveOrUpdate()方法将瞬时对象与数据库相关联，并将数据对应的插入数据库中，此时该瞬时对象转变成持久化对象。

持久态

处于该状态的对象在数据库中具有对应的记录，并拥有一个持久化标识。如果是用 hibernate 的 delete()方法，对应的持久对象就变成瞬时对象，因数据库中的对应数据已被删除，该对象不再与数据库的记录关联。

当一个 session 执行 close()或 clear()、evict()之后，持久对象变成脱管对象，此时持久对象会变成脱管对象，此时该对象虽然具有数据库识别值，但它已不在 HIbernate 持久层的管理之下。

持久对象具有如下特点：

1. 和 session 实例关联；

2. 在数据库中有与之关联的记录。

脱管态

当与某持久对象关联的 session 被关闭后，该持久对象转变为脱管对象。当脱管对象被重新关联到 session 上时，并再次转变成持久对象。

脱管对象拥有数据库的识别值，可通过 update()、saveOrUpdate()等方法，转变成持久对象。

脱管对象具有如下特点：

1. 本质上与瞬时对象相同，在没有任何变量引用它时，JVM 会在适当的时候将它回收；

2. 比瞬时对象多了一个数据库记录标识值。

简述 Hibernate 和 JDBC 的优缺点? 如何书写一个 one to many 配置文件.

Hibernate 就是封装了 JDBC，他写一条 hql 语句，可以在不同数据库中使用，不用修改 hql 语句，但是关联查询效率低。

JDBC 是基础的链接数据库的框架，效率高，但是 mysql、oracle、sql service 等不同的数据库要写不同的 sql 语句。

比如 Class 和 Student 吧，就是一个班级对应多个学生：

在 Class 的配置文件中追加（Class.hbm.xml）

```
<!-- 追加集合属性的配置 -->
<set name="students" lazy="false" fetch="join" cascade="all" inverse="true">
    <!-- 设置关联字段 -->
    <key column="classId" />
    <!-- 设置关联关系 -->
    <one-to-many class="Studnet" />
</set>
```

在 Student 的配置文件中（Student.hbm.xml）

```
<many-to-one name="class" column="classId" lazy="false" fetch="join" class="Class">
</many-to-one>
```

iBatis 与 Hibernate 有什么不同?

相同点：屏蔽 jdbc api 的底层访问细节，使用我们不用与 jdbc api 打交道，就可以访问数据。

jdbc api 编程流程固定，还将 sql 语句与 java 代码混杂在了一起，经常需要拼凑 sql 语句，细节很繁琐。

ibatis 的好处：屏蔽 jdbc api 的底层访问细节；将 sql 语句与 java 代码进行分离;提供了将结果集自动封装称为实体对象和对象的集合的功能,queryForList 返回对象集合,用 queryForObject 返回单个对象；提供了自动将实体对象的属性传递给 sql 语句的参数。

Hibernate 是一个全自动的 orm 映射工具,它可以自动生成 sql 语句,ibatis 需要我们自己在 xml 配置文件中写 sql 语句，hibernate 要比 ibatis 功能负责和强大很多。因为 hibernate 自动生成 sql 语句，我们无法控制该语句，我们就无法去写特定的高效率的 sql。对于一些不太复杂的 sql 查询，hibernate 可以很好帮我们完成，但是，对于特别复杂的查询，hibernate 就很难适应了，这时候用 ibatis 就是不错的选择，因为 ibatis 还是由我们自己写 sql 语句。

写 Hibernate 的一对多和多对一双向关联的 orm 配置?

