**\* Spring的优点有什么?**

1. Spring是分层的架构，你可以选择使用你需要的层而不用管不需要的部分
2. Spring是POJO编程，POJO编程使得可持续构建和可测试能力提高
3. 依赖注入和IoC使得JDBC操作简单化
4. Spring是开源的免费的
5. Spring使得对象管理集中化合简单化

**\* 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：Type2 IoC: Setter injection对象创建之后，将被依赖对象通过set方法设置进去  
方式三：Type3 IoC: Constructor injection对象创建时，被依赖对象以构造方法参数的方式注入  
Spring的方式

**\* 简单描述下IOC(inversion of control)的理解**

一个类需要用到某个接口的方法，我们需要将类A和接口B的实现关联起来，最简单的方法是类A中创建一个对于接口B的实现C的实例，但这种方法显然两者的依赖（Dependency）太大了。而IoC的方法是只在类A中定义好用于关联接口B的实现的方法，将类A，接口B和接口B的实现C放入IoC的 容器（Container）中，通过一定的配置由容器（Container）来实现类A与接口B的实现C的关联。

**\* Spring对很多ORM框架提供了很好支持，描述下在spring使用hibernate的方法**

在context中定义DataSource，创建SessionFactoy，设置参数；DAO类继承HibernateDaoSupport，实现具体接口，从中获得HibernateTemplate进行具体操作。在使用中如果遇到OpenSessionInView的问题，可以添加OpenSessionInViewFilter或OpenSessionInViewInterceptor

**\* 请介绍下spring的事务管理**

spring提供了几个关于事务处理的类：  
TransactionDefinition //事务属性定义  
TranscationStatus //代表了当前的事务，可以提交，回滚。  
PlatformTransactionManager这个是spring提供的用于管理事务的基础接口，其下有一个实现的抽象类AbstractPlatformTransactionManager,我们使用的事务管理类例如DataSourceTransactionManager等都是这个类的子类。

一般事务定义步骤：

TransactionDefinition td = new TransactionDefinition();  
TransactionStatus ts = transactionManager.getTransaction(td);  
try  
{ //do sth  
transactionManager.commit(ts);  
}  
catch(Exception e){transactionManager.rollback(ts);}

spring提供的事务管理可以分为两类：编程式的和声明式的。编程式的，比较灵活，但是代码量大，存在重复的代码比较多；声明式的比编程式的更灵活。

编程式主要使用transactionTemplate。省略了部分的提交，回滚，一系列的事务对象定义，需注入事务管理对象.  
void add()  
{  
transactionTemplate.execute( new TransactionCallback(){  
pulic Object doInTransaction(TransactionStatus ts)  
{ //do sth}  
}  
}

声明式：

使用TransactionProxyFactoryBean:

PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED,readOnly

围绕Poxy的动态代理 能够自动的提交和回滚事务  
org.springframework.transaction.interceptor.TransactionProxyFactoryBean

PROPAGATION\_REQUIRED–支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。

PROPAGATION\_SUPPORTS–支持当前事务，如果当前没有事务，就以非事务方式执行。

PROPAGATION\_MANDATORY–支持当前事务，如果当前没有事务，就抛出异常。

PROPAGATION\_REQUIRES\_NEW–新建事务，如果当前存在事务，把当前事务挂起。

PROPAGATION\_NOT\_SUPPORTED–以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

PROPAGATION\_NEVER–以非事务方式执行，如果当前存在事务，则抛出异常。

PROPAGATION\_NESTED–如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则进行与PROPAGATION\_REQUIRED类似的操作。

**\* 如何在spring的applicationContext.xml使用JNDI而不是DataSource**

可以使用”org.springframework.jndi.JndiObjectFactoryBean”来实现。示例如下：  
<bean id=”dataSource”>  
    <property name=”jndiName”>  
        <value>java:comp/env/jdbc/appfuse</value>  
    </property>  
</bean>

**\* 在spring中是如何配置数据库驱动的**

org.springframework.jdbc.datasource.DriverManagerDataSource”数据源来配置数据库驱动。示例如下：  
<bean id=”dataSource”>  
    <property name=”driverClassName”>  
        <value>org.hsqldb.jdbcDriver</value>  
    </property>  
    <property name=”url”>  
        <value>jdbc:hsqldb:db/appfuse</value>  
    </property>  
    <property name=”username”><value>sa</value></property>  
    <property name=”password”><value></value></property>  
</bean>

