Spring

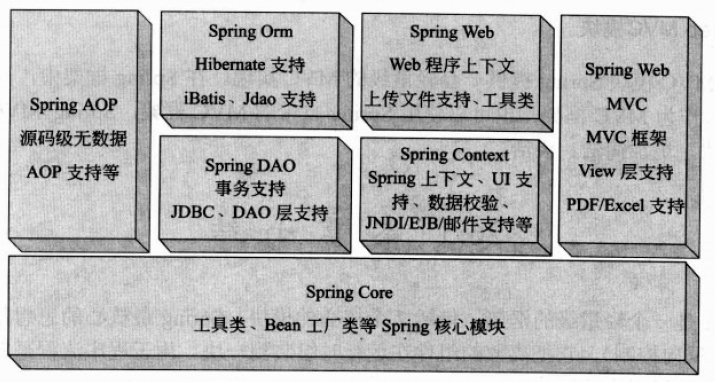
1. Spring框架是个轻量级的j2ee框架。所谓轻量级，是指不依赖于容器就能运行的，也不依赖于特定的规范如J2EE规范等。Struts、Hibernate也是轻量级的。Spring以IOC、AOP为主要思想，能够协同Struts、Hibernate、WebWork、jsf、ibatIS等众多的框架。不同于Struts、hibernate等，Spring不提供某种功能，它只是将所有的组件部署到Spring中，管理、维护、执行它们。Spring并不是官方的框架，而是Rod Johnson领导的开源的、免费的、民间的框架。
2. 销毁生成的bean的方法：factory.destroySingletons();
3. 反向控制的原理

在编码阶段，既没有实例化对象，也没有设置依赖关系，而把它交给Spring，由Spring在运行阶段实例化、组装对象。这种做法颠覆了传统的写代码实例化、组装对象、然后一步步执行的做法，因此，被称为反向控制，或者反转控制。

1. 面向切面编程的原理

面向切面的编程的思想就是在执行某些代码（目标方法）前执行另外的代码（通知），使程序更灵活、扩展性更好，可以随便地添加、删除某些功能。Servlet中的Filter便是一种Aop思想的实现。Spring提供非常灵活的Aop机制。

1. spring的拦截器
2. Spring框架的组成



1. core模块，是框架的核心类库，Spring的所有的功能均依赖于该类库，主要实现Ioc功能。Spring的所有功能都是借助IOC实现的。
2. AOP模块，Spring的Aop库，提供了AOP机制，并提供各种常用的拦截器，允许自定义、配置方法拦截器、拦截的对象等。
3. ORM模块，提供对常用的ORM框架的管理、辅助支持。Spring支持Hibernate、iBatis、JDO等各种ORM框架。Spring并不提供自己的ORM实现，只是对现有的ORM框架进行封装，并提供对它们的管理，例如，事务管理等。
4. Dao模块，提供JDBC的支持，对JDBC进行了封装，允许JDBC使用Spring的资源，并能统一管理JDBC的事务。Spring也不提供JDBC实现。
5. Web模块，提供对Struts、WebWork、JSF等各种Web框架的支持。Spring能够管理这些框架，将Spring的资源如数据源、Bean等注射给框架，也能在执行框架方法前后插入Spring的拦截器。
6. Context模块，提供框架式Bean访问方式，其它程序可以通过Context访问Spring的Bean资源，类似于JNDI。
7. Web MVC模块，提供一套轻量级的MVC实现。在Spring框架中，开发者可以选择Struts作为MVC框架，也可以使用Spring自带的MVC框架。SpringMVC与Struts等框架相比，更加简洁、灵活。
8. springioc

控制反转:把对象的创建、初始化、销毁等工作交给spring容器来做。由spring容器控制对象的声明周期。好处：程序员不必再关心对象的创建，更专注业务逻辑

1. 创建对象的三种方式
2. 默认的无参构造函数
3. 静态工厂

<bean id="helloWorld\_Factory" class="cn.itheima03.spring.ioc.create.HelloWorldFactory" factory-method="getInstance"></bean>

1. 实例工厂
2. Spring的scope
3. spring创建对象默认是单例模式
4. 多例模式需要指定scope为prototype
5. 还有web项目中涉及的request、session、Global session
6. 初始化对象的时机
7. 在默认情况下，spring容器启动的时候，就创建对象了。

有利于开发，当spring容器启动的时候，如果配置错误就会报错。

1. 当从spring容器中获取bean，当bean调用方法的时候创建对象。如果所有的bean按照这种模式加载，spring容器启动的时候是不会报错的，如果一个bean中有大量的数据，需要的是用到的情况下加载。
2. Spring容器的初始化和销毁：init

Lazy-init=”true or false”

前提条件：spring中的bean是单例的

在一个bean中可以提供init方法和destroy方法，创建对象完成之后，执行init指定的方法，当spring容器关闭的时候，执行destroy方法，该方法由spring容器来执行，由该方法释放资源。

在多例模式下，spring容器不负责对象的销毁，由程序员把该对象设置为null，不管在配置文件中lazy-init设置成什么样的值，在context.getBean时才需要创建对象，而且不负责销毁。

1. DI的三种方式
2. 使用构造器注入
3. 使用属性setting方法注入

原理：

1、启动spring容器

2、spring容器为person创建对象

3、利用property中的name属性可以把set方法拼接出来

4、利用java的反射机制，给属性赋值

说明：在默认的情况下，第3、4步在spring容器启动的时候完成了，把这样的过程叫做bean的装配

说明：把list,set,map的值写在配置文件中，大部分的值都是固定的，用来装配这些集合的。例如：可以把hibernate所有的映射文件装配在list中，而这个装配可以写在配置文件中。

