Java程序员面试题集

# 1 、Struts 2中，Action通过什么方式获得用户从页面输入的数据，又是通过什么方式把其自身的数据传给视图的？

答：Action从页面获取数据有三种方式：

①通过Action属性接受参数

②通过域模型获取参数

③通过模型驱动获取参数 （ModelDriven<T>）

Action将数据存入值栈(Value Stack)中，视图可以通过表达式语言（EL）从值栈中获取数据。

# 2、简述Struts 2是如何实现MVC架构模式的。

答：MVC架构模式要求应用程序的输入、处理和输出三者分离，将系统分成模型（Model）、视图（View）、控制器（Controller）三个部分，通过控制器实现模型和视图的解耦合，使得应用程序的开发和维护变得容易，如下图所示。其中，模型代表了应用程序的数据和处理这些数据的规则，同时还可以为视图提供的查询保存相关的状态，通常由JavaBean来实现，模型的代码写一次就可以被多个视图重用；视图用来组织模型的内容，它从模型中获得数据，并将数据展现给用户，在Struts 2中通常由JSP、Freemarker模板等来实现；控制器负责从客户端接受请求并将其转换为某种行为，行为完成后再选择一个视图来呈现给用户，控制器本身不需要输出任何内容，它只是接收请求并决定调用哪个模型组件去处理请求，StrutsPrepareAndExecuteFilter过滤器是Struts 2中的核心，它和一系列的Action构成了Struts 2中的控制器。

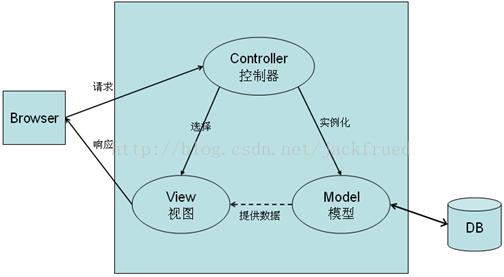


图-1 MVC架构模式图

# 3、阐述Struts 2如何实现用户输入验证。在你做过的项目中使用的是那种验证方式，为什么选择这种方式？

答：Struts 2可以使用手动验证和自动验证框架实现用户输入验证。自动验证框架是将对输入的验证规则放在XML文件中，这种方式比较灵活，可以在不修改代码的情况下修改验证的规则。

# 4、阐述Struts 2中的Action如何编写？Action是否采用了单例？

答：Struts2的Action有三种写法：

①POJO

②实现Action接口重写execute()方法

③继承ActionSupport类

Action没有像Servlet一样使用单实例多线程的工作方式，很明显，每个Action要接收不同用户的请求参数，这就意味着Action是有状态的，因此在设计上使用了每个请求对应一个Action的处理方式。

# 5、Struts 2中的Action并没有直接收到用户的请求，那它为什么可以处理用户的请求，又凭什么知道一个请求到底交给哪个Action来处理？

答：Struts2的核心过滤器接收到用户请求后，会对用户的请求进行简单的预处理(例如解析、封装参数)，然后通过反射来创建Action实例，并调用Action中指定的方法来处理用户请求。

要决定请求交给哪一个Action来处理有两种方式：1利用配置文件：可以在配置文件中通过<action>标签配置和请求对应的Action类以及要调用的方法；2利用约定：Struts2中可以使用约定(convention)插件，例如约定xxx总是对应XxxAction，这是对约定优于配置理念的践行。

# 6、你经常用到的Struts 2常量配置有哪些？

答：

①struts.i18n.encoding– 指定默认编码

②struts.objectFactory/ struts.objectFactory.spring.autoWire – 对象工厂 / Spring的自动装配方式(名字、类型)

