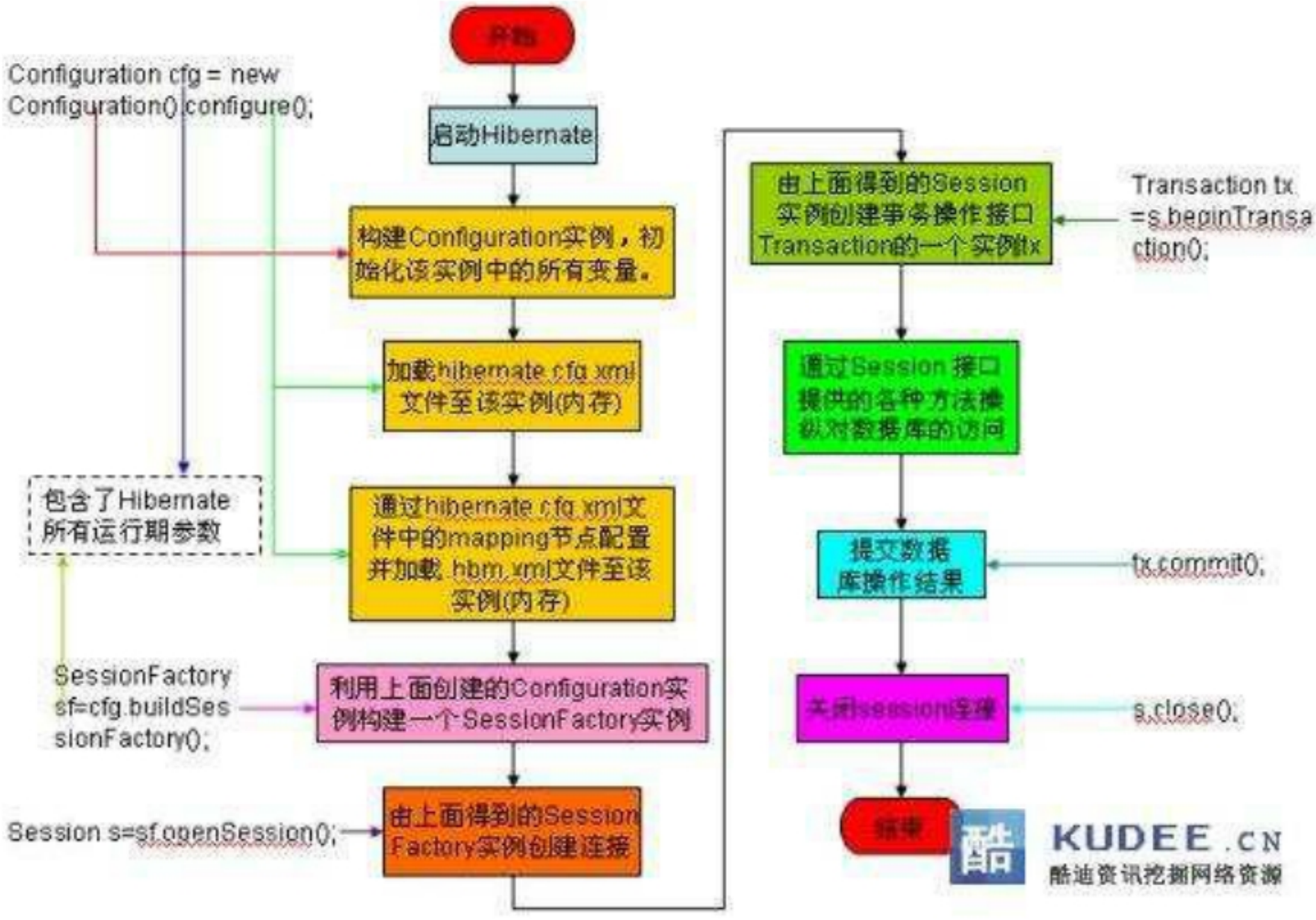
Java WEB 开发 框架面试题

tangbi 整理

题目 1：Hibernate 工作原理及为什么要用？

原理：

hibernate，通过对 jdbc 进行封装，对 java 类和 关系数据库进行 mapping，实现了对关系数据库的面向对象方式的操作，改变了传统的 jdbc + sql 操作数据的方式，从而使开发人员可以话更多精力进行对象方面的开发



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 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。

题目 2： 什么是 Hibernate 延迟加载?