多对一双向关联

注意:那个外键(两个约束必须要执行同一个外键)

```
<many-to-one name="dept"    class="Dept"    column="deptId">
</many-to-one>
<set name="emps" >
    <key column="deptId"></key>
    <one-to-many class="Emp"/>
</set>
```

注意 deptId,多对一和一对多,外键的名字必须一致

hibernate inverse 属性的作用

hibernate 配置文件中有这么一个属性 inverse，它是用来指定关联的控制方的。

inverse 属性默认是 false，若为 false，则关联由自己控制，若为 true，则关联由对方控制。

Hibernate 如何实现数据表映射的继承关系

1、两个表，子类重复父类的属性。
2、一个表，子类父类共用一个表
3、两个表，子类引用父类的主键，享用公共的字段或属性。

hibernate 进行多表查询每个表中各取几个字段，也就是说查询出来的结果集没有一个实体类与之对应如何解决

解决方案一，按照 Object[]数据取出数据，然后自己组 bean

解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2 field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

如何查看 Hibernate 生成并执行的 sql

在定义数据库和数据库属性的文件 applicationConfig.xml 里面，把 hibernate.show_sql 设置为 true

这样生成的 SQL 就会在控制台出现了
注意：这样做会加重系统的负担，不利于性能调优

比较 Hibernate 的三种检索策略优缺点

1 立即检索；
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与它关联的对象；
缺点：1.select 语句太多；2.可能会加载应用程序不需要访问的对象白白浪费许多内存空间；
2 延迟检索：
优点：由应用程序决定需要加载哪些对象，可以避免可执行多余的 select 语句，以及避免加载应用程序不需要访问的对象。因此能提高检索性能，并且能节省内存空间；
缺点：应用程序如果希望访问游离状态代理类实例，必须保证他在持久化状态时已经被初始化；
3 迫切左外连接检索
优点：1 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地冲一个对象导航到与它关联的对象。2 使用了外连接，select 语句数目少；
缺点：1 可能会加载应用程序不需要访问的对象，白白浪费许多内存空间；2 复杂的数据库表连接也会影响检索性能；

Hibernate 拒接连接、服务器崩溃的原因

1. db 没有打开
2. 网络连接可能出了问题
3. 连接配置错了
4. 驱动的 driver，url 是否都写对了
5. LIB 下加入相应驱动，数据连接代码是否有误
6. 数据库配置可能有问题
7. 当前联接太多了，服务器都有访问人数限制的
8. 服务器的相应端口没有开，即它不提供相应的服务
9 hibernate 有哪些缓存，分别怎么使用?
10 你对 hibernate 的了解到了一个什么样的程度?
11 写出一个 sql 语句体现 hibernate 中一对多的关系

struts 是什么？

struts1 是基于 JSP 和 servlet 的一个开源的 Web 应用框架，使用的是 MVC 的设计模式。

struts2 是基于 webwork 技术的框架，是 sun 和 webwork 公司联手开发的一个功能非常齐全的框架，struts2 和 struts1 没有任何关系，是一个全新的框架

谈谈你对 Struts 的理解。

1. struts 是一个按 MVC 模式设计的 Web 层框架，其实它就是一个大大的 servlet，这个 Servlet 名为 ActionServlet，或是 ActionServlet 的子类。我们可以在 web.xml 文件中将符合某种特征的所有请求交给这个 Servlet 处理，这个 Servlet 再参照一个配置文件（通常为 /WEB-INF/struts-config.xml）将各个请求分别分配给不同的 action 去处理。

一个扩展知识点：struts 的配置文件可以有多个，可以按模块配置各自的配置文件，这样可以防止配置文件的过度膨胀；

2.ActionServlet 把请求交给 action 去处理之前，会将请求参数封装成一个 formbean 对象（就是一个 java 类，这个类中的每个属性对应一个请求参数），封装成一个什么样的 formbean 对象呢？看配置文件。

3.要说明的是，ActionServlet 把 formbean 对象传递给 action 的 execute 方法之前，可能会调用 formbean 的 validate 方法进行校验，只有校验通过后才将这个 formbean 对象传递给 action 的 execute 方法，否则，它将返回一个错误页面，这个错误页面由 input 属性指定，（看配置文件）作者为什么将这里命名为 input 属性，而不是 error 属性，我们后面结合实际的运行效果进行分析。

4.action 执行完后要返回显示的结果视图，这个结果视图是用一个 ActionForward 对象来表示的， actionforward 对象通过 struts-config.xml 配置文件中的配置关联到某个 jsp 页面，因为程序中使用的是在 struts-config.xml 配置文件为 jsp 页面设置的逻辑名，这样可以实现 action 程序代码与返回的 jsp 页面名称的解耦。