③struts.devMode– 是否使用开发模式

④struts.locale– 指定默认区域，默认值是en\_US

⑤struts.i18n.resources– 国际化使用的资源文件

⑥struts.enable.DynamicMethodInvocation– 是否允许动态方法调用

# 7、简述Struts2的异常处理机制。

答：Struts 2提供了声明式的异常处理机制，可以在配置文件中加入如下代码：

<global-exception-mappings>

<exception-mapping exception=”…” result=”…”/>

</global-exception-mappings>

93、说一下你对约定优于配置（CoC）的理解。

答：约定优于配置(convention over configuration)，也称作按约定编程，是一种软件设计范式，旨在减少软件开发人员需做决定的数量，获得简单的好处而又不失灵活性。CoC本质是说，开发人员仅需规定应用中不符约定的部分。例如，如果模型中有个名为Sale的类，那么数据库中对应的表就会默认命名为sales。只有在偏离这一约定时，例如将该表命名为products\_sold，才需写有关这个名字的配置。如果您所用工具的约定与你的期待相符，便可省去配置；反之，你可以配置来达到你所期待的方式。遵循约定虽然损失了一定的灵活性，不能随意安排目录结构，不能随意进行函数命名，但是却能减少配置。更重要的是，遵循约定可以帮助开发人员遵守构建标准，包括各种命名的规范，这对团队开发是非常有利的。

# 8、Struts2中如何实现I18N？

答：首先，为不同语言地区编写不同的资源文件；然后在Struts 2配置文件中配置struts.i18n.custom.resources常量；在Action中可以通过调用getText()方法读取资源文件获取国际化资源。

# 9、简述拦截器的工作原理以及你在项目中使用过哪些自定义拦截器。

答：Struts 2中定义了拦截器的接口以及默认实现，实现了Interceptor接口或继承了AbstractInterceptor的类可以作为拦截器。接口中的init()方法在拦截器被创建后立即被调用，它在拦截器的生命周期内只被调用一次，可以在该方法中对相关资源进行必要的初始化。每拦截一个请求，intercept()方法就会被调用一次。destory()方法将在拦截器被销毁之前被调用, 它在拦截器的生命周期内也只被调用一次。

项目中使用过的有权限拦截器、执行时间拦截器、令牌拦截器等。

96、如何在Struts2中使用Ajax功能？

答：以下是Struts 2中实现Ajax的可选方式：

①JSON plugin+ jQuery

②DOJO plugin

③DWR (DirectWeb Remoting)

# 10 、谈一下拦截器和过滤器的区别。

答：拦截器和过滤器都可以用来实现横切关注功能，其区别主要在于：

①拦截器是基于Java反射机制的，而过滤器是基于接口回调的。

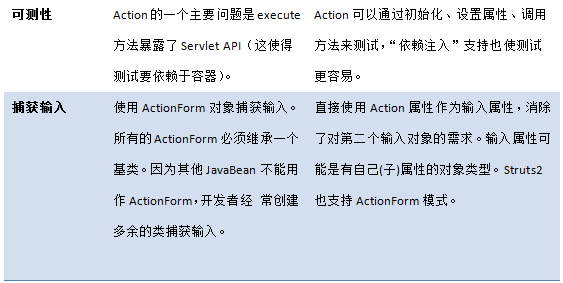
②过滤器依赖于Servlet容器，而拦截器不依赖于Servlet容器。

③拦截器只能对Action请求起作用，而过滤器可以对所有请求起作用。

④拦截器可以访问Action上下文、值栈里的对象，而过滤器不能。

# 11、谈一下Struts 1和Struts 2的区别。

答：



# 12、谈一下你的项目选择Struts 2的理由

答：①Action是POJO，没有依赖Servlet API，具有良好的可测试性；②强大的拦截器简化了开发的复杂度；③支持多种表现层技术：JSP、Freemarker等等；④灵活的验证方式；⑤国际化(I18N)支持；⑥声明式异常管理；⑦通过JSON插件简化Ajax；⑧通过Spring插件跟Spring整合。

【补充】有人为选择和评判Web框架提出了20条标准，包括：开发人员的工作效率（能用1-5天搭建一个CRUD页面吗）、开发人员的看法（用起来有意思吗）、学习曲线（学了一个星期或一个月后能干活吗）、项目健康状况（项目陷入绝境了吗）、开发人员的充足性（能找到经验丰富的开发人员吗）、就业趋势（将来能招到人吗）、模板化（遵循DRY原则吗）、组件（自带日期选择器之类的控件吗）、Ajax（是否支持异步调用和局部刷新）、插件或附加项（能加入Facebook集成之类的功能吗）、扩展性（默认的控制处理的并发用户数能到500+吗）、测试支持（能够做测试驱动的开发吗）、I18N和L10N（有多国语言、地域支持吗）、校验（能轻松校验用户输入并迅速反馈吗）、多编程语言支持（能够同时使用多种语言开发吗）、文档的质量（常见的用例和问题都在文档中有体现吗）、出版的图书（有没有行业专家使用了它并分享了自己的使用经验）、REST支持（能按HTTP协议的设计宗旨使用该协议吗）、移动支持（是否很容易就能支持Android、iOS和其他移动智能终端）、风险程度（能不能做大型项目）。很明显，Java其实算不上最优秀的Web开发语言，但是它却满足了这20条中的很多，尤其是充足的开发人员、成熟的解决方案这两点，而且Java的生态系统是非常良好的，这也是在Java已经显得江河日下的今天大家仍然一如既往的将其作为开发语言首选的原因。