延迟加载机制是为了避免一些无谓的性能开销而提出来的，所谓延迟加载就是当在真正需要数据的时候，才真正执行数据加载操作。在 Hibernate 中提供了对实体对象的延迟加载以及对集合的延迟加载，另外在 Hibernate3 中还提供了对属性的延迟加载。

题目 3：Hibernate 中类之间的关联关系有几种?(如：一对多、多对多的关系)

many-to-one、one-to-many、many-to-many、 one-to-one

题目 4： 说下 Hibernate 的缓存机制

一、hibernate 一级缓存

（1）hibernate 支持两个级别的缓存，默认只支持一级缓存；

（2）每个 Session 内部自带一个一级缓存；

（3）某个 Session 被关闭时，其对应的一级缓存自动清除；

二、hibernate 二级缓存

(1) 二级缓存独立于 session，默认不开启；

题目 5： Hibernate 的查询方式

本地 SQL 查询、Criteria、Hql

题目 6： 如何优化 Hibernate?

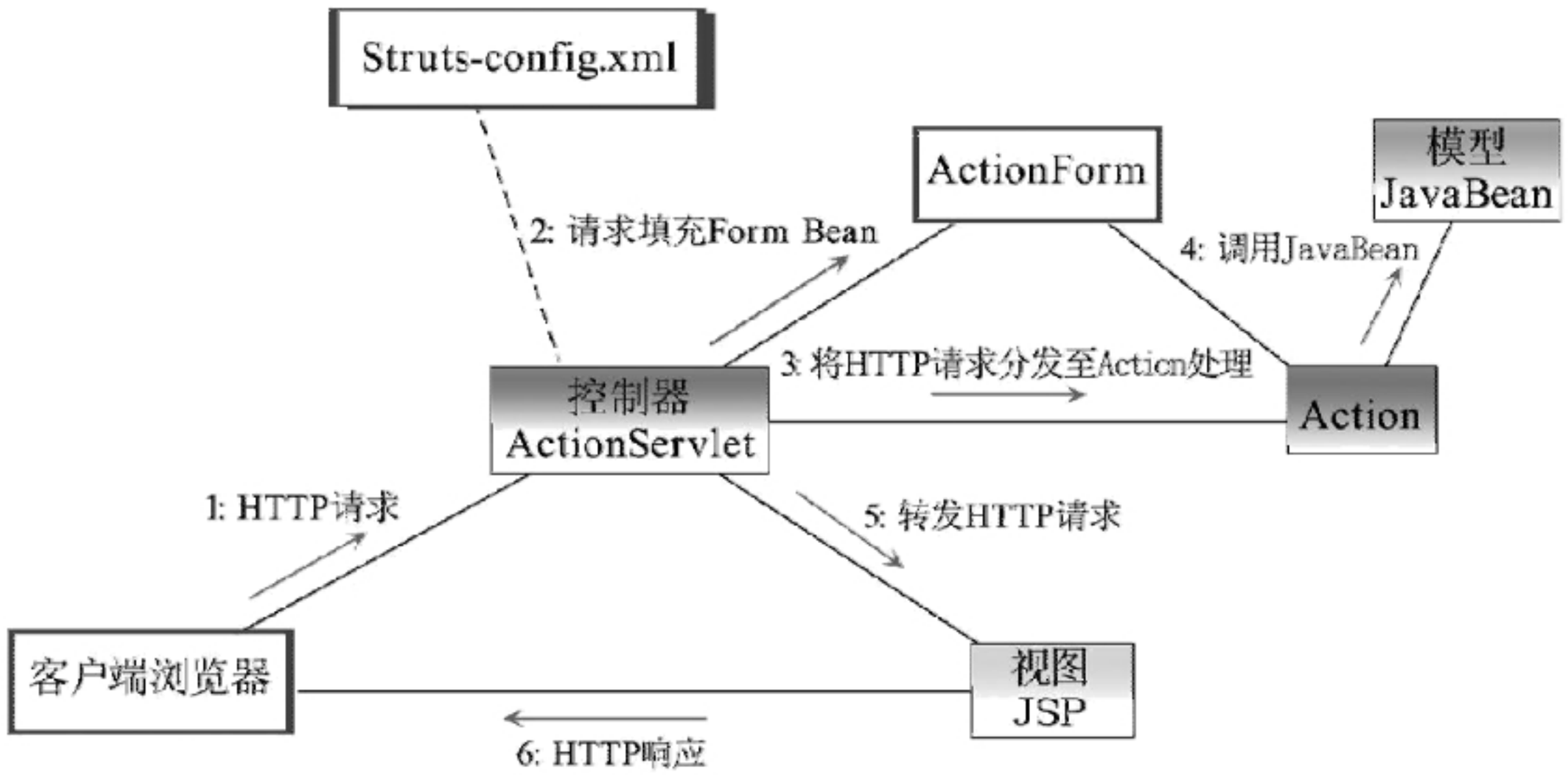
1.使用双向一对多关联，不使用单向一对多

2.不用一对一，用多对一取代

3.配置对象缓存，不使用集合缓存

题目 7： Struts 工作机制？为什么要使用 Struts?

工作机制：



Struts 的工作流程:

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:

1. 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。
2. 基于 Struts 开发的应用:

不用再考虑公共问题

专心在业务实现上

结构统一,易于学习、维护

