 今天在51cto上看见了几个关于struts,spring,hibernte的问题，感觉这些问题还不错，所以我特意一一研究并解答。 [查看问题来源](http://www.blogjava.net/tbwshc/archive/2012/07/29/381270.html#384245)

 Struts2:

        1.为什么每次请求都要创建一个Action对象？

            re: 为了防止线程迸发，如果每次都使用同一个Action进行数据修改和传递的话，容易引起线程迸发，使线程不安全。

        2.ModelDriven拦截器的配置中refreshModelBeforeResult解决了什么问题？

            re:在数据输出之前将数据接收的Bean里面的数据清空。解决了数据混淆的可能性。

        3.为什么在web.xml中配置的StrutsPrepareAndExecuteFilter要过滤所有的请求？

            re:

                No1.可以将数据进行过滤。

                No2.可以对数据进行转码处理，将数据进行统一的编码。

        4.关于ValueStack(值栈)？

            re:堆值，堆里面的值。

             struts2背后的数据传输机制，实际上就是对OGNL的封装，OGNL主要的功能就是赋值与取值（context,root两种方式），Struts2正是通过ValueStack来进行赋值与取值的！

           ValueStack是一个接口，而OgnlValueStack是strtus2中的缺省实现。ValueStack中的数据，分两个部分存放：root和context（这与OGNL中的概念一致），同时                                ValueStack暴露相关的接口：

            void setValue(String expr, Object value);

             Object findValue(String expr);

            用来通过OGNL表达式对ValueStack中的数据进行操作！

ValueStack中的root对象是CompoundRoot，CompoundRoot继承了ArraryList，提供了额外的方法：push()和pop()方法，用来对root对象中所包含的数据进行存取！

public class CompoundRoot extends ArrayList {

public CompoundRoot() {

}

public CompoundRoot(List list) {

super(list);

}

public CompoundRoot cutStack(int index) {

return new CompoundRoot(subList(index, size()));

}

public Object peek() {

return get(0);

}

public Object pop() {

return remove(0);

}

public void push(Object o) {

add(0, o);

}

}

正是通过这两个方法，CompoundRoot变成了一个栈结构！压栈操作，将导致对象被放到CompoundRoot的第0个元素上（第0个元素是栈顶），其它对象被依次往后移动；出栈操作，将导致CompoundRoot的第0个元素被移除（即栈顶元素被弹出），其它对象被依次往前移动！

OGNL不支持多个root对象，而struts2能够支持多个root对象，它对OGNL做了扩展。

如果某个OGNL表达式被传递给ValueStack（即调用ValueStack的setValue或findValue方法），而表达式中包含有对root对象的访问操作，ValueStack将依次从栈顶往栈底搜索CompoundRoot对象中所包含的对象，看哪个对象具有相应的属性，找到之后，立刻返回。

在Struts2中，一个请求在最终到达Action的方法之前，Action对象本身会被压入ValueStack（实际上就是放到ValueStack的CompoundRoot中），所以Action对象是CompoundRoot中的一个元素。

          3.Struts2是如何实现MVC设计模式的？

            re: 所谓MVC模式，即 模型(Model),视图(View)和控制，其目的是实现Web系统的职能分工。其中Model层实现系统中的业务逻辑，通常可以用JavaBean或EJB来实现； View层用于与用户的交互，通常用JSP来实现； Controller层是Model与View之间沟通的桥梁，它可以分派用户的请求并选择恰当的视图以用于显示，同时它也可以解释用户的输入并将它们映射为模型层可执行的操作。

        在Struts2里面是将每次页面的请求进行处理，然后将请求需要输出的数据转发的需要做数据显示的页面，Struts2只是起一个数据接受和转接的功能，就是Controler控制器，而传来数据的页面叫View显示层，Struts2将数据提交给进行处理的类叫Model模型层，专门进行数据处理和数据库的链接。

     Spring：

          1.为什么要用spring?

             re:

                  No1.主要将各个框架进行有效的联系起来，缩短实际编程的时间，起一个润滑剂的作用。

　　            No2.可以将各个框架进行有效的管理和控制，让数据传输中安全。

          2.关于IOC/DI。

            re:   IOC(控制翻转或是依赖注入（DI）):所谓控制反转，即将讲程序的控制权由代码转交给了容器，当某角色（可能是一个java实例，调用者）需要另外一个角色（另外一个java实例，被调用者）的协助时，在传统的程序设计中通常有调用者创建被调用者的实例，但是在spring,创建被调用者不再是由调用者完成，而是交由spring容器完成，然后注入被调用者。

              这样将所有组件在Spring提供的外部容器中加载，提高安全性，减低耦合性，使各个框架或者类之间的依赖性降低。

           3.什么是声明式的事务管理？为什么要使用声明式的事务管理？Spring如何实现声明式的事务管理？

             re: Spring的声明式事务顾名思义就是采用声明的方式来处理事务。这里所说的声明，就是指在配置文件中申明。用在Spring配置文件中声明式的处理事务来代替代码式的处理事务。这样的好处是，事务管理不侵入开发的组件（spring为非侵入式框架），具体来说，业务逻辑对象就不会意识到正在事务管理之中，事实上也应该如此，因为事务管理是属于系统层面的服务，而不是业务逻辑的一部分，如果想要改变事务管理策划的话，也只需要在定义文件中重新配置即可；在不需要事务管理的时候，只要在设定文件上修改一下，即可移去事务管理服务，无需改变代码重新编译，这样维护起来极其方便。

          　声明事物管理主要是将在进行对数据库中数据进行添加或者修改的时候需要执行事物管理。主要是避免在执行数据修改和添加的时候数据添加或者修改不完全，导致数据丢失。

             Spring是使用AOP面向切面的思想\*(面向切面编程，可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术)进行事物管理的。因而声明式事务是以方法为单位，Spring的事务属性自然就在于描述事务应用至方法上的策略，在Spring中事务属性有以下四个参数：

