- spring
  - spring核心技术
    - ioc
      - 概念
        - 1.什么是spring
          - spring是java的开源框架， 核心技术是ioc，aop。 spring也是一个容器，容器放的是java对象。
          - 通过spring框架可以实现解耦合。 解决业务对象之间的耦合。也可以解决业务对象和非业务对象之间的耦合
        - 2.IoC:控制反转
          - 把对象的创建，属性赋值，依赖关系都交给代码之外的容器，由容器实现对象的管理。
          - 把控制对象的权利转移给代码之外的容器。IoC是一个理论，概念，思想
          - IoC的重要技术实现DI(依赖注入)： 通过名称就可以获取对象， 对象创建，属性赋值都由容器内部实现
          - Spring使用DI的技术， 底层实现是反射机制
      - 控制反转：把创建对象new 对象，给属性赋值交给容器实现。 spring是一个框架也叫做容器
      - spring是容器：存放对象的，可以存放service，dao，工具类等对象。
      - 怎么使用spring：把程序中要使用的java对象都交给spring，让spring作为一个工厂加工对象。 我们在程序中通过名称获取要使用的对象。
      - 创建对象的注意地方
        - 1.spring默认调用无参数构造方法，创建对象。
        - 2.创建spring容器ApplicationContext时，会把配置文件中所有对象都创建，备用。
      - di依赖注入
        - 依赖注入：就是给类中的简单类型，引用类型的属性赋值
        - di给属性赋值有两种语法
          - 1.在spring的xml配置文件中，使用标签和属性， <bean>和<property>
          - 2.在java的源代码中，使用注解
        - di分类（方式）
          - 1.set注入，也叫做设值注入
            - spring框架调用类中的set方法，通过set方法可以完成属性赋值
            - 1.简单的属性赋值 <property name="set方法set之后的部分" value="此属性值" />
            - 2.引用类型赋值 <property name="set方法set之后的部分" ref="bean的id" />
          - 2.构造注入
            - spring调用类中的有参数构造方法，通过构造方法完成属性赋值
            - <constructor-arg>: 一个标签表示构造方法的 一个参数
              - 1.name ：构造方法的形参名
              - 2.index：构造方法参数的位置，从0开始
              - 3.省略index属性
          - 3.引用类型自动注入
            - spring框架根据byName,byType规则给引用类型赋值
            - 1.byName:按名称注入， 把引用类型的属性名称当做是bean的id使用，从spring容器中获取这个bean赋值给引用类型
            - 2.byType:按类型注入，把和引用类型的数据类型同源的对象，从容器中获取，赋值给引用类型
            - 同源关系
              - 1.类型一样
              - 2.父子类关系的
              - 3.接口和实现类关系的
        - 多配置文件的使用方式
          - 包含关系的配置文件：一个总的文件，包含其他的多个配置文件
          - <import resource="classpath:其他配置文件的路径"/>
          - 使用通配符，<import resource="classpath: spring-*.xml"/>
        - 基于注解的di
          - 使用注解完成对象创建，属性赋值
          - 使用注解的基本步骤：
            - 1.在类中加入注解，例如@Component
            - 2.在spring的配置文件中加入注解驱动的标签 <context:component-scan="包名" />
          - 创建对象的注解
            - 1.@Component：创建对象，调用类的无参数构造方法
            - 2.@Repository:创建dao对象的，访问数据库
            - 3.@Service: 创建service对象， service是业务类，能够执行业务方法，可以有事务的功能
            - 4.@Controller:创建控制器对象， 控制器能够接受用户的请求，显示请求的处理结果
          - 给属性赋值
            - 1.简单类型，使用@Value
            - 2.引用类型
              - 1.@Autowired: spring框架中提供的
                - 1.byType: 默认是byType，从容器中找到同源关系的对象，进行赋值
                - 2.byName:
                  - 1.@Autowired:赋值用的
                  - 2.@Qualifier: 指定对象的名称，使用这个名字的对象，赋值给引用类型
              - 2.@Resource
                - 是jdk中的提供的注解
                - 1.byType: @Resource默认是byName,如果byName失败，再使用byType方式
                - 2.byName:需要指定@Resource的属性name,name是对象的名称
      - aop面向切面编程
        - 概念
          - 1.什么是aop —— 要给业务方法增加功能，在原有的业务方法代码不改变的情况下，增加一些功能。
          - 2.aop作用
            - 1.实现解耦合： 业务功能和其它的非业务功能的解耦合
            - 2.减少重复的代码
            - 3.专注业务方法的实现，不用考虑其他的代码
          - 3.Aspect:切面 —— 给业务方法增加的功能， 切面通常是非业务功能，例如事务功能，日志，权限管理
          - 4.JoinPoint:连接点 —— 业务方法，表示这个方法执行时，要增加切面的功能
          - 5.Pointcut:切入点 —— 是多个连接的集合，表示切面在这些方法执行时， 要增强切面的功能
          - 6.Advice：通知，增强 —— 表示切面功能的执行时间
        - 实现aop的框架
          - 1.spring框架：本身实现了aop的功能，spring框架自身的一些功能，用的是自己的aop
          - 2.aspectj框架：可以注解，xml配置文件实现aop
        - aspectj框架的使用
          - 作用aop，切面的要素
            - 1.切面的功能
            - 2.切面的执行位置 —— 切入点表达式 execution(方法的定义)