新手也可写出好程序

题目 10:为什么要用 spring?

Spring 是一个轻量级的 IOC 和 AOP 框架。

IOC(控制反转)意味着将你设计好的类交给系统去控制,而不是在你的类内部控制。这称为控制反转

AOP(面向切面),它将那些影响多个类的行为封装到可重用的模块中,面向对象是把问题从同类事物中抽象出来,面向切面是把问题从不同类问题中抽象出来。

1. hibernate 中 get()与 load()区别

请注意如果没有匹配的数据库记录,load()方法可能抛出无法恢复的异常(unrecoverable exception)。 如果类的映射使用了代理(proxy),load()方法会返回一个未初始化的代理,直到你调用该代理的某方法时才会去访问数据库。若你希望在某对象中创建一个指向另一个对象的关联,又不想在从数据库中装载该对象时同时装载相关联的那个对象,那么这种操作方式就用得上的了。 如果为相应类映射关系设置了 batch-size, 那么使用这种操作方式允许多个对象被一批装载(因为返回的是代理,无需从数据库中抓取所有对象的数据)。 如果你不确定是否有匹配的行存在,应该使用 get()方法,它会立刻访问数据库,如果没有对应的行,会返回 null。

17. [Hibernate 题目]判断题:使用 save/persist 一个对象时,便立即向数据库发送执行 insert sql 语句?

1) persist 把一个瞬态的实例持久化,但是并"不保证"标识符被立刻填入到持久化实例中，标识符的填入可能被推迟到 flush 的时间。
2) persist"保证"当它在一个 transaction 外部被调用的时候并不触发一个 Sql Insert，这个功能是很有用的。
3) save 会立即执行 Sql insert，不管是不是在 transaction 内部还是外部.

