第三阶段面试题

1.IOC的概念

控制反转(InversionofControl,英文缩写为IOC)是一个重要的面向对象编程的法则来消减计算机程序的偶和问题，也是轻量级的Spring框架的核心。

把对象的创建交给spring创建我们就可以称之为控制反转。

如：

```
<bean id="myJob" class="com.itqf.job.MyJob"></bean>
```

2.注入的三种方式

set注入

调用对象的set方法给属性赋值，调用的是无参数的构造方法创建对象。

```
<!--
        等价于：BookDaoImpl dao = new BookDaoImpl();
        IoC（控制反转）：把对象的创建反转给spring来创建

-->
<bean id="bookDao" name="dao" class="com.itqf.dao.BookDaoImpl"></bean>

<!--等价于：BookServiceImpl bookService = new BookServiceImpl();
    DI（依赖注入）:把依赖关系交给spring注入
 -->
<bean id="bookService" class="com.itqf.service.BookServiceImpl">
    <!--
     等价于：bookService.setDao(bookDao);
     -->
    <property name="dao" ref="bookDao"></property>
</bean>
```

构造注入

Spring容器调用指定的构造方法，给对象属性赋值。

```
<bean id="s1" class="com.itqf.di.Student">
    <!--参数下标注入 -->
    <constructor-arg  index="0" value="李四"></constructor-arg>
    <constructor-arg index="1" value="18"></constructor-arg>
    <constructor-arg index="2" ref="car"></constructor-arg>
</bean>
<!--构造器注入-->
<bean id="s2" class="com.itqf.di.Student">
    <!--构造方法的参数名称注入
     跟属性无关，而是构造方法的参数名有关
     -->
     <constructor-arg name="name" value="王二麻子" ></constructor-arg>
     <constructor-arg name="age" value="10"></constructor-arg>
     <constructor-arg name="c" ref="car"></constructor-arg>
</bean>
```

p标签注入

```
<!-- p标签注入 -->
<bean id="s3" class="com.itqf.di.Student" p:name="尼古拉斯.赵四"
p:age="18" p:car-ref="car"></bean>
```

3MyBatis的缓存机制

提供了一级缓存和二级缓存

一级缓存：基于PerpetualCache的HashMap本地缓存，其存储作用域为Session，当Sessionflush或close之后，该Session中的所有Cache就将清空。二级缓存与一级缓存及至相同，默认也是采用PerpetualCache,HashMap存储，不同在于其存储作用域为Mapper(Namespace)，并且可定义第三方存储源，如Ehcache框架等。

对于缓存数据更新机制，当某一个作用域（一级缓存Session/二级缓存Namespace）进行了C/U/D操作后，默认该作用域下所有select中的缓存将被clear。

MyBatis中一级缓存是默认开启的，即在查询中（一次SqlSession中）。只要当SqlSession不关闭，那么你的操作会默认存储使用一级缓存。

4.解释Spring支持的几种bean的作用域

Spring框架支持以下五种bean的作用域：

```
singleton ： bean在每个Spring ioc 容器中只有一个实例。

prototype：一个bean的定义可以有多个实例。

request：每次http请求都会创建一个bean，该作用域仅在基于web的Spring ApplicationContext情形下有效。
session：在一个HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext情形下有效。

global-session：在一个全局的HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext
情形下有效。

缺省的Spring bean 的作用域是Singleton
```