1. Bean的注解注入

@Autowired默认按类型装配，@Resource默认按名称装配，当找不到与名称匹配的bean才会按类型装配。如果按名称装配，可以结合@Qualifier注解一起使用。@PostConstruct指定bean的初始化方法，@PreDestroy指定bean的销毁方法。

@Service用于标注业务层组件、

@Controller用于标注控制层组件（如struts中的action）、

@Repository用于标注数据访问组件，即DAO组件。

而@Component泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

springIOC和DI的意义。参见文档系统和mvc系统：实现完全的面向接口编程

1. 面向切面编程
2. 代理模式——静态代理、JDK动态代理（代理对象和目标对象实现了共同的接口）、CGLIB做代理（代理对象是目标对象的子类）、javasisit、Spring的动态代理。
3. 切面的几个名词

切面、通知是切面中的方法、目标类、目标接口、切入点、织入是形成代理对象的方法体的过程。

1、代理对象的方法=通知+目标方法

2、只有符合切入点的时候，才让通知与目标方法结合

3、通知就是切面中的方法

4、动态代理模式做到了通知(事务)与目标方法(cud的操作)的松耦合

Jdk动态代理的缺点

在拦截器中,切入点的判断是非常复杂的；虽然实现了切面与目标类的松耦合，但是在拦截器中还得实现结合过程。

Spring容器内部会自动判断：如果目标类实现了接口，则采用jdkproxy；如果目标类没有实现接口，则采用cglibproxy

通知

前置通知

1、在目标方法执行之前执行

2、方法中可以有一个参数：PointJoint，从这个参数中可以获取目标方法的信息

3、前置通知的应用场合：

1、可以执行日志

2、如果目标方法涉及到数据库的操作，前置通知可以做开启事务的操作

3、如果要统计一个方法被访问的次数，可以通过前置通知设计计数器来实现

后置通知

1、在目标方法执行之后执行

2、可以获取目标方法的返回值

3、如果目标方法遇到异常，后置通知将不再执行

4、应用场合：

1、可以在后置通知中提交事务

2、在完成业务逻辑操作以后，可以执行一些额外的操作：发email等

最终通知

1、在目标方法执行之后执行

2、无论目标方法是否遇到异常，都执行

3、相当于代码中的finally

异常通知

1、可以获取目标方法抛出的异常

2、应用场合：

在mvc结构中，如果把service层的所有的类当做目标类，该通知可以成为异常处理的统一接口

环绕通知

1. 可以控制目标方法的执行

3、说明

1、在spring容器中，spring容器会自动根据目标类是否实现接口，选择采用哪种动态代理实现代理对象

1、目标类实现了接口，则采用jdk动态代理

代理对象和目标对象实现了相同的接口

2、目标类没有实现接口，则采用cglib动态代理

代理类是目标类的子类

2、我们可以利用springAOP很好的把hibernate中的重复性代码抽取出来

1、写一个事务类 切面

两个方法：

开启事务 前置通知

提交事务 后置通知

2、目标类

3、代理类

生成的代理类的方法：

1、开启事务

2、执行数据库的操作

3、事务的提交

3、我们可以利用springAOP的特点，给目标类创建异常处理切面

4、在目标类的上下文可以建立很多个切面，这些切面可以处理：事务、日志、计数器、缓存等

4、扩展

AOP在企业中做到了：

日志可以由专门的人处理，因为现在日志的数量和种类越来越多了

事务由专门的人处理，现在的系统，事务变得很复杂

缓存由专门的人处理：有些人估计更擅长于缓存的操作

业务逻辑由专门的人处理：有些人估计对需要把握特别准确

在客户端生成的代理对象的方法把这些内容全部整合在了一起，但是上面的各个内容是松耦合的

Spring的申明式事物处理

原理是基于aop的思想。声明式是程序员对目标方法进行声明。事务处理，spring容器负责处理事务。切面是由spring容器来完成的，目标类是程序员写的service层的各个方法。

目的是让程序员不再关注事务，而只关注做自己的事情。

利用aop可以让程序员做数据库的操作，让spring容器做事务。

JBPM的handler类中能用spring声明式事务处理吗？不行。

3、spring中处理事务的框架

1、顶级接口

PlatformTransactionManager

getTransaction

commit

rollback

2、抽象类

AbstractPlatformTransactionManager

对commit、rollback方法进行了实现

3、具体类

DataSourceTransactionManager，HibernateTransactionManager等

4、说明：

在spring内部，都是面向接口编程(这点从HbiernateTemplate中可以体会),而接口的实现类是需要通过spring配置文件注入进去的，所以任何一个数据库操作框架只要知道spring的结构，很容易整合到spring容器中

4、spring声明式事务处理的缺点

从原理上可以看出，如果一个类的方法中有数据库的操作，如果该类的事务处理想交给spring容器来做，那么必须满足：

1、把该类放入到spring容器中

2、为该类生成代理对象

如果存在这样的情况：

例如在jbpm工作流中，有些类是由jbpm内部调用的，但是这些类的方法中如果需要事务呢？答案是用spring容器做事务做不到

5、关于声明式事务处理的session

由于必须保证数据库操作的session和事务的session是同一个session,所以session必须由当前线程产生

事务处理对应的包是