**\* spring中的applicationContext.xml能不能改为其他名字**

ContextLoaderListener是一个ServletContextListener, 它在你的web应用启动的时候初始化。缺省情况下， 它会在WEB-INF/applicationContext.xml文件找Spring的配置。 你可以通过定义一个<context-param>元素名字为”contextConfigLocation”来改变Spring配置文件的位置。示例如下：

<listener>  
    <listener-class>org.springframework.web.context.ContextLoaderListener

    <context-param>  
        <param-name>contextConfigLocation</param-name>  
        <param-value>/WEB-INF/xyz.xml</param-value>  
    </context-param>

    </listener-class>  
</listener>

**\* 在web中如何配置spring**

在J2EE的web应用里面配置spring非常简单，最简单的只需要把spring得ContextLoaderListener添加到你的web.xml文件里面就可以了，示例如下：  
<listener>  
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>

**\* 在spring中如何定义hibernate Mapping？**

添加hibernate mapping 文件到web/WEB-INF目录下的applicationContext.xml文件里面。示例如下：  
<property name=”mappingResources”>  
    <list>  
        <value>org/appfuse/model/User.hbm.xml</value>  
    </list>  
</property>

**\* 两种依赖注入的类型是什么?**

两种依赖注入的类型分别是setter注入和构造方法注入。

**setter注入**： 一般情况下所有的java bean, 我们都会使用setter方法和getter方法去设置和获取属性的值，示例如下：  
public class namebean {  
     String      name;    
     public void setName(String a) {  
        name = a; }  
     public String getName() {  
        return name; }  
    }  
我们会创建一个bean的实例然后设置属性的值，spring的配置文件如下：  
<bean id=”bean1″  >  
   <property   name=”name” >  
       <value>tom</value>  
   </property>  
</bean>  
Spring会调用setName方法来只是name熟悉为tom  
**构造方法注入**：构造方法注入中，我们使用带参数的构造方法如下：  
public class namebean {  
     String name;  
     public namebean(String a) {  
        name = a;  
     }     
}  
我们会在创建bean实例的时候以new namebean(”tom”)的方式来设置name属性, Spring配置文件如下：  
<bean id=”bean1″ >  
    <constructor-arg>  
       <value>My Bean Value</value>  
   </constructor-arg>  
</bean>  
使用constructor-arg标签来设置构造方法的参数。

**\* 解释一下Dependency Injection(DI)和IOC（inversion of control）?**

参考答案：依赖注入DI是一个程序设计模式和架构模型， 一些时候也称作控制反转，尽管在技术上来讲，依赖注入是一个IOC的特殊实现，依赖注入是指一个对象应用另外一个对象来提供一个特殊的能力，例如：把一个数据库连接已参数的形式传到一个对象的结构方法里面而不是在那个对象内部自行创建一个连接。控制反转和依赖注入的基本思想就是把类的依赖从类内部转化到外部以减少依赖  
应用控制反转，对象在被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用，传递给它。也可以说，依赖被注入到对象中。所以，控制反转是，关于一个对象如何获取他所依赖的对象的引用，这个责任的反转

**\* Spring中BeanFactory和ApplicationContext的作用和区别**

作用：

1. BeanFactory负责读取bean配置文档，管理bean的加载，实例化，维护bean之间的依赖关系，负责bean的声明周期。  
2. ApplicationContext除了提供上述BeanFactory所能提供的功能之外，还提供了更完整的框架功能：

a. 国际化支持  
b. 资源访问：Resource rs = ctx. getResource(”classpath:config.properties”), “file:c:/config.properties”  
c. 事件传递：通过实现ApplicationContextAware接口  
3. 常用的获取ApplicationContext的方法：  
FileSystemXmlApplicationContext：从文件系统或者url指定的xml配置文件创建，参数为配置文件名或文件名数组  
ClassPathXmlApplicationContext：从classpath的xml配置文件创建，可以从jar包中读取配置文件  
WebApplicationContextUtils：从web应用的根目录读取配置文件，需要先在web.xml中配置，可以配置监听器或者servlet来实现  
<listener>  
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>  
<servlet>  
<servlet-name>context</servlet-name>  
<servlet-class>org.springframework.web.context.ContextLoaderServlet</servlet-class>  
<load-on-startup>1</load-on-startup>  
</servlet>  
这两种方式都默认配置文件为web-inf/applicationContext.xml，也可使用context-param指定配置文件  
<context-param>  
<param-name>contextConfigLocation</param-name>  
<param-value>/WEB-INF/myApplicationContext.xml</param-value>  
</context-param>