18. [Hibernate 题目]：指出一下代码哪里错误使用了 Hibernate。

背景简介：Board 是一个实体类，id 是它的主键，name 和 description 是他的两个属性。BoardDao 是 Board 实体的数据访问对象，BoardBo 是业务对象，用户提交变更 Board 对象的请求，由 Struts 的 BoardAction 接收，调用 BoardBo 处理。HibernateUtil.currentSession() 用于返回当前请求的 Session 对象。

```
1. //数据访问层代码：BoardDao.java
2.      public Board loadBoard(Long id) {
3.          Session session = HibernateUtil.currentSession();
4.          return session.load(Board.class, id);
5.      }
6.      public void updateBoard(Board board) {
7.          Session session = HibernateUtil.currentSession();
8.          session.update(board);
9.      }
10.
11.     //业务对象层代码：BoardBo.java
12.     private BoardDao boardDao;
13.     public void updateBoard(Long id, String name, String descrip
   tion) {
14.         Board board = boardDao.loadBoard(id);
15.         board.setName(name);
16.         board.setDescription(description);
17.         boardDao.updateBoard(board);
18.     }
19.
20.     //Web 控制器代码：BoardAction.java
21.     private BoardBo BoardBo;
22.     public ActionForward execute(
23.ActionMapping mapping,
24.ActionForm form,
25.HttpServletRequest request,
26.HttpServletResponse response) throws Exception {
27.         String id = request.getParameter("id");
28.         String name = request.getParameter("name");
29.         String description = request.getParameter("description")
   ;
30.         boardBo.updateBoard(id, name, description);
```

```
31.             return mapping.findForward("update-success");
32.         }
```

```
6.          public void updateBoard(Board board) {
7.                  Session session = HibernateUtil.currentSession();
            Transaction t=session.beginTransaction();
8.                  session.update(board);
            t.commit();
9.          }
```

简单叙述一下 Spring 中 BeanFactory 与 ApplicationContext 的差别

使用 BeanFactory 从 xml 配置文件加载 bean:

```
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;

public class XmlConfigWithBeanFactory {

    public static void main(String[] args) {
        XmlBeanFactory factory = new XmlBeanFactory(new FileSystemResource(
                "build/beans.xml"));

    }
}
```

使用 ApplicationConText 从 xml 配置文件加载 bean:

```
public class XmlConfigWithApplication{

    public static void main(String[] args){
        ApplicationContext application = new ClassPathXmlApplicationContext(beans.xml"));
         application.getBean("BeanName");
    }
}
```

简而言之，BeanFactory 提供了配置框架和基本的功能， 而 ApplicationContext 为它增加了更强的功能，这些功能中的一些或许更加接近

J2EE 并且围绕企业级应用。一般来说，ApplicationContext 是 BeanFactory 的完全超集， 任何 BeanFactory 功能和行为的描述也同样被认为适用于 ApplicationContext

相对于 BeanFactory 而言,ApplicationContext 提供了以下扩展功能.

(a) 国际化支持 (b) 资源访问

(c) 事件传播 (d) 多实例加载

2. 写一段程序，使用 springAPI 读取 classpath 下的一个 xml 文件，并解析

(1)Resource resource=new ClassPathResource("appcontext.xml");
BeanFactory factory=new XmlBeanFactory(resource);

(2)ClassPathXmlApplicationContext appcontext=new ClassPathXmlApplicationContext("appcontext.xml");
BeanFactory factory=(BeanFactory)appcontext;

4. 说说在 hibernate 中使用 Integer 做映射和使用 int 做映射之间有什么差别

Integer code 和 int code;的区别:
Integer 是对象. code = null; 对象可以为空.
int 是普通类型, 不可能 = null.
根据你的数据库 code 是可以空的, 故应该映射成 Integer.
你没理由 hbm.xml 里写 Integer, 类里却写 int

(1)使用 Spring 如何简化了 Hibernate 编码?
通过 org.springframework.orm.hibernate3.support.HibernateDaoSupport 类支持数据库操作,且封装了事务.

```
public class AccountDAO extends HibernateDaoSupport     implements IAccountDAO{
```