# 13、Struts 2中如何访问HttpServletRequest、HttpSession和ServletContext三个域对象

答：有两种方式：

①通过ServletActionContext的方法获得；

②通过ServletRequestAware、SessionAware和ServletContextAware接口注入。

# 14、Struts 2中的默认包struts-default有什么作用？

答：它定义了Struts 2内部的众多拦截器和Result类型，而Struts 2很多核心的功能都是通过这些内置的拦截器实现，如：从请求中把请求参数封装到action、文件上传和数据验证等等都是通过拦截器实现的。在Struts 2的配置文件中，自定义的包继承了struts-default包就可以使用Struts 2为我们提供的这些功能。

# 15、简述值栈(Value-Stack)的原理和生命周期

答：Value-Stack贯穿整个 Action 的生命周期，保存在request作用域中，所以它和request的生命周期一样。当Struts 2接受一个请求时，会创建ActionContext、Value-Stack和Action对象，然后把Action存放进Value-Stack，所以Action的实例变量可以通过OGNL访问。由于Action是多实例的，和使用单例的Servlet不同，  每个Action都有一个对应的Value-Stack，Value-Stack存放的数据类型是该Action的实例，以及该Action中的实例变量，Action对象默认保存在栈顶。

# 16、SessionFactory是线程安全的吗？Session是线程安全的吗，两个线程能够共享同一个Session吗？

答：SessionFactory对应Hibernate的一个数据存储的概念，它是线程安全的，可以被多个线程并发访问。SessionFactory一般只会在启动的时候构建。对于应用程序，最好将SessionFactory通过单例的模式进行封装以便于访问。Session是一个轻量级非线程安全的对象（线程间不能共享session），它表示与数据库进行交互的一个工作单元。Session是由SessionFactory创建的，在任务完成之后它会被关闭。Session是持久层服务对外提供的主要接口。Session会延迟获取数据库连接（也就是在需要的时候才会获取）。为了避免创建太多的session，可以使用ThreadLocal来取得当前的session，无论你调用多少次getCurrentSession()方法，返回的都是同一个session。

# 17、Session的load和get方法的区别是什么？

答：主要有以下三项区别：

① 如果没有找到符合条件的记录, get方法返回null,load方法抛出异常

②get方法直接返回实体类对象, load方法返回实体类对象的代理

③ 在Hibernate 3之前，get方法只在一级缓存(内部缓存)中进行数据查找, 如果没有找到对应的数据则越过二级缓存, 直接发出SQL语句完成数据读取; load方法则可以充分利用二级缓存中的现有数据；当然从Hibernate 3开始，get方法不再是对二级缓存只写不读，它也是可以访问二级缓存的

简单的说，对于load()方法Hibernate认为该数据在数据库中一定存在可以放心的使用代理来实现延迟加载，如果没有数据就抛出异常，而通过get()方法去取的数据可以不存在。

# 18、Session的save()、update()、merge()、lock()、saveOrUpdate()和persist()方法有什么区别？

答：Hibernate的对象有三种状态：瞬态、持久态和游离态。游离状态的实例可以通过调用save()、persist()或者saveOrUpdate()方法进行持久化；脱管状态的实例可以通过调用 update()、0saveOrUpdate()、lock()或者replicate()进行持久化。save()和persist()将会引发SQL的INSERT语句，而update()或merge()会引发UPDATE语句。save()和update()的区别在于一个是将瞬态对象变成持久态，一个是将游离态对象变为持久态。merge方法可以完成save()和update()方法的功能，它的意图是将新的状态合并到已有的持久化对象上或创建新的持久化对象。按照官方文档的说明：(1)persist()方法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"，当它在一个事务外部被调用的时候并不触发一个Insert语句，当需要封装一个长会话流程的时候，一个persist这样的函数是需要的。(3)save"不保证"第2条,它要返回标识符，所以它会立即执行Insert语句，不管是不是在事务内部还是外部。update()方法是把一个已经更改过的脱管状态的对象变成持久状态；lock()方法是把一个没有更改过的脱管状态的对象变成持久状态。