5.在 Spring中如何注入一个java集合

```
Spring提供以下几种集合的配置元素：
  <list>类型用于注入一列值，允许有相同的值。
  <set> 类型用于注入一组值，不允许有相同的值。
  <map> 类型用于注入一组键值对，键和值都可以为任意类型。
  <props>类型用于注入一组键值对，键和值都只能为String类型。
  如
        <!--集合注入-->
    <bean class="com.itqf.collection_di.Car" id="car1">
        <property name="brand"  value="保时捷"></property>
        <property name="price" value="100"></property>
        <property name="color" value="白色"></property>
    </bean>
    <bean class="com.itqf.collection_di.Car" id="car2">
        <property name="brand"  value="宾利"></property>
        <property name="price" value="100"></property>
        <property name="color" value="白色"></property>
    </bean>
    <bean id="myCollections" class="com.itqf.collection_di.MyCollections">
        <property name="arrays">
            <array>
                <value>张三</value>
                <value>李四</value>
                <value>王二</value>
            </array>
        </property>
        <property name="list">
            <list>
                <value>JavaEE</value>
                <value>bigDate</value>
                <value>go</value>
            </list>
        </property>
        <property name="cars">
            <list>
                <ref bean="car1"></ref>
                <ref bean="car2"></ref>
            </list>
        </property>
        <property name="set">
            <set>
                <value>篮球</value>
                <value>乒乓球</value>
                <value>羽毛球</value>
            </set>
        </property>
        <property name="map">
            <map>
                <entry>
                    <key><value>name</value></key>
                    <value>尼古拉斯.赵四</value>
                </entry>
                <entry>
                <key><value>age</value></key>
                <value>18</value>
              </entry>
            </map>
        </property>
        <property name="properties">
            <props>
                <prop key="driverClass">com.mysql.jdbc.Driver</prop>
```

```
                <prop key="url">jdbc:mysql:///mydb</prop>
            </props>
        </property>

    </bean>
```

6.动态代理设计模式

动态代理类是一个在运行时由开发人员所指定的一列接口的实现。动态代理接口是一种由代理类实现的接口，并且是一个java.lang.reflect.Proxy类的实例。每一个代理实例都与一个调用处理器对象相联，这个调用处理器实现了java.lang.reflect.InvocationHandler接口。在代理实例上的一个方法调用是通过其中之一的代理接口被转发到与这个代理实例相联的调用处理的invoke方法上。一个java.lang.reflect.Method对象会决定那一个方法会被调用，一个类型为java.lang.Object的数组包含调用的参数。调用处理器会适当地解码方法的调用（encodedmethodinvocationasappropriate），并且它（调用处理器）的返回结果被作为在代理实例上方法调用返回的结果而返回。

如：

```
package com.itqf.jdk_proxy;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;
public class MyProxyFactory {

    public static  UsersService createProxy(){
        //1,创建被代理对象
        final UsersService usersService = new UsersServiceImpl_AOP();
        //2,创建增强类
        final MyLog myLog = new MyLog();
        //3, 创建代理对象
        //jdk中提供了一个类：Proxy.newProxyInstance
       return (UsersService) Proxy.newProxyInstance(usersService.getClass().getClassLoader(),
                usersService.getClass().getInterfaces(),
               new InvocationHandler() {

            public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
                        TransactionManager.begin();
                       myLog.beforeLog();
                       //调用真正的方法
                        Object o= null;
                       try{
                            o =  method.invoke(usersService,args);
                            TransactionManager.commit();
                       }catch(Exception e){
                           e.printStackTrace();
                           TransactionManager.rollback();
                       }
                       myLog.afterLog();


                       return o;
                   }
               });

    }
}
```

7. Spring中通知的类型

```
通知是个在方法执行前或执行后要做的动作，实际上是程序执行时要通过SpringAOP框架触发的代码段。
Spring切面可以应用五种类型的通知：
  before：前置通知，在一个方法执行前被调用。
  after：在方法执行之后调用的通知，无论方法执行是否成功。
  after-returning：仅当方法成功完成后执行的通知。
  after-throwing：在方法抛出异常退出时执行的通知。
  around：在方法执行之前和之后调用的通知。
```

8.Hibernate框架工作原理

Hibernate是一个开放源代码的对象关系映射（ORM）框架，它对JDBC进行了非常轻量级的对象封装，使得java程序员可以随心所欲的使用对象编程思维来操纵数据库。

工作原理：

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

9.Hibernate对象的三种状态是什么

瞬时态：一个Java对象创建之后，还没新增到数据库之前的状态

特点：

(1)不和Session实例关联

(2)在数据库中没有和瞬时对象关联的记录

持久态：当调用save()或者是saveOrUpdate()方法之后的状态，保存到数据库的状态

特点：

(1)和Session实例关联

(2)在数据库中有和持久对象关联的记录

游离态：当调用session的Close方法或者清空session之后对象的状态

特点：

(1)本质上和瞬时对象相同

(2)只是比瞬时对象多了一个数据库记录标识值id.

