SSM知识点报告

Spring学习：  
  spring的优点 降低耦合度，容器提供单利模式支持，对于主流框架提供主流支持，独立于各种服务器，低侵入式设计 代码的污染度极低  
Spring核心IOC(控制反转)/DI(依赖注入) AOP(面向切面)：  
    IOC:Spring容器利用工厂模式为我们创建所需的对象 需要时无需创建 直接我们调用Spring为我们创建的对象可

DI:  
  Spring为我们提供了三种注入方式 分别是set注入 构造注入和接口注入  
AOP:  
 在java面向对象（OOP）时我们将事务抽象成一个个对象 而面向切面是我们将一个个对象某些类似的方面抽象成切面  
总结：  
  Spring 就是想让代码和代码直接 模块和模块直接没有任何关联 通过配置文件管理  
Spring中的bean 的作用域：  
  singleton bean 在每个spring只request每次http请求都会创建一个bean 它只会在 Spring ApplicationContext 下有效  
session 每次bean 对应一个实例怎么开启注解装配<context:annotation-config/>默认是不开启的  
 MyBatis：  
   1.Mybatis 不完全是一个ORM框架(Object Relational Mapper) 它将Java对象和数据库表关联起来 将结果集和入参都封装到pojo类中Hibernate 是一个完全的ORM框架技术 Hibernate概念中java的三种状态（持久、游离、瞬时）  
 瞬时状态（Transient） ：不和session相关联。  
   2.在数据库中没有和瞬时对象关联的记录游离状态（Detached）：本质上和瞬时状态相同  
3.只是比瞬时状态多了个数据库标识的id持久状态（Persistent）：和session相关联  
 4.在数据库中有瞬时对象关联的记录#{}和${}的区别， #{}是预编译、${}是占位符推荐使用#{}可以防止sql注入 提高安全性  
 5.当实体类中的属性和数据库中的列名不一致时怎么办第一种方法查询的sql语句起一个别名第二种方法通过<resultMap>一一关联  
 6.通过什么来进行一对一查询  以及一对多查询通resultMap中的association来进行查association 的属性perproty  映射的数据库字段或者列名JavaType  完整类名或别collection 的属性  ofType  完整的类名或者别名  
  
  
SSM 框架面试知识点  
 1.谈谈你对SpringMVC的理解它是基于组件技术的 全部的应用对象 无论是控制器 视图 组件都是基于java开发的 并且与Spring紧密相连  
 2.不依赖Spring API 但是实现的时候依赖于Servle

1. 可以使用任何视图  不限于JSP开发  
   4.各种资源请求映射策略  
   5.易扩展  
   SpringMVC 和struts2的区别  
   1.SpringMVC的入口是Servlet  而struts2是基于filter过滤器  
   2.SpringMVC是基于方法开发的  请求的参数是传到形参上面 可以设置单利或者多利开发（建议单利）struts2是基于类开发 传递参数是类的属性  只能是多例的  
    3.struts 采用值栈存储请求和响应的数据 而SpringMVC 是通过RequestMapper找到对应方法  把值传到形参上面得到数据jdbc的理解以及存储过程  
     是在大型数据库系统中，一组为了完成特定功能的SQL语句集，存储在数据库中 经过第一次编译后再次调用不需要再次编译，用户通过制定存储过程的名字并给出参数来执行它，存储过程是数据库中的一个重要对

存储过程的优点  
  1.存储过程只在创造时进行编译 以后每次执行存储过程不需要重新编译 所以存储过程客提高数据./库执行速度  
 2.安全性高

3.可重复使用

4.可将复杂操作存储过程封装起来与数据库提供的事务处理结合一起使用

实现类

ClassPathXmlApplicationContext：对应类路径下的xml配置文件

FileSystemXmlApplicationContext：对应文件系统中的XML格式的配置文件(也就是说这个配置文件可以不放在项目中，而是从磁盘中加载)

AnnotationConfigApplicationContext: 当我们使用注解配置容器对象时，需要使用此类来创建 spring 容器。它用来读取注解

IOC中bean标签

bean 标签 :