Struts1 执行流程：

1、客户端浏览器发出 HTTP 请求。2、根据 web.xml 配置，该请求被 ActionServlet 接收。3、根据 struts-config.xml 配置，ActionServlet 先将请求中的参数填充到 ActionForm 中，然后 ActionServlet 再将请求发送到 Action 进行处理。4、是否验证，需要验证则调用 ActionForm 的 validate 方法，验证失败则跳转到 input，成功则继续。5、Action 从 ActionForm 获得数据，

调用 javabean 中的业务方法处理数据。6、Action 返回 ActionForward 对象，跳转到相应 JSP 页面或 Action。7、返回 HTTP 响应到客户端浏览器。

Struts2 的执行流程：

Struts 2 框架本身大致可以分为 3 个部分：核心控制器 FilterDispatcher、业务控制器 Action 和用户实现的企业业务逻辑组件。核心控制器 FilterDispatcher 是 Struts 2 框架的基础，包含了框架内部的控制流程和处理机制。业务控制器 Action 和业务逻辑组件是需要用户来自己实现的。用户在开发 Action 和业务逻辑组件的同时，还需要编写相关的配置文件，供核心控制器 FilterDispatcher 来使用。

Struts 2 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。

说说 struts1 与 struts2 的区别。

1.都是 MVC 的 WEB 框架,

2 struts1 的老牌框架，应用很广泛，有很好的群众基础，使用它开发风险很小，成本更低！struts2 虽然基于这个框架，但是应用群众并多，相对不成熟，未知的风险和变化很多，开发人员相对不好招，使用它开发项目的风险系数更大，用人成本更高！

3.struts2 毕竟是站在前辈的基础设计出来，它会改善和完善 struts1 中的一些缺陷，struts1 中一些悬而未决问题在 struts2 得到了解决。

4.struts1 的前端控制器是一个 Servlet,名称为 ActionServlet,struts2 的前端控制器是一个 filter，在 struts2.0 中叫 FilterDispatcher，在 struts2.1 中叫 StrutsPrepareAndExecuteFilter。

5.struts1 的 action 需要继承 Action 类，struts2 的 action 可以不继承任何类；struts1 对同一个路径的所有请求共享一个 Action 实例，struts2 对同一个路径的每个请求分别使用一个独立 Action 实例对象，所有对于 struts2 的 Action 不用考虑线程安全问题。

6.在 struts1 中使用 formbean 封装请求参数,在 struts2 中直接使用 action 的属性来封装请求参数。

7.struts1 中的多个业务方法放在一个 Action 中时（即继承 DispatchAction 时），要么都校验，要么都不校验；对于 struts2，可以指定只对某个方法进行校验，当一个 Action 继承了 ctionSupport 且在这个类中只编写了 validateXxx()方法，那么则只对 Xxx()方法进行校验。

8.与 Struts1 不同，Struts2 对用户的每一次请求都会创建一个 Action，所以 Struts2 中的 Action 是线程安全的。

9.struts 配置文件中的 redirect 视图的 url 不能接受参数，而 struts2 配置文件中的 redirect 视图可以接受参数。

Action 是不是线程安全的？如果不是，有什么方式可以保证 Action 的线程安全？如果是，说明原因

不是

在 spring 中用 scope="prototype"来管理。

分析一下 struts 是如何实现 MVC 的

m：JavaBean 或结合 EJB 组件或者 pojo 构成

c：Action 来实现

v：一组 JSP 文件及 struts 标签等构成。

Struts 工作机制？为什么要使用 Struts?

工作机制：

在 web 应用启动时就会加载初始化 ActionServlet,ActionServlet 从 struts-config.xml 文件中读取配置信息,把它们存放到各种配置对象当 ActionServlet 接收到一个客户请求时,将执行如下流程.

