/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spring \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
框架就是你在实际开发中，可以让使用者减少很多重复的代码、让代码的结构更加清晰，耦合度更低，后期维护方便  
一、Spring框架概述  
 1、Spring是轻量级的开源的JavaEE框架.  
 2、Spring可以解决企业应用开发的复杂性.  
 3、Spring有两个核心部分：IOC和Aop.  
 IOC：控制反转(Inversion of Control)把创建对象过程交给Spring进行管理.9  
 Aop: 面向切面(AOP为Aspect Oriented Programming)不修改源代码进行功能增强.  
  
 4、Spring特点  
 (1) 方便解耦，简化开发  
 (2) Aop编程支持  
 (3) 方便程序测试  
 (4) 方便和其他框架进行整合  
 (5) 方便进行事务操作  
 (6) 方便API开发难度(如：JDBC进行封装)  
  
 |-- 导入spring jar包  
 |-- core container  
 |-- beans spring-beans-5.2.6.RELEASE.jar  
 |-- core spring-core-5.2.6.RELEASE.jar  
 |-- context spring-context-5.2.6.RELEASE.jar  
 |-- expression spring-expression-5.2.6.RELEASE.jar  
 |-- 日志包 commons-logging-1.1.1.jar  
  
 5、IOC(概念和原理)  
 5.1、  
 (1) 控制反转、把对象创建和对象之间的调用过程，交给Spring进行管理  
 (2) 使用IOC目的：为了耦合度降低  
 (3) 做入门案列就是IOC实现  
  
 5.2、IOC容器  
 (1) IOC底层原理  
 (2) IOC接口(BeanFactory)  
 (3) IOC操作Bean管理(基于xml)  
 (4) IOC操作Bean管理(基于注解)  
  
  
 5.3、IOC底层原理  
 (1) xml解析、工厂模式、反射  
  
 5.4、IOC(接口)  
 (1) IOC思想基于IOC容器完成，IOC容器底层就是对象工厂  
 (2) Spring提供IOC容器实现两种方式：两个接口  
 |-- BeanFactory: IOC基本实现方式，是Spring内部使用接口，不提供开发人员进行使用  
 加载配置文件的时候不会创建对象，在获取（使用）对象的时候才会去创建对象  
 |-- ApplicationContext:BeanFactory接口的子接口，提供更多强大的功能，一般由开发人员进行使用  
 加载配置文件的时候就会把配置文件对象进行创建  
 |--ApplicationContext接口的实现类  
 |-- FileSystemXmlApplicationContext //配置文件在盘符下的路径  
 |-- ClassPathXmlApplicationContext //配置文件在src下的内容  
  
 6、IOC操作Bean管理(基于xml方式)  
 6.1 什么是Bean管理  
 (1) Bean管理是指两个操作：Spring创建对象，Spring注入属性  
 6.2 基于xml配置文件方式实现  
 |-- 基于xml方式创建对象  
  
 |-- 基于xml方式注入属性  
 | DI:依赖注入，就是注入属性  
 |-- 第一种注入方式：使用set方法进行注入  
 |-- 第二种注入方式：使用有参构造进行注入  
  
  
 6.2 Spring 有两种类型bean,一种是普通bean,另一种是工厂bean(FactoryBean)  
 |-- 普通bean: 再配置文件中定义bean类型就是就是返回类型  
 |-- 工厂bean: 在配置文件中定义bean类型可以和返回类型不一样  
  
 6.3 bean作用域  
 在Spring里面，默认情况下，bean是单实例对象  
  
 如何设置多实例，在bean标签内有scope属性  
 单实例多次创建对象相同即只有一种实例  
 多实例多次创建对象不相同即可以创建多种实例  
 singleton单实例，prototype多实例  
 spring默认的是一个单实例对象  
 区别：设置scope值是singleton的时候，加载spring配置文件时候就会创建单实例对象  
 设置scope值是prototype的时候，不是在加载spring配置文件时创建对象，在调用getBean方法时候创建对象 为多实例对象  
  
 6.4 bean标签属性autowire,配置自动注入装配  
 autowire属性常用的两个值：  
 byName根据属性名称注入: 注入值bean的id值和类属性名称一样  
 byType根据属性类型注入  
  
 7、IOC操作Bean管理(基于注解方式)  
 javax.annotation-api-1.3.2.jar  
 spring-aop-5.2.6.RELEASE.jar  
 (1) Spring 针对 Bean 管理中创建对象提供注解  
 @Component  
 @Service  
 @Controller  
 @Repository  
 (2) 基于注解方式实现属性注入  
 ① @Autowired：根据属性类型进行自动装配  
 ② @Qualifier：根据名称进行注入  
 这个@Qualifier 注解的使用，和上面@Autowired 一起使用  
 ③ @Resource：可以根据类型注入，可以根据名称注入  
 ④ @Value：注入普通类型属性  
二、AOP  
 1、  
 ① 面向切面编程， 利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率  
 通述描述：不通过修改源代码方式添加新的功能  
 ② AOP底层使用动态代理  
 |-- 有两种情况动态代理：  
 |-- 第一种情况：有接口实现类代理对象，使用JDK动态代理  
 |-- 第二种情况：没有接口情况，使用CGLIB动态代理  
  
 2、AOP(术语)  
 ① 连接点：类里面哪些方法可以增强，这些方法称为连接点  
 ② 切入点：实际被真正增强的方法，称为切入点  
 ③ 通知(增强)：实际增强的逻辑部分称为通知(增强)  
 |-- 通知有多种类型  
 |-- 前置通知 后置通知 环绕通知 异常通知 最终通知  
  
  
 ④ 切面：是动作 把通知应用到切入点过程  
  
 3、AOP操作  
 (1) Spring 框架一般都是基于AspectJ实现AOP操作  
 ① AspectJ不是Spring组成部分，独立AOP框架，一般把AspectJ和Spring框架一起使用，进行AOP操作  
 (2) 基于AspectJ实现AOP操作  
 ① 基于xml配置文件实现  
 ② 基于注解方式实现(使用)  
 (3) 在项目工程里面引入AOP相关依赖  
 ① spring-aspects-5.2.6.RELEASE.jar  
 ② com.springsource.net.sf.cglib-2.2.0.jar  
 ③ com.springsource.org.aopalliance-1.0.0.jar  
 ④ com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar  
  
 4、切入表达式  
 (1) 切入表达式作用：知道对哪个类里面的哪个方法进行增强  
 (2) 语法结构  
 execution([权限修饰符][返回类型][方法名称]([参数列表]))  
 举例1：对com.atguigu.dao.BookDao类里面的add进行增强  
 execution(\* com.atguigu.dao.BookDao.add(..))  
 举例2：对com.atguigu.dao.BookDao类里面的所有方法进行增强  
 execution(\* com.atguigu.dao.BookDao.\*(..))  
 举例3：对com.atguigu.dao包里面的所有类 类里面的所有方法进行增强  
 execution(\* com.atguigu.dao.\*.\*(..))  
三、JdbcTemplate  
 (1) JdbcTemplate: Spring框架对JDBC进行封装，使用JdbcTemplate更方便实现对数据库操作  
 (2) 准备工作：① 引入相关jar包，② 在spring配置文件配置数据库连接池  
 ③ 配置JdbcTemplate对象，注入DataSource, ④ 创建service类，创建dao类，在dao注入JdbcTemplate对象  
 (3)① JdbcTemplate对象里面update(sql,Obejct...args)方法实现数据的添加，修改，删除操作  
 ② JdbcTemplate对象里面queryForObject(String sql,Class<T> requiredType)实现查询返回某个值  
 ③ JdbcTemplate对象里面queryForObject(String sql,RowMapper<T> rowMapper,Object...args)实现查询返回对象  
 ④ JdbcTemplate对象里面query(String sql,RowMapper<T> rowMapper,Object...args)方法实现查询返回集合  
 ⑤ JdbcTemplate对象里面batchUpdate(String,List<Object[]> batchArgs)方法实现对数据的批量添加，修改，删除操作  
  
  
四、事务概念  
 (1) 事务是数据库操作最基本单元，逻辑上一组操作，要么都成功，如果有一个失败所有操作都失败  
 (2) 事务四个特性(ACID)  
 ① 原子性(automicity)  
 ② 一致性(consistency)  
 ③ 隔离性(isolation)  
 ④ 持久性(durability)  
 (3) 建议事务添加到JavaEE三层结构里面Service层(业务逻辑层)  
 (4) Spring进行事务管理操作有两种方式：编程式管理和声明式管理(使用)  
 (5) 声明式事务管理  
 ① 基于注解方式(使用)  
 ② 基于xml配置文件方式  
 (6) 在Spring进行声明式事务管理，底层使用AOP原理  
 (7) @Transactional,这个注解添加到类上面，也可以添加到方法上面  
 ① 如果把这个注解添加到类上面，这个类里面所有的方法都添加事务  
 ② 如果把这个注解添加到方法上面，只为这个方法添加事务  
 (8) 事务操作(声明式事务管理参数配置)  
 ① propagation: 事务传播行为  
 |-- 事务方法：对数据库表数据进行变化的操作  
 |-- Spring框架事务传播行为有7种  
 REQUIRED 如果事务方法里面调用的其他方法没有添加事务注解本身方法添加了注解，在调用里面的方法时，会创建新事务  
 REQUIRED\_NEW 无论方法是否有事务，都会创建新的事务  
 ② isolation:事务隔离级别  
 脏读：一个未提交事务读取到另一个未提交事务的数据  
 不可重复读：一个未提交事务读取到另一个提交事务修改数据  
 幻读：一个未提交数据事务读取到另一个提交事务添加数据  
 |-- 通过设置隔离级别解决读的问题  
 read uncommitted: 读未提交  
 read commit: 读已提交  
 repeatable read: 可重复读  
 serializable: 序列化  
 ③ timeout: 超时时间  
 事务需要在设定的时间内进行提交，如果不提交进行回滚  
 默认是是-1 就是不超时  
 ④ readOnly:是否只读  
 |-- 读：查询操作 写：添加修改删除操作  
 |-- 默认值是false 表示可以查询添加修改删除操作  
 |-- 设置值为true之后，只能查询  
 ⑤ rollbackFor: 回滚  
 |-- 设置出现哪些异常进行事务回滚  
 ⑦ noRoolbackFor  
 |-- 设置出现哪些异常不进行事务回滚  
  
 (9) 事务操作(xml声明式事务管理)  
 ① 配置事务管理器  
 ② 配置通知  
 ③ 配置切入点和切面  
  
  
##############################易错易忘点###########################  
什么是 XML?  
 XML 指可扩展标记语言（EXtensible Markup Language）  
 XML 是一种标记语言，很类似 HTML  
 XML 的设计宗旨是传输数据，而非显示数据  
 XML 标签没有被预定义。您需要自行定义标签。  
 XML 被设计为具有自我描述性。  
 XML 是 W3C 的推荐标准  
  
XML 与 HTML 的主要差异  
 XML 不是 HTML 的替代。  
 XML 和 HTML 为不同的目的而设计：  
 XML 被设计为传输和存储数据，其焦点是数据的内容。  
 HTML 被设计用来显示数据，其焦点是数据的外观。  
 HTML 旨在显示信息，而 XML 旨在传输信息。  
  
HTML是一种超文本传输协议，规定了浏览器与服务端之间数据传输的格式，是一种标识性的代码语言，  
 它的中文翻译是“超文本标记语言”，主要是通过HTML标签对网页中的文本、图片、声音等内容进行描述。  
  
  
1、通过xml创建对象，给属性赋值，一定要有set方法——即通过set方法注入属性的值  
2、<bean class="com.atguigu.service.UserService"> 在不写id标签的时候spring会默认id为类名 其首字母小写  
 UserService userService = context.getBean(UserService.class); 也可以不写name参数

**2.1、IoC(控制反转)**

　　首先想说说**IoC（Inversion of Control，控制反转）**。这是**spring的核心**，贯穿始终。**所谓IoC，对于spring框架来说，就是由spring来负责控制对象的生命周期和对象间的关系。**这是什么意思呢，举个简单的例子，我们是如何找女朋友的？常见的情况是，我们到处去看哪里有长得漂亮身材又好的mm，然后打听她们的兴趣爱好、qq号、电话号、ip号、iq号………，想办法认识她们，投其所好送其所要，然后嘿嘿……这个过程是复杂深奥的，我们必须自己设计和面对每个环节。传统的程序开发也是如此，在一个对象中，如果要使用另外的对象，就必须得到它（自己new一个，或者从JNDI中查询一个），使用完之后还要将对象销毁（比如Connection等），对象始终会和其他的接口或类藕合起来。

　　那么IoC是如何做的呢？有点像通过婚介找女朋友，在我和女朋友之间引入了一个第三者：婚姻介绍所。婚介管理了很多男男女女的资料，我可以向婚介提出一个列表，告诉它我想找个什么样的女朋友，比如长得像李嘉欣，身材像林熙雷，唱歌像周杰伦，速度像卡洛斯，技术像齐达内之类的，然后婚介就会按照我们的要求，提供一个mm，我们只需要去和她谈恋爱、结婚就行了。简单明了，如果婚介给我们的人选不符合要求，我们就会抛出异常。整个过程不再由我自己控制，而是有婚介这样一个类似容器的机构来控制。**Spring所倡导的开发方式**就是如此，**所有的类都会在spring容器中登记，告诉spring你是个什么东西，你需要什么东西，然后spring会在系统运行到适当的时候，把你要的东西主动给你，同时也把你交给其他需要你的东西。所有的类的创建、销毁都由 spring来控制，也就是说控制对象生存周期的不再是引用它的对象，而是spring。对于某个具体的对象而言，以前是它控制其他对象，现在是所有对象都被spring控制，所以这叫控制反转。**

**2.2、DI(依赖注入)**

**IoC的一个重点是在系统运行中，动态的向某个对象提供它所需要的其他对象。这一点是通过DI（Dependency Injection，依赖注入）来实现的**。比如对象A需要操作数据库，以前我们总是要在A中自己编写代码来获得一个Connection对象，有了 spring我们就只需要告诉spring，A中需要一个Connection，至于这个Connection怎么构造，何时构造，A不需要知道。在系统运行时，spring会在适当的时候制造一个Connection，然后像打针一样，注射到A当中，这样就完成了对各个对象之间关系的控制。A需要依赖 Connection才能正常运行，而这个Connection是由spring注入到A中的，依赖注入的名字就这么来的。那么DI是如何实现的呢？ Java 1.3之后一个重要特征是反射（reflection），它允许程序在运行的时候动态的生成对象、执行对象的方法、改变对象的属性，spring就是通过反射来实现注入的。

理解了IoC和DI的概念后，一切都将变得简单明了，剩下的工作只是在spring的框架中堆积木而已。

反射

**三、我对IoC(控制反转)和DI(依赖注入)的理解**

　　在平时的java应用开发中，我们要实现某一个功能或者说是完成某个业务逻辑时至少需要两个或以上的对象来协作完成，在没有使用Spring的时候，每个对象在需要使用他的合作对象时，自己均要使用像new object() 这样的语法来将合作对象创建出来，这个合作对象是由自己主动创建出来的，创建合作对象的主动权在自己手上，自己需要哪个合作对象，就主动去创建，创建合作对象的主动权和创建时机是由自己把控的，而这样就会使得对象间的耦合度高了，A对象需要使用合作对象B来共同完成一件事，A要使用B，那么A就对B产生了依赖，也就是A和B之间存在一种耦合关系，并且是紧密耦合在一起，而使用了Spring之后就不一样了，创建合作对象B的工作是由Spring来做的，Spring创建好B对象，然后存储到一个容器里面，当A对象需要使用B对象时，Spring就从存放对象的那个容器里面取出A要使用的那个B对象，然后交给A对象使用，至于Spring是如何创建那个对象，以及什么时候创建好对象的，A对象不需要关心这些细节问题(你是什么时候生的，怎么生出来的我可不关心，能帮我干活就行)，A得到Spring给我们的对象之后，两个人一起协作完成要完成的工作即可。

　　所以**控制反转IoC(Inversion of Control)是说创建对象的控制权进行转移，以前创建对象的主动权和创建时机是由自己把控的，而现在这种权力转移到第三方**，比如转移交给了IoC容器，它就是一个专门用来创建对象的工厂，你要什么对象，它就给你什么对象，有了 IoC容器，依赖关系就变了，原先的依赖关系就没了，它们都依赖IoC容器了，通过IoC容器来建立它们之间的关系。

　　这是我对Spring的IoC**(控制反转)**的理解。DI**(依赖注入)**其实就是IOC的另外一种说法，DI是由Martin Fowler 在2004年初的一篇论文中首次提出的。他总结：**控制的什么被反转了？就是：获得依赖对象的方式反转了。**

##############################spring##############################  
  
1、 bean生命周期  
 \* 1、通过无参构造器创建bean实例  
 \* 2、为bean的属性设置值和其他bean引用(调用set方法)  
 \* 3、调用bean的初始化方法(需要进行配置)  
 \* 4、通过getBean获取对象  
 \* 5、当容器关闭的时候，调用bean的销毁的方法(需要进行配置)  
 \*  
 \* 加上后置处理器后bean生命周期一共是七步  
 \* 1、通过无参构造器创建bean实例  
 \* 2、为bean的属性设置值和其他bean引用(调用set方法)  
 \* 3、把bean实例传递bean后置处理器的方法  
 \* 4、调用bean的初始化方法(需要进行配置)  
 \* 5、把bean实例传递bean后置处理器的方法  
 \* 6、通过getBean获取对象  
 \* 7、当容器关闭的时候，调用bean的销毁的方法(需要进行配置)  
 \*  
2、在 Spring 中，构成应用程序主干并由Spring IoC容器管理的对象称为bean。  
bean是一个由Spring IoC容器实例化、组装和管理的对象。  
3、概念简单明了，我们提取处关键的信息：  
 bean是对象，一个或者多个不限定  
 bean由Spring中一个叫IoC的东西管理  
 我们的应用程序由一个个bean构成

#############################SpringMVC#########################  
⭐⭐⭐⭐⭐SpringMVC搭建框架的步骤：  
 ① 在pom.xml中添加需要的依赖  
 ② 在web.xml中配置springMVC的编码过滤器，配置DispatcherServlet前端控制器 并且设置springMVC的配置文件的位置  
 ③ 在resource中创建springMVC配置文件，添加扫描组件，配置thymeleaf的视图解析器  
  
WEB-INF下面的内容都是只能由服务器级别才能访问，客户端并不能访问。什么是客户端级别?什么是服务器级别呢  
  
转发就是服务器级别，浏览器的地址不会变，因为，客户端发送一个请求，服务器受理之后，发现要请求内容还要再去别的请求，  
那么转发就是服务器自己去处理完成。不麻烦客户端(浏览器)了，所以客户端上面的地址栏不会改变。  
重定向：就是客户端级别的。服务器收到请求后，发现还要去请求别的内容，但是服务器自己不想处理，那么就告诉客户端，  
你自己去处理吧，那么客户端就去请求那个别的内容了。所以客户端（浏览器）地址栏就会改变了。  
一、springMVC简介：  
 1、MVC是一种软件架构的思想，将软件按照模型、视图、控制器来划分.  
 M：Model，模型层，指工程中的JavaBean,作用是处理数据  
 JavaBean分为两类Bean:  
 |—— 一类称为实体类Bean: 是专门存储业务数据的，如Student、User  
 |—— 一类称为业务处理Bean: 指Service 或 Dao对象  
 V: View 视图层： 指工程中的html或jsp等页面，作用是与用户进行交互，展示数据  
 C: Controller 控制层：指工程中的servlet,作用是接收请求和响应浏览器  
 MVC工作流程：用户通过视图层发送请求到服务器，在服务器中请求被Controller接收，Controller调用相应的Model层处理请求  
 处理完毕将结果返回到Controller,Controller再根据请求处理的结果找到相应的View视图，渲染数据后最终响应给浏览器  
  
二、三层架构分为表述层（或表示层）、业务逻辑层、数据访问层 表述层表示前台页面和后台servlet  
  
三、① @RequestMapping注解：处理请求和控制器方法之间的映射关系  
 @RequestMapping注解的value属性可以通过请求地址匹配请求，/表示上下文路径：http://localhost:8080/springMVC/  
 ② @RequestMapping注解的位置  
 @RequestMapping标识一个类：设置映射请求的路径的初始信息  
 @RequestMapping标识一个方法：设置映射请求请求路径的具体信息  
 ③ @RequestMapping  
 |-- value属性： 通过请求的请求地址匹配请求映射  
 是一个字符串类型的数组，表示该请求能够匹配多个请求地址所对应的请求  
 value属性必须设置，至少通过请求地址匹配请求映射  
 |-- method属性: method属性通过请求的请求方式(get或post)匹配请求映射  
 如果请求参数不满足method属性，则浏览器报错：405：Request method 'POST' not supported  
  
 |-- params属性：可以通过四种表达式设置请求参数和请求映射的匹配关系  
 "param"：要求请求映射所匹配的请求必须携带param请求参数  
 "!param"：要求请求映射所匹配的请求必须不能携带param请求参数  
 "param=value"：要求请求映射所匹配的请求必须携带param请求参数且param=value  
 "param!=value"：要求请求映射所匹配的请求必须携带param请求参数但是param!=value  
 |--header了解：是一个字符串类型的数组，可以通过四种表达式设置请求头信息和请求映射的匹配关系  
 注：对于处理指定请求方式的控制器，SpringMVC中提供了@RequestMapping的派生注解  
 > 处理get请求的映射--> @GetMapping  
 > 处理post请求的映射--> @PostMapping  
 > 处理put请求的映射--> @PutMapping  
 > 处理delete请求的映射--> @DeleteMapping  
 常用的请求方式有get,post,put,delete  
四、@RequestParam  
 @RequestParam是将请求参数和控制器方法的形参创建映射关系  
 @RequestParam注解一共有三个属性：  
 value：指定为形参赋值的请求参数的参数名  
 required：设置是否必须传输此请求参数，默认值为true  
 若设置为true时，则当前请求必须传输value所指定的请求参数，若没有传输该请求参数，且没有设置  
 defaultValue属性，则页面报错400：Required String parameter 'xxx' is not present；若设置为  
 false，则当前请求不是必须传输value所指定的请求参数，若没有传输，则注解所标识的形参的值为null  
 defaultValue：不管required属性值为true或false，当value所指定的请求参数没有传输或传输的值  
 为""时，则使用默认值为形参赋值  
五、域对象共享数据  
 ① 使用ServletAPI向request域对象共享数据  
 ② 使用ModelAndView向request域对象共享数据  
 ModelAndView有Model和View的功能  
 Model主要用于向请求域共享数据  
 View主要用于设置视图，实现页面跳转  
 ③ 使用Model向request域对象共享数据  
 ④ 使用map向request域共享数据  
 ⑤ 使用ModelMap向request域对象共享数据  
Model、ModelMap、Map类型的参数其实本质上都是BindingAwareModelMap类型  
  
RequestEntity封装请求报文的一种类型，需要在控制器方法的形参中设置该类型的形参  
  
六、拦截器的三个抽象方法  
 preHandle:控制器方法执行之前执行preHandle(),其boolean类型返回值表示是否拦截或者放行，  
 返回true为放行，即调用控制器方法；返回false表示拦截，即不调用控制器方法  
 postHandle:控制器方法执行之后执行  
 afterComplation:处理完视图和模型数据，渲染视图完毕之后执行  
  
  
  
##########################重定向和转发的区别###############################  
1、请求次数：重定向是浏览器向服务器发送一个请求并收到响应后再次向一个新地址发出请求，  
 转发是服务器收到请求后为了完成响应跳转到一个新的地址；重定向至少请求两次，转发请求一次；  
2、地址栏不同：重定向地址栏会发生变化，转发地址栏不会发生变化；  
3、是否共享数据：重定向两次请求不共享数据，转发一次请求共享数据（在request级别使用信息共享，使用重定向必然出错）；  
4、跳转限制：重定向可以跳转到任意URL，转发只能跳转本站点资源；  
5、发生行为不同：重定向是客户端行为，转发是服务器端行为；  
  
  
请求报文，get请求没有请求体，而是拼接在地址栏中 post有请求体  
@RestController标识在控制器的类上，就相当于为类添加了@Controller注解，并且为其中的每个方法添加了@ResponseBody注解  
  
  
  
############################Filter的作用##########################  
1、Filter过滤器它是JavaEE的规范，也就是接口  
2、Filter过滤器的作用：拦截请求，过滤响应  
 拦截请求常见的场景有：权限检查 日记操作 事务管理.....  
  
  
  
  
#############################其他注解############################  
@PathVariable注解：绑定它传过来的值到方法的参数上  
@RequestBody可以获取请求体，需要在控制器方法设置一个形参，使用@RequestBody进行标识，当前请求的请求体就会为当前注解所标识的形参赋值  
@ResponseBody用于标识一个控制器方法，可以将该方法的返回值直接作为响应报文的响应体响应到浏览器