# Spring

## 什么是Spring?

Spring是一个容器,管理bean(对象)与bean之间的关系,使用spring进行管理,

可以整合框架,和struts,hibernate

集成redis,mq,mongodb…都交给spring容器管理

**作用域:**

**Singleton 单例**:

当一个bean被设置为单例时,springIoc中就只会存在一个共享bean实例,只要id与bean配置的项匹配,就会对这个实例进行共享访问,唯一实例存储在单例缓存中,这里的单例与设计模式的单例不一样,,设计模式单例是只存在一个class,这里的单例是一个容器只对应一个bean..

**Prototype 原型**(多例:调用bean方法,才能初始化)

原型作用域部署的bean,每一次请求都会创建一个bean,相当于new,但是有一点是,spring不能对原型的bean的整个生命周期负责,容器在初始化,装饰,配置或者是原型实例化,交给客户端,原型就不在管理了

**Request** http请求都会创建一个新的bean,并且只在当前request作用域里有效

**Session** http请求都会创建一个新的bean,只在当前session作用域里

## 核心技术

1.IOC 管理bean与bean之间的关系

2. DI 依赖注入

3. AOP 面向切面编程 (利用了AOP做事务管理) 与传播行为

## Spring的好处

解耦,单例节约内存,

单例缺点:线程不安全

任何对象初始化的过程,都要注入Spring容器

## IOC

是spring的核心内容

作用:创建对象,处理对象的依赖关系

控制反转 将bean的对象交给spring容器进行管理

实现原理:反射---创建对象 解析xml,,demo4j

解析xml,获取bean calss地址,通过java反射,进行实例化对象,第三方返回对象

### 创建对象方式

:

1.注解

2.调用无参构造器 (默认)

,3.有参构造器,

4.工厂创建对象 :

1. 静态方法创建对象,
2. 非静态方法创建对象

**对象:**

单例:JVM只能运行存在一次

多例:每次运行都会创建一次

Rquest:请求作用域

Session:对象和session绑定管理

Spring对象默认是单例

单例分为懒汉式,饿汉式

Spring对象的单例是饿汉式(线程安全,访问统一对象)

判断是否是单例

通过构造函数—反射

注: Bean Id相同---,

分为两种情况:

在xml中,不允许id重复,会报错

在xml和注解中,不会报错

## DI

依赖注入 解决对象之间的依赖

### 注入方式:

1.构造函数

2.常见set方式:

.Set执行方法(属性必须有set方法,通过反射)

在xml中,通过反射,找对应的类中的方法,通过反射机制,将xml中配置的对象,传参到方法中,就实现了set方法

3.,p标签注入

4.注解方式

@Resource: 默认以名称进行查询 (JDK1.6以上)

@Autowired:”默认以类型进行查询

## AOP

### 代理模式

**:作用**:对目标对象进行访问的方式(中介)

**好处**: 提高安全性

代理分为静态代理,动态代理:

**静态代理:** 产生代理类(中介)

**动态代理**: 不产生代理类,

分为:JDK动态代理 反射机制

Cglib动态代理 字节码

### SpringAop

面向切面编程(思想)

应用场景:权限控制,事务管理 日志打印 性能统计

AOP分为关注点:重复代码

切面:提取重复的代码

切入点:需要拦截的方法

#### 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

## 事务

概念;保证事务的一致性

分为:

一致性:

一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

原子性:

原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响

隔离性:

隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离

持久性:

持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性 的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误

### Spring的事务分类

1. 编程事务(手动挡事务)

Try/cat进行捕获异常.如果异常进行回滚

事务的底层就是通过aop的前置,后置通知,前置开启事务,后置提交事务,捕获回滚

2.声明式事务(自动挡事务)”

声明式事务(xml,注解方式) 在方法执行完成后提交事务

声明式事务注意事项:

事务的程序运行时如果没有发生错误.自动提交事务,如果程序发生异常,则会自动回滚,如果使用了tyr/catch捕获异常,一定要在catch里面手动回滚

注解方式”: 在xml里配置事务开启注解

@Transactional

## 事务的7种传播行为”

常用的:1.REQUIRES :支持当前事务,如果当前没有事务,就新建一个事务

1. REQUIRES\_new:r如果当前存在事务,,就帮当前的事务挂起,互不影响
2. PROPAGATION\_SUPPORTS--支持当前事务，如果当前没有事务，就以非事务方式执行。
3. PROPAGATION\_MANDATORY--支持当前事务，如果当前没有事务，就抛出异常。
4. PROPAGATION\_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。
5. PROPAGATION\_NEVER--以非事务方式执行，如果当前存在事务，则抛出异常。

# 

# SpringBoot异常处理

## @ExceptionHandler

功能:自动捕获controller层出现的制定类型的异常,并对该异常进行相应的异常处理,比如在controller层处理…exception类型异常,就可以在controller层的类中定义方法

如果想要一个方法处理多种类型异常,就需要在({…,…})以’,’逗号分隔

## @RestControllerAdvice

用于定义@ExceptionHandler,@InitBinder,@ModelAttribute,并应用到所有@RsquesMapping,@PostMapping,@GetMapping注解中

@ControllerAdvice和@RestControllerAdvice都可以指向控制器的一个子集

当我们定义了一个自定义返回参数格式时,希望得到同意的返回,发生异常,也希望将异常统一返回,就需要用到@ControllerAdvice或者@RestControllerAdvice([如果全部异常处理返回json.就可以使用@RestControllerAdvice](mailto:如果全部异常处理返回json.就可以使用@RestControllerAdvice)代替@ControllerAdvice)这样在方法上就可以不需要添加@ResponseBody

# 统一异常:

@RestControllerAdvice 表示对当前类进行了异常方面的增强

只要某个类上有@ResController注解,就会对他进行异常处理

@ExceptionHander 异常的类型必须是…中定义的自定义异常类

然后返回自定义的结果集…

# 正常结果集处理:

当某个类有@RestController注解,就会走到异常处理所实现的@ResponseBodyAdvice方法中,执行两个方法 第一分方法,是判断方法或类上是否有自定的注解,,如果有就返回false,如果没有就返回true,一旦返回true,就会执行第二个方法,第二个方法中传一个Object类型参数,,,然后在进行判断传过来的参数是否是null,如果过是空,返回new的结果集,判断传来的值是否是定义的结果集类型,如果是,就进行转化,,,以上两种都不是,就将参数值赋给new的结果集中的data属性