(1)检索和用户请求匹配的 ActionMapping 实例,如果不存在,就返回请求路径无效信息;

(2)如果 ActionForm 实例不存在,就创建一个 ActionForm 对象,把客户提交的表单数据保存到 ActionForm 对象中;

(3)根据配置信息决定是否需要表单验证.如果需要验证,就调用 ActionForm 的 validate()方法;

(4)如果 ActionForm 的 validate()方法返回 null 或返回一个不包含 ActionMessage 的 ActuibErrors 对象, 就表示表单验证成功;

(5)ActionServlet 根据 ActionMapping 所包含的映射信息决定将请求转发给哪个 Action,如果相应的 Action 实例不存在,就先创建这个实例,然后调用 Action 的 execute()方法;

(6)Action 的 execute()方法返回一个 ActionForward 对象,ActionServlet 在把客户请求转发给

ActionForward 对象指向的 JSP 组件;
(7)ActionForward 对象指向 JSP 组件生成动态网页,返回给客户;

为什么要用:
JSP、Servlet、JavaBean 技术的出现给我们构建强大的企业应用系统提供了可能。但用这些技术构建的系统非常的繁乱，所以在此之上，我们需要一个规则、一个把这些技术组织起来的规则，这就是框架，Struts 便应运而生。

基于 Struts 开发的应用由 3 类组件构成：控制器组件、模型组件、视图组件

Struts 的 validate 框架是如何验证的?
在 struts 配置文件中配置具体的错误提示，再在 FormBean 中的 validate()方法具体调用。

struts 中的几个关键对象的作用(说说几个关键对象的作用)

Action：控制器类。ActionForm：表单对象。DynaValidatorForm：动态 form。ActonMapping:配置文件中 action 节点的信息。

说下 Struts 的设计模式

MVC 模式: web 应用程序启动时就会加载并初始化 ActionServler。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，ActionServler 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。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，ActionForward 对象指定的 jsp 生成动 态的网页，返回给客户。

Struts 优缺点

Struts 跟 Tomcat、Turbine 等诸 多 Apache 项目一样，是开源软件，这是它的一大优点。使开发者能更深入的了解其内部实现机制。Struts 开放源码框架的创建是为了使开发者在构建基于 Java Servlet 和 JavaServer Pages（JSP）技术的 Web 应用时更加容易。Struts 框架为开放者提供了一个统一的标准框架，通过使用 Struts 作为基础，开发者能够更专注于应用程序的商业逻辑。Struts 框架本身是使用 Java Servlet 和 JavaServer Pages 技术的一种 Model-View-Controller（MVC）实现.

具体来讲,Struts 的优点有:

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:

一、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。

二、 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。

三、 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。

四、 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。

五、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。

六、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。

七、 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。

八、 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。

九、 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 application event，application event 和 component event 相比是一种粗粒度的事件。

Struts 重要的表单对象 ActionForm 是一种对象，它代表了一种应用，这个对象中至少包含几个字段， 这些字段是 Jsp 页面表单中的 input 字段，因为一个表单对应一个事件，所以，当我们需要将事件粒度细化到表单中这些字段时，也就是说，一个字段对应一个事件时，单纯使用 Struts 就不太可能，当然通过结合 JavaScript 也是可以转弯实现的。

谈谈 Struts 中的 Action servlet。

1、ActionServlet 类是 Struts 框架的内置核心控制器组件，它继承了 javax. servlet.http.HttpServlet 类，Struts 的启动一般从加载 ActionServlet 开始，因此它在 MVC 模型中扮演中央控制器的角色。

2、在 Struts 中，它的主要作用是用来接收用户的请求信息，然后根据系统配置要求将请求传递给相应的 Action 对象。

在具体实现时，它首先要判断 Action 对象是否存在，如果不存在则先创建该对象；在请求被接收后，控制器会将其传递给一个 Action 实例，这一过程同样会判断实例是否存在，如果不存在则需先创建该实例的 execute()方法。

此类在 struts1.x 中特别重要，相当于中央控制器，当请求过来之时，首先经过此类过滤，然后，根据请求转向相应的 action。

但是 struts2 的核心是 webworks，过滤拦截由 webworks 来做就不是 struts1.x 了，所以对它就没再提。

拦截器和过滤器的区别

首先要明确什么是拦截器、什么是过滤器

1.1 什么是拦截器:

拦截器，在 AOP（Aspect-Oriented Programming）中用于在某个方法或字段被访问之前，进行拦截然后在之前或之后加入某些操作。拦截是 AOP 的一种实现策略。