**\* 在web环境下如何配置applicationContext.xml文件**

<listener>  
  <listener-class>  
   org.springframework.web.context.ContextLoaderListener  
  </listener-class>  
 </listener>  
 或：  
 <servlet>  
  <servlet-name>context</servlet-name>  
   <servlet-class>  
    org.springframework.web.context.ContextLoaderServlet  
   </servlet-class>  
  <load-on-startup>1</load-on-startup>  
 </servlet>  
 通过如下方法取出applicationContext实例:  
 ApplicationContext ac=WebApplicationContextUtils.getWebApplicationContext(this.getServletContext);

**\* 如何配置spring + struts？**

在struts-config.xml加入一个插件，通过它加载applicationContext.xml  
在struts-config.xml修改action-mapping标记，具体action交给了DelegateActionProxy¬  
通过DelegateActionProxy进入一spring的环境。υ  
在spring的applicationContext.xml加入¬<bean name=”/login” class=”" singleton=”false”/>

**\* Spring 和 hibernate的配置文件中的主要类型有哪些？如何配置?**

在myeclipse中先加入spring环境再加入hibernate环境。  
 如果spring与hibernate结合在一起可以不需要hibernate.cfg.xml文件是否正确?  
 spring+hibernate的配置文件中的主要类有那些?如何配置?  
  dataSource  
  sessionFactory:hibernate.cfg.xml  
  transactionManager  
  userDao (extends HibernateDaoSupport)   
   sessionFactory  
  facade  
  proxy  
   sessionFactory  
   transactionManager  
   facade

**\* 在spring中如何配置容器的事物管理，相关的类有哪些?**

Datasouce  
   transactionManager  
   userDao要注入  
    Datasouce  
   Proxy代理  
    Target:userDao：代理对象(目标对象)  
    transactionAttributes(那些方法需要事务处理)  
    transactionManager(事务处理服务)

**\* 在spring中如何配代码的事务管理器**

Datasouce  
   transactionManager  
   userDao要注入  
    Datasouce  
    transactionManager  
  通过如下类实现  
    TransactionTemplate  
    JdbcTemplate

**\* Spring中有几种事物管理，分别是什么?**

代码管理的事务处理  
TransactonTemplate的execute方法中的内部类TransactionCallback中的doInTransaction方法中使用。  
public void make()  
 {   
  TransactionTemplate jtm=new TransactionTemplate(this.getTransactionManager());  
  jtm.execute(new myClass1());  
 }  
 public class myClass1 implements TransactionCallback  
 {

  public Object doInTransaction(TransactionStatus trans)  
  {  
   JdbcTemplate jdbc=new JdbcTemplate(dataSource);  
   jdbc.execute(”insert into customer(customerName) values(’b')”);  
   jdbc.execute(”insert into customer(customerName) values(’b')”);  
   return null;  
  }    
 }  
   容器管理的事务处理

**\* spring中的jdbc与传统的jdbc有什么区别?**

Spring的jdbc:节省代码，不管连接(Connection)，不管事务、不管异常、不管关闭(con.close() ps.close )

  JdbcTemplate(dataSource):增、删、改、查  
  TransactionTemplate(transactionManager):进行事务处理

**\* Spring配置的主要标签有什么?有什么作用?**

<beans>  
   <bean id=”” class=”” init=”” destroy=”” singleton=””>  
    <property name=””>  
     <value></value>  
    </property>  
    <property name=””>  
     <ref local></ref>  
    </property>  
   </bean>  
</beans>

**\* 如何在spring中实现国际化?**

在applicationContext.xml加载一个bean  
<bean id=”messageSource” class=”org.springframework.context.support.ResourceBundleMessageSource”>  
  <property name=”basename”>  
   <value>message</value>  
  </property>  
</bean>  
 在src目录下建多个properties文件¬  
 对于非英文的要用native2ascii¬ -encoding gb2312 源  目转化文件相关内容  
 其命名格式是message\_语言\_国家。¬  
 页面中的中显示提示信息，键名取键值。¬  
 当给定国家，系统会自动加载对应的国家的properties信息。¬  
 通过applictionContext.getMessage(“键名”,”参数”,”区域”)取出相关的信息。¬

