···**·spring 核心技术**

springIOC DI AOP 事务传播行为()

**什么是spring?**

spring是一个容器 将管理每个bean(对象)与beans(对象)之间的关系

使用spring进行管理

**好处** : 解耦 单例(节约内存)

任何对象初始化过程,全部都要交给spring管理.

**springIOC ?**

控制反转 将bean对象交给spring容器进行管理

反射--创建对象 解析xml demo4j

( 不用springIOC 的话 分为三层

控制层---web层

业务逻辑层--service层

数据访问层--dao层 )

**DI** 依赖注入 依赖 解决对象之间的依赖关系 ( ! )

**spring环境搭建**

xml (复杂)

注解---为主流

spring jar包 项目核心的为spring-core spring-context

spring-orm 与数据库的映射

aop是依赖于cglib 动态代理

**int与integer的区别**

int 基本数据类型 默认值0 不能为null

Integer 引用数据类型 int包装类 可以为null

单例singleton jvm只能允许存在一次

多例prototype 每次运行都会创建一次

request 请求作用域

session 对象和session绑定管理

**spring对象默认是单例的**

**怎么证明对象是单例的?** 查看构造函数 通过反射

使用**单例 注意事项:** 线程安全问题

**单例** 分为懒汉式饿汉式 spring中使用的是 饿汉式 线程安全

**多例** 不会产生线程安全问题

**spring bean id相同 会出现什么问题?** 报错

springIOC 默认创建 无构造函数

@repository 表示dao注入spring

@service 表示service注入spring

**@autowired 和@resource区别**

@autowired 默认是以类型查找的

@resource默认是以名称进行查找的 jdbc new对象的时候才用到

**代理设计模式?** 作用:提高对目标对象进行访问方式(中介)

好处: 提高安全性 aop,事务

静态代理和动态代理

**静态代理和动态地理区别?**

静态代理需要生成代理类

动态代理不需要代理类

spring AOP 面向切面编程 应用场景: 权限控制,事务管理,日志打印,性能统计

**什么地方会用到AOP?** 代码重复

在不同的方法中 但是需要相同的操作

**AOP** 关注点 重复代码

切面 抽取重复代码

切入点 拦截方法

**springAOP方式**

注解

xml---- 事务管理

**springaop 有哪些?**

前置通知 后置通知 异常通知 环绕通知

**springAOP方式 注解**

@before 前置通知 在方法之前通知

@after 后置通知:在方法之后通知

@afterrun 运行通知:在方法运行时通知

@afterthrowing 异常通知:方法异常之后通知

@execution 环绕通知:在方法前后进行通知

spring

**操作数据库** --放入页面进行展示

三层架构 二层架构 一层架构 mvc架构

一层架构 操作数据,业务逻辑,页面控制 全部卸载一个类里面,一个方法

二层架构 业务逻辑层和数据访问层,页面控制层

三层架构 控制层,业务逻辑层,数据访问层

mvc 模型层,视图层,控制层

**mvc 是一种软件架构思想 不是设计模式**

**设计模式** 重构代码,提高代码的重用性

spring 把一个bean配置到xml中

**什么是spring?**

spring是一个容器 就是把每个bean(实体类)与bean的关系交给第三方容器进行管理

那么这个容器就是spring,整个对象的生命周期管理

**spring的核心** ioc(控制) DI (依赖) AOP(面向切面编程)

**DI与IOC的区别是什么?**

DI依赖注入 IOC是创建对象

**spring创建对象有哪些方式**

构造函数 默认走的是无参构造函数

工厂 注解

set是注入用的

spring可以整合很多框架 继承redis,mq,数据源

spring集成redis

都是一个概念 都是把这个集成对象交给spring容器管理

**springIOC原理>**

1 解析xml配置 获取bean class地址

2 使用java的反射机制 进行实例化对象

3 返回对象

spring默认单例 会发生线程安全问题

#### Aop使用方式

XML, 注解

注解方式:----在类上面使用

@Aspect (定义切面 表示类是切面类)

@Component(注入spring容器中)

------方法上---

@before(前置通知)

@after(后置通知)

@afterThowing(异常通知)

@around(环绕通知)

XML方式

在xml中配置方法名,根据方法名称进行拦截

**单例在哪里使用?** 我们用的spring默认是单例的 单例会发生线程不安全问题

**spring beanid 重复会怎么办?** 会报错

在xml文件中 beanID是不允许重复的

注解和xml是不会报错的

**在spring中有四种生命周期**

prototype 多例 当调用getbeen方法才会初始化

singleton 单例

request http请求request作用里面使用

session http请求session作用里面使用

**spring中 如何给对象的属性赋值?**  DI 依赖注入

1.通过构造函数

2. 通过set方法给属性注入值

3 p名称空间

4 自动装配

5 注解

**set注入原理?**

属性必须要有一个set方法

**springaop 切面编程 应用场景** 权限 日志 控制 事务

**事物原理就是AOP**

**代理设计模式** jdk动态代理 cglib动态代理

jdk代理(反射)和cglib代理区别(asm)

springboot 简化xml配置 全部用注解配置 而且是微服务架构

**@resource与@autiwer区别**

@autiwer spring框架自带 默认以类型查找

@resource 是jdk1.6 默认以名称查找

AOP 编程就是把代码进行分离

(关注点 切面 切入点 )

**什么是事务?**

事务就是保证数据的一致性

**事物的特性**

原子性 不可分割

隔离性 互不影响 在并发编程 两个线程 同时操作事务

一致性 事务执行之前和执行之后都必须处于一致性状态

持久性 对数据库的数据的改变是永久性的

**项目中怎么用事务?(spring中的事务)**

1 编程事务 (理解成开车的手动挡)

2 声明式事务 (理解成开车的自动挡) xml方式 注解方式

**编程事务**

begin() 开启事务

commit() 提交事务

rollback() 回滚 数据库发生异常 要回滚

事务在方法结束就提交

**声明时的事物 当抛出异常的时候才会回滚**

**spring传播行为**

ropagation\_required 如果当前有事务 ,其他的就用当前的事物 不会新增事务

ropagation\_required\_new 如果当前有事务 不会加入当前事务 会新增事务

ropagation\_supports 支持当前事务 如果当前没有事务 就以非事务方式进行

ropagation\_mondatory 支持当前事务 如果当前没有事务 就抛出异常

ropagation\_not\_supported 以非事务方式执行操作 如果当前存在事务 就把当前事务挂起

ropagation\_never 以非事务方式执行 如果当前存在事务 抛出异常