在 Webwork 的中文文档的解释为——拦截器是动态拦截 Action 调用的对象。它提供了一种机制可以使开发者可以定义在一个 action 执行的前后执行的代码，也可以在一个 action 执行前阻止其执行。同时也是提供了一种可以提取 action 中可重用的部分的方式。

谈到拦截器，还有一个词大家应该知道——拦截器链（Interceptor Chain，在 Struts 2 中称为拦截器栈 Interceptor Stack）。拦截器链就是将拦截器按一定的顺序联结成一条链。在访问被拦截的方法或字段时，拦截器链中的拦截器就会按其之前定义的顺序被调用。

1.2. 拦截器的实现原理:

大部分时候，拦截器方法都是通过代理的方式来调用的。Struts 2 的拦截器实现相对简单。当请求到达 Struts 2 的 ServletDispatcher 时，Struts 2 会查找配置文件，并根据其配置实例化相对的拦截器对象，然后串成一个列表（list），最后一个一个地调用列表中的拦截器。

1.3 什么是过滤器

过滤器是一个程序，它先于与之相关的 servlet 或 JSP 页面运行在服务器上。过滤器可附加到一个或多个 servlet 或 JSP 页面上，并且可以检查进入这些资源的请求信息。在这之后，过滤器可以作如下的选择:

①以常规的方式调用资源（即，调用 servlet 或 JSP 页面）。

②利用修改过的请求信息调用资源。

③调用资源，但在发送响应到客户机前对其进行修改。

④阻止该资源调用，代之以转到其他的资源，返回一个特定的状态代码或生成替换输出。

1.4 Servlet 过滤器的基本原理

在 Servlet 作为过滤器使用时，它可以对客户的请求进行处理。处理完成后，它会交给下一个过滤器处理，这样，客户的请求在过滤链里逐个处理，直到请求发送到目标为止。例如，某网站里有提交“修改的注册信息”的网页，当用户填写完修改信息并提交后，服务器在进行处理时需要做两项工作：判断客户端的会话是否有效；对提交的数据进行统一编码。这两

项工作可以在由两个过滤器组成的过滤链里进行处理。当过滤器处理成功后，把提交的数据发送到最终目标；如果过滤器处理不成功，将把视图派发到指定的错误页面。

2、拦截器与过滤器的区别 ：

1. 拦截器是基于 java 的反射机制的，而过滤器是基于函数回调。

2. 拦截器不依赖与 servlet 容器，过滤器依赖与 servlet 容器。

3. 拦截器只能对 action 请求起作用，而过滤器则可以对几乎所有的请求起作用。

4. 拦截器可以访问 action 上下文、值栈里的对象，而过滤器不能访问。

5. 在 action 的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次

拦截器的代码实现(以 struts2 为例)：

1、在 xml 文件中如何定义拦截器

```
<interceptors>
    <interceptor name="filterIPInterceptor" class="com.xxxx.web.FilterIPActionInterceptor" />
    <interceptor-stack name="filterIPStack">
        <interceptor-ref name="defaultStack" />
        <interceptor-ref name="filterIPInterceptor" />
    </interceptor-stack>
</interceptors>
```

Struts 的 validate 框架是如何验证的?

在 struts 配置文件中配置具体的错误提示，再在 FormBean 中的 validate()方法具体调用。

spring 是什么?

spring 是一个集成了许多第三方框架的大杂烩，其核心技术是 IOC（控制反转，也称依赖注入）和 AOP（面向切面编程）

AOP 让开发人员可以创建非行为性的关注点，称为横切关注点，并将它们插入到应用程序代码中。使用 AOP 后，公共服务 （比如日志、持久性、事务等）就可以分解成方面并应用到域对象上，同时不会增加域对象的对象模型的复杂性。 IOC 允许创建一个可以构造对象的应用环境，然后向这些对象传递它们的协作对象。正如单词‘倒置’所表明的，IOC 就像反过来的 JNDI。没有使用一堆抽象工厂、服务定位器、单元素（singleton）和直接构造（straight construction），每一个对象都是用其协作对象构造的。因此是由容器管理协作对象（collaborator）。