10.Hibernate对象的三种状态如何转换的

1.**瞬时对象转为持久对象**：

(1)通过Session的save()和saveOrUpdate()方法把一个瞬时对象与数据库相关联，这个瞬时对象就成为持久化对象。

(2)使用fine(),get(),load()和iterater()方法查询到的数据对象，将成为持久化对象。

2.**持久对象转为脱管对象**：

(1)当执行close()或clear(),evict()之后，持久对象会变为脱管对象。

3.**脱管对象转为持久对象**：

(1)通过Session的update(),saveOrUpdate()和lock()等方法，把脱管对象变为持久对象。

11.对象关系映射（ObjectRelationalMapping，简称ORM）

ORM是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将java程序中的对象自动持久化到关系数据库中。本质上就是将数据从一种形式转换到另外一种形式。 WhyORM? 面向对象的开发方法是当今企业级应用开发环境中的主流开发方法。常见的ORM框架有：Hibernate，MyBatis等.

12.hibernate拒绝连接、服务器崩溃的原因?最少写5个

1.db没有打开 2.网络连接可能出了问题 3.连接配置错了 4.驱动的driver，url是否都写对了 5.LIB下加入相应驱动，数据连接代码是否有误 6.数据库配置可能有问题 7.当前联接太多了，服务器都有访问人数限制的 8.服务器的相应端口没有开，即它不提供相应的服务

13.MyBatis与Hibernate有什么不同?

相同点：屏蔽jdbcapi的底层访问细节，使用我们不用与jdbcapi打交道，就可以访问数据。

jdbcapi编程流程固定，还将sql语句与java代码混杂在了一起，经常需要拼凑sql语句，细节很繁琐。

Mybatis的好处：屏蔽jdbcapi的底层访问细节；将sql语句与java代码进行分离;提供了将结果集自动封装称为实体对象和对象的集合的功能，queryForList返回对象集合，用queryForObject返回单个对象；提供了自动将实体对象的属性传递给sql语句的参数。

Hibernate是一个全自动的orm映射工具，它可以自动生成sql语句,Mybatis需要我们自己在xml配置文件中写sql语句，hibernate要比Mybatis功能负责和强大很多。因为hibernate自动生成sql语句，我们无法控制该语句，我们就无法去写特定的高效率的sql。对于一些不太复杂的sql查询，hibernate可以很好帮我们完成，但是，对于特别复杂的查询，**hibernate**就很难适应了，这时候用Mybatis就是不错的选择，因为ibatis还是由我们自己写sql语句。

13.描述Struts2框架（工作原理）

A．Struts2是一个典型的MVC前端的框架，它是以WebWork为核心。

B．**tomcat启动的时候会加载web.xml**、**核心控制器**FilterDispatcher**会加载并解析struts.xml

C．客户端会发送一个请求到action、**FilterDispatcher**会根据后缀名进行拦截

D．FilterDispatcher根据struts.xml的配置文件信息找到某个action对应的某个类里的指定方法

E．执行相关的业务逻辑最后返回一个String

F．里配置name的属性值与返回的String进行匹配,跳转到指定的jsp页面

14.struts2框架的核心控制器是什么？它有什么作用?

Struts2框架的核心控制器是StrutsPrepareAndExecuteFilter。 作用： 负责拦截由/*指定的所有用户请求，当用户请求到达时，该Filter会过滤用户的请求。默认情况下，如果用户请求的路径 不带后缀或者后缀以.action结尾，这时请求将被转入struts2框架处理，否则struts2框架将略过该请求的处理。