**\* 在spring中如何实现事件处理**

事件  
  Extends ApplicationEvent  
监听器  
  Implements ApplicationListener  
事件源  
  Implements ApplicationContextAware  
在applicationContext.xml中配置事件源、监听器  
先得到事件源，调用事件源的方法，通知监听器。

**\* 如何将spring加入web容器中**

在web.xml中加入如下同容,在启动web服务器时加载/WEB-INF/applicationContext.xml中的内容。  
<servlet>  
<servlet-name>context</servlet-name>  
<servlet-class>  
org.springframework.web.context.ContextLoaderServlet  
</servlet-class>  
<load-on-startup>1</load-on-startup>  
</servlet>  
通过如下类得到ApplicationContext实例  
   WebApplicationContextUtils.getWebApplicationContext

**\* Spring如何实现资源管理?**

使用  
applicationContext.getResource(“classpath:文件名”):在src根目录下，在类路径下  
applicationContext.getResource(“classpath:/chap01/文件名”): 以src根目录下的基准往下走。  
applicationContext.getResource(“file:c:/a.properties”)：在系统文件目录下。

**\* Spring的ApplicationContext的作用?**

beanFactory  
国际化(getMesage)  
资源管理:可以直接读取一个文件的内容(getResource)  
加入web框架中(加入一个servlet或监听器)  
事件处理

**\* spring的核心是什么，各有什么作用？**

BeanFactory：产生一个新的实例，可以实现单例模式  
BeanWrapper：提供统一的get及set方法  
ApplicationContext:提供框架的实现，包括BeanFactory的所有功能

**\* Spring中aop的关键名词有哪些？各有什么作用?**

拦截器: 代理  
装备(advice)  
目标对象  
关切点:条件  
连接点:方法、属性

**\* Spring与struts的区别？**

strusts：是一种基于MVC模式的一个web层的处理。  
Spring:提供了通用的服务，ioc/di aop,关心的不仅仅web层，应当j2ee整体的一个服务，可以很容易融合不同的技术struts hibernate ibatis ejb remote springJDBC springMVC

**\* spring与struts的面试题**

1.struts  
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

2.MVC，分析一下struts是如何实现MVC的

3.struts中的几个关键对象的作用(说说几个关键对象的作用)

4.spring  
说说AOP和IOC的概念以及在spring中是如何应用的

5.Hibernate有哪几种查询数据的方式

6.load()和get()的区别

回答：  
1.Struts1 Action是单例模式并且必须是线程安全的，因为仅有Action的一个实例来处理所有的请求。单例策略限制了Struts1 Action能作的事，并且要在开发时特别小心。Action资源必须是线程安全的或同步的。  
Struts2 Action对象为每一个请求产生一个实例，因此没有线程安全问题。（实际上，servlet容器给每个请求产生许多可丢弃的对象，并且不会导致性能和垃圾回收问题）

2.struts是用一组类,servlet 和jsp规范实现mvc的

3.ActionFrom ActionServlet Action struts-config.xml

4.spring的核心就是IOC,通过指定对象的创建办法,描述对象与服务之间的关系,而不生成对象

5.3种,hql 条件查询() 原生sql

6.load()方法认为该数据一定存在,可以放心的使用代理来延时加载 ,如果使用过程中发现了问题,就抛出异常;  
get()方法一定要获取到真实的数据,否则返回null

**\* Spring, hibernate ,struts面试题**

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：

\* 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?

\* Hibernate2延迟加载实现：a)实体对象  b)集合（Collection）  
\* 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。  
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、

说下Hibernate的缓存机制

\* 内部缓存存在Hibernate中又叫一级缓存，属于应用事物级缓存  
\* 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  
Hibernate的查询方式  
Sql、Criteria,object comptosition  
Hql：

\* 属性查询  
\* 参数查询、命名参数查询  
\* 关联查询  
\* 分页查询  
\* 统计函数

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

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：  
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。

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。

单例模式

Factory(工厂模式)：

定义一个基类===》实现基类方法（子类通过不同的方法）===》定义一个工厂类（生成子类实例）

===》开发人员调用基类方法

Proxy(代理模式)