作用：用于配置对象让spring来创建。默认的情况下调用的是这个类的无参构造器，如果说没有无参构造器，这个类就不能被加载进容器。

属性：

|  |  |
| --- | --- |
| id | 给对象在容器中提供一个唯一标识符，用于获取对象 |
| class | 指定类的全限定类名，用于反射创建对象 |
| scope | 指定对象的作用范围（下面有一个表专门细说这个） |
| init-method | 指定类中的初始化方法名称（实例化时候被调用） |
| destroy-method | 指定类中销毁的方法命名（销毁时候被调用） |

bean 的作用范围和生命周期 （scope）

|  |  |
| --- | --- |
| singleton | 默认值，表示这个类是一个单例 |
| prototype | 表示这个类是多例 |
| request | WEB 项目中,Spring 创建一个 Bean 的对象,将对象存入到 request 域中 |
| session | WEB 项目中,Spring 创建一个 Bean 的对象,将对象存入到 session 域中 |
| global session | WEB 项目中,应用在 Portlet 环境.如果没有 Portlet 环境那么 globalSession 相当于 session |

两个最为重要的值scope="singleton" 和scope="prototype"

单例对象：scope="singleton"

一个应用只有一个对象的实例

生命周期

对象出生：当应用加载，创建容器的时候，对象就被创建了

对象存活：只要容器在，这个对象就一直存在

对象死亡：当应用卸载，容器销毁，这个就没了

多例对象：scope="prototype"

每次访问获取这个对象的时候，都会创建新的实例

生命周期

对象出生：当使用对象时，创建新的对象实例（容器创建的时候不会被创建）

对象存活：只要对象在使用中，就一直活着

对象死亡：当对象长时间不用时，被 java 的垃圾回收器回收

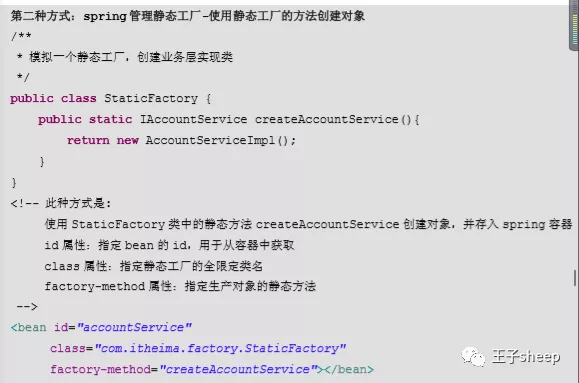
实例化Bean的三种方式

使用默认的无参构造函数

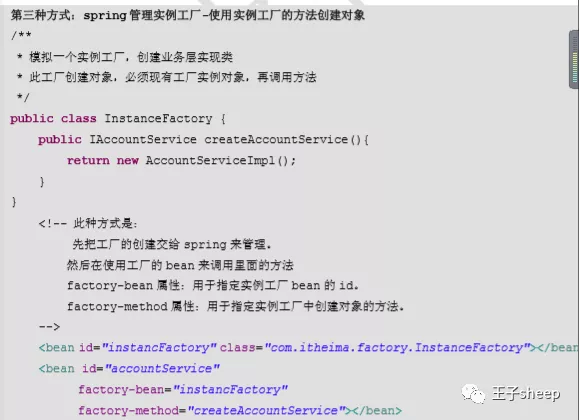
这种方式在没有无参构造的时候不能创建

<bean id="accountService" class="com.itheima.service.impl.AccountServiceImpl"/>

spring管理静态工厂，使用静态工厂的方法创建对象



第三种方式，spring管理实例工厂，使用实例工厂的方法创建对象



其中第一种方法最为常用。

03

spring 的依赖注入

什么是依赖注入

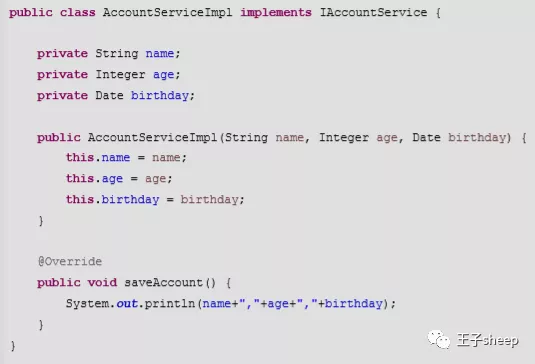
依赖注入：Dependency Injection。它是 spring 框架核心 ioc 的具体实现。我们的程序在编写时，通过控制反转，把对象的创建交给了 spring，但是代码中不可能出现没有依赖的情况。ioc 解耦只是降低他们的依赖关系，但不会消除。例如：我们的业务层仍会调用持久层的方法。那这种业务层和持久层的依赖关系，在使用 spring 之后，就让 spring 来维护了。简单的说，就是坐等框架把持久层对象传入业务层，而不用我们自己去获取