(2)Spring 如何与 Struts 集成?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans >
        <form-bean name="regForm" type="demo.form.RegForm" />

    </form-beans>

    <global-exceptions />
    <global-forwards />
```

```
<action-mappings >
    <action
        attribute="regForm"
        name="regForm"
        path="/reg"
        scope="request"
        type="org.springframework.web.struts.DelegatingActionProxy">
        <forward name="ok" path="/ok.jsp" />
    </action>
    </action-mappings>

    <message-resources parameter="demo.ApplicationResources" />
    <plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
          <set-property property="contextConfigLocation"
value="/WEB-INF/classes/applicationContext.xml" />
    </plug-in>
</struts-config>
```

(3)如何使用 Spring2.0 实现声明式事务？

```
<!--通用事务管理器-->
<bean id="TransactionManager"
      class="org.springframework.orm.hibernate3.HibernateTransactionManager">
      <property name="sessionFactory" ref="sessionFactory" />
</bean>

<!--声明一个通知，用以指出要管理哪些事务方法及如何管理-->
<tx:advice id="txAdvice" transaction-manager="TransactionManager">
      <tx:attributes>
            <!-- 对 get/load/search 开头的方法要求只读事务 -->
            <tx:method name="get*" propagation="SUPPORTS"
                  read-only="true" />
            <tx:method name="load*" propagation="SUPPORTS"
                  read-only="true" />
            <tx:method name="search*" propagation="SUPPORTS"
                  read-only="true" />
            <!-- 对其它方法要求事务 -->
                  <tx:method name="*" propagation="REQUIRED" />
      </tx:attributes>
</tx:advice>

<!--声明一个 config，用以将通知和目标业务类关联起来-->
<aop:config>
      <!-- 添加事务支持,因为前面配置的 transactionManager 是专对 Hibernate 的事务管理
器-->
```

```
        <aop:pointcut id="bizMethods" expression="execution(* demo.*.*(..))" />
        <!-- 织入 -->
        <aop:advisor advice-ref="txAdvice" pointcut-ref="bizMethods" />
</aop:config>
```

(4)依赖注入的好处是？

程序可扩展性更强;

利于并行开发；

(5)Spring 怎么实现依赖注入？

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

        <bean id="userdao" class="demo.dao.UserDAO"/>

        <bean id="usermanager" class="demo.biz.UserManager">
                <property name="userdao" ref="userdao"/>
        </bean>
</beans>
```

(6)“面向方面编程”的好处是？

将程序中涉及的公共问题集中解决

(7)和 SQL 相比，HQL 有哪些特点？

HQL 是面向对象的查询语言。select Fw 表示查询 Fw 对象

(8)如何配置单向多对一关联？

```
<class name="Jd" table="TBL_JD">
                <id name="jdid" column="jdid" type="long">
                        <generator class="identity" />
                </id>
                <property name="jdname" column="jd" type="string" />
                <many-to-one name="qx" class="Qx" column="qxid" />
</class>
```

(9)如何配置单向一对多关联？

```
<class name="Qx" table="TBL_QX">
                <id name="qxid" column="qxid" type="long">
                        <generator class="native" />
                </id>
```

```
        <property name="qxname" column="qx" type="string" />
        <set name="jds" >
            <key column="qxid" />
            <one-to-many class="Jd" />
        </set>
</class>
```

(10)如何配置双向一对多关联？

```
<class name="Jd" table="TBL_JD">
        <id name="jdid" column="jdid" type="long">
            <generator class="identity" />
        </id>
        <property name="jdname" column="jd" type="string" />
        <many-to-one name="qx" class="Qx" column="qxid" />
</class>
<class name="Qx" table="TBL_QX">
        <id name="qxid" column="qxid" type="long">
            <generator class="native" />
        </id>
        <property name="qxname" column="qx" type="string" />
        <set name="jds" >
        <key column="qxid" />
        <one-to-many class="Jd" />
            </set>
</class>
```