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。

**\* 一些spring与hibernate的面试题**

1、 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。

2、 Spring的Bean有多种作用域，包括：

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

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

4、 Hibernate中的update()和saveOrUpdate()的区别

5、 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。

答案：

1、好莱坞原则€€€€不要打电话找我，我会打给你的。IoC将创建的职责从应用程序代码搬到了框架中。Spring对Setter注入和构造方法注入提供支持。（详见http://martinfowler.com/articles/injection.html，以及http: //www.redsaga.com/spring\_ref/2.0/html/beans.html#beans-factory- collaborators）

2、 除application（详见Spring framework 2.0 Reference的3.4节bean的作用域）

3、 Spring是完整的一站式框架，而Struts仅是MVC框架，且着重于MVC中的C。Spring有三种方式整合Struts：使用 Spring 的 ActionSupport 类整合 Struts；使用 Spring 的 DelegatingRequestProcessor 覆盖 Struts 的 RequestProcessor；将 Struts Action 管理委托给 Spring 框架，动作委托最好。（详见使用Spring 更好地处理Struts 动作）

Spring 2.0新增一种方式：AutowiringRequestProcessor。（详见http://www.javaeye.com/topic/24239）

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

如果对象在该session中已经被持久化，不进行操作；

对象的标识符属性(identifier property)在数据库中不存在或者是个暂时的值，调用save()方法保存它；

如果session中的另一个对象有相同的标识符抛出一个异常；

以上皆不符合则调用update()更新之。

5、 在context中定义DataSource，创建SessionFactoy，设置参数；DAO类继承HibernateDaoSupport，实现具体接口，从中获得HibernateTemplate进行具体操作。

在使用中如果遇到OpenSessionInView的问题，可以添加OpenSessionInViewFilter或OpenSessionInViewInterceptor。（详见Spring framework 2.0 Reference的12.2节Hibernate）

声明式事务需声明事务管理器，在context中设置指定属性，用确定和。

**\* 在spring中如何更加高效的使用JDBC**

使用Spring框架提供的模板类JdbcTemplete可以是JDBC更加高效

代码如下：JdbcTemplate template = new JdbcTemplate(myDataSource);

DAO类的例子：

public class StudentDaoJdbc implements StudentDao {  
private JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {  
this.jdbcTemplate = jdbcTemplate;  
}  
more..  
}

配置文件：

<bean id=”jdbcTemplate” class=”org.springframework.jdbc.core.JdbcTemplate”>  
<property name=”dataSource”>  
<ref bean=”dataSource”/>  
</property>  
</bean>  
<bean id=”studentDao” class=”StudentDaoJdbc”>  
<property name=”jdbcTemplate”>  
<ref bean=”jdbcTemplate”/>  
</property>  
</bean>  
<bean id=”courseDao” class=”CourseDaoJdbc”>  
<property name=”jdbcTemplate”>  
<ref bean=”jdbcTemplate”/>  
</property>  
</bean>

**\* 在spring如何创建一个数据连接池**

<bean id=”dataSource” class=”org.apache.commons.dbcp.BasicDataSource”>  
<property name=”driver”>  
<value>${db.driver}</value>  
</property>  
<property name=”url”>  
<value>${db.url}</value>  
</property>  
<property name=”username”>  
<value>${db.username}</value>  
</property>  
<property name=”password”>  
<value>${db.password}</value>  
</property>  
</bean>

**\* 在spring中如何配置一个bean来从JNDI到dataSource**

<bean id=”dataSource” class=”org.springframework.jndi.JndiObjectFactoryBean”>  
<property name=”jndiName”> <value>java:comp/env/jdbc/myDatasource</value>  
</property>  
</bean>

**\* 请介绍下spring中bean的作用域**

在spring2.0之前bean只有2种作用域即：singleton(单例)、non-singleton（也称 prototype），Spring2.0以后，增加了session、request、global session三种专用于Web应用程序上下文的Bean。因此，默认情况下Spring2.0现在有五种类型的Bean。

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”singleton”/>

这里的scope就是用来配置spring bean的作用域，它标识bean的作用域。

在spring2.0之前bean只有2种作用域即：singleton(单例)、non-singleton（也称 prototype），Spring2.0以后，增加了session、request、global session三种专用于Web应用程序上下文的Bean。因此，默认情况下Spring2.0现在有五种类型的Bean。当然，Spring2.0对 Bean的类型的设计进行了重构，并设计出灵活的Bean类型支持，理论上可以有无数多种类型的Bean，用户可以根据自己的需要，增加新的Bean类型，满足实际应用需求。