　       　No1.传播行为

    　     No2.隔离级别

 　        No3.只读提示

 　        No4.事务超时期间

           4.把spring和hibernate集成，定义事务管理特性的时候，为何要将除了添加、删除、更新操作之外的方法，即主要执行查询任务的方法定义为read-only？

              re: 因为添加和删除和更新都涉及到数据库的修改，而查询并为涉及数据修改，所以只需要定义只读，这样可以提高效率，进行更加方便的事务管理。而事务管理主要是为添加和删除和更新设计的。

          Hibernate:

             1.请你谈谈你对OR映射的理解？

                 re:将每个实体类都映射为数据库中的一个表，或将数据库中的表映射为一个实体类。

             2.很多人说Hibernate不适合大项目，性能有问题，你是如何理解的？

                re:因为Hibernate属于大型框架，里面对方法的封装比较多，让SQL控制权降低。这样的话，适用性就会降低，在不必要的操作上会浪费时间。但是只要将Hibernate里的配置进行优化，也能提高其性能。（关于hibernate的具体优化请见下回分解吧。）

             3.Hibernate为什么一定要定义一个数据库标识？

                re:可以让映射文件变得简洁，消除无用的噪音(比如TBL\_前缀等)。Hibernate使用的默认策略是几乎什么都不做，所以使用标识的作用就是使数据操作更加简洁和方便.

             4.为什么Hibernate建议你的实体类实现hashCode和equals方法？

               re:因为Hibernate使用了一级和二级缓存，很多查询为了提高效率Hibernate都会先从缓存中进行查找，然后再从数据库进行查找。而HashCode是唯一的，所以这样避免数据操作出现数据混淆的可能，而equals的作用是对比Hibernate缓存中数据是否一致。

             5.谈谈你对Hibernate实体类中的数据库标识与数据库主键之间关系的认识？

               re:数据库标识是为了方便和简洁映射文件，而主键是为了使数据不会重复。

              6.谈谈你对Hibernate关联映射与数据库外键之间关系的认识？

               re：Hibernate在进行映射的时候会将数据库的关系也进行映射将数据库中的外键也使用标签的形式进行管理，这样在人为操作的时候就不需要手动的管理数据库关系了。

              7.调用session.save()方法，hibernate一定会发出insert语句吗？谈谈你的理解

               re:

　               不会，具体执行步骤如下：

　                                　No1. 首先在Session内部缓存中进行查找，如果发现则直接返回。

　　                                No2. 执行实体类对应的Interceptor.isUnsaved方法(如果有的话)，判断对象是否为未保存状态。

　　                                No3. 根据unsaved-value判断对象是否处于未保存状态。

　　                                No4. 如果对象未保存(Transient状态)，则调用save方法保存对象。

　　                                No5. 如果对象未已保存(Detached状态)，则调用update方法将对象与Session重新关联。

              8.调用session.update()方法，hibernate一定会发出update语句吗？谈谈你的理解

               re:

                   不会，具体步骤同上。

               6.请你聊一下以下名词、概念或用法：lazy、lazy=”extra”、inverse、fetch、fetch=”join”、fetch=”subselect”、batch-size

                re:

        lazy懒加载，hibernate映射文件默认的lazy = true.lazy=“extra”extra属性是不大容易重视的,其实它和true差不多,但有个小的智能的地方是,即调用集合的size/contains等方法的时候，hibernate并不会去加载整个集合的数据，而是发出一条聪明的SQL语句，以便获得需要的值，只有在真正需要用到这些集合元素对象数据的时候，才去发出查询语句加载所有对象的数据

　　  inverse控制翻转，主要是为了让谁去维护关系，一般是在主表中配置，将维护关系的只能交给主键。

　  　fetch取值，fetch="join"主要是在查询的时候Hibernate会自动查询有关联的表。fetch="join",hibernate会通过select语句使用外连接来加载其关联实体或集合，此时lazy会失效.（也可以通过设置lazy为false,实现加载关联表）

　  　fetch="subselect",另外发送一条select语句抓取在前面查询到的所有实体对象的关联集合

　  　batch-size配置这个属性是让Hibernate在执行批量的数据库操作

              7.配置了lazy=”true”一定会实现懒加载吗？

                 re：

                     不一定，因为如果再配置中你也使用fetch属性的话此时lazy就会失效。（lazy设为true，同理）

              8.请你谈谈Hibernate中的“N+1”问题

               re:

                    所谓N+1问题，即hibernate在做关联查询的时候有几个关联对象，就会有多少个查询语句，这便是N，而+1是查询实体本生还需要一个查询语句，所以hibernate查询一个有关联的实体便会产生N+1条查询语句，这便是N+1.对于hibernate的N+1问题。而Hibernate中使用N+1策略时执行Criteria时，会将自动进行SQL构造，并且映射到实体Bean中，就只会返回一个结果，这样不仅提高了性能，而且在处理返回值的时候也变得比较轻松。

             9.请你谈谈Hibernate有哪些最佳实践？

                re:

                    就目前本人没有用过太多的框架，就不和其他框架做比较。在此就简单说一下心得吧。

                    No1.对象-关系映射。

                       hibernate支持关系型数据库，并且它与sql紧密结合的程度较高，有自己的查询语言hql,它与sql的区别在于hql是面向对象的而不是针对数据模型。但是如果需要也可以在hibernate中使用 sql.并且hibernate还提供了灵活的映射机制，以及session级缓存，和sessionfactosy级缓存。只是session级缓存是非线程安全的，如果多线程访问可能引起数据存取的不一致，甚至系统宕机。