Spring 即是一个 AOP 框架，也是一 IOC 容器。Spring 最好的地方是它有助于您替换对象。有了 Spring，只要用 JavaBean 属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。

你对 Spring 的理解。

1.Spring 实现了工厂模式的工厂类（在这里有必要解释清楚什么是工厂模式），这个类名为 BeanFactory（实际上是一个接口），在程序中通常 BeanFactory 的子类 ApplicationContext。Spring 相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的类名和实例对象的属性。

2. Spring 提供了对 IOC 良好支持，IOC 是一种编程思想，是一种架构艺术，利用这种思想可以很好地实现模块之间的解耦。IOC 也称为 DI（Depency Injection），什么叫依赖注入呢？

譬如，Class Programmer

```
{
    Computer computer = null;
    public void code()
    {
        //Computer computer = new IBMComputer();
        //Computer computer = beanfacotry.getComputer();
        computer.write();
    }
```

```
    public void setComputer(Computer computer)
    {
        this.computer = computer;
    }
}
```

另外两种方式都由依赖，第一个直接依赖于目标类，第二个把依赖转移到工厂上，第三个彻底与目标和工厂解耦了。在 spring 的配置文件中配置片段如下：

```
<bean id=”computer” class=”cn.itcast.interview.Computer”></bean>

<bean id=”programmer” class=”cn.itcast.interview.Programmer”>
    <property name=”computer” ref=”computer”></property>
</bean>
```

3. Spring 提供了对 AOP 技术的良好封装， AOP 称为面向切面编程，就是系统中有很多各不相干的类的方法，在这些众多方法中要加入某种系统功能的代码，例如，加入日志，加入权限判断，加入异常处理，这种应用称为 AOP。实现 AOP 功能采用的是代理技术，客户端程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明，有两种方式可以实现相同的方法声明，一是实现相同的接口，二是作为目标的子类在，JDK 中采用 Proxy 类产生动态代理的方式为某个接口生成实现类，如果要为某个类生成子类，则可以用 CGLIB。在生成的代理类的方法中加入系统功能和调用目标类的相应方法，系统功能的代理以 Advice 对象进行提供，显然要创建出代理对象，至少需要目标类和 Advice 类。spring 提供了这种支持，只需要在 spring 配置文件中配置这两个元素即可实现代理和 aop 功能，例如，

```
<bean id=”proxy” type=”org.spring.framework.aop.ProxyBeanFactory”>
    <property name=”target” ref=””></property>
    <property name=”advisor” ref=””></property>
</bean>
```

spring 工作机制及为什么要用?

1.spring mvc 请所有的请求都提交给 DispatcherServlet,它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。

2.DispatcherServlet 查询一个或多个 HandlerMapping,找到处理请求的 Controller.

3.DispatcherServlet 请请求提交到目标 Controller

4.Controller 进行业务逻辑处理后，会返回一个 ModelAndView

5.Dispathcher 查询一个或多个 ViewResolver 视图解析器,找到 ModelAndView 对象指定的视图对象

6.视图对象负责渲染返回给客户端。

为什么用:

{AOP 让开发人员可以创建非行为性的关注点，称为横切关注点，并将它们插入到应用程序代码中。使用 AOP 后，公共服务（比如日志、持久性、事务等）就可以分解成方面并应用到域对象上，同时不会增加域对象的对象模型的复杂性。

IOC 允许创建一个可以构造对象的应用环境，然后向这些对象传递它们的协作对象。正如单词 倒置 所表明的，IOC 就像反 过来的 JNDI。没有使用一堆抽象工厂、服务定位器、单元素（singleton）和直接构造（straight construction），每一个对象都是用其协作对象构造的。因此是由容器管理协作对象（collaborator）。

Spring 即使一个 AOP 框架，也是一 IOC 容器。Spring 最好的地方是它有助于您替换对象。有了 Spring，只要用 JavaBean 属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。}

Spring 优缺点:

它是一个开源的项目，而且目前非常活跃；它基于 IoC（Inversion of Control，控制反转）和 AOP 的构架多层 j2ee 系统的框架，但它不强迫你必须在每一层中必须使用 Spring，因为它模块化的很好，己的需要选择使用它的某一个模块；它实现了很优雅的 MVC，对不同的数据访问技术提供了统一的接口，采用 IoC 使得可以很容易的实现 bean 的装配，提供了简洁的 AOP 并据此实现 Transcation Managment，等等