1、singleton作用域

当一个bean的作用域设置为singleton，那么Spring IOC容器中只会存在一个共享的bean实例，并且所有对bean的请求，只要id与该bean定义相匹配，则只会返回bean的同一实例。换言之，当把一个bean定义设置为singleton作用域时，Spring IOC容器只会创建该bean定义的唯一实例。这个单一实例会被存储到单例缓存（singleton cache）中，并且所有针对该bean的后续请求和引用都将返回被缓存的对象实例，这里要注意的是singleton作用域和GOF设计模式中的单例是完全不同的，单例设计模式表示一个ClassLoader中只有一个class存在，而这里的singleton则表示一个容器对应一个bean，也就是说当一个bean被标识为singleton时候，spring的IOC容器中只会存在一个该bean。

配置实例：

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”singleton”/>

或者

<bean id=”role” class=”spring.chapter2.maryGame.Role” singleton=”true”/>

2、prototype

prototype作用域部署的bean，每一次请求（将其注入到另一个bean中，或者以程序的方式调用容器的getBean()方法）都会产生一个新的bean实例，相当于一个new的操作，对于prototype作用域的bean，有一点非常重要，那就是Spring不能对一个 prototype bean的整个生命周期负责，容器在初始化、配置、装饰或者是装配完一个prototype实例后，将它交给客户端，随后就对该prototype实例不闻不问了。不管何种作用域，容器都会调用所有对象的初始化生命周期回调方法，而对prototype而言，任何配置好的析构生命周期回调方法都将不会被调用。清除prototype作用域的对象并释放任何prototype bean所持有的昂贵资源，都是客户端代码的职责。（让Spring容器释放被singleton作用域bean占用资源的一种可行方式是，通过使用 bean的后置处理器，该处理器持有要被清除的bean的引用。）

配置实例：

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”prototype”/>

或者

<beanid=”role” class=”spring.chapter2.maryGame.Role” singleton=”false”/>

3、request

request表示该针对每一次HTTP请求都会产生一个新的bean，同时该bean仅在当前HTTP request内有效，配置实例：

request、session、global session使用的时候，首先要在初始化web的web.xml中做如下配置：

如果你使用的是Servlet 2.4及以上的web容器，那么你仅需要在web应用的XML声明文件web.xml中增加下述ContextListener即可：

<web-app>  
…  
<listener>  
<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>  
</listener>  
…  
</web-app>

如果是Servlet2.4以前的web容器,那么你要使用一个javax.servlet.Filter的实现：

<web-app>  
..  
<filter>  
<filter-name>requestContextFilter</filter-name>  
<filter-class>org.springframework.web.filter.RequestContextFilter</filter-class>  
</filter>  
<filter-mapping>  
<filter-name>requestContextFilter</filter-name>  
<url-pattern>/\*</url-pattern>  
</filter-mapping>  
…  
</web-app>

接着既可以配置bean的作用域了：

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”request”/>  
4、session

session作用域表示该针对每一次HTTP请求都会产生一个新的bean，同时该bean仅在当前HTTP session内有效，配置实例：

配置实例：

和request配置实例的前提一样，配置好web启动文件就可以如下配置：

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”session”/>

5、global session

global session作用域类似于标准的HTTP Session作用域，不过它仅仅在基于portlet的web应用中才有意义。Portlet规范定义了全局Session的概念，它被所有构成某个 portlet web应用的各种不同的portlet所共享。在global session作用域中定义的bean被限定于全局portlet Session的生命周期范围内。如果你在web中使用global session作用域来标识bean，那么，web会自动当成session类型来使用。

配置实例：

和request配置实例的前提一样，配置好web启动文件就可以如下配置：

<bean id=”role” class=”spring.chapter2.maryGame.Role” scope=”global session”/>

6、自定义bean装配作用域

在spring 2.0中作用域是可以任意扩展的，你可以自定义作用域，甚至你也可以重新定义已有的作用域（但是你不能覆盖singleton和 prototype），spring的作用域由接口org.springframework.beans.factory.config.Scope来定义，自定义自己的作用域只要实现该接口即可，下面给个实例：