            - 3.切面执行的时间 —— 使用注解表示切面代码的执行时间
          - 实现aop的基本步骤
            - 1.加入一个aspectj的依赖 ——
              ```
              <dependency>
              <groupId>org.springframework</groupId>
              <artifactId>spring-aspects</artifactId>
              <version>5.2.5.RELEASE</version>
              </dependency>
              ```
            - 2.定义业务方法
            - 3.定义切面类
              - 1类的上面加入@Aspect
              - 2类的里面加入@Before等注解
            - 4.在spring的配置文件加入对象的声明
          - 1.@Before:前置通知 —— 在目标方法之前先执行的
          - 2.@AfterReturning:后置通知 —— 在目标方法执行之后执行的， 能够获取到目标方法的执行结果
          - 3.@Around:环绕通知 —— 在目标方法的前和后都能增强功能， 可以修改目标方法的执行结果， 控制目标方法是否执行。
          - 4.@AfterThrowing:异常通知 —— 在目标方法抛出异常时执行的，通过Exception类型的参数，能够获取到异常信息
          - 5.@After:最终通知 —— 总是会被执行的。
          - 6.@Pointcut：定义和管理切入点的注解
          - 动态代理 —— 代理对象是指由框架创建的一种内存中的对象，通过这个对象执行方法时，可以执行切面的功能代码。实现功能的增强
        - spring集成mybatis
          - 使用spring的ioc技术，把mybatis框架中使用的对象交给spring创建，管理，赋值
          - 1.使用业务比较有名连接池： 连接池也是一个类，使用的阿里的DruidDataSource —— 创建这个类的对象，使用<bean>
          - 2.创建SqlSessionFactory对象，使用SqlSessionFactoryBean在内部创建的SqlSessionFactory
          - 3.创建dao的代理对象：访问数据库的操作， 使用 MapperScannerConfigurer 在内部创建dao对象， 执行SqlSession.getMapper(接口.class)
        - spring框架处理事务
          - 事务：一系列的操作步骤，多个sql语句的集合。
          - 在程序中事务是放在service类的方法上面的， 因为在service方法里面会使用多个dao执行数据库的操作
          - spring框架中的事务的处理： 是把事务处理抽象为一个通用的模型，对大多数项目都是使用的
            - 把事务的处理委托给事务管理器：PlatformTransactionManager和他的实现类
            - 事务管理器类：
              - 1.使用jdbc或者mybatis框架：DataSourceTransactionManager
              - 2.使用hibernate框架，HibernateTransactionManager
          - 使用事务定义接口（TransactionDefinition）的常量，控制方法的事务行为
            - 1.隔离级别（5个）
              - 1.DEFAULT：采用 DB 默认的事务隔离级别
              - 2.READ_UNCOMMITTED：读未提交
              - 3.READ_COMMITTED：读已提交
              - 4.REPEATABLE_READ：可重复读
              - 5.SERIALIZABLE：串行化
            - 2.传播行为(7)
              - 1.PROPAGATION_REQUIRED：默认的，需要事务，可以使用已经存在的，没有时，创建新事务
              - 2.PROPAGATION_REQUIRES_NEW：需要新事务
              - 3.PROPAGATION_SUPPORTS：支持事务，没有事务也可以
              - 4.PROPAGATION_MANDATORY
              - 5.PROPAGATION_NESTED
              - 6.PROPAGATION_NEVER
              - 7.PROPAGATION_NOT_SUPPORTED
            - 3.超时时间（1） —— 业务方法的最长执行时间
          - 事务实现
            - 1.使用注解@Transactional
              - 1.它是spring框架自己的实现方式， 底层是aop环绕通知
              - 2.实现步骤：
                - 1.加入依赖：spring-tx
                - 2.在配置文件中，声明事务管理器 <bean id="transactionManager" class="...DataSourceTransactionManager">
                - 3.开启事务注解驱动：告诉spring框架现在是用注解处理事务, <tx:annotation-driven transaction-manager="transactionManager" />
                - 4.在service的实现类，public方法的上面加入@Transactional
              - 注解方式：适合中小项目 ， 需要事务的方法不多的情况。
            - 2.使用aspectj框架的aop方式
              - 1.使用aspectj框架的实现aop，管理事务，底层环绕通知
              - 2.实现步骤：
                - 1.加入依赖：spring-aspects
                - 2.在配置文件中，声明事务管理器对象
                - 3.配置事务的通知（切面）：使用<tx:advice>配置方法的事务属性
                - 4.配置增强器：配置切入点表达式
              - 3.适合的是大型项目，代码和事务的配置是完全分离的， 声明式事务
          - web项目中使用spring
            - 使用监听器创建容器对象
            - ContextLoaderListener作用
              - 1.创建容器对象WebApplicationContext
              - 2.把容器对象放入到ServletContext作用域
              - 3.创建容器对象，把spring配置文件所有对象都创建好