# 19、阐述Session加载实体对象的过程。

答：Session加载实体对象的步骤是：

① Session在调用数据库查询功能之前, 首先会在缓存中进行查询, 在一级缓存中, 通过实体类型和主键进行查找, 如果一级缓存查找命中且数据状态合法, 则直接返回

③ 如果一级缓存没有命中, 接下来Session会在当前NonExists记录(相当于一个查询黑名单, 如果出现重复的无效查询可以迅速判断, 从而提升性能)中进行查找, 如果NonExists中存在同样的查询条件,则返回null

③ 对于load方法, 如果一级缓存查询失败则查询二级缓存, 如果二级缓存命中则直接返回

④ 如果之前的查询都未命中, 则发出SQL语句, 如果查询未发现对应记录则将此次查询添加到Session的NonExists中加以记录, 并返回null

⑤ 根据映射配置和SQL语句得到ResultSet,并创建对应的实体对象

⑥ 将对象纳入Session(一级缓存)管理

⑦ 执行拦截器的onLoad方法(如果有对应的拦截器)

⑧将数据对象纳入二级缓存

⑨返回数据对象

# 20、Query接口的list方法和iterate方法有什么区别？

答：

1) list方法无法利用缓存,它对缓存只写不读; iterate方法可以充分利用缓存, 如果目标数据只读或者读取频繁, iterate可以减少性能开销

2) list方法不会引起N+1查询问题, 而iterate方法会引起N+1查询问题

108、Hibernate如何实现分页查询？

答：通过Hibernate实现分页查询，开发人员只需要提供HQL语句、查询起始行数(setFirstresult()方法)和最大查询行数(setMaxResult()方法)，并调用Query接口的list()方法，Hibernate会自动生成分页查询的SQL语句。

# 21、锁机制有什么用？简述Hibernate的悲观锁和乐观锁机制。

答：有些业务逻辑在执行过程中往往需要保证数据访问的排他性，于是需要通过一些机制保证在此过程中数据被锁住不会被外界修改，这就是所谓的锁机制。

Hibernate支持悲观锁和乐观锁两种锁机制。悲观锁，顾名思义，它悲观的认为在数据处理过程中一定存在修改数据的并发事务(包括本系统的其他事务或来自外部系统的事务)，于是将处理的数据设置为锁定状态。悲观锁必须依赖数据库本身的锁机制才能真正保证数据访问的排他性。乐观锁，顾名思义，对并发事务持乐观态度(认为对数据的并发操作很少发生)，通过更加宽松的锁机制解决悲观锁排他的数据访问对系统性能造成的严重影响。最常见的乐观锁是通过数据版本标识来实现的，读取数据时获得数据的版本号，更新数据时将此版本号加1，然后和数据库表对应记录的当前版本号进行比较，如果提交的数据版本号大于数据库中此记录的当前版本号则更新数据，否则认为是过期数据。

# 22、阐述实体对象的三种状态以及转换关系。

答：Hibernate中对象有三种状态：临时态（transient）、持久态（persistent）和游状态（detached），如下图所示。

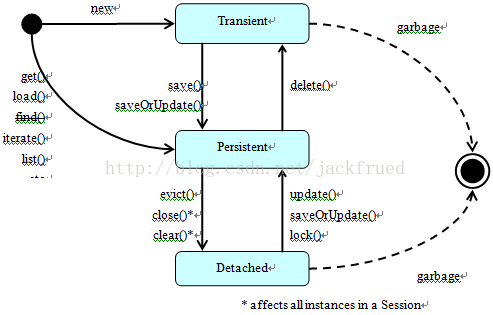


图 Hibernate实体状态转换图

临时状态：当new一个实体对象后，这个对象处于临时状态，即这个对象只是一个保存临时数据的内存区域，如果没有变量引用这个对象，则会被JVM的垃圾回收机制回收。这个对象所保存的数据与数据库没有任何关系，除非通过Session的save或者saveOrUpdate把临时对象与数据库关联，并把数据插入或者更新到数据库，这个对象才转换为持久对象。

持久状态：持久化对象的实例在数据库中有对应的记录，并拥有一个持久化标识。对持久化对象进行delete操作后，数据库中对应的记录将被删除，那么持久化对象与数据库记录不再存在对应关系，持久化对象变成临时状态。持久化对象被修改变更后，不会马上同步到数据库，直到数据库事务提交。

游离状态：当Session进行了close、clear或者evict后，持久化对象虽然拥有持久化标识符和与数据库对应记录一致的值，但是因为会话已经消失，对象不在持久化管理之内，所以处于游离状态（也叫脱管状态）。游离状态的对象与临时状态对象是十分相似的，只是它还含有持久化标识。

# 23、如何理解Hibernate的延迟加载机制。在实际应用中，延迟加载与session关闭的矛盾是如何处理的？

答：延迟加载就是并不是在读取的时候就把数据加载进来，而是等到使用时再加载。Hibernate使用了虚拟代理机制实现延迟加载。返回给用户的并不是实体本身，而是实体对象的代理。代理对象在用户调用getter方法时就会去数据库加载数据。但加载数据就需要数据库连接。而当我们把会话关闭时，数据库连接就同时关闭了。

延迟加载与session关闭的矛盾一般可以这样处理：

① 关闭延迟加载特性。这种方式操作起来比较简单，因为hibernate的延迟加载特性是可以通过映射文件或者注解进行配置的，但这种解决方案存在明显的缺陷。首先，出现no session or session was closed就证明了系统中已经存在主外键关联，如果去掉延迟加载的话，则每次查询的开销都会变得很大。

②在session关闭之前先获取需要查询的数据(Hibernate.initialize()方法)。

③ 使用拦截器(Interceptor)或过滤器(Filter)控制Session。

# 24、举一个多对多关联的例子，并说明如何实现多对多关联映射。

答：例如：商品和订单、学生和课程都是典型的多对多关系。可以在实体类上通过@ManyToMany注解配置多对多关联或者通过映射文件中的<set>和<many-to-many>标签配置多对多关联，但是通常情况下，可以将多对多关联转换成两个多对一关联来实现多对多关联映射。

# 25、谈一下你对继承映射的理解。

答：继承关系的映射策略有三种：

①每个继承结构一张表(table per class hierarchy)

②每个子类一张表(table per subclass)

③ 每个具体类一张表(table per concrete class)

第一种方式属于单表策略，其优点在于查询子类对象的时候无需表连接，查询速度快，适合多态查询；缺点是可能导致表很大。后两种方式属于多表策略，其优点在于数据存储紧凑，其缺点是需要进行连接查询，不适合多态查询。

# 26、简述Hibernate常见优化策略。

答：

①制定合理的缓存策略

② 采用合理的Session管理机制

③ 尽量使用延迟加载特性

④设定合理的批处理参数

⑤ 如果可以, 选用UUID作为主键生成器

⑥如果可以, 选用基于version的乐观锁替代悲观锁

⑦ 在开发过程中, 开启hibernate.show\_sql选项查看生成的SQL, 从而了解底层的状况;开发完成后关闭此选项

⑧ 数据库本身的优化(合理的索引, 缓存, 数据分区策略等)也会对持久层的性能带来可观的提升, 这些需要专业的DBA提供支持

# 27、谈一谈Hibernate的一级缓存、二级缓存和查询缓存。

答：Hibernate的Session提供了一级缓存的功能，默认总是有效的，当应用程序保存持久化实体、修改持久化实体时，Session并不会立即把这种改变提交到数据库，而是缓存在当前的Session中，除非显示调用了Session的flush()方法或通过close()方法关闭Session。通过一级缓存，可以减少程序与数据库的交互，从而提高数据库访问性能。

SessionFactory级别的二级缓存是全局性的，所有的Session可以共享这个二级缓存。不过二级缓存默认是关闭的，需要显示开启并指定需要使用哪种二级缓存实现类(可以使用第三方提供的实现)。一旦开启了二级缓存并设置了需要使用二级缓存的实体类，SessionFactory就会缓存访问过的该实体类的每个对象，除非缓存的数据超出了指定的缓存空间。

一级缓存和二级缓存都是对整个实体进行缓存，不会缓存普通属性，如果希望对普通属性进行缓存，可以使用查询缓存。查询缓存是将HQL或SQL语句以及它们的查询结果作为键值对进行缓存，对于同样的查询可以直接从缓存中获取数据。查询缓存默认也是关闭的，需要显示开启。