我们建立一个线程的scope，该scope在表示一个线程中有效，代码如下：

publicclass MyScope implements Scope …{  
privatefinal ThreadLocal threadScope = new ThreadLocal() …{  
protected Object initialValue() …{  
returnnew HashMap();  
}  
};  
public Object get(String name, ObjectFactory objectFactory) …{  
Map scope = (Map) threadScope.get();  
Object object = scope.get(name);  
if(object==null) …{  
object = objectFactory.getObject();  
scope.put(name, object);  
}  
return object;  
}  
public Object remove(String name) …{  
Map scope = (Map) threadScope.get();  
return scope.remove(name);  
}  
publicvoid registerDestructionCallback(String name, Runnable callback) …{  
}  
public String getConversationId() …{  
// TODO Auto-generated method stub  
returnnull;  
}  
}

**\* 请介绍 一下spring的bean的生命周期**

一、Bean的定义  
Spring通常通过配置文件定义Bean。如：

<?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=”HelloWorld” class=”com.pqf.beans.HelloWorld”>  
<property name=”msg”>  
<value>HelloWorld</value>  
</property>  
</bean>  
</beans>

这个配置文件就定义了一个标识为 HelloWorld 的Bean。在一个配置文档中可以定义多个Bean。

二、Bean的初始化  
有两种方式初始化Bean。  
1、在配置文档中通过指定init-method 属性来完成  
在Bean的类中实现一个初始化Bean属性的方法，如init()，如：  
public class HelloWorld{  
public String msg=null;  
public Date date=null;

public void init() {  
msg=”HelloWorld”;  
date=new Date();  
}  
……  
}  
然后，在配置文件中设置init-mothod属性：  
<bean id=”HelloWorld” class=”com.pqf.beans.HelloWorld” init-mothod=”init” >  
</bean>

2、实现 org.springframwork.beans.factory.InitializingBean接口  
Bean实现InitializingBean接口，并且增加 afterPropertiesSet() 方法：

public class HelloWorld implement InitializingBean {  
public String msg=null;  
public Date date=null;

public void afterPropertiesSet() {  
msg=”向全世界问好！”;  
date=new Date();  
}  
……  
}

那么，当这个Bean的所有属性被Spring的BeanFactory设置完后，会自动调用afterPropertiesSet()方法对Bean进行初始化，于是，配置文件就不用指定 init-method属性了。

三、Bean的调用  
有三种方式可以得到Bean并进行调用：  
1、使用BeanWrapper  
HelloWorld hw=new HelloWorld();  
BeanWrapper bw=new BeanWrapperImpl(hw);  
bw.setPropertyvalue(”msg”,”HelloWorld”);  
system.out.println(bw.getPropertyCalue(”msg”));

2、使用BeanFactory  
InputStream is=new FileInputStream(”config.xml”);  
XmlBeanFactory factory=new XmlBeanFactory(is);  
HelloWorld hw=(HelloWorld) factory.getBean(”HelloWorld”);  
system.out.println(hw.getMsg());

3、使用ApplicationConttext  
ApplicationContext actx=new FleSystemXmlApplicationContext(”config.xml”);  
HelloWorld hw=(HelloWorld) actx.getBean(”HelloWorld”);  
System.out.println(hw.getMsg());

四、Bean的销毁  
1、使用配置文件中的 destory-method 属性  
与初始化属性 init-methods类似，在Bean的类中实现一个撤销Bean的方法，然后在配置文件中通过 destory-method指定，那么当bean销毁时，Spring将自动调用指定的销毁方法。

2、实现 org.springframwork.bean.factory.DisposebleBean接口  
如果实现了DisposebleBean接口，那么Spring将自动调用bean中的Destory方法进行销毁，所以，Bean中必须提供Destory方法。

**\* Spring中如何获取bean**

通过xml配置文件

bean配置在xml里面，spring提供多种方式读取配置文件得到ApplicationContext.

第一种方式：FileSystemXmlApplicationContext

通过程序在初始化的时候，导入Bean配置文件，然后得到Bean实例:  
ApplicationContext ac = new FileSystemXmlApplicationContext(”applicationContext.xml”)  
ac.getBean(”beanName”);

第二种方式：WebApplicationContextUtil

在B/S系统中,通常在web.xml初始化bean的配置文件，然后由WebAppliCationContextUtil得到ApplicationContext.例如：  
ApplicationContext ctx = WebApplicationContextUtils.getRequiredWebApplicationContext(ServletContext sc);  
ApplicationContext ctx =   WebApplicationContextUtils.getWebApplicationContext(ServletContext sc);