15.Hibernate中get和load之间的区别（重点）：

相同点：

都可以根据ID查询一个对象

不同点：

加载方式不同：

> 当lazy=true时，代表是懒加载,get和load之间就有区别

get不管查询出来的对象是否被使用，都会立即发送SQL语句

load如果查询出来的对象没有被使用，就不会立即发送SQL语句,等需要用该对象的时候，才会发送SQL

>当lazy=false时，代表含义是立即加载，get和load都会立即发送SQL

返回结果：

>load检索不到记录时,会抛ObjectNotFoundException异常

16.Hibernate缓存概述

```
一级缓存（session级别缓存），也叫事务级别的缓存

二级缓存（sessionFactory缓存）,也叫应用级缓存

三级缓存（查询缓存）

一级缓存的生命周期和session的生命周期保持一致，hibernate默认就启用了一级缓存，不能将其关闭，可以通过session.clear()和session.evict(object)来管理一级缓存。其中get,load,iterate都会使用一级缓存，一级缓存缓存的是对象。一级缓存只查询主键时有用。

 二级缓存的生命周期和sessionFactory的生命周期保持一致，可以跨session,被多个session共享，可以手动开启并指定缓存插件如ehcache,oscache等。二级缓存也只能缓存对象。二级缓存只查询主键时有用。

三级缓存也叫查询缓存，查询缓存依赖二级缓存，所以在使用查询缓存之前配置好二级缓存。可以指定条件缓
```

17.Hibernate核心类或接口

1.**Configuration**接口

对Hibernate进行配置，以及对它进行启动。（加载hibernate.cfg.xml）并创建一个SessionFactory对象。

2.**SessionFactory**接口

SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建 Session对象。SessionFactory是线程安全的。

3.**Session**接口

Session（会话）接口是Hibernate应用使用的主要接口。Session接口负责执行被持久化对象的CRUD操作(增删改查)。Session对象是非线程安全的。Session相当于jdbc的connection。

4.**Query****接口****

总之Query接口负责执行各种数据库查询。它可以使用HQL语句或SQL 语句两种表达方式。

5.**Transaction**接口

Transaction（事务）接口是一个可选的API。负责操作相关的事务

18.Hibernate配置文件中CASECADE属性作用：

配置对象之间的级联操作：

None：不级联

Save-update：保存或者更新的时候，级联操作

Delete：删除的时候级联操作

All：保存，更新或者删除都级联操作

19.简述MyBatis框架

Mybatis和Hibernate一样，是一款开源的ORM框架的技术。

Mybatis的机制原理：

①Mybatis支持普通的SQL查询、存储过程和高级映射的持久层框架。

②Mybatis将大量的SQL语句从程序里面剖离出来，配置在配置文件中，实现SQL的灵活配置。

20.简述MyBatis框架

Mybatis和Hibernate一样，是一款开源的ORM框架的技术。

Mybatis的机制原理：

①Mybatis支持普通的SQL查询、存储过程和高级映射的持久层框架。

②Mybatis将大量的SQL语句从程序里面剖离出来，配置在配置文件中，实现SQL的灵活配置。

21.什么是bean自动装配?

Spring容器可以自动配置相互协作beans之间的关联关系。这意味着Spring可以自动配置一个bean和其他协作bean之间的关系，通过检查BeanFactory的内容里没有使用和元素。

22.解释自动装配的各种模式?

自动装配提供五种不同的模式供Spring容器用来自动装配beans之间的依赖注入:

no：默认的方式是不进行自动装配，通过手工设置ref属性来进行装配bean。

byName：通过参数名自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byName。之后容器试图匹配、装配和该bean的属性具有相同名字的bean。

byType：通过参数的数据类型自动自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byType。之后容器试图匹配和装配和该bean的属性类型一样的bean。如果有多个bean符合条件，则抛出错误。

constructor：这个同byType类似，不过是应用于构造函数的参数。如果在BeanFactory中不是恰好有一个bean与构造函数参数相同类型，则抛出一个严重的错误。

autodetect：如果有默认的构造方法，通过construct的方式自动装配，否则使用byType的方式自动装配。

自动装配有如下局限性：

重写：你仍然需要使用和设置指明依赖，这意味着总要重写自动装配。

原生数据类型:你不能自动装配简单的属性，如原生类型、字符串和类。

模糊特性：自动装配总是没有自定义装配精确，因此，如果可能尽量使用自定义装配。

23.Spring的传播特性

1.PROPAGATION_REQUIRED(**propagation_required**):如果存在一个事务，则支持当前事务。如果没有事务则开启

2.PROPAGATION_SUPPORTS(**propagation_supports**):如果存在一个事务，支持当前事务。如果没有事务，则非事务的执行

3.PROPAGATION_MANDATORY(propagation_mandatory):如果已经存在一个事务，支持当前事务。如果没有一个活动的事务，则抛出异常。