优点:

a. 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。

b. Spring 能消除在许多工程中常见的对 Singleton 的过多使用。根据我的经验，这是一个很大的问题，它降低了系统的可测试性和面向对象的程度。

c. 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 of Control 的使用（在下面讨论）帮助完成了这种简化。

d.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。

e. 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。

f. 使用 Spring 构建的应用程序易于单元测试。

g.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。

h. 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。

i. 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）。

Spring 确实使你能通过最简单可行的解决办法来解决你的问题。而这是有有很大价值的。

缺点：

使用人数不多、jsp 中要写很多代码、控制器过于灵活，缺少一个公用控制器。

Spring 对多种 ORM 框架提供了很好的支持，简单描述在 Spring 中使用 Hibernate 的方法，并结合事务管理。

getHiberanteTemplate 里面提供了 save，update，delete，find 等方法。

简单说一个：如果配置了声明式事务，当执行 getHibernateTemplate 的各种方法的时候，事务会自动被加载 ，如果没有配置事务，那么以上操作不会真正的被同步到数据库，除非配置了 hibernate 的 autocommit=true

spring 的事务有几种方式？谈谈 spring 事务的隔离级别和传播行为。

spring 事务分两种形式，声明式事务和编程式事务，spring 提供了一个事务的接口

PaltformTractionManager 接口，针对不同的事务，spring 进行了不同的实现,对 hibernate 事务的实现 HIbernateTractionManager,对 JDBC 的 JdbcTractionManager,

DataSourceTractionManager 以及 JdoTractionManager。接口 platformTractionManager 提供了三个方法，获取事务，提交和回滚的方法。

spring 的事务有几种方式？谈谈 spring 事务的隔离级别和传播行为。

声明事务和编程事务

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

Spring 的依赖注入是什么意思? 给一个 Bean 的 message 属性, 字符串类型, 注入值为 "Hello" 的 XML 配置文件该怎么写?

依赖注入是 Spring IOC 的主要作用，依赖就是指属性，意思就是说将属性利用 Spring 注入到程序中，而非 new。

<bean id="message" class="message 所在类的路径名">

<property name="message" value="Hello"/>

</bean>

IOC 和 AOP

Ioc，控制反转（也称作依赖注入）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。

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。

简述你对IoC(Inversion of Control)的理解,描述一下Spring中实现DI(Dependency Injection)的几种方式。

方式一：接口注入，在实际中得到了普遍应用，即使在 IOC 的概念尚未确立时，这样的方法也已经频繁出现在我们的代码中。

方式二：对象注入，对象创建之后，将被依赖对象通过 set 方法设置进去

方式三：构造方法注入，对象创建时，被依赖对象以构造方法参数的方式注入

Spring 的 Bean 有多种作用域

singleton、prototype、request、session、global session、application、自定义

简单描述 Spring 与 Struts 的不同之处，整合 Spring 与 Struts 有哪些方法，哪种最好，为什么？

Spring 主要管理组件之间的依赖，提供 AOP 实现。Struts 在程序结构中起到控制器的作用。

三框架整合连接数据库的方法

第一种在 Spring applicationContext.xml 中连接：

oracle.jdbc.driver.OracleDriver

jdbc:oracle:thin:@localhost:1521:test

cpiclh

cpiclh

com/Hibernate/Pojo/FactUsers.hbm.xml

org.hibernate.dialect.Oracle9Dialect true

第二种在 Hibernate hibernate.cfg.xml 中连接：

root

jdbc:oracle:thin:@192.168.0.1:1521:test

org.hibernate.dialect.Oracle9Dialect root

oracle.jdbc.driver.OracleDriver

第三种在 Tomcat 中的 apache-tomcat-5.5.17\conf\Catalina 目录下放一个

和目录同名的 XML，内容如下

<="" p="">

username="root" password="root" driverClassName="oracle.jdbc.driver.OracleDriver"

url="jdbc:oracle:thin:@192.168.0.1:1521:test"/>