其中 servletContext sc 可以具体 换成 servlet.getServletContext()或者 this.getServletContext() 或者 request.getSession().getServletContext();

另外，由于spring是注入的对象放在ServletContext中的，所以可以直接在ServletContext取出WebApplicationContext 对象：

WebApplicationContext webApplicationContext = (WebApplicationContext) servletContext.getAttribute(WebApplicationContext.ROOT\_WEB\_APPLICATION\_CONTEXT\_ATTRIBUTE);

**\* Spring框架有哪几部分组成?**

Spring框架有七个模块组成组成，这7个模块(或组件)均可以单独存在，也可以与其它一个或多个模块联合使用，主要功能表现如下：

**☞ Spring 核心容器（Core）：**提供Spring框架的基本功能。核心容器的主要组件是BeanFactory，她是工厂模式的实现。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代码程序分开。

**☞ Spring AOP：**通过配置管理特性，Spring AOP模块直接面向方面的编程功能集成到了Spring框架中，所以可以很容易的使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，不用依赖于EJB组件，就可以将声明性事务管理集成到应用程序中。

**☞ Spring ORM：**Spring框架集成了若干ORM框架,从而提供了ORM的对象关系工具,其中包括 JDO、Hibernate、iBatis和TopLink。所有这些都遵从Spring的通用事务和DAO异常层结构。

**☞ Spring DAO：**JDBC DAO抽象层提供了有意义的异常层次的结构，可用该结构来管理异常处理和不同数据供应商抛出的异常错误信息。异常层次结构简化了错误处理，并且大大的降低 了需要编写的异常代码数量（例如，打开和关系连接）。Spring DAO的面向JDBC的异常遵从通用的DAO异常层结构。

**☞ Spring WEB：**Web上下文模块建立在上下文模块（Context）的基础之上，为基于Web服务的应用程序提供了上下文的服务。所以Spring框架支持 Jakarta Struts的集成。Web模块还简化了处理多部分请求及将请求参数绑定到域对象的工作。

**☞ Spring上下文（Context）：**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 MVC：**Spring的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容纳的大量视图技术，包括JSP、Velocity、Tiles、iText和Pol

**\* 使用spring有什么好处?**

◆Spring能有效地组织你的中间层对象,无论你是否选择使用了EJB。如果你仅仅使用了Struts或其他的包含了J2EE特有APIs的framework，你会发现Spring关注了遗留下的问题，。

◆Spring能消除在许多工程上对Singleton的过多使用。根据我的经验，这是一个主要的问题，它减少了系统的可测试性和面向对象特性。  
◆Spring能消除使用各种各样格式的属性定制文件的需要,在整个应用和工程中，可通过一种 一致的方法来进行配置。曾经感到迷惑，一个特定类要查找迷幻般的属性关键字或系统属性,为此不得不读Javadoc乃至源编码吗？有了Spring，你可 很简单地看到类的JavaBean属性。倒置控制的使用(在下面讨论)帮助完成这种简化。  
◆Spring能通过接口而不是类促进好的编程习惯，减少编程代价到几乎为零。  
◆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。  
◆使用Spring构建的应用程序易于单元测试。  
◆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。  
◆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，它们适于许多web应用。例如,Spring能使用AOP提供声明性事务而不通过使用EJB容器，如果你仅仅需要与单个的数据库打交道，甚至不需要JTA实现。  
■Spring为数据存取提供了一致的框架,不论是使用JDBC或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解决你的问题。这些特性是有很大价值的。  
总结起来，Spring有如下优点：  
◆低侵入式设计，代码污染极低  
◆ 独立于各种应用服务器，可以真正实现Write Once,Run Anywhere的承诺  
◆Spring的DI机制降低了业务对象替换的复杂性  
◆Spring并不完全依赖于Spring，开发者可自由选用Spring框架的部分或全部

**\* 什么是spring，它有什么特点?**

Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。

◆**轻量**——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并 且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

◆**控制反转**——Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不 是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。  
◆**面向切面**——Spring提供了面向切面编程的丰富支持，允许通过分离应用的 业务逻辑与系统级服务（例如审计（auditing）和事务（）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们 并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。  
◆**容器**——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是 一种容器，你可以配置你的每个bean如何被创建——基于一个可配置原型（prototype），你的bean可以创建一个单独的实例或者每次需要时都生 成一个新的实例——以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。  
◆**框架**——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了你。