4.PROPAGATION_REQUIRES_NEW(propagation_requires_new):总是开启一个新的事务。如果一个事务已经存在，则将这个存在的事务挂起。

5.PROPAGATION_NOT_SUPPORTED(**propagation_not_supported**):总是非事务地执行，并挂起任何存在的事务。

6.PROPAGATION_NEVER(**propagation_never**):总是非事务地执行，如果存在一个活动事务，则抛出异常

7.PROPAGATION_NESTED(**propagation_nested**)：如果一个活动的事务存在，则运行在一个嵌套的事务中.如果没有活动事务,则按TransactionDefinition.PROPAGATION_REQUIRED属性执行。

24.Spring事务的隔离级别

1.ISOLATION_DEFAULT(**isolation_default**)：这是一个PlatfromTransactionManager默认的隔离级别，使用数据库默认的事务隔离级别.

另外四个与JDBC的隔离级别相对应

2.ISOLATION_READ_UNCOMMITTED(**isolation_uncommitted**)：这是事务最低的隔离级别，它充许令外一个事务可以看到这个事务未提交的数据。

这种隔离级别会产生脏读，不可重复读和幻像读。

3.ISOLATION_READ_COMMITTED(**isolation_read_committed**)：保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数据

4.ISOLATION_REPEATABLE_READ(**isolation_repeatable_read**)：这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。

它除了保证一个事务不能读取另一个事务未提交的数据外，还保证了避免下面的情况产生(不可重复读)。

5.ISOLATION_SERIALIZABLE(**isolation_serializable**)这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行。

除了防止脏读，不可重复读外，还避免了幻像读。

其中的一些概念的说明：

脏读:指当一个事务正在访问数据，并且对数据进行了修改，而这种修改还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个数据。因为这个数据是还没有提交的数据，那么另外一个事务读到的这个数据是脏数据，依据脏数据所做的操作可能是不正确的。

不可重复读:指在一个事务内，多次读同一数据。在这个事务还没有结束时，另外一个事务也访问该同一数据。那么，在第一个事务中的两次读数据之间，由于第二个事务的修改，那么第一个事务两次读到的数据可能是不一样的。这样就发生了在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。

幻觉读:指当事务不是独立执行时发生的一种现象，例如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时，第二个事务也修改这个表中的数据，这种修改是向表中插入一行新数据。那么，以后就会发生操作第一个事务的用户发现表中还有没有修改的数据行，就好象发生了幻觉一样。

25.什么是重量级?

答：轻量级是指它的创建和销毁不需要消耗太多的资源，意味着可以在程序中经常创建和销毁session的对象；重量级意味不能随意的创建和销毁它的实例，会占用很多的资源。

26.什么是SpringMVC？简单介绍下你对springMVC的理解?

Spring MVC是一个基于MVC架构的用来简化web应用程序开发的应用开发框架，它是Spring的一个模块,无需中间整合层来整合 ，它和Struts2一样都属于表现层的框架。在web模型中，MVC是一种很流行的框架，通过把Model，View，Controller分离，把较为复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

27.SpringMVC的流程?

（1）用户发送请求至前端控制器DispatcherServlet；

（2）DispatcherServlet收到请求后，调用HandlerMapping处理器映射器，请求获取Handle；

（3）处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet；

（4）DispatcherServlet通过HandlerAdapter处理器适配器调用处理器；

（5）执行处理器(Handler，也叫后端控制器)；

（6）Handler执行完成返回ModelAndView；

（7）HandlerAdapter将Handler执行结果ModelAndView返回给DispatcherServlet；

（8）DispatcherServlet将ModelAndView传给ViewResolver视图解析器进行解析；

（9）ViewResolver解析后返回具体View；

（10）DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）

（11）DispatcherServlet响应用户。

28.Springmvc的优点

（1）它是基于组件技术的。全部的应用对象,无论控制器和视图,还是业务对象之类的都是 java组件.并且和Spring提供的其他基础结构紧密集成.