构造函数注入

就是使用类中的构造函数，给成员变量赋值。

注意：赋值的操作不是我们自己做的，而是通过配置 的方式，让 spring 框架来为我们注入

假如有这么一个类



使用构造函数的方式给成员变量赋值

要求：类中需要提供一个对应参数列表的构造函数

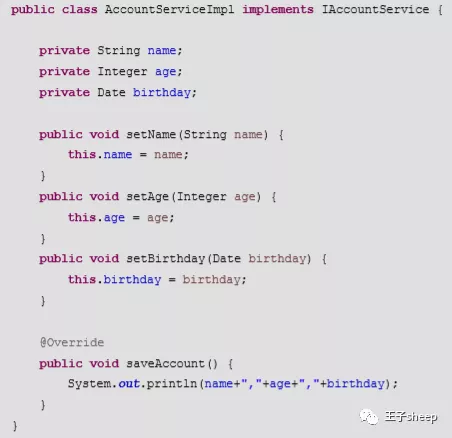
涉及的标签：  <constructor-arg>，这个标签的几个属性

|  |  |
| --- | --- |
| index | 指定参数在构造函数参数的索引位置 |
| type | 指定参数在构造函数中的数据类型 |
| name | 指定参数在构造函数中的名称 |
| 上面这几个都是给哪个参数赋值，一般index和type合起来用 | |
| value | 能给赋值的类型是基本数据类型和string类型 |
| ref | 赋值为引用类型，也就是这个数据也在容器中 |

<bean id="accountService" class="com.service.impl.AccountServiceImpl">  <constructor-arg name="name" value=" 张三 "></constructor-arg> <constructor-arg name="age" value="18"></constructor-arg> <constructor-arg name="birthday" ref="now"></constructor-arg> </bean> <bean id="now" class="java.util.Date"></bean>

 例如上面类利用构造函数赋值就是这样。

set 方法注入 通过实体类中的set方法设置这个值  
还是这个类来说明



通过配置文件给 bean 中的属性传值：使用 set 方法的方式

涉及的标签：   property  属性如下：

|  |  |
| --- | --- |
| name | set方法后面的部分，一般都是属性名 |
| ref | 引用赋值的其他bean类型 |
| value | 基本数据类型和string类型 |

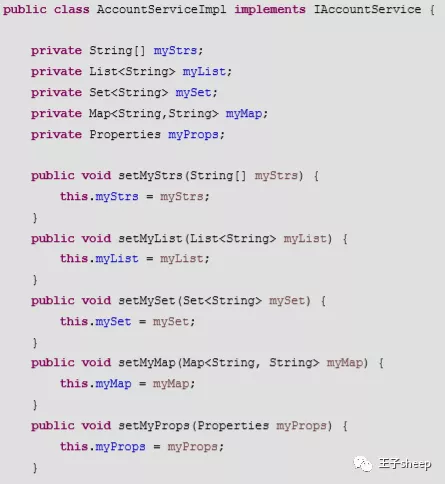
<bean id="accountService" class="com.itheima.service.impl.AccountServiceImpl"> <property name="name" value="test"></property> <property name="age" value="21"></property>   <property name="birthday" ref="now"></property> </bean> <bean id="now" class="java.util.Date"></bean>

还有最后一种是p名称空间注入，本质上就是调用set方法，也不常用，就不介绍了

最后是注入复杂的注入集合属性

就是给类中的集合成员传值，它用的也是set方法注入的方式，只不过变量的数据类型都是集合。我们这里介绍注入数组，List,Set,Map,Properties

例如有这么一个类



好了至此这个基本注入方式和基本的知识点也就差不多了

补充一个知识点：

**spEl表达式（#{}）**

引用bean，例如根据bean的id获取bean    <property name="car" value="#{car}">这里的car是对应配置中的一个id为car的<bean>

引用其他bean的属性值，直接bean的id.属性名

调用静态或者非静态方法

使用运算符