java面试笔记

一.必考面试题

1、对象的概念

一个对象就是一个整体，其中包含各种属性

2、类

类是对象的抽象，用来描述对象的行为和状态

3、super

调用父类的方法或属性

4、this

指的是当前类

5、static

静态修饰符，被修饰的方法或者属性可以直接使用，不需要重新new，一般写单例模式的时候使用

6、final

修饰符，表示终态，被修饰的对象不能改变，修饰的类不能被继承，修饰的方法不能被重写，修饰的变量在定义得时候需要初始化，并且也不能在改变

finally：表示表示最终处理用在cry/catch块中

7.重载

方法名相同,但是参数数量或者类型不一样

8.重写

子类继承父类的方法,对父类的方法进行重新编写

9.线程

声明周期:线程一旦启动就会执行run方法,直到run方法中的任务执行完毕结束,也可以手动结束,使用stop/destroy方法

创建线程的方法:

(1)实现runnable接口

(2)继承Thread类

优先级:可以设置线程优先级,但是不能保证优先级高的一定先执行

10.集合

用于放置同一类元素

List有序

set无序

Map键值

二.重点面试题

1.运行异常与一般异常

2.ArrayList与Vector的区别

3.事务概念,特性

4.类的三大特性

5.forward与redirect

6.提高数据查询效率

7.单例模式

8.OGNL表达式取值

9.列举Java中常用的设置模式

10.HashMap与HashTable异同

11.sleep()与wait()

12.线程同步的方法

13.反射

14.jsp与Servlet的区别于联系

15.xml解析

16.启动异常(runtime Exception)

17.Connection与Connections

18.JVM加载Class机制

19.数据库连接池的工作机制

20.IO流,字节流,字符流

21.java序列化

22.避免重复提交

23.tcp/ip三次握手

24.存储过程

三.框架面试题

1.Hibernate中不看数据库,XML,查询语句如何得知表结构

2.Struts框架工作流程

3.Spring的IOC(DI),AOP

4.Hibernate关联关系

5.AOP和OOP,IOC和DI

6.Hibernate中对象的3种状态

7.Hibernate延迟加载

8.Hibernate中的缓存机制

9.

四.数据库面试题

1.查询工资排名3-5的员工信息

select \* from (select ename,sal,rownum num from (select ename,sal from emp where sal is not null order by sal desc))

where num between 3 and 5;

2.用一条sql语句查询出姓名重复的员工/学员姓名并统计数量

select ename count(\*) from emp group by name

having count(\*)>1 order by count(\*) desc;

3.按照工资进行排序(工资相同则排名相同)

select e.empno,e.ename,e.salary,dense\_rank() over(order by salary desc)

mingic from emp e;

4.查询每个部门工资最高的前两名员工信息

select \* from (select e.empno,e.dempno,e.ename,e.salary,dense\_rank() over (partion by dempno order by salary desc )

mingic from emp e)x where x.mingic<=2;

**Struts2笔记**

1、拷贝struts.xml文件

拷贝目录：Java Resources--src

2、拷贝Struts 的jar文件

源目录：\struts-2.3.30\lib

目标目录：/WebContent/WEB-INF/lib

3、配置web.xml文件

将filter部分拷贝到web.xml文件中

<filter>

<filter-name>struts2</filter-name>

<filter-class>

org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

4、修改、配置struts.xml文件

//一下为最基本一个Demo

<struts>

<package name="default" namespace="/" extends="struts-default">

<action name="login">

<result name="success">/Login.jsp</result>

</action>

</package>

</struts>

5、Struts常量定义

<constant name="struts.devMobe" value="true"></constant>

//意为开发模式，对数据进行修改而无需重启服务器(热修改)

//实测Tomcat7 Struts2.3.2不支持

6、查看Struts源码、API文档

源码位置：\struts-2.3.30\src\core\src\main\java

API文档：\struts-2.3.30\docs\struts2-core-apidocs

//文档位置改变，在项目中不能实时更新

//文档位置支持中文

7、编写Struts.xml文件时自动提示

window--preferences--搜索catalog--xml catalog--add--key:http://struts.apache.org/dtds/struts-2.3.dtd

location:file System:E:\\struts-2.3.30\lib\struts2-core-2.3.30\struts-2.3.dtd

key type :URI

8、namespace决定访问路径

namespace="",可以访问任意路径的action

namespace="/",可以访问指定路径的action

9、项目拷贝

本机项目(不在工作)：file--import--General/Existing Projects into Workspace

--Broser

本机项目(工作空间):Ctrl + C--Ctrl + V --修改项目名--右键项目--Properties--

搜索web--Web Project Setting--将原来的项目名修改为现在的项目名

10、定义Action类的3种方法

(1)自己创建一个普通的类，里边包含public String execute()方法

(2)实现Action接口,重写execute()方法

(3)继承Actionsuport类,重写execute()方法

//推荐使用第三种方法,因为ActionSupport类中已经实现Action接口，并且

本身封装了很多丰富的方法

11、Struts1和Struts2在Action实现方面的区别

Struts1定义的Action类在每一次访问时都会是同一个Action类，只有一个实例，

多个用户访问的时候，会出现线程同步问题

Struts2定义的Action类在服务器的每一次访问的时候都会new一个Action实例出来，

这样就可以避免线程同步问题

12、Struts2中的访问路径问题：

Struts2中的访问路径是根据Action的路径而不是JSP的路径来确定，所以尽量不要使用相对路径

统一使用绝对路径进行访问(在访问标签上添加或者整个页面使用指定basePath)

路径获取：<% String path = request.getContextPath();

String basePath = request.getScheme()+ "://" + request.getServerName()+":"

+ request.getServerPort() + path +"/";

%>

路径添加：<base href="<%=basePath%>"/>

13、Struts2中的文件上传

struts2中有专门的文件上传,但是其默认上传文件最大为2M(上传超出2M的文件后台报错)

修改struts.xml:

常量：<constant name="struts.multipart.maxSize" value="20971520" />

<!-- 设置允许最大值 20MB(1024\*1024\*20) -->

拦截器:

<interceptor-ref name="fileUpload">

<param name="maximumSize">5242880</param>

</interceptor-ref>

<interceptor-ref name="defaultStack" /> <!-- 默认拦截器栈 -->

//这里所有的数值都不能以算式的形式存在(5\*1024\*1024 错)

14、Action接收前台参数

在Action中定义参数名与前台一致,并生成set和get方法,则可以直接在后台获取

对于对象 eg:实体类user,属性name,password,前台user.name,后台定义user,同样生成set和get方法

后台取值user.name

对于其后台数据不一致的地方,可以使用对象传输模型,最后生成对象模型(简单来说就是实际的实体类模型只有name

和password两个属性,但是前台注册有一个确认密码这个参数传到后台,这是可以再定义一个实体类,其中除了name和

password两个属性之外还有一个confimpassword属性,这个实体类就是对象传输模型DTO:data transform object)

使用ModelDriven接收参数,需要Action实现ModelDriven<>接口(不常用),这时候需要自己new实体对象,但是不需要生成

set和get方法

15、接收参数的中文乱码问题

在struts.xml中配置常量i18n为:GBK/utf-8

16、如何配置Struts中的属性

参考Struts2-core-xxx.jar中的org.apache.struts2包下的default.properties文件

17、读一个系统源码的顺序:界面 -- 业务 -- 数据库 (一个业务一条线读下来,不要把全部页面读完了再看业务)

18、在没有对Action中的属性进行赋值,但是又在前台使用s:property标签取值是则会获取不到,但是不会报错,因为Struts2

没有走构造方法,如果使用s:debug标签时,没有对属性进行赋值就会报错

19、项目命名规则

(1)数据库名:项目名称(eg:example)

(2)要有数据库表的说明文档

(3)数据库表命名:通常以"t\_"/"\_"开头(避免与关键字重名,以t开头是因为Oracle数据库表名不支持"\_"开头),表名全用大写或者

使用驼峰标示(eg:\_User,\_ShoppingItem 推荐使用)

(4)数据库表中字段命名原则:就简,保持和属性名一致(尽量不要与数据库命名冲突)

(5)包命名:

[1] 按照项目模块命名:

eg: com.xxx项目.user.entity

com.xxx项目.user.action

com.xxx项目.user.dao

com.xxx项目.user.service

com.xxx项目.admin.entity

com.xxx项目.admin.action

com.xxx项目.admin.dao

com.xxx项目.admin.service

[2] 按照项目功能命名(推荐):

eg: com.xxx项目.entity.user/admin

com.xxx项目.action.userAction/adminAction

com.xxx项目.dao.userDao/adminDao

com.xxx项目.service.userService/adminService

(6)Action命名:xxxAction

(7)页面命名:模块名\_动作\_结果.html/jsp(eg:User\_add\_success.jsp),对于Struts2而言,可以考虑使用通配符配置(\*\_\*)

20、项目开发顺序(顺序并不固定)

(1)建立项目界面原型(整体的界面布局)

(2)建立配置文件(Struts/Hibernate/Spring)

[1]确定namespace

[2]确定package

[3]确定Action的名称

[4]确定result界面

[5]根据需要修改原型界面的name(改成可以使用通配符进行配置的形式)

[6]测试

[7]做好整体规划

(3)建立数据库(或者实体类)

(4)建立实体层

(5)建立Service层

(6)单元测试(使用Junit)

**Hibernate笔记**

准备工作：下载Hibernate包 (官网：http://www.hibernate.org)

数据库的驱动包(根据自己的数据可类型下载相应的驱动)

1、Hibernate说明文档

路径：\hibernate-release-4.2.21.Final\documentation\manual\en-US\html\_single

2、创建对应的数据库以及表

3、创建一个Java项目,并且创建实体类(与数据库表对应)

4、配置Hibernate的XML文件

从说明文档(步骤1中已经提到)中拷贝hibernate.cfg.xml(或者直接到下载的Hibernate文件夹

中搜索hibernate.cfg.xml)

在src目录下新建文件:hibernate.cfg.xml,将说明文档中的配置拷贝到其中(或者直接将搜索到的xml文件复制到src目录下)

hibernate.cfg.xml中的标签说明详见此说明文档同一目录下的hibernate.cfg.xml

配置关联实体类的xml文件:Xxx.hbm.xml(这里的Xxx表示实体类名)

同样在说明文档或者Hibernate文件夹中找到,创建在和实体类同一个包下(具体参数详见同目录下User.hbm.xml)

2、导包

使用Hibernate的jar包:\lib\required 下边所有的jar包,以及连接相应数据库的驱动jar包

直接新建user library,将hibernate有关的包全部加进来

具体实现:window -- perference -- java -- build path --user library -- new --命名自己创建的包

--ok--add external jars--选中Hibernate 的包--ok(这里并没有将其加入到特定项目中)

或者:鼠标右键项目--build path -- configure build path -- libraries -- add library -- user library

-- next -- user libraries -- new -- 命名自己创建的包 -- ok --

add external jars--选中Hibernate 的包--ok

3、测试(建议使用junit进行单元测试)

(详见同目录下Demo.java)

4、使用Annotation注解

(1)导包(按照之前的步骤,里边已经包含所需要的jar,hibernate-release-4.2.21.Final版里边已经封装)

(2)在实体类前加:@Entity //这里需要 import javax.persistence.Entity;

(3)定义主键,在主键前边加:@Id

(4)配置Hibernate的XML文件:hibernate.cfg.xml

添加:<mapping class="com.hibernate.entity.Admin" />

(5)高级配置:

注解数据库表：@Table(name = "table name" catlog = "database name" schema = "dbo")

//在实体类的前边

注解列名:@Column( name="columnName") //在实体类的对应变量的get方法前边

不产生持久化:@Transient //只在实体类中存在,并不会在数据库中产生相应的列

时间日期精度:@Temporal(TemporalType.TIMESTAMP)

//TIMESTAMP:时间和日期,DATE:仅日期,TIME:时间

//一般用TIMESTAMP,在使用时java中数据类型为java.util.Date

5、单元测试工具--junit

管理类库的方式:将开发代码和测试代码分开(借鉴meven,Ansroid Studio在新建项目的时候会自动生成测试类)

使用说明:

Java 框架的做法 :创建Junit test case测试类 -- 选择需要测试的类(也可以指定类中的方法) --

在需要测试的方法前边加注解:@Test(自己写的测试方法也需要加注解)

6、单例模式

使用单例模式创建 SessionFactory(SessionFactory的创建很耗时)来优化数据库链接

Action 不用单例模式

7、Oracle数据库表名不能以下划线开头

8、数据库配置文件可参考hibernate.properties(hibernate4.2.21\project\etc)

9、联合主键注意事项,除了配置文件之外,还需要实现java.io.sirializable接口以及重写主键类的equals()方法和hashCode()方法(一般很少用到)

Annotation版:(1)@Embeddable 注解在主键类 @Id 注解在实体类主键上

(2)@EmbeddedID 注解在实体类主键上(\*)

(3)@Id 注解在实体类主键上 @IdClass 注解在实体类上,参数为主键类

10、常用主键生成策略:native(使用Annotation注解默认策略)、identity(支持Mysql,SQL Server)、sequence(Oracle数据库)、

uuid(都支持,但是主键数据类型为String)

11、Hibernate核心开发接口

(1)Configuration:

[1]管理配置信息

[2]用来产生SessionFactory

[3]可以再configure()方法中指定配置文件

12、关于Session的面试问题

openSession()与getCurrentSession()的区别:

openSession()每次打开新的Session(一个Session相当于一个连接),手动close

getCurrentSession是如果当前环境中有Session,则直接拿过来使用,并不会创建新的Session,没有的时候再创建新的Session

当一个Session已经做了提交之后,那么这个Session已经不存在了,使用getcCurrentSession获取到的并不是刚才的Session

用途:界定事务边界(比如添加日志)

getCurrentSession在提交事物之后自动close

推荐使用getCurrentSession(使用getcCurrentSession必须要指定上下文环境)

openSession不能与getCurrentSession混用,即使上边使用openSession,下边使用的getCurrentSession也不是用一个Session

上下文:thread和jta

thread是指当前线程,一般用于在一个数据库中进行操作,jta是指分布式事物管理,一般用于在多个数据库之间进行操作,jta的transmanager

有应用服务器提供,使用的驱动也不相同

13、对象的3种状态:transient,persistent,detached

transient:内存中有一个对象,没有ID,缓存中也没有

persistent:内存中有,缓存中也有,数据库有,有ID

detached:内存中有,缓存中没有,数据库中有,有ID

3种状态的区分:

(1)有么有id

(2)id在数据库中有没有

(3)内存中有没有Session(缓存)

14、Hibernate中load()方法与get()方法的区别(面试题)

(1)load()返回的是代理对象,只有在需要获取对象的时候才会发出sql语句

(2)get()会直接从数据库加载,没有延迟

(3)不存在对应记录的时候表现不一样(load()在没有发出sql语句之前即使所查询的对象在数据库不存在,依旧不会报错,但是由于get()是直接发出sql

语句去数据库查询的,因此,如果数据库中没有这一条的话就会报错)

15、Hibernate中clear()方法

无论是load()还是get()方法,都会首先查找缓存(一级缓存),如果没有才会从数据库中查找,调用clear()方法可以强制清除session缓存,

clear()方法不会与数据库打交道

16、Hibernate中flush()方法

flush()方法会强制将缓存中的数据域数据库中的数据保持一致(事物在执行commit()之前会自动flush())

17、关系映射(重要,面试+实用,重在外键,主键次之)

一对一(了解):

单向外键:数据库int fk\_id 实体类@onetoone

双向外键:数据库int fk\_id 实体类两边@onetoone(),数据库中没有fk\_id的一边设置@onetoone(mappedBy="xxx"),"xxx"表示

另一实体类中对应的属性(凡是双向外键关联,必设mappedBy)

\*单向和多向在数据库中的表现没有差别,但是在Java程序中表现不一样,单向的只能从一方找到另一方,但是双向的可以两边相互找到

外键联合主键@columns({@column(name="xxx",referencedColumnName="yyy"),

{@column(name="xxx",referencedColumnName="yyy") ...})(不重要)

多对一:单向：

多的一方 @manytoone @joincolmn(name="xxx"):指定多的一方对应表中生成的字段

一对多:单向:

一的一方使用set @onetomany @joincolmn(name="xxx"):"xxx"指定多的一方对应表中生成的字段

双向:单的一方set @onetomany(mappedBy="aaa"):"aaa"为一的一方在多的一方对应的属性,

多的一方@manytoone @joincolmn(name="xxx"):"xxx"指定多的一方对应表中生成的字段

多对多: 单向:主动方set被动方,@ManyToMany

@JoinTable(name="t\_s", //"t\_s"生成中间表的名称

joinColumns={@JoinColumn(name="tid")}, //"t\_id"当前主动表在中间表中生成的字段名

inverseJoinColumns={@JoinColumn(name="sid")}) //"s\_id"被动表在中间表中生成的字段名

被动方不用设置任何属性

双向(少用): 双向需要在双方都建立对应关系

在多对多关系中,若要向一方插入数据,则需要双方进行关联

eg: e1.getProjects().add(p1);

e2.getProjects().add(p1);

p1.getEmployees().add(e1);

p1.getEmployees().add(e2);

多对一/一对多CRUD:需要建立关联(一般从多的一方进行操作方便一些),设置@manytoone(cascade={cascadeType.All})

/@onetomany(cascade={cascadeType.All})

/cascade:在关系映射中,一张表变动,与其有关的表中数据也跟着改变

双向必须设mappedBy()

在多对一的关系表中,从多的一方查询数据会默认将一的一方数据给查询出来,默认属性fetch=fetchType.eager,但是如果也将其设置为lazy,

则会出现在Session没有提交之前可以获取到一的一方数据,但是Session提交之后并不能获取到,而且如果提交之前没有去主动获取一的一方数据,

Hibernate也不会生成相应的sql语句

(一般不用)若要查询一的一方,同时将一对应的多的数据全部取出,则需要设置fetch=fetchType.eager //这时候默认为lazy

18、主键映射:将一个实体类作为另一个实体类的一部分,共同生成一张数据库表(作为嵌入的类不需要作实体类映射)

19、在有关联的数据库表之间删除对象：

(1)在做delete和update之前,先load,除了精确知道id外

(2)如果消除关联关系,先设定关系为null,再删除对应记录

(3)使用hql语句进行删除(常用)

(4)一般从多的一方删除只删除一条数据,从一的一方删除,则将对应的多的一方全部删除

20、QL查询(HQL和EJBQL)

功能:NativeSQL > HQL > EJBQL(jpa ql) > QBC(Query By Criteria) > QBE(Query By Example)

总结:QL应该和导航关系结合,共同为查询服务

导航关系:ManyToOne,OneToMany,ManyToMany

21、Hibernate性能优化

(1)session.clear()的使用,尤其是在不断分页循环的过程中

[1]在一个大集合中进行遍历,遍历Msg,从中取出敏感字样的对象

[2]另外一种形式的内存泄露 //面试题:Java中有内存泄露吗? A:语法级别没有,但是在实际应用中会导致内存泄露

(2)1+N问题 (典型面试题)

问题描述:在一对多/多对一关系映射中,在取出多的一方数据的时候,会顺带将一的一方数据也取出来,如果取出的数据较多,则会多出很多次的查询

优化:

[1]在ManyToOne一方设置fetchType.lazy

[2]使用BatchSize(szie = n) //在一的一方实体类上边添加注解,意思是每次从中取 n 条数据,减少发出sql语句的数量

\*[3]left join fetch 在写hql语句的时候,设置外连接,这样可以只发一条sql语句

(3)list与iterate的区别(不重要,一般就用List,主要用于面试):

[1]List直接获取对象所有数据,而Iterate现获取对象ID,等到需要用的时候再通过ID获取对象

[2]List连续发出两次数据请求,需要每次从数据库中获取,但是iterate在第二次不会像数据库发出请求,直接从Session缓存中读取

[3]在多对一/一对多关系中,List可以与left join fetch 搭配使用,但是iterate不能

(4)Hibernate缓存(不重要,面试题)

1)一级缓存:Session级别的缓存

2)二级缓存:SessionFactory级别的缓存,可以跨越Session的存在

使用场景:

[1]经常被访问的对象

[2]改动不大/不会经常改动

[3]数量有限

eg:用户权限,组织机构

load默认使用二级缓存,iterate默认使用二级缓存

list默认往二级缓存里边存数据,但是查询的时候不会在二级缓存里边进行查找

3)查询缓存(三级缓存):当两个查询语句一样的时候,打开查询缓存,则会只发出一条SQL语句

4)缓存算法(只做了解):当缓存占满之后从哪里开始将数据清出缓存

LRU:Least Resently Used,距离时间最远的

LFU:Least Frequently Used,使用次数最少的

FIFO:First In First Out,先进先出

设置缓存算法(ehcache缓存机制):memoryStoreEvictionPolicy="LRU"

22、事物并发处理(面试)

事物概念:要么都完成,要么都不完成

事物特性:ACID

Atomicity:原子性

Consistency:一致性

Isolation:隔离性

Durability:永久性

事物并发可能出现的问题:

脏读:读了另外一个事物没有提交的数据

不可重复读:在同一个事物中,前后两次读取的数据不一样

幻读:在一个事物读取的时候,另一个事物进行插入和删除,影响了读取的结果

事物隔离机制:

read-uncommitted:能够读取未提交的事物,以上三种错误仍然会出现

read-committed:不会出现脏读,但是仍然会有重复读和幻读现象

repeatable read:可重复读(相当于给某一事物锁定,只有当前事物完成,其他事物才能访问数据)

serializable:序列化,可以解决任何问题

事物级别越高,效率越低,一般情况下设置为read-committed,这种模式下不会出现脏读,而且效率也可以,然后手动解决二次重复读和幻读问题

在Hibernate中设置事物的隔离级别属性:

hibernate.connection.isolation ,value=1,2,,4,8 ,分别对应以上4种事物隔离机制,不设置默认使用数据库本身的事物隔离机制

Hibernate解决脏读问题:

悲观锁:借助数据库,在每次执行事物的时候将数据锁住,事物执行结束之后再将锁解开,效率低

具体实现:在获取数据的时候设置,load/get中的第三个参数:LockMode.UPGRADE

乐观锁:不用借助数据库,在数据的表中添加相应的字段,当一个事物(事物1)将数据获取之后,字段不会改变,在另一个事物也获取

当前数据的时候,不会受影响,但是当这个事物(事物2)对数据进行了修改之后,字段就会发生改变,这时之前的事物(事物1)要将

数据进行修改的时候就会检查字段有没有改变,如果改变则重新获取数据,没有改变则直接修改,效率高

具体实现:在实体类中添加verson字段,在其get方法上加注解@verson //不用手动设值

每次更新之前检查比较verson值

last、Hibernate重点

(1)ID生成策略(重点是AUTO)

(2)Hibernate核心开发接口

(3)关系映射

(4)Hibernate性能优化(尤其是面试)

(5)HQL语句

**Spring笔记**

Spring知识点记录

1.面向接口编程(面向抽象的编程)

2.应用分层:entity,service,dao,daoimpl,utils...

3.Spring原理(个人理解)

(1)Spring的配置文件中有beans根节点,bean子节点,bean中有一系列的属性

(2)关于bean的属性:id:用于标示,class:映射的类,property:与其他bean的关联,将关系设定清楚之后可以实现使用一个类的时候,可以直接将

与其有关联的类一并获取到(个人认为这是主要优势)

(3)ClassPathXmlApplicationContext类:对Spring的xml配置文件进行解析,包括bean的属性以及bean包含的property,然后将解析到的数据放到一个

容器中(Map容器),接着又进行了一些操作处理,比如执行set方法(这样可以在以后从容器中获取到对象之后直接使用,不需要new一个新的对象)

(4)ClassPathXmlApplicationContext类同时实现了BeanFasctory接口,而BeanFactory接口中定义了getBean()方法,用于读取配置文件中bean的id值,

相当于定义了一个从容器中获取数据的通道

(6)使用:直接通过从ClassPathXmlApplicationContext容器中调用getBean()方法即可获取到所需对象(这里获取到的实际上是对象的id,然后通过反射

找到对应的类)

4.Spring优势以及重点:

(1)轻量级框架

\*(2)IoC容器 --- 控制反转

\*(3)AOP 面向切面编程

\*(4)对事物的支持

\*(5)对框架的支持(整合现有技术)

5.使用IoC创建对象的3中方式(了解)

(1)无参构造方法来创建(默认,常用):要有参数的set和get方法(主要是set)

(2)有参构造方法来创建:

重写对象类的有参构造方法,参数为需要传递的属性值

在配置文件中对参数赋值方式:

[1]参数下标(index)的方式

[2]参数名称的方式(name)

[3]参数类型(type)

(3)创建动/静态工厂:创建一个user工厂,返回user对象,仍然使用了user的有参构造方法

6.Spring配置文件

bean:最常用的,用于设置类对象,

id为标识符,具有唯一性,没有设置id,则name默认为标识符,若果配置了id,又配置了name,那么name为别名,此时name

可以设置多个别名,中间可以用",",";"和" "分开

class是bean的全限定名=包名+类名

如果不配置id和name,那么可以根据applicationContext.getBean(Class)来获取对象,但是两个类名一样的话就行不通

alias:为bean设置别名,可以设置多个别名(不过没什么必要)

import:引入其他的配置文件,一般用于团队协作开发

7.Spring注入(重点)

(1)构造器注入(见IOC创建对象)

\*(2)setter注入:要求被注入的属性必须有set方法

(3)P,C命名空间注入

8.bean的作用域(了解)

singleton:单例(默认)

prototype:原型,每次获取bean都会产生一个新的对象(整合框架是使用该模式)

request:发出请求时产生新的对象

session:在会话范围内一个对象

global session:只在portlet下有用,表示application

application:在应用范围内一个对象

9.autowire自动装配(了解)

作用:简化配置文件

byName:根据名称(set方法名后边部分,可能会与属性名不一致)去查找相应的bean,

byType(不建议使用):根据类型自动装配,这种模式下不需要bean的id,但是同一种类型的bean只有一个

constructor:当通过构造器实例化bean时,使用byType装配构造方法

全局自动装配需要在头文件中添加default-autowire属性

推荐不使用自动装配,而使用annotation

10.静态代理(重点)

角色分析:

抽象角色:一般使用接口或者抽象类来实现

真实角色:被代理的角色

代理角色:代理真实角色,代理之后一般会做一些附属的操作

客户:使用代理角色来进行一些操作

好处:使得真实角色处理的业务更加纯粹,不再去关注一些公共的事情

公共的业务由代理来完成,实现了业务的分工

公共业务发生拓展时,变得更加集中和方便

缺点:类变多了,多了一个代理类;工作量变大了;工作效率降低

11.动态代理

分类:

基于接口:eg:jdk动态代理

基于类:eg:hibernate使用的cglib

现在使用javasist来生成动态代理

使用:一个动态代理一般代理某一类业务,一个动态可以代理多个类(或者接口)

12.AOP

(1)定义:Aspect Oriented Pragramming ,面向切面编程

(2)在Spring中的作用:

提供声明式服务(声明事务)

允许用户自定义切面

aop在不改变原来代码的情况下,增加新的功能,AOP属于横向编程模式,通过代理的方式实现的

(3)好处:同代理的好处

(4)名词解释:

关注点:增加的某个业务,如日志,安全,缓存,事务,异常等等

切面:一个关注点的模块化

连接点:只某个方法的执行

通知:在切面的某个特定的连接点上执行的动作

织入:把切面连接到其他的应用程序类型或者对象上,病创建一个被通知的对象

(5)AOP在Spring中实现

[1]通过Spring API实现

实现MethodBeforeActive接口需要导入 aopallianc.jar(该jar包需要自行下载)

使用aop还需要另外一个jar文件:aspectj-weaver.jar

[2]自定义类实现

\*[3]注解实现

(6)AOP思想:AOP是将公共业务(如日志,安全)和领域业务结合,当执行领域业务时,将会把公共业务加进来.实现公共业务的重复利用,领域业务更纯粹,

程序员专注于领域业务,其本质还是动态代理的封装

13.Spring mybatis 整合

[1]整合必备jar:mybatis-spring-xxx.jar (除了两个框架自身的jar之外还需此jar)

[2]整合方案:

1)xxDaoImpl实现SqlSessionTemplate接口,同时在配置文件中配置sqlSessionTemplate(具体见/Spring\_Mybatis\_010项目中Spring配置)

2)xxDaoImpl继承SqlSessionDaoSupport类,不需要配置sqlSessionTemplate(具体见/Spring\_Mybatis\_020项目中Spring配置)

3)使用mybatis注解,xxDaoImpl不需要继承类也不需要实现接口,在其类中定义xxMapper,并生成set方法,配置文件中需要配置xxMapper

(具体见/Spring\_Mybatis\_030项目中Spring配置)

4)不用Mybatis的配置文件,同时使用注解(不需要配置xxMapper.xml)(mybatis-spring-1.3.jar开始支持),在配置sqlSessionFactory的时候

使用configuration属性(具体见/Spring\_Mybatis\_040项目中Spring配置) //个人推荐使用

14.Struts-Spring-Mybatis整合

[1]jar包管理:主要解决公共jar包以及整合以来jar包问题

公共jar:日志类jar:commens-logging.xxx.jar ,log4j,javaassist 等等

依赖jar:struts2-spring-plugin-xxx.jar //Struts-Spring整合所依赖的jar

mybatis-spring-xxx.jar //Mybatis-Spring整合依赖的jar

[2]Struts2使用set,get方法无法获取到前台form传递的值问题

在Spring AOP中对事物的范围设置太广,导致Struts的请求也被拦截

一般设置事物范围为:service层:

<aop:pointcut expression="execution(\* com.demo.service.impl..\*.\*(..))" id="point"/>

15.Spring使用注解

daoImpl-----@Repository("xxxDao")

serviceImpl----@Service("xxxService")

action----@Controller("xxxAction") @Scope("prototype")

属性----@Autowired

**三大框架中的一些名词:**

1.JPA:Java persistence api,Java持久化API

2.O/R Mapping:object relationship mapping ,对象关系映射

3.设计模式:单例,工厂,代理,原型,

4.JTA:java transaction API:Java中管理事物的API

5.IoC:Inversion of Control,控制反转 (或者成为DI:Dependency Injection,依赖注入)

6.AOP:Aspect Oriented Pragramming ,面向切面编程