（2）不依赖于ServletAPI(目标虽是如此,但是在实现的时候确实是依赖于Servlet的)

（3）可以任意使用各种视图技术,而不仅仅局限于JSP

（4） 支持各种请求资源的映射策略

（5）它应是易于扩展的

29.SpringMVC的主要组件?

（1）前端控制器DispatcherServlet（不需要程序员开发）

作用：接收请求、响应结果相当于转发器，有了DispatcherServlet就减少了其它组件之间的耦合度。

（2）处理器映射器HandlerMapping（不需要程序员开发）

作用：根据请求的URL来查找Handler

（3）处理器适配器HandlerAdapter

注意：在编写Handler的时候要按照HandlerAdapter要求的规则去编写，这样适配器HandlerAdapter才可以正确的去执行Handler。

（4）处理器Handler（需要程序员开发）

（5）视图解析器ViewResolver（不需要程序员开发）

作用：进行视图的解析根据视图逻辑名解析成真正的视图（view）

（6）视图View（需要程序员开发jsp）

View是一个接口，它的实现类支持不同的视图类型（jsp，freemarker，pdf等等）

30.springMVC和struts2的区别有哪些?

（1）springmvc的入口是一个servlet即前端控制器（DispatchServlet），而struts2入口是一个filter过虑器（StrutsPrepareAndExecuteFilter）。

（2）springmvc是基于方法开发(一个url对应一个方法)，请求参数传递到方法的形参，可以设计为单例或多例(建议单例)，struts2是基于类开发，传递参数是通过类的属性，只能设计为多例。

（3）Struts采用值栈存储请求和响应的数据，通过OGNL存取数据，springmvc通过参数解析器是将request请求内容解析，并给方法形参赋值，将数据和视图封装成ModelAndView对象，最后又将ModelAndView中的模型数据通过reques域传输到页面。Jsp视图解析器默认使用jstl。

31.SpringMVC怎么样设定重定向和转发的?

（1）在返回值前面加"forward:"就可以让结果转发,譬如"forward:user.do?name=method4"

（2）在返回值前面加"redirect:"就可以让返回值重定向,譬如"redirect:http://www.baidu.com"

32.SpringMvc怎么和AJAX相互调用的?

通过Jackson框架就可以把Java里面的对象直接转化成Js可以识别的Json对象。具体步骤如下：

（1）加入Jackson.jar

（2）在配置文件中配置json的映射

（3）在接受Ajax方法里面可以直接返回Object,List等,但方法前面要加上@ResponseBody注解。

33.如何解决POST请求中文乱码问题，GET的又如何处理呢?

（1）解决post请求乱码问题：

在web.xml中加入：

CharacterEncodingFilter

org.springframework.web.filter.CharacterEncodingFilter

encoding

utf-8

CharacterEncodingFilter

/*

（2）get请求中文参数出现乱码解决方法有两个：

①修改tomcat配置文件添加编码与工程编码一致，如下：

<ConnectorURIEncoding="utf-8" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

②另外一种方法对参数进行重新编码：

String userName= new String(request.getParamter("userName").getBytes("ISO8859-1"),"utf-8")

ISO8859-1是tomcat默认编码，需要将tomcat编码后的内容按utf-8编码。

34.Mybatis中#{}和${}的区别是什么?

#{}是预编译处理，${}是字符串替换。

Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

Mybatis在处理时，就是把${}替换成变量的值。

使用#{}可以有效的防止SQL注入，提高系统安全性。

35.通常一个Xml映射文件，都会写一个Dao接口与之对应，请问，这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗?

Dao接口，就是人们常说的Mapper接口，接口的全限名，就是映射文件中的namespace的值，接口的方法名，就是映射文件中MappedStatement的id值，接口方法内的参数，就是传递给sql的参数。Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MappedStatement，举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到namespace为com.mybatis3.mappers.StudentDao下面id=findStudentById的MappedStatement。在Mybatis中，每一个、、、标签，都会被解析为一个MappedStatement对象。

Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。

Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。

38.Mybatis是如何进行分页的？分页插件的原理是什么?

