一,为什么要用hibernate

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是基于Jdbc的主流的持久化框架，是一个优秀的ORM实现，它很大程度的简化了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，是一个轻量级的框架；映射的灵活性很出色，能支持各种关系的数据库，从一对一到多对多的各种复杂关系。

二 hibernate如何延迟加载

1．Hibernate2延迟加载实现：a)实体对象 b)集合Conllection

2．Hibernate3提供了属性的延迟加载功能

当hibernate在查询数据库的时候，数据并没有存在于内存中，当程序真正对数据库操作是，对象才存在于内存中，就实现了延迟加载；它节省了服务器的内存开销，从而提高了服务器的性能。

三hibernate怎么样实现类之间的关系（如：一对一，多对多）

类与类之间的关系主要体现在表与表之间的关系操作，它们都是对对象进行操作，我们程序中把所有的表和类都映射在一起；它们通过配置文件中的many-one，one-many，many-many来实现

四 说下hibernate的缓存机制

1. 内部缓存存在hibernate中又叫一级缓存，属于应用事务级缓存
2. 二级缓存：

A 应用级缓存

B 分布式缓存

条件：数据部会被第三方修改，数据大小在可接受的范围，数据跟新频率低，同一数据被系统频繁使用，非关键性数据

C第三方缓存的实现

五 hibernate的查询方式

Sql Criteria, t comptosition

HQL

1. 属性的查询
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

六 如何优化hibernate

1. 使用双向一对多关联，不适用单项一对多关联
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，不适用集合缓存
5. 一对多集合使用bag，多对多集合使用Set
6. 继承类使用显示多态
7. 表字段要少，关联不要怕多，有二级缓存撑腰

Struts

一Struts工作机制？为什么要用Struts？

工作机制：

在web应用启动是就会加载初始化ActionSevlet,ActionServlet

从Struts-config.xml文件中读取配置信息，把他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，把客户提叫的表单数据保存到ActionForm中
3. 根据配置信息决定是否需要表单验证，如果需要就调用ActionForm的validate()方法
4. 如果ActionForm的validate（）方法返回null值或是返回一个不包含ActionMessage()的ActuibEErrors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，如果响应的Action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ActionForword对象，ActionServlet把客户请求的转发给ActionForword指向的JSP组件
7. ActionForword对象指向JSP生成动态的页面返回给客户

为什么要用Struts:

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，但是用这些技术构建的系统非常的繁乱，所以在此之上，我们需要一个原则，一个把这些技术组织在一起的规则，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

二 说下Struts的validate（）框架是如何验证的

在Struts的配置文件中配置具体的错误提示，再在formBean中的validate（）方法中具体调用

三 说下Struts的设计模式

MVC设计模式：web应用程序启动时会加载并初始化ActionServlet。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，ActionServlet根据Struts-config.xml文件配置好的设置决定是否需要验证，如果需要就调用ActionForm的Validate（）方法，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方法从ActionForm对象中获取数据，完成业务逻辑，返回一个ActionForword对象，ActionServlet再把客户请求转发给ActionforWord对象指定的JSP组件，ActionForword对象指定的JSP组件生产动态的网页，返回给客户。

Spring工作机制以及为什么要用

一 Spring工作机制以及问什么要用

工作机制：

1. springMvc请所有的请求都提交给DispathcherServlet，它会委托应用系统的其他模块负债负债对请求进行真正的处理工作。
2. DispartcherServlet查询一个或者多个Handlmapping，找到处理请求的Controller。
3. DispatcherServlet请求提交到目标Controller。
4. Controllet进行业务逻辑的处理后返回一个ModelAndView
5. DispatcherServlet查询一个或多个ViewResolver视图解析器，找到modelAndView对象指定的试图解析器
6. 视图对象负债渲染返回给客户端

为什么要用：

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（straightconstruction），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 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：JDBCDAO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）。SpringDAO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。

　　共2页: 1 [2]

　　内容导航

　　第 1 页：Hibernate工作原理及用的理由(1) 第 2 页：Hibernate工作原理及用的理由(2)

　　 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 = newXMLBeanFactory(newFileInputSteam("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 publicbooleangetUserCreditHistoryInformation(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);publicvoid 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); publicStringgetFromEmail(); public void setFromEmail(String fromEmail) ;publicString getPassword(); public void setPassword(Stringpassword) ;public String getSmtpHost() ; public voidsetSmtpHost(StringsmtpHost); public String getUserId() ; publicvoid setUserId(StringuserId);

　　您正在阅读：Hibernate、Spring和Struts工作原理及使用理由

　　编辑推荐

　　Hibernate框架如何实现ORM

　　Hibernate入门教程 Hibernate关系映射详解

　　Hibernate结合MYSQL数据库简单教程