Mybatis使用RowBounds对象进行分页，它是针对ResultSet结果集执行的内存分页，而非物理分页，可以在sql内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。

分页插件的基本原理是使用Mybatis提供的插件接口，实现自定义插件，在插件的拦截方法内拦截待执行的sql，然后重写sql，根据dialect方言，添加对应的物理分页语句和物理分页参数。

39.简述Mybatis的Xml映射文件和Mybatis内部数据结构之间的映射关系?

Mybatis将所有Xml配置信息都封装到All-In-One重量级对象Configuration内部。在Xml映射文件中，标签会被解析为ParameterMap对象，其每个子元素会被解析为ParameterMapping对象。标签会被解析为ResultMap对象，其每个子元素会被解析为ResultMapping对象。每一个

答:

① Spring IoC容器找到关于Bean的定义并实例化该Bean。

② Spring IoC容器对Bean进行依赖注入。

③ 如果Bean实现了BeanNameAware接口，则将该Bean的id传给setBeanName方法。

④ 如果Bean实现了BeanFactoryAware接口，则将BeanFactory对象传给setBeanFactory方法。

⑤ 如果Bean实现了BeanPostProcessor接口，则调用其postProcessBeforeInitialization方法。

⑥ 如果Bean实现了InitializingBean接口，则调用其afterPropertySet方法。

⑦ 如果有和Bean关联的BeanPostProcessors对象，则这些对象的postProcessAfterInitialization方法被调用。

⑧ 当销毁Bean实例时，如果Bean实现了DisposableBean接口，则调用其destroy方法。

48.Springmvc的核心是什么，请求的流程是怎么处理的，控制反转怎么实现的?

springmvc是基于servlet的前端控制框架,核心是ioc和aop(基于spring实现)
核心架构的具体流程步骤如下:
1、首先用户发送请求—>DispatcherServlet，前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行
处理，作为统一访问点，进行全局的流程控制;
2、DispatcherServlet—>HandlerMapping， HandlerMapping 将会把请求映射为HandlerExecutionChain 对象（包含一个Handler 处理器（页面控制器）对象、多个
HandlerInterceptor 拦截器）对象，通过这种策略模式，很容易添加新的映射策略;
3、DispatcherServlet—>HandlerAdapter，HandlerAdapter 将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器;
4、HandlerAdapter—>处理器功能处理方法的调用，HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处
理方法，完成功能处理;并返回一个ModelAndView 对象（包含模型数据、逻辑视图名）;
5、ModelAndView的逻辑视图名—> ViewResolver， ViewResolver 将把逻辑视图名解析为具体的View，通过这种策略模式，很容易更换其他视图技术;
6、View—>渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构，因此
很容易支持其他视图技术;
7、返回控制权给DispatcherServlet，由DispatcherServlet返回响应给用户，到此一个流程结束
IOC控制反转的实现是基于spring的bean工厂,通过获取要创建的类的class全限定名称,反射创建对象

49.Mybatis如何实现批量操作?

1、批量新增、删除、更新、查询等都可以

2、配合动态sql，比如foreach 循环

3、批量查询 通过in关键字和foreach标签实现

50.描述spring事务隔离级别?

@Transactional(isolation = Isolation.READ_UNCOMMITTED)
读取未提交数据(会出现脏读，不可重复读) 基本不使用
@Transactional(isolation = Isolation.READ_COMMITTED)
读取已提交数据(会出现不可重复读和幻读)
@Transactional(isolation = Isolation.REPEATABLE_READ)
可重复读(会出现幻读)
@Transactional(isolation = Isolation.SERIALIZABLE)
串行化

MYSQL：默认为REPEATABLE_READ级别
SQLSERVER：默认为READ_COMMITTED

脏读 ：一个事务读取到另一事务未提交的更新数据
不可重复读 ：在同一事务中，多次读取同一数据返回的结果有所不同，换句话说，
后续读取可以读到另一事务已提交的更新数据. 相反，"可重复读"在同一事务中多次
读取数据时，能够保证所读数据一样，也就是后续读取不能读到另一事务已提交的更新数据
幻读 ：一个事务读到另一